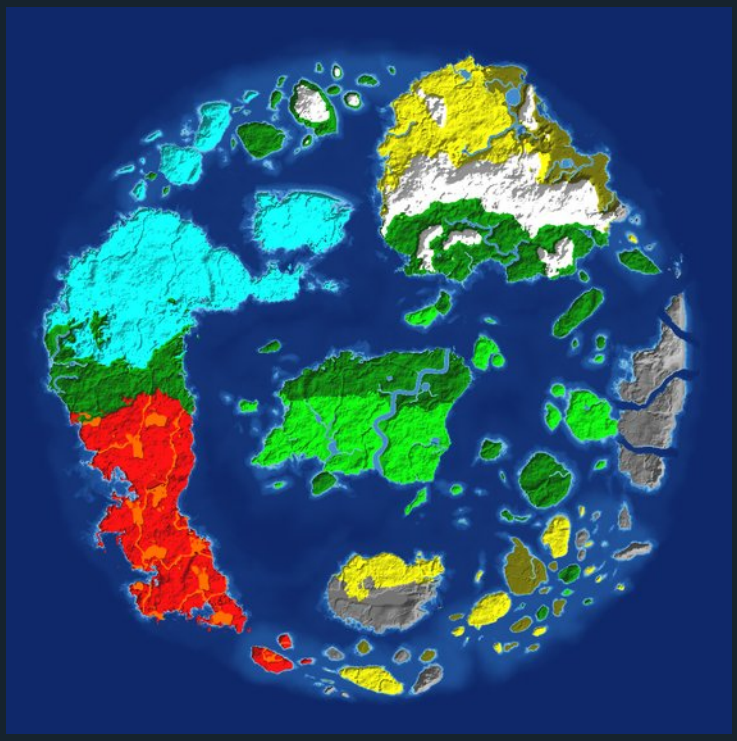


Better Continents Map-Making

A skill for AI agents

Author a Valheim world that reads as natural geography and gives every game location room, and prove both before a human plays it.

Written by Wubarrk, with Claude (Anthropic) as the drafting agent

Learned on the ProximaMaxi realm map: thirteen versions and four test worlds,
22 to 23 September 2026; brought up to date for 0.10 on 4, 5, 7, 9, 14 and 26 October 2026

For Valheim 1.0 and Better Continents 0.10.x (github.com/RGlabs84/BetterContinents10)

For any agent, Claude, Gemini or another, at any level: one that can only talk with the human, or one with a shell on the machine that holds the game, the mod and the toolkit

Licensed under the GNU General Public License, version 3, WITHOUT ANY WARRANTY. The licence text is the final appendix; the Markdown source, the build scripts and the toolkit are attached to this PDF (AI-skill-source.zip).

Contents

About this document . . . 8
How to read the samples . . . 9

License . . . 11

1 The skill . . . 12
What you are being asked to do . . . 12
How the work goes . . . 12
What you can do decides how you work . . . 13
The round, step by step, at each level . . . 14
Ask once, at the start . . . 15
The contract with the human . . . 15
The workspace . . . 16
Check it: ready to start . . . 17

2 How images become a world . . . 19
The square, the pixel and north . . . 19
The files . . . 21
The height maths . . . 23
Tall worlds: Amounts above 2 . . . 27
Fine heights . . . 28
The biome map . . . 31
The heat map . . . 33
The cold map and the location exclusion mask . . . 34
The lava map . . . 34
The forest map . . . 35
The location map . . . 36
The alt-biome map . . . 38
Baking, the config and the world . . . 39

3 What the game demands of a map . . . 41
The game's grid . . . 41
Regions and alt biomes . . . 42
Locations . . . 44
What the important locations want . . . 46
The game's fixed circles: the Deep North's look and sea, the Ashlands sea . . . 49
A tall world: what the game assumes about height . . . 49

4 What "reads natural" means 51
The tells, and the fixes 51
The tells at a glance 54
Relief that reads, in numbers 54
The reference world 55
Where the human's taste showed 55

5 The generator, as a pattern 56
Skeleton 56
The parts 56
A minimal generator you can copy 57
The Amount, in the generator 62
Changing the map 62

6 The workflow 64
At your level 64
Step 0: know the state 64
Step 1: agree the change 65
Step 2: snapshot before you touch anything 65
Step 3: change the generator 65
Step 4: generate at full size 65
Step 5: measure 65
Step 6: render, and show the human 66
Step 7: publish the version 67
Step 8: stage into the game profile 67
Step 9: the human builds a NEW world 68
Step 10: read the log 68
Step 11: replay the log 69
Step 12: hand off 70
The live loop: tuning a world that is already open 70

7 The traps that cost worlds 76
What each trap looks like, and the exact fix 82

8 A validated build, end to end 88
The inputs 89
What a good round looks like, and a bad one 89
The generator (gen.log) 90
check.py 90

relief.py . . . 90
altbiomes.py . . . 91
locations.py . . . 91
The game (log-13-ProxEra01.txt) . . . 92
The replay (locations.py --log) . . . 92

9 Export and import . . . 94
bc_export: what it writes . . . 94
A real export, file by file . . . 96
What export.cfg pins, and what else an export now carries . . . 98
The editor pack, bc_import edited, and Gaea . . . 99
The options (all per export; no config key holds them) . . . 99
What is in editor/ . . . 100
The value and the grid . . . 101
The way back . . . 101
The traps of the editor round trip, in the order they cost . . . 102
Using an export with BetterWorlds . . . 103
Cutting and pasting between an export and a generated map . . . 103
Presets and bc_import . . . 104
Biome precision . . . 106
Multiplayer: the settings download . . . 107
Traps . . . 108

10 What a new world does differently . . . 110
Which world do you have . . . 110
At a glance . . . 112
Mixing versions . . . 113
World Size: the whole world . . . 114
Worlds past 16 km: Wide Sectors . . . 115
A big world: start, memory and joining . . . 118
Heightmap Alpha: blend a map into the game's own terrain . . . 118
Expand World Data on a new world . . . 119
Memory: tiles, and Compact Maps . . . 120
The vegetation twin guard . . . 121
Water follows rebuilt ground . . . 121
The console tools . . . 122
The config . . . 122

11 Recipes . . . 124
Before any recipe: the places, the checks and the proof . . . 124

Recipe 1: a small island world from hand-drawn maps . . . 127
Recipe 2: a continent with planted alt biomes and pinned locations . . . 131
Recipe 3: a big world past 16 km, with Wide Sectors . . . 134
Recipe 4: a 16k world with tall terrain and fine heights . . . 137
Recipe 5: export, edit and re-import . . . 140
Recipe 6: a server and its players . . . 142

12 Baked placements . . . 144
What a baked piece is . . . 144
At your level . . . 145
Part 1: baking in the game . . . 145
The setup . . . 145
What a bake takes, and what stays . . . 146
Worked example: a test village, baked, undone and unbaked . . . 146
Undo, unbake and the way out . . . 148
On a server . . . 148
Worlds made without Better Continents' maps . . . 149
Part 2: what a town file does in a world . . . 149
How a file reaches a world . . . 149
What a load keeps and replaces . . . 149
Live pieces, protection, orphans . . . 150
Consumables and decor . . . 150
Ground, paint and clear masks . . . 150
Part 3: writing a town compiler . . . 151
The file in one table . . . 151
Palette entries . . . 151
Records . . . 152
Rules a compiler keeps . . . 152
The sample writer: bcp.py . . . 153
Worked example: the sample town, written and checked . . . 153
Testing a file . . . 154
Part 4: tuning and checking . . . 155
The settings . . . 155
Shadows, and shadow proxies . . . 155
What to read . . . 156
Recipe: a town file, from a script to a world . . . 156
Traps . . . 157

13 Water, lakes and clearing . . . 158
At your level . . . 158

Part 1: The sea . . . 158
What the game does, and what Better Continents does instead . . . 158
Which worlds get it . . . 159
Authoring for it . . . 160
Check it . . . 160
Part 2: Lakes (experimental) . . . 161
What Better Continents does with a lake . . . 161
The lake file . . . 161
Make lakes from your heights . . . 162
The console: bc_lake . . . 163
Tarn's own files, and the Tarn mod . . . 163
Part 3: Clearing (experimental) . . . 164
The mask . . . 164
Write one . . . 164
Part 4: bc genloc . . . 165
Traps . . . 165

Appendix A Tool reference . . . 166
What each tool prints . . . 167
If you cannot run the toolkit . . . 170
Which tools assume Amount 2, and which assume the 21,000 m square . . . 170
What the checkers reproduce, and what they cannot . . . 171
When the game changes . . . 172

Appendix B The config that goes with the map . . . 173
On any machine . . . 174
The Gale profile . . . 175
Biome precision and the Export group (0.9.0, not written by the staging script) . . . 175
Big maps, big worlds and tall worlds (0.10.3, not written by the staging script) . . . 176
Baked placements (0.10.4, not written by the staging script) . . . 176
Water and lakes (0.10.5, not written by the staging script) . . . 177
The world . . . 177
Config reference, generated from the mod . . . 177
Scope: who reads a setting, and when . . . 178
[00 BetterContinents.Debug] . . . 178
[01 BetterContinents.Global] . . . 179
[02 BetterContinents.Heightmap] . . . 180
[03 BetterContinents.Biomemap] . . . 181
[04 BetterContinents.Forest] . . . 182
[05 BetterContinents.StartPosition] . . . 182

[06 BetterContinents.Maps] . . . 182
[07 BetterContinents.Misc] . . . 183
[08 BetterContinents.AltBiomes] . . . 185
[09 BetterContinents.Export] . . . 187
[10 BetterContinents.BakedPlacements] . . . 187
[11 BetterContinents.Water] . . . 188
[12 BetterContinents.Lakes] . . . 189
The image maps . . . 190
How each legend reads . . . 191
The default biome legend . . . 191
The bc console . . . 192

Appendix C The game's own numbers . . . 195
Every enabled location, in the game's placement order . . . 195
Locations that ask for high ground, and what a heightmap needs . . . 199
The 32 alt biomes and their placement windows . . . 200
The biome legend . . . 201
ProximaMaxi's alt-biome legend (map version 13) . . . 201

Appendix D Glossary and sources . . . 202
Glossary . . . 202
Sources, and how to check a claim . . . 206
Where to look a fact up . . . 208
The map's versions, as a study . . . 209

Appendix E placements.bcp, format 1 . . . 210
The layout . . . 210
What the reader checks . . . 211
Better Continents' additions . . . 212
How Better Continents writes a layer . . . 213

Appendix F GNU General Public License, version 3 . . . 214

About this document

This is a training document for an AI agent, whatever its model and whatever it is allowed to do. An agent that can only write messages follows it by giving the human exact steps; one that can read files, run commands or have the game run does more of the work itself. Chapter 1 names those four levels (Talk, Files, Commands, Game), and every chapter says where the steps differ. It teaches one skill, exactly as it is practised in Wubarrk's Better Continents workspace:

> **Rule:** Author Better Continents image maps for Valheim so that the world reads as natural geography AND every location the game wants to place has room, and prove both before a human plays it.

Since Better Continents 0.10.4 it also teaches a second, smaller skill that grew out of the first (chapter 12): **bake what players built, or what a town compiler wrote, into a world's layer so that the world carries far fewer objects, and keep every piece recoverable.**

It is written for Valheim 1.0 and Better Continents 0.10.x, the fork maintained by Wubarrk at github.com/RGlabs84/BetterContinents10. Everything in it was learned on one real project, the ProximaMaxi realm map, between 22 and 23 September 2026: thirteen map versions, four test worlds, and every check written because a world had already been lost without it. It was brought up to date for Better Continents 0.10 on 4 and 5 October 2026: Heightmap Amount 1 is the mod's default now, a new world is laid out to its World Size, and the config reference is generated from the mod's own source. On 7 October 2026 it gained what 0.10.3 added: maps up to 16,384 px, Heightmap Amounts up to 81 with the rules of a high world, fine heights, worlds past 16 km with wide sectors, and exports at 16,384 px. The same day every chapter gained samples to copy, worked examples, pictures and checks, with notes for agents that cannot run commands, and a new chapter 11 of complete recipes.

New in 0.10.4 (October 2026): baked placements. Chapter 12 and appendix E were added on 9 October 2026: baking a base or a town in the game with `bc_bake` and the export window's Bake buildings tab, undoing and unbaking, town files (`placements.bcp`) and how to write and check one, the settings that draw them, shadow proxies, and the traps (chapter 7, traps 57 to 68). The toolkit gained `bcp.py`, a writer and checker of town files.

New in 0.10.5 (October 2026): water, lakes and clearing. Chapter 13 was added on 14 October 2026: Better Continents' own water, which draws the calm and the red sea where the world's maps put them and makes boats feel what is drawn; lakes at any height and a clearing mask, both **experimental** (they come from Raven Iron's Tarn), with their exact file formats, how to make and check them, and `bc_lake`; and `bc genloc`. Chapters 2, 3 and 11 and traps 54 and 55 say what the new water changes, traps 66 and 68 what 0.10.5 fixed in baking, and traps 69 to 81 are new. The toolkit gained `lakekit.py`, a lake-file and mask writer and checker.

New in 0.10.7 (October 2026): the export for BetterWorlds and the editor pack. Chapter 9 gained the cold map (`coldmap.png`, `nocold`), the exported location exclusion mask, `world.json`, the manifest's new keys, what `export.cfg` pins, the editor pack for Gaea and other terrain editors (`bc_export ... editor`, `bc_import edited`, with its traps and what has not been checked against a real Gaea), and a short section on BetterWorlds. Chapter 6 gained `bc regen` words (`dry`, `biome=`, `min=`, `max=`, `pos=`, `zone=`, `chance=`), the `pins` word of `bc genloc`, the Upgrade World notes and the Expand World Data and Expand World Size lines; traps 82 to 88 are new.

How to use it, if you are the agent:

1. Read chapters 1 to 3 in full before touching anything. Chapter 1 is the work and how to share it with the human at your level. Chapters 2 and 3 are the model of how the game and the mod turn images into a world, and the demands the game makes of a map. Nothing later makes sense without them. Chapter 2 derives the height maths for any Heightmap Amount, from 1 to a tall world's 81, and fine heights; settle the Amount before you author a height.

2. Chapter 4 is the design goal. Chapter 5 is how the generator is built, so that you can change it or write your own; it has a minimal generator to copy. Chapter 6 is the workflow, step by step, with the exact commands and the acceptance criterion for each step, and the live loop with the `bc` commands for tuning a world that is already open. Do the steps in order and never skip a check.
3. Chapter 7 is the list of traps. Each one cost a test world. Read it before your first build and again whenever a result looks wrong.
4. Chapter 8 walks through a real validated build, so you know what good numbers look like.
5. Chapter 9 is the other way to start a map: exporting a real world (vanilla or Better Continents) instead of, or alongside, `gen.py`, and turning a folder back into a New World with `bc_import`.
6. Chapter 10 is what a new world (settings version 12, which is what Better Continents 0.10 makes unless a preset that ships with the mod or one saved by 0.9, an export of a world resized under 0.9 or made under Expand World Size at another size, or `Override version` says 11) does differently from an older one (version 11, which is what 0.9 made): which world is which, mixing versions, world size, worlds past 16 km and their wide sectors (read that before you choose a size past 16,350 m: a wide world needs 0.10.3 or later for good), a big world's start, memory and joins, Heightmap Alpha, exact location pins, Expand World Data, memory and Compact Maps, the twin guard. Check which kind of world you have (`bc info`) before you trust a number in the other chapters.
7. Chapter 11, Recipes: complete setups you can copy, from a small island world to a 16k world, each with its config, legends, commands, expected log lines and pictures.
8. Chapter 12, Baked placements, is a skill of its own, for buildings rather than maps: read it when a human wants to bake a base or a town, asks how to remove the mod from a world with baked pieces, or wants a town file written. Its part 1 needs chapter 1 only; parts 2 and 3 (town files and a compiler) also need chapter 2's zones and heights; appendix E is the file's every byte.
9. Chapter 13, Water, lakes and clearing: read part 1 whenever a map has a Deep North or a heat map over the sea (it decides where the sea is calm and red), and parts 2 and 3 when a human wants lakes above the sea or cleared roads. Lakes and clearing are experimental: say so, and have the human try them on a copy of a world first.
10. The appendices are the reference: the tools, the config (a page written by hand and a page generated from the mod), the game's own placement numbers, a glossary, where every fact came from, and the town file's format.

How to read the samples

- **Code blocks are exact.** A config block is the lines to write, under the section header shown, spelled as the config spells them. A command is the whole command. A file is the whole file. A log line is the line as the game or the mod prints it; a placeholder in angle brackets (`<path>`, `<name>`) stands for your own value, and `N` for a number. A line too long for the page is broken at a space, and the rest of it starts the next line after a grey `»`: in the file or the log it is one line. A `\` at the end of a command's line is the shell's own: the command goes on, and it runs as written.
- **"Real"** next to an output means it was copied from a real run or a real log, and says which. Output without that word is the form a line takes, written from the source code that prints it.
- **Boxes.** A **Rule** (green) is a rule that holds. A **Trap** (red) cost a world. A **Human** box (amber) is a working rule of the human's. A **Check** box (blue) is how to prove the step before you move on.
- **"If you can ..."** notes say what changes at your level (chapter 1). Read the one for your level.
- **Do / Don't** tables put a good choice beside the bad one.
- **Pictures** are real: previews of real map versions, maps made by the samples, and diagrams computed from the rules in the text. Each caption says which, and what to notice.

The toolkit (the generator and every checker) is attached to this PDF as a zip; see the License page. On Wubarrk's machine it lives at `/home/rohan/WubarrkCODING/BetterContinents10/.work/BetterContinents-TestMaps/`, and the paths in this document are that machine's. On another machine, unpack the zip and substitute your paths; chapter 1 has the places Valheim keeps its saves on Windows and Linux, and the files a mod profile holds.

What this document is not: it is not the Better Continents guide. The guide (`BetterContinents-Guide.pdf`, the edition for Valheim 1.0 and Better Continents 0.10.x) documents every setting and map type of the mod, and a separate `BetterContinents-Export-Guide.pdf` walks a newcomer through exporting and importing, as `BetterContinents-Baking-Guide.pdf` (Better Continents 0.10.4 or later) walks a player through baking and `BetterContinents-Water-Guide.pdf` (0.10.5 or later) through the water, lakes and clearing. This document assumes the mod works as the guide says and teaches how to make a map that the game will accept and a human will enjoy.

License

Better Continents map-making: a skill for AI agents was written by Wubarrk, with Claude (Anthropic's model) as the drafting agent, and is released under the **GNU General Public License, version 3** (GPL-3.0), the licence of the Better Continents Guide it accompanies, so that the two documents can be shipped and modified together under one licence. Copyright 2026 Wubarrk.

It is distributed **WITHOUT ANY WARRANTY**; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. The full text of the licence is reproduced in the final appendix, GNU General Public License, version 3.

The tools attached to this PDF are Wubarrk's and carry the same licence. The two data files `locations_1.0.15.json` and `altbiomes_1.0.15_full.json` are tables of placement parameters read out of Valheim's game data by the attached extractors: facts about the game, reproduced for interoperability. Valheim is the property of Iron Gate Studio.

Better Continents is LGPL-2.1 (created by billw2012, maintained by Jere Kuusela from 2023 to 2026 and by Wubarrk since September 2026). The Better Continents Guide is GPL-3.0 (written by billw2012). Neither is reproduced here beyond short statements of behaviour. The generated page of appendix B states the mod's settings, their ranges and descriptions, which `make_b_config.py` reads from the mod's source: facts about the mod, set down for interoperability, with the scripts attached and none of the mod's code.

Attachment. The file `AI-skill-source.zip` is attached to this PDF; open your PDF viewer's attachments panel to save it. It contains:

- `docs/`: the Markdown source of this document and its images;
- `build_ai_skill_pdf.py` and `build_pdf.py`: the build scripts (the second is the guide's renderer, which the first reuses); `make_b_config.py`, which writes the generated page of appendix B from the mod's source, and `make_c_data.py`, which writes appendix C from the game data and the mod's source; and `png_extras.py`, the heightmap-record and grey-alpha helper (chapters 2, 6 and 10);
- `LICENSE`: the GPL-3.0 text;
- `toolkit/`: the generator, the checks, the renderers, the extractors and the data they extracted, the staging script, and the ProximaMaxi map's README and legends, laid out as they are in the workspace.

1 The skill

What you are being asked to do

A Better Continents world is made from a handful of images: a heightmap, a biome map, a heat map, a forest map, a lava map, a location map and, since Better Continents 0.8.1, an alt-biome map. The mod reads them once, when a world is created, and bakes them into the world. The game then places its several hundred location types (bosses, dungeons, villages, wrecks, altars) on that terrain by its own rules, and chooses which of its 32 alt biomes go on which regions, by its own rules.

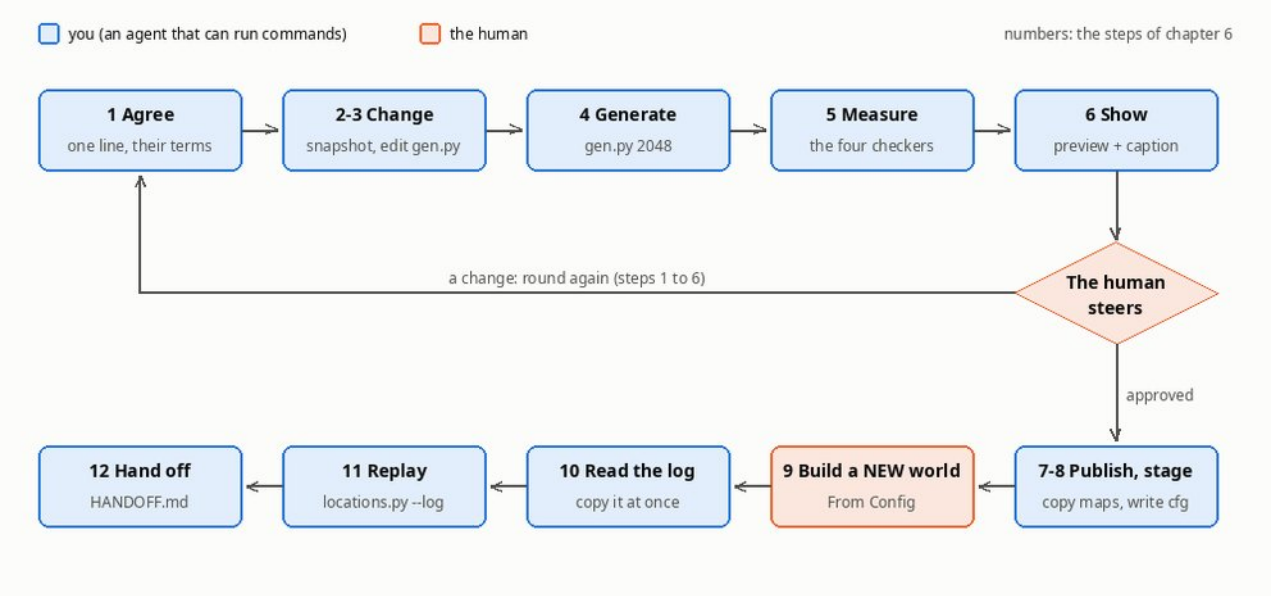
The skill has three halves that pull against each other:

1. **The world must read as natural geography.** Not blobs and stripes: fractal coasts, ragged biome fronts, a mountain crest that wanders, rivers that drain, rolling ground, cliffs where the wind hits. The human judges this from pictures. Chapter 4.
2. **Every location the game wants must have room.** Hildir's cavern needs mountain 200 m up; the frozen ships need Deep North water 5 to 15 m deep, 8 to 9.75 km from the centre; Bonemass wants flat swamp under 2 m above the water; the goblin towers want 100 plains sites 512 m apart. A beautiful map that breaks the game's progression is a failed map. Chapter 3.
3. **It must be proved before a human plays it.** A world takes a human minutes to build and hours to judge, and a map bug (the wrong waterline, a stray heat pixel, the wrong height scale) is invisible in a picture and total in the game. Every claim in this document comes with the check that proves it. Chapter 6.

Since Better Continents 0.10.4 a world can also carry **baked placements**: buildings that a player baked in the game, or towns that a program wrote into a `placements.bcp`, drawn and solid on every player's screen but no longer objects of the world. Helping a human bake, and writing and checking a town file, is the subject of chapter 12. It follows the same contract as the maps (a dry run before every change, a check before a human plays, the human's world read-only unless they ask), and its own measure: the objects a bake takes out of the world.

How the work goes

You do not draw the images by hand. You write and run a **generator** (a Python program that evaluates the world at every pixel and writes the PNGs), you **measure** its output with the checkers, you **render** a preview and **show it** to the human, the human **steers**, and you iterate. When the human is satisfied you **stage** the map into the game's mod profile, the human builds a **new world**, you read the game's **log** and **replay** it with the checker to confirm that the game agreed with your prediction. The full loop is chapter 6.



A diagram of one round, chapter 6's twelve steps as a loop, coloured by who acts when the agent can run commands. Notice that the human's two moments are the ones that decide: steering from the picture, and building the world.

The unit of work is a map **version**. Every version is a snapshot of the generator and its output in `generator/history/`, numbered, with the numbered previews the human saw in `previews/`. ProximaMaxi went through 13 versions in two days; the human looked at every one.

A faster loop exists for settings, once a world is open: the host changes a value or reloads a picture from the game's console (`bc`, a cheat command; Debug Mode switches cheats on) and Better Continents regenerates the world's zones (the last section of chapter 6). It tunes; it does not replace building a new world and replaying its log, which is how a map is proved.

What you can do decides how you work

This document is written for any agent, whatever model it is and whatever its environment lets it do. The round above is always the same; what changes is **who does each step**. Find your level before you start, and say it to the human in one line ("I can read and write your files and run Python, but I cannot see the game: you will build the world and I will read the log").

Level	You can	The human does	You show a picture by	You prove a map by
Talk	write messages, nothing else	every step: edits files, runs the tools, plays	describing what matters and naming the file to open	the log lines and checker output they paste back
Files	read and write the human's files	runs programs, launches the game	giving the file's path, or a page or inline image if your environment can show one	reading the log file and the checkers' output files yourself
Commands	run commands: Python, the toolkit, image tools	launches the game, builds the world, plays	rendering a preview and showing it	running the checkers, then replaying the game's log
Game	have Valheim with Better Continents run, by you or by the human for you, and read its log	plays and judges	a preview, and in-game pictures (`bc scr` saves the minimap)	building a new world from the maps and replaying its log

The first three add up: an agent that runs commands can also read files and talk. **Game** is about the world, not your tools: it is there when a world can be built and its log read, by you or by the human. Most

agents work at **Commands**, with the human at **Game**. Where a step differs by level, this document says so in a note that begins "If you can ...". Read the note for your level and skip the others.

How to tell your level, in three questions:

1. Can you open a file on the human's machine and see what is in it? If not, you are at **Talk**.
2. Can you run `python3 --version` and see what it prints? If not, you are at **Files**.
3. Can Valheim with Better Continents be started on a machine whose log you, or the human, can read? If yes, the round reaches **Game**; if not, you can make and check maps, but not prove them in the game.

> **Rule: Never pretend to a level you do not have.** An agent at Talk that writes "I ran the checker" has invented a number. Say what you could not check, and give the human the exact command or click that checks it.

If you can only talk: give the human everything as exact text to copy: the whole file with its name and folder, the whole command, the button to press. Never "adjust the config": write the section header and the line. Ask for the output back in a code block, and read it before the next step. Sample, for one step:

```
Please open this file in a text editor:
  <your profile folder>/BepInEx/config/BetterContinents.cfg
Find the section that starts with the line  [02 BetterContinents.Heightmap]
and change the line that starts with  Heightmap Amount  so that it reads exactly:
  Heightmap Amount = 2
Save the file. Then send me the lines under [02 BetterContinents.Heightmap]
as they are now.
```

If you can read and write files: write the files yourself, then tell the human exactly what to run and what to send back ("in the `generator` folder, run `python3 check.py ../fire-and-ice | tee check.log` and tell me when it has finished; I will read `check.log`").

If you can run commands: run the round yourself, capped (below), and stop at every point where the human must look or act: the picture, the staging, the world.

If the game can be run: read the log after every world the human builds, and replay it (chapter 6, steps 10 and 11).

The round, step by step, at each level

Step (chapter 6)	Talk	Files	Commands
1. Agree the change	you	you	you
2. Snapshot the generator	the human copies `gen.py`	you copy it	you copy it
3. Change the generator	you write the new code; the human pastes it	you edit it	you edit it
4. Generate	the human runs `gen.py`, sends the last lines	the human runs it; you read `gen.log`	you run it
5. Measure	the human runs the four checkers, sends the output	the human runs them; you read the logs	you run them
6. Render and show	the human renders and looks; you say what to look at	the human renders; you name the file	you render and show
7 and 8. Publish, stage	the human copies the files and edits the config	you copy and write the config	you run the staging script
9. Build a new world	the human	the human	the human
10 and 11. Read and replay the log	the human sends the log lines you name	you read the log	you read and replay it

Step (chapter 6)	Talk	Files	Commands
12. Hand off	you write the handoff text; the human saves it	you write `HANDOFF.md`	you write `HANDOFF.md`

Ask once, at the start

The human's setup decides every path and every message. Ask once, in one message, and keep the answers in the handoff file. A sample you can send as it is:

```
Before I start, five questions:
1. Which mod manager do you use for Valheim (Gale, r2modman, the Thunderstore app, or none),
   and what is the profile called?
2. Which Better Continents version is installed? (Your BepInEx log has a line
   "Loading [Better Continents 0.10.x]" near the top.)
3. Which folder should the maps go in? (Better Continents reads them from the
   config's Directory.)
4. Can you see pictures I send you here, or should I give you file paths to open?
5. Will you build the test worlds and send me the log, or can I read the log myself?
```

The contract with the human

These are Wubarrk's working rules. Each was stated after something went wrong, so none is optional.

> **Human: Show a picture after every meaningful change**, with a one-line caption of what changed and what is next. Do not batch several rounds of changes before showing anything. The human steers the design from the pictures ("i like it 06", "refine and work off of it"); a picture they have not seen is a decision they could not make. Deliver each picture the way your environment can and the human can see: an image in the conversation, a page they open, a link, or the file's path for them to open. If you cannot show images at all, describe what matters (what changed, where on the map, what to look at) and give the path. Ask once how your human wants them, and keep to it: what reaches one human may not reach another. Wubarrk's go to one page that is updated at the same address, because files attached to the conversation never reached them.

A caption that works, and one that does not:

Do	Don't
14-full.png: the spawn coast now has bays and headlands every 300 to 800 m; the rest is as in 13. Next: the river mouths, unless you want more coast.	Updated the map.
14d-spawn.png (a close-up of the spawn, 4x): the beach is 210 m wide; the hills start 125 m inland. Look at the north shore.	Here are some pictures.

> **Human: Keep their layout.** What goes where (which continent is spawn, where the Ashlands arm points, that the Mistlands are an eastern island chain) is the human's. Make the shapes natural inside that layout. Ask before moving a landmass.

> **Human: The design goal is a realistic-feeling world**, "official-feeling". Judge every change against that. Chapter 4 lists the tells that break it.

> **Human: Their saves are read-only.** Never delete, rename, copy over or "fix" a world in `worlds_local` unless they say so ("nuke Proxima" is the form it takes). Valheim's own save plumbing drops mod sidecars during rename and copy, so even a helpful copy can destroy a world. A setting can rewrite a world too: `Wide Sectors = On` converts every existing big world the profile then opens, one way (chapter 10). Never set it unless they ask for a world to be converted, after a backup of that world. A bake changes a world as well: never confirm a `bc_bake` change that the human has not

> asked for, and read its dry run with them first (chapter 12). And say it before you stage a map that binds its world: one made past 16,350 m (wide by default) or with fine heights needs Better Continents 0.10.3 or later from then on, and one read at a Heightmap Amount above 5 loses its high-world rules in an older version (chapters 2, 3 and 10).

> **Human: Mods live in Gale profiles, never in the Steam game folder.** Gale hardlinks plugin DLLs from a shared cache: copying over one writes through to every profile. Check the link count (`stat -c %h <file>`) before writing to any file in a profile; rename the old file out of the way instead of overwriting it. The staging script does this correctly; use it.

If the human uses another mod manager, or none: find the profile folder first (in a manager, its profile settings open it; without one, BepInEx lives in the game's own folder). Everything this document says about a profile is inside it: `BepInEx/plugins/` for the mod, `BepInEx/config/BetterContinents.cfg` for the config, `BepInEx/LogOutput.log` for the log. Whether another manager links its files as Gale does was not checked for this document: rename an old file out of the way rather than writing over it, whatever the manager.

> **Human: Run every heavy command capped and niced**: `(ulimit -v 6000000; nice -n 10 <command>)`. A preview render that was handed pixel coordinates instead of fractions tried to build an image hundreds of times the map's size, exhausted memory and crashed the human's editor. The cap is the safety net; if a command dies at the cap, fix the input, do not raise the cap.

`ulimit` and `nice` are commands of the Linux and macOS shells. **If you run commands on Windows**, there is no such cap: run one heavy command at a time, keep the toolkit at the size it was built and measured at (2048 px), and stop a run that keeps growing.

> **Human: Commit or push only when asked. Be terse**: a short status line, then act. Keep the handoff file (`HANDOFF.md`) current so the next session, human or agent, can pick up where you stopped.

A handoff entry that the next session can act on (a sample; the numbers are map version 13's):

```
## State (2026-09-23)
Map version 13 (gen-13-relief.py), staged into BC-Tester, built as ProxEra01.
Measured: land 27.7%; Bonemass 4/5; VERDICT placed nothing: Mistlands_Giant2 only.
The human said: "looks good on my end". Open: nothing asked.
Log copy: generator/history/log-13-ProxEra01.txt
```

> **Human: The human launches the game and builds the world.** You stage the files, tell them the exact sequence (chapter 6, step 9), and read the log afterwards.

The workspace

On Wubarrk's machine:

Path	What
`/home/rohan/WubarrkCODING/BetterContinents10/.work/BetterContinents-TestMaps/`	the map workspace (not a git repository)
`generator/gen.py`	the ProximaMaxi generator: `python3 gen.py 2048 <out dir>`
`generator/mapscale.py`	the ONE place the Heightmap Amount lives (this workspace's 2.0; the mod's own default is 1); every tool reads it
`generator/check.py`, `relief.py`, `altbiomes.py`, `locations.py`	the checks (appendix A)
`generator/preview.py`, `altmap_preview.py`	the renderers
`generator/history/`	snapshots: `gen-NN-name.py`, `out-NN-2048/` with their `*.log`, copies of game logs

Path	What
`previews/`	the numbered pictures the human saw
`fire-and-ice/`	the current map (map version 13 of ProximaMaxi) and its README
`locations/`	the game's location list (`locations_1.0.15.json`) and its extractor
`altbiomes/`	the game's alt-biome list (`altbiomes_1.0.15_full.json`), its extractor, research notes
`stage-into-gale.sh`	writes the mod DLL and the config into a Gale profile
`ai-skill/`	this document: `docs/`, `build_ai_skill_pdf.py`, `make_b_config.py` and `make_c_data.py` (they write appendices B and C), `png_extras.py`
`HANDOFF.md`	the current state, kept for the next session
`/home/rohan/WubarrkCODING/BetterContinents10/`	the Better Continents fork (git; `ALTBIOMES.md`, `MIGRATE-1.0.md`, `SettingsSchema.cs`, which appendix B is generated from)
`/home/rohan/WubarrkCODING/BetterContinents10/.work/BetterContinents-Guide/`	the guide's Markdown and `build_pdf.py`
`/home/rohan/WubarrkCODING/libs-Tools/`	the Valheim 1.0 reference: decompiled game code, `*-FACTS.md` notes
`~/.local/share/com.kesomannen.gale/valheim/profiles/BC-Tester/`	the Gale test profile: `BepInEx/config/BetterContinents.cfg`, `BepInEx/LogOutput.log`
`~/.config/unity3d/IronGate/Valheim/worlds_local/`	the worlds; each is a folder with a `BetterContinents` settings file inside
`~/upy/bin/python`	a Python with Pillow (for `altmap_preview.py`); everything else is the standard library

On another machine, unpack the toolkit attached to this PDF (it keeps the same folders: `generator/`, `locations/`, `altbiomes/`) and put your paths in place of these. Valheim's own save folder, which holds `worlds_local/` and Better Continents' own `BetterContinents/` folder (exports, presets), is in the same place on every machine of a kind (as the export guide lists them):

```
Windows:
  C:\Users\<you>\AppData\LocalLow\IronGate\Valheim
Linux, the native game:
  ~/.config/unity3d/IronGate/Valheim
Linux, the Windows game under Proton:
  ~/.local/share/Steam/steamapps/compatdata/892970/pfx/drive_c/users/steamuser/AppData/LocalLow/IronGate/Valheim
```

Read the workspace's own documents before the decompiled game code: `HANDOFF.md`, `fire-and-ice/README.md`, the fork's `ALTBIOMES.md`, `altbiomes/RESEARCH-*.md` and the `libs-Tools/*-FACTS.md` files. They cite the game code by file and line, so a claim can be checked in a minute instead of re-derived in an hour.

Check it: ready to start

Three facts decide every number in the chapters that follow. Check them before the first change, at your level (the second command runs in `generator/`).

```
grep -m1 "Loading \[Better Continents" <profile>/BepInEx/LogOutput.log
python3 -c "import mapscale; print(mapscale.AMOUNT, mapscale.SEA, mapscale.CEILING)"
python3 --version
```

What good looks like (the first line as a real log prints it, with x the patch version installed):

```
[Info   :   BepInEx] Loading [Better Continents 0.10.x]
2.0 0.15 340.0
```

> **Check:** The mod is 0.10.x (the alt-biome map needs 0.8.1 or later; an Amount above 5, fine heights or a world past 16,350 m with wide sectors needs 0.10.3 or later). `mapscale.py` holds the Amount you mean to author for: 2.0 prints `2.0 0.15 340.0` (waterline pixel 0.15, ceiling 340 m above the water); 1.0 prints `1.0 0.3 140.0`. Python 3 runs. **If you can only talk**, ask the human to search the log for `Loading [Better Continents` and to tell you the Heightmap Amount line of their config, and do the Amount's arithmetic yourself (chapter 2).

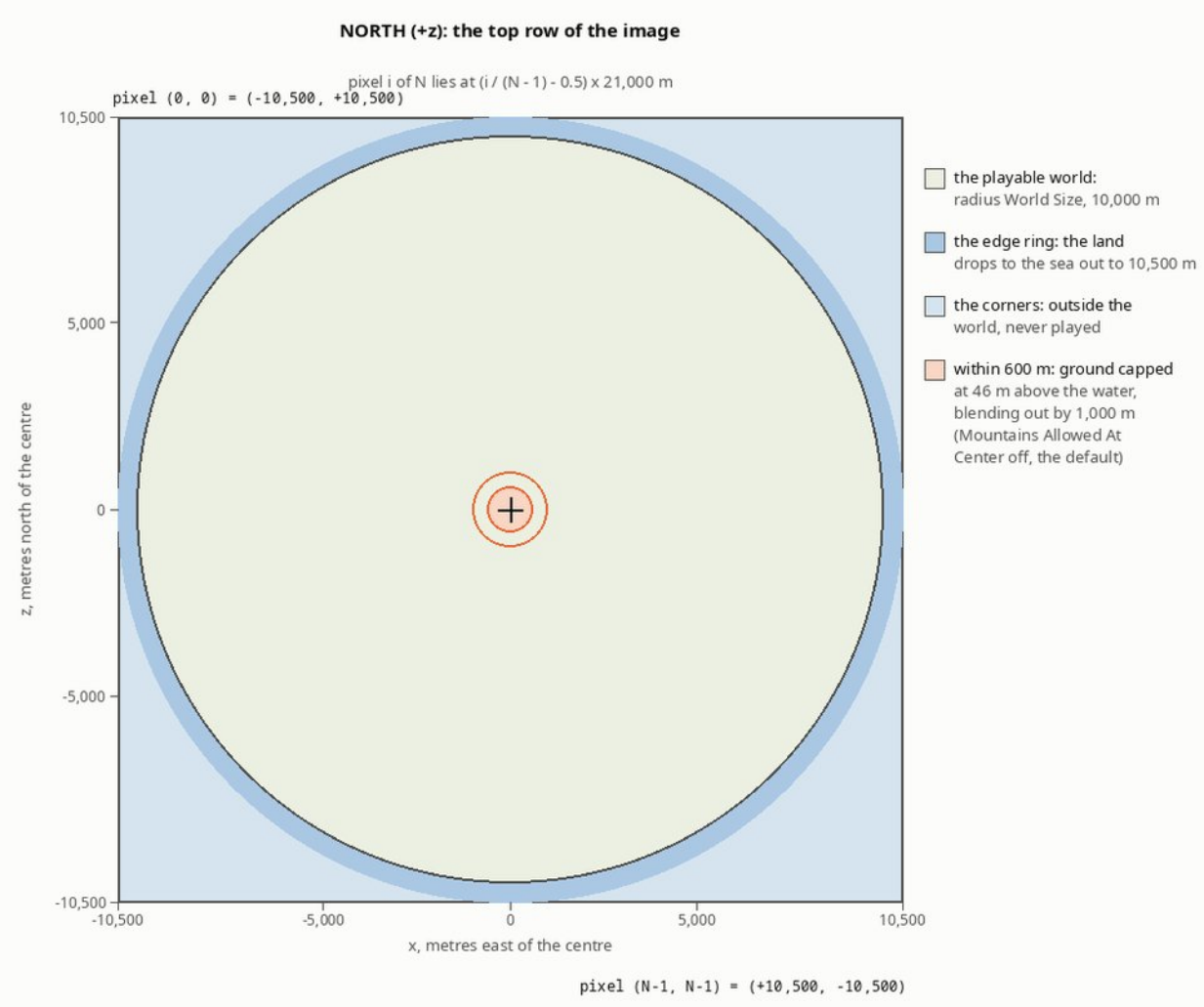
2 How images become a world

This chapter is the model you must hold in your head. Every number in it is derived from the mod's and the game's code, not guessed, and the derivations are given so that you can check them when versions change. Where a version 12 world (what Better Continents 0.10 makes for a new world) behaves differently from a version 11 one (what 0.9 made), the text says "a new world" and "an older world", and chapter 10 gathers the differences and says exactly which world is which.

Each part below gives the rule, a sample you can copy (a file, a config line, a command, a log line), a short worked example, and a **Check** box that says how to prove it. The Python samples use only the standard library; any Python 3 runs them. **If you can only talk**, do the arithmetic yourself and give the human the numbers; the samples show the formulas in their plainest form.

The square, the pixel and north

Every Better Continents map covers one square, centred on the world centre (0, 0). The top of the image is north (+z), the left edge is west (-x). Better Continents flips the image vertically on load so that this holds; author what you see.



A diagram of the map square at the default sizes, drawn to scale: the playable world (radius 10,000 m), the edge ring to 10,500 m, the corners outside the world, and the centre, where the ground is capped unless Mountains Allowed At Center is on. Notice that the top row of the image is north and that the first and last pixels sit on the square's edges.

The square's side is **2 x (World Size + Edge Size)**: the world's radius and the edge ring that the land drops away over, doubled. At the default sizes, 10,000 m and 500 m, that is **21,000 m**, from -10,500 m to +10,500 m on each axis. Two things decide whether a world's maps follow its sizes:

- **A new world** (settings version 12, which is what Better Continents 0.10 gives a world it makes From Config or from a folder of your own maps) spans exactly that: change World Size or Edge Size and the square changes with them, and so does the rest of the world's layout (chapter 10).
- **An older world** (version 11 and earlier) always spans 21,000 m, whatever World Size says: World Size only moves its edge. A world under Expand World Size spans the size that mod sets.

`bc info` shows which you have: its first line is `Version 12` for a new world and a lower number (11 for a world made by 0.9) for an older one, and a new world that is not at the default size adds `The maps span World Size and Edge Size: N m across`. Not every world made in 0.10 is a new world (for example, a world made from a preset that ships with the mod is version 11, chapter 10), so read the line. Everything in this document that is quoted in metres from the centre assumes the default sizes and the 21,000 m square, which is also what the workspace's generator and checkers assume.

A pixel is not a cell: pixel `i` of `N` lies at `(i / (N - 1) - 0.5) x side`, so the spacing is `side / (N - 1)`, and the first and last pixels sit on the square's edges.

Image size	Metres per pixel (21,000 m square)
2048 x 2048	10.26
4096 x 4096	5.13
8192 x 8192	2.56
10,501 x 10,501	2.00 exactly
16,384 x 16,384 (the largest)	1.28

ProximaMaxi is 2048. The game's own biome grid (chapter 3) is 12 m, so 2048 is already finer than anything the game samples for biomes and locations; bigger images only buy terrain detail, at a cost in generation time that grows with the square of the size. A bigger square spreads the same pixels wider: 2048 px over a 65,528 m square (World Size 32,264 + Edge Size 500) are 32 m apart, coarser than the game's grid, and the 16,383 px maps of a real world of that size are 4.00 m apart. Memory used to be the other cost (a 10,501 px map set held 1.32 GB); maps are now held in tiles and the same set takes about 99 MB (chapter 10).

Sample: pixels and metres, both ways.

```
SIDE = 21000.0                          # 2 x (World Size + Edge Size): the default sizes

def pixel_to_world(i, j, n):            # i from the west edge, j down from the north edge
    return (i / (n - 1) - 0.5) * SIDE, (0.5 - j / (n - 1)) * SIDE    # (x east, z north)

def world_to_pixel(x, z, n):
    return round((x / SIDE + 0.5) * (n - 1)), round((0.5 - z / SIDE) * (n - 1))

print(world_to_pixel(-108, -2201, 2048))   # (1013, 1238)
print(pixel_to_world(1013, 1238, 2048))    # (-107.7..., -2200.5...)
```

Worked example. ProximaMaxi's start temple is meant to stand at x -108 m, z -2,201 m (108 m west and 2.2 km south of the centre). On its 2048 px maps that is pixel (1013, 1238): 1013 pixels from the left edge, 1238 down from the top. Back again, that pixel lies at (-107.7, -2,200.5), within half a pixel (5 m) of the aim.

> **Check:** `check.py` prints each pinned site's distance from the centre (`StartTemple ... 2203 m from centre` on map version 13), and the square root of $108^2 + 2{,}201^2$ is 2,203.6. A distance that is off by thousands of metres means a flipped axis (north at the bottom) or a square of the wrong side.

The playable world ends at a radius of World Size (10,000 m by default) and the land drops away to the sea over the edge ring, to World Size + Edge Size. Land within about 700 m of the playable edge is possible but the edge water bites at it; ProximaMaxi's rim chains stop at 0.90 to 0.96 of the half-width.

> **Rule:** World Size + Edge Size past **16,350 m** (a square wider than 32,700 m) makes a **big world**, where the game's own filing of objects goes wrong, in up to three ways that begin at about 16.4 km, 22.1 km and 32.7 km from the centre (chapter 10). A NEW world that big is made with **wide sectors** by default (`[07 BetterContinents.Misc] Wide Sectors = Auto`), and a wide world needs Better Continents 0.10.3 or later from then on. Read chapter 10's "Worlds past 16 km" before you choose such a size. World Size + Edge Size can go up to 65,000 m; at 2 m a pixel 16,384 px cover 32.7 km (World Size 15,850 + Edge Size 500), at 4 m a pixel 65.5 km.

The files

The mod loads every correctly named file from the config's `Directory` (the "project directory"):

File	Format	What it does
`heightmap.png`	16-bit greyscale	terrain height; the whole vertical range of the world
`biomemap.png` + `biomemap.txt`	8-bit RGB + legend	the biome of every pixel; colours from the legend
`heatmap.png`	16-bit greyscale	Ashlands heat; ALSO decides what is Ashlands (heat > 0)
`forestmap.png`	16-bit greyscale	tree density; higher is more forest (with the config in appendix B)
`lavamap.png`	16-bit greyscale	lava in the Ashlands; the game's Ashlands vegetation mask
`locationmap.png` + `locationmap.txt`	8-bit RGB + legend	pinned sites for the start temple, bosses and vendors
`altbiomemap.png` + `altbiomemap.txt`	8-bit RGB + legend	planted alt biomes
`coldmap.png`	16-bit greyscale	the Deep North: weather, snow and calm sea wherever it is above black (`Coldmap Scale`, default 10, multiplies it)
`no_spawns_mask.png` (or `nospawns.png`)	grey picture	the location exclusion mask: no default random location (crypts, dungeons, camps) is placed wherever it is above black
`heightmap-fine.png` (optional)	8-bit greyscale	refines every height of a tall world to 1/256 of a 16-bit step (fine heights, below)

A file that is not there is not used, and the world goes on without that map: the Directory is a folder of whatever maps you want, no more. Every map is a square. The generator writes every map of a set at one size, which keeps them simple to check; the fine picture must be the heightmap's own size.

Sample: a map folder and the config that reads it. ProximaMaxi's, as it stands:

```
fire-and-ice/
  heightmap.png      biomemap.png      biomemap.txt
  heatmap.png        forestmap.png     lavamap.png
  locationmap.png    locationmap.txt   altbiomemap.png    altbiomemap.txt
  README.md          (not a map: ignored)
```

```
[00 BetterContinents.Debug]
Directory = /home/rohan/WubarrkCODING/BetterContinents10/.work/BetterContinents-TestMaps/fire-and-ice

[07 BetterContinents.Misc]
SelectedPreset = From Config
```

On Windows the line reads like `Directory = C:\Users\<you>\Maps\MyWorld` (the guide's form). Appendix B's page on the config (`b-config.md`) has the whole file that goes with ProximaMaxi.

Use 16-bit PNG for the greyscale maps: one 8-bit step of the heightmap is 0.78 m at Heightmap Amount 1 and 1.57 m at Amount 2, enough to terrace every beach, where one 16-bit step is 3 mm or 6 mm. The generator's `png.py` writes 16-bit greyscale and 8-bit RGB PNGs with nothing but the standard library.

A map can be up to **16,384 x 16,384** pixels. Save every map as a **plain PNG**: not interlaced, in 8- or 16-bit grey, grey with alpha, 8-bit RGB or RGBA, or a palette. Better Continents reads such a picture a few rows at a time, with little memory; any other picture (interlaced, 16-bit colour, another format) is read whole, about 1 GB for a 16,384 px colour picture while it is read, and for a picture of 4,096 px or more the log says so (`<file> is a picture of a kind that is read whole ...`). `png.py` writes plain PNGs. A bigger picture is refused before it is read, and the world is made without that map. The line, from a real log (its path replaced by `<file>`):

```
[Error] [BetterContinents] Cannot use texture <file>: it is 16385 x 16385 pixels, and the
» largest map Better Continents reads is 16384 x 16384. Scale it down (at 2 m a pixel, 16384
» pixels hold a world 32.7 km across).
```

`[07 BetterContinents.Misc] Max Map Size` (4096, 8192 or 16384, the default) sets a lower limit on one machine, for one short of memory, and then the refusal names the setting. A map that a world already stores is always read. Making a world from 16,384 px maps took 17 to 39 s and 0.5 to 1.5 GB above the idle server at its highest (measured on one machine), and a new world with any map more than 8,192 px across is made compact unless `Compact Maps` is Off (chapter 10). While the maps are read, the game shows its "Please wait"; before Better Continents 0.10.3 the game window froze instead.

Do	Don't
16-bit grey for the height, heat, forest and lava maps	8-bit grey heights: terraces of 0.78 m (Amount 1) on every beach
8-bit RGB for the biome, location and alt-biome maps, exact legend colours	colour maps painted with a soft brush, smoothed or anti-aliased
plain PNGs: not interlaced	interlaced PNGs, 16-bit colour: read whole, about 1 GB at 16,384 px
every map a square, the fine picture the heightmap's size	a map that is not square (`Cannot use texture ...: its width (W) does not match its height (H)`)

Sample: what a PNG is, as Better Continents sees it. A standard-library script (save it as `pnginfo.py`); it prints the size, the kind, whether the file is plain, and any text record:

```
import struct, sys

KIND = {(0, 8): '8-bit grey', (0, 16): '16-bit grey', (4, 8): '8-bit grey + alpha',
        (4, 16): '16-bit grey + alpha', (2, 8): '8-bit RGB', (6, 8): '8-bit RGBA',
        (2, 16): '16-bit RGB', (6, 16): '16-bit RGBA'}
PLAIN = ((0, 8), (0, 16), (4, 8), (4, 16), (2, 8), (6, 8))   # and any palette
for path in sys.argv[1:]:
    with open(path, 'rb') as f:
        data = f.read()
    if data[:8] != b'\x89PNG\r\n\x1a\n':
        print(f'{path}: not a PNG')
        continue
    at, text, other = 8, [], []
    while at < len(data):
        size, kind = struct.unpack('>I4s', data[at:at + 8])
        body = data[at + 8:at + 8 + size]
        if kind == b'IHDR':
            w, h, depth, ctype, _, _, interlace = struct.unpack('>IIBBBBB', body)
        elif kind == b'tEXt':
            key, _, value = body.partition(b'\0')
            text.append(key.decode('latin-1') + ': ' + value.decode('latin-1'))
        elif kind in (b'zTXt', b'iTXt') or (kind == b'tRNS' and ctype != 3):
            other.append(kind.decode())
        at += 12 + size
```

```
    name = KIND.get((ctype, depth), f'type {ctype}, {depth}-bit')
    name = 'palette' if ctype == 3 else name
    plain = interlace == 0 and not other and (ctype == 3 or (ctype, depth) in PLAIN)
    print(f'{path}: {w} x {h}, {name}, ' + ('interlaced' if interlace else 'not interlaced')
          + ''.join(', ' + o for o in other) + (': plain' if plain else ': READ WHOLE')
          + ('' if w == h else ', NOT SQUARE'))
    for t in text:
        print('    ' + t)
```

Its output on ProximaMaxi's folder (real):

```
heightmap.png: 2048 x 2048, 16-bit grey, not interlaced: plain
biomemap.png: 2048 x 2048, 8-bit RGB, not interlaced: plain
heatmap.png: 2048 x 2048, 16-bit grey, not interlaced: plain
locationmap.png: 2048 x 2048, 8-bit RGB, not interlaced: plain
```

> **Check:** run it on every map before you stage. Every line ends `: plain`, every size is the one you meant, and the heightmap is `16-bit grey` (or `16-bit grey + alpha` for Heightmap Alpha, chapter 10). In the game, each map read logs `Time to load <path>: N ms`. **If you can only talk**, ask the human for the image's properties in their editor (size, bit depth, colour mode) and for the `Time to load` lines of the log.

Other map types exist (terrain, paint, moss, spawn, vegetation, roughness); ProximaMaxi does not use them and this document does not teach them. The guide does, and appendix B lists all fourteen kinds with their file names, config keys and console commands, generated from the mod.

The height maths

This is the single most important derivation in the document. Getting it wrong drowned a whole world once (every lowland biome 30 m under water) and mis-scaled another (25 m under water everywhere).

From the mod's code (`WorldGeneratorPatch.GetBaseHeightV3`, `GetBiomeHeightWithHeight`, and the export's `WorldExportMath`) and the game's (`ZoneSystem.m_waterLevel = 30f`, `WorldGenerator.GetHeightMultiplier() = 200`), with `A` the Heightmap Amount, `Sea Level` the config key `Sea Level Adjustment` (0 to 1, default 0.5) and `SLA` the offset it makes:

```
v               = grey / 65535              the 16-bit pixel as 0..1
finalHeight     = v * A - 0.15 + SLA        (Blend 1, Add 0, Mask 0, no noise)
SLA             = Lerp(1, -1, Sea Level)    0 at the default Sea Level of 0.5
height (m)      = finalHeight * 200         the game's y; no biome curve
water surface   = 30 m                      finalHeight 0.15
altitude (m)    = height - 30               metres above the water
```

The parentheses are the config's defaults for Heightmap Blend, Add and Mask and no height noise layers, which the formula assumes; the location rules read the altitude.

Two words are used for two different things. **Height** is the game's y, with the water surface at 30 m; the export's `README.txt` and `export.cfg` write the formula as "metres" of height. **Altitude** is height above the water, and is what the game's location rules and this document's tables speak of. A heightmap that spans heights of -30 m to 170 m spans altitudes of -60 m to +140 m.

At the default Sea Level the maths simplifies, and these five lines are all you need:

```
altitude (m)       = (v * A - 0.30) * 200
waterline pixel    = 0.30 / A
ceiling            = (A - 0.30) * 200 m above the water   (pixel 1.0)
floor              = -60 m, 60 m below the water          (pixel 0.0, at any A)
one 16-bit step    = 200 * A / 65535 m
```

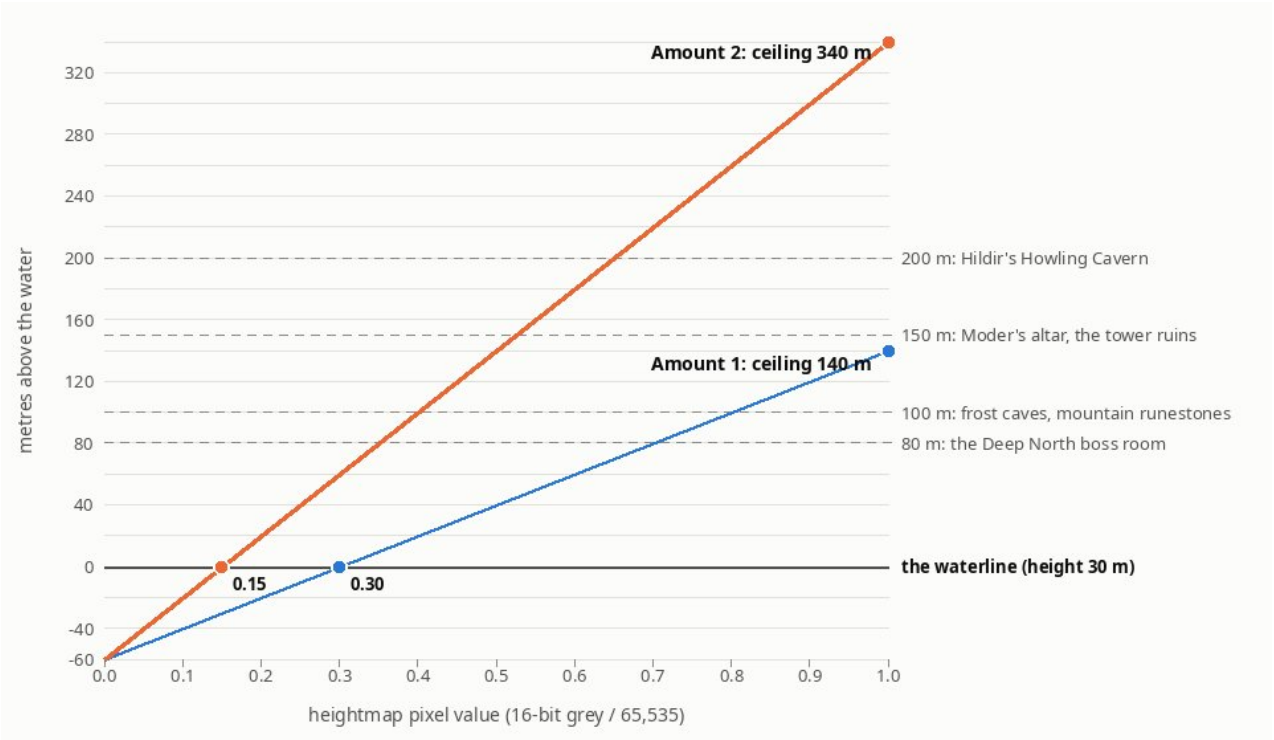
A different Sea Level shifts the whole range: the waterline pixel becomes `(0.30 - SLA) / A`, and each 0.1 below 0.5 lifts all the land 40 m out of the water (`SLA = 1 - 2 x Sea Level`). Everything below assumes the default.

Heightmap Amount 1 is Better Continents' default everywhere (the config, the console, the export's default and the import's fallback). Compare the amounts a map is likely to use:

Amount A	Waterline pixel	Ceiling above water	Heights spanned
1 (the default)	**0.300**	**140 m**	-30 m to 170 m
1.6	0.1875	260 m	-30 m to 290 m
2 (ProximaMaxi)	**0.150**	**340 m**	-30 m to 370 m

And what a pixel means at each (the last rows are out of reach at Amount 1):

Altitude (m above the water)	Pixel at Amount 1	Pixel at Amount 2	What it is
-60	0.000	0.000	deepest seabed (pixel 0 at any Amount)
-16	0.220	0.110	Deep North water that carries its biome (the frozen ships)
-1.4	0.293	0.1465	wadeable shallows
0	**0.300**	**0.150**	**waterline**
+26	0.430	0.215	finalHeight 0.28, the rule that pulls ground down near the centre
+80	0.700	0.350	the Deep North boss room wants level ground this high
+100	0.800	0.400	eleven mountain location types: the frost caves, drake nests, mountain runestones...
+140	**1.000**	0.500	**the Amount 1 ceiling**
+150	out of reach	0.525	Moder's altar, the mountain tower ruins
+200	out of reach	0.650	Hildir's Howling Cavern
+254	out of reach	0.785	the highest peak on ProximaMaxi
+340	out of reach	**1.000**	**the Amount 2 ceiling**



The same maths as a chart, computed from the formula above: metres above the water against the heightmap pixel at Amount 1 (blue) and Amount 2 (orange), with the heights the game's mountain and Deep North locations ask for. Notice that the blue line stops at 140 m, under Moder's altar and Hildir's cavern, and that both lines start at -60 m.

The game's location rules speak of "altitude": a location with `minAltitude 200` wants pixel 0.65 at Amount 2, and no ground at all at Amount 1. Appendix C lists every location that asks for 30 m or more, with the pixel it needs at each Amount.

> **Rule: Choose the Amount before you author a single height, and keep it in one place.** A map is only right at the Amount it was authored for. At **Amount 1** the highest ground is 140 m above the water, and four location types ask for more: Hildir's Howling Cavern (200 m), Moder's altar and the two mountain tower ruins (150 m); the rest of the mountain sites (100 m) crowd into the top fifth of the range, pixels 0.80 to 1.00. At **Amount 2** the ceiling is 340 m, every vanilla mountain location fits and the range can carry them all. On a 1.0 map the cavern placed 0 of 3 in a real build, and Hildir's mountain quest could not be done. A site pinned on the location map is placed wherever you pin it (see the location map, below), so a pin can put Moder's altar on an Amount 1 summit; the game's random pass cannot.

When to use which. Use **Amount 1** when the map's ground stays within 140 m of the water (a lowland or island world, low relief), or when the map must read right with the mod's default config and no `Heightmap Amount` line. Use **Amount 2** when the map has the mountains the vanilla progression expects (the cavern, the altar, the tower ruins): the 16-bit range then spans 400 m instead of 200, so a grey step is 6 mm instead of 3 mm, still far finer than anything visible. Go higher only for ground more than 340 m above the water ("Tall worlds", below). Whatever the Amount, state it in the config, because the default is 1.

Sample: grey values and metres.

```
A = 2.0                                  # the Heightmap Amount the config will say

def grey(metres):                        # metres above the water -> 16-bit grey
    return round((0.30 / A + metres / (200 * A)) * 65535)
```

```
def metres(g):                              # 16-bit grey -> metres above the water
    return (g / 65535 * A - 0.30) * 200

print(grey(250), round(metres(50790), 2), grey(0), grey(-60))    # 50790 250.0 9830 0
```

Worked example 1: a 250 m peak. The peak needs `A = 0.30 + 250 / 200 = 1.55` at least. Take 2 (headroom, and every vanilla mountain location fits). Its pixel is `0.15 + 250 / 400 = 0.775`, grey 50,790; the waterline is grey 9,830; the seabed at -60 m is grey 0. In the config:

```
[02 BetterContinents.Heightmap]
Heightmap Amount = 2
```

Worked example 2: what a pixel means. A pixel of grey 30,000 is `v = 0.4578`: 31.6 m above the water at Amount 1, 123.1 m at Amount 2. The same file, read at the wrong Amount, puts every hill 90 m higher or lower; that is how a world came out 25 m under water everywhere.

How to work at the Amount you chose:

1. **Fix it first.** Every constant of a generator that is written in metres goes through one function of the Amount. In the workspace that function is `H(metres) = SEA + metres / (200 x A)` in `gen.py`, with `SEA = 0.30 / A` from `mapscale.py`; at Amount 1 it is `0.30 + metres / 200`.
2. **Keep it in one file.** `generator/mapscale.py` holds `AMOUNT = 2.0`, the workspace's choice, and the generator, `check.py`, `preview.py`, `altbiomes.py`, `locations.py`, `relief.py` and `stage-into-gale.sh` all read it from there. Never hardcode it anywhere else. `mapscale.metres(v)` and `mapscale.value(m)` convert both ways. To work at Amount 1 in this workspace, change `AMOUNT` in a copy of the generator folder, never in the original that ProximaMaxi's staged config depends on, and read appendix A for what else assumes 2.
3. **Say it in the config.** `[02 BetterContinents.Heightmap] Heightmap Amount = 2.0`. The staging script writes it from `mapscale.py`, so ProximaMaxi's config states 2 although the mod's default is 1. A folder that holds an `export.cfg` states its own Amount there (chapter 9), and it wins over `BetterContinents.cfg`.
4. **Let the map carry it.** A world export writes the Amount and Sea Level it encoded for into the heightmap's PNG (a text chunk) and a new world that reads the file at other values warns in the log; `bc info` shows the record. `png_extras.py` (appendix A) puts the same record on a generated heightmap.
5. **Prove the scale.** `check.py` counts what is dry per biome at the Amount `mapscale.py` holds, and a drowned world shows at once (chapter 6, step 5).

Sample: the record, written and read back (real output; `pnginfo.py` is the sample above):

```
$ python3 ../ai-skill/png_extras.py record heightmap.png --set 2
heightmap.png: recorded Heightmap Amount = 2; Sea Level Adjustment = 0.5
$ python3 pnginfo.py heightmap.png
heightmap.png: 512 x 512, 16-bit grey, not interlaced: plain
    BetterContinents: Heightmap Amount = 2; Sea Level Adjustment = 0.5
```

A new world that reads a recorded heightmap at other values logs, as the mod's source writes it:

```
[Warning] <path>/heightmap.png was made by a world export for Heightmap Amount 2 and Sea Level
» Adjustment 0.5, but this new world reads it at Heightmap Amount 1 and Sea Level Adjustment
» 0.5, so its land is not at the heights it was made for. The export's export.cfg has the
» settings it needs.
```

> **Check:** the settings dump the log prints when the world is created, and `bc info` in the world, have the line `Heightmap amount 2, blend 1, add 0, mask 0` (real, from a ProximaMaxi world): the first number is the Amount the world was made with. With a recorded heightmap there is also

> `Heightmap made by a world export for Heightmap Amount 2, Sea Level Adjustment 0.5`. **If you can only talk**, ask the human to type `bc info` in the console (F5, with cheats on) and send you the lines that begin `Version` and `Heightmap amount`.

> **Trap:** The Amount is copied into a world from the config **as it stands when the world is created**. Before 0.9.0 that meant as it stood when the game was launched; since 0.9.0 Better Continents re-reads the config file whenever it changes on disk, so a world created after staging reads the new values. A map drawn for 1.6 and read at 1.0 came out 25 m under water everywhere. The staging script still refuses to run while Valheim runs, and the safe order does not change: stage, quit Valheim, relaunch, then create. Whatever the order, check `Heightmap amount` in the world's settings dump (`bc info`, or the log at creation).

Two more things the mod does to heights with a heightmap loaded:

- Vanilla's rivers are bypassed (`Heightmap Override All = true`, the default), so **rivers and lakes must be carved into the heightmap** or the world has none.
- Unless `Mountains Allowed At Center` is on, the mod caps ground within about 600 m of the world centre at 46 m above the water, blending back to the heightmap by 1,000 m. Keep high ground away from the centre, or turn the setting on. (The rule is the game's own, which the mod keeps: within 600 m of the centre, ground above finalHeight 0.38, 46 m above the water, is cut to 46 m; ground below that is left as it is.)

Tall worlds: Amounts above 2

Since Better Continents 0.10.3 `Heightmap Amount` takes 0 to 81 (it stopped at 5 before), and the export's `Default Heightmap Amount` and `bc_export amount=` take 0.01 to 81 (chapter 9). The five lines above hold at every Amount: the floor stays 60 m below the water, and the waterline, the ceiling and the step move with A.

Amount A	Waterline pixel	Ceiling above water	Heights spanned	One 16-bit step
1	0.300	140 m	-30 m to 170 m	3 mm
2	0.150	340 m	-30 m to 370 m	6 mm
5	0.060	940 m	-30 m to 970 m	15 mm
10	0.030	1,940 m	-30 m to 1,970 m	31 mm
20	0.015	3,940 m	-30 m to 3,970 m	61 mm
40	0.0075	7,940 m	-30 m to 7,970 m	0.12 m
81 (the most)	0.0037	16,140 m	-30 m to 16,170 m	0.25 m

> **Rule: Use the smallest Amount that holds the highest peak.** Pixel 1.0 is the ceiling, so a peak `h` metres above the water needs `A = 0.30 + h / 200` (a peak of 16,000 m of height, 15,970 m above the water, needs 80.15). The step grows with A, and at high Amounts it shows as stairs on gentle slopes; where the stairs still show at the smallest Amount, add fine heights (next section).

Worked example: a range 2,800 m high. The highest peak stands 2,800 m above the water, so `A = 0.30 + 2,800 / 200 = 14.3`. The land then reaches (14.3 - 0.15) x 200 = 2,830 m of height, a 16-bit step is 200 x 14.3 / 65,535 = 4.4 cm, and the world is a high world (above 5). The config:

```
[02 BetterContinents.Heightmap]
Heightmap Amount = 14.3
High Terrain = Auto
```

and the line a real build of such a world logged at creation:

```
High terrain: this world's heightmap is read at Heightmap Amount 14.3, above the 5 that older
» versions allowed, so its land can reach 2830 m: Better Continents patches the game's rules
» that hold a height (what is inside a dungeon, where the ground is looked for from 3830 m,
» altitude limits of 1000 m and more, the grass, the AI's tiles)
```

Above 5, a high world. Valheim 1.0 was made for land under about 400 m, and a few of its rules hold an absolute height. A heightmap read at an Amount above 5, the most any version before 0.10.3 allowed, makes a **high world**, and Better Continents then applies its own versions of those rules (chapter 3 says which, and what stays the game's own). A world read at 5 or less runs the game's own rules, as every older world does, and one of them shows on a map whose ground rises above 500 m of height (470 m above the water): the game's grass finds no ground up there. In a high world the settings dump in the log, and `bc info`, have a `High terrain: ...` line with the top height the world is read at. `[02 BetterContinents.Heightmap] High Terrain` decides: **Auto** (the default) switches the rules on for a world whose heightmap is read at an Amount above 5, **On** gives them to every Better Continents world whatever its heights, and **Off** to none. It is a world setting: a new world takes it when it is made (From Config, `bc_import`, or a Directory's `export.cfg`, which carries it only when it is not Auto) and keeps it, and stores it only when it is not Auto; `bc h ht [auto|on|off]` changes it in a running world (Debug Mode, on the machine that runs the world), and players already connected keep the old mode until they join again. Off costs a world with high ground: no grass above 500 m, no plants, creatures or locations above about 1,000 m, no terrain, weather or building above 3,000 m (the game takes it for the inside of a dungeon), and no ground found above 6,000 m.

Keep `Heightmap Override All` **on** (the default) where the heights must be the ones you drew. With it off, the game makes its own biome heights on top of the heightmap, and the Mountain biome doubles whatever is over 80 m: a 1,000 m peak in the heightmap stands about 1,900 m, a 2,000 m peak about 3,950 m and Amount 81's 16,170 m about 32,350 m, with 0 to 40 m of the biome's own noise and about 4 m for each unit of slope on top.

> **Rule: A world read at an Amount above 5 needs Better Continents 0.10.3 or later** to play as it was made: an older version opens it without the high-world rules, and there a player more than 3,000 m up counts as inside a dungeon, the grass finds no ground above 500 m, and items, tombstones, plants and creature spawns find none above 2,000 to 6,000 m. An older version cannot make one either: it reads a config's `Heightmap Amount` above 5 as 5 (trap 56). A world whose `High Terrain` is On or Off is not opened by an older version at all: it stops at the setting it does not know (`Unknown feature` in its log) and treats the world as one without Better Continents. Tell the human before you make one.

Fine heights

A 16-bit step is 3 mm at Amount 1 and a quarter of a metre at Amount 81, where it shows as stairs on gentle slopes. **Fine heights** (Better Continents 0.10.3 and later) add 8 bits to every height from a second picture beside the heightmap: named after it with `-fine` before the extension (`heightmap.png`, `heightmap-fine.png`), 8-bit grey, the same size. The file format is fixed, format 1:

```
c  = the heightmap's 16-bit value
f  = the fine picture's byte as a signed number, -128 to 127 (a negative f is the byte 256 + f)
v  = (256 * c + f) / 16,776,960         the height value 0..1, in place of c / 65,535
```

16,776,960 is 256 x 65,535, so `f = 0` gives the heightmap's own value and each unit of `f` is 1/256 of a 16-bit step: 24 bits in all, about a millimetre at Amount 81. A **4-bit** fine picture uses only multiples of 16 (-128 to 112), steps of 1/16 of a 16-bit step, and 4 bits is usually enough. On a 16,383 px map set at Amount 81 the fine picture put the heights within 0.9 mm of the reference, where the heightmap alone was up to 109 mm off, for 5.7 MB more in the world file and 1.8 s more to make the world (measured on one machine).

The writer, as the fork's `tools/fine_ref.py` (the format's executable spec) does it, for a height value `v` (0..1) and `x = v x 65,535`:

```
c = floor(x + 0.5)                          exactly what a plain 16-bit writer makes
f = round((x - c) x 256), to even           kept in -128..127
    4 bits: f = 16 x round((x - c) x 16)    kept in -128..112
then f = 0 under water and on steep ground  (a neighbour a step or more away): good practice
```

So the heightmap is exactly the one a plain writer makes, and a world that does not read the fine picture has the heights it always had. `gen.py` truncates instead (`int(... * 65535)`, chapter 5): for fine heights, write `c` as above.

The fine picture may carry a record, a PNG text chunk named `BetterContinents` like the heightmap's Amount record:

```
Fine Format = 1; Fine Bits = 4; Heightmap CRC-32 = 2723AFF1
```

The record is optional, and its parts may come in any order; `Fine Bits` is for people (the reader counts the bits the bytes use). The CRC-32 is zlib's, of the whole `heightmap.png` file: a fine picture whose record names another CRC belongs to another heightmap and is not used, and one whose record names none is trusted as it is. Make the fine picture and its record **last**, after the heightmap's own Amount record (chapter 6, step 7): writing that record, or saving the heightmap from an editor, changes the file's bytes.

Sample: a writer for the pair, standard library only. It takes the exact heights (values 0..1, row 0 north) and writes both files and the record, by the rules above, at 4 bits:

```
import struct, zlib

def _png(path, w, h, depth, rows, text=None):     # grey PNG, filter 0, optional record
    def chunk(tag, data):
        crc = zlib.crc32(tag + data) & 0xFFFFFFFF
        return struct.pack('>I', len(data)) + tag + data + struct.pack('>I', crc)
    head = struct.pack('>IIBBBBB', w, h, depth, 0, 0, 0, 0)
    out = b'\x89PNG\r\n\x1a\n' + chunk(b'IHDR', head)
    out += chunk(b'IDAT', zlib.compress(b''.join(b'\0' + r for r in rows), 9))
    if text:                                      # the record, a tEXt chunk
        out += chunk(b'tEXt', b'BetterContinents\0' + text.encode('latin-1'))
    with open(path, 'wb') as f:
        f.write(out + chunk(b'IEND', b''))

def write_pair(folder, size, values, amount):
    xs = [v * 65535.0 for v in values]
    c = [int(x + 0.5) for x in xs]                # what a plain 16-bit writer makes
    sea = 0.30 / amount * 65535.0                 # the waterline, at Sea Level 0.5
    fine = []
    for i, x in enumerate(xs):
        f = 16 * round((x - c[i]) * 16)           # 4 bits; Python's round() is to even
        f = max(-128, min(112, f))
        y, k = divmod(i, size)
        near = (i - size if y else i, i + size if y < size - 1 else i,
                i - 1 if k else i, i + 1 if k < size - 1 else i)
        steep = any(abs(xs[j] - x) >= 1.0 for j in near)
        fine.append(0 if x < sea or steep else f)  # 0 under water and on steep ground
    rows = [struct.pack('>%dH' % size, *c[r * size:(r + 1) * size]) for r in range(size)]
    _png(f'{folder}/heightmap.png', size, size, 16, rows)
    with open(f'{folder}/heightmap.png', 'rb') as h:
        crc = zlib.crc32(h.read()) & 0xFFFFFFFF      # of the heightmap as written: so, last
    rows = [bytes(f & 0xFF for f in fine[r * size:(r + 1) * size]) for r in range(size)]
    _png(f'{folder}/heightmap-fine.png', size, size, 8, rows,
         'Fine Format = 1; Fine Bits = 4; Heightmap CRC-32 = %08X' % crc)
```

It keeps the whole map in Python lists, so it is for small maps and for learning; a 16,384 px map needs a writer built for memory. The fork's `fine_ref.py` (numpy; about 1.5 GB for a 16,384 px pair) has a `write` command that makes a pair from an existing heightmap at another Heightmap Amount (`fine_ref.py write <folder> --from-heightmap <png> --amount A`). On a 256 px test slope at Amount 40 (5 cm a pixel, under half a 12 cm step), the fork's checker accepted the pair (real output, shortened):

```
$ ~/upy/bin/python <fork>/tools/fine_ref.py check <folder> --amount 40
Fine heights: heightmap-fine.png refines heightmap.png with 4 bits a pixel.
  heightmap-fine.png: 256 x 256, 8-bit grey, 0.00 MB, record "Fine Format = 1; Fine Bits = 4;
» Heightmap CRC-32 = 1170A9F0" (after the image data)
  f is not 0 on 61,422 of 65,536 pixels (93.7%), from -128 to 112; 4 bits used (the bytes are
» multiples of 16)
  Heightmap Amount 40 (given), Sea Level Adjustment 0: a step of heightmap.png is 12.21 cm;
» with 4 bits, 7.63 mm
```

When the world does not use the fine picture, a `Fine heights: ...` line in the log says why, and the settings dump repeats it. The checks, in their order:

1. there is no fine file (then nothing is said);
2. the world is version 11, which has no fine heights (an older world, or one made with `Override version` 11; chapter 10);
3. Heightmap Alpha as it was before 0.10 (Legacy) reads the heightmap as 8-bit grey (as an older world does);
4. `Compact Maps` is Off (chapter 10);
5. the heightmap cannot be read (an error);
6. the heightmap is not 16-bit grey (grey, or grey and alpha);
7. the fine picture cannot be read (an error);
8. it is bigger than the largest map this machine reads (16,384 px, or `Max Map Size`);
9. it is not 8-bit grey (no alpha, no palette);
10. it is not the heightmap's size;
11. its record's Fine Format is not 1;
12. its record names another CRC-32 (`... was made for another heightmap.png (its record names CRC-32 ...; this one is ...). If heightmap.png was edited, make heightmap-fine.png again or delete it.`);
13. its image data cannot be read (an error);
14. every byte is 0, so it changes nothing (`... the world is the one heightmap.png makes alone`).

Heightmap Alpha on a new world is not a reason: a 16-bit grey and alpha heightmap, read as the blend (chapter 10), takes fine heights too.

A fine picture that is used logs `Fine heights: heightmap-fine.png refines heightmap.png with N bits a pixel: ground in steps of ... at Heightmap Amount A (heightmap.png alone: ...)`, makes the world compact (`Compact Maps is on for this world: its heightmap has fine heights ...`), and gives the settings dump and `bc info` a `Fine heights: N bits a pixel on top of the heightmap's 16 ...` line. The fork's `tools/fine_ref.py check <folder>` gives the same verdict offline: run it with `~/upy/bin/python`, which has the numpy and Pillow it needs, and capped (a 16,384 px pair takes about 1.5 GB). The line a real 16,383 px world at Amount 14.3 logged:

```
Fine heights: heightmap-fine.png refines heightmap.png with 4 bits a pixel: ground in steps of
» 2.73 mm at Heightmap Amount 14.3 (heightmap.png alone: 4.4 cm), 2.9 MB of tiles, 0.09 bits a
» pixel. A Better Continents older than this one cannot read a world made with fine heights.
```

Check: `fine_ref.py check` says `refines heightmap.png with 4 bits a pixel` before you stage, and the creation log has the `Fine heights: ... refines ...` line and no `Fine heights:` line that gives a reason from the list above. Then remember the rule below.

Rule: A world made with fine heights needs Better Continents 0.10.3 or later for good. It is always compact and stores its heights in a new block format: Better Continents 0.10.0 to 0.10.2 cannot read it (`Failed to load settings from <file>: a map saved in format 2, which this version of Better Continents cannot read` in their log), switch Better Continents off for the world, give it vanilla terrain and leave its settings file alone (and 0.9 reads no compact world). Use fine heights only where the steps show, and tell the human before you make such a world.

A world export writes the fine picture with its record when the export's `fine=` option asks for it: `bc_export fine=auto|off|4|8`, or the export window's Fine heights choice. `auto`, the default, writes 4 bits where one 16-bit step is more than 1.5 cm, from a Heightmap Amount of about 4.92; `fine=4` and `fine=8` write it at any Amount, and `fine=off` never (chapter 9). It is an option of each export, not a config key. The workspace's `png.py` and `png_extras.py` write neither (appendix A); the sample above does.

The biome map

The legend `biomemap.txt` (written with the defaults if it is missing) has one line per biome, `Name: RRGGBB`. The ten names of Valheim 1.0 are below; with Expand World Data installed a legend may also name the biomes its `expand_biomes` yaml adds. A line that reads badly is logged and skipped, not the whole legend: an unknown name's colour reads as None, and a vanilla biome with a bad colour keeps its default one (appendix B shows the rule for every legend kind).

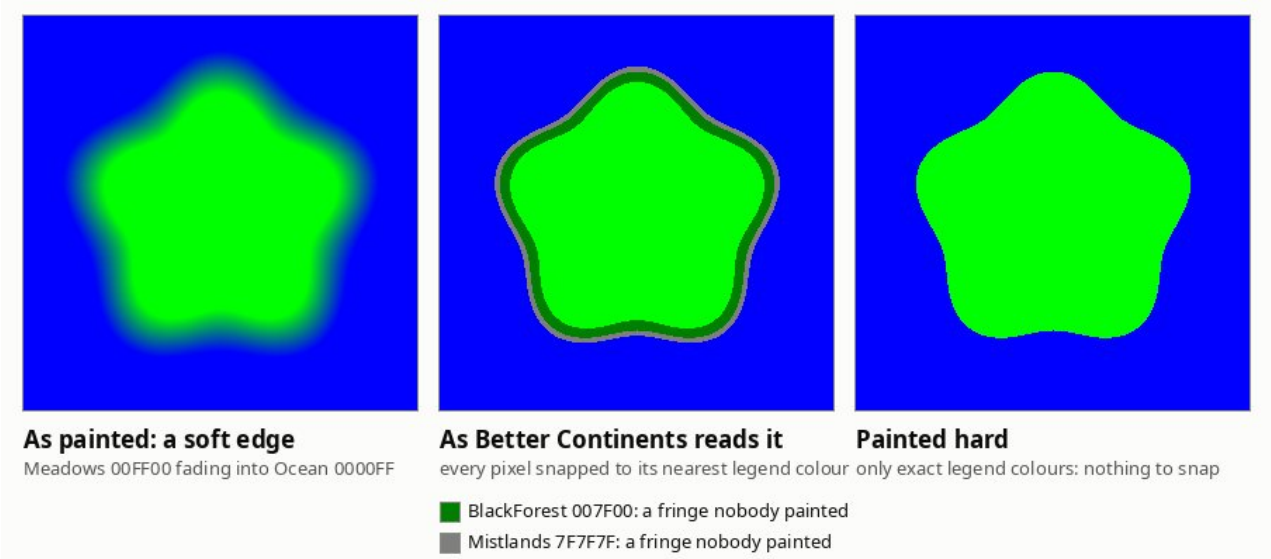
Sample: `biomemap.txt`, exactly as Better Continents writes it when the file is missing (copy it as it is):

```
None: 000000
Meadows: 00FF00
BlackForest: 007F00
Swamp: 7F7F00
Mountain: FFFFFF
Plains: FFFF00
Mistlands: 7F7F7F
AshLands: FF0000
DeepNorth: 00FFFF
Ocean: 0000FF
```

A colour may also be written `#RRGGBB` or `r,g,b` (`Meadows: 0,255,0`). A comment is a whole line that starts with `#`; never put a comment after an entry in this legend or the location legend, because it becomes part of the colour and the entry is skipped.

Facts that matter when you paint it:

- **Pixels snap to the nearest legend colour**, so keep every edge hard. A blended pixel between Meadows and Ocean (0, 128, 128) snaps to Mistlands, and a soft coastline grows a fringe of Mistlands. The generator writes exact colours.
- **None** (black) means "let vanilla decide". Never leave it on a map you control: vanilla's biome there will not match your heights, and a None region counts as valid for every alt biome.
- **A biome you leave out is skipped**, and every location that needs it is reported as failed. A map without Mistlands has no Queen.
- **Where water gets a land biome.** Vanilla keeps the land's biome in the water down to 26 m depth and calls it Ocean only deeper. Some coastal locations depend on this: the frozen ships lie in *Deep North* water, the dvergr harbours stand in *Mistlands* water. On a map, water is whatever you paint it, and painting Ocean right up to the waterline leaves those locations nowhere to go. But the alt-biome rules (chapter 3) punish a region whose edge is under water. ProximaMaxi labels Deep North water to 16 m depth and Mistlands water to 1 m, never against another biome's shore, and nothing else.
- The **Deep North** in Valheim 1.0 is a real biome with its own locations; vanilla decides its weather, snow and calm sea by a fixed geographic test, and the mod makes the weather and the snow follow your Deep North pixels instead. Since Better Continents 0.10.5 the calm sea follows them too, drawn and felt (chapter 13); the terrain shader's Deep North look stays at the game's fixed circle (chapter 3).
- The mod votes the biome at a world point from the four surrounding pixels (bilinear weights, the heaviest biome wins). The checkers reproduce that vote.
- The **ground** inside a 64 m terrain zone follows the biome map only as finely as `Biome precision` allows (chapter 9): at 0, the vanilla value, a zone takes its biome from its four corners.



A computed example of the snap: the same Meadows island painted with a soft edge (left), read by the mod's rule, every pixel to the legend colour nearest in RGB (middle), and painted with exact colours (right). Notice the two fringes the soft edge grows, Black Forest nearer the land and Mistlands nearer the sea: neither was painted, and both would be real biomes in the world.

Do	Don't
paint with exact legend colours and hard edges (a pencil, not a brush; no anti-aliasing)	scale, blur, feather or anti-alias a biome map
label water with a land biome only where its locations need it (Deep North to 16 m, Mistlands to 1 m)	paint Ocean up to the waterline of the Deep North and the Mistlands
give every biome you want its own colour	leave black (None) on a map you control

Do	Don't
resize with nearest-neighbour sampling, if you must resize	resize with bilinear or bicubic sampling (it blends colours)

> **Check:** the log line Better Continents prints for every biome map it reads gives each biome's share. Compare it with `check.py`'s `% map` column: on ProximaMaxi they agree to the hundredth. The real line, from a ProximaMaxi world:
>
> `Biome map (2048 x 2048): Meadows 3.04%, Swamp 1.44%, Mountain 2.72%, BlackForest 6.41%, Plains 3.17%, AshLands 3.79%, DeepNorth 6.69%, Ocean 70.13%, Mistlands 2.60%`
>
> A biome you did not paint in that line (a few tenths of a percent of Mistlands on a map without Mistlands) is a soft edge somewhere. A `None` share means black pixels.

The heat map

The mod answers the game's "is this Ashlands?" question with `heat > 0`, literally.

> **Trap:** `heatmap.png` must be **exactly zero** everywhere that is not Ashlands. One stray non-zero pixel turns that ground into Ashlands, silently. The generator asserts this before writing, and `check.py` counts `heat > 0 on land that is not AshLands` and requires 0.

Heat should still extend into the sea around the Ashlands as a ramp rising toward the coast: that is vanilla's own ocean gradient and it is what damages ships on the approach. Since Better Continents 0.10.3 it is also what colours the sea red (per pixel since 0.10.5): the water is drawn red wherever heat > 0 over the sea, with no falloff past the heat map's pixels (chapters 3 and 13), so the heat field decides how far the red sea reaches. So bound the heat field geometrically (an ellipse around the Ashlands landmass, faded), not by tracing the biome edge. The config's `Heatmap Scale` (default 10) multiplies the pixel; most heat effects cap at 1, so a pixel value of 0.1 is already the full effect, and ProximaMaxi's maximum pixel is 0.16.

> **Rule: Write a heat map even for a world with no Ashlands.** With a heat map (and `Heatmap Scale` above 0), the sea's heat is the map's heat, so an all-black `heatmap.png` keeps every sea cool. Without one, the game's own Ashlands gradient stays (its ring laid out to the world's size on a new world): the sea past that ring in the south is still hot, damages ships and is drawn red, on a map that has no Ashlands at all. What is Ashlands ground then follows the biome map's AshLands pixels (the mod's code: `GetAshlandsOceanGradient` and `IsAshlands` read the heat map only when a world has one).

Sample: a heat field that keeps the rules, a pattern for a generator (`sstep` is the smooth step of chapter 5's minimal generator; `CX, CZ, RX, RZ` are an ellipse round your Ashlands, in metres):

```
def heat(x, z, biome):                    # 0..1, written as 16-bit grey
    if biome not in ('AshLands', 'Ocean'):
        return 0.0                        # never on other land: heat > 0 IS the Ashlands
    d = ((x - CX) / RX) ** 2 + ((z - CZ) / RZ) ** 2      # 1.0 on the ellipse's rim
    h = 0.16 * sstep(1.0, 0.5, d)         # full well inside, fading to 0 at the rim
    return max(h, 0.02) if biome == 'AshLands' else h    # every Ashlands pixel is hot
```

Sample: an all-black heat map (`png.py`, chapter 5):

```
import png
SIZE = 2048
png.write_gray16('heatmap.png', SIZE, SIZE, [0] * (SIZE * SIZE))   # no Ashlands; a cool sea
```

> **Check:** `check.py` prints the three invariants; all three must be 0, as on map version 13 (real):
>
> ```
> dry ground labelled Ocean: 0   (minimap paints it as sea)
> heat > 0 on land that is not AshLands: 0   (MUST be 0 - IsAshlands is heat > 0)
> AshLands pixels with no heat: 0
> max heat 0.160
> ```

The cold map and the location exclusion mask

Two more optional pictures, in the same Directory and read by the same square (chapter 2's pixel to world rules).

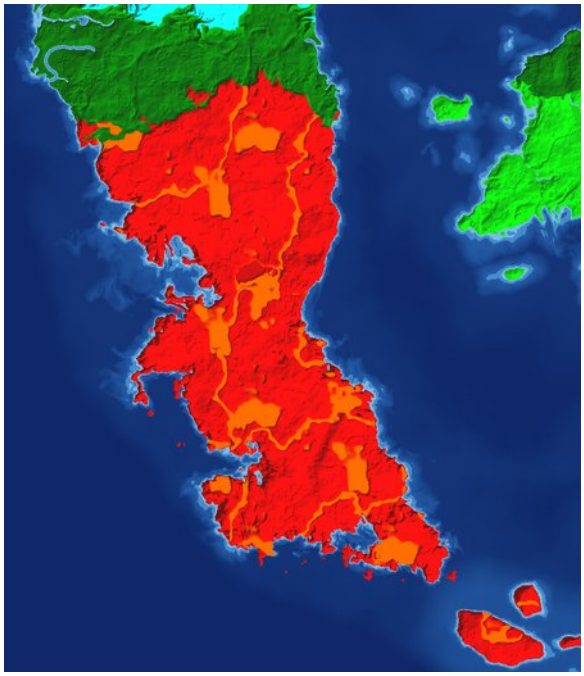
`coldmap.png` (`[06 BetterContinents.Maps] Coldmap File`, `Coldmap Scale`, console `bc cold`): 16-bit grey, the Deep North. Wherever `scale x pixel / 65535` is above 0 the game's `IsDeepnorth` answers yes, which drives the cold weather, the snow and calm sea (with `[01 BetterContinents.Global] Deep North Water` on, the default, the wave fade the sea follows is that value clamped to 0..1). Without a cold map the Deep North follows the biome map's Deep North pixels, and the game's own rule where the biome map says nothing. The same trap as the heat map: black exactly everywhere that is not Deep North. The usual form is a ramp over the 200 m at the border (the game's own wave fade) divided by the scale, never below one grey step, which is what an export writes (chapter 9).

`no_spawns_mask.png` (`Location Exclusion Mask`): wherever the mask is above 1 % of white, the game places none of its random locations; the start temple and the dots of `locationmap.png` are exempt. It keeps towns and roads clean without using `Skip Default Locations`, which would end all of the game's own placement.

Three more settings in `[01 BetterContinents.Global]` decide what a heightmap world adds on top of its heights: `Mountain Noise` (default 1; 0 is perfectly flat to the heightmap, on Mountain biomes), `Mistlands Noise` (default 1, the Mistlands crags), and `Ashlands Water` (default true: whether the Ashlands' red sea follows the world's hot sea, not the game's south circle). Lowlands, towns and shores stay faithful to the heightmap whatever the noise is. A world made from an export sets both noises to 0, because the exported heights already hold them (chapter 9).

The lava map

The game decides Ashlands lava from its Ashlands vegetation mask; the mod loads `lavamap.png` (16-bit) as that mask. Lava appears where the value is at least 0.6. Locations read the same mask through their vegetation window: `LeviathanLava` wants 0.6 to 1.0 (it stands in lava), the charred fortress and the Ashlands boss location want at most 0.1, the Morgen holes at most 0.3, the fortress ruins at most 0.8. So a map with no lava map has no lava leviathans, and a map with lava everywhere has no fortresses. ProximaMaxi paints pools (about a tenth of the Ashlands) and streams down the tributary channels, kept off the beaches; the ground is let down into the pools so that they read as basins.



Map version 13's Ashlands arm as `preview.py` draws it: the lava map tinted orange from a value of 0.25, fully from 0.75 (the game's lava starts at 0.6). Notice the pools in basins and the streams down the channels, and the wide red ground between them, where the fortresses (at most 0.1) can stand.

Location	Lava map value it wants	So it needs
`LeviathanLava` (the lava leviathans)	0.6 to 1.0	lava (the game draws lava from 0.6)
the fortress ruins	at most 0.8	ground that is not deep in a pool
the Morgen holes	at most 0.3	ground clear of lava
`CharredFortress`, `FaderLocation` (the Ashlands boss)	at most 0.1	ground well clear of lava

> **Check:** `locations.py` places `LeviathanLava` and `CharredFortress` at their full counts (100 and 20 on map version 13); a short count with the reason `vegetation` means too little lava, or too little lava-free ground.

The forest map

Vanilla's forest factor is a noise value where **lower means more forest** (`InForest` is `factor < 1.15`). The mod normalises it through `InverseLerp(1.850145, 0.145071, factor)` into a 0-to-1 "forestness" `n` (0 bare, 1 dense) and combines it with the map value `fmap` as `Lerp(n, n * fmap, Multiply) + fmap * Add`; the result is forest (`InForest`) above about 0.41. What black means depends on the pair:

- At the defaults (`Forestmap Multiply = 1`, `Forestmap Add = 1`) the result is `fmap * (n + 1)`: black, and anything below about 0.2, is a clearing; 0.41 or more is forest everywhere; in between, the game's own forest pattern decides.
- With `Forestmap Multiply = 0` and `Forestmap Add = 1` it is `n + fmap`: black is the game's own forest, and a **higher map value adds trees**, up to forest everywhere from about 0.41.

Worked example: the same map value under the two pairs. The result is forest where it passes 0.41:

Map value	Defaults (1 and 1): result `fmap x (n + 1)`	Additive (0 and 1): result `n + fmap`
0 (black)	0: a clearing everywhere	n: the game's own forest
0.15	0.15 to 0.30: a clearing everywhere	forest where the game's n is above 0.26
0.30	forest where n is above 0.37	forest where n is above 0.11
0.45	0.45 to 0.90: forest everywhere	forest everywhere

The two configs, to copy:

```
[04 BetterContinents.Forest]
# the defaults: black is a clearing
Forestmap Multiply = 1
Forestmap Add = 1
```

```
[04 BetterContinents.Forest]
# ProximaMaxi's pair: black is the game's own forest, and the map adds trees
Forestmap Multiply = 0
Forestmap Add = 1
Forest Factor Overrides All Trees = true
```

Rule: Choose the pair by what black must mean. ProximaMaxi uses the additive pair (0 and 1), so its map adds woods to the game's own pattern; where open ground must stay open (heath, the slopes above a tree line), keep the defaults, where black is a clearing. Leave `Forest Amount` at 0.5 (neutral: its offset is `Lerp(1, -1, value)`) and `Forest Scale` at 0.5, the game's own size of forest and clearing patches: the default since Better Continents 0.10.2, while 0.10.1 and older default to 1, five times that size, so write 0.5 there. Set `Forest Factor Overrides All Trees = true`: without it only the vegetation the game itself ties to the forest factor follows the map, and the flag adds the firs, pines, swamp trees, roots and webs (13 prefabs, listed in the Guide's Forest page).

Do not hold tree density down to protect the minimap. Vanilla paints the forest overlay only in Meadows, Plains and Black Forest; Swamp, Mountain and Deep North draw none and Mistlands has its own mist channel. A minimap that reads as forest everywhere was a bug in the mod's minimap patch (fixed in 0.8.0), not honest density.

Check: the settings dump has `Forestmap multiply 0, add 1` (the pair the world was made with) and `Forest Factor overrides all trees` when the flag is on (both real lines, from a ProximaMaxi world). In the world, walk a forest edge you painted: the trees should stop where the map says.

The location map

`locationmap.png` pins specific sites: black is "nothing here", and any pixel of a legend colour asks for that location at that spot. The legend `locationmap.txt` is `LocationName: r,g,b` with the game's prefab names; ProximaMaxi pins nine:

```
StartTemple: 255,0,0
Eikthyrnir: 255,153,0
GDKing: 0,255,0
Vendor_BlackForest: 0,0,255
Hildir_camp: 128,0,255
Bonemass: 0,255,255
Dragonqueen: 74,134,232
GoblinKing: 255,255,0
Mistlands_DvergrBossEntrance1: 255,0,255
```

That is the file as it is, one entry a line: copy it as it is. A missing legend is written with the mod's default one, 88 entries, which gives the same colours to the first seven of these and others to Hildir's camp (255,105,180) and the Queen's entrance (153,0,255); a legend you write yourself holds only the locations you pin.

Colours must match exactly. A pixel is a pin only where its colour is a legend colour to the last unit; unlike the biome map, nothing snaps. A soft-edged dot therefore makes one real pin (its exact core) and a ring of colours no legend names, which the log lists, one line per colour, as the mod's source writes it:

```
No spawns are mapped to color #<RRGGBB> (which has N spawn positions defined)
```

How a pin becomes a position depends on the world's version:

- **A new world** puts a pin **exactly on its pixel**, on the same grid the height and biome maps use (`i / (size - 1)`): a map of 10,501 px over 21,000 m pins on the 2 m grid of its heights. A pin of several adjacent pixels of one colour lands on the one nearest its middle, so the middle pixel of a 3 x 3 block is the site. Locations that share a colour are dealt out with a seeded random, so the same map gives the same world every time. An export folder's maps are the exception: an export writes each location's pixel for the older corner reading (`i / size`), and a world made from one reads it so.
- **An older world** picks one pixel of the pin at random and uses its corner (`i / size`), up to a pixel south-west of where the other maps put it. Paint a 3 x 3 block rather than a lone pixel so that the pick stays within a pixel of the site you chose.

The generator's 3 x 3 blocks are right for both. `locations.py` models a pin as the mean position of the colour's pixels at `i / (size - 1)`, which is the middle of one block: the position a new world pins.

Sample: painting a pin, a 3 x 3 block of one exact colour around pixel `(i, j)` of an `n` px map held as a flat list of RGB tuples:

```
def pin(pixels, n, i, j, colour):
    for dj in (-1, 0, 1):
        for di in (-1, 0, 1):
            pixels[(j + dj) * n + (i + di)] = colour

pin(lm, 2048, 1013, 1238, (255, 0, 0))     # the start temple of the worked example above
```

The mod registers these sites before the game places anything, then the game's own placement runs for everything else (unless `Skip Default Locations` is on, which you do not want: it skips every default location except the start temple). **The game does not check a pinned site.** It is registered where you put it, whatever the biome or the altitude, and it counts toward the location's quantity; the game's random pass then places the rest by its rules. A real build showed it: Moder's altar was pinned on a mountain below its 150 m minimum, and the game placed that one site and failed the other two on altitude. The generator therefore searches for a suitable pixel near each intended site (right biome, off the river banks) rather than painting the colour at a fixed coordinate: a boss in the sea or an altar under water is a bad world, though the game would accept it.

The game keeps one location per 64 m zone, so a pin whose zone already holds a location is not placed. The log says so (`Location map: X at P was NOT placed: its zone Z already holds H`); keep sites more than a zone apart.

> **Rule:** `Override Start Position` (`[05 BetterContinents.StartPosition]`, with Start Position X and Y in metres from the centre, up to 1,000,000 either way) moves the start temple. A new world places it **before** the pins, so a pin in its zone cannot take it; an older world places it after them and loses it to a pin there. A start that could not be placed logs `The start position override was NOT applied` with the zone and what holds it: move the start or the pin.

> **Check:** at creation the log has, for each pin, a `Found #RRGGBB area of 9 size at ...` line (9 is the 3 x 3 block), a `Selected <name> for spawn position <x>, <y>` line (the position in fractions of the map), and then `Position of <name> (1/N) overriden: set to <position>` when the site is registered; no `was NOT placed` and no `No spawns are mapped to color` line. Real lines, from a ProximaMaxi world (an older world's log, before exact pins):
>
> ```
> Found #FF0000 area of 9 size at 1012, 1240, selected position 0.4941406, 0.3955078
> Selected StartTemple for spawn position 0.4941406, 0.3955078
> ```
>
> In the world, `bc bosses` puts a pin on the large map for the bosses, the start temple and the trader.

Everything you paint here is a matter of the map's story; everything you do not paint is decided by the algorithm in chapter 3.

The alt-biome map

`altbiomemap.png` plants the game's alt biomes on regions of your choice; black is unplanted (the game's random placement decides there). The legend `altbiomemap.txt` has one line per colour:

```
<alt biome>[ + <alt biome> ...]: <colour>      plant where the image has this colour
<alt biome>[ + <alt biome> ...]: at <x>, <z>   plant on the whole region under x, z
none: <colour>                                 no alt biome here, planted or random
```

The `at <x>, <z>` form takes world positions in metres.

Names are the game's internal names (appendix C lists all 32, with the mod's default colour for each). `*Mistlands` is a wildcard, `!Name` forces past the content rules, `+` stacks. Colours are `RRGGBB`, `#RRGGBB` or `r,g,b`; pixels within 20 RGB units of a legend colour count as it, and a colour within 40 units of another is warned about. Black is refused as a legend colour. At most 254 entries.

Sample: `altbiomemap.txt`, with each kind of line. Here, unlike the biome legend, a comment may follow an entry when the `#` has a space on each side:

```
# My island's alt biomes. Black in altbiomemap.png is left to the game.
Mushroom: FF969C                  # the meadow east of the river, painted in the picture
Wolf Mountain: at 1709, 3508      # the whole Mountain region under x 1709, z 3508 (metres)
none: 004080                      # this meadow stays plain, planted or random
```

The first two names and colours are ProximaMaxi's (its whole legend is in appendix C; its own file uses only colour lines), and `004080` is more than 40 RGB units from every colour of that legend, so it draws no warning beside them.

How planting behaves (the fork's `ALTBIOMES.md` is the authority):

- A colour **claims only the base biomes its alt biomes belong to**. Dark Meadows painted across a meadow and its beach takes the meadow and leaves the beach alone. So painting past a biome edge is harmless, except for a multi-biome alt (Lantern takes seven biomes, Mushroom two): its colour claims all of them, so it must not spill onto neighbouring land you did not mean.
- Every 4-connected patch of one colour becomes **a region of its own**, carved out of the game's region; what is left of the game's region stays one region. Nothing is merged. On Ashlands, Deep North and Ocean, all patches of one colour form ONE region (the game treats those biomes as one region each), counted once.
- Planted regions **ignore the game's occurrence rules** (size, distance, height, position, neighbour, chance) and keep its content rules (disabled alt biome, wrong base biome, incompatible pair) unless forced with `!`.

- Planted regions **count toward each alt biome's minimum and maximum** when the game then places the rest at random on unplanted regions (the quota rule). Plant Mushroom three times and the game plants no more (its maximum is 3); plant Fortress Mountain once and the game may add one.
- The grid is 12 m: a patch narrower than a cell or two vanishes or breaks up. Paint whole features.
- The game builds this grid for every world: the alt-biome `Mode = Off` means no alt biomes, not no grid.
- A world carries its own baked copy; editing the PNG or the legend changes nothing for an existing world.
- On a new world laid out to a size other than vanilla's the grid covers the whole world, and the `Grid` setting has no effect (chapter 10).

Do	Don't
paint whole features: an island, a valley bounded by rivers and coast	paint a speck or a one-cell tongue (it becomes a region of its own and uses a slot)
plant up to an alt biome's maximum, and leave the rest to chance	plant past the maximum and expect the game to add more
keep a multi-biome alt's colour (Lantern, Mushroom) on the land you mean	let Lantern's colour spill onto neighbouring biomes or their water
keep legend colours more than 40 RGB units apart	use two colours so close that pixels between them may go to either

The game log reports the outcome at every world load: one summary line, one line per colour, and one line per alt biome with the reason it did or did not place (`+3 planted (counted toward the maximum)`). The summary from a real ProximaMaxi world:

```
Alt biomes: planted: 23 region(s), 23 alt biome(s) applied, 0 protected (none), 0 skipped for
» the base biome, 0 refused (disabled or incompatible)
```

> **Check:** at creation the log has `Alt-biome legend altbiomemap.txt: N colours, M points.` (19 colours and 0 points on ProximaMaxi) and `Alt-biome map <path>: 2048x2048, N planted pixels, decoded in N ms.`, and no warning that pixels `match no legend colour (within 20) and are treated as unplanted` (that warning lists the most common stray colours and the nearest legend colour to each). At every load, the summary above with `0 refused`. In the world, `bc ab info` reports what the alt-biome pipeline produced for the world and why each alt biome did or did not place, and `bc_altbiomes here` describes the region you stand in.

Baking, the config and the world

> **Rule: Maps are baked into the world at creation.** The mod stores every map, with the effect of its legend, in the world's `BetterContinents` settings file (`worlds_local/<name>/BetterContinents`, an extension-less file inside the world folder). Editing a PNG afterwards does nothing to that world. Every map change needs a brand new world, except that `bc reload` (a cheat command, run on the host) reads a changed picture into the loaded world (chapter 6).

"Baked" here means the maps. A `placements.bcp` beside them (Better Continents 0.10.4 or later) is read the same way, into the same settings file: the world's layer of baked buildings and towns, which chapter 12 covers.

> **Rule: The config is read when the world is created.** `BepInEx/config/BetterContinents.cfg` in the profile is read at launch and again whenever the file changes, and its values are baked into a world when the world is created. The safe sequence is still stage, quit the game, relaunch, then create the world.

> **Rule: The preset must be `From Config`.** At New World, the dropdown beside the seed must say `From Config` (the config's `SelectedPreset = From Config` pre-selects it). Anything else and the mod does not read the config's `Directory`: `Disabled` generates vanilla without a warning, and a preset makes the world from its own settings and maps (a preset that ships with the mod holds none, and makes a version 11 world, chapter 10).

Sample: the sequence to give a human, word for word (it is chapter 6's step 9):

```
1. Quit Valheim if it is running, then start it again with the profile that has
   Better Continents.
2. Start the game with your character, then press New World in the world list.
   Type a NEW name (never one you have used before).
3. Set the dropdown beside the seed to "From Config". Press Create.
4. Wait: the world takes longer than a vanilla one to make. Then play into it.
5. Quit, and send me the log: <profile>/BepInEx/LogOutput.log (it is replaced every
   time the game starts, so copy it before you start the game again).
```

> **Check:** the creation log says `applying selected preset 'From Config'` (real line: `... first time save of ProxEra01, applying selected preset 'From Config'`), then `Time to load <path>` for each map of the Directory, then the settings dump, whose first line is `Version 12` for a new world. A line naming another preset, or no `Time to load` lines, means the world was not made from your maps: make a new one.

On a world that uses Better Continents, the server and every player must run exactly the same Better Continents version as the host (a join with another version is refused, chapter 10), but they do not need the images: the world carries them. A dedicated server cannot create a Better Continents world from a config: Better Continents makes a world's settings only from the game client's New World screen (a server that creates a world on its first start gives it none, so it runs vanilla). Create the world on a client, then copy its folder to the server; joining players get the maps from the server.

3 What the game demands of a map

The game does not care how a map looks. It samples the world on a grid, cuts it into regions, and runs two placement algorithms over it: one for alt biomes, one for locations. Both are replayed by the workspace's checkers (`altbiomes.py`, `locations.py`), which were validated line for line against real builds, at the default world size. This chapter is what those algorithms want, so that you can design for them instead of discovering their refusals in the game log.

The game's grid

At every world load, `AltBiomeWorldData.GenerateBiomePoints` samples the biome and the height on a 2048 x 2048 grid of 12 m cells centred on the origin: point (i, j) is at `(i - 1024) * 12 + 6` metres on each axis. Points further than 10,500 m from the centre are marked Ocean at height -1000. This grid is the source of truth for the base biome of every terrain chunk's corners, for alt biomes, and for location candidates. It is regenerated on every load, on the server and on every client (the game never reads its cache), and nothing about it is sent over the network: clients and server agree only because they hold the same maps, which the world file guarantees.

A new world (chapter 10) whose size is not vanilla's gets a grid of its own: the same 12 m points, as many as cover World Size + Edge Size with vanilla's margin (`2 x ceil(1024 x radius / 10,500)` points a side: 2048 at the default size, 3190 at a 16,350 m radius, 6392 at 32,764 m), and every point inside that radius is sampled, so alt biomes and locations reach the whole world. A bigger grid costs time and memory on every machine at every load: about 0.2 GB at a 16,350 m radius, about 0.8 GB at 32,764 m. Better Continents builds it on every core: about 5 s at 32,764 m on a 32-core machine, where the game's own way takes 29 s on one core (measured on one machine; fewer cores take longer, and a world on the game's own terrain gains less). From a radius of 42,000 m the grid stops growing at 8192 points a side, and its points spread wider than 12 m. On an older world the grid stays vanilla's: 2048 points to 10,500 m.

Worked example: the grid of a world. At the default size point (0, 0) of the grid lies at (-1024 x 12 + 6) = -12,282 m on each axis and point (2047, 2047) at +12,282 m; the corners past 10,500 m are Ocean at height -1000, so only the disc counts. A new world of radius 16,350 m (World Size 15,850 + Edge Size 500) gets `2 x ceil(1024 x 16,350 / 10,500)` = `2 x ceil(1594.5)` = 3,190 points a side; one of radius 32,764 m gets 6,392.

```
import math
def grid_points(radius):              # radius = World Size + Edge Size, metres (a new world)
    return min(8192, 2 * math.ceil(1024 * radius / 10500))
print(grid_points(10500), grid_points(16350), grid_points(32764))     # 2048 3190 6392
```

Check: on a new world that is not at the default size, `bc info` prints `World size N` and `The maps span World Size and Edge Size: N m across, and the world is laid out to that size`. The checkers in this workspace model only the default grid (appendix A): for another size, read their numbers as a guide, not a proof.

Regions and alt biomes

`GenerateSectors` flood-fills 4-connected grid points of one biome into **regions** (the game calls them sectors). Ashlands, Deep North and Ocean are each ONE world-wide region, however many patches they have. For every region the game records:

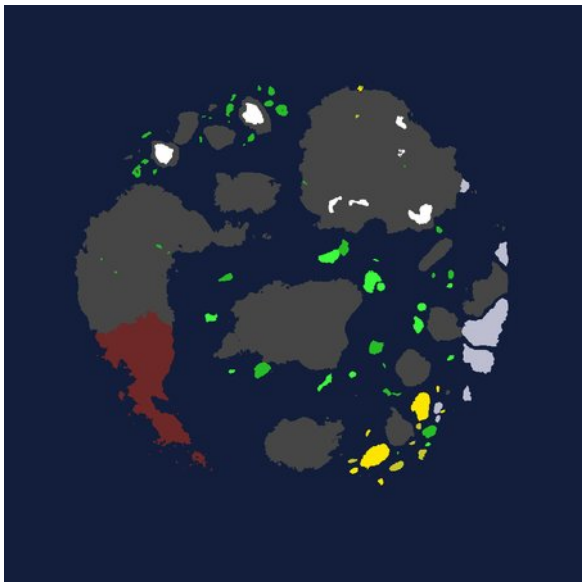
- `EdgeCount`: the number of border cells;
- `DistanceFromCenter`: the distance of the mean border position from the origin (a region wrapped round the origin measures near 0);
- `HeightAvg`: `(lowest + highest) / 2` of the border points' absolute heights.

Then `GenerateAltBiomes` goes through each base biome and each of its alt biomes, shuffles the biome's regions with a seed from the world seed, and gives the alt biome to regions in that order while its count is below its maximum (ignoring the chance roll while below the minimum). Every region must pass `BiomeSector.CanAddModifier`:

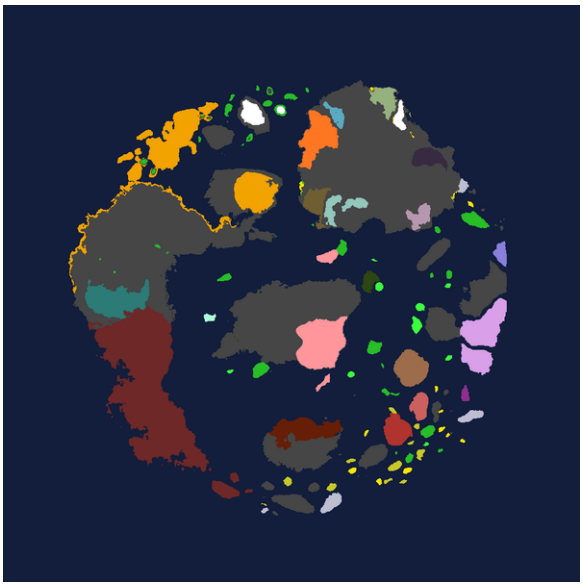
- distance from the centre at least the alt biome's `minDistanceFromCenter` (500 to 2,000 m);
- `EdgeCount` inside the alt biome's window, e.g. [66, 199) for Mushroom, [199, 299) for the three Mountain alts (which are mutually exclusive), [99, 599) for Rockless Mistlands, [20, 80) for Kalhygge Black Forest;
- `HeightAvg` at least 30 absolute, i.e. the border's (min + max) / 2 at or above the waterline;
- no incompatible alt biome already on the region;
- `m_requireNeighbor` is broken in vanilla: any non-zero value means never placed (no vanilla alt biome sets it).

What this means for a map:

> **Rule: Big regions get nothing; islands and mid-sized patches are the hosts.** A continent's forest has an edge count in the thousands and qualifies for no alt biome. For a roundish region the edge count is about half its radius in metres, so a window of [66, 199) means regions about 130 to 400 m in radius, [199, 299) about 400 to 600 m. A map that wants the game to place alt biomes at random needs islands and patches of those sizes, in every biome, beyond the distance minimum. A map that plants them (chapter 2) can put them anywhere.



07b-altbiome-hosts.png



13h-altbiome-regions.png

Left: map version 07's regions as `altbiomes.py` draws them before any planting: grey cannot host an alt biome at random, bright (in the base biome's colour) hosted one in at least one of 200 simulated seeds, dark red is the one Ashlands region. Notice that every continent is grey and the hosts are islands and small patches. Right: map version 13, with the 23 planted regions in their legend colours.

Worked example: an island the game will give Mushroom at random. Mushroom's window is `EdgeCount` 66 to 198, so a round island of radius about 130 to 400 m (260 to 800 m across): 25 to 78 pixels across on a 2048 px map of the default square. It must lie at least 500 m from the centre (Mushroom's minimum), and its border's (lowest + highest) / 2 must reach 30 m of height: give it a cliff or a headland on the coast. Then, at random, it is one of many candidates; to be sure of it, plant it (chapter 2).

The lines `altbiomes.py` prints for unplanted regions that can host (real, map version 13): the region, its base biome, its mean border position and distance, `edge` (EdgeCount), `h` (HeightAvg, absolute) and how often each alt biome landed on it over the seeds:

```
  #102   Meadows     at (  3731,   293)  3742 m  edge   85  h  32.4   Dark Meadows 63%, Lantern
» 54%, Birch Meadows 35%, Dandelion Meadows 34%
  #72    Meadows     at (  2526, -3446)  4272 m  edge   66  h  37.5   Dark Meadows 58%, Lantern
» 51%, Raspberry Meadows 36%, Dandelion Meadows 32%
  #104   Meadows     at (  6660,   340)  6669 m  edge   57  h  31.1   Menhir 26%, Smalltree
» Meadows 22%
```

Region #104 has an edge of 57, under the 66 most Meadows alts need, so only the two with a window from 33 (Menhir, Smalltree Meadows) can land there; its `h` of 31.1 is 1.1 m above the waterline: a border of beach all round, which barely passes.

> **Rule: A host must have a real cliff somewhere on its shore.** An all-beach island's border sits at the waterline, HeightAvg 30 exactly, a coin flip. Give host islands some high ground on the coast.

Do	Don't
islands and patches of 130 to 400 m radius for most alt biomes, 400 to 600 m for the three Mountain alts	rely on the game to give a continent's biome an alt biome at random (its edge count is in the thousands)
a cliff or headland on every host's shore	an all-beach island (HeightAvg 30, a coin flip)
hosts beyond each alt biome's minimum distance (500 to 2,000 m)	hosts near the centre
plant what the story needs, and let the game fill the rest	plant everything, and then wonder why the game adds nothing (planted regions count toward the maxima)

> **Trap: Water labelled with a land biome pulls the region's edge under water.** With Deep North water labelled to 16 m, the Deep North region's border lies on the seabed and its HeightAvg fails; on a global biome that costs nothing (Deep North hosts only Lantern anyway), but on a Meadows island it takes the host away. The first attempt at labelling every coast to 26 m took the hosts from Birch Meadows and the Mistlands alts and made 123 planted regions out of 21. Label only the biomes whose locations need water, only as deep as they need (Deep North 16 m, Mistlands 1 m), never against a foreign shore, and pull a multi-biome alt's painted water 3 px back from foreign shores.

Other facts you will need:

- In the game's data **Lantern is the only alt biome that takes Deep North** (it takes seven biomes), and **no alt biome takes Ashlands**. A "Deep North alt island" can only be Lantern; an Ashlands alt would need a modded alt biome (Expand World Data), untested with 0.8.1 planting.
- Alt biomes do not change the displayed biome name; they change spawns, vegetation, locations, weather and music, and some add locations of their own (Ruin Black Forest adds 4, Goblin Plains 4, Hut Swamp 3, Death Plains 3, Dark Meadows 1, Fortress Mountain 1). Those add-on locations are placed only inside that alt biome's regions and appear in the log with suffixes such as `_1` and `_sunk`.
- Planting (chapter 2) overrides all of this for the regions you paint, and counts toward the maxima. ProximaMaxi plants 19 of the 32 (23 regions) on natural features and leaves the rest to chance; `altbiomes.py` showed all 32 reaching their minimum in 100% of 200 seeds, and the game agreed.
- Dev console (cheats): `biomeinfo`, `altbioms` (sic), `findbiome <text>`, `findbiometp <text>`, `genloc alt` (re-rolls placement, server). The mod's `bc ab info` reports every legend line's result.

> **Check:** `altbiomes.py <map dir> overlay.png 200` (appendix A) prints the planted regions, the table of every alt biome with its window and how often it `reaches min` over 200 seeds (all 32 at 100.0% on map version 13), and the random hosts. In the game, the world-load log has the `Alt biomes: planted: ...` summary (chapter 2) and one line per alt biome; `bc ab info` reports what the pipeline produced and why each alt biome did or did not place, and `bc_altbiomes here` describes the region you stand in. **If you can only talk**, ask the human to search the log for `Alt biomes:` and to send you every line that has it.

Locations

`ZoneSystem.GenerateLocations` runs once, when the world is created, over every location in the game's list: the `_ZoneSystem` prefab's own 130 plus the `_LocationList_*` add-ons for the Ashlands, Mistlands, Deep North, Hildir, the mountain caves and the 1.0 content. Prioritized locations (bosses, vendors, the progression dungeons) go first with 60,000 attempts each; the rest get 12,000. For each attempt:

1. A candidate **zone** (64 m square) is taken from a random point of the grid above, of one of the location's biomes (`AllPointsAboveSeaLevel` unless the location allows negative altitude), or from around the centre for a centre-first location. A zone that already holds a location is skipped.
2. The zone centre must have the location's **biome area**: `Median` means the same biome at the eight points 64 m around it; `Edge` means not; `Everything` means either.
3. **Six random points** in the zone are tried in turn. Each must pass: distance from the world centre (`m_minDistance`, `m_maxDistance`, the window the game counts as a centre-distance failure, which a new world laid out to its size scales by World Size / 10,000; and `m_minDistanceFromCenter`, `m_maxDistanceFromCenter` where a location sets them, checked further on, not counted, and never scaled), biome (the mod's four-pixel vote), **altitude** (bilinear height less the 30 m water level, within `minAltitude..maxAltitude`; a high world lifts a `maxAltitude` of 1,000 m or more, below), forest (the forest factor within `forestTresholdMin..Max` if `inForest`), **terrain delta** (highest minus

lowest of ten random heights within the exterior radius, within `minTerrainDelta..max`), distance from similar locations (`minDistanceFromSimilar`, and the same for the location's group), and vegetation (the Ashlands lava mask, `minimumVegetation..maximum`).

4. A **unique** location (Haldor, Hildir's camp, the Bog Witch, the big rock clearing, the ancient upgrade station) gets up to its quantity of candidate sites like any other; when a player first reaches one it is spawned and the other candidates are removed. One legal site is therefore enough, and the checker places one and stops.

The mod registers the location map's sites first, so a pinned boss occupies its zone before the random pass, and counts toward its quantity; a pinned site is not checked against any of the rules above (chapter 2). One location per zone means a 64 m grid caps density: a biome of 14 km^2 has at most 3,400 zones, and the goblin towers alone want 100 sites 512 m apart.

When a location cannot fill its quantity the game logs `Failed to place all <name>, placed N out of M` (with the tries and the time after it). When all locations are done it logs one line with `Genloc duration`. Real lines, from the build of map version 13:

```
[Warning: Unity Log] 09/23/2026 12:47:31: Failed to place all Bonemass, placed 4 out of 5 with
» 0 tries in 00:00:00.8604500
[Info   : Unity Log] 09/23/2026 12:47:48: Loading: Done. Total seconds sinze ZNet start:
» 21.498534, Genloc duration: 18089.898 ms, (Total genloc time: 00:00:18.3660060, locations:
» 183, iterations: 0)
```

("sinze" is the game's own spelling.) The checker prints the same count with the reason counts by category: `zoneTaken`, `area`, `centreDist`, `biome`, `altitude`, `terrainDelta`, `similar`, `notSimilar`, `forest`, `vegetation`; a line that starts with `!` is short of its quantity, and `P` marks a prioritized location.

Worked example: why Hildir's cavern failed on the Amount 1 map. The checker's lines for map version 11, which was authored at Amount 1 and built as `Proxima`, replayed against the game's log (`--log`, real):

```
! Dragonqueen                          1/3    P Mountain                 game: 1/3
» [zoneTaken 22, area 19062, centreDist 11403, biome 50, altitude 234043]
! Hildir_cave                          0/3    P Mountain                 game: 0/3
» [zoneTaken 295, area 19129, centreDist 10704, biome 12, altitude 232740]
! StoneTowerRuins04                    0/50     Mountain                 game: 0/50
» [zoneTaken 142, biome 2615, altitude 68533]
! MountainCave02                      12/120    Mountain                 game: 12/120
» [zoneTaken 3217, biome 2088, altitude 50268, similar 270]
```

Read the reasons: `altitude` dominates every line, by ten to one. The cavern wants 200 m and the tower ruins 150 m; at Amount 1 nothing stands higher than 140 m, so no attempt can pass. The game agreed exactly (`game: 0/3`). The fix was not in the shapes: map version 12 went to Amount 2 (chapter 2), and on map version 13 the checker places all four in full (`Dragonqueen 3/3`, `Hildir_cave 3/3`, `StoneTowerRuins04 50/50`, `MountainCave02 120/120`). Moder's one site on the Amount 1 map is the pin, which is not checked.

The reason that dominates	What it means for the map
`altitude`	the biome sits at the wrong height: raise or lower it, or change the Amount
`area`	the biome is too fragmented for a `Median` location: make bigger patches
`biome`	the random point fell on another biome: usually harmless, look at the others
`terrainDelta`	the ground is too steep or too flat for the location's window
`similar`	the group's spacing: not enough area for the quantity
`vegetation`	the lava map: too much lava, or too little (chapter 2)
`zoneTaken`	other locations hold the zones: area again

What the important locations want

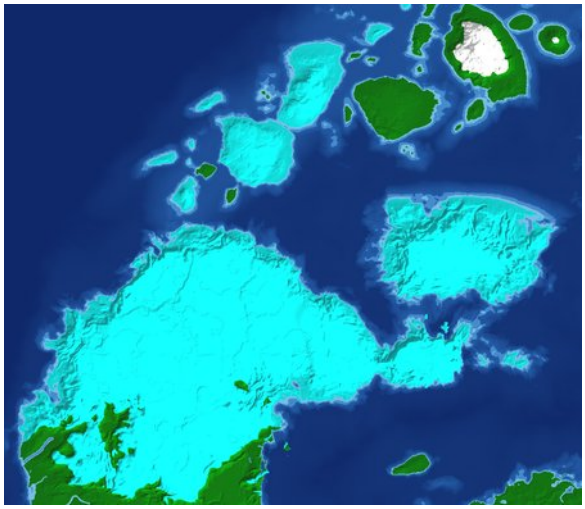
Numbers from the game data (`locations/locations_1.0.15.json`); altitude is metres above the water, delta is the terrain delta window, distances in metres. Appendix C has the full table of the 170 enabled location types.

Location	Biome, area	Qty	Altitude	Delta	Other
StartTemple	Meadows, Median	1	3..1000	0..3	prioritized; forest factor 1..5
Eikthyrnir	Meadows, Median	3	1..1000	0..3	prioritized
GDKing (the Elder)	BlackForest, Median	4	1..1000	0..5	3,000 m from similar
Bonemass	Swamp, Median	5	**0..2**	0..4	3,000 m from similar: flat bog at the water
Dragonqueen (Moder)	Mountain, Median	3	**150..500**	0..4	3,000 m from similar
GoblinKing (Yagluth)	Plains, Median	4	1..1000	0..4	3,000 m from similar
Mistlands_DvergrBossEntrance1 (the Queen)	Mistlands, Median	5	1..20	0..40	group DvergrBoss, 2,048 m
FaderLocation (the Ashlands boss)	AshLands, Median	3	1..300	0..40	lava mask at most 0.1
DN_Bossroom (the Deep North boss)	DeepNorth, Everything	3	**80..5000**	**0..6**	1,024 m from similar: 80 m up AND level over 32 m
Vendor_BlackForest (Haldor)	BlackForest, Median	10, unique	1..1000	0..2	512 m from similar
Hildir_camp	Meadows, Everything	10, unique	1..1000	0..2	1,000 m from similar
Hildir_crypt	BlackForest, Median	3	1..1000	0..2	3,000 m
Hildir_cave (the Howling Cavern)	Mountain, Median	3	**200..5000**	0..40	2,000 m
Hildir_plainsfortress	Plains, Median	3	8..1000	0..4	3,000 m
BogWitch_Camp	Swamp, Median	10, unique	1..1000	0..2	1,000 m
StoneTowerRuins04 / 05 (mountain ruins)	Mountain, Everything	50 each	**150..1000**	6..40	group Mountainruin, 128 m
MountainCave02 (frost caves)	Mountain, Everything	120	**100..5000**	0..40	group mountaincaves, 200 m
Runestone_Mountains	Mountain, Everything	100	100..1000	0..3	128 m
FrozenShip01..03_DN	DeepNorth, Everything	50 each	**-15..-5**	0..100	**8,000..9,750 m from the centre**; snaps to water
ShipWreck01_DN	DeepNorth, Everything	170	-0.5..0.5	0..2	the shoreline itself
Mistlands_Harbour1	Mistlands, **Edge**	100	**-1..-0.25**	2..20	snaps to water
Mistlands_Viaduct1	Mistlands, Median	100	-1..15	2..40	group Harbour, 128 m
Mistlands_DvergrTownEntrance1 (infested mines)	Mistlands, Median	120	12..1000	5..40	prioritized
Mistlands_DvergrTownEntrance2	Mistlands, Median	120	1..20	0..40	prioritized

Location	Biome, area	Qty	Altitude	Delta	Other
Mistlands_Giant1 / Giant2	Mistlands, Median	250 / 85	-1..1000	0..3 / 0..4	group Giant, 180 / 256 m
SunkenCrypt4	Swamp, Median	175	**0..2**	0..4	prioritized; 64 m
Crypt2 / 3 / 4 (burial chambers)	BlackForest, Everything	200 each	1..1000	0..2	128 m
TrollCave02	BlackForest, Median	200	3..1000	**5..10**	256 m: wants a slope
GoblinCamp2	Plains, Everything	200	2..1000	0..3	250 m
StoneTower1 / StoneTower3 (goblin towers)	Plains, Everything	50 each	1..1000	0..3	group Goblintower, **512 m**
StoneHenge1..5	Plains, Everything	5 each	5..1000	0..3	group Stonehenge, **1,000 m**
TarPit1	Plains, Median	100	5..60	0..2	128 m
LeviathanLava	AshLands, Everything	100	10..1000	0..10	**lava mask 0.6..1.0**
CharredFortress	AshLands, Everything	20	20..1000	0..10	prioritized; lava mask at most 0.1
NorthVillage	DeepNorth, Median	135	2..1000	0..3	70 to 85 m from similar (huts cluster)
MorkBorg	DeepNorth, Everything	40	30..10000	0..300	275 m

Read the table as design requirements:

- **Mountains** must reach 200 m above the water somewhere (the cavern) and 150 m across a wide area (100 tower ruins 128 m apart, Moder's altar). Hence Heightmap Amount 2 (a 340 m ceiling; Amount 1 stops at 140 m, chapter 2) and peaks of 250 m.
- **Deep North** must have flat ground 80 m up (the boss room: three sites 1 km apart, level within 6 m over 32 m) and its water must be Deep North to 15 m depth in the 8 to 9.75 km ring (the frozen ships). ProximaMaxi's icefields are flat-topped plateaus at 105 to 125 m.



12e-deepnorth.png



12d-mistlands.png

Left: the Deep North of map version 12 (cyan), where the icefields first appeared: flat-topped plateaus for the boss room, and the isles of the cold archipelago. Right: the eastern Mistlands chain of map version 12 (grey), cut into islands by straits, so that it has a long shore for the harbours and the low ground the mine entrances want.

- **Swamp** must be nearly flat and nearly at the waterline: Bonemass and the sunken crypts want 0 to 2 m. ProximaMaxi's bog is 1.5 to 5.5 m (swamp mean 5.8 m); Bonemass places 4 of 5, and one is enough (the five want 3 km between them).
- **Mistlands** must have lowland shore under 20 m (harbours, viaducts, lighthouses, mine entrances) as well as its high ground; ProximaMaxi's lowland valleys meet the sea as shore, sunk 10 m.
- **Plains** are quantity-limited by area: 100 goblin towers 512 m apart need about 26 km^2 of plains and ProximaMaxi has 14 km^2. Shortfalls in the Goblintower, Stonehenge and Giant groups are area, not bugs.
- **Ashlands** need lava for the leviathans and lava-free ground for the fortresses.
- **Black Forest** needs slopes (troll caves want a 5 to 10 m delta) and flat forest (crypts, ruins) both.

> **Rule: Layout is radial.** Vanilla places bosses and dungeons by distance from the world centre (Meadows at the centre, Mistlands, Ashlands and Deep North at the rim), and several locations carry explicit distance windows (the frozen ships 8,000 to 9,750 m; the big rock clearing at least 1,000 m; the ancient upgrade station at least 500 m). A map that puts the Meadows at radius 6,000+ has nowhere legal for meadow structures. ProximaMaxi: spawn 2,203 m out, the Elder 916 m, Haldor 4,284 m, Moder 6,015 m, Yagluth 6,574 m, Hildir 6,629 m, Bonemass 7,871 m, the Queen 7,976 m.

The distances above are for a world of vanilla's size. A new world laid out to its own World Size (chapter 10) keeps most of the layout at its own scale: the window each location keeps in `m_minDistance` and `m_maxDistance`, and the game's own biome bands and rings where no biome map decides, are multiplied by World Size / 10,000, so a 15,000 m world asks for Hildir's cavern (1,000 to 8,000 m in vanilla) 1,500 to 12,000 m out. Four windows live in the other pair of fields, `m_minDistanceFromCenter` and `m_maxDistanceFromCenter`, which Better Continents does not scale (its `WorldSizePatch` scales only the first pair, and the game's `ZoneSystem` checks the second as it is): the frozen ships still want Deep North water 8,000 to 9,750 m from the centre in a world of any size, the ancient upgrade station at least 500 m, the big rock clearing at least 1,000 m (and its scaled minimum, where that is larger), and the combat ruin (`CombatRuin01`) at least 1,500 m. Design the rest of the radial layout in fractions of the

radius and it holds at any size; put the frozen ships' Deep North water in that fixed ring. (With Expand World Data installed its location data is already in fractions of the radius, and Better Continents does not scale it twice.)

> **Check:** `locations.py <map dir>` (appendix A) ends with a `VERDICT` and a line of every progression and quest site with its count (`progression and quest sites: StartTemple 1/1, Eikthyrnir 3/3, GDKing 4/4, Bonemass 4/5, ...` on map version 13). After the human has built the world, `locations.py <map dir> --log <copy of LogOutput.log>` puts the game's own count on every line. **If you can run commands but not Python's toolkit**, count the game's failures with `grep -c "Failed to place all" LogOutput.log` (34 on map version 13). **If you can only talk**, ask the human to search the log for `Failed to place all` and to send you every line; any boss, Haldor, Hildir or the Bog Witch in that list is a failed map.

The game's fixed circles: the Deep North's look and sea, the Ashlands sea

The game's shaders draw three things by fixed circles of their own, whatever the world's size or its biome map. Better Continents 0.10.5 draws the sea itself where the world's sea differs from the game's (chapter 13), so the last two follow your maps; the first stays the game's:

- **The Deep North's ground.** The terrain shader marks the Deep North with a circle centred at (0, -4,000), with a radius of about 12,000 m. Outside it farmland is not drawn as soil, mountain ground is raised as deep snow (2.1 m) and the shore band sits 2 m lower.
- **The Deep North's calm sea.** The game's water shader draws the calm sea by the same circle, 12,000 m from (0, -4,000), fading over 200 m. **Since Better Continents 0.10.5** its own water draws the calm sea at the **world's** Deep North (`IsDeepnorth`: your biome map's Deep North, or the ring laid out to the world's size), fading over the same 200 m, and the waves that boats and swimmers float on (`DeepNorthWaveFade`, `CalcWave`) follow the same field, so drawn and felt agree. 0.10.3 and 0.10.4 kept both at the game's circle; 0.8.0 to 0.10.2 moved only the felt waves, so the two parted (trap 55).
- **The Ashlands sea's red.** The game's water shader colours the Ashlands sea by its own ring: past 12,000 m (plus 100 m x the game's `WorldAngle`) from (0, +4,000), with a ragged edge 25 m wide. The red follows the world's own hot sea, by the test that heats a swimmer and burns a ship (`GetAshlandsOceanGradient` above 0: heat > 0 on a world with a heat map, else the ring laid out to the world's size): since 0.10.5 per pixel, with the game's 25 m edge, by Better Continents' own water; in 0.10.3 and 0.10.4 (and since then wherever a player's game draws the game's water) by colouring each 64 m patch of water, its edge in 64 m steps softened by the share of each patch that is hot. Only the sea counts, so lava on land does not tint the water at its shore.

At the default size the outside of the Deep North circle is the north, where vanilla's Deep North lies. In a big world it takes in places far from the north too: the terrain shader's Deep North look is the game's own rule there, so lay out a big world knowing where the circle is. The calm sea and the red sea are yours to draw (chapter 13): Deep North over the sea makes it calm, heat over the sea makes it red, with no falloff past the heat map's pixels (chapter 2). A world with no Deep North over water has waves everywhere.

A tall world: what the game assumes about height

Valheim 1.0 was made for land under about 400 m, and a few of its rules hold an absolute height. A world whose heightmap is read at an Amount above 5 is a **high world** (chapter 2), and Better Continents then applies its own versions of those rules:

- **Inside a dungeon** means more than 3,000 m over the ground below, not just higher than 3,000 m: terrain, weather and building keep working on a mountain, and dungeons, which stand 5,000 m above their entrance, still count as inside.

- **The ground is found at any height**: logging in at the logout point, items, tombstones, a zone's plants, creature spawn points and grass. The game's grass finds no ground above 500 m of height, and the rest none above 2,000 to 6,000 m, where its searches start.
- **Plants, creatures, locations and grass may stand at any height**: their upper altitude limits of 1,000 m or more are lifted; lower ones stay, such as the tree line at 280 m.
- Parts of locations that switch off over 10,000 m stay on; the AI can walk at any height; random events and the Deep North weather follow the player's position on the ground; a new player's Valkyrie starts over the start point.

A world read at 5 or less runs the game's own rules, so an older world plays as before. That is `High Terrain = Auto`, the default (`[02 BetterContinents.Heightmap]`, a setting a new world takes and keeps, chapter 2): On gives every Better Continents world these rules, whatever its heights, and Off none, which costs a world with high ground its grass above 500 m, its plants, creatures and locations above about 1,000 m, its terrain, weather and building above 3,000 m (the game takes that for the inside of a dungeon), and any ground found above 6,000 m.

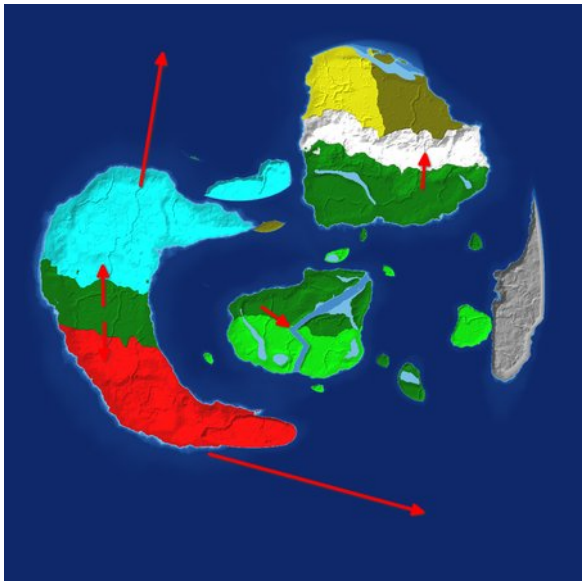
> **Rule: A location window under 1,000 m keeps its top, at any Amount.** Moder's altar still wants Mountain 150 to 500 m above the water, level within 4 m; the Ashlands boss 1 to 300 m; the tar pits 5 to 60 m; the Queen's entrance 1 to 20 m; and the tree line stays at 280 m. A range of tall peaks therefore needs ground of its biome inside those windows too (shoulders, shelves, foothills), or a pin (chapter 2). `locations.py` keeps every upper limit as the game data gives it, so in a high world it under-counts the locations above 1,000 m (appendix A).

What stays the game's own at any height:

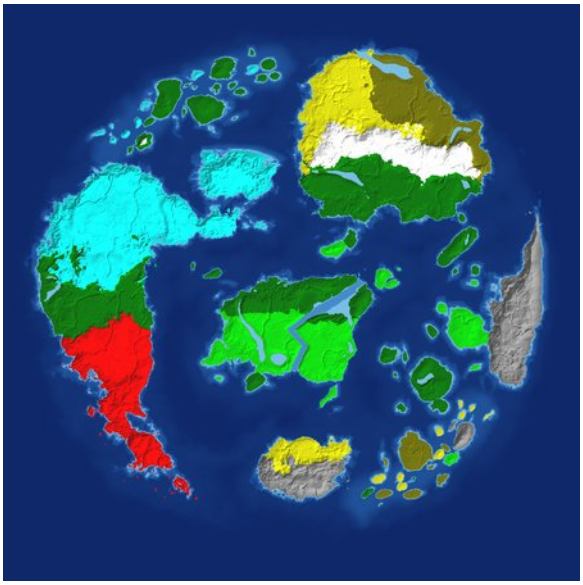
- Fog hides terrain beyond roughly 0.6 to 1.4 km on a clear day (60 to 140 m in rain), and distant terrain is drawn only out to 1.2 km: from a summit the ground far below is lost in the fog.
- Clouds stay at 24 to 424 m of height, with the horizon cloud band up to 2,000 m: from above you look down on them.
- Munin does not land on ground over 2,000 m.
- The map's height shading comes in steps of 8 m over 8 km and 16 m over 16 km.

4 What "reads natural" means

The human's standing goal: "we want a realistic feeling map", "official-feeling". This chapter is what that meant in practice on ProximaMaxi: the tells that broke the illusion, each found in a picture the human looked at, and how each was fixed. The pictures are the evidence.



`01-parked-before.png`



`06-archipelagos.png`

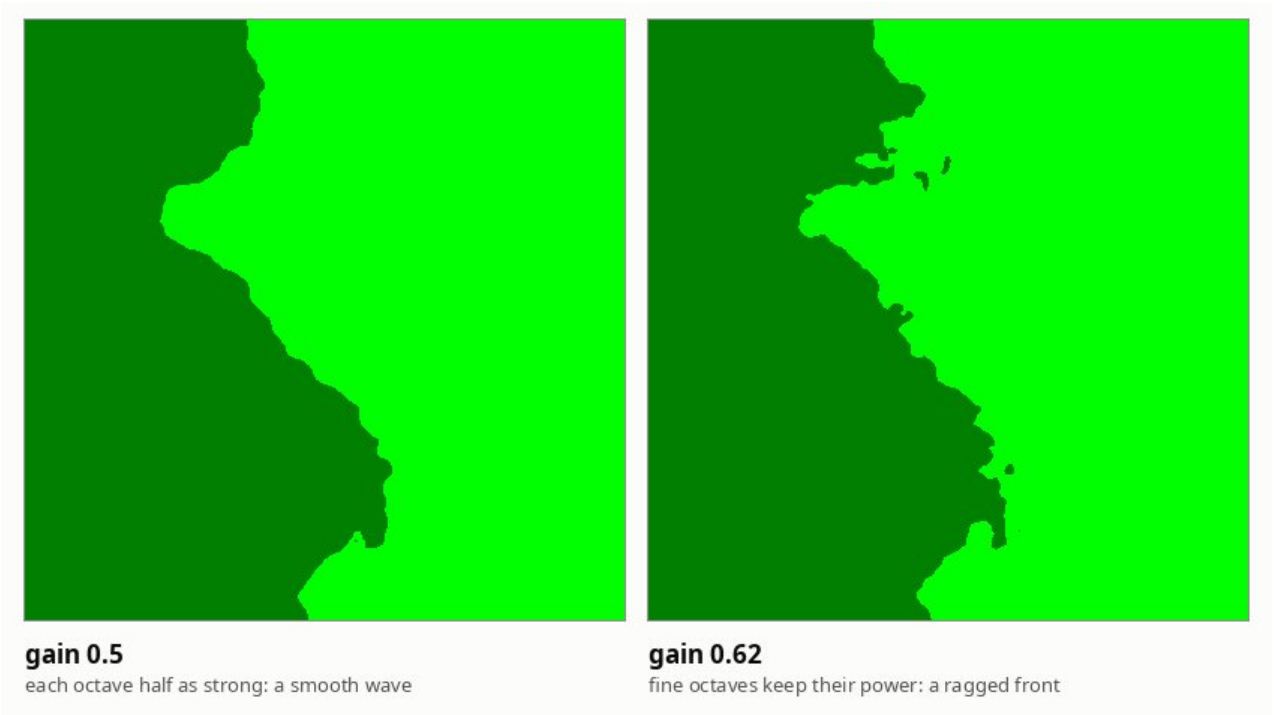
Left: version 01, the layout as first drawn: elliptical landmasses, biome bands as stripes, round snow discs, islands parked in a row (a warp bug had displaced them all by the same vector). Right: version 06, the first the human liked ("i like it 06"): the same layout with fractal coasts, ragged fronts, a wandering crest and two archipelagos. Everything after 06 was refinement.

The tells, and the fixes

1. **Biome borders as ruled lines or latitude stripes.** A front drawn with ordinary fractal noise (gain 0.5) is a smooth wave at every scale. Use rough noise (gain 0.62, so that the fine octaves keep real power), give it a second-order domain warp so that the lobes hook into tongues instead of rippling, and let fronts follow terrain: Deep North climbs ridges, forest sits in valleys. ProximaMaxi's `front_offset()` combines a 4.5 km drift, 1.5 km hooked lobes and 70 m raggedness:

```
frontn = FBM(6, 4, SEED + 18, gain=0.62)   # rough: the fine octaves keep their power

def front_offset(u, v):           # how far a front is pushed off its nominal line here
    # a second-order warp, so that the lobes hook into tongues
    qu = (warpd(u * 3.0 + 0.1, v * 3.0) - 0.5) * 0.10
    qv = (warpd(v * 3.0, u * 3.0 + 0.1) - 0.5) * 0.10
    return ((front1(u * 1.5, v * 1.5) - 0.5) * 0.20              # the gentle drift
            + (frontn((u + qu) * 2.4 + 0.3, (v + qv) * 2.4 + 0.8) - 0.5) * 0.40)
```



Tell 1: the same front drawn with gen.py's noise (the same lattices, six octaves) at gain 0.5 and at gain 0.62, computed for this page. Notice that the left edge is a smooth wave at every scale, and that the right one is ragged down to the pixel, with outliers of forest beyond the line, as a real woodland edge has.

2. **Heights that step at biome borders.** Blend each biome's terrain across the front rather than switching at it.
3. **A mountain band of constant width, and round white snow discs.** Use a wandering crest (a polyline spine with spurs and side valleys off it), radial ridges on isolated peaks, and label snow from the FINISHED height so that it climbs valleys and thins on spurs instead of drawing a circle.
4. **Smooth oval coastlines and uniform shelves.** Coast noise must have power at 200 m to 1 km: three scales (a gulf, bays and headlands, and a fine roughness so that the waterline is never smooth), plus a domain warp. Every landmass field must be on ONE shared scale so that islands get the same crinkle as continents (measured per landmass, a 370 m island got a tenth of a continent's detail).
5. **Evenly scattered round island dots.** Rotate and elongate islands, give them satellites, put chains on a shared shelf, throw skerries off cliffed coasts.
6. **Every coast a gentle beach.** Windward coasts and coasts under high ground are cliffs.
7. **Straight rivers, or no rivers.** With a heightmap the game draws no rivers, so carve them, and carve them where water would actually drain: ProximaMaxi computes the drainage network on the terrain (Priority-Flood on a 20 m grid, accumulation, streams from 0.2 km^2, full rivers at 15 km^2) and cuts every stream downhill into a bigger river and out to sea, with lakes where rivers cross low ground and in natural basins. The starter river beside spawn is a hand-drawn Catmull-Rom meander because the human said the first one "looks too sharp and unnatural".



09d-spawn-closeup.png



10d-spawn-river.png

Tell 7: the starter river beside the spawn, map version 09 (left) and map version 10 (right). Notice the straight reaches and hard bends of the first, which the human called "too sharp and unnatural", and the meander that replaced it, with streams joining it downhill.

8. **Ground that reads flat.** Kilometre-scale ridges plus footstep hummocks still read as flat. The hills people notice are at 100 to 300 m wavelength, about 4 m RMS in meadows. Fade relief in within about 100 m of the shore, not over a 500 m beach ramp, or every coast is a tilted plain. In gen.py:

```
ROLL = {"deepnorth": 3.0, "ashlands": 3.5, "mistlands": 0.0, "mountain": 4.0,
        "bforest": 4.5, "plains": 2.5, "meadows": 3.5, "swamp": 0.0}    # RMS metres
ROLL_K = 7.680                     # 1 / the noise's standard deviation (about 0.13)
roll = (rolln(u * 16.0, v * 16.0) - 0.5) * ROLL_K    # unit RMS, 220 m and down
# each biome's height then adds ROLL[biome] * roll, times `inland`,
# which rises from 0 at the shore
```

9. **A biome border that steps in 64 m blocks.** At Biome precision 0, the vanilla value, a 64 m terrain zone takes its ground, grass, vegetation and spawns from its four corners, so a border painted to the pixel still shows on the zone grid. A new world can raise Biome precision to 31 (cells of 2 m); an older world stops at 5 (chapter 9). It changes what a point looks like and grows, never the terrain's height or where a location may go.



12i-start-coast.png



13i-start-coast.png

The spawn coast in map version 12 (left) and map version 13 (right), the same crop. The human said of 12 "the world reads too flat"; 13 adds rolling relief at 220 m and below and a 210 m beach with the relief starting 125 m inland.

The tells at a glance

Tell	Don't	Do	See it in a preview
1. Fronts	ruled lines, latitude stripes, smooth waves	rough noise (gain 0.62), a second-order warp, fronts that follow terrain	does a border look drawn with a ruler or a compass?
2. Steps at fronts	each biome's terrain switched at the border	relief blended across the front	a cliff or a terrace exactly on a colour change
3. Mountains	a band of constant width, round white discs	a wandering crest with spurs; snow from the finished height	a white circle or a white stripe of even width
4. Coasts	smooth ovals, one shelf width everywhere	noise at 200 m to 1 km plus fine roughness; one scale for every landmass	an island that looks like an egg
5. Islands	round dots, evenly spaced, in a row	rotated, elongated, in chains on a shelf, with satellites and skerries	a row of dots, or dots on a grid
6. Beaches	a gentle beach on every coast	cliffs on windward coasts and under high ground	the same pale shelf all round every island
7. Rivers	straight lines, or none	drainage: streams downhill into rivers, out to sea; lakes in basins	a river that runs uphill, or a straight blue line
8. Relief	kilometre ridges and footstep bumps only	hills at 100 to 300 m wavelength, about 4 m RMS in meadows	a lowland with no shading at the preview's scale
9. Zone steps	(at Biome precision 0) borders that step in 64 m blocks	Biome precision above 0 where it matters (chapter 9)	only in the game: a border that stairs in squares

> **Check:** a 30-second look at every preview, before you show it: (1) no ruler or compass lines; (2) no white disc; (3) no egg-shaped island and no row of dots; (4) every river runs to the sea or a lake; (5) magenta anywhere means dry land labelled Ocean (`preview.py`'s bug flag): stop and fix it. Then `relief.py` for tells 6 and 8 (numbers below). **If you cannot see images**, ask the human these five questions about the picture, by name, and wait for the answers.

A map does not have to author everything to read natural. On a new world Heightmap Alpha (chapter 10) fades a drawn landmass into the game's own terrain, which already reads natural at the scales you did not draw; only the part the human cares about needs the work above.

A tall world (an Amount above 5, chapter 2) has two tells of its own, which ProximaMaxi never met:

- **Stairs on gentle slopes.** A 16-bit step is 0.12 m at Amount 40 and 0.25 m at 81, and a gentle slope shows each one. Use the smallest Amount that holds the highest peak, and fine heights where the steps still show (chapter 2).
- **A summit view that ends in fog.** Fog hides terrain beyond roughly 0.6 to 1.4 km on a clear day and distant terrain is drawn only to 1.2 km, so from a high summit the ground far below is lost, and the clouds (24 to 424 m) lie under you (chapter 3). The previews show a whole range at once; on the ground the player sees a kilometre or so of it.

Relief that reads, in numbers

`relief.py` prints, per biome, the RMS height change over 50 m (the hills you walk over) and over 250 m (the hills you see), and the mean height in 50 m bands out from the water. The numbers behind "it reads flat":

	Meadows, 50 m	Meadows, 250 m	Coast, first 50 m	50 to 100 m	100 to 150 m
Map version 12 (too flat)	2.9 m	7.4 m	3.4 m	6.8 m	10.4 m
Map version 13 (approved)	4.0 m	9.1 m	5.5 m	11.8 m	15.6 m

Worked example: "the world reads too flat". The human said it of map version 12. The numbers above agreed: Meadows 2.9 m over 50 m and a coast that rose 6.8 m in its first 100 m. Map version 13 added `ROLL` (rolling relief at 220 m and below) and cut the beach ramp to about 210 m with the relief starting about 125 m inland; `relief.py` then read 4.0 m and 11.8 m, and the human said "looks good on my end". The command, and the header and Meadows line of its output on map version 13 (real; the full output is in chapter 8):

```
(ulimit -v 6000000; nice -n 10 python3 relief.py history/out-13-2048)
biome        rms dh 50 m   rms dh 250 m   (metres)
Meadows          4.02         9.07
```

Map version 13's `ROLL` (RMS metres per biome): Deep North 3, Ashlands 3.5, Mountain 4, Black Forest 4.5, Plains 2.5, Meadows 3.5, Mistlands 3 inside its own relief, the bog 0.5. Re-measure after touching `ROLL`, the beach ramp or any biome's relief. If the human still finds it flat, raise Meadows to 4.5 and Black Forest to 5.5 and re-run `locations.py`; if the plains get too rough for the goblin towers (delta 0 to 3), lower Plains to 2.0.

> **Trap:** A value-noise FBM's output is compressed round 0.5 (its standard deviation is about 0.13, not 0.29). `(noise - 0.5) * 2 * amplitude` is therefore worth about a quarter of the amplitude you asked for. The first relief pass "looked like nothing" for this reason. Scale by 1 / std (`ROLL_K = 7.68`) before treating amplitude as metres, and measure the result with `relief.py` rather than trusting the constant.

The reference world

Wubarrk considers **Era02**, a real pre-1.0 Better Continents world, correct: Amount 1.0, 33.8% of the image land, land averaging 33.6 m. Its high plateau and abrupt continental drop-off are artefacts of how it was drawn, not targets. ProximaMaxi is 27.7% land, with the per-biome means in chapter 8.

Where the human's taste showed

Things the human asked for, verbatim, and what they became: "the river looks too sharp and unnatural" (the meander redraw); "make various alt deep north islands, move wolf mountain" (the Deep North isles of the cold archipelago, Lantern on all of them, Wolf Mountain on the west massif's twin peaks with random Wolf Mountain switched off); "we def want some mushroom meadows" (Mushroom planted three times, the game's maximum: the whole meadow east of the starter river, bounded by river, forest and coast instead of a circle, plus two islands); "the Ashlands is too small, add island chains along the edges" (the arm fattened, burning isles off its tip, four rim chains); "the shoreline is a little too flat, the world reads too flat" (map version 13). Designed for a dark realm: the dangerous alt variants got the landmarks, the quiet ones were left to chance, and Hildir's island stayed plain.

If you can only talk, the human's words are your measurements: ask what they see ("Does the coast north of the spawn look smooth or ragged?"), name the tell from the table above, and propose the one change that fixes it. **If you work with a human who paints the maps by hand** in an image editor instead of a generator, the same tells apply: a rough brush or a noise filter for coasts and fronts, never a blur on the biome map (chapter 2: blended colours snap to fringes), and hard edges between biome colours.

GPL-3.0 (Wubarrk, 2026-10-26); see the License page

5 The generator, as a pattern

`generator/gen.py` is 2,300 lines and specific to ProximaMaxi, but it is built from a small set of parts that any map generator for this game needs. Learn the parts; then change the map by changing the constants and selectors, or write your own generator on the same skeleton. Run it with `python3 gen.py 2048 <out dir>` (about 40 s on all cores at 2048; use 512 for a quick look). It needs only the standard library.

Skeleton

```
value noise  ->  FBM(octaves, base frequency, seed, gain): tileable, wrapping lattices
domain warp  ->  (u, v) -> (wu, wv); moderate, leashed per landmass, anchored per island
landmass fields  ->  one shared scale: crescent, spawn continent, second continent, Mistlands
                     chain, southern landmass, islands, satellites, archipelagos, skerries
coast noise  ->  three scales added to every field; the leash contour carries it too
biome fronts ->  front_offset(): drift + hooked lobes + rough edge; snow from the
                 finished height
heights      ->  per-biome relief blended across fronts, beach and inland ramps, cliffs,
                 massifs, a spine with spurs, Deep North icefields, Mistlands valleys, ROLL
per pixel    ->  evaluate(u, v) -> (height value, biome, heat, forest, location region,
                 alt plant)
parallel     ->  multiprocessing over row bands -> arrays
post passes  ->  carve_rivers() (drainage), label_shallows(), write_altbiomemap(),
                 location sites
write        ->  png.py: 16-bit greyscale and 8-bit RGB, plus the legends
                 (png_extras.py: the Amount record; a grey + alpha file for Heightmap Alpha)
```

The parts

Value noise and FBM. `FBM(octaves, base, seed, gain=0.5)` sums bilinear samples of random lattices whose resolution doubles per octave. The lattices wrap in both axes, so callers can sample past [0, 1] (the warp pushes coordinates negative) and no octave introduces a seam. Named instances carry their purpose: `coast`, `coast2`, `detail`, `ridge`, `riv1..3`, `lakes`, `front1`, `front2`, `frontn` (gain 0.62), `cliffn`, `spurn`, `skn`, `coastn`, `lavan`, `rolln`. One seed constant, `SEED = 20260922`, offsets them all, so a map is reproducible.

Domain warp, leashed and anchored. Warped coordinates make every outline non-elliptical, but a warp big enough to be interesting evaluates a continent's field far from where the continent is and strands its biomes across the map. Each landmass field is capped against its own UNWARPED ellipse plus slack (the "leash"), and the leash contour carries the coastal noise, because wherever it clips it IS the coastline. Small features (islands) subtract the warp at their own centre ("anchored"), which is what fixed the parked-islands bug of version 01: unanchored, every island in an archipelago was displaced by the same warp vector and landed in a row.

One scale for every landmass. Fields are measured in the crescent's units (about 4.2 km), not each in its own radii, so that coastal noise and beach ramps, added in those units, come out the same size on a 370 m island and a 4 km continent.

Biome fronts. `front_offset(u, v)` displaces a nominal front line; fronts follow terrain where the biome should (snow from the finished height, forest into valleys). Biome is decided from a handful of scalar fields per landmass (latitude on the crescent, distance along the spine, the side of a divide on the second continent), each offset by the front noise.

Heights. Every biome has a base relief, a `CLIFF` weight (how much of its coast is cliff) and a `ROLL` RMS, blended across the fronts. Ramps: `beach = sstep(0, 0.05 * (1 - 0.8 * cliff), land)`, about 210 m, is the platform the shore rises on; `inland = sstep(0, 0.03 ..., land)`, about 125 m, is where the relief and the hummocks fade in. Special terrain: `MASSIFS` (isolated peaks with radial ridges), `SPINE` (the range crest as a polyline; `spine_query()` gives the distance along and across it, spurs from

`spurn`), `dn_relief()` (flat-topped icefields 105 to 125 m for the Deep North boss room), `mist_relief()` (relief plus lowland valleys sunk 10 m so that the harbours have shore). The `H(metres)` helper converts metres to a pixel value through `mapscale`.

Islands. `ISLANDS` entries are (x, z, rx, rz, kind); `add_satellites()` adds smaller companions cleared of their parent; `add_archipelagos()` lays chains along polylines (`ARCHIPELAGOS`, with a size factor for the rim chains) and buckets them into a grid (`ISL_GRID`) so that `evaluate()` only tests nearby islands. Every island gets its own rotated, elongated, noise-shaped outline (`island_shape`).

evaluate(u, v, dbg=None). The whole world at one point: the heightmap value, the biome, the heat, the forest density, the location-search region and the alt plant. Pass a dict as `dbg` to get every intermediate value back:

```
cd generator && python3 -c "import gen; d = {}; print(gen.evaluate(0.495, 0.605, d)); print(d)"
```

probes one point without generating anything. This is how you debug "why is this pixel Plains".

Rivers by drainage. `carve_rivers(hm, bc)` averages the height onto a 20 m grid with half a metre of noise (so that the flood's tie-break wanders instead of running along the grid's eight angles), runs Priority-Flood from the sea inward to get a downhill direction for every cell and the fill level of every basin, accumulates drainage area, starts streams at 0.2 km^2, widens to full rivers at 15 km^2, and stamps the channels into the heightmap (`river_profile`). Basins whose fill level is above the terrain become lakes. Map version 13: 65 mouths, 144 reaches, 43 km of water, 16 lakes. The return value is how far each pixel was cut, so that location sites keep off the banks.

Shallows with a biome. `label_shallows(hm, bc)` floods outward from Deep North and Mistlands land into the sea, giving water down to 16 m and 1 m respectively the land's biome, never within 1 px of another biome's shore, and never heated water.

Alt planting. `ALT_PLANTS` is a list of (alt biome, base biome, selector): `island k`, `massif k`, `landmass name`, `patch (landmass, cx, cz, r)` with a ragged edge, `eastbank` (the whole meadow east of the starter river), `mistseg k` (one island of the Mistlands chain, bounded by its straits), `isles` (every island of a kind in a box). `alt_plant()` evaluates them per pixel; `write_altbiomemap()` keeps each plant's largest connected piece (specks would become separate one-point regions), grows whole-feature plants 2 px onto other biomes and the sea (so that the game plants the region whole, with no fringe left behind), keeps patches 2 px inside their own biome (a spit or a forest tongue is a lone point on the game's grid, and painting it made a one-point planted region that used up a slot), pulls a multi-biome alt's painted water 3 px back from foreign shores, and writes the legend.

Location sites. `LOCATIONS` holds the nine pinned sites as (name, colour, region, x, z) in world fractions; after generation the nearest pixel of the wanted region that no river cut into (3 px margin) is chosen and painted as a 3 x 3 block, and `locationmap.txt` is written.

Heat and lava. Heat is keyed to the crescent's Ashlands arm and the burning isles as a geometric ramp, extended into the sea, and asserted zero elsewhere. Lava: `lavan` pools at 11 cycles, about a tenth of the Ashlands, the ground let down into them, plus streams down the tributary channels, kept off the beach.

A minimal generator you can copy

`gen.py` is too big to copy whole into a new project. This one is the same skeleton in about 120 lines: value noise and FBM, a domain warp, one landmass with a rough coast, a beach and inland ramp, rolling hills, a crest, snow from the finished height, a ragged forest front, a pinned start temple, the clipping guard and the legends. It writes a complete map set in the formats chapter 2 gives (plain PNGs, exact legend colours); for this page it was checked with `check.py` and chapter 2's `pnginfo.py`, not built in the game. Save it as `generator/mini_gen.py`, beside `png.py` and `mapscale.py`, which it imports; it needs nothing else.

```
"""mini_gen.py: one island as a complete Better Continents map set, built like gen.py.

    python3 mini_gen.py <size> <out dir>      (in generator/, beside png.py and mapscale.py)

Standard library only. Every height is metres above the water, made a pixel by H() at the
Amount in mapscale.py (the one place it lives). Default world size: the square is 21,000 m.
"""
import math, os, random, sys
import mapscale, png

SIZE, OUT = int(sys.argv[1]), sys.argv[2]
SIDE = 21000.0                      # 2 x (World Size 10,000 + Edge Size 500)
A = mapscale.AMOUNT                 # 2.0 in this workspace; the mod's default is 1
PEAK, SNOW = 190.0, 95.0            # metres above the water; PEAK < (A - 0.30) * 200
ROLL, ROLL_K = 3.5, 7.68            # hills, RMS metres; 1 / std of an FBM's output
SEED = 7

def H(m):                           # metres above the water -> heightmap value 0..1
    return 0.30 / A + m / (200.0 * A)

class FBM:                          # value noise: octaves double, lattices wrap (as gen.py)
    def __init__(self, octaves, base, seed, gain=0.5):
        rng, amp, tot, self.layers = random.Random(seed), 1.0, 0.0, []
        for i in range(octaves):
            n = base * 2 ** i
            grid = [[rng.random() for _ in range(n)] for _ in range(n)]
            self.layers.append((grid, n, amp))
            tot += amp; amp *= gain
        self.inv = 1.0 / tot
    def __call__(self, u, v):
        s = 0.0
        for g, n, amp in self.layers:
            x, y = (u % 1.0) * n, (v % 1.0) * n
            x0, y0 = int(x), int(y)
            fx, fy = x - x0, y - y0
            fx, fy = fx * fx * (3 - 2 * fx), fy * fy * (3 - 2 * fy)
            x0, y0 = x0 % n, y0 % n
            x1, y1 = (x0 + 1) % n, (y0 + 1) % n
            a = g[y0][x0] + (g[y0][x1] - g[y0][x0]) * fx
            b = g[y1][x0] + (g[y1][x1] - g[y1][x0]) * fx
            s += (a + (b - a) * fy) * amp
        return s * self.inv

coast = FBM(6, 4, SEED + 1, gain=0.62)   # rough: keeps power at 200 m to 1 km
warp = FBM(4, 3, SEED + 2)
hills = FBM(4, 6, SEED + 3)
front = FBM(5, 4, SEED + 4, gain=0.62)

def sstep(a, b, t):
    t = min(1.0, max(0.0, (t - a) / (b - a)))
    return t * t * (3 - 2 * t)

def seg(px, pz, ax, az, bx, bz):    # distance from a point to a segment
    dx, dz = bx - ax, bz - az
    t = max(0.0, min(1.0, ((px - ax) * dx + (pz - az) * dz) / (dx * dx + dz * dz)))
    return math.hypot(px - ax - t * dx, pz - az - t * dz)

def evaluate(x, z):
    """The world at one point (x east, z north, metres): (altitude, biome, forest 0..1)."""
    u, v = x / SIDE + 0.5, 0.5 - z / SIDE
    wu = u + (warp(u, v) - 0.5) * 0.10                  # domain warp: no ellipses
    wv = v + (warp(v + 0.3, u + 0.7) - 0.5) * 0.10
    r = math.hypot(x - 300.0, z - 200.0) / 5000.0       # an island of about 5 km radius
    f = r + (coast(wu * 2, wv * 2) - 0.5) * 1.0          # the coastline is where f = 1
    inland = (1.0 - f) * 5000.0                          # rough metres from the shore
    if inland <= 0:                                     # sea: a shelf, then down to -60 m
        return max(-60.0, -0.5 + inland / 25.0), 'Ocean', 0.0
    alt = 0.5 + 9.5 * sstep(0, 220, inland)              # beach, then a low platform
```

```
    alt += ROLL * (hills(u * 3, v * 3) - 0.5) * ROLL_K * sstep(40, 160, inland)
    # a crest 3.2 km long, north-east of the centre, with a ragged foot
    d = seg(x, z, 1400.0, 1300.0, 4200.0, 2900.0)
    crest = PEAK * (0.7 + 0.6 * (front(wu * 6, wv * 6) - 0.5))
    alt += crest * sstep(1500, 0, d + (coast(wu * 6, wv * 6) - 0.5) * 900) ** 1.4
    edge = 2800.0 + (front(wu * 4, wv * 4) - 0.5) * 3600.0   # a ragged forest front
    if alt > SNOW + (front(u * 7, v * 7) - 0.5) * 40:    # snow from the FINISHED height
        return alt, 'Mountain', 0.0
    if math.hypot(x, z) > edge:
        return alt, 'BlackForest', 0.55
    return alt, 'Meadows', 0.25

COLOUR = {'Meadows': (0, 255, 0), 'BlackForest': (0, 127, 0), 'Mountain': (255, 255, 255),
          'Ocean': (0, 0, 255)}
os.makedirs(OUT, exist_ok=True)
hm, bm, fm, biome = [], [], [], []
for py in range(SIZE):
    for px in range(SIZE):
        x = (px / (SIZE - 1) - 0.5) * SIDE               # (i / (N - 1) - 0.5) x side
        z = (0.5 - py / (SIZE - 1)) * SIDE               # row 0 is north
        alt, b, forest = evaluate(x, z)
        hm.append(H(alt)); bm.append(COLOUR[b]); biome.append(b)
        fm.append(int(forest * 65535))

clipped = sum(1 for p in hm if p > 1.0)          # the guard: never write clipped ground
assert clipped == 0, '%d pixels above the %.0f m ceiling of Amount %g' % (
    clipped, (A - 0.30) * 200, A)
N2 = SIZE * SIZE
png.write_gray16(f'{OUT}/heightmap.png', SIZE, SIZE,
                 [int(min(1.0, max(0.0, p)) * 65535 + 0.5) for p in hm])
png.write_rgb8(f'{OUT}/biomemap.png', SIZE, SIZE, bm)
png.write_gray16(f'{OUT}/heatmap.png', SIZE, SIZE, [0] * N2)   # no Ashlands; a cool sea
png.write_gray16(f'{OUT}/forestmap.png', SIZE, SIZE, fm)
with open(f'{OUT}/biomemap.txt', 'w') as f:
    f.write(''.join('%s: %02X%02X%02X\n' % ((n,) + c) for n, c in COLOUR.items()))

# The start temple: the Meadows pixel nearest (-800, -600), painted as a 3 x 3 block.
def dist(i):
    return math.hypot((i % SIZE / (SIZE - 1) - 0.5) * SIDE + 800,
                      (0.5 - i // SIZE / (SIZE - 1)) * SIDE + 600)
want = min((i for i in range(N2) if biome[i] == 'Meadows'), key=dist)
lm = [(0, 0, 0)] * N2
for dy in (-1, 0, 1):
    for dx in (-1, 0, 1):
        lm[want + dy * SIZE + dx] = (255, 0, 0)
png.write_rgb8(f'{OUT}/locationmap.png', SIZE, SIZE, lm)
with open(f'{OUT}/locationmap.txt', 'w') as f:
    f.write('StartTemple: 255,0,0\n')
land = 100.0 * sum(1 for b in biome if b != 'Ocean') / N2
top = max((p - 0.30 / A) * 200 * A for p in hm)
print('land %.1f%%, highest %.0f m, start temple at pixel (%d, %d); done %d'
      % (land, top, want % SIZE, want // SIZE, SIZE))
```

Run it, check it and render it like any map of the workspace (chapter 6, steps 4 to 6):

```
cd generator
(ulimit -v 6000000; nice -n 10 python3 mini_gen.py 512 /tmp/mini)
(ulimit -v 6000000; nice -n 10 python3 check.py /tmp/mini)
(ulimit -v 6000000; nice -n 10 python3 preview.py /tmp/mini /tmp/mini.png 1)
```

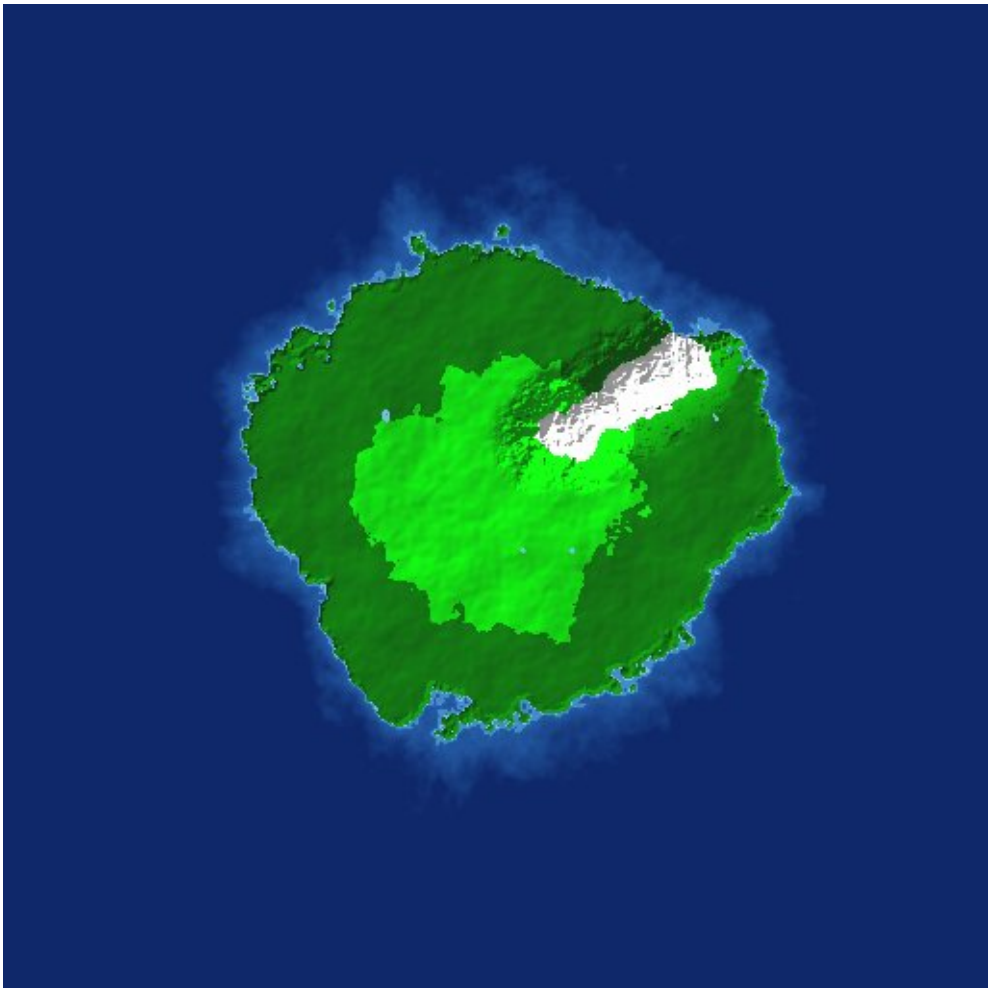
What it printed (real, at the workspace's Amount 2; 512 px took about 3 s on one machine, and the work grows with the square of the size):

```
land 19.2%, highest 162 m, start temple at pixel (236, 270); done 512
512x512   land 19.1% of the image   (reference world Era02: 33.8%)
highest 162 m   deepest -60 m

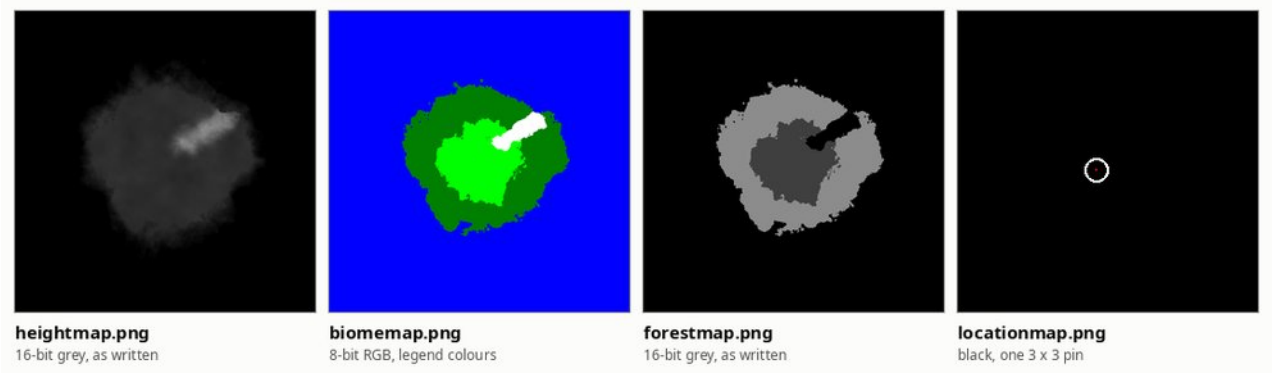
biome         % map  % dry   mean    min    max
Ocean         80.85    0.0  -56.6  -60.0   -0.5
BlackForest   12.73   99.9   12.8   -1.6  100.2
Meadows        5.42   99.9   15.1   -0.6  100.5
Mountain       1.00  100.0  124.7   83.5  162.5

dry ground labelled Ocean: 0   (minimap paints it as sea)
heat > 0 on land that is not AshLands: 0   (MUST be 0 - IsAshlands is heat > 0)
AshLands pixels with no heat: 0
max heat 0.000

  StartTemple                     Meadows        10.9 m      999 m from centre
```



The minimal generator's map set as `preview.py` *draws it, 512 px, unedited. Notice the ragged edge between the meadow (light green) and the forest, and the snow that follows the crest instead of drawing a circle. Notice also what one noise scale cannot do: the island's outline is still roundish, with none of the bays and headlands that* `gen.py`*'s three coast scales and leash give (chapter 4, tell 4).*



The four map files of the same run as they are on disk, each scaled to 256 px (the fifth, `heatmap.png`, is all black). Notice how dark the heightmap is: at Amount 2 the waterline is grey 0.15 and the highest crest only 0.55, so a correct heightmap of a low island looks nearly black in an image viewer. The location map is black with one 3 x 3 pin, circled here.

The config that goes with it (the forest pair stays at the defaults, so the forest map's black is a clearing, chapter 2):

```
[00 BetterContinents.Debug]
Directory = /tmp/mini

[02 BetterContinents.Heightmap]
Heightmap Amount = 2

[07 BetterContinents.Misc]
SelectedPreset = From Config
```

Worked example: the same map at Amount 1. With `AMOUNT = 1.0` in a copy of `mapscale.py`, the run stops before it writes anything (real):

```
AssertionError: 546 pixels above the 140 m ceiling of Amount 1
```

The guard did its job: the crest reaches 162 m and Amount 1 ends at 140 m. Lower the heights, not the guard: with `PEAK, SNOW = 110.0, 70.0` the same run printed `land 19.2%, highest 101 m, start temple at pixel (236, 270); done 512`. The config then says `Heightmap Amount = 1` (or nothing, since 1 is the default; say it anyway).

What it leaves out, on purpose, and where each part is in `gen.py`: the rivers (`carve_rivers`), the shallows with a biome (`label_shallows`), the alt-biome map (`write_altbiomemap`), the lava and heat of an Ashlands, more biomes and landmasses, and a site search for every boss. A world made from it has a meadow, a forest and a mountain, and every location that needs another biome fails (`locations.py` lists them): it is a skeleton to grow, not a world to play. Chapter 11 has complete setups.

If you cannot run Python, the human can still run it: give them the file as it is, the three commands above and the line to send back (the last line it prints, and `check.py`'s output). **If no one can run Python**, a map can be painted by hand in an image editor that writes 16-bit grey PNG: chapter 2's rules (exact colours, hard edges, the heights by the formula) are the whole contract, and the game's log checks the files (`Time to load` for each map, the `Biome map (N x N): ...` shares, chapter 2).

The Amount, in the generator

Every height in `gen.py` is a constant in metres above the water, turned into a pixel by one function, `H(metres) = SEA + metres / (200 x A)`, with `SEA = 0.30 / A` and `A = mapscale.AMOUNT` (2.0 here). That is why changing the Amount does not change the map's shape, and why a lower Amount clips it: a pixel cannot exceed 1.0, so at Amount 1 every mountain above 140 m (the range uplift, `MASSIFS`: peaks to 254 m) flattens against the top while the icefields' plateaus at 105 to 125 m survive, and `gen.py` clamps without a message. `check.py` shows it as a `highest` equal to the ceiling. If you write your own generator on this skeleton, give it the same two properties, one place for `A` and a guard that refuses to write clipped ground:

```
clipped = sum(1 for p in pixels if p > 1.0)
ceiling = (A - 0.30) * 200
assert clipped == 0, '%d pixels above the %.0f m ceiling of Amount %g' % (clipped, ceiling, A)
```

For an Amount 1 map, lower the range, the massifs and the snow line below 140 m and expect Hildir's cavern, Moder's altar and the two tower ruins to be out of reach (appendix C); pin Moder's altar on the summit if the story needs it, because a pinned site is not checked (chapter 2). Then record the Amount in the written heightmap (`png_extras.py record heightmap.png --set 1`, chapter 6 step 7).

A tall map works the same way: set `AMOUNT` in a copy of the generator folder to the smallest Amount that holds the highest peak (chapter 2), and `H()` and the guard follow it. Two things the toolkit does not do yet:

- **It writes no fine picture.** `gen.py` keeps every height as a double in metres (`hm`) until the last loop of `main()` truncates it to 16 bits (`int((SEA + hm[i] / scale) * 65535)`), so the exact value is there to split into `c` and `f` as chapter 2 shows; but `png.py` writes only 16-bit grey and 8-bit RGB, and `png_extras.py` writes no fine record.
- **It holds every map whole**, in Python lists and arrays of `SIZE x SIZE`, so its memory grows with the square of the size: a 16,384 px map is 64 times a 2048 px one. A run that dies at the cap means the size is too big for the toolkit as it is; never raise the cap (chapter 1).

Heightmap Alpha (chapter 10) needs a different last step: a heightmap with an alpha channel is 16-bit grey plus 16-bit alpha (PNG colour type 4), which `png.py` does not write. Write the height and the alpha as two 16-bit greyscale PNGs and join them with `png_extras.py gray-alpha`. The alpha is your design tool for a partial map: opaque where the generator's ground is wanted, fading to transparent where the game's own terrain should take over.

Changing the map

- Move or resize a landmass: its tuple (`CR_OUT`, `CR_ARM_S`, `MIST`, `PM`, `ISLANDS[k]` ...). Ask the human first; that is their layout.
- More or less of a biome: the front constants. The Ashlands/Black Forest front went from 0.09 to 0.05, in three places, when the human said the Ashlands was too small; the plains/swamp divide (`g = wx - 0.34`) moved 840 m east for more plains.
- Higher peaks: the range uplift (`62 + 158 * arch^0.6`, peaks 254 m), the `MASSIFS` heights, and the snow line (58 m; raising the uplift without raising it doubles the Mountain biome). Keep every peak below the Amount's ceiling (340 m at 2, 140 m at 1).
- Relief: `ROLL`, `CLIFF`, the beach and inland ramps.
- Alt biomes: `ALT_PLANTS` and `ALT_COLOUR`.
- Everything else: the generator's own comments explain each constant's reason, often with the version it changed in. Keep that habit: a constant without its reason is the next session's trap.

Worked example: "the Ashlands is too small". The human said it of map version 11. The change, as `diff` shows it between the two snapshots (real lines, shortened):

```
$ diff history/gen-11-rivers-lakes-mushroom.py history/gen-12-locations-lava-rim.py
<     south = fas - 0.09
>     south = fas - 0.05
<                 elif fas > 0.09:
>                 elif fas > 0.05:
<             w_as = sstep(0.09 - FW, 0.09 + FW, fas)
>             w_as = sstep(0.05 - FW, 0.05 + FW, fas)
< MASSIFS = [(0.14, 0.68, 0.045, 95.0), (0.44, 0.32, 0.052, 92.0), (0.16, 0.35, 0.047, 98.0),
> MASSIFS = [(0.14, 0.68, 0.045, 175.0), (0.44, 0.32, 0.052, 170.0), (0.16, 0.35, 0.047, 180.0),
```

The front moved in all three places that read it (move one and the biome, its blend and its terrain disagree). The massifs rose in the same version because map version 12 went to Amount 2, where they fit. What `check.py` measured before and after (real numbers from its two logs, set side by side; version 12 also added the burning isles off the arm's tip, so not all of the growth is the front's):

```
version 11:  AshLands  2.26% of the map   Mountain highest 129.4 m   highest 129 m
version 12:  AshLands  3.79% of the map   Mountain highest 253.9 m   highest 254 m
```

After any change: snapshot, regenerate, measure, render, show. Chapter 6.

6 The workflow

One round, from a change to a proven world. Do the steps in order. Every step has a command and an acceptance criterion; a step whose criterion fails is not done. `$T` is the workspace, `/home/rohan/WubarrkCODING/BetterContinents10/.work/BetterContinents-TestMaps`; `NN` is the version number you are making. Every heavy command is written with the cap; keep it.

This chapter starts from nothing, with `gen.py`. Chapter 9 is the other starting point: `bc_export` a real world (vanilla or Better Continents) instead, at the Heightmap Amount and Sea Level the workspace's maps use (`bc_export amount=2` for ProximaMaxi's 2; the export's own default is 1), and cut pieces of it into `fire-and-ice/` (or the reverse) before Step 1 below. After Step 12, the last section is the faster loop for tuning settings in a world that is already open.

At your level

The steps below are written for the **Commands** level on Wubarrk's machine, with the human at **Game**. Chapter 1's table, "The round, step by step, at each level", says who does each step at **Talk**, **Files** and **Commands**; the notes here and in the steps that begin "If you can" say what changes.

If you can only talk, give the human every path and command written out in full, one per line, and ask for the output back as text; never assume they ran something you did not see the result of. **If you can edit files but not run commands**, you can still write the config (step 8) and read the log (step 10); ask the human to run the generator and the checks, and to paste their last lines. **On another machine than Wubarrk's**, unpack the toolkit, and read every path below as that machine's: the config and the log are in the profile's `BepInEx/` folder, the worlds in the game's save folder (chapters 1 and 11 list where it is).

A round in short (map version 13, chapter 8): the human said "the world reads too flat"; `relief.py` was written to measure it; the generator's `ROLL`, ramps and bog changed (the result: Meadows 4.02 m rms over 50 m); a regenerate at 2048 px; the four checks (land 27.7%, every lowland biome above 90% dry, 23 planted regions, every progression site placed); three previews shown to the human; the human approved; publish; no restaging (the config had not changed); the human's new world; the log (`Genloc duration: 18089.898 ms`, `locations: 183`, 34 short lines); the replay (148 of 170 lines exactly as the checker said). One working session.

Every command below runs in the generator's folder:

```
cd /home/rohan/WubarrkCODING/BetterContinents10/.work/BetterContinents-TestMaps/generator
```

Step 0: know the state

Read `$T/HANDOFF.md` (the current version, what the human last said, what is open), then `$T/fire-and-ice/README.md`. Check that the mod and the game are what you think:

```
P=~/.local/share/com.kesomannen.gale/valheim/profiles/BC-Tester
grep -m1 "Loading \[Better Continents" "$P/BepInEx/LogOutput.log"
head -20 /home/rohan/WubarrkCODING/libs-Tools/CURRENT-GAME-VERSION.md
python3 -c "import mapscale; print(mapscale.AMOUNT, mapscale.SEA, mapscale.CEILING)"
```

Criterion: `Loading [Better Continents 0.10.x]` (a world it makes From Config is settings version 12; the alt-biome map needs 0.8.1 or later; an Amount above 5, fine heights or a world past 16,350 m with wide sectors needs 0.10.3 or later, to make the world and to play it, chapters 2 and 10); Valheim 1.0 (the data files match the current build; a new game build means re-running the extractors, appendix A); and the Amount you chose (chapter 2): for ProximaMaxi 2.0, waterline 0.15, ceiling 340; at Amount 1 it would print `1.0 0.3 140.0`. The line prints the workspace's Amount, not the mod's default (1), which the staged config overrides.

Step 1: agree the change

State in one line what you will change and why, in the human's terms ("fatten the Ashlands arm and add burning isles off its tip"). If it moves a landmass or changes the layout, ask first. If it is a refinement of shape, relief, rivers, alt plants or locations, proceed.

Step 2: snapshot before you touch anything

```
diff -q history/gen-13-relief.py gen.py || cp gen.py history/gen-NN-<short-name>.py
```

The last snapshot is the previous version; `diff history/gen-12-locations-lava-rim.py gen.py` shows what the last session changed. Never lose a version the human has seen.

Step 3: change the generator

Edit `gen.py`. Write the reason for every new constant in a comment beside it, with the version. Probe single points with `evaluate(u, v, dbg)` while you work, and run at 512 for a quick look:

```
(ulimit -v 6000000; nice -n 10 python3 gen.py 512 /tmp/quick) && \
(ulimit -v 6000000; nice -n 10 python3 preview.py /tmp/quick /tmp/quick.png 1)
```

Step 4: generate at full size

```
mkdir -p history/out-NN-2048
(ulimit -v 6000000; nice -n 10 python3 gen.py 2048 history/out-NN-2048) 2>&1 \
  | tee history/out-NN-2048/gen.log
```

Criterion: it ends with `done 2048`; no `!! no site found` line (every pinned location found a pixel); the rivers line and the `alt ...` lines look like map version 13's (chapter 8). About 40 s. The toolkit was built and measured at 2048; a much larger size costs memory with the square of the size (chapter 5).

Step 5: measure

Four checks, in this order, each saved beside the output:

```
(ulimit -v 6000000; nice -n 10 python3 check.py history/out-NN-2048) \
  | tee history/out-NN-2048/check.log
(ulimit -v 6000000; nice -n 10 python3 relief.py history/out-NN-2048) \
  | tee history/out-NN-2048/relief.log
(ulimit -v 6000000; nice -n 10 python3 altbiomes.py history/out-NN-2048 \
  ../previews/NNh-altbiome-regions.png 200) | tee history/out-NN-2048/altbiomes.log
(ulimit -v 6000000; nice -n 10 python3 locations.py history/out-NN-2048) \
  | tee history/out-NN-2048/locations.log
```

> **Check:** `check.py` (seconds). Land 25 to 35% of the image. Every lowland biome mostly dry: Meadows, Plains and Black Forest above 90% dry with means of 10 to 25 m; Swamp about 90% dry with a mean of 4 to 8 m; Mountain 100% dry with a maximum above 200 m; Deep North maximum above 100 m. The three invariants: `dry ground labelled Ocean: 0`, `heat > 0 on land that is not AshLands: 0`, `AshLands pixels with no heat: 0`. Every pinned location listed with its biome and a sane height. A lowland biome near 0% dry, or a mean within a metre or two of the waterline, is a drowned world: stop.

Check: `relief.py` (about a minute). Meadows about 4 m over 50 m and 9 m over 250 m; the coast profile climbing about 10 m in the first 100 m. Compare with the previous version's log, not with an absolute; the human's eye is the judge.

Check: `altbiomes.py` (a few minutes with 200 seeds). Planted regions: the number you planted (23 on map version 13), each on its intended base biome, no one-point regions (edge under 20). Every alt biome `reaches min` at 100% (or you know why not). It also writes the regions overlay picture.

Check: `locations.py` (a minute or two). VERDICT `placed nothing:` should name only `Mistlands_Giant2` (crowded out by Giant1; cosmetic). Every progression and quest site at full count: StartTemple, Eikthyrnir, GDKing, Bonemass (at least 1 of 5; map version 13 has 4), Dragonqueen, GoblinKing, the Queen's entrance, FaderLocation, DN_Bossroom, Haldor, Hildir's camp, crypt, cave and plains fortress, the Bog Witch, CharredFortress, the three PlaceofMystery. Every mountain, Deep North, harbour, frozen ship and lava location at full count. The `!` lines (short of quantity) should be the area-limited ones you expect: goblin towers, henges, giants, swords, some swamp huts, shipwrecks. Read the reason counts on each `!` line: `altitude` dominating means the biome sits at the wrong height; `area` means the biome is too fragmented for a Median location; `similar` means the group's spacing, i.e. area.

Step 6: render, and show the human

```
(ulimit -v 6000000; nice -n 10 python3 preview.py history/out-NN-2048 \
  ../previews/NN-full.png 2)
(ulimit -v 6000000; nice -n 10 python3 preview.py history/out-NN-2048 \
  ../previews/NNd-spawn.png -2 0.40 0.52 0.62 0.72)
(ulimit -v 6000000; nice -n 10 ~/upy/bin/python altmap_preview.py history/out-NN-2048 \
  ../previews/NN-full.png ../previews/NNg-alt-labelled.png)
```

`preview.py` draws land in its biome colour, shaded by height and lit from the north-west, water by depth with the wadeable shelf in light blue, lava orange, and land labelled Ocean in magenta (a bug flag). The scale argument downsamples (2 turns 2048 into 1024); a negative scale enlarges a crop. A crop is **four fractions of the map** (x0 y0 x1 y1, 0..1, y down); the tool refuses anything else. Number every picture so that the human can name it ("i like it 06").

If you cannot show a picture yourself, give the human the command that renders it and the path it writes, and ask what they see; a picture they never saw is a decision they could not make.

Human: Show the picture the way you agreed with the human at the start (chapter 1) and the way your environment can: an image in the conversation, a page they open, a link, or the file's path for them to open (for Wubarrk: one page, updated at the same address). If you cannot show images at all, describe what matters (what changed, where on the map, what to look at) and give the path. Add one line: what changed, what is next. Then stop and wait. Do not go round again on your own; the next change is theirs to choose. If they say it is good, go on to step 7. If they ask for a change, go back to step 1.

Step 7: publish the version

When the human approves:

```
cp history/out-NN-2048/*.png history/out-NN-2048/*.txt ../fire-and-ice/
cp gen.py history/gen-NN-<short-name>.py
```

Update `../fire-and-ice/README.md` (the layout, the height scale table, the alt plants) and `../HANDOFF.md` (what version NN is, what changed, the measured numbers, what the human said).

Then put the Amount on the heightmap itself, so that Better Continents can warn when a config disagrees with it (chapter 2): `python3 ../ai-skill/png_extras.py record ../fire-and-ice/heightmap.png --set 2` adds a text chunk, `Heightmap Amount = 2; Sea Level Adjustment = 0.5`, that image editors keep and that does not change a pixel. Run it with the Amount `mapscale.py` holds. The two runs, setting the record and then reading it back:

```
$ python3 ../ai-skill/png_extras.py record ../fire-and-ice/heightmap.png --set 2
../fire-and-ice/heightmap.png: recorded Heightmap Amount = 2; Sea Level Adjustment = 0.5
$ python3 ../ai-skill/png_extras.py record ../fire-and-ice/heightmap.png
../fire-and-ice/heightmap.png: Heightmap Amount 2, Sea Level Adjustment 0.5
```

A heightmap with none prints `<file>: no Better Continents record` (map version 13's, published before the record existed, does). A map with fine heights gets its `heightmap-fine.png` and the fine record after this, never before: the fine record holds the CRC-32 of `heightmap.png`'s bytes, which this record changes, and a fine picture whose CRC does not match is not used (chapter 2).

Step 8: stage into the game profile

```
../stage-into-gale.sh "BC-Tester"
```

It refuses to run while Valheim is running (the safe order is stage, quit, relaunch, create; chapter 2). It copies the Better Continents DLL (the path you give it, else the repo's `dist/plugins/` build) into `BepInEx/plugins/Wubarrk-BetterContinents/` (renaming any other Better Continents DLL to `.bak` rather than overwriting a hardlinked file), and writes `BepInEx/config/BetterContinents.cfg` with `Directory` pointing at `fire-and-ice/`, `Heightmap Amount` from `mapscale.py` (2.0, where the mod's own default is 1), the forest settings, `SelectedPreset = From Config`, `Debug Mode = true` (it turns cheats on when you spawn, for the `bc` commands, and does not reveal the map; most `bc` changes do, `bc reset` for one, so that the world can be checked without sailing it; turn it off for a world meant to be played) and the alt-biome override `Wolf Mountain: enabled=false`. Appendix B shows the file. If the config did not change since the last staging (same Amount, same directory), restaging is not needed: the game reads the images at world creation.

If you cannot run the staging script (another machine, or the Talk or Files level), write the same lines into `BepInEx/config/BetterContinents.cfg` of the profile that has Better Continents, each under its section, and leave the rest of the file as it is:

```
[00 BetterContinents.Debug]
Enabled = true
Debug Mode = true
Directory = /path/to/fire-and-ice

[02 BetterContinents.Heightmap]
Heightmap Amount = 2

[04 BetterContinents.Forest]
Forest Amount = 0.5
Forest Factor Overrides All Trees = true
Forestmap Multiply = 0
Forestmap Add = 1
```

```
[07 BetterContinents.Misc]
SelectedPreset = From Config

[08 BetterContinents.AltBiomes]
Overrides = Wolf Mountain: enabled=false
```

Copy the DLL into the profile's `BepInEx/plugins/` by renaming the old one first (trap 19), never by copying over it.

The script writes the whole file, so every key it does not write is at the mod's default after the next start, the 0.10.3 keys included: `Max Map Size = 16384`, `Compact Maps = Auto`, `Wide Sectors = Auto`, `High Terrain = Auto` (appendix B). Never write `Wide Sectors = On` into a profile's config unless the human asks for a world to be converted, after its backup: it converts every existing big world the profile then opens, one way (chapters 1 and 10).

Step 9: the human builds a NEW world

Tell them exactly this:

1. Quit Valheim if it is running. Launch it from Gale with the `BC-Tester` profile.
2. New World. A NEW name (never reuse a name: the old world's folder and cache would alias it). Set the preset dropdown beside the seed to **From Config**. Create.
3. It takes about 20 s longer than vanilla (`Genloc duration: 18089 ms` on map version 13). A world with a bigger World Size takes longer still: its alt-biome grid is bigger (chapter 10). With maps of 16,384 px, Create shows "Please wait" for 17 to 39 s (measured on one machine) while the maps are read (before 0.10.3 the game window froze instead): tell the human before, so that they wait for it.

Do not build it for them, and do not touch their `worlds_local`.

Step 10: read the log

The log is the profile's `BepInEx/LogOutput.log` (the first line below holds its whole path on Wubarrk's machine). It is overwritten at every launch: copy it to `history/log-NN-<world>.txt` at once.

```
L=~/.local/share/com.kesomannen.gale/valheim/profiles/BC-Tester/BepInEx/LogOutput.log
cp "$L" history/log-NN-<world>.txt
grep -c "Failed to place all" "$L"
grep -E -e "Genloc duration|Alt biomes: planted|Alt biomes \(world load\)|refused" \
  -e "Altbiomemap file|Lavamap file|Selected .* for spawn|NOT placed|NOT applied|Exception" \
  "$L" | grep -v "GetBiome error"
grep "Failed to place all" "$L" \
  | sed -E 's/.*Failed to place all ([^,]+), placed ([0-9]+) out of ([0-9]+).*/\1 \2\/\3/'
```

If you can only talk, ask the human to open the log in a text editor, search for `[BetterContinents]` and `Failed to place all`, and paste those lines; or to run the three commands above and paste the output.

Lines of a real build's log (map version 13, made by Better Continents 0.8.1, so the settings dump said `Version 0` where a world made by 0.10.x says `Version 12`):

```
[Info   :   BepInEx] Loading [Better Continents 0.8.1]
[Info   : Unity Log] [BetterContinents] Heightmap amount 2, blend 1, add 0, mask 0
[Info   : Unity Log] [BetterContinents] Alt biomes: planted: 23 region(s), 23 alt biome(s)
» applied, 0 protected (none), 0 skipped for the base biome, 0 refused (disabled or
» incompatible)
[Info   : Unity Log] 09/23/2026 12:47:48: Loading: Done. Total seconds sinze ZNet start:
» 21.498534, Genloc duration: 18089.898 ms, (Total genloc time: 00:00:18.3660060, locations:
» 183, iterations: 0)
```

(`sinze` is the game's own spelling.)

Criterion, in order:

- `Loading [Better Continents 0.10.x]` (the exact version you staged), then `Alt biomes: patches bound` at startup.
- At creation: `applying selected preset 'From Config'`; `Time to load .../heightmap.png` and the other six files, `Altbiomemap file (2048)`, `Lavamap file (2048)`; nine `Selected <name> for spawn position` lines (one per pinned site); the settings dump with `Version 12` (a new world; a `Version 11` there means that `Override version` is set, trap 31) and `Heightmap amount 2, blend 1, add 0, mask 0` (the Amount you staged); and nine `Position of <name> (1/N) overriden: set to ...` lines, one per pinned site, with no `was NOT placed` or `override was NOT applied` line (a pin whose zone was taken). A heightmap whose recorded Amount the config disagrees with logs a warning that names both.
- What 0.10.3 added, where the map uses it: a map more than 8,192 px across makes the world compact, and a line names that map; no `is a picture of a kind that is read whole` line (chapter 2); at an Amount above 5, `High terrain: ...` in the settings dump, with the top height; with fine heights, `Fine heights: heightmap-fine.png refines heightmap.png with N bits a pixel ...` and no `Fine heights: ...` line that says why it is not read (chapter 2); past 16,350 m, `Sectors: this new world reaches N m, so it is made with wide sectors (Wide Sectors is Auto): every zone out to 65504 m has a sector and save chunks of its own.` (chapter 10).
- `Alt biomes: split 23 planted region(s)`, `Alt biomes: planted: 23 region(s), 23 alt biome(s) applied, ... 0 refused`; per alt biome, `+N planted (counted toward the maximum)` where you planted.
- `Genloc duration: ... ms, ... locations: 183` and a `Failed to place all` count near the checker's `!` count (34 on map version 13). Every progression site absent from the failed list (Bonemass 4/5 is fine).
- No `Exception` from BetterContinents. `GetBiome error A -> B pos -> x, y` warnings once a second are vanilla's `Player.UpdateBiome` noticing that its 12 m sector grid disagrees with the exact biome at a shoreline: harmless.

Step 11: replay the log

```
(ulimit -v 6000000; nice -n 10 python3 locations.py ../fire-and-ice \
  --log history/log-NN-<world>.txt) | tee history/out-NN-2048/locations-vs-game.log
```

Every line now carries `game: N/M` or `game: all placed`. Criterion: the checker and the game agree on which locations are short, and the counts are within a handful (Unity's random numbers differ from Python's; map version 13 matched 148 of 170 lines exactly, the biggest gap GoblinCamp2 147 vs 118). A location the game placed far fewer of than the checker, or the reverse, is a modelling gap: read the line's reason counts and the location's rules in appendix C before changing the map. In a high world the checker still applies upper altitude limits of 1,000 m or more, which the world lifts (chapter 3), so the game can place more of the locations that stand on ground that high.

> **Human:** Report in three lines: the counts (locations placed, short lines, planted regions), the sites (all progression sites placed, or which are not), and the one thing to look at in game. Then the human plays and judges. Their words ("looks good on my end", "reads too flat") start the next round at step 1.

Step 12: hand off

Update `HANDOFF.md`: the state table at the top, what the version is, the numbers, the open ideas and the working rules. Record the world's name and where its log copy is. A session that ends without a current handoff wastes the next one's first hour.

The live loop: tuning a world that is already open

Steps 1 to 12 build a new world for every change, which is right for the maps: they are baked when a world is created. For settings, and for trying a changed picture, the host of a test world can work live instead. It is a faster loop for the human, not a replacement for steps 9 to 11: nothing proved by it counts until a new world has been built and its log replayed.

The loop is the `bc` commands, which are cheat commands. It needs cheats on (`[00 BetterContinents.Debug] Debug Mode = true` switches them on when you spawn, and the staging script sets it; typing `devcommands` does too), the machine that runs the world (single player, or the host) and a world that uses Better Continents: the zone regeneration refuses a vanilla world. The regeneration comes with the `bc` commands, not with Debug Mode. In a world that uses Better Continents, on the machine that runs it, Debug Mode also shows `Better Continents Debug Mode Enabled!` at the top left and adds a Better Continents window with the same values as the `bc` console tree: a button in the Esc menu and on the large map, and Alt+F8 while either is open (it does nothing elsewhere). Debug Mode does not reveal the map; most `bc` changes do, since they redraw the minimap and explore all of it (a change to Biome precision and the alt-biome ones do not redraw it).

The `bc` command is registered for the host only, but the Server devcommands mod lets any player with devcommands type it on their own machine. There a change alters only that player's copy of the settings and their own view of the ground, never the world or the other players, and the zone regeneration refuses ("it runs on the machine that runs the world"). Make `bc` changes on the host or in single player; `bc info` on a joined player's machine shows the settings the server sent. The loop:

1. **Change a value** in the game's console (F5). `bc h am 1.8` sets Heightmap Amount, `bc g sl 0.45` Sea Level, `bc fo am 0.6` Forest Amount, `bc b p 8` Biome precision, `bc h alpha true` Heightmap Alpha. `bc <group> <name>` alone prints the current value, `bc <group> help` lists a group, and appendix B lists every setting with its console name. Or, after the generator wrote new pictures into the Directory, **reload them**: `bc reload hm` (also `bm`, `fom`, `heat`, `lava`, `ab`, ...) reads one picture and its legend again, `bc reload all` every map the world has. A reload reads the picture by the loaded world's own rule (chapter 10): a version 12 world reads a location map's pins exactly, a version 11 world at the corner, as 0.9.4 did.
2. **Better Continents redoes what the change touches**: the noise, its patches, the minimap, the alt-biome grid, the terrain that is loaded (the distant terrain and the grass too, and a zone that is loading at that moment), and then **the zones**. The terrain rebuild stalls the game for a second or two (1.6 s measured). Better Continents regenerates the zones itself, in single player or on the host, in a world that uses Better Continents, spread over frames: the console prints a start line, progress lines and a finish line (quoted in full below). Every generated zone is reset except the ones listed below, and each reset zone generates again with the new settings, its locations included, when someone comes near, so go and look. `bc regen` does the zone part on demand: it regenerates the zones without rebuilding the loaded terrain, since nothing changed and the ground already matches the settings, and it does not redraw the minimap or rebuild the alt-biome grid, which `bc reset` does first. A change made, or `bc regen` typed, while a regeneration runs starts it over, after the zone it was in is finished. The config key `[00 BetterContinents.Debug] Debug Reset Command` is empty by default for this reason; a console command written there, such as Upgrade World's `zones_reset start`, runs instead of the regeneration (a config that still holds the old default, `zones_reset start`, is set to empty once, with a line in the log that says how to go back). A zone that fills again could

double its trees; the twin guard (chapter 10) keeps a plant out when a plant of the same kind already stands within a metre of its spot.

3. **Look.** `bc scr` saves the minimap as a PNG under `<save data>/BetterContinents/<world>/` (2048 px unless you pass a size), `bc info` prints the settings the world now has, and `bc locs` counts the locations it placed.
4. **Carry the answer out.** When the numbers are right, write them into the config (or the generator) and build a NEW world as in steps 9 to 11. A world edited this way is a test world; `bc savepreset <name>` keeps its settings as a New World preset if the human wants to compare.

What the regeneration leaves alone. A zone and the eight around it (the 3 x 3 block) are left alone when it holds

- something a player built or planted: `Piece.SetCreator` is called only by `Player.PlacePiece`, so a creator on an object means a player placed it (buildings, chests, ships, portals, crops and saplings while they grow);
- a tombstone;
- ground work: what the hoe, the cultivator and the pickaxe do (fields, paths, roads, levelled or dug ground) lives only in the zone's `_TerrainCompiler` object, which has no creator, so the regeneration looks inside it. Crops stay with their field this way, because a grown crop has no creator either;
- you, while you are inside a dungeon (the game builds dungeons high above the ground and counts any height over 3000 m as inside; in a high world, more than 3,000 m over the ground below, chapter 3). Your block is the 3 x 3, or the widest block that a connected player keeps, if that is wider.

Also kept. A player connected from another machine keeps the zones their game has loaded around them (their simulation distance: 5 x 5 by default, never less than the 3 x 3; a player who has joined but has no character yet counts where their game says they are), because their game still has the settings it joined with. This is looked at again at the start of every frame, so a player who moves keeps the zones around where they are now, and a piece you place during the run keeps its zone from then on: a zone that has not begun and lies there is left alone, and so is one that holds something that stays when its turn comes, with the 3 x 3 around it; a zone that has begun is always finished. The ground around your own character on the surface is regenerated like the rest.

Locations stay whole. A location can reach over its zone's edge, so the zones one location reaches into are kept or reset together, and so are those of any location that shares a zone with it: if any zone of such a group is within reach of something that stays (a zone nobody has generated yet counts), every generated zone of the group is kept, and a zone left alone at its turn takes the group's zones that have not had their turn with it. When a group is reset, the parts of its locations that stand in a zone nobody has generated yet are cleared with them, so resetting again does not leave a second copy.

Never removed, in a zone that is reset or not: players, what players built, tombstones and tamed animals. A tree a player grew from a sapling is made new when the sapling grows (`Plant.Grow`), so it has no creator and counts as the world's: it goes unless something nearby keeps its zone. A creature that a creature spawner made goes with its spawner, wherever it has walked to, unless it is tamed or stands in a zone that is left alone.

The ground is not kept. A zone that is kept keeps what stands in it, its trees and rocks too, but the ground of every zone, kept ones included, is built from the world's settings, so after a change it follows the new settings. What players did with the hoe, the cultivator or the pickaxe is stored as changes to the ground and rides on the new ground. A change that lowers the ground under a base leaves its pieces standing in the air, and the game breaks a piece that has lost its support a few seconds later (in the 0.10 test a workbench and a floor left about 3 m up broke after about 8 s); a change that raises the ground buries them, and the trees and rocks of a kept zone float or sink the same way. So make big height changes before building, or away from what you want to keep. `bc regen` on its own changes no ground.

Pace and order. The zones nearest a player go first. A frame stops after 6 ms or 1,500 objects, whichever comes first (the clock is read every 32 objects); with other players connected a frame takes at most 100 objects and the work about 6,000 a second, since every destroyed object is 12 bytes to every client. A zone's objects go first. The zone is finished (its location marked not placed, no longer generated, its loaded copy dropped, so that it generates again) only when the zones around it, and those of its location, have had their objects destroyed or were left alone, so that nothing is built over objects that are about to go.

Saves and stops. A zone is never saved half emptied, and a location is never saved half cleared: a save during the run (an autosave, logging out) finishes the zone being emptied, clears the rest of a location the work has begun on, and stores the zones it has emptied as not generated, and they generate again when someone comes near. A request that starts the run over finishes the zone it was in first, and the new run resets the zones that the first had emptied but not finished, whatever has come near them since (the log says `Zone regeneration: starting over.`). Closing the world stops the run; `bc regen` in the next session starts it again. When an error stops the run, the console says so, the zones it had emptied and left waiting for their neighbours are finished, so they generate again when someone comes near, and the zones it had not reached are as they were: `bc regen` resets them.

Borders. Where a zone that stays meets a zone that was reset, the height edits along the shared edge and corner are cleared on the side that stays (the paint stays), for the zones actually reset, when the work is done and at every save, so the two grounds meet. A run that is cut short leaves what it still owes to the next run or the next save.

Limits. A zone that was already reset is not brought back when something is built beside it afterwards. A location whose home zone the game generates while the work is running (someone came near it) can end up half placed, or placed twice; `bc regen` again sets it right. Where objects share one sector, the scan of the world's objects is slower: beyond about 16.4 km from the centre in a world without wide sectors, beyond 65.5 km in a wide one (chapter 10).

The lines it prints go to the console and to the log. With made-up numbers:

```
Zone regeneration: resetting 1,204 of 1,500 generated zones; 296 left alone (near something a
» player built or worked on, a tombstone, or a player; 40 of them to keep a location whole).
Zone regeneration: 121 of 1,204 zones (10%), 8,531 objects removed.
Zone regeneration: 241 of 1,204 zones (20%), 17,022 objects removed.
...
Zone regeneration finished in 12.4 s: 1,199 zones reset, 85,321 objects removed. 3 objects stay
» (players, what players made, tombstones, tamed animals). 3 zones were left alone after all: a
» player, or something a player made, was in or beside them before their turn. 2 more zones
» were left alone to keep a location whole. The ground at the edge of 1 left-alone zone was
» adjusted to meet the reset zones. A reset zone generates again with the new settings when
» somebody comes near.
```

What the lines say:

- The first line: N zones are reset out of M generated and K are left alone (N + K = M). The part after the semicolon is there only when a location keeps some of them.
- A progress line comes every tenth of the way, only when 20 or more zones are to be reset, and counts the zones dealt with, reset or left alone.
- The last line: only its first sentence, the zones reset and the objects removed, is always there. Each later one appears when it applies, in this order: the objects that stay (`1 object stays`, `3 objects stay`); the zones left alone after all because a reason to stay appeared before their turn (`1 zone was`, `3 zones were`); the zones left alone to keep a location whole (`2 more zones were`, or `2 zones were` when the sentence before it is absent); the ground at a border, one sentence whatever the count, with `was` for one zone or many (`The ground at the edge of 1 left-alone zone was adjusted to meet the reset zones.` or `The ground at the edge of 3 left-alone zones`

`was adjusted to meet the reset zones.`); `1 error, see the log.` or `2 errors, see the log.`; and, when at least one zone was reset, the closing sentence.

Other lines:

- `Zone regeneration: no generated zone to reset (296 generated zones are left alone).` takes the place of the first line when every generated zone is left alone (`1 generated zone is` for one); the finish line follows with 0 zones reset.
- `Zone regeneration stopped after 340 of 1,204 zones: the world closed.` takes the place of the last line's first sentence, and its closing sentence is missing, when the world closes while the work is going.
- `Zone regeneration stopped by an error, see the log.` when an error ends the work. The log then has `Zone regeneration: 3 zones left waiting were finished, so they generate again with the new settings when somebody comes near. The zones it had not reached are as they were: 'bc regen' resets them.` (`1 zone left waiting was finished, so it generates again ...` for one).
- `Zone regeneration: load a world first.`, `Zone regeneration: it runs on the machine that runs the world: in single player, or on the host.` and `Zone regeneration: this world does not use Better Continents.` when there is no world, this machine does not run it, or the world is a vanilla one (a world without Better Continents is the game's own: its zones are never emptied, and nothing is rebuilt); and, for a word it does not understand, which starts nothing, `bc regen does not know 'x'. bc regen [dry] [biome=<names>] [min=<m>] [max=<m>] [pos=<x>,<z>] [zone=<x>,<z>] [chance=<0..1>]. 'bc regen help' says what each does.` (or a line naming the bad value, such as `bc regen: 'x' is not a biome.`).
- The screen says `Better Continents: preparing to regenerate the zones ...` and then `Better Continents: regenerating the zones, 37% ...` while it runs. The log alone has `Zone regeneration: starting over.` when a request replaces a run, and `Zone regeneration: stopped, the world closed.` when the world closed before the plan was made.

Resetting only part of the world: `bc regen` words. Plain `bc regen` is as above. Words after it narrow what it resets; every rule above still holds (zones near a base, a tombstone or a player are left alone, a location that reaches into such a zone is kept whole, players, what players built, tombstones and tamed animals are never removed), and a word only ever takes zones *out* of the reset:

```
bc regen [dry] [biome=<names>] [min=<m>] [max=<m>] [pos=<x>,<z>] [zone=<x>,<z>] [chance=<0..1>]
```

Word	Meaning
`dry`	counts what would be reset and changes nothing: the numbers, the biomes of those zones (at their middle), the nearest zones first, the whole list in the log
`biome=Meadows,BlackForest`	zones where any corner is one of those biomes (a zone half in the biome is in); names as the biome map's legend has them, any case, comma between; `black_forest` and `Black-Forest` read too
`min=<m>` `max=<m>` (`radius=<m>` is `max`)	zones whose middle is at least or at most this many metres from `pos` (the world's centre when there is none)
`pos=<x>,<z>`	where `min` and `max` measure from, in metres
`zone=<x>,<z>`	one zone by zone coordinates (a zone is 64 m; a place's zone is its x and z divided by 64, rounded)
`chance=0.25`	each remaining zone is reset with this chance (past 1 is a percentage: `chance=25`); a zone keeps its draw for as long as the request stands

```
bc regen dry biome=AshLands min=3000      then, if the numbers are right:   bc regen biome=AshLands min=3000
bc regen zone=12,-40
bc regen max=2000 chance=0.5
```

> **Rule:** run `dry` first, then the same words without it. A zone the words leave out is treated like a zone near a player's base, so the area that really resets can be a little smaller than asked (a location that reaches over the border is kept whole). With a console command in `Debug Reset Command` the words are refused (a console command takes no filters): empty the setting. Only on the machine that runs the world (single player or the host), in Debug Mode, as plain `bc regen`; `dry` is refused while a regeneration runs, and says so.

`bc genloc`**.** Forces the game's location generation again (chapter 13, part 4, has the details), also when the game's own run is stuck: it says what it found, runs the generation, says what changed when the run ends (how many locations there are, which types changed in number, how many wait for somebody to come near, which locations the game found fewer places for than they ask, and anything of the location map or the start position that could not be placed), and sends the new location icons to the players. Locations already placed and every generated zone stay; the location map's pins and the start position go in first. A run that is alive is not touched and no second one starts; a taken slot with no sign of life is freed first. `bc genloc alt` passes to the game's alt-biome branch (`genloc alt`). `bc genloc pins` only puts the location map's pins and the start position into the registry and generates nothing (a pin already there stays), for a registry that something else emptied. `bc genloc help` says more. It runs where the world runs (single player, the host, the dedicated server); a joined client is told so and nothing happens. In the host's `bc` tree: Debug Mode.

Upgrade World (Jere Kuusela's mod) on a Better Continents world (checked offline against Upgrade World 1.87's own files, not run in a game; it is needed on the server and the admin's client). Upgrade World asks the game for biomes, heights, forests, locations and vegetation, so `zones_reset`, `zones_generate`, `biomes_count` and the vegetation commands follow this world's maps, and a zone it resets generates again through the game (the vegetation map, the twin guard and the baked layer apply as always). What to know:

- **Its World radius** (`[1. General]` in `upgrade_world.cfg`, default 10,500 m) bounds its commands: past it they do not list zones that are not generated yet. `[07 BetterContinents.Misc] Upgrade World Radius` (Off by default, read while the game runs, applied when a world loads) raises it to cover World Size + Edge Size, rounded up to a whole zone (64 m); it is never lowered; Upgrade World writes the value to `upgrade_world.cfg` when it saves its config. Off, the log says once, on a world it does not cover, that the setting exists.
- `locations_restore` builds a fresh registry and merges it with the old one zone by zone. Better Continents puts the location map's pins and the start position into the fresh registry, as the game's `genloc` does, so a restore keeps each pin. If a pin is missing afterwards: `bc genloc pins`. (Without it, `locations_restore <id>` removes that id's pins that nobody has visited.)
- `locations_add`**,** `upgrade tarpits|mountain_caves|bogwitch|bearcave|combatruins|hildir` run Upgrade World's own copy of the game's location search, with every question answered by Better Continents' biomes, heights and forest, and a point in the exclusion map refused. An excluded point counts as one tried, so an excluded area can end with fewer locations than asked; on a world with `Skip Default Locations` nothing is added (only the pins exist).
- `upgrade ashlands` **and** `upgrade deepnorth` **choose zones by distance from a point of the game's own layout** (more than 11,600 m from a point 4,000 m north or south of the centre), not by biome. On a map with its own biome map or size that is not necessarily the Ashlands or the Deep North; Better Continents says so once in the log. *On a Better Continents map use* `bc regen dry biome=AshLands`*, then* `bc regen biome=AshLands`*, not* `upgrade ashlands`. `upgrade mistlands` selects by biome and follows the biome map.
- `world_gen` and `upgrade *_worldgen` change the game's world-generation version in memory (what the maps decide does not change). `objects_remove`, `clean_*` and `prune_savesystem` delete

objects directly, including a baked layer's live pieces in the zones they empty (not tried with a baked world). `ChunkSaveFix` (Upgrade World 1.85 and later) assumes the game's own 64-per-axis save grid: with Wide Sectors on, its clean-up of emptied chunks has not been tried.

Expand World Data and Expand World Size (Jere Kuusela's mods). Expand World Data 1.79 moved two of its classes (`WorldInfo`, `BiomeManager`) into namespaces; the bridge used to look them up by their old names and silently did nothing (Expand World Data never learned the world's size, its lava biomes and terrain were not read, and a new world stored its biomes as numbers instead of names). Better Continents now finds them under both names, and by name and member if a release moves them again. The log says, once at each start, what it reached: `"Expand World Data" <version> detected. Applying compatibility: size yes, biome names yes, lava and terrain yes, minimap height yes.` A *no* names a part that does not work (checked against the real files of 1.73 and 1.79). Expand World Size 1.44 and later widen the game's save grid themselves (up to 65,536 m) for a world past 16,384 m, which is Wide Sectors' job with a different chunk layout: while that grid is wider than the game's, a new world is not made wide, `Wide Sectors` stays off for the session in every mode, and a save with wide-sector chunks is not loaded (the game's load error is set, nothing is saved over it, and the player sees a message naming both). Remove Expand World Size, or set its save grid to the game's (`Save system size` 0 with a world no bigger than 16,384 m), to keep such a world.

What the loop does not do: where the locations go is decided once, when a world is created. A reset zone builds its location again on the same spot, on the ground as it is now, so a changed location map, or a change that moves ground out of a location's altitude window, still needs a new world. Zones left alone (the list above) keep what stands in them, but not their ground (**The ground is not kept**, above). And the loop changes the loaded world's settings, not the config file.

> **Human:** Debug Mode is for a test profile: it turns cheats on when you spawn, which the `bc` commands that rewrite the world need, and adds the settings window. Switch it off (`Debug Mode = false`) for a world meant to be played.

7 The traps that cost worlds

Each of these was found the hard way. The symptom is what you see; check the cause before you touch the map.

#	Symptom	Cause	Fix
1	Every lowland biome under water; the world is bare peaks in an empty sea	The waterline was taken as pixel 0.15 (the pre-1.0 value, and the right pixel only at Amount 2) instead of 0.30 / Amount	Derive the waterline (chapter 2); `check.py` on the correct scale
2	The whole map 25 to 30 m under water, evenly, or the mountains clipped flat	The map was authored for one Heightmap Amount and the world was created at another (the default is 1)	`mapscale.py` is the one source; stage, quit, relaunch, create; a heightmap with a recorded Amount (`png_extras.py`, an export) warns in the log
3	The staged config had no effect on the world just built	Before 0.9.0 the config was read at game launch only; the game was running during staging	Quit and relaunch before creating; the staging script refuses to run while Valheim runs
4	Ground far from the Ashlands behaves as Ashlands (heat, lava, ash spawns)	A non-zero heat pixel there; the mod's IsAshlands is `heat > 0`	Heat exactly 0 off the Ashlands; `check.py` requires it
5	You edited a PNG and the world did not change	Maps are baked into the world at creation	New world, new name
6	Hildir's cavern 0 of 3, tower ruins 0 of 50, frost caves 12 of 120	Amount 1 (the default): a 140 m ceiling; the locations want 100 to 200 m	Amount 2 and peaks above 250 m (chapter 2), or accept the loss and pin what you need
7	Frozen ships 0 of 50, Deep North wrecks 22 of 170, harbours 7 of 100	Water painted Ocean to the waterline; those locations want water of their biome	`label_shallows()`: Deep North to 16 m, Mistlands to 1 m
8	After labelling water, Birch Meadows and the Mistlands alts lost their hosts; 123 planted regions	A region whose edge is under water fails HeightAvg; a multi-biome colour spilled onto foreign shores	Label only what needs it, never against a foreign shore; pull multi-biome alt water 3 px back
9	Dozens of one-point alt-biome regions (71 "Lantern" Black Forest regions)	A multi-biome alt's colour claimed single grid points of a neighbouring biome	The pull-back above; patches painted 2 px inside their own biome
10	A planted patch used up a slot but reads as nothing in game	A spit or tongue was a lone point on the 12 m grid, a region of its own	Keep specks off; paint whole features; check `altbiomes.py` for regions with edge under 20
11	Relief added and nothing visible; "reads too flat"	FBM output is compressed round 0.5 (std 0.13); `ROLL` was worth a quarter of its metres	`ROLL_K = 1 / std`; measure with `relief.py`
12	Every coast a tilted plain climbing 20 m over 500 m	Relief faded in over the same ramp as the beach platform	Beach 210 m, relief from 125 m inland
13	Islands "parked" in a row	Every island displaced by the same warp vector	Anchor the warp at each feature's centre
14	A patch of Plains beside the Deep North	The warp evaluated a continent's field far from it	Leash each field to its unwarped ellipse plus slack
15	The preview tool ran the machine out of memory and crashed the editor	Crop given in pixels; the tool wanted an image hundreds of times the map	Crops are fractions 0..1 (refused otherwise); every heavy command under `ulimit -v 6000000`
16	The Deep North list calls its ships "TarPit1"	`m_prefabName` is stale in the add-on location lists	Name a location by its prefab asset path (`_prefab_path`), as the checker does
17	A vanilla world, no mod, no warning	The preset dropdown said `Disabled` (a new config file's default), not `From Config`	Set it to `From Config`; the config's `SelectedPreset` pre-selects it
18	Two Better Continents DLLs; which one loads is a coin toss	The Thunderstore package and the fork share a BepInEx GUID	The staging script renames the other to `.bak`
19	A mod file corrupted in every Gale profile at once	`cp` over a hardlinked DLL wrote through Gale's shared cache	Check `stat -c %h`; rename, never overwrite
20	A world vanished from the list, or its map was orphaned, after a rename or copy	Valheim's save plumbing carries only its own extensions and drops the mod's sidecar	Never touch the human's saves; the mod's `SaveSystemPatch` compensates in 0.8.x

#	Symptom	Cause	Fix
21	Weather flips at sea on a map with Deep North in its middle rows	Vanilla passes the camera HEIGHT to `IsDeepnorth`; the mod fixes it on worlds with a biome map	Use 0.8.1; the log says `Deep North weather: EnvMan asks IsDeepnorth at the camera's x and z`
22	World creation hangs forever on the progress bar (0.7.x)	A biome missing from the map made a vanilla index throw inside the location coroutine	Fixed in 0.8.0; on older versions include every biome
23	`GetBiome error A -> B pos` once a second	Vanilla's `Player.UpdateBiome` comparing the 12 m sector grid with the exact biome at a shoreline	Harmless; ignore
24	Wolf Mountain appears on the snowy archipelago island as well as where painted	Random placement added one (about one world in five)	`Overrides = Wolf Mountain: enabled=false` in the alt-biome config
25	A painted colour planted nothing	It matched no legend colour within 20 RGB units, or the alt biome does not belong to that base biome	The mod's warning at bake lists the unmatched colours; `bc ab info` in game
26	A quantity-10 location placed 1, or 5 of 10	It is unique: candidates are placed, one is spawned, the rest removed	Expected; the checker's `!` mark ignores it
27	Alt biomes differ between server and clients	Vanilla syncs nothing; clients regenerate from their own maps	The same world file everywhere; 0.8.1 pushes the server's placement and logs a match or a warning
28	A per-type decompile says the game caches the alt-biome grid	A stale `Decompiled_<version>/` folder from an older build; since 1.0.15 the game regenerates the grid on every load	Confirm details in `libs-Tools/1.0/DECOMPILED/assembly_valheim.decompiled.cs`, and check the per-type folder is the current build's
29	You used `libs-Tools/OLD/` as the reference	It holds older builds, each with a complete, current-looking decompile	`libs-Tools/1.0/` and the current `Decompiled_<version>/` (with trap 28); compare the md5 of `assembly_valheim.dll` with the installed game's
30	The minimap reads as forest everywhere	A minimap mask patch in the mod (before 0.8.0) collapsed biome channels	Not tree density; update the mod
31	The land stops short of the world's edge, or changing World Size changed nothing	The world is version 11: made by 0.9, or made in 0.10 from a preset that ships with the mod or one saved by 0.9, from an export of a world resized under 0.9 or made under Expand World Size at another size, or with `Override version = 11`. Its maps always span 21,000 m and World Size only moves the edge	Make a NEW world From Config with Better Continents 0.10 and `Override version` empty; `bc info` says `Version 12` and, off the default size, `The maps span World Size and Edge Size` (chapter 10)
32	Far out in a big world a player is sent thousands of objects that are not near them; portals vanish, or a patch's objects do; zones more than 32.7 km east, west, north or south spawn no creatures of their own after a restart	The world has the game's own sectors and World Size + Edge Size past 16,350 m: past about 16.4 km every object shares one list, from about 22.1 km a patch of zones is saved under the file of every portal, and a save moves the spawners of zones past 32,767 m on either axis to the 20 km line	A new world past 16,350 m is made wide (`Wide Sectors = Auto`, the default). An existing one keeps the problems until `Wide Sectors = On` converts it, one way, after a backup; the damage done before stays (chapter 10)
33	Trees, bushes or rocks have a double standing on the same spot	A zone was filled again over its own objects, or a vegetation map's `+prefab` met another entry for the same plant	The twin guard keeps new twins out of the zones every Better Continents world generates from now on; `bc_twins` counts the old ones and `bc_twins remove confirm` removes them (chapter 10)
34	`The start position override was NOT applied`, or `Location map: X ... was NOT placed`, in the log	The 64 m zone is already held by a pin (an older world places the start last) or by another location	Move the start or the site more than a zone from the other; a new world places the start first
35	A pinned boss stands in the sea, or high ground below its altitude, and the game accepted it	The game does not check pinned sites	Search for a legal pixel near the intended site, as the generator does; read `check.py`'s list of pinned sites
36	A compact world opens as vanilla terrain on Better Continents 0.9	0.9 cannot read the compressed tiles: it switches Better Continents off for that world and generates vanilla terrain. `Compact Maps = Auto`, the default, makes every new world with a map over 8,192 px, or with fine heights, compact	For a world that 0.9 users will open, `Compact Maps = Off`, or Auto with every map at 8,192 px or less and no fine heights; a server and its clients need the same version anyway
37	Heightmap Alpha seems to do nothing, or the heights are stepped	The setting is off, or the world is older (with it on, 8-bit heights and the alpha unused), or the PNG has no alpha channel (it needs 16-bit grey with 16-bit alpha)	A new world, and a heightmap written with `png_extras.py gray-alpha` (chapter 10)

#	Symptom	Cause	Fix
38	`altbiomes.py`, `locations.py` or `check.py` disagree with a world of another size, or a high world	The toolkit's checkers assume the 21,000 m square, World Size 10,000, the 2048-point 12 m grid and the game's own altitude limits, which a high world lifts from 1,000 m up	Compare default-size worlds; for another size change their constants (appendix A); in a high world expect more of the high locations than `locations.py` counts (chapter 3)
39	After a `bc` change the trees and rocks next to a base, a field, a road, a tombstone or another player are as they were, and float or sink where the ground moved; pieces left in the air break a few seconds later	The zone regeneration leaves the zones within one zone of anything a player built, a tombstone, ground work (fields, paths, roads, levelled or dug ground), the zones a player on another machine has loaded, or you inside a dungeon as they are, with their trees and rocks; a location that reaches into one of them is kept whole. Their ground is not kept: it is built from the new settings like all ground	Test away from the base and the fields, and make big height changes before building; the console's `Zone regeneration: resetting N of M generated zones; K left alone` line says how many zones were kept (chapter 6)
40	A square of water looks deeper or shallower than the water around it, after a `bc` change or `bc reset`	Each zone's water read the sea's depth once, when the zone loaded, and ground rebuilt in place afterwards left it at the old depth (fixed for Better Continents worlds in 0.10)	Update the mod; before 0.10, go about 300 m away and back, or log out and in (chapter 10)
41	A new world (version 12) plays by the older rules after it was opened once in Better Continents 0.9 (seams at another World Size, Heightmap Alpha no longer blending)	0.9 reads a version 12 world as version 11, and its next save writes it as 11, so 0.10 then treats it as made by 0.9	Copy the world folder before it meets 0.9; play a new world with 0.10, and keep the server and every player on one version (chapter 10)
42	Forests and clearings in patches far larger than the game's	`Forest Scale = 1`, the default of Better Continents 0.7.20 to 0.10.1: its curve makes 0.5 the game's own size and 1 five times that	`Forest Scale = 0.5` (the default since 0.10.2), then a new world: a world keeps the scale it was made with (chapter 2)
43	Trees stand where the forest map is black	The additive pair (`Forestmap Multiply = 0`, `Add = 1`): black is the game's own forest there, not a clearing	Keep the defaults (Multiply 1, Add 1) where black must be a clearing (chapter 2)
44	`Cannot use texture <file>: it is W x H pixels, and the largest map Better Continents reads is ...` in the log, and the world has no such map	The picture is over 16,384 px, or over this machine's `Max Map Size` (the message then names the setting)	Scale it down, or raise `Max Map Size` to 16384 (chapter 2)
45	Making a world from big maps takes gigabytes of memory; the log says `... is a picture of a kind that is read whole`	The picture is interlaced, 16-bit colour or not a PNG, and is read whole (about 1 GB for a 16,384 px colour picture)	Save plain PNGs: not interlaced, 8- or 16-bit grey, grey with alpha, 8-bit RGB or RGBA, or a palette (chapter 2)
46	Gentle slopes of a tall world show stairs	A 16-bit step is 200 x Amount / 65,535 m: 0.12 m at Amount 40, 0.25 m at 81	The smallest Amount that holds the highest peak; fine heights where the steps still show (chapter 2)
47	`heightmap-fine.png` made no difference	The world did not read it, and a `Fine heights: ...` line in the log says which check failed (chapter 2): often `... was made for another heightmap.png (its record names CRC-32 ...)`, when `heightmap.png` was written again (an editor, `png_extras.py record`) after the fine record was made; or `Compact Maps` is Off	Make the fine picture and its record last, after every write of the heightmap (chapter 6, step 7); `fine_ref.py check` tells before a world is made
48	A world made with fine heights opens as vanilla terrain	It was opened by Better Continents 0.10.0 to 0.10.2, which cannot read its block format (or by 0.9, which reads no compact world)	Play it with 0.10.3 or later only; back up the world folder before changing versions (chapter 2)
49	Two of every chest, and every far object doubled, in a big world	A wide world was saved by an older version (0.10.2 or older, or the game without Better Continents), which writes its objects twice; no setting stops that. An older Better Continents also opens a wide world as one without Better Continents (`Unknown feature: 68` in its log), with vanilla terrain	Restore the backup, and never open a wide world with an older version; a join with another version is refused, so the risk is the machine that runs the world (chapter 10)

#	Symptom	Cause	Fix
50	High up, a player counts as inside a dungeon, or the grass, items, tombstones or creature spawns find no ground	A world read at an Amount above 5 opened by an older version, without the high-world rules; or a world read at 5 or less, which keeps the game's own (its grass finds no ground above 500 m of height)	Play a high world with 0.10.3 or later; at 5 or less, keep the ground under 500 m of height or set `High Terrain = On` before the world is made (chapters 2 and 3)
51	Mountains far higher than drawn: a 1,000 m peak stands about 1,900 m	`Heightmap Override All` is off: the game's own biome heights are made over the heightmap's, and the Mountain biome doubles what is over 80 m	Keep `Heightmap Override All = true`, the default, where the heights must be the drawn ones (chapter 2)
52	Far from the north of a big world, farmland is not drawn as soil, mountain ground is deep snow, the shore band sits lower	The game's terrain shader marks the Deep North with a fixed circle at (0, -4,000), radius about 12,000 m, and gives everything outside it those looks	The game's own rule; Better Continents does not change it. Lay out a big world knowing where the circle is (chapter 3)
53	A big world takes longer to start; New World shows "Please wait" for half a minute	Start time and the alt-biome grid grow with the size (measured on one machine, a 32-core server: about 11 s at the default size, 18 s at 24,500 m on the game's own terrain, 16 s at World Size 32,264 + Edge Size 500 with five 16,384 px maps; the grid is built on every core, so fewer cores take longer); New World reads the maps in the background, 17 to 39 s with 16,384 px maps (before 0.10.3 it read them on the main thread, and the game window froze)	Expected. Set World Size to how far players will really go, and tell the human before they create (chapter 6, step 9; chapter 10)
54	The sea around a map's Ashlands is not red, or red water lies where the map has no Ashlands	The game's water shader colours the Ashlands sea by its own ring, past about 12 km from (0, +4,000). Before 0.10.3 that ring was all there was; since 0.10.3 the red follows the world's hot sea (in 64 m steps in 0.10.3 and 0.10.4, per pixel with Better Continents' own water since 0.10.5), which on a world with a heat map is only where heat > 0 over the sea, with no falloff past the heat map's pixels	Better Continents 0.10.3 or later (0.10.5 for the per-pixel edge), and heat drawn over all the sea that should be red, as `gen.py`'s faded ellipse does (chapters 2, 3 and 13)
55	At the Deep North's edge boats rock on water drawn calm, or sit still on drawn waves	The game's water shader draws the calm sea by its fixed circle, and Better Continents 0.8.0 to 0.10.2 moved the waves that float boats with the biome map (0.10.0 to 0.10.2 with the world's size too), so the two parted. 0.10.3 and 0.10.4 kept both at the game's circle	0.10.5 or later: Better Continents' own water draws the calm sea at the world's own Deep North, and the felt waves follow the same field (chapter 13). On one machine only: trap 71
56	A tall map's world comes out far too low: peaks short of their height, low ground under water	Better Continents 0.10.2 or older made it. There `Heightmap Amount` accepts 0 to 5, so BepInEx clamps a config's value above 5 to 5 when it reads it (and may write 5 back into `BetterContinents.cfg`), and a NEW world made From Config gets Amount 5. An existing world keeps the Amount stored in its own settings, which BepInEx does not read	Make a tall world with 0.10.3 or later; check `Heightmap amount` in the settings dump at creation, and the value in `BetterContinents.cfg` after an older version has run (chapter 2)
57	A baked town is gone from a world (neither drawn nor solid), or the whole world has the game's own terrain	The world was opened by the game alone, which opens it without its layer, or by Better Continents 0.10.3 or older, which stops at the layer's key and opens the world as one without Better Continents (for a world made from maps: the game's own terrain, as in trap 48)	Open a world with a layer with 0.10.4 or later only; neither of the others saves the layer away, so it comes back. To leave the mod: trap 65
58	`The baked layer <path> cannot be read (...): the world is made without it`	The town file breaks a rule of format 1 (appendix E)	`python3 bcp.py check <file>` names the problem; fix the compiler; `bc_bake load` adds the file to the world made without it
59	The whole town file refused: `... it is marked decor (flag 32), which only a Static or Copy entry can be, and its role is Live`	Decor set on a Live or Seat entry	Decor only on Static or Copy entries; the same prefab as decor and as a Live piece is two entries

#	Symptom	Cause	Fix
60	Players pick the town's flax, berries and mushrooms, and they grow back; or a kind meant as scenery can be picked	A consumable kind without palette flag 32 is placed once as real objects of the game (by design)	Flag 32 on the Static or Copy entries meant as scenery, collision None or Boxes; real ones placed earlier stay until their zone is reset or they are picked
61	A town stands 30 m under the ground	`y` was written as metres above the water; format 1's `y` is the game's world height (the water at 30 m), and so is the ground's base	`y = altitude + 30` in mm; `bcp.py check --zone ZX ZZ` prints the decoded heights
62	A town's door or station is taken away and made again with every new file	Its id changed, or it moved more than 5 cm or 1 degree: Live pieces are matched by (source, id)	Keep each Live piece's id from one file to the next
63	After a newer town file, a chest stands where the town has none, unprotected	A container that holds items is never taken away: it is an orphan	`bc_bake orphans`, then `bc_bake orphans destroy` (`items` for the ones that hold items)
64	Some baked pieces are not drawn; `bc_bake info` lists `Pieces the layer names that this game lacks`	A wrong prefab name, or a mod this machine does not have	Fix the name, or name a vanilla stand-in as a later candidate; collision Boxes stays solid without the mod
65	A base vanished after the mod was removed or the layer dropped	A baked piece exists only in the layer	Before: `bc_bake unbake world` (`all` for a town file's pieces), confirm, save. After a removal: install 0.10.4 or later again (the layer is still in the world); after a drop: the file kept under `BetterContinents/pre-drop/`
66	A world lost its layer after a rename, move or backup restore; or, with 0.10.4, `bc_bake undo` finds no undo file after a rename	The game's own rename and backup carry the settings file only while Better Continents is installed. Undo files are kept in `BetterContinents/bakes/<world name>-<world id>/`, a folder that keeps the name the world had when the folder was made; 0.10.4 looked only under the new name	Rename, move or restore only with Better Continents installed. Since 0.10.5 a renamed world finds its undo folder by the world's id (uid), and the log says `bc_bake: this world was renamed; its undo files are still in <folder>.`; with 0.10.4, finish and check changes (`bc_bake list`) before renaming
67	`bc_bake: the pieces in this area have N different sets of values (health, creator, ...) and a layer holds 65,535 for one bake. Bake a smaller area.`	One bake keeps at most 65,535 sets of values	Bake the area in parts
68	`bc_bake load: placements.bcp cannot be read (...)` after typing the confirm line a dry run printed (0.10.4)	0.10.4's dry run of `bc_bake load` with no file printed the file's name alone (`'bc_bake load placements.bcp confirm' loads it.`); typed back, a name without a folder is not looked for in the Directory	Since 0.10.5 the dry run echoes the words as typed (`'bc_bake load confirm' loads it.` when no file was named), and that line is typed back as it stands. With 0.10.4: `bc_bake load confirm`, or the full path; the tab's Confirm does it right
69	A big world's sea is calm past about 12 km from (0, -4,000), and the far sea flat in a storm, though the map's Deep North is far away	The game's water shader draws calm everything past its own circle: Better Continents 0.10.4 or older drew the game's water; or this world is no candidate for Better Continents' sea (a world of the game's size without maps, a GameTerrain world, a version 11 world)	0.10.5 or later; `bc_water` names the cause (chapter 13). A world of the game's size gets it from a biome map or a heat map
70	Waves everywhere, no calm sea at all, in a world with Better Continents' water	The calm sea is the world's own Deep North: the biome map has none over water (VALtima's world: `0.0 % of the world is Deep North (calm)`)	Paint Deep North over the sea where it must be calm (chapter 13); check the share in `bc_water`
71	On one machine only, boats rock on water drawn calm, or sit still on drawn waves	That machine draws the game's water (Own Water off, no shader support such as macOS, another mod's water shader), while the felt waves are the world's on every machine; or another mod patches the wave functions	`bc_water` on that machine: `the game's own water is drawn: <why>`, or `another mod, <name>, patches <method>`
72	A world with lakes or a clearing mask opens with vanilla terrain	It was opened by Better Continents 0.10.4 or older, which stops at DataKey 72 or 73 (`Unknown feature: 72`) and opens the world as one without Better Continents	0.10.5 or later on every machine; never let an older version save it (its new zones would be vanilla)
73	A lake is not built; `Lakes: lake '<name>': its point (<x>, <z>) is dry: ground <g> m, level <l> m`	No cell within 12 m of the point is under the level on the generated ground: the point is outside the basin, or the level is under the ground there	The point at the basin's deepest place, inside the box, and a level above the ground there; `lakekit.py check-lakes --heights`

#	Symptom	Cause	Fix
74	A lake ends in a straight wall of water; `... its water reaches the edge of its box - the basin is not closed there, so the shore is cut off at the box`	The box is too small for the level, or the rim has a gap on the builder's 4 m grid of the generated ground	A bigger box or a lower level; `lakekit.make_lake` closes it as `bc_lake at` does
75	`Lakes: <path>: line <n>: ...` warnings, or fewer lakes than lines	A line with fewer than 9 fields or a field that is not a number is left out; an empty box, a point outside the box or a level under 30.5 m is kept with a warning	Fix the line; numbers with a dot and no thousands separator; `level` is the absolute height (the sea at 30)
76	Trees, rocks and bushes still stand on the roads and lake beds a clearing mask marks	The mask acts only on zones generated after it is in the world; or the Tarn mod is installed (trap 78)	A new world from the folder; or `bc_lake regen confirm`, which regenerates every unprotected generated zone of the world
77	Grass shows through a lake, or grows on a cleared road	Grass is not vegetation: it follows the ground's paint	Paint lake beds and roads as dirt or paved in the paint map
78	Lakes and clearing do nothing; `bc_lake info` and the screen say `Better Continents' lakes are off while the Tarn mod is installed.`	Better Continents steps aside while the Tarn mod (`com.raveniron.tarn`) is installed; a player with Tarn sees dry hollows where the others see lakes	Remove Tarn from every machine; the world's lakes and mask were kept and come back
79	A mask refused: `not a TARNCLR1 file ...`, `... the clearing mask's pixels are ... under the 1 cm ...`, or too few bytes	The header was written big-endian or with another layout, the half-width is mistyped, or the bits are short of ceil(size^2 / 8) bytes	`TARNCLR1`, `<i4` size, `<f4` half-width, then `np.packbits`; `lakekit.py check-mask`
80	The cleared ground is mirrored or shifted against the roads	The rows were written south to north, or the half-width is not the maps' (World Size + Edge Size)	Row 0 is the north edge, each row west to east; the half-width your maps cover; test a road point with `bc_lake mask at`
81	`genloc` does nothing at all, or a world comes back with only its placed locations	The game's slot for the location run was left taken by a run that threw (the game's `genloc` checks only that slot); or the world was saved in the middle of a run	`bc genloc` (chapter 13) frees the slot and runs; stay in the world until its `done` line
82	`bc regen <words>` starts nothing and prints `bc regen does not know '...'`, `bc regen: '...' is not a biome.`, or says the Debug Reset Command setting holds a console command	A word it does not understand starts nothing (it removes objects); or `Debug Reset Command` is not empty, and a console command takes no filters	Fix the word (`bc regen help`); empty `Debug Reset Command`; run `bc regen dry ...` first (chapter 6)
83	Upgrade World's `upgrade ashlands` or `upgrade deepnorth` resets the wrong ground on a map with its own biome map or size	They choose zones by distance from a point of the game's own layout (more than 11,600 m from a point 4,000 m north or south of the centre), not by biome	`bc regen dry biome=AshLands`, then `bc regen biome=AshLands` (chapter 6)
84	After Upgrade World's `locations_restore` a location map pin is missing	The registry it builds did not hold the pin (without Better Continents' help `locations_restore <id>` removes that id's pins that nobody has visited)	`bc genloc pins`, which puts the pins and the start position back and generates nothing
85	Upgrade World's commands leave the outer ring of a big world out	Its World radius is 10,500 m by default: zones past it that are not generated yet are not listed	`[07 BetterContinents.Misc] Upgrade World Radius` On, or raise World radius in `upgrade_world.cfg`
86	With Expand World Data 1.79 or later: the world's size, biome names, lava and terrain are not applied, only warnings in the log	Better Continents from before the fix looked its classes up by their old names	A Better Continents with the fix; the log's `Applying compatibility: size yes, biome names yes, lava and terrain yes, minimap height yes` shows what it reached
87	With Expand World Size 1.44 or later, a save with wide-sector chunks is not loaded and a message names both mods	Expand World Size's save grid (wider than the game's) and Wide Sectors lay a save out differently	Remove Expand World Size, or set its save grid to the game's (`Save system size` 0 with a world no bigger than 16,384 m)

#	Symptom	Cause	Fix
88	Land edited in a terrain editor comes back wrong, or a rebuilt world lacks the Deep North or has doubled mountain noise	A Normalized range, a turned picture, biomes and locations that do not follow the ground, an export folder separated from its `export.cfg`, a map over 256 MiB for BetterWorlds	Chapter 9: "The traps of the editor round trip" and its trap table (11 to 15)

What each trap looks like, and the exact fix

The table says why; this part shows what the human sees or what the log prints, and the exact change. A quoted log line is Better Continents' own text (each carries `[BetterContinents]` in `LogOutput.log`), with `<...>` for the parts that are the world's own; a line that is the game's own says so. Where nothing is logged, the sample is what the human sees. A config line is shown under its section header; set it, then make a NEW world with a new name (trap 5) unless the fix says otherwise.

How to read a fix	It means
`[02 BetterContinents.Heightmap]` then `Heightmap Amount = 2`	that line under that section of `BetterContinents.cfg`
`bc ...`	the console (F5) in a loaded world, cheats on (Debug Mode, or `devcommands`), on the machine that runs it
`bc_export ...`, `bc_import ...`, `bc_twins ...`	the console, any player
`bc_bake ...`	the console, no cheats; a change runs where the world runs (an admin on a server adds `server`; chapter 12)
`bc_water`, `bc_lake ...`	the console, no cheats; `bc_lake`'s changes run where the world runs (an admin adds `server`; chapter 13)
`python3 lakekit.py ...`	the toolkit's lake and mask checker (`toolkit/lakes/`, chapter 13)
`python3 check.py ...`, `python3 bcp.py ...`	the toolkit (appendix A)

1. The drowned world. *Sample:* the human spawns in water; the large map is open sea with bare peaks; `check.py` lists Meadows, Plains and Black Forest near 0% dry. *Fix:* the waterline pixel is `0.30 / Amount` (0.15 at Amount 2); write heights with `v = (altitude / 200 + 0.30) / Amount` and re-run `check.py` until each lowland biome is above 90% dry.

2. The Amount the map was not drawn for. *Sample (log):* the dump says `Heightmap amount 1, blend 1, add 0, mask 0` for a map drawn for 2, and a heightmap that carries a record warns: `<file> was made by a world export for Heightmap Amount 2 and Sea Level Adjustment 0.5, but this new world reads it at Heightmap Amount 1 and Sea Level Adjustment 0.5, so its land is not at the heights it was made for. The export's export.cfg has the settings it needs.` *Fix:* `[02 BetterContinents.Heightmap]` then `Heightmap Amount = 2` (the Amount the map was drawn for), and record it on the map: `python3 png_extras.py record heightmap.png --set 2`.

3. A config change that did not reach the world. *Sample:* the creation dump shows the old value. *Fix:* quit Valheim, start it again, then New World. Since 0.9.0 Better Continents reads the file again when it changes, but the staging script still refuses to run while the game runs; keep that order.

4. Ashlands where there is none. *Sample:* ash rain, lava damage or Ashlands creatures on a meadow. *Fix:* heat exactly 0 off the Ashlands; `check.py` must print `heat > 0 on land that is not AshLands: 0`.

5. A PNG edited, the world unchanged. *Sample:* the island you moved is where it was. *Fix:* New World with a new name. For a test world only, `bc reload hm` (or `bm`, `all`) reads the picture into the loaded world (chapter 6).

6. Mountain locations missing at Amount 1. *Sample (the game's own log line):* `Failed to place all Hildir_cave, placed 0 out of 3 ...`. *Fix:* `[02 BetterContinents.Heightmap]` then `Heightmap Amount = 2`, with the map drawn for it and peaks above 250 m.

7. Water locations missing. *Sample (the game's own):* `Failed to place all FrozenShip01_DN, placed 0 out of 50 ...`. *Fix:* paint Deep North water as DeepNorth to 16 m deep and Mistlands water as Mistlands to 1 m (the generator's `label_shallows()`), never against another biome's shore.

8. Alt-biome hosts lost after labelling water. *Sample:* `altbiomes.py` counts 123 planted regions where you painted far fewer. *Fix:* label only the water that needs it; pull a multi-biome alt's colour 3 px back from water.

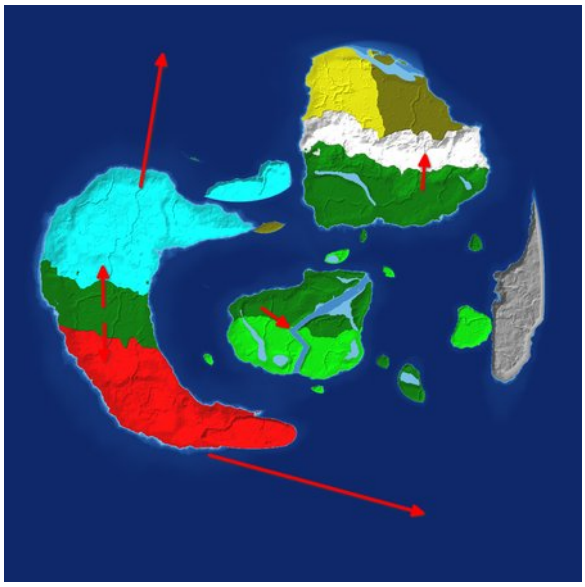
9. One-point alt-biome regions. *Sample:* `altbiomes.py` lists dozens of `Lantern` regions on Black Forest with an edge under 20. *Fix:* paint patches 2 px inside their own biome, and pull multi-biome colours back from neighbours.

10. A planted slot that reads as nothing. *Sample:* a planted region with `edge` under 20 in `altbiomes.py`, and no alt biome to be seen there in game. *Fix:* erase specks; paint whole features, several 12 m cells wide.

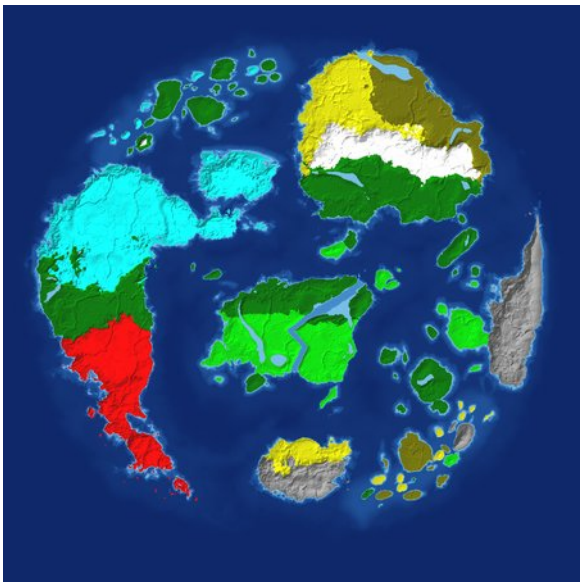
11. Relief that does not show. *Sample:* "reads too flat"; `relief.py`'s rms numbers barely moved. *Fix:* `ROLL_K = 1 / std` of the noise, and judge with `relief.py` against the previous version's log.

12. Tilted-plain coasts. *Sample:* every coast climbs evenly about 20 m over 500 m. *Fix:* a beach platform of 210 m and relief faded in from 125 m inland.

13. Islands parked in a row. *Sample:* the left of the two pictures that follow (map version 01). *Fix:* anchor the warp at each feature's centre.



`01-parked-before.png`



`06-archipelagos.png`

Map version 01, with the warp bug that parked the islands in a row (left), and version 06, the same layout after the fixes of chapter 4 (right).

14. A biome far from its continent. *Sample:* a patch of Plains beside the Deep North. *Fix:* leash each field to its unwarped ellipse plus slack.

15. A preview that ate the memory. *Sample:* the machine slows to a halt, or the tool stops with `crop 820.0 1000.0 1300.0 1480.0: give fractions of the map, 0..1, with x0 < x1 and y0 < y1`. *Fix:* crops are fractions, `python3 preview.py <dir> out.png -2 0.40 0.52 0.62 0.72`, under `(ulimit -v 6000000; nice -n 10 ...)`.

16. Stale location names. *Sample:* a Deep North wreck listed as `TarPit1`. *Fix:* name locations by their prefab asset path, as `locations.py` does.

17. A vanilla world, no warning. *Sample (log):* `[Saving][<name>] bWorldBeingCreated flag set, first time save of <name>, applying selected preset 'Disabled'`, then `[Saving][<name>]`

`BC is disabled for this world, skipping save`. *Fix:* `[07 BetterContinents.Misc]` then `SelectedPreset = From Config`, and **From Config** in the New World dropdown.

18. Two Better Continents DLLs. *Sample:* two files match `BepInEx/plugins/**/BetterContinents*.dll`. *Fix:* rename every one but the version you want to `.bak` (never overwrite: trap 19).

19. One copy corrupted every profile. *Sample:* `stat -c %h <dll>` prints more than 1 (a hardlink). *Fix:* rename the old file out of the way, then write the new one.

20. A world gone after a rename or copy. *Sample:* the world is missing from the list, or loads as vanilla. *Fix:* never touch the human's saves; copy a world as its whole folder, `worlds_local/<World>/`, `BetterContinents` file included.

21. Weather flipping at sea. *Sample:* snow and Deep North weather in a band across the sea. *Fix:* Better Continents 0.8.1 or later; the log says `Deep North weather: EnvMan asks IsDeepnorth at the camera's x and z (this world has a biome map or high terrain)`.

22. Creation hangs on the progress bar (0.7.x). *Fix:* Better Continents 0.8.0 or later; on older versions put every biome on the map.

23. `GetBiome error A -> B pos -> x, y` **once a second.** The game's own line, harmless. *Fix:* none.

24. A Wolf Mountain where none was painted. *Fix:* `[08 BetterContinents.AltBiomes]` then `Overrides = Wolf Mountain: enabled=false`.

25. A colour that planted nothing. *Sample (log):* `Alt-biome map altbiomemap.png: <N> pixels in <K> colours match no legend colour (within 20) and are treated as unplanted. Most common:` and the colours. *Fix:* paint the legend's exact colour, or add a line for it to `altbiomemap.txt`; plant an alt biome only on its own base biomes (appendix C).

26. A unique location placed once. *Sample (the game's own):* `Failed to place all BogWitch_Camp, placed 5 out of 10 ...`. *Fix:* none; one is spawned and the rest removed.

27. Alt biomes differ between server and clients. *Fix:* one world file everywhere and Better Continents 0.8.1 or later on every machine; the server's placement is pushed to each client.

28 and 29. The wrong decompile. *Sample:* a claim about the game that the running game does not bear out. *Fix:* `md5sum` the reference `assembly_valheim.dll` against the installed game's, and use the current build's folder.

30. A minimap that reads as forest. *Fix:* Better Continents 0.8.0 or later.

31. A version 11 world. *Sample:* `bc info` begins `Version 11`, and a world of another size has no line `The maps span World Size and Edge Size: <N> m across, and the world is laid out to that size`. *Fix:* `[00 BetterContinents.Debug]` then `Override version =` (empty), and a NEW world From Config.

32. A big world on the game's sectors. *Sample (log, at load):* `Sectors: this world reaches past 16,350 m. Unless its save was made with wide sectors, it keeps the game's own sectors while Wide Sectors is Auto: what lies beyond 16.4 km shares one sector. Wide Sectors On converts it when it loads (the game saves a copy of the world first).` *Fix:* for a new world, nothing (Auto makes it wide). For this one, back up `worlds_local/<World>/`, then `[07 BetterContinents.Misc] Wide Sectors = On`, load it once, and set `Auto` again.

33. Twin trees. *Fix:* `bc_twins` counts them; `bc_twins remove` shows what would go; `bc_twins remove confirm` removes them, after a backup (on a dedicated server, `bc_twins server remove confirm`).

34. A pin not placed. *Sample (log):* `Location map: <name> at <position> was NOT placed: its zone <zone> already holds <other> (the game keeps one location per zone).` or `The`

`start position override was NOT applied: ...`. *Fix:* move the pin more than 64 m (a zone) from the other site.

35. A boss in the sea. *Sample:* `check.py`'s list of pinned sites shows a negative height or the wrong biome. *Fix:* move the pin to a legal pixel near the intended site.

36. A compact world in Better Continents 0.9. *Sample (0.9's log):* `Unknown feature`, and vanilla terrain. *Fix:* for a world 0.9 users open, `[07 BetterContinents.Misc]` then `Compact Maps = Off`.

37. Heightmap Alpha does nothing. *Sample (dump):* `Heightmap alpha: 8-bit heights, the alpha unused (a world made by an older Better Continents)`, or no alpha line at all. *Fix:* `[02 BetterContinents.Heightmap]` then `Heightmap Alpha = true`, a heightmap written by `png_extras.py gray-alpha`, and a new world; the dump then says `Heightmap alpha: full precision, blended with the game's own terrain by the alpha`.

38. The checkers disagree with a world of another size. *Fix:* compare default-size worlds, or change the checkers' constants in a copy (appendix A).

39. Trees left floating after a `bc` change. *Sample (console):* `Zone regeneration: resetting 1,204 of 1,500 generated zones; 296 left alone (...)`. *Fix:* test away from bases and fields; make big height changes before building.

40. A square of odd water. *Fix:* Better Continents 0.10 or later; on older versions go 300 m away and back.

41. A new world played by older rules after 0.9 opened it. *Fix:* restore the copy made before; keep every machine on one version.

42. Huge forest patches. *Sample:* forests and clearings five times the game's size. *Fix:* `[04 BetterContinents.Forest]` then `Forest Scale = 0.5`, and a new world.

43. Trees on black. *Sample (dump):* `Forestmap multiply 0, add 1`. *Fix:* `Forestmap Multiply = 1` and `Forestmap Add = 1` under `[04 BetterContinents.Forest]` where black must be a clearing.

44. A map refused. *Sample (log):* `Cannot use texture <file>: it is 20000 x 20000 pixels, and the largest map Better Continents reads is 16384 x 16384. Scale it down (...)`. *Fix:* scale the map to 16,384 px or less, or set `[07 BetterContinents.Misc]` `Max Map Size = 16384` where the message names that setting.

45. Gigabytes to read one map. *Sample (log):* `<file> is a picture of a kind that is read whole (<size> x <size>: <N> MB at once) ...`. *Fix:* save it as a plain PNG: not interlaced; 8- or 16-bit grey, grey and alpha, 8-bit RGB or RGBA, or a palette.

46. Stairs on gentle slopes. *Sample:* steps of 0.25 m on a slope at Amount 81. *Fix:* the smallest Amount that holds the highest peak, and fine heights (chapter 11, recipe 4).

47. Fine heights not used. *Sample (log):* `Fine heights: heightmap-fine.png was made for another heightmap.png (its record names CRC-32 <A>; this one is <B>). If heightmap.png was edited, make heightmap-fine.png again or delete it.` *Fix:* make the pair last, `python3 fine_ref.py write ...`, and check with `python3 fine_ref.py check <folder>`.

48. A fine-heights world opened by 0.10.0 to 0.10.2. *Sample (their log):* `Failed to load settings from <file>: a map saved in format 2, which this version of Better Continents cannot read`, and vanilla terrain. *Fix:* 0.10.3 or later.

49. Two of every chest. *Sample (an older version's log):* `Unknown feature: 68`. *Fix:* restore the backup; never open a wide world with an older version.

50. High ground without the high-world rules. *Sample (log, High Terrain Off):* `High terrain: High Terrain is Off for this world, whose heightmap is read at Heightmap Amount <A> (...): the game's own height rules stand, so there is no grass above 500 m, ...`. *Fix:*

`[02 BetterContinents.Heightmap]` then `High Terrain = Auto` (or `On` for a world read at 5 or less with ground above 500 m), a new world, and 0.10.3 or later.

51. Mountains twice as high as drawn. *Sample (dump):* no `Heightmap overrides ALL` line. *Fix:* `[02 BetterContinents.Heightmap]` then `Heightmap Override All = true`.

52. Snowy mountains and soil-less farms far from the north. The game's terrain shader; no fix. Lay out a big world knowing where the circle is.

53. "Please wait" for half a minute. *Sample (log):* `New world <name>: making its settings from 'From Config' while the game goes on`, then `New world <name>: its settings are ready, in <N> s`. *Fix:* none; tell the human first.

54. The red sea in the wrong place. *Fix:* Better Continents 0.10.3 or later (per pixel since 0.10.5), and heat drawn over all the sea that should be red.

55. Boats rocking on calm water. *Fix:* 0.10.5 or later, where drawn and felt follow the world's own Deep North; on one machine only, trap 71.

56. A tall world far too low. *Sample (dump of a world made by 0.10.2 or older):* `Heightmap amount 5, blend 1, add 0, mask 0` where the config said 81. *Fix:* make it with 0.10.3 or later, and check `Heightmap amount 81` in the dump.

57. A layer opened by an older version. *Sample (the older version's log):* `Failed to load the save file. Unknown feature: 70`. *Fix:* quit without building anything; open the world with 0.10.4 or later. Before a human downgrades, removes the mod, or lends the world to someone without it: trap 65.

58. A town file refused. *Sample (log):* `The baked layer <path> cannot be read (zone 7,8, record 0: a Live record has no id): the world is made without it. 'bc_bake load' adds a layer to a world later.` *Fix:* `python3 bcp.py check <file>` gives the same words; fix the compiler (every Live record needs an id), write the file again, and `bc_bake load` it.

59. Decor on a Live or Seat entry. *Fix:* a Live entry is a real piece with an id and a Seat a local chair; neither is scenery. Set flag 32 only on Static and Copy entries.

60. Real berries in a town. *Sample (log):* `Baked placements: RaspberryBush is a consumable (a player picks it up or harvests it): its records are placed once as ordinary objects of the game when a zone generates, never baked, drawn or protected.` *Fix:* if the human wants them as scenery, flag 32 on their entries (chapter 12). In one test of a town file with decor, 581 consumable records were still placed as real objects because their entries lacked the flag (VALtima's decor walk, 8 October 2026).

61. A sunken town. *Fix:* write `y` and the ground's base as the game's world height; check with `bcp.py check --zone`.

62. A Live piece made again. *Sample (the dry run of `bc_bake load`):* `The live pieces would follow: <N> replaced, <M> kept.` where the town did not move those pieces. *Fix:* a compiler keeps a table of ids by building and piece, and reuses it.

63. An orphan. *Fix:* by design: a player's things are never destroyed. Remove it with `bc_bake orphans destroy items confirm` only when the human says so.

64. A missing piece. *Sample (log):* `none of BFP_Shelf1, piece_chest_warderobe is in this game: those baked pieces are not drawn`. *Fix:* the right name, or a stand-in.

65. The mod removed without unbaking. *Fix:* see the table; say this to a human who asks how to uninstall.

66. Rename, move, restore. *Sample (log, 0.10.5, after a rename):* `bc_bake: this world was renamed; its undo files are still in <folder>.` *Fix:* as every Better Continents world (trap 20). The undo files stay in the folder of the name the world had when it was made; since 0.10.5 a renamed world finds that folder by its id, and `bc_bake undo` works as before.

67. Too many sets of values. *Fix:* a smaller radius, or a box.

68. `load ... confirm`**.** *Sample (0.10.5,* `bc_bake load` *with no file):* `bc_bake load: a dry run, nothing changes. 'bc_bake load confirm' loads it.` *Fix:* type that line back as it stands. With 0.10.4: `bc_bake load confirm`, or `bc_bake load <full path> confirm`.

69 to 71. The sea. *Sample (*`bc_water`*):* `Water: the game's own water is drawn: <why>.`, with *why* one of `this world's sea is the game's`, `Own Water is off`, `another mod has replaced the game's water shader (...)`, `the water shader is not supported on this graphics device`. *Fix:* chapter 13, part 1.

72. A lake world in an older version. *Sample (the older version's log):* `Unknown feature: 72`. *Fix:* quit without building anything; open the world with 0.10.5 or later.

73 to 75. A lake that is not what was meant. *Sample (log):* `Lakes: lake '<name>': its point (<x>, <z>) is dry: ground <g> m, level <l> m`, or `Lakes: <path>: line 6: 4 fields where a lake has 9 (name;x;z;level;minX;minZ;maxX;maxZ;depth): left out`. *Fix:* `python3 lakekit.py check-lakes lakes.txt --heights <grid.npy> --half-width <m>` before the world is made.

76 and 77. Clearing that did not clear. *Fix:* a new world; paint the ground the mask clears.

78. Tarn installed. *Fix:* remove Tarn; nothing is lost.

79 and 80. A mask refused or misplaced. *Fix:* `python3 lakekit.py check-mask clearmask.bin <x> <z>` on a road point and a point beside it.

81. `genloc`**.** *Sample (real):* `Genloc: starting. 33 location(s) now: 33 placed (kept, with every zone that is generated), 0 not placed ...`, then `Genloc: done in 23.5 s: 33 -> 12315 location(s), ...`. *Fix:* `bc genloc`; stay until `done`.

82 to 85. Zone and location commands. *Sample:* `bc regen does not know 'biom='. bc regen [dry] [biome=<names>] ...`, or `bc regen: the Debug Reset Command setting holds a console command ('zones_reset start'), which 'bc regen' runs, and a console command takes no filters. Empty that setting to use them.` *Fix:* the words and the order of chapter 6 (`dry` first).

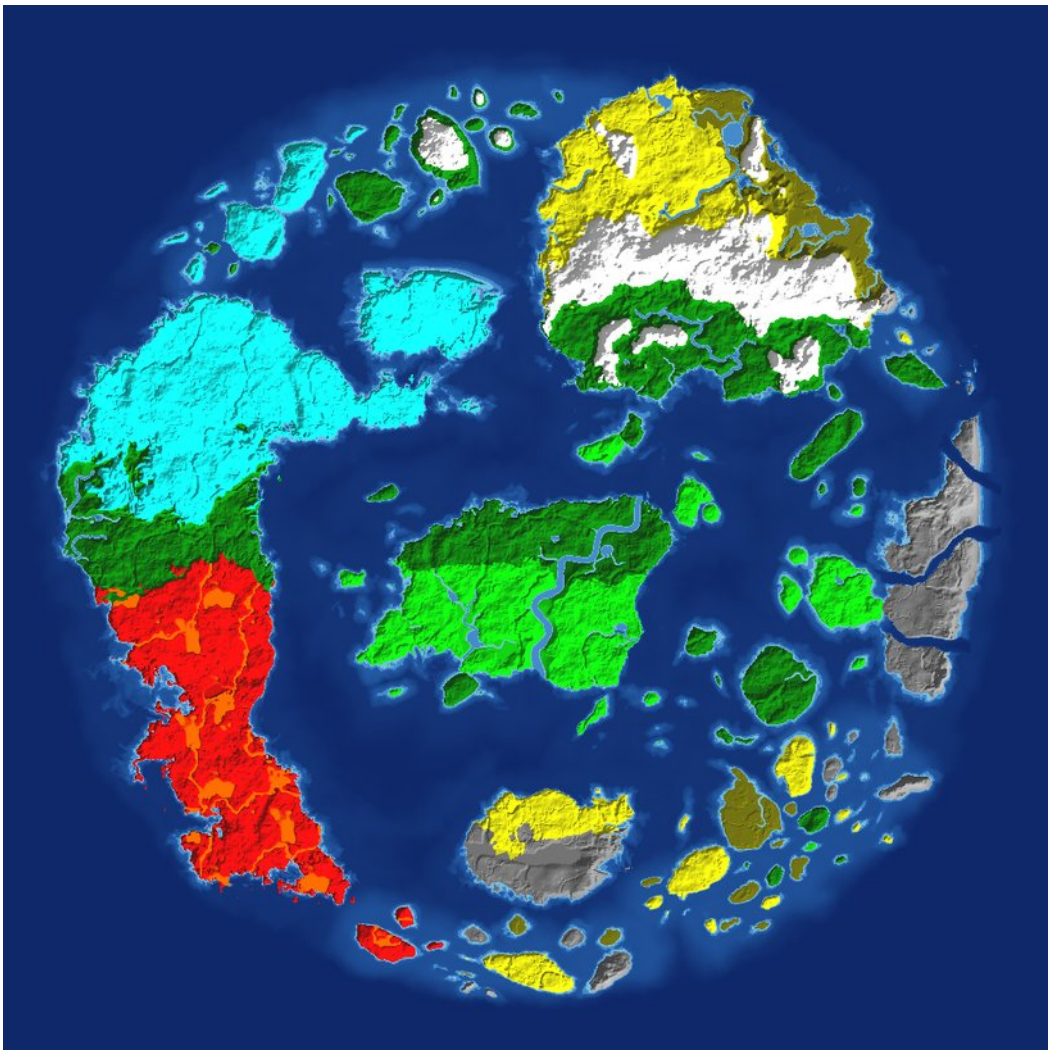
86 and 87. Jere Kuusela's mods. *Sample (log):* `"Expand World Data" 1.79 detected. Applying compatibility: size yes, biome names yes, lava and terrain yes, minimap height yes.` (a *no* is the part that does not work); `Sectors: this new world reaches <m> m, and Expand World Size's save grid (<n> zones a side) files its objects, so it is not made with wide sectors.` *Fix:* as the table says.

88. The editor round trip. *Fix:* chapter 9.

> **Trap:** The one that repeats: staging while the game runs (3). It is silent, it looks like a generator bug, and it wastes a world. The staging script's refusal is the guard; do not work around it.

8 A validated build, end to end

Map version 13 of ProximaMaxi, built by the human as the world `ProxEra01` on 23 September 2026 and approved ("looks good on my end"). These are the numbers a good round produces; compare yours with them.



Map version 13 as `preview.py` renders it: the 21 km square, north up. The crescent in the west (Deep North on its northern arm; the Ashlands arm to the south-west with burning isles and lava), the spawn continent at the centre with the starter river, the second continent to the north-east with its 250 m range, the Mistlands chain on the eastern edge, a Plains and Mistlands landmass to the south, two archipelagos and four rim chains.

> **Check:** These are the numbers of a real build made by Better Continents 0.8.1, from a map authored for Heightmap Amount 2. They still show what a good round looks like, because the checkers model the game's rules, not the mod's version. A world that 0.10 makes From Config from the same folder is a new world (version 12), and prints a few things differently: the settings dump begins `Version 12` (it said `Version 0` in this log), and every pin sits exactly on its pixel, the middle of its 3 x 3 block, where 0.8.1 and 0.9 picked one pixel of the block at random and used its corner (a pick up to a pixel from the middle, a corner up to a pixel south-west of where the other maps put it), so site coordinates move by up to about two pixels (`Override version = 11`, set before the world is made, gives 0.9's reading back). The staged config is the same: it always stated `Heightmap Amount = 2.0`, which matters more now that the mod's default is 1. What 0.10.3 added changes nothing here: 2048 px maps at the

> default size and Amount 2 use none of it (maps over 8,192 px, Amounts above 5, fine heights, worlds past 16,350 m).

The inputs

What went into the world, so that the round can be repeated (chapter 11, recipe 2, is the same setup as a recipe):

```
fire-and-ice/                    (2048 x 2048 each)
  heightmap.png    16-bit grey   for Amount 2: the waterline is pixel 0.15, white 340 m up
  biomemap.png     8-bit RGB     biomemap.txt: the ten default colours, "Meadows: 00FF00" ...
  heatmap.png      16-bit grey   exactly 0 off the Ashlands; at most 0.160 (check.py: max heat)
  forestmap.png    16-bit grey   read additively (Forestmap Multiply 0, Add 1)
  lavamap.png      16-bit grey   pools and streams, lava where 0.6 or more
  locationmap.png  8-bit RGB     nine 3 x 3 pins; locationmap.txt "StartTemple: 255,0,0" ...
  altbiomemap.png  8-bit RGB     altbiomemap.txt: 19 colours, "Mushroom: FF969C" ...
```

The config was appendix B's: `Directory` at the folder, `Heightmap Amount = 2`, the forest pair, `SelectedPreset = From Config`, and `Overrides = Wolf Mountain: enabled=false` under `[08 BetterContinents.AltBiomes]`.

What a good round looks like, and a bad one

Check	This build (good)	A bad round shows	Trap
Land share (`check.py`)	27.7% of the image	far less land than the design: the waterline or the Amount is wrong	1, 2
Lowland biomes dry	Meadows 94.9%, Black Forest 96.3%, Plains 98.2%, Swamp 90.7%	a lowland biome near 0% dry, or a mean height within a metre or two of the water	1
The three invariants	`dry ground labelled Ocean: 0`, `heat > 0 on land that is not AshLands: 0`, `AshLands pixels with no heat: 0`	any of them above 0	4
Mountain height	maximum 252.9 m	under 150 m: no Moder site	6
Planted alt biomes	23 regions, each on its base biome, all 32 alt biomes reach their minimum	many more regions than you painted, or regions with an edge under 20	8, 9, 10
Locations (`locations.py`)	every progression site at full count, `placed nothing:` only `Mistlands_Giant2`	a boss, Hildir's cave or the frozen ships short	6, 7
The game's log	`Genloc duration: 18089.898 ms`, 183 locations, 34 `Failed to place all` lines, none a progression site	`applying selected preset 'Disabled'`, `Heightmap amount 1` for a map drawn for 2, a `was NOT placed` line	17, 2, 34
The replay	148 of 170 lines exactly as the checker said, the rest within a handful	a location far off the checker's count: a modelling gap to read before changing the map	

> **Check it:** at every level, the last two rows decide. If you can only talk, ask the human for the log lines of chapter 6, step 10, and compare them with this chapter's.

The generator (gen.log)

```
  79 islands with satellites, 74 in archipelagos
  terrain 35s
  rivers: 65 mouths, 144 reaches, 43 km of water, 29 km of dry gully, 16 lakes (33 basins offered)
  shallows: 65277 px of Deep North and Mistlands water given the biome of their coast
  alt Mushroom                34795 px
  ... 22 more alt lines (Lantern: global biome, every piece kept) ...
  StartTemple          -> world (   -108,   -2201)  [c1meadows]
  ... eight more pinned sites ...
done 2048
```

check.py

```
2048x2048   land 27.7% of the image   (reference world Era02: 33.8%)
highest 253 m   deepest -60 m

biome         % map  % dry   mean    min    max
Ocean         70.13    0.0  -46.2  -60.0   -0.0
DeepNorth      6.69   79.2   64.6  -16.0  131.3
BlackForest    6.41   96.3   20.2   -8.2   80.3
AshLands       3.79  100.0   43.7   -1.2   78.4
Plains         3.17   98.2   20.1   -7.3   67.1
Meadows        3.04   94.9   14.5   -6.8   35.6
Mountain       2.72  100.0  132.7   39.5  252.9
Mistlands      2.60   93.6   23.1   -1.0   98.3
Swamp          1.44   90.7    5.8   -7.8   64.3

dry ground labelled Ocean: 0
heat > 0 on land that is not AshLands: 0
AshLands pixels with no heat: 0
max heat 0.160

  StartTemple                     Meadows        16.0 m     2203 m from centre
  Dragonqueen                     Mountain      181.4 m     6015 m from centre
  Mistlands_DvergrBossEntrance1   Mistlands      25.9 m     7976 m from centre
  ... (nine sites)
```

Deep North is 79% dry because its water to 16 m carries the biome (the frozen ships); the other sub-100% figures are rivers and lakes. Swamp's mean of 5.8 m is the bog Bonemass needs.

relief.py

```
biome        rms dh 50 m   rms dh 250 m   (metres)
BForest          8.05        25.36
DeepNorth       12.08        25.39
Ashlands         9.07        17.65
Plains           6.82        19.48
Mountain        16.91        53.06
Meadows          4.02         9.07
Mistlands        6.11        17.78
Swamp            9.12        31.89

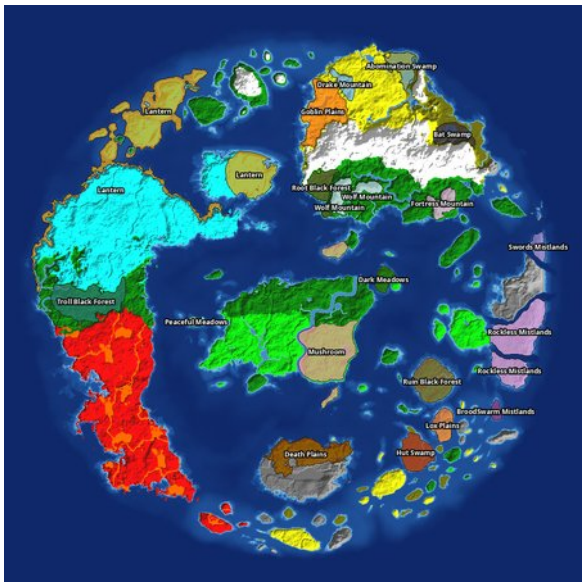
mean height (m) by distance from the water, 50 m bands from the shore outward:
Meadows     5.5  11.8  15.6  17.7  18.5  19.4  20.6  21.6  21.9  21.5 ...
Swamp       2.9   5.9   7.7   8.8  10.0  10.6  13.2  14.1  14.6  12.0 ...
```

altbiomes.py

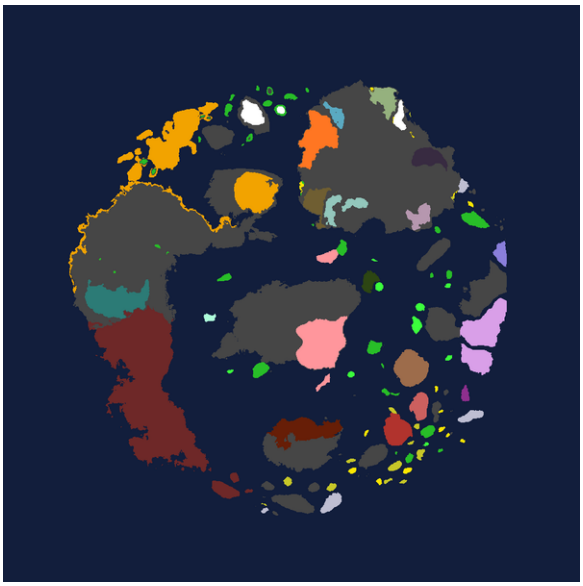
```
309 sectors; 295 are land; 23 planted
  AshLands (global): edge 4010, dist 7583 m
  DeepNorth (global): edge 6335, dist 5589 m

Planted regions (they ignore the placement rules):
  #303   Wolf Mountain            Mountain    at (  1709,  3508)  3903 m  edge  341  h 106.4
  #297   Lantern                  DeepNorth   at ( -5554,  4599)  7211 m  edge 5265  h  82.2
  ... 21 more ...

alt biome                 targets                    dist      edge min-max plant valid   total  reaches min
Mushroom                  Meadows+BlackForest         500 [ 66,199)   1-3       3    20    3.00  100.0%
Wolf Mountain             Mountain                   2000 [199,299)   1-5       2     2    2.46  100.0%
Rockless Mistlands        Mistlands                  2000 [ 99,599)   2-5       2     3    2.63  100.0%
... all 32 at 100.0%
```



13g-alt-labelled.png



13h-altbiome-regions.png

Left: the planted alt biomes labelled over the preview (`altmap_preview.py`). Right: the regions overlay `altbiomes.py` writes: legend colour = planted, bright = can host an alt biome at random, grey = cannot (too big, too small, too close to the centre, or an edge under water).

locations.py

```
170 locations from the game (... + the add-on lists), 16 more added by alt biomes (listed only)
location map sites registered first: Bonemass (4191, 6663), GoblinKing (1472, 6407), ...
  StartTemple                          1/1    P Meadows
  Eikthyrnir                           3/3    P Meadows
! Bonemass                             4/5    P Swamp      [zoneTaken 90, area 25324, biome
» 97, altitude 197688, terrainDelta 1937, similar 7776]
! SunkenCrypt4                      136/175  P Swamp      [zoneTaken 3839, area 25272, biome
» 163, altitude 182721, terrainDelta 97, similar 1812]
  Dragonqueen                          3/3    P Mountain
  DN_Bossroom                          3/3    P DeepNorth
  Hildir_cave                          3/3    P Mountain
  StoneTowerRuins04                   50/50     Mountain
  MountainCave02                     120/120    Mountain
  FrozenShip01_DN                     50/50     DeepNorth
  ShipWreck01_DN                     170/170    DeepNorth
  LeviathanLava                      100/100    AshLands
! StoneTower3                          1/50     Plains     [... similar ...]

VERDICT
  placed nothing: Mistlands_Giant2
  progression and quest sites: StartTemple 1/1 Eikthyrnir 3/3 GDKing 4/4 Bonemass 4/5
» Dragonqueen 3/3 ...
```

Bonemass's reason counts say what the swamp is: of 60,000 attempts, 197,688 point tests failed on altitude (the bog is mostly above 2 m) and 25,324 zones on area; 4 altars placed, which is enough. The Goblintower group's `similar` count is the plains' 14 km^2 against 100 towers 512 m apart.

The game (`log-13-ProxEra01.txt`)

```
[BetterContinents] [Saving][ProxEra01] bWorldBeingCreated flag set, first time save of
» ProxEra01, applying selected preset 'From Config'
[BetterContinents] Time to load .../fire-and-ice/heightmap.png: 157 ms
[BetterContinents] Alt-biome legend altbiomemap.txt: 19 colours, 0 points.
[BetterContinents] Alt-biome map .../altbiomemap.png: 2048x2048, 329922 planted pixels, decoded
» in 74 ms.
[BetterContinents] Selected StartTemple for spawn position 0.4941406, 0.3955078  (+ 8 more)
[BetterContinents] Alt biomes: split 23 planted region(s) out of 22 region(s) (13 fully planted
» and reused), 227235 of 4194304 grid points planted.
[BetterContinents] Alt biomes (world load): mode Random, placement seed world seed, grid 2048 x
» 12 m to 10500 m (WorldEdge), 296 sectors, 63 with alt biomes (23 planted, 40 random), 23
» planted region(s)
[BetterContinents]   Mushroom [Meadows|BlackForest] 0/1-3 chance 0.2 +3 planted (counted toward
» the maximum): 20 of 110 regions qualify ...
[BetterContinents]   Wolf Mountain [Mountain] 0/0-5 chance 0.2 +2 planted (counted toward the
» maximum): disabled for random placement
Loading: Done. ... Genloc duration: 18089.898 ms, (... locations: 183 ...)
Failed to place all Bonemass, placed 4 out of 5 with 0 tries in 00:00:00.8604500  (1 of 34)
```

The 34 short lines, as `placed/quantity`:

```
Bonemass 4/5   SunkenCrypt4 135/175   Mistlands_DvergrTownEntrance1 64/120
Mistlands_DvergrTownEntrance2 42/120   BogWitch_Camp 5/10 (unique)   SwampRuin2 22/30
GoblinCamp2 118/200   StoneTower1 15/50   StoneTower3 1/50   StoneHenge4 4/5   TrollCave02 144/200
SwampHut5 11/25   SwampWell1 15/25   ShipWreck03 6/25   ShipWreck04 2/25   Runestone_Swamps 92/100
BigRockClearing 1/10 (unique)   Mistlands_GuardTower3_ruined_new 33/50
Mistlands_Lighthouse1_new 40/100   Mistlands_Viaduct2 63/150   Mistlands_Giant1 158/250
Mistlands_Giant2 0/85   Mistlands_Swords2 26/33   Mistlands_Swords3 2/33   NorthVillage 98/135
WoodVillage2 5/15   StoneTowerRuins07..10_sunk 0/10 each   SwampHut1_1 8/50   SwampHut2_1 1/50
SwampHut3_1 2/50   GoblinCamp2_1 0/5
```

The last nine are alt-biome add-on locations, placed only inside their alt biome's regions (Ruin Black Forest's sunken towers want Black Forest shallows the map does not label; a cosmetic gap). Everything mountain, Deep North, harbour, frozen ship and lava, and every progression site, is absent from the list: placed in full.

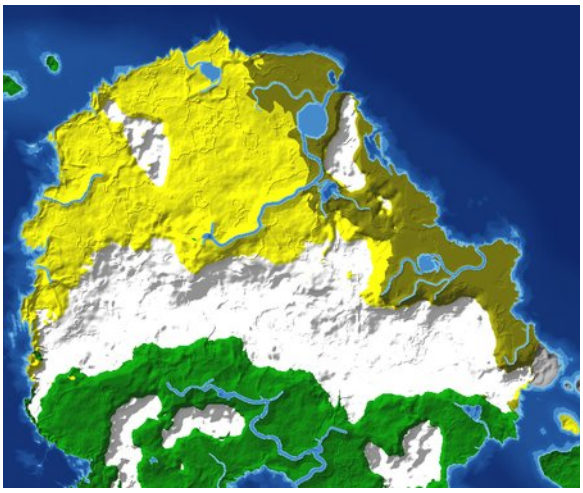
The replay (`locations.py --log`)

148 of 170 lines matched the game exactly; the other 22 were within a handful, with the checker mostly a little higher (GoblinCamp2 147 vs 118, TrollCave02 160 vs 144, SunkenCrypt4 136 vs 135) and sometimes lower (GuardTower3 19 vs 33, Lighthouse 36 vs 40). That is the checker's known error: Python's random numbers are not Unity's, and the vanilla forest and Mistlands vegetation noise are modelled, not reproduced. The verdicts agreed.

GPL-3.0 (Wubarrk, 2026-10-26); see the License page

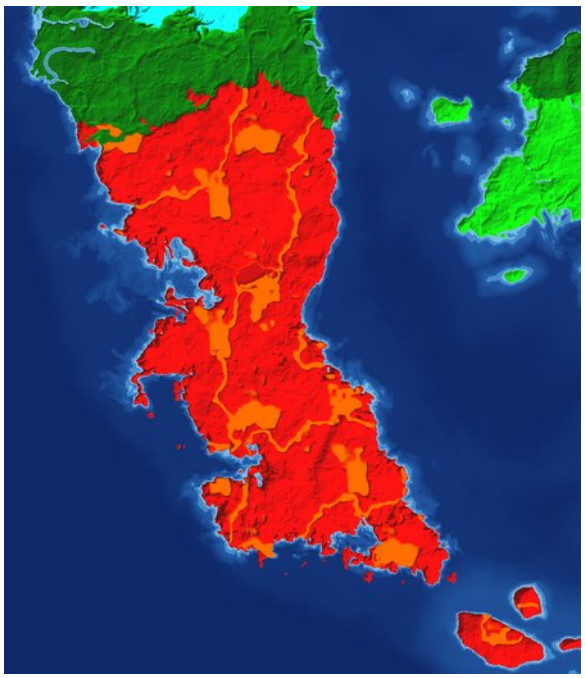


13d-spawn.png



13a-northeast.png

Close-ups from the previews the human saw: the spawn meadow with the starter river and Mushroom east of it (left); the second continent's range with Wolf Mountain's twin peaks and the Deep North isles beyond (right).



The Ashlands arm, with lava pools and streams drawn orange, the burning isles off its tip and the heat ramp's sea.

The whole round, from "the world reads too flat" to the approved world, took one working session: measure (`relief.py` was written for the purpose), change (`ROLL`, the ramps, the bog), regenerate, four checks, three previews shown to the human, the human's approval, publish, no restaging (the config had not changed), the human's new world, the log, the replay.

9 Export and import

A second way to start a map: instead of `gen.py` building one from nothing, `bc_export` writes one out of a world that already exists — vanilla or Better Continents, yours or someone else's — as exactly the kind of files chapter 2 already taught you to read. `bc_import` (or a New World preset the export makes for you) turns such a folder, edited or not, back into a new world. Both are console commands built into Better Continents (since 0.9.0); neither needs this workspace's toolkit. On a dedicated server both, like the other console tools (`bc_cache`, `bc_altbiomes`, `bc_twins`), answer the admin who ran them as well as the console that asked.

- **A real starting point.** Exporting a vanilla world gives you a heightmap, biome map and location map that Valheim's own generator built, with every location already placed on it — something to paint alt biomes or a terrain map onto, or to compare against `gen.py`'s output.
- **A second opinion on your own map.** Export the world you built from `fire-and-ice/` and read its `manifest.json` and `locationmap.txt` back, instead of trusting only a hand-copied log (chapter 6, steps 10 and 11).
- **A piece to paste.** A river mouth, an island, a crater: cut it from an exported world's `heightmap.png` and `biomemap.png` and paste it into `fire-and-ice/`'s, or the reverse. See "Cutting and pasting" below.

> **Rule:** `bc_export` encodes heights for Heightmap Amount 1 and Sea Level 0.5 unless told otherwise: Better Continents' default Amount, which is not `mapscale.py`'s 2 (chapter 2). For a map you mean to combine with this workspace's own maps, export at the workspace's Amount: `bc_export amount=2`, or set `[09 BetterContinents.Export] Default Heightmap Amount = 2`. Exports made by Better Continents 0.9.0 to 0.9.4 were at 2, and a config written by those versions that still holds 2 is set to 1 once, with a line in the log, when 0.10 first reads it: set it to 2 again afterwards if you want it. Whatever the Amount, the export records it in `export.cfg` and in the heightmap.

`bc_export`: what it writes

Console, any player, not a cheat command (it works for a guest on someone else's server too, within what that guest's client can see):

```
bc_export [size] [amount=1] [sealevel=0.5] [heatscale=10] [edge=auto|on|off]
          [forest=additive|exact] [fine=auto|off|4|8] [noheight] [nobiomes] [nolocations]
          [noforest] [noheat] [nocold] [noaltbiomes] [nolava] [nomoss] [nopaint] [nosources] [nopreset]
          [perpoint]
          [editor | editoronly] [eformat=tif,r32] [eunit=v|metres] [eamount=auto|<n>] [esize=<n>]
          [emasks=default|all|none|<list>] [eprotectpad=<m>] [eprotectfeather=<m>]
bc_export status | cancel | server <the same options> | help
```

The last two lines are the editor pack (below); `nocold` leaves out the cold map, and `cold` (the default) is accepted too. The generated usage line of appendix B lists the options up to `perpoint`; the editor ones are in `bc_export help`.

`size` is one number, 128 to 16384 px a side (the in-game window offers 1024 / 2048 / 4096 / 8192 / 16384; 4096 is the default). `[09 BetterContinents.Export] Largest Size` (4096, 8192 or 16384, the default) is the largest export one machine makes: the window hides bigger sizes, `bc_export` refuses them with a message that names the setting, and a larger `Default Size` is used as it. `amount=` takes 0.01 to 81 (chapter 2's tall worlds; at 81 white is 16,170 m of height, in steps of a quarter of a metre). `fine=` chooses the fine heights of chapter 2 (`heightmap-fine.png`, below): `auto`, the default, `off`, `4` or `8` bits; the window has the same choice. It is an option of each export: no config key holds it. In a loaded world the host also has `bc export run|status|cancel` in the `bc` tree. A status box (**Hud Hotkey**, F9 by default) and a window with an Export tab and an Import tab (**Window Hotkey**, F7) do the same thing with

buttons instead of typing, once `[09 BetterContinents.Export] Hud` is turned on; on a server, its **Hud** and **Allow Export** apply to every connected client too.

It writes into `<save data>/BetterContinents/<world>/export-<time>/`, a sibling of `worlds_local/`, not inside it. On Linux, for example:

```
~/.config/unity3d/IronGate/Valheim/BetterContinents/<world>/export-2026-09-24-14-05-33/
```

What the folder holds:

File	What it holds
`heightmap.png`	16-bit height, chapter 2's own encoding at the export's Amount and Sea Level, which a PNG text chunk records (`Heightmap Amount = 1; Sea Level Adjustment = 0.5`; the only text chunk in it). It is byte for byte the same whether or not the export writes `heightmap-fine.png`
`biomemap.png` + `.txt`	the biome, in Better Continents' own colours, and their legend
`locationmap.png` + `.txt`	one pixel per location, and its legend (see the client trap below)
`heightmap-fine.png`	the fine heights of chapter 2, with the record that ties them to this `heightmap.png` (`Fine Format = 1; Fine Bits = <bits used>; Heightmap CRC-32 = <the CRC-32 of the heightmap.png written>`), as `fine=` says: `auto`, the default, writes 4 bits where one grey step (200 x Amount / 65,535 m) is more than 1.5 cm, so from a Heightmap Amount of about 4.92 (an export at Amount 5 gets one), and none below that, where the folder is byte for byte as before; `fine=4` and `fine=8` write it at any Amount, `fine=off` never. The fine value is 0 under water (below 30 m) and where the ground rises a whole step or more to a neighbouring pixel, so the file is small: 5.7 MB beside an 86.7 MB `heightmap.png` at 16,384 px. A fine file of all zeros is not kept. After editing `heightmap.png`, delete `heightmap-fine.png` or export again
`altbiomemap.png` + `.txt`	every alt biome exactly as placed — only written if the exported world had Better Continents 0.8.1+ and alt biomes on it
`forestmap.png`, `heatmap.png`	forest factor, Ashlands heat. The heat map is exact for the game's own Ashlands test: above black exactly where the world is Ashlands (the old gradient line differed by up to about 100 m at the border)
`coldmap.png`	16-bit grey, the Deep North: above black wherever the world is Deep North, whose weather, snow and calm sea it decides. `Coldmap Scale` 10: cold = value x 10; inside the Deep North the value is the game's wave fade (a ramp over the 200 m at its border, 0 to 1) divided by 10, never below one grey step. A world whose own cold map is in use (and Sources on) gets that map copied byte for byte, with its own scale. Written unless `nocold`; the window has a **Cold** switch after Heat
`no_spawns_mask.png`	only when the world has a location exclusion mask and Sources is on: that mask, at the top of the folder (the name a `Directory` loads) and in `sources/`. No location is placed where it is above black (more than 1 % of white); the start temple and the location map's own dots are exempt
`world.json`	always: `{ "WorldRadius": 20000, "EdgeSize": 500, "Layout": "scaled" }`, one line, BetterWorlds' format. `Layout` is `scaled` for a settings version 12 export and `vanilla` for version 11 (a resized 0.9.x world's export). It has no `Sectors` member on purpose
`editor/`	only with `editor` or `editoronly`: the editor pack, below
`lavamap.png`, `mossmap.png`, `paintmap.png` (+`.txt`)	Ashlands lava, Mistlands ground cover, terrain paint — the guide teaches these; ProximaMaxi does not use them

File	What it holds
`placements.bcp`	since 0.10.4, only for a world with baked placements: its layer as it is (every bake, the town file's records, the record of who baked what, the ground and paint), so a world made from the folder has the same layer (chapter 12). The heights above are sampled without the layer's ground, so it is applied once in the new world
`lakes.txt`, `clearmask.bin`	since 0.10.5, only for a world that has lakes above the sea (`lakes.txt`) or a clearing mask (`clearmask.bin`; both experimental, Tarn's formats, chapter 13): the world's own, so a world made from the folder, in any of the three ways, has the same lakes and mask. With lakes, `export.cfg` also holds `Lake Depth Scale`
`sources/`	only if the exported world was itself Better Continents: its own maps as its settings kept them, for reference (not loaded back); a world with fine heights also gives `sources/heightmap-fine.png`
`export.cfg`	the settings that rebuild the world (a `[NN Group]` file, the same shape as appendix B's)
`README.txt`	a walkthrough for someone who has never done this: the ways to make a world from the export, what every file is, the numbers
`manifest.json`	the same numbers (`size`, `heightmapAmount`, `seaLevel`, `totalSize`...) as machine-readable JSON; `fineBits` and `fineEncoding` only when the export writes `heightmap-fine.png`. Keys that 0.10.7 added, all new, none changed (`format` is still `bc-export/1`): `settingsVersion` (12, or 11 when the maps span other than World Size + Edge Size give), `exportRevision` (2; absent means an older export: test this, not `format`), `worldSize`, `edgeSize`, `layout` (`scaled` or `vanilla`), `wideSectors`, `highTerrain` (0 Auto, 1 On, 2 Off), `cold` (`{ scale, deepNorthPixels, source: "sampled" or "the world's own map" }`, null with no cold map), `fileBytes` (name to bytes for the top-level files that existed when the manifest was written), `locations.pinGrid` (`"Corner"`), and `editorPack` with the pack

Every map is on by default (the `no...` switches turn one off), and a New World preset is made too unless you pass `nopreset` — see "Presets and `bc_import`" below. `bc_export status` and `bc_export cancel` work while it runs. An export holds about 100 MB of memory while it runs, at any size, because each map is written as it is sampled (measured on one machine: 8,192 px in 43 s, 16,384 px in about 3 minutes on 8 workers; a 16,384 px export of a plain procedural world of World Size 15,850 + Edge Size 500 was about 131 MB of PNGs, and noisy, hand-edited maps are larger). The preset is the costly part: building it took about 0.8 GB above the process's own memory at 8,192 px and about 1.4 GB at 16,384 px (measured on one machine, all nine maps; 50 MB at 2,048 px, 200 MB at 4,096 px), so **the export makes its preset itself only up to 8,192 px**; above that it leaves the preset to `bc_import`, run in the main menu, and its `README.txt` and `manifest.json` say so.

A real export, file by file

An export of a test world (a procedural Better Continents world of World Size 15,850 + Edge Size 500, exported on a dedicated server during 0.10.3's testing, at 2,048 px and Heightmap Amount 81, so with fine heights) holds:

```
export-2026-10-06-23-29-58/
  heightmap.png       2,820,905 bytes   16-bit grey, with its record
  heightmap-fine.png      4,843 bytes   8-bit grey, with its record
  biomemap.png          243,345 bytes   biomemap.txt        162 bytes
  locationmap.png        70,471 bytes   locationmap.txt   4,777 bytes
  altbiomemap.png        31,563 bytes   altbiomemap.txt   2,261 bytes
  forestmap.png           8,222 bytes   heatmap.png     276,837 bytes
  lavamap.png            27,317 bytes   mossmap.png     564,612 bytes
```

```
  paintmap.png          255,309 bytes   paintmap.txt        198 bytes
  export.cfg              4,418 bytes   manifest.json     3,570 bytes
  README.txt              7,135 bytes
```

The two records, PNG text chunks named `BetterContinents`: `heightmap.png`'s reads `Heightmap Amount = 81; Sea Level Adjustment = 0.5`, and `heightmap-fine.png`'s `Fine Format = 1; Fine Bits = 4; Heightmap CRC-32 = 75887BB8`.

Its legends are the formats chapter 2 teaches. `biomemap.txt` is the ten default lines (`None: 000000` to `Ocean: 0000FF`); `locationmap.txt` gives every location type a colour of its own (`AbandonedLogCabin02: 230,184,175`, `BogWitch_Camp: 250,206,150`, ...); `altbiomemap.txt` lists the alt biomes as the game placed them, a set to a colour:

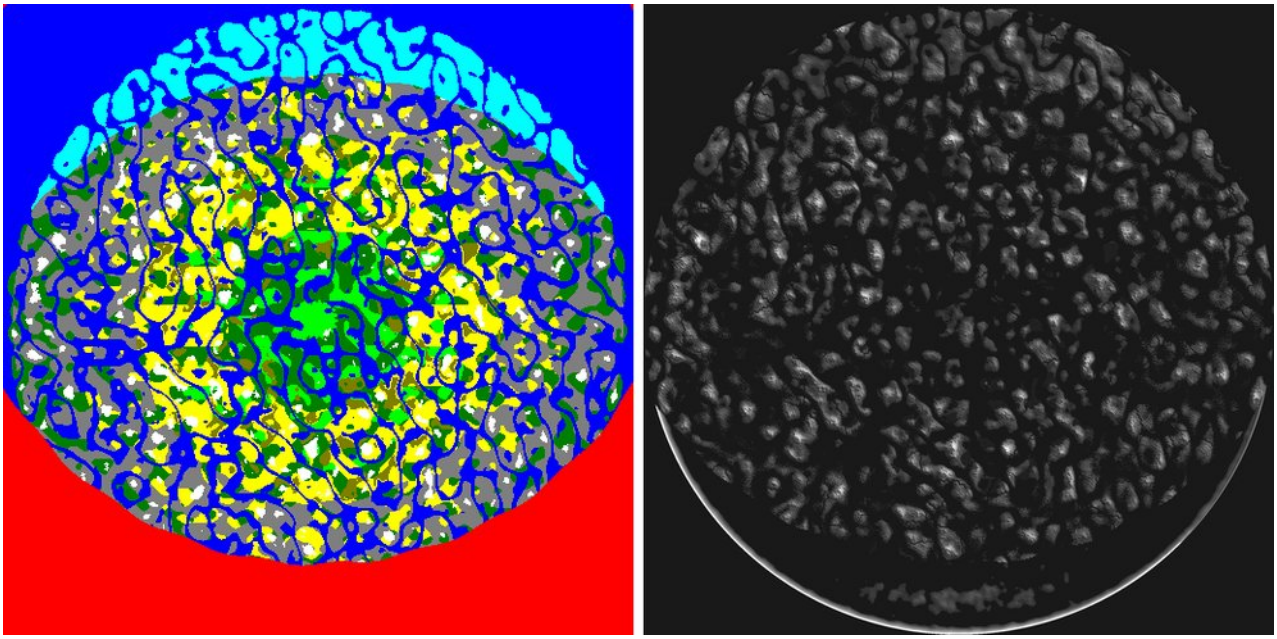
```
# Better Continents alt-biome map exported from "FineEx" on 2026-10-06 23:29.
# One colour per set of alt biomes, exactly as the game placed them; black is unplanted. Load it with
# Mode = PlantedOnly so the game adds none at random. A leading ! forces a name the content rules would refuse.
Rockless Mistlands: D99FE8   # 4 region(s)
Ruin Black Forest + Kalhygge Black Forest: 88FC5B   # 1 region(s)
```

`manifest.json` has every number a tool needs (an excerpt):

```
"size": 2048, "totalSize": 32700, "worldRadius": 15850, "totalRadius": 16350, "metresPerPixel":
» 15.974597,
"pixelToWorld": "x = (column / (size - 1) - 0.5) * totalSize; z = (0.5 - row / (size - 1)) *
» totalSize; row 0 is north",
"heightmapAmount": 81, "seaLevel": 0.5, "seaLevelAdjustment": 0,
"heightEncoding": "metres = (pixel / 65535 * heightmapAmount - 0.15 + seaLevelAdjustment) *
» 200",
"fineBits": 4, "fineEncoding": "signed offset, 1/256 step, format 1",
"waterlineValue": 0.003704, "floorMetres": -30, "ceilingMetres": 16170,
"locations": { "complete": true, "instances": 12358, "types": 176, ... },
"import": "bc_import FineEx/export-2026-10-06-23-29-58"
```

and `README.txt` ends with the numbers, in words:

```
Heights: metres = (v x 81 - 0.15 + 0) x 200, v = pixel / 65535.
One step of the 16-bit grey is 0.247 m. heightmap-fine.png refines it: its byte f for a pixel,
» read as a signed number (0 to 127,
then 255 is -1 down to 128 is -128), counts 256ths of a step, in multiples of 16 here, and v =
» (256 x pixel + f) / 16776960 takes the place of
pixel / 65535. f is 0 under water and where the ground rises a whole step or more to a
» neighbouring pixel.
Pixel (column c, row r) is world x = (c / 2047 - 0.5) x 32700, z = (0.5 - r / 2047) x 32700.
The location map alone uses c / 2048 and r / 2048 (Better Continents' location loader): up to a
» pixel off the rest.
```



That export's `biomemap.png` as written (left: blue Ocean, green Meadows, dark green Black Forest, olive Swamp, yellow Plains, grey Mistlands, white Mountain, cyan Deep North, red Ashlands) and its `heightmap.png` shaded for this page (right), each shrunk to 512 px.

When an export ends, the console and the log say:

```
World export finished in <N> s: <save folder>/BetterContinents/<world>/export-<date>-<time>
World export: <one summary line each>
World export: README.txt in the folder says how to make a world from it, in three ways.
```

> **Check it:** open `manifest.json` and read `size`, `totalSize`, `heightmapAmount` and `seaLevel`; they are the numbers every later step depends on. If you can run commands, `mapcheck.py` (chapter 11) prints each picture's kind and the two records, and says whether `heightmap-fine.png` still fits `heightmap.png`. If you can only talk, ask the human for those four numbers from `manifest.json`.

> **Trap:** an export made on a guest's own client only knows the locations the server has shown that player on the map so far — not the full set. `locationmap.png` is written anyway, with Skip Default Locations left off so the game places the rest, and both `bc_export status` and the export's own README.txt say so. Export on the host, or have an admin run `bc_export server ...` on the dedicated server, for a location map worth trusting.

> **Trap:** below 2048 px a pixel is wider than the game's 12 m alt-biome sector grid (chapter 3), so `altbiomemap.png`'s region borders can move by up to a pixel — the same reason `gen.py` never runs below 2048 (chapter 2). Export at 2048 px or more whenever alt biomes matter. That is for the 21,000 m square: a pixel is `2 x (World Size + Edge Size) / (size - 1)`, so a bigger world needs more pixels for the same, 5,462 or more for World Size 32,264 + Edge Size 500.

What `export.cfg` pins, and what else an export now carries

Everything below is additive: no file, key or name an export had is gone or different.

- `[06 BetterContinents.Maps] Coldmap Scale = 10` (or the world's own scale, when its own cold map was copied), written only when `coldmap.png` is. Cold is read back as pixel / 65535 x scale; the Deep North is wherever it is above 0.

- `[01 BetterContinents.Global] Mountain Noise = 0` and `Mistlands Noise = 0`, written with the heights. The heights already hold the mountain and Mistlands noise, so a world rebuilt from them must not add it a second time, whatever the player's own `BetterContinents.cfg` says. The price, to say once to the human: the noise is baked in at pixel size, so detail finer than a pixel (the noise's 2.5 m and 10 m features) is not carried; BetterWorlds and a Better Continents rebuild both start from those baked heights.
- `[01 BetterContinents.Global] Ashlands Water` and `Deep North Water`: the loaded world's values (a Better Continents world), so the rebuilt world floats and colours its seas the same way.
- `[02 BetterContinents.Heightmap] High Terrain = On|Off`: written when the world chose one, and now also when the world is Auto but its own Heightmap Amount and the export's sit on different sides of 5, so that Auto would decide differently.

> **Trap:** these pins live in `export.cfg`. A folder copied to somewhere that reads it as a plain `Directory` with no `export.cfg` (or an `export.cfg` you deleted) takes `Mountain Noise` and `Mistlands Noise` from the player's own config, default 1: the world then adds the noise to heights that already hold it. Keep `export.cfg` with the folder, or set both to 0 by hand.

The cold map and the exclusion mask are why a rebuilt world keeps the Deep North and its protected zones. Before, a world made from an export had the game's ring or the biome map for its Deep North. If the world's own cold map is off (`Coldmap Scale` 0), the export samples a new one; the original stays in `sources/`.

The editor pack, `bc_import edited`, and Gaea

A heavy terrain editor (Gaea, World Machine, Houdini) sculpts and erodes in 32-bit floats, which a 16-bit PNG cannot hold. `bc_export ... editor` writes the ground that way beside the usual maps, and `bc_import edited` brings the edited height back as a normal export folder. You, the agent, cannot run Gaea: the human does. You give them the exact lines and read the report they paste back. In short: *export with* `editor`*, edit the height, save it in* `editor/return/`*,* `bc_import edited`*, then* `bc_import <export>-edited`*.*

The options (all per export; no config key holds them)

Word	Meaning	Default
`editor` / `noeditor`	write `editor/`	off
`editoronly`	`editor`, and no biome, location, forest, heat, cold, alt-biome, lava, moss, paint or sources maps and no preset. `heightmap.png`, `export.cfg`, `README.txt`, `manifest.json` and `world.json` stay (the way back needs `heightmap.png`)	
`eformat=tif,r32`	the float files: `tif` is `height.tif` (32-bit float TIFF, any value), `r32` is `height.r32` (headerless floats clamped to 0 to 1). `exr` is refused ("not written yet")	`tif`
`eunit=v|metres`	`metres` also writes `height-metres.tif`, the same grid in true metres	`v`
`eamount=auto|<n>`	the Heightmap Amount of the pack's values, 0.01 to 81; `auto` is the larger of 2 and the world's own	`auto`
`esize=<128..Largest Size>`	pixels per side of the pack; above 16384 is refused ("comes in a later version")	the export's size
`emasks=default|all|none|a,b,c`	from `water,inland,slope,forest,biomes,index,soft,protect`; `default` is all but `soft`; `rivers` is refused	`default`

Word	Meaning	Default
`eprotectpad=<m>`, `eprotectfeather=<m>`	metres a protected disc grows past a location's radius, and the width of its soft edge	8, 32

`etile`, `eregion`, `epitch` and `eexrchannel` are known and refused ("comes in a later version"). `bc_export server editor ...` forwards the options unchanged: a dedicated server writes its own pack under its own save folder. The export window has the same choices in an **Editor pack** block below the map switches, and an **Editor return** section in its Import tab.

```
bc_export 4096 editor                          the pack with the defaults
bc_export 8192 editor eformat=tif,r32 emasks=all eunit=metres
bc_export 4096 editoronly                      heightmap.png and editor/ only
bc_export 2048 editor esize=8192 nobiomes      a 2048 px export whose pack is 8192 px
```

Trap: with `heightmap.png` and the pack both on, the ground is sampled twice (the pack has its own pass at its own size), so the export takes about the heightmap pass longer. `editoronly`, or `noheight editor`, keeps it to one. The window says the pack takes about as long as an export's terrain pass and about the same memory (an estimate); an offline stand-in measured 26 s and 56 MB more memory for a 16384 px pack with the default masks, on cheap stand-in terrain: a real world's terrain is slower.

What is in `editor/`

```
editor/
  height.tif  height.r32  height-metres.tif   the ground (the last two optional)
  editor.json             georeference and definitions, format "bc-editor/1"
  README-editor.txt       the steps with this export's numbers filled in
  probe.tif               64 x 64 float orientation test: high side north, a bright 5 x 5 patch
» in the north-west quarter
  masks/                  water.png, water-inland.png, slope.png, forest-density.png,
» biome-<Name>.png,
                          biome-soft-<Name>.png, biome-index.png (+ biome-index.txt: index:
» name: R,G,B), protect-locations.png
  return/                 empty: the editor's result goes here
```

`editor/` is never read by the game or by a `Directory` load. The existing maps are listed in `editor.json`'s `referencedMaps`, not copied. The masks:

Mask	Bits	Definition
`water`	8	255 where the ground height is under 30 m, else 0
`water-inland`	8	the same, where the biome is not Ocean: lakes, rivers, marsh
`slope`	16	`round(slope_degrees / 90 * 65535)`; slope = atan(hypot(gx, gz)) from the float heights as written (floor included), central differences over one pixel pitch, one-sided at the border
`forest-density`	16	`round(density * 65535)`, the game's forest factor normalised 0 bare to 1 densest (absolute, unlike `forestmap.png`, which is a difference)
`biome-<Name>`	8	255 where the point biome is that one; a file for each biome the zone corners and a coarse grid of points found, so a biome of a few pixels may have none (counted in the export's notes; `biome-index.png` still has it)
`biome-index`	8	the biome's index; `biome-index.txt` lists them
`biome-soft-<Name>`	16	`round(weight * 65535)`: how much of the ground's height blend the biome has (the four zone-corner biomes with the smoothstep weights; a biome with several corners gets the sum). Needs the zone blend
`protect-locations`	8	255 within (the larger of a location's two radii + pad), falling linearly to 0 over the feather; the maximum over all locations. Needs the locations: host or dedicated server (a client knows only the map-icon ones, and the export says so)

The value and the grid

```
fh     = h / 200                                  h = metres at the pixel (zone blend and edge drop-off applied)
v      = (fh + 0.15 - SeaLevelAdjustment) / amount        amount = the pack's Heightmap Amount
file   = max(v, 0)                                floor only; no upper clamp (height.r32 is also clamped to 1)
metres = (file * amount - 0.15 + SeaLevelAdjustment) * 200      SeaLevelAdjustment = Lerp(1, -1, seaLevel)
```

At amount 2 and Sea Level 0.5: v = 0 is -30 m, v = 1 is +370 m, the sea (30 m) is v = 0.15. The same arithmetic as `heightmap.png`, except: no edge drop-off undo (past the world's radius the pack includes the drop-off; the way back handles the ring); ground deeper than the floor and the corners of the square (the game returns about -400 m there) are flattened to 0 (`editor.json` `height.clippedLow`); ground above 1.0 stays above 1.0 in `height.tif` (`clippedHigh`; the export's summary warns and names the `eamount` that keeps v inside 0..1). The grid is every Better Continents map's: pixel i is at `(i / (size - 1) - 0.5) * totalSize` on both axes, file row 0 is **north**, columns run east. Not the cell-centre grid an image editor may assume.

TIFF (what the writer produces): classic TIFF 6.0, little endian, uncompressed, one 32-bit float sample, strips of up to 64 rows, Orientation 1, ImageDescription `BetterContinents: Height Encoding = bc-v; Heightmap Amount = 2; Sea Level Adjustment = 0.5; Orientation = north-up`. 16384 x 16384 is 1 GiB (the 4 GiB limit of classic TIFF; the writer refuses more); a file is `<name>.tmp` until it is whole. **R32**: W x H little-endian floats, top row first.

The way back

```
bc_import edited [<number> | <folder> | <world>] [unit=v|metres] [range=lo,hi]
» [amount=auto|<n>] [sealevel=auto|<s>]
                  [orient=auto|none|flipy|flipx|rot180|transpose|antitranspose|cw90|ccw90]
» [ering=keep|gaea]
                  [fine=8|4|off] [files=<path>] [size=<n>] [preset]
bc_import edited help | bc_import edited cancel | bc_import status
```

It finds the export as `bc_import` does (a folder with spaces needs no quotes; `files="C:\My Maps\height.tif"` does), runs on a worker, and writes a **new** folder `<export>-edited` (or `-edited-2`, ...) beside the export, which is never changed. It works in the main menu or in a world. It reads `editor/return/` (or `files=`): `height.tif`, `.tiff`, `.r32`, `.raw`, `.png` (16-bit grey) in that preference, else the only file of that kind there; several, none preferred: it asks for `files=`.

- **TIFF:** classic or BigTIFF, either byte order, **one** grey sample of 8, 16 or 32 bits (unsigned, or 32-bit float), strips or tiles, **uncompressed or Deflate** (compression 8 and 32946), predictor 1, 2 (8 and 16 bit) or 3 (floating point). LZW, PackBits, JPEG, several samples and planar are refused by name ("save it uncompressed or with Deflate, or as R32"). The TIFF Orientation tag is ignored (a note says so). **R32/RAW:** headerless, little endian, the side the square root of the length. EXR is refused. Square, 16 to 16384 px; a size other than the export's is fine (every map is read by its position on the world; the result's heightmap keeps the returned size, the other maps theirs).
- **Numbers:** `unit=v` (the default when `editor.json` exists): metres = (f x amount_pack - 0.15 + sla_pack) x 200; `unit=metres`: metres = f; `range=lo,hi`: metres = lo + f x (hi - lo). Without `editor.json`, give one of the last two. The result is encoded for the pack's amount and sea level by default; `amount=auto` picks the smallest of 2, 3, 4, 5, 6, 8, 10, 16, 32, 64, 81 whose top holds the highest edited ground, `sealevel=auto` raises the sea level in steps of 0.05 until the floor holds the lowest. Both are written to `export.cfg` and into the PNG's record. Ground outside the range is clipped, counted and said: `N pixels below X m and M above Y m were clipped: amount=auto keeps them.`
- **Precision:** `heightmap.png` (16-bit) + `heightmap-fine.png` (8 fine bits by default; `fine=4`, `fine=off`): within 1 mm of the editor's metres on gentle ground at amount 2 (about 0.02 mm in the test), within half a grey step (3 mm) on steep ground and anything under water, which get no fine bits by the usual rule.

- **Edge ring:** `ering=keep` (default) copies the pixels past the world's radius from the export's `heightmap.png` (it holds the height before the drop-off, which Better Continents applies again), only when the returned size equals the export's; `ering=gaea` works the pre-drop-off value out from the returned height.
- **Orientation:** `orient=auto` compares a 256 x 256 averaged copy of the picture turned each of the 8 ways with the pack's own `height.tif` (mean and spread taken out) and takes the best one if the runner-up's error is at least 4 times larger; else it stops and prints the eight scores (`cannot tell how the returned picture is turned ... say orient=...`). A flat world or a symmetrical picture cannot be told; say `orient=`. Without `editor/height.*` it is taken as it is and the report says "not checked". The turn is applied to the heights only.
- **Memory:** a returned file is read a row at a time, never whole: a 16384 px return took 11 s and about 55 MB more on an offline test world (measured there; a real world has not been measured).

The new folder holds `heightmap.png`, `heightmap-fine.png` (if any pixel needs it), every other top-level file of the export copied byte for byte (maps, legends, `placements.bcp`), a rewritten `export.cfg` (`Directory` is the new folder, the Heightmap Amount and Sea Level are the result's, the rest as exported), a new `manifest.json` (`modVersion`, `exportedAt`, `heightmapAmount`, `seaLevel`, `seaLevelAdjustment`, `heightEncoding`, `waterlineValue`, `floorMetres`, `ceilingMetres`, `clippedLow`, `clippedHigh`, `heightRangeMetres` rewritten; `heightmapSize` when the returned size differs; `preset` null; `editorPack` and `seconds` dropped; `import` names the new folder; `notes` gains a line; and `editedFrom { export, returnFile, returnSize, orientation, orientationFound, unit, range, edgeRing, editorJson, clippedLow, clippedHigh, restoredPixels, heightRangeMetres }`) and a `README.txt`. The pack and `sources/` stay with the export. Then `bc_import <export>-edited` (or `preset` in the first command) makes the New World preset, exactly as for any export.

Report lines to look for: `height.tif: 4096 x 4096 pixels, 32-bit float TIFF, uncompressed, strips.`, `Orientation: flipy ...`, `Heights 50 m to 70.5 m; encoded for Heightmap Amount 2 and Sea Level 0.5 (-30 m to 370 m).`, `Done in 1 s: <folder>`, `Next: bc_import <world>/<folder>`.

The traps of the editor round trip, in the order they cost

1. **A Normalized or Custom Output Range misreads every height.** Build with Output Range = **Raw**. The way back stops with `this looks like a Normalized build` when the return spans exactly 0..1 and the pack did not; then say `range=lo,hi` (metres) or `unit=metres`. The check is not made when the pack's own top is 0.98 or more, since then 0..1 is no surprise, or when `unit=` or `range=` was given.
2. **Orientation.** Never flip a file by hand. Have the human load `probe.tif` once (high side north, the bright patch at the upper left), and use `orient=auto`; when it cannot tell, `orient=flipy` and the like.
3. **The 16384 px cap and half a pixel.** Maps are read up to 16384 px, so an edit at 1 m over a 21 km world is stored at 1.3 m; and whether the editor counts a pixel as a point or a cell is open, so a feature can sit up to half a pixel off (0.64 m at 16384, 2.6 m at 8192, 5.1 m at 4096 in a 21 km world).
4. **Biomes, alt biomes, locations, forest, lava, moss and paint do not follow the ground.** The location map is copied byte for byte, never transformed. A boss altar is still placed where its dot is, under a mountain the human raised. Repaint the PNGs (start from `biome-index.png`) or keep the edit away from `protect-locations.png`. The editor's own masks do not come back in this version.
5. `bc_import edited` **makes a NEW folder.** The export is never changed; the world is made from `<export>-edited` by the usual `bc_import <folder>` and a New World (the rule above).
6. **Compression.** Uncompressed or Deflate only; LZW, PackBits and JPEG are refused by name; save as R32 if in doubt.

7. **Lakes above the sea** (Tarn style) are not in the pack's water masks, which are the game's 30 m plane.

> **Human:** none of the Gaea steps has been checked against a real Gaea build, and `editor.json`'s `gaea` block says so. Open points: Gaea's row order and origin for TIFF and R32 (north at the top? `probe.tif` shows it, and `orient=auto` covers it either way); its pixel registration (node or cell: half a pixel at most); whether its File node in Raw mode keeps values above 1 or below 0 from a float TIFF (the pack keeps v near 0..1 at amount 2, and floors below 0); whether "Height = amount x 200 m" is what Gaea's 1.0 means (it matters only to Gaea's own absolute-height nodes); which compression Gaea's own TIFF writer uses; whether it imports the masks' PNGs; its EXR and tile numbering (not read or written yet); time and memory on a real world. The method that settles most of them takes minutes: `bc_export 512 editoronly`, load `editor/probe.tif` and `editor/height.tif` in Gaea and note where north is and what the File node's minimum and maximum read, build the height at 512 px (Raw, TIFF 32-bit), save it as `editor/return/height.tif`, run `bc_import edited`, and send back the first lines of the report and the file Gaea wrote. Say all this plainly before the human spends an evening on a world.

Using an export with BetterWorlds

BetterWorlds is another mod; its documentation is its own. A Better Continents export folder is what it reads as it is: the export writes `coldmap.png`, `no_spawns_mask.png` (when the world has one), `world.json`, and the manifest's `settingsVersion`, `exportRevision`, `worldSize`, `edgeSize`, `layout` and `fileBytes`, so that it need not guess the size or the layout. The export's `README.txt` names its ways in, which are BetterWorlds' own commands: type `bw_import newest` in the console, or use *Import project* in its *Utilities & Layers*, or run `python3 tools/betterworlds.py --migrate <this folder> --output <folder>`. Quote only those; what they do is BetterWorlds' business. One limit to say: BetterWorlds takes at most 256 MiB for each map file and skips a larger one; the export adds a note (README and manifest) when a map is over that, while it can still be redone at a smaller size.

Cutting and pasting between an export and a generated map

Two maps line up pixel for pixel, so an area cut from one pastes straight into the other at the same place, when all of these agree:

- **Size** (pixels per side): `gen.py 2048 ...` pastes onto a `bc_export 2048 ...`, not a `4096` one.
- **Heightmap Amount and Sea Level**: `mapscale.py`'s `AMOUNT` (2 for ProximaMaxi) and Better Continents' default Sea Level of 0.5. `bc_export`'s own Amount default is 1, so pass `amount=2` (or set `Default Heightmap Amount` under `[09 BetterContinents.Export]`, appendix B); its Sea Level default is 0.5.
- **World Size and Edge Size**, i.e. the square's total width (`totalSize` in `manifest.json`). `gen.py` assumes the vanilla square, 21 km (10,000 m World Size + 500 m Edge Size, chapter 2). Exporting a **vanilla** world always uses those same two numbers. Exporting a **Better Continents** world uses whatever World Size/Edge Size *that* world was built with, which only matches the generator's square if it, too, used 10,000/500 (a new world's maps span its own sizes, chapter 2, so its export is `2 x (World Size + Edge Size)` wide; an older world's maps always span 21,000 m, so compare `totalSize`).

When all three agree: paste the same area from every map you keep (heights, biomes, forest, alt biomes, heat) so they stay consistent, and copy over any legend line (`.txt`) the other side's legend lacks. Heights blend over about a pixel at the seam; a biome pixel is read as whichever of the four nearest pixels is in the majority; a location dot must stay a single pixel touching no other dot of its colour. Only the *generated* world is exported — buildings, terrain edits and anything placed after world creation are not — so an export is a starting point, never a savegame.

> **Check:** before trusting a paste, open both folders' `manifest.json` and compare `size`, `heightmapAmount`, `seaLevel` and `totalSize`. All four equal is the whole rule; anything else and the pasted area lands at the wrong place, the wrong height, or both. A generated heightmap has no manifest: `python3 png_extras.py record heightmap.png` prints the Amount and Sea Level recorded in any heightmap, or says it has none.

Presets and `bc_import`

A folder of maps — an export, or `fire-and-ice/` itself — becomes a New World choice one of three ways:

- **A preset** (`bc_import`'s default): a self-contained `<name>.BetterContinents` file plus a picture, in `<save data>/BetterContinents/presets/`. It keeps working if the folder is later moved or deleted, and it freezes the maps as they were at import — edit the PNGs afterwards and re-import to update it. Every successful `bc_export` of up to 8,192 px already makes one, named `"<world> <time>"`; it appears in the New World dropdown but is **not** selected, so exporting a world never silently changes what the next New World is made from. A bigger export leaves it to `bc_import` (above).
- **The config** (`bc_import ... config`, or the HUD's "Use as config"): copies `export.cfg`'s settings into `BepInEx/config/BetterContinents.cfg` (the old file is kept beside it, timestamped), points `Directory` at the folder, and sets `SelectedPreset = From Config`. Every New World made with **From Config** afterwards reads the folder's PNGs as they are *then* — the same effect `stage-into-gale.sh` gets for `fire-and-ice/` by hand (appendix B), except this way works from inside a running game, because Better Continents now re-reads its config file whenever it changes, not only once at launch (no more quitting and relaunching to pick up an edited config before the next New World). Every `export.cfg` lists `Override version` empty, so this also clears an `Override version` of your own: set it again afterwards (chapter 10).
- `Directory` **alone** (Better Continents 0.9.4 and later): set only `[00 BetterContinents.Debug] Directory` to an export folder and create the New World with **From Config**. When the folder holds an `export.cfg`, Better Continents lays it over `BetterContinents.cfg` for that world exactly as an import builds its preset: `export.cfg` wins for every key it lists; `Directory` stays the config's own, so a moved or copied export loads from where it is now (the `Directory` line inside `export.cfg` names the folder it was written in, and is ignored); nothing is written to either file; keys that are not world settings (the ones appendix B marks live or local: `Debug Mode`, `Debug Reset Command`, `NexusID`, `Config Version`, `SelectedPreset`, `Settings Transfer Rate`, `Map Memory`, `Max Map Size`, `Wide Sectors` and the `[09 BetterContinents.Export]` group) and unknown keys are skipped as in an import. An export writes `High Terrain` into `export.cfg` only when the world's is On or Off, and never `Wide Sectors`. `BepInEx/LogOutput.log` says `Directory <folder> holds a world export, so the new world uses its export.cfg over BetterContinents.cfg: <N> setting(s) differ from your config.` (or `... every setting it lists already matches.`) and then one line per setting that differs, e.g. `[02 BetterContinents.Heightmap] Heightmap Amount = 2 (BetterContinents.cfg: 1)`. A folder with no `export.cfg` — `fire-and-ice/` included — is read exactly as before, every setting from `BetterContinents.cfg`. The overlay also applies after `bc_import ... config`, which points `Directory` at the folder too: to use a value of your own, edit `export.cfg` or delete it, never just `BetterContinents.cfg`. Presets carry their own settings and are unaffected; so is every existing world.

> **Trap:** a heightmap read at the wrong Heightmap Amount is the wrong height everywhere, and drowns the world when the Amount is too small. An export stores heights for the Amount it was made at (`export.cfg`, `manifest.json` and the heightmap itself say which): `height = (pixel/65535 × amount − 0.15 + Lerp(1, −1, sea level)) × 200` metres, the water surface being at 30 m

(chapter 2). At Amount 1 the waterline is pixel 0.30, black is −30 m and white 170 m of height (altitudes of −60 m and +140 m), and ground above that is clipped (the export says how many pixels); at Amount 2 the waterline is 0.15 and white is 370 m of height (altitude +340 m), which keeps every vanilla mountain. An export made by 0.9.0 to 0.9.4 is at Amount 2: read at 1, the waterline moves from 0.15 to 0.30, and the export reported on 2026-10-02 had 30.7 % land at Amount 2 and 2.5 % at Amount 1, a world about 99 % ocean. Before 0.9.4 that is what `Directory` alone did with an export folder; it also kept the config's Forestmap Multiply/Add (1/1, the export needs 0/1: wrong forest density), Skip Default Locations (false: a second set of locations next to a complete location map) and alt-biome `Mode` (Random: extra alt biomes next to the planted ones). In any version, a bare `heightmap.png` copied without its `export.cfg` needs the Heightmap Amount and Sea Level Adjustment it was encoded for set by hand. Since 0.10 the PNG carries them, and a new world that reads it at other values warns in the log (and `bc info` shows the record).

What an import prints, to the console and the log:

```
World import: Preset "<world> <date> <time>" is made and selected: create a New World to use it.
World import: <what it holds>.
World import: saved as <save folder>/BetterContinents/presets/<name>.BetterContinents.
```

With `config` instead, it copies `export.cfg` into `BetterContinents.cfg` and says `World import: create a New World now; it reads the PNGs in the folder as they are when you create it. No restart is needed.`; a folder with no `export.cfg` gets `World import: the folder has no export.cfg, so only Directory changed; every other setting is yours.` A setting it skips is listed as `World import: ignored <line>`.

A worked round trip (chapter 11, recipe 5, has it step by step): `bc_export 2048 amount=2` in the world; raise an island in `heightmap.png` in a 16-bit editor and delete `heightmap-fine.png` if there is one; `bc_import`; New World with a new name. Check it: the creation dump says `Heightmap amount 2`, and the island stands where you raised it.

Command	What it does
`bc_import`	the newest finished export of the loaded (or last-played) world; makes and selects a preset
`bc_import list`	numbers every export folder, newest first, with world, time, size
`bc_import <number>`	the export on that line of the list
`bc_import <world>`	that world's newest finished export
`bc_import <folder>`	any folder of maps: a full path, or one under `BetterContinents/`
`bc_import ... config`	the same target, applied to `BetterContinents.cfg` instead of a preset
`bc_import edited ...`	brings back the height an editor made from an export's `editor/return/`, as a new folder `<export>-edited` (see "The way back")
`bc_import status` / `help`	what the last import did / this table

`bc_import` is not a cheat: it works for any player, in the main menu or in a world (there is no New World screen to refresh from the main menu, but the console still builds and selects the preset, ready for when you open one). The host's `bc` tree has the same thing as `bc import run|list|status`.

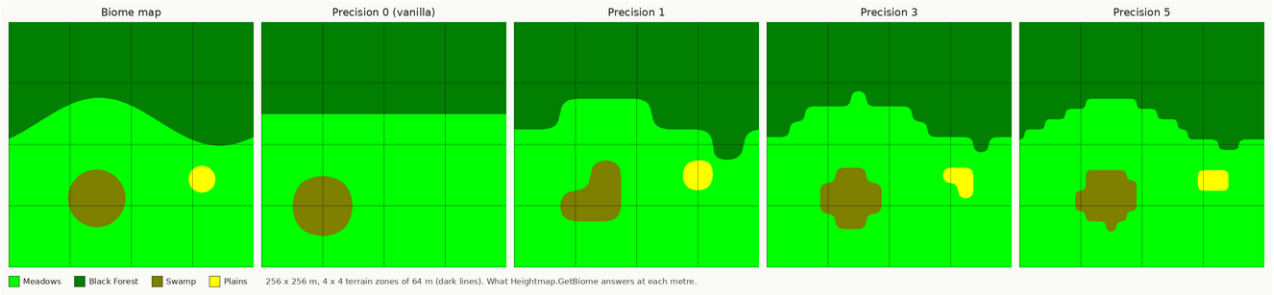
Which settings version the world gets from a folder (chapter 10). A folder with an `export.cfg` (a world export), whether it comes in through `bc_import`, an export's own preset or a Directory, gives a new world, version 12, when its maps span what its own World Size and Edge Size give (`manifest.json`'s `totalSize` equals 2 × (World Size + Edge Size) from `export.cfg`): an export of a version 12 world, or of a vanilla-size 0.9 world. Any other export, an export of a resized 0.9 world or one made under Expand World Size at another size, gives a version 11 world, and none of the new-world behaviour; its maps span 21,000 m, as a resized 0.9 world's did, or Expand World Size's size when that mod is installed, so an export made under Expand World Size is rebuilt as it was only with it installed at that size. The log's import note says so.

A folder of your own maps with no `export.cfg` (`fire-and-ice/`) is read from `BetterContinents.cfg` and gives version 12; set `Override version` to 11 before the world is made (a Directory or From Config world gets it as its version, and a world made from a preset is saved in it; only the preset file that `bc_import` builds ignores it) to get 0.9.4's reading of the same files.

> **Rule:** whichever way you load a folder, always **create a new world**. Better Continents only reads maps when a world is created (chapter 7's trap 5, "you edited a PNG and the world did not change"); a preset or `From Config` changes what the *next* New World is built from, never a world that already exists.

Biome precision

`[03 BetterContinents.Biomemap] Biome precision` (console `bc b p`) works again from Better Continents 0.9.0. At 0 (vanilla), everything inside a 64 m terrain zone — ground texture, grass, vegetation, spawn points — takes its biome from the zone's four corners, so a painted border only really shows up on that 64 m grid no matter how precisely you painted it. At 1 and up the zone is split into `(precision + 1) x (precision + 1)` cells, each reading the biome map on its own, with a corner every `64 / (precision + 1)` m: 32 m at 1, 11 m at 5. **A new world goes up to 31** (a corner every 2 m, the grid of a 10,501 px map); an older world stops at 5. A finer precision builds more cells in every zone, so raise it only as far as the painted detail needs, and try the cost on the machine that will run the world.



What `Heightmap.GetBiome` answers across four 64 m zones, drawn offline from the mod's own grid code (not a screenshot): at precision 0 a Swamp island or a Plains sliver either swallows a whole zone or disappears, because only the four corners are ever asked; by precision 5 the shape you painted survives.

> **Check:** precision changes what a point **looks like and grows**, never where a **location** or a **planted alt biome** is *allowed*. `IsBiomeEdge`, a location's biome-area test (chapter 3) and an alt-biome region's own corner/sector rules (chapter 3, appendix C) all keep vanilla's four-corner, 12 m answer regardless of Biome precision — `locations.py` and `altbiomes.py` model them correctly whatever precision you set. Precision never changes terrain height either.

> **Human:** nothing about Biome precision had run in a real game as of 0.9.0 — not the ground textures, not vegetation and spawns at precision 1-5, not `bc b p` while zones are loading. Say so plainly if you turn it on for someone, and more so above 5 (new worlds only), which is newer still. It is checked offline against the mod's own test suite, including a byte-identical comparison with vanilla's terrain builder at every precision level, but the human is the first to actually see it rendered.

Multiplayer: the settings download

A joining client receives the world's settings and maps (the same data an export holds, packed for the network) from the host or dedicated server before the join finishes, and caches them by id, so the transfer happens once per world revision. The client plays the settings version the server sends, whatever its own config says, and on a world that uses Better Continents it must run exactly the server's version of the mod (chapter 10). Valheim pins every Steam connection to about 150 KB/s; a 2048 px world is 11.8 MB (about 93 s), a 4096 px world with a paint map several times that. Two 0.9.0 additions:

- `[07 BetterContinents.Misc] Settings Transfer Rate` — `Vanilla` (leave Valheim's rate), `KB256`, `KB384`, `KB512` (the default), `KB768`, `MB1`, `MB1_5`, `MB3` or `Unlimited` (a fixed 100 MB/s, not a literal 0). Read on the sending side only, at the moment a client joins: a client's value does nothing, a PlayFab (crossplay) connection cannot be raised at all, and Better Continents re-reads the file on change, so no restart. Two facts that bit on 2026-09-25: GameNetworkingSockets has NO adaptive bandwidth, a connection sends at its SendRateMin, so both SendRateMin and SendRateMax are set on the one connection (raising only Max changed nothing); and a dedicated server's sockets belong to the game-server Steam interface (`SteamGameServerNetworkingUtils`), where the client wrapper throws "Steamworks is not initialized". The transfer loop itself (one 128 KiB chunk a server frame, at most two in flight) tops out near 4 MB/s (3.85 MB/s measured). The server's log shows `Settings transfer rate: ... (server setting)`, then what the download should take, `The <size> MB of settings take about <time> s to send at that rate.`, and at the end `Settings sent: <N> bytes in <s> s (<K> KB/s)`.
- `bc_cache` — `bc_cache export` on the machine that runs the world (or on a client that already holds the world) writes `<world>-<id>.bcworld` plus a `.txt` note into `<save data>/BetterContinents/<world>/`; `bc_cache import <file or folder>` (main menu or in a world) puts it in the local cache, never overwriting, with the id recomputed from the bytes; `bc_cache list` shows the cache. At startup every `.bcworld` under `BepInEx/plugins/` and in `BepInEx/config/BetterContinents/seed/` is imported, so a mod pack holding nothing but the file pre-seeds every installer. The id changes with any change to the world's Better Continents settings, so re-export after editing the world. The `.bcworld` of a version 12 world imports in 0.10 only: 0.9.x refuses it as `not a Better Continents settings package (bad header)`. A `.bcworld` file, like a world's settings package, can be up to 1.5 GB (the file's limit was 512 MB before 0.10.3).

A third way to shorten the join is to make the world smaller: **Compact Maps** (chapter 10; `Auto`, the default, makes every new world with a map over 8,192 px, or with fine heights, compact) saves and sends the maps compressed. On the 10,501 px map set measured there, the world file drops from 126 MB to 10.5 MB, and five 16,384 px maps take 52.6 MB against 352.5 MB as pictures; a joining player downloads that much less. A big world still takes minutes to download at the default rate (chapter 10 gives the times): hand out its `.bcworld`.

> **Human:** the rate presets ran for real on 2026-09-25, a client joining a local 1.0.15 dedicated server with an 11.8 MB world: 78 s at `Vanilla`, 23 s at `KB512`, 3 s at `Unlimited`. A pre-seeded join (`bc_cache`) has not run in a real game yet; it is covered by the offline suites (tools/livecfg-tests) only. The 93 s baseline is the ProximaMaxi server's, at Valheim's own rate.

Traps

#	Symptom	Cause	Fix
1	Two maps (an export and a generated one, or two exports) pasted together, but the piece lands in the wrong place or at the wrong height	Their Size, Heightmap Amount, Sea Level or World Size/Edge Size (`totalSize`) differ	Compare all four in `manifest.json` first; see "Cutting and pasting" above
2	A pasted or hand-authored area sits a fixed number of metres off from the rest of the map	Sea Level (or Heightmap Amount) differs between that piece and the whole	Keep every piece at the same Amount and Sea Level: the workspace's Amount 2 (`bc_export amount=2`, the same as `mapscale.py`) and Sea Level 0.5
3	An imported export is missing most bosses, dungeons and vendors	It was `bc_export`ed on a client, which only knows the locations the server has shown that player	Export on the host, or `bc_export server ...` on the dedicated server (admins)
4	You imported a folder, or edited its PNGs afterwards, and the world you are in did not change	Maps are only read when a world is **created**	Create a NEW world from the preset or `From Config`; an existing world keeps what it was made with
5	An exported alt-biome map's regions do not match what `altbiomes.py` predicted for the same PNGs	The export was below 2048 px, wider than the game's 12 m alt-biome grid	Export at 2048 px or more whenever alt biomes matter
6	Players still wait a minute or more to join after `Settings Transfer Rate` was raised	It was set on a client, or the connection is PlayFab (crossplay)	Set it in the server's (or host's) config; or hand out a `.bcworld` (`bc_cache export`) so the join needs no download
7	A world made `From Config` with `Directory` set to an export folder is almost all ocean	Before 0.9.4 that path ignored `export.cfg`: the heightmap was read at the config's Heightmap Amount 1, not the export's 2	Better Continents 0.9.4 or later; or the export's preset, `bc_import`, or `bc_import ... config`. A bare `heightmap.png` needs Amount 2 / Sea Level Adjustment 0.5 by hand
8	A setting changed in `BetterContinents.cfg` makes no difference to the new world	`Directory` points at a folder whose `export.cfg` lists that key; since 0.9.4 `export.cfg` wins	Edit `export.cfg`, or delete it; the log lists every setting where the two differ
9	A 16,384 px export made no New World preset	By design: above 8,192 px the export leaves the preset to `bc_import` (building it decodes every map), and its `README.txt` says so	Run `bc_import` on the folder, on a machine with the memory
10	The export window offers no 16,384 px, or `bc_export 16384` is refused	`Largest Size` in `[09 BetterContinents.Export]` is 4096 or 8192 on this machine; the message names it	Set `Largest Size = 16384`
11	Land edited in a terrain editor comes back upside down, mirrored, or far too tall or flat	The editor turned the picture, or built a Normalized or Custom Output Range	Build with Output Range = Raw; `bc_import edited orient=auto` (or `orient=flipy` and the like, `probe.tif` shows which), `range=lo,hi` or `unit=metres` for a range you know; see "The traps of the editor round trip"
12	`bc_export: unknown option 'editor'` (or `nocold`)	This Better Continents is older than the one that has the editor pack and the cold map	The same, newer version on every machine that exports
13	A world rebuilt from an export has mountain or Mistlands crags twice as rough	Its `export.cfg` (which pins `Mountain Noise` and `Mistlands Noise` to 0) was left behind, so the player's own config's 1 applies	Keep `export.cfg` with the folder, or set both to 0

#	Symptom	Cause	Fix
14	A boss or town sits under, or inside, a mountain raised in the editor; the new biome does not follow the new land	Biomes, locations and the other maps do not follow edited ground; the location map is copied unchanged	Repaint the pictures, or edit around `protect-locations.png`
15	BetterWorlds skips a map of the export	The file is over 256 MiB, which BetterWorlds takes at most (the export's notes say so)	Export at a smaller size

10 What a new world does differently

Better Continents 0.10 changed what a world can be like, and one number decides which worlds get the changes: the **settings version** that every world stores in its `BetterContinents` file. 0.10 reads it, and each change in this chapter that alters what a world generates switches on only for **version 12**, which is what 0.10 gives a world it creates. A world made by 0.9.x is **version 11**: it keeps generating its terrain, biomes and locations as it did under 0.9.4, and it is saved as 0.9.4 saved it. There is no separate "0.9.4 mode": one code base runs every world, and the version picks the old or the new rule where the rule is used.

This document says **a new world** for a version 12 world and **an older world** for a version 11 one, and chapter 2 already uses both where the model needs them. Not every world made in 0.10 is a new world: a world made from a preset that ships with the mod or one that 0.9 saved, from an export of a world resized under 0.9 or made under Expand World Size at another size, or with `Override version` 11 is version 11 (next section). And what keeps its 0.9.4 behaviour is every existing world and preset, not every map file: a folder of PNGs of your own, used to make a world in 0.10, is read the 0.10 way (exact pins, the start before the pins, the alpha blend, finer biome precision, Expand World Data's rules), so the same folder makes a slightly different world than it made in 0.9.4. This chapter says what the differences are and what each means to someone making maps.

Better Continents 0.10.3 added more, decided by a world's size, its Amount and its maps: **worlds past 16 km** with wide sectors (below), maps up to 16,384 px (chapter 2), tall worlds (chapters 2 and 3) and fine heights (chapter 2, for a version 12 world only), and it made Compact Maps automatic for big maps (below).

> **Rule: A world past 16,350 m binds its version.** A new world whose World Size + Edge Size is past 16,350 m is made with wide sectors by default (`Wide Sectors = Auto`), and a wide world needs Better Continents 0.10.3 or later from then on: an older Better Continents opens it as a world without Better Continents, and an older version that saves it writes its objects twice. Read "Worlds past 16 km" below before you choose such a size, and tell the human.

Which world do you have

`bc info` prints the world's settings, and the first line is its version: `Version 12` for a new world, `Version 11` for an older one (`bc` is a cheat command, which Debug Mode switches on when you spawn, and it runs in single player or on the host). The BepInEx log prints the same line, the first of the settings dump, when a world loads (single player or the host), when a new world's settings are made, on the server each time it sends the settings to a joining player who has no cached copy, and on a joining player's machine when the server's settings arrive (downloaded or from its cache). A save prints nothing there: the game saves on a background thread whose log lines never reach the file (the world's `BetterContinents` file is still written at every save). A new world at the default size prints no `The maps span ...` line, so the `Version` line is the one to read; the Esc-menu hint shows only the mod's version and ENABLED or DISABLED. A world with no `BetterContinents` file in its folder is vanilla and has no version. A world stored by a much older Better Continents shows a number below 11 until it is loaded again after its first save (a save writes 11 but never changes the number the loaded world holds), as under 0.9.x.

The first lines of `bc info` in a new world made From Config at World Size 24,000 (the rest of the dump follows):

```
Version 12
...
World size 24000
The maps span World Size and Edge Size: 49000 m across, and the world is laid out to that size
```

How a world gets its version:

- A world made **From Config** (no export folder) is version 12, unless `Override version` says otherwise (below).
- A **preset** gives the world the preset's own version, saved as at least 11 (while `Override version` is set, every world is saved in that version instead: see below). A preset saved by 0.9.x makes a version 11 world. A preset saved by 0.10 from a version 12 world (`bc savepreset`), or made by `bc_import` or an export, makes a version 12 one (11 by the export rule below). **The four presets that ship with the mod (Continents, Pangaea, Swirls, Vanillaish) are old version 7 files, so a world made from one is version 11**: its creation log says `Version 7`, and its file is saved as 11. They hold no map files and vanilla's size, where the two versions generate alike.
- A world made from a **folder of maps** (`[00 BetterContinents.Debug] Directory`, `bc_import`, an export's own preset):
  - A folder with an `export.cfg` (a world export) gives version 12 when its maps span what its own World Size and Edge Size give (`manifest.json`'s `totalSize` equals 2 x (World Size + Edge Size) of `export.cfg`): an export of a version 12 world, or of a vanilla-size 0.9 world. Otherwise it gives version 11 (an export of a resized 0.9 world, or one made under Expand World Size at another size). The import's log note says so, and such a world gets none of the new-world behaviour. Its maps span 21,000 m, as a resized 0.9 world's did, or Expand World Size's size when that mod is installed, so an export made under Expand World Size is rebuilt as it was only with Expand World Size installed at that size.
  - A folder of maps you made yourself, with no `export.cfg` (`fire-and-ice/`), makes a new world, version 12. The same files make a slightly different world than 0.9.4 made from them (exact pins, the start before the pins, the alpha blend, finer precision, Expand World Data's rules); set `Override version` to 11 before you create the world to get the 0.9.4 result.
- `Override version` (`[00 BetterContinents.Debug]`, empty by default, for testing): a whole number of 11 or more is the version a From Config world gets, and the world is saved in it, so it plays the same before and after its first save. The same number is also the version **every** world is saved in while it is set, and the one a host sends to joining players: 11 re-saves a version 12 world as 11, which loses what 12 added from its next load, and 12 re-stamps a version 11 world, so new zones follow the 0.10 rules and seams are likely where they meet the old ones. A number below 11 is not ignored: it writes the old fixed layout and drops data. `bc_import` and an export's own preset ignore it when they make the preset file (a world made from any preset is saved in it, like every world), and `bc_import ... config` clears it, because every `export.cfg` lists `Override version` empty. Leave it empty, on a server above all.
- There is **no upgrade from 11 to 12**. Nothing in 0.10 converts an existing world; only `Override version` re-stamps one at a save, which is not recommended.

> **Rule:** Read the world's version before you trust a rule in this chapter: the first line of `bc info`, `Version 12` or `Version 11`. A world you make From Config with `Override version` empty is 12, and so is one made from a folder of your own maps. A world made from a preset that ships with the mod or one that 0.9 saved, from an export of a world resized under 0.9 or made under Expand World Size at another size, or with `Override version` 11 is 11, and nothing converts it to 12 (only `Override version` re-stamps, for testing).

At a glance

What	An older world (version 11)	A new world (version 12)
The maps' span	21,000 m, whatever World Size says	2 x (World Size + Edge Size), the land dropping away over Edge Size; 21,000 m at the default 10,000 / 500
The rest of the world: the alt-biome grid, the location search, the game's biome bands, the Ashlands and Deep North rings, lakes and streams, the minimap	Vanilla's; World Size only moves the game's edge	Scaled to the world when World Size or Edge Size is not 10,000 / 500; the alt-biome `Grid` setting has no effect
Heightmap Alpha (off by default)	On: the heightmap is read as 8-bit grey and the alpha is never used. Off: 16-bit grey	On: 16-bit grey with a 16-bit alpha, blended with the game's own terrain by it. Off: 16-bit grey
Biome precision	Up to 5 (cells of about 11 m); a higher value is stored but used as 5	Up to 31 (cells of 2 m in a 64 m zone). The config and `bc` accept 0 to 31 for every world
A location map's pins (fixed when the world is made)	One pixel of the pin, picked at random, at its corner: up to a pixel south-west of where the other maps put it; shared colours dealt out at random	On the pixel nearest the pin's middle, where the other maps put it; shared colours dealt out the same way every time. A world export's pins are still read at the corner
The start position override and a pin in the same 64 m zone	The pin wins and the start is dropped (0.10 logs an error for it)	The start is placed first and the pin is dropped
Expand World Data's altitude rules	The maps' heights replace them (except where the rough map is white)	They apply over the maps' heights, as over the game's own
Expand World Data's `lava: true` biomes (a biome map and no heat map)	Only the Ashlands are hot	They are hot too: lava burns and the Ashlands weather applies
A terrain legend's ground names	An Expand World Data biome's name reads as a bad colour (a warning, Meadows ground)	A legend may name its biomes' grounds
Biome precision and territory colours	The precision grid's colour replaces a territory's	Precision keeps a territory's ground colour
Expand World Data's biomes on a biome map	Stored by number, so renumbering its yaml swaps biomes	Stored with their names and followed if the numbers move; a biome it no longer has reads as None
Compact Maps	Not available: a version 11 world is never compact	As `Compact Maps` says when the world is made: Auto (the default) when a map is over 8,192 px across or the heightmap has fine heights, On always
Fine heights (`heightmap-fine.png`)	Never used	Used when the fine picture fits its heightmap and Compact Maps is not Off; the world is then compact and needs 0.10.3 or later (chapter 2)
The world file	Saved as version 11, as 0.9.4 saves it	Saved as 12; 0.9.x reads it as 11 and cannot read a compact one, and 0.10.0 to 0.10.2 cannot read one made with fine heights
A `.bcworld` file	Accepted by 0.9.x and 0.10	Accepted by 0.10 only: 0.9.x refuses it as a bad header

Several things apply to **every** world, old or new, and are not in the table: Heightmap Amount's default of 1 (chapter 2), the Amount an export records in its heightmap (chapter 9), maps held in tiles, the vegetation twin guard, water that follows rebuilt ground, the game's edge following each world's own World Size, the console tools and the config (all below), and the zone regeneration after `bc` changes (chapter 6: in single player or on the host, in a world that uses Better Continents, with the world's own settings, so an older world edited with `bc` keeps its 0.9 rules). None of them changes what a world's settings generate, except the twin guard, which changes the vegetation of zones that are not generated yet. Three things that 0.10.3 added follow a world's size, its Amount or its maps: wide sectors past 16,350 m (below), the high-world rules above an Amount of 5 (chapter 3), and maps up to 16,384 px, read a few rows at a time (chapter 2).

Mixing versions

- **Multiplayer.** On a world that uses Better Continents, the server and every player must run exactly the same version of the mod, whatever the world's settings version: a 0.9.x player cannot join a 0.10 server, a 0.10 player cannot join a 0.9.x server, and two different 0.10 releases refuse each other too. A world without Better Continents has no check. Players play the settings version the server sends, whatever their own config says. Update the server and every player together.
- **A new world opened in 0.9.x** is not supported: copy the world folder first. 0.9.x reads a version 12 world as version 11 (maps spanning 21,000 m whatever World Size says, no layout, the heightmap's alpha read as 8 bits, and so on), so zones generated from then on follow the 0.9.x rules: a world of non-default size can show seams, and the pins already in the world stay. Its next save writes the world as version 11, and from then on 0.10 treats it as made by 0.9.x. A world made with Compact Maps cannot be read at all: 0.9.x switches Better Continents off for it, generates vanilla terrain that stays in the world, and leaves the file alone. A preset saved by 0.10 loads in 0.9.x as a version 11 world (a Compact Maps one makes a world without Better Continents).
- **A 0.9.x world in 0.10 and back.** With `Override version` empty, 0.10 writes a version 11 world's file exactly as 0.9.4 does, so the world opens in 0.9.x as before. `BetterContinents.old` beside the file holds only the save before the last: back up `worlds_local/<World>` before changing versions.
- **A 0.10.3 world in an older version.** Some worlds cannot go back. Back up the world folder before changing versions, and never open such a world with an older Better Continents, in single player or on a server:
  - a wide world (below): an older Better Continents stops at the world's marker (`Unknown feature: 68` in its log) and opens it as a world without Better Continents, with vanilla terrain, and a save by an older version, or by the game alone, writes its objects twice (two of every chest);
  - a world made with fine heights: 0.10.0 to 0.10.2 cannot read it and give it vanilla terrain;
  - a world read at an Amount above 5: an older version opens it without the high-world rules (chapter 3); one whose `High Terrain` is On or Off it opens as a world without Better Continents (`Unknown feature`);
  - a compact world: 0.9 cannot read it and gives it vanilla terrain; every 0.10 version reads it (but for fine heights, above).

World Size: the whole world

On a new world, World Size and Edge Size lay out the whole world, not only its edge. The square the maps cover is `2 x (World Size + Edge Size)` on a side, the land dropping away over Edge Size, and what the game places by its distance from the centre follows the world's size by two measures: the alt-biome grid and the minimap follow World Size + Edge Size, and the locations, the game's biome bands, the rings, lakes and streams scale by World Size / 10,000:

- **The alt-biome grid** covers the world: 12 m points out to its edge, with vanilla's margin (2048 points a side for the default radius, 3190 at 16,350 m, 6392 at 32,764 m; from a radius of 42,000 m it stays at 8192, its points then wider apart). The game places its locations at the grid's points, so they reach the whole world. A bigger grid is built at every world load, on every machine (the game keeps no cache of it), and costs time and memory: about 0.2 GB at a 16,350 m radius, and about 0.8 GB and about 5 s at 32,764 m (measured on one machine with 32 cores; Better Continents builds it on every core, where the game's own way takes 29 s on one).
- **The locations** are searched for within World Size, and the distance window each one keeps in `m_minDistance` and `m_maxDistance` is multiplied by World Size / 10,000 (not under Expand World Data, whose location data is already in fractions of the radius): at World Size 15,000, Hildir's cavern (1,000 to 8,000 m in vanilla) is placed 1,500 to 12,000 m out. The second window that four locations set, `m_minDistanceFromCenter` and `m_maxDistanceFromCenter`, is **not** scaled, because the game checks it as it is: the frozen ships want Deep North water 8,000 to 9,750 m from the centre in a world of any size, and the combat ruin, the big rock clearing and the ancient upgrade station keep their minimums of 1,500, 1,000 and 500 m (chapter 3).
- **The game's own biome bands** (where no biome map decides), the Swamp's limit, the **Ashlands and Deep North rings** with their gaps, and the area the game looks in for **lakes and stream sources**, all at World Size / 10,000 of their vanilla distances.
- **The red of the Ashlands sea** (since 0.10.3) follows the world's hot sea: the ring laid out to the world's size, or the heat map where there is one (chapter 3).
- **The Deep North's calm sea** (since 0.10.5, with Better Continents' own water) follows the world's Deep North: the ring laid out to the world's size, or the biome map where there is one, drawn and felt alike (chapter 13).
- **The minimap's pixels** grow with the world, its texture keeping its size.
- The **terrain's detail** (each biome's hills, rocks, noise) keeps its size.
- **Not the terrain shader's Deep North look** (chapter 3). In 0.10.3 and 0.10.4 the calm sea stayed at the game's fixed circle too, drawn and felt; 0.10.0 to 0.10.2 scaled only the felt waves to the world's size.

A world at the default 10,000 m and 500 m is laid out as vanilla is, so nothing here changes for it; the rules about size matter once you move World Size or Edge Size. On **every** world, old or new, the game's edge (the kill zone, the ship push-back and the water edge) follows that world's own World Size, applied again for each world loaded in a session; 0.9.4 could keep the first world's.

Past **16,350 m** of World Size + Edge Size a world is big, and "Worlds past 16 km", next, applies to it: read it before you choose such a size.

What this means when you author:

- Pick the sizes in the config **before** you generate: the maps must cover `2 x (World Size + Edge Size)`, and World Size + Edge Size can go up to 65,000 m. A map can be up to 16,384 px: at 2 m a pixel that covers a square of 32.7 km, at 4 m one of 65.5 km. Design the layout in fractions of the radius (chapter 3) and it holds at any size, but for the frozen ships: keep Deep North water in their fixed ring, 8,000 to 9,750 m from the centre.

- Location **quantities** are the game's: a world with twice the radius has the same number of each location over four times the area, so the density of dungeons, altars and ruins falls with the square of the size.
- The game's **Deep North circle**, centred at (0, -4,000) with a radius of about 12,000 m, does not grow with the world. The terrain shader's Deep North look keeps to it, so a big world gets that look far from the north. The calm sea (since 0.10.5) and the red of the Ashlands sea follow the world: draw Deep North over the sea you want calm and heat over the sea you want red (chapters 2, 3 and 13).
- The workspace's generator and checkers (`gen.py`, `check.py`, `altbiomes.py`, `locations.py`) assume the default sizes: the 21,000 m square, a 2048-point grid and 10,000 m. For another size, change those constants in a copy (appendix A lists them) or compare only default-size worlds.
- The alt-biome `Grid` setting (WorldEdge or Vanilla) has no effect on a world laid out to its own size, whose grid is always the world's. It matters only on an older world with a World Size below vanilla's, where WorldEdge stops the grid at the world's edge and Vanilla samples vanilla's 10,500 m.
- Start Position X and Y accept -1,000,000 to 1,000,000 m on every world, so a start can be placed in a bigger world (the config stopped at ±10,500 m, and the console's x and y at 0 to 1).
- **With Expand World Size installed**, its size wins over the world's own on every world, for the maps and for the layout: it lays out the world as before. It must then be installed on every machine or none, and must not be added to a world Better Continents laid out. Its *World Stretch* magnifies Better Continents' maps, and the log warns once when a Better Continents world runs with a World Stretch other than 1. The larger of its size and World Size + Edge Size decides whether a world needs wide sectors (next section).

> **Check:** after creating a world at another size, `bc info` has the line `The maps span World Size and Edge Size: N m across, and the world is laid out to that size` with the `N` you expect (under Expand World Size the part after the comma is left off), and the log has one `The minimap's ... pixels are ... m` line. A world with the line missing is an older world (trap 31).

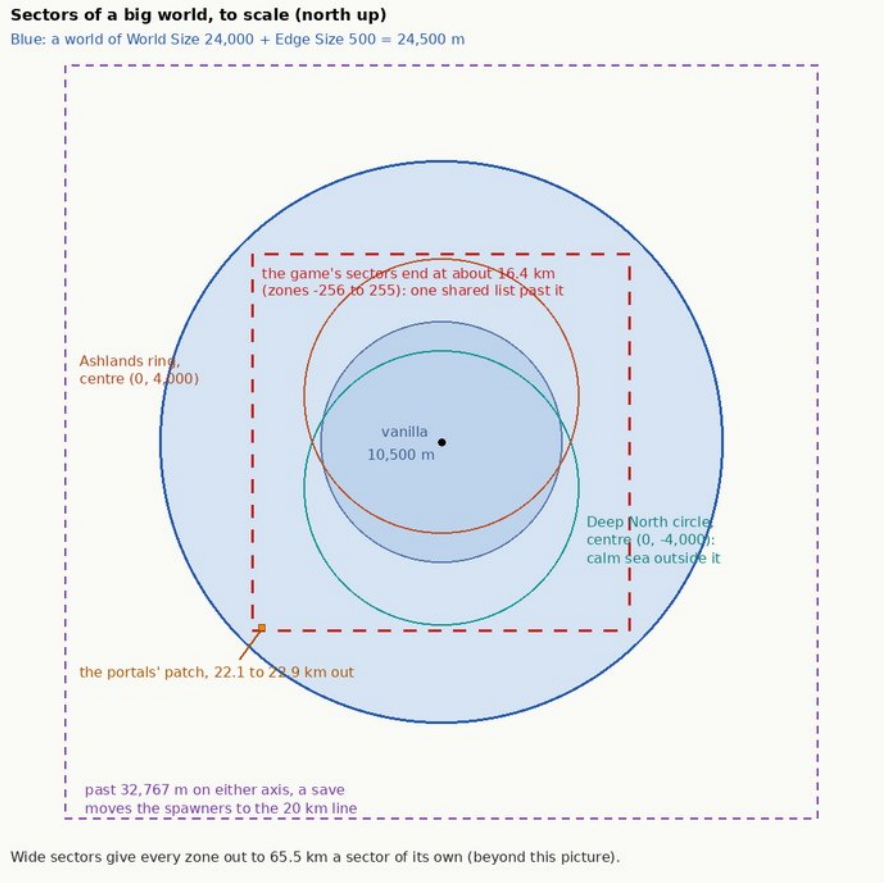
Worlds past 16 km: Wide Sectors

Read this before you choose a World Size + Edge Size past 16,350 m: part of it cannot be undone.

Valheim 1.0 files its objects (trees, rocks, creatures, buildings, dropped items) by 64 m zone, in a 512 x 512 array of sectors around the centre, and the array stops at zones -256 to 255: 16,352 m to the east and north, 16,416 m to the west and south. Past that the game's own code goes wrong in three ways, each from its own distance:

World Size + Edge Size	What goes wrong without wide sectors
up to 16,350 m	Nothing: wide sectors never apply, and the save is byte for byte what it was
past 16,350 m	**One shared list.** Every object more than about 16.4 km out shares one list, the sector of zone (-256, -256). A player near any of it is sent the whole list, which grows with every area generated or built out there and is saved as one chunk. On a test world of 24.5 km a player at zone 260 was sent 5,742 objects, 965 of them its own
past about 22,100 m	**A patch shares the portals' file.** A patch of 8 x 8 zones about 22 km south-west of the centre (22.1 to 22.9 km out) is saved under the key of the file that holds every portal in the world, and the later write wins: either the patch's objects are lost, or at the next load the portals are gone and the patch's objects are read as portals (read from the game's code; the test world had no portals to lose)
past 32,767 m	**Spawners moved to the 20 km line.** Each zone has one hidden spawner (`_ZoneCtrl`) at its centre, at y = 0, placed once, when the zone is first generated, and spawning only while a player is within 32 m of it on both axes. The game saves it exactly while both its coordinates are within ±32,767 m; for a zone further out on either axis, a save moves that coordinate to ±20,000 m, so after a restart those zones spawn no creatures of their own. A zone at 25,600 m by 25,600 m is saved where it is

Which of the three a world meets follows from its World Size + Edge Size; a world past 32,767 m meets all three.



The three distances to scale, drawn for this page from the numbers above, with a world of World Size 24,000 + Edge Size 500 in blue: the game's sectors end at about 16.4 km, the portals' patch lies near their south-west corner, and the spawner line is at 32,767 m on either axis. The game's fixed Deep North and Ashlands circles are drawn too.

Wide sectors give every zone from -1024 to 1023, out to 65.5 km, a sector and save chunks of its own, on the server and on every client: the shared list and the 20 km clamp are gone, and the south-west patch is kept apart from the portals. They cost about 32 MB of memory on each machine while a wide world is loaded, and planning a save takes 21 to 31 ms instead of 3 to 4 ms (measured on one machine). Up to 16,350 m nothing changes.

`[07 BetterContinents.Misc] Wide Sectors` is read by the machine that runs the world (single player, the host or a dedicated server), when a world is made and each time it loads; joining players follow the world. It is that machine's own setting: no world, export or import carries it, and an `export.cfg` line for it is ignored. A wide world says so in its own settings, which is what an older Better Continents stops at.

`Wide Sectors`	**A new world past 16,350 m**	**An existing world past 16,350 m**
Auto (the default)	is made wide	keeps the sectors it has
On	is made wide	is converted to wide when it loads, one way
Off	gets the game's sectors	keeps the sectors it has; a wide world stays wide

- **With the default, a new big world is wide from the start**, and there is nothing to do.
- **A wide world needs Better Continents 0.10.3 or later from then on.** An older Better Continents stops at the world's marker (`Unknown feature: 68` in its log) and opens it as a world without Better Continents, with vanilla terrain. An older version (0.10.2 or older, or the game without Better

Continents) that saves it writes its objects twice, and they come back as two of each, two of every chest: one 0.10.2 save doubled 121 objects, and every far object is doubled once anything in the shared list changes. No setting can stop an older version from doing this. A join with another version is refused, so the risk is the machine that runs the world: never open a wide world there with an older version.

- **An existing big world** (made before 0.10.3, or with `Off`) keeps the problems its size brings until it is converted: back up its folder (`worlds_local/<World>`), set `Wide Sectors = On` on the machine that runs it, and load it. Before the converted world's first save the game saves a copy of the old save, `<world>_backup_<date and time>` beside it, and a warning in the log names it; the conversion puts the shared list and the misfiled patch back where they belong (measured: 5,735 objects out of the shared list, 121 out of the portals' file, no duplicates; the log counts them, `Sectors: N objects lie beyond the game's sectors ...`). It cannot undo damage already done: objects already doubled stay doubled, spawners already moved to the 20 km line stay there, and what the portals' file already overwrote stays lost. It is one way, and the human's choice (chapter 1).
- World Size + Edge Size can go up to **65,000 m**; beyond 65.5 km objects share one sector again. The practical limit is start time and memory (next section): set World Size to how far players will really go. With `Map Edge Drop-off` off the world has no edge, but the sectors still follow World Size + Edge Size.

In the log, a new world made wide has the line `Sectors: this new world reaches N m, so it is made with wide sectors (Wide Sectors is Auto): every zone out to 65504 m has a sector and save chunks of its own.`, and a world that runs on wide sectors `Sectors: every zone from -1024 to 1023 (to 65504 m) has a sector and save chunks of its own ...`. With `Off`, a new big world logs `Sectors: Wide Sectors is Off, so this new world, which reaches N m, has the game's own sectors ...`, and a wide one `Sectors: Wide Sectors is Off, but this world stays wide: its settings say it is a wide world.`

A worked example. The human wants a world about two and a half times as wide as vanilla.

	Bad	Good
The size	World Size 24,000 with the 2,048 px maps drawn for 21 km: the same picture stretched, 23.9 m a pixel	maps drawn for 49,000 m, 4,096 px or more (11.97 m a pixel at 4,096)
The sectors	`Wide Sectors = Off`, or a world made by 0.10.2 at this size: one shared list past 16.4 km, the portals' patch past 22.1 km	`Wide Sectors = Auto`: the new world is made wide
The human	told nothing	told first that the world needs 0.10.3 or later from then on, on the machine that runs it and every player's

The config lines:

```
[01 BetterContinents.Global]
World Size = 24000
Edge Size = 500

[07 BetterContinents.Misc]
Wide Sectors = Auto
```

What the log then says at creation, and `bc info` in the world:

```
World size 24000
The maps span World Size and Edge Size: 49000 m across, and the world is laid out to that size
Sectors: this new world reaches 24500 m, so it is made with wide sectors (Wide Sectors is
» Auto): every zone out to 65504 m has a sector and save chunks of its own.
Wide sectors: every zone out to 65504 m has a sector and save chunks of its own, where the game
» gives everything past 16.4 km one shared sector
```

and at every load `Sectors: every zone from -1024 to 1023 (to 65504 m) has a sector and save chunks of its own (2048 x 2048 sectors, 32 MB), where the game has 512 x 512.` Chapter 11, recipe 3, is the whole setup with maps and legends.

> **Check it:** `bc info` in the world shows the `Wide sectors:` line for a wide world and none for one on the game's sectors. A world past 16,350 m without it is the trap-32 world: back it up before deciding anything.

A big world: start, memory and joining

Measured on one machine, a Linux dedicated server; the join times follow from the transfer rate.

- **Starting.** On a 32-core server, a vanilla-size world starts in about 11 s, a 24,500 m world on the game's own terrain in about 18 s, and a world of World Size 32,264 + Edge Size 500 with five 16,384 px maps in about 16 s. Fewer cores take longer: the alt-biome grid is built on every core. Since 0.10.3 the game's lake merge is fast. It runs on every machine at every load, and at World Size 32,264 it took 137 s with no maps; with that world's maps it was still running when the server was stopped after 416 s. It now takes 0.3 to 2 s and finds the same lakes. That is for a world Better Continents lays out, not one laid out by Expand World Size.
- **The alt-biome grid** is built on every machine that loads the world: 6392 points a side at 32,764 m, about 0.8 GB and about 5 s on 32 cores (chapter 3).
- **Making the world** from 16,384 px maps takes 17 to 39 s behind the game's "Please wait" (before 0.10.3 the game window froze for that long), and 0.5 to 1.5 GB above the idle server at its highest (chapter 2). With five 16,383 px maps the whole process peaked at 1.80 GB, where Better Continents 0.10.2 needed 5.83 GB. The log brackets the wait: `New world <name>: making its settings from 'From Config' while the game goes on`, then `New world <name>: its settings are ready, in <N> s`.
- **Maps.** A 16-bit 16,384 px map is 537 MB decoded. Maps more than 8,192 px across make the world compact (with `Compact Maps = Auto`, the default), and `Map Memory` is how much decoded map data a machine keeps (below).

Joining. A player who does not have the world yet is sent the world file at `Settings Transfer Rate` (chapter 9; `KB512`, 512 KB/s, by default):

World file	At `KB512` (the default)	At `MB3`
52.6 MB	about 1.7 minutes	about 17 s
139 MB	about 4.4 minutes	about 44 s
203 MB	about 6.5 minutes	about 65 s

Each join that needs a download also costs the server one long frame: about 0.3 s for 52.6 MB, 1.3 s for 203 MB. Raise the rate if the server's upload carries it; better, hand out the world's `.bcworld` (`bc_cache export`, chapter 9), so that players join without a download.

Heightmap Alpha: blend a map into the game's own terrain

`[02 BetterContinents.Heightmap] Heightmap Alpha = true` (off by default) lets a heightmap carry an alpha channel that says, pixel by pixel, how much of the map to use. On a new world the heightmap is read as **16-bit grey with a 16-bit alpha** (a PNG of colour type 4), the heights exactly as a plain grey heightmap gives them:

- alpha 1 (opaque): the world's height is the heightmap's, as always;
- alpha 0 (transparent): the height is the game's own, from its formula for the world's seed;

- in between: the two mixed, `game x (1 - alpha) + map x alpha`.

With Heightmap Override All (the default), the biome heights use this blended base height. The use is a map that covers only part of the world: an island or a continent drawn on the game's own sea, with its edges faded into vanilla terrain, instead of a whole world that must be authored. The biome map still decides biomes wherever it has a colour. An older world reads the setting as it always did: with Heightmap Alpha on, as 8-bit grey with the alpha unused; with it off, as 16-bit grey, which every world does. So the setting only gains something on a new world, where a fine picture refines a grey and alpha heightmap too (chapter 2).

`png_extras.py gray-alpha out.png height.png alpha.png` (appendix A) joins two 16-bit grey PNGs of one size into such a file, white alpha being the map. The settings dump of a new world made with it says `Heightmap alpha: full precision, blended with the game's own terrain by the alpha`; an older world's says `Heightmap alpha: 8-bit heights, the alpha unused (a world made by an older Better Continents)`. In **every** world, `bc h alpha true` or `false` (a `bc` command: it needs cheats and runs on the host) decodes the picture the world holds again at once, by that world's own reading: a new world blends, an older world re-reads it as 8-bit grey with the alpha unused (with the setting off, both read 16-bit grey). Before, a change did nothing until the world was loaded again.

> **Human:** the blend is checked offline against the mod's own test suite, with the game's formula stood in for by a value; as of this edition nobody has watched a Heightmap Alpha world in a real game. Say so if you offer it.

Expand World Data on a new world

Expand World Data (EWD) is a separate mod that changes the game's biomes, locations and heights through yaml files. Better Continents has always supported it; a new world follows it more closely in five places, each of which shows only where EWD is installed:

1. **Its altitude rules apply over the map.** A biome's or a territory's altitude multiplier and delta, water depth, height limits and the lava dip are applied to the heights the maps give, as they are to the game's own. On an older world the heightmap's ground replaced them wherever a roughmap was not 1. The checkers do not model EWD, so a world with such rules has heights that differ from the heightmap's by those rules.
2. **Its lava biomes are hot** (`lava: true`) where the biome map puts them, so lava burns and the weather there is the Ashlands', on a world with a biome map and no heat map (or Heatmap Scale 0); with a heat map the heat decides. An older world heats only the Ashlands.
3. **The terrain map's legend** may name an EWD biome for a ground colour, which the world keeps as that biome's ground colour. An older world reads such a name as a colour that does not parse.
4. **Biome precision keeps EWD's territory ground colours** instead of replacing them with the precision grid's.
5. **The biome map's added biomes are saved with their names.** EWD numbers its biomes in its yaml's order, and an older world stores the biome map by those numbers, so reordering the yaml swaps biomes. A new world follows the names; a biome EWD no longer has reads as None.

A biome legend may name EWD's added biomes by their names (chapter 2). None of this needs anything from the map folder.

Memory: tiles, and Compact Maps

Every world holds its maps in memory as **tiles** of 128 x 128 pixels. A tile of one value, such as open sea or the canvas around a painted area, is held as that single value. Every sampled value is exactly what it was. On a real map set (a 10,501 px heightmap, forest map and biome map: 2 m per pixel over the 21 km square), a world holds about 99 MB instead of about 1.32 GB. A large map is far less of a memory problem than it was.

Compact Maps (`[07 BetterContinents.Misc] Compact Maps`) goes further. A compact world keeps its maps compressed in memory, decodes each tile when it first reads it, and saves and sends them compressed. On that same map set the world file is 10.5 MB instead of 126 MB, and a loaded world holds about 21 MB. With 16,384 px maps it is what keeps a world practical: as a picture, each 16,384 px biome, spawn or vegetation map is 268 MB of raw bytes, in the world file and in every join, and five 16,384 px maps took 52.6 MB in a compact world file against 352.5 MB as pictures.

`Compact Maps`	**A new world is made compact**
Auto (the default)	when any of its maps but the location and alt-biome maps is more than 8,192 px across, or its heightmap has fine heights; the log says which
On	always
Off	never, not even with 16,384 px maps: its maps are saved as pictures (a map over 8,192 px with a warning in the log), and no `heightmap-fine.png` is read

What the log says when a new world is made compact, or (the third line, a warning) is kept as pictures although it is big:

```
Compact Maps is Auto and Heightmap <path> is 16384 x 16384 pixels (more than 8192 across), so
» this world is compact. A map that large is kept, saved and sent to joining players as
» compressed tiles: as pictures it would take gigabytes of memory and a world file of hundreds
» of megabytes.
Compact Maps is on for this world: its heightmap has fine heights (heightmap-fine.png), which a
» world keeps as compressed tiles, and so it does its maps.
Compact Maps is Off, so this world keeps its maps as pictures although Heightmap <path> is
» 16384 x 16384 pixels (more than 8192 across). As pictures such maps take gigabytes of memory,
» and the world file and what every joining player is sent come to hundreds of megabytes.
```

and `bc info` in a compact world: `Compact Maps: the maps are held and saved as compressed tiles, each decoded when the world reads it`.

A config written by 0.10.0 to 0.10.2 holds `true` or `false` there: `true` reads as On, `false` as Auto. Rules:

- It is read only when a world is made (From Config, `bc_import`, a Directory's `export.cfg`), and only for a new world: a version 11 world is never compact, and `Override version` 11 switches it off. A world keeps the way it was made: a compact world stays compact, and a later change of the setting changes no existing world. A preset made while it was on carries the tiles, so every world made from that preset is compact.
- A world made with fine heights is always compact (chapter 2).
- **Better Continents 0.9 cannot read a compact world**: it stops at the tiles (`Unknown feature` in its log), switches Better Continents off for the world, generates vanilla terrain, which stays in the world, and leaves the settings file alone. Versions 0.10.0 to 0.10.2 read it, unless it was made with fine heights (chapter 2). For a world that 0.9 users will open, set `Compact Maps = Off`, or keep Auto with every map at 8,192 px or less and no fine heights.
- `Map Memory` (default 512 MB, 64 to 65,536) is how much decoded map data a compact world may keep on this machine; past it, the tiles read least recently are dropped and decoded again when they are read. 512 holds every tile of a world's maps up to about 8,192 px across, so that each is decoded once; one 16-bit 16,384 px map is 537 MB decoded, so for a 16k world on a server with memory to spare use

1024 to 2048, and a server short of memory can use less (each player's surroundings take about a megabyte). It applies at once and has no effect on any other world.

- It is a convenience for storage and joins, not a way to author: nothing about a map's values changes, and a compact world's pictures are written back out from its tiles when it is exported.

The vegetation twin guard

Two copies of one plant on one spot ("everything had doubles") came from a zone being filled twice over its own objects, or from two entries for one prefab meeting, a vegetation map's `+prefab` being one. In **every** Better Continents world, old or new, a plant is now not placed within 1 m of the same kind of object (the same prefab) that already stands in its zone, or that another vegetation entry placed there in the same pass. It changes vegetation only, never terrain, and only in zones that are generated from now on: a zone that is already generated keeps what it has. An entry never blocks itself, so vanilla's densities and groups stay for every prefab with one entry, except beside an object of that prefab that already stands in its zone (one of a location placed just before, or of an earlier fill); where two entries place the same prefab (a vegetation map's, an alt biome's added vegetation) a second plant within 1 m of the first is dropped, so a zone generated from now on can hold slightly fewer plants than in 0.9.4. Zones reset by a zone command or by Better Continents' own regeneration, and a world put back together from mismatched save files, can no longer double their trees.

`bc_twins` finds the twins a world already has, and removes them:

```
bc_twins                          count them (a client counts what it holds)
bc_twins <radius>                 count plants within <radius> m (0.1 to 5; default 1)
bc_twins remove [radius]          show what a cleanup would delete (exact twins,
                                  or near ones too with a radius)
bc_twins remove confirm [radius]  delete them
bc_twins server remove confirm    the same on a dedicated server (admins only)
```

The copy nearest the ground stays; an object a player made never goes; crops that players grow only count when they stand on one spot. Removal runs where the world is, the host or a dedicated server, a thousand objects a frame, and cannot be undone: back up the world first. In the host's `bc` tree it is `bc twins ...`.

Water follows rebuilt ground

Each 64 m zone has a water volume that works out how deep the sea is under it once, when it starts (`WaterVolume.cs:84-88`: `DetectWaterDepth`, then `SetupMaterial`), and never again. It draws the water dark and reflective where the sea is deep and clear, with the sea bed showing, where it is shallow, and sizes the waves by the same depths. `Heightmap.Regenerate` (`Heightmap.cs:348`) rebuilds a zone's terrain in place and refreshes the terrain's own depths, but not the water's. In vanilla that happens when a terrain edit loads after its zone; a `bc` change and `bc reset` do it to every loaded zone. A zone whose depth moved then kept its old water: a 64 m square with straight edges, deep among shallows or the reverse (trap 40). In **every** Better Continents world, old or new, a postfix on `Heightmap.Regenerate` (`BetterContinents.WaterPatch.cs`) now gives a rebuilt zone's water what a freshly loaded zone's gets whenever its depths no longer match the terrain's; it does nothing while they match, and a world without Better Continents is left alone.

The console tools

`bc_export`, `bc_cache`, `bc_import`, `bc_altbiomes`, `bc_twins` and, since 0.10.4, `bc_bake` (chapter 12) are commands of their own that every player has, where `bc` itself is a cheat for whoever runs the world. They now share one plumbing: each answers the console that asked, the log (`bc_altbiomes` skips it: its reports are long) and, when a dedicated server runs it for an admin (`bc_export server ...`, `bc_twins server ...`, `bc_bake server ...`), that admin. A server export tells the admin when it ends. The host's `bc` tree also has `bc cache` and `bc twins`. Every group of the tree lists itself with `bc <group> help` (`bc h help`, `bc heat help`, ...), and appendix B has the generated table of the groups and commands.

`bc h alpha`, `bc reload lm` and `bc reload terrain` (`bc` commands: cheats on, on the host) read their map again at once, by the loaded world's own rule: a version 12 world reads the location map's pins exactly, a terrain legend's Expand World Data ground names and the heightmap's alpha as a blend; a version 11 world reads them as 0.9.4 did (chapter 6).

The config

0.10 reads an existing `BetterContinents.cfg` as it is: every key keeps its name and section (two are gone, below) and the new keys appear in it. Except where noted, a setting is a default for the worlds you create afterwards, and an existing world uses what it stored; the exceptions are `Override version`, below, and the settings each machine keeps for itself (appendix B marks them live: `Settings Transfer Rate`, `Map Memory`, `Max Map Size`, `Wide Sectors` and the `[09 BetterContinents.Export]` group).

- **Heightmap Amount is 1 everywhere**: the config, the console, the export's default and the import's fallback (chapter 2). A `BetterContinents.cfg` whose `[09 BetterContinents.Export] Default Heightmap Amount` is 2, the default of 0.9.0 to 0.9.4, is set to 1 once, with a log line; `[07 BetterContinents.Misc] Config Version` records that it happened. Set the value again afterwards if 2 is what you want.
- **Debug Reset Command is empty by default**: Better Continents regenerates the zones itself (chapter 6), and a console command written there runs instead. A `BetterContinents.cfg` that still holds the 0.9 default, `zones_reset start`, is set to empty once, with a log line that says how to go back; `Config Version` records it (3 since 0.10.2, which sets an old `Forest Scale = 1` to 0.5 once the same way, chapter 2).
- **Two keys are gone**: Fix Water Color and Ocean Channels (`[01 BetterContinents.Global]`), with the `bc g fixwatercolor` command: the first never did anything and the second matters only to settings formats 1 to 6. A world that saved them still loads them.
- **New keys in 0.10.0**: `Compact Maps` and `Map Memory` in `[07 BetterContinents.Misc]` (above) and the hidden `Config Version`.
- **In 0.10.3**: `Heightmap Amount` takes 0 to 81 (it took 0 to 5), `Default Heightmap Amount` 0.01 to 81 and `Default Size` 128 to 16384. New keys: `Max Map Size` in `[07 BetterContinents.Misc]` and `Largest Size` in `[09 BetterContinents.Export]`, both 4096, 8192 or 16384 (the default) and both applied while the game runs (chapters 2 and 9); `Wide Sectors` in `[07 BetterContinents.Misc]`, Auto (the default), On or Off, read by the machine that runs a world when the world is made and each time it loads (above); and `High Terrain` in `[02 BetterContinents.Heightmap]`, Auto (the default), On or Off, which a new world takes when it is made and keeps (chapters 2 and 3). `Compact Maps` becomes Auto (the default), On or Off; an old `true` reads as On and `false` as Auto. Fine heights have no key: each export chooses them (`bc_export fine=auto|off|4|8`, chapter 9).
- **Small changes that hold for every world**: `Spawnmap File` and `spawnmap.png` are only ever the creature spawn map (a missing `locationmap.png` is no longer looked for under that name, and nothing beside it is renamed); a new config file selects the preset `Disabled`, while an existing `Vanilla`

stays and still makes a world without Better Continents, with an error in the log; and the `bc` console's defaults and ranges are the config's own, since both read the same declaration.

- **Ranges**: `Start Position X` and `Y` take -1,000,000 to 1,000,000 m (the config stopped at ±10,500, the console at 0 to 1), and `Biome precision` takes 0 to 31 in the config and the console (a version 11 world uses at most 5).
- `Override version` **is read at every save**, so editing it changes the next autosave of a loaded world and what a host sends to joining players.
- Every other section and key keeps its name, so an existing `BetterContinents.cfg` reads as before. Appendix B is the generated reference of every one.

11 Recipes

Six complete setups, each of which works on its own: the goal, the map sizes and the memory and time to expect, the folder, the whole config with its section headers, the legend files exactly as Better Continents reads them, the steps at each level, the log lines that prove it worked, the checks to do in the game and the mistakes that recipe invites. Every number is a measured one, with where it was measured; a number nobody measured is said to be unmeasured.

Recipe	What you make	Map size	Main settings
1	A small island world from hand-drawn maps	2,048 px over 21 km	the defaults, Amount 2
2	A continent with planted alt biomes and pinned locations	2,048 px over 21 km	Amount 2, the forest pair, an alt-biome override
3	A big world past 16 km, with Wide Sectors	4,096 px over 49 km	World Size 24,000, Wide Sectors Auto
4	A 16k world with tall terrain and fine heights	16,384 px over 65.5 km	Amount 81, High Terrain Auto, fine heights
5	An export, edit and re-import round trip	any, 2,048 to 4,096 px	`bc_export`, `bc_import`
6	A server and its players	the world's own	a client-made world, the transfer rate, `.bcworld`

A seventh, a town file from a script to a world (Better Continents 0.10.4 or later), closes chapter 12, with the baking it belongs to.

The levels. Chapter 1 names four levels of what you can do, and every recipe says what to do at each:

Level	You can	So in a recipe you
Talk	only write messages	give the human every click, path and file content, and ask for the log lines back
Files	read and write the human's files	write the legends and the config yourself, and read the log
Commands	run commands	also make the maps with a script, check them, and render pictures
Game	have Valheim with Better Continents run, by you or the human, and read its log	also prove the world from its log and from `bc info`

The first three add up; **Game** is about the world, not your tools (chapter 1). Whatever your level, the human makes the world in the game's New World screen unless they have asked you to.

Before any recipe: the places, the checks and the proof

Where things are. From the Guide's install and export chapters:

What	Where
The config	`BepInEx/config/BetterContinents.cfg`: in the game folder, or in the mod manager's profile folder
The log	`BepInEx/LogOutput.log`, beside the config; written again at every start of the game, so copy it first
The worlds	`worlds_local/` in the game's save folder (below)
Exports, presets, `.bcworld` files	`BetterContinents/` in the same save folder

The save folder (chapter 1 has the same list):

```
Windows:
  %USERPROFILE%\AppData\LocalLow\IronGate\Valheim
Linux, the native game:
  ~/.config/unity3d/IronGate/Valheim
Linux, the Windows game under Proton:
  ~/.local/share/Steam/steamapps/compatdata/892970/pfx/drive_c/users/steamuser/AppData/LocalLow/IronGate/Valheim
```

How a config line is written. `BetterContinents.cfg` is a BepInEx file: a section header in square brackets, then `Key = value` lines under it, a dot for decimals. Edit the lines that are there, or add the ones that are not, under their own section; Better Continents writes every other key at its default the next time the game starts, and it reads the file again whenever it changes, so a New World made afterwards in the same session uses the new values. A key under the wrong section is not that setting. The recipes show only the lines that differ from the defaults, plus `Enabled`, which is the switch.

The folder check. Before a world is made, check the folder. This script walks each PNG's chunks without decoding a pixel, and reads the legends (standard Python, no packages): on a real 16,384 px export of 15 files it took 0.2 s and 99 MB. Save it as `mapcheck.py`:

```
#!/usr/bin/env python3
"""Checks a Better Continents map folder before a world is made from it: each PNG's size and
kind, the height records, whether heightmap-fine.png still fits heightmap.png, and the legends.
Standard library only; decodes no pixel. Usage: python3 mapcheck.py <folder>"""
import os, re, struct, sys, zlib

KINDS = {0: 'grey', 2: 'RGB', 3: 'palette', 4: 'grey+alpha', 6: 'RGBA'}
GREY16 = ('heightmap.png', 'forestmap.png', 'heatmap.png')        # 16-bit grey: smooth values
COLOUR = ('biomemap.png', 'locationmap.png', 'altbiomemap.png')   # exact legend colours
BIOMES = {'none', 'meadows', 'blackforest', 'swamp', 'mountain', 'plains', 'mistlands',
          'ashlands', 'deepnorth', 'ocean'}
HEX_OR_RGB = re.compile(r'#?[0-9A-Fa-f]{6}|\d{1,3}\s*,\s*\d{1,3}\s*,\s*\d{1,3}')

def png_info(path):
    with open(path, 'rb') as f:
        data = f.read()
    if data[:8] != b'\x89PNG\r\n\x1a\n':
        return None
    w, h, depth, ctype, _, _, interlace = struct.unpack('>IIBBBBB', data[16:29])
    texts, at = [], 8
    while at < len(data):
        n, kind = struct.unpack('>I4s', data[at:at + 8])
        if kind == b'tEXt':
            key, _, text = data[at + 8:at + 8 + n].partition(b'\0')
            texts.append((key.decode('latin-1'), text.decode('latin-1')))
        at += 12 + n
    return w, h, depth, KINDS.get(ctype, '?'), interlace, texts, zlib.crc32(data)

def main(folder):
    bad = 0
    def say(ok, text):
        nonlocal bad
        bad += not ok
        print(('  ok   ' if ok else '  FIX  ') + text)
    infos = {}
    for name in sorted(os.listdir(folder)):
        if name.endswith('.png'):
            info = png_info(os.path.join(folder, name))
            if info is None:
                say(False, f'{name}: not a PNG'); continue
            w, h, depth, kind, interlace, texts, crc = infos[name] = info
            say(w == h and w <= 16384 and not interlace,
                f'{name}: {w} x {h}, {depth}-bit {kind}'
                + (', interlaced (read whole: save it plain)' if interlace else '')
                + ('' if w == h else ', not square'))
            if name in GREY16 and (depth != 16 or kind not in ('grey', 'grey+alpha')):
```

```
                say(False, f'{name}: save it as 16-bit grey')
            if name in COLOUR and (depth != 8 or kind not in ('RGB', 'RGBA', 'palette')):
                say(False, f'{name}: save it as 8-bit RGB')
            for key, text in texts:
                if key == 'BetterContinents':
                    print(f'       {name} record: {text}')
    fine = infos.get('heightmap-fine.png')
    if fine and 'heightmap.png' in infos:
        record = {}
        for key, text in fine[5]:
            if key == 'BetterContinents':
                for part in text.split(';'):
                    if '=' in part:
                        k, v = part.split('=', 1); record[k.strip()] = v.strip()
        named, actual = record.get('Heightmap CRC-32'), '%08X' % infos['heightmap.png'][6]
        say(fine[2] == 8 and fine[3] == 'grey', 'heightmap-fine.png is 8-bit grey')
        say(fine[0] == infos['heightmap.png'][0],
            'heightmap-fine.png is the size of heightmap.png')
        say(named in (None, actual),
            f'heightmap-fine.png names heightmap.png CRC-32 {named}; heightmap.png is {actual}'
            + ('' if named in (None, actual) else ': heightmap.png was written again, '
               'make the fine file again or delete it'))
    for legend in ('biomemap.txt', 'locationmap.txt', 'altbiomemap.txt'):
        path = os.path.join(folder, legend)
        if not os.path.exists(path):
            if os.path.exists(path[:-4] + '.png'):
                print(f'  note {legend} is missing: Better Continents writes the default'
                      ' legend')
            continue
        for number, line in enumerate(open(path, encoding='utf-8-sig'), 1):
            line = line.strip()
            if not line or line.startswith('#'):
                continue
            name, colon, colour = (line.partition(': at ') if ': at ' in line
                                   else line.rpartition(':'))
            colour = colour.split(' #')[0].strip()
            good = (bool(colon) and bool(HEX_OR_RGB.fullmatch(colour))
                    or legend == 'altbiomemap.txt' and ': at ' in line)
            if legend == 'biomemap.txt' and good and name.strip().lower() not in BIOMES:
                print(f'  note {legend} line {number}: {name.strip()} is not a Valheim 1.0'
                      ' biome (an Expand World Data one?)')
            if not good:
                say(False, f'{legend} line {number}: {line!r} is not "Name: colour"')
    print('no problem found' if not bad else f'{bad} problem(s) to fix')
    return 1 if bad else 0

if __name__ == '__main__':
    sys.exit(main(sys.argv[1]))
```

On recipe 1's folder, and on a folder of recipe 4's kind (made with `fine_ref.py`, below):

```
$ python3 mapcheck.py MyIslands
  ok   biomemap.png: 2048 x 2048, 8-bit RGB
  ok   heightmap.png: 2048 x 2048, 16-bit grey
  ok   locationmap.png: 2048 x 2048, 8-bit RGB
no problem found

$ python3 mapcheck.py tall81
  ok   heightmap-fine.png: 1024 x 1024, 8-bit grey
       heightmap-fine.png record: Fine Format = 1; Fine Bits = 4; Heightmap CRC-32 = D39773FB
  ok   heightmap.png: 1024 x 1024, 16-bit grey
       heightmap.png record: Heightmap Amount = 81.0; Sea Level Adjustment = 0.5
  ok   heightmap-fine.png is 8-bit grey
  ok   heightmap-fine.png is the size of heightmap.png
  ok   heightmap-fine.png names heightmap.png CRC-32 D39773FB; heightmap.png is D39773FB
no problem found
```

And on a broken legend: `FIX biomemap.txt line 2: 'BlackForest 007F00' is not "Name: colour"`. It does not look at the pixels: a biome map painted with soft edges passes it (chapter 2 says why that matters), and only the game's log proves a world.

The proof every recipe shares. Each Better Continents line in `LogOutput.log` carries `[BetterContinents]` (the full line reads `[Info : Unity Log] [BetterContinents] ...`). When the human presses Create, a world made from a folder logs, in this order (`<...>` is the world's own):

```
New world <name>: making its settings from 'From Config' while the game goes on
Time to load <folder>/heightmap.png: <N> ms                 (one line per map the folder has)
New world <name>: its settings are ready, in <N> s
[Saving][<name>] bWorldBeingCreated flag set, first time save of <name>, applying selected preset 'From Config'
Version 12
Heightmap file (<size>) <folder>/heightmap.png
Heightmap amount 2, blend 1, add 0, mask 0                  (the Amount you set)
Heightmap overrides ALL
Biomemap file (<size>) <folder>/biomemap.png
```

What you read	It means
`applying selected preset 'From Config'`	the world was made from the config and its `Directory`
`applying selected preset 'Disabled'`, then `BC is disabled for this world, skipping save`	a vanilla world: the dropdown said Disabled (trap 17)
`Version 12`	a new world (chapter 10); `Version 11` means `Override version` is set
`Heightmap amount 2`	the Amount the heights are read at: it must be the Amount the map was drawn for
`Heightmap file (2048)`, `Biomemap file (2048)`	the map was read, at that size; `Heightmap disabled` means it was not found
`Cannot use texture <file>: it is W x H pixels ...`	the map was refused (trap 44)

> **Check it:** in the game, with Debug Mode on (or `devcommands` typed), `bc info` in the console (F5) prints the same dump for the loaded world. Without Debug Mode, read the log.

Recipe 1: a small island world from hand-drawn maps

Goal. Five islands in the vanilla 21 km square, drawn by hand or by a short script: a meadow island at the spawn, a mountain island high enough for Moder, a swamp and plains island, a Mistlands island and a plains isle. No Ashlands and no Deep North: a world without them has none of their locations (no Fader, no Deep North content), which is right for a building or casual world and wrong for a full playthrough.

Sizes and costs. Maps of 2,048 px (10.25 m a pixel) over World Size 10,000 + Edge Size 500, the size every recipe of this document was proved at; 1,024 px (20.5 m a pixel) works for a quick look. The script below took 9 s at 1,024 px and 34 s at 2,048 px (77 MB of memory) on one machine, standard Python only. Making the world: not measured for this map; a 2,048 px set of seven maps at the same size took 18 s of location placement (`Genloc duration: 18089 ms`, chapter 8).

The folder.

```
MyIslands/
  heightmap.png      16-bit grey, 2048 x 2048
  biomemap.png       8-bit RGB, 2048 x 2048, only legend colours, hard edges
  biomemap.txt       the biome legend
  locationmap.png    8-bit RGB, black, one 3 x 3 dot for the start temple
  locationmap.txt    the location legend
```

The config (`BepInEx/config/BetterContinents.cfg`):

```
[00 BetterContinents.Debug]
Enabled = true
Directory = C:\Users\<you>\Maps\MyIslands

[02 BetterContinents.Heightmap]
Heightmap Amount = 2

[07 BetterContinents.Misc]
SelectedPreset = From Config
```

World Size 10,000 and Edge Size 500 are the defaults, so they need no line. Amount 2 because the default, 1, has a ceiling of 140 m above the water, under Moder's 150 m (chapter 2); at Amount 2 white is 340 m.

The legends, exactly as Better Continents reads them. `biomemap.txt` is `Name: colour`, one a line (a colour is `RRGGBB`, `#RRGGBB` or `r,g,b`; blank lines and `#` lines are skipped):

```
Ocean: 0000FF
Meadows: 00FF00
BlackForest: 007F00
Swamp: 7F7F00
Mountain: FFFFFF
Plains: FFFF00
Mistlands: 7F7F7F
```

`locationmap.txt` names one location by its prefab name:

```
StartTemple: 255,0,0
```

What to draw. At Amount 2 and the default Sea Level, a grey value `v` (0 to 1) stands `(v x 2 - 0.30) x 200` m above the water:

Ground	Altitude	Grey (0..1)	16-bit value	8-bit view
deep sea floor	-60 m	0	0	0
the waterline	0 m	0.15	9,830	38
a beach	2 m	0.155	10,158	40
meadows	10 to 25 m	0.175 to 0.2125	11,469 to 13,926	45 to 54
Moder's lowest site	150 m	0.525	34,406	134
white	340 m	1	65,535	255

Paint the heightmap in 16 bits (an 8-bit picture steps every 1.57 m at Amount 2, chapter 2). Paint the biome map with anti-aliasing off on every brush and fill, so that every pixel is a legend colour; paint the water around each island Ocean. Keep the spawn island's middle under 46 m: within 1,000 m of the centre Better Continents cuts higher ground to 46 m above the water unless `Mountains Allowed At Center = true` (`[01 BetterContinents.Global]`), fully within 600 m and fading out at 1,000 m. Draw the start temple's dot as a 3 x 3 block of 255,0,0 on dry meadow.

Or make the maps with a script (Commands level). This one writes all five files, with nothing but the standard library; change `ISLANDS` to change the layout. Save it as `island.py`:

```
#!/usr/bin/env python3
"""A small island world for Better Continents, over the vanilla square (World Size 10,000 +
Edge Size 500 = 21,000 m): heightmap.png (16-bit grey) for Heightmap Amount 2 and Sea Level
Adjustment 0.5, biomemap.png (8-bit RGB), locationmap.png (black, with a 3 x 3 pin for the
start temple) and their legends. Standard library only.
Usage: python3 island.py <size, e.g. 2048> <folder> [World Size + Edge Size, default 10500]"""
import math, os, struct, sys, zlib

AMOUNT = 2.0          # Heightmap Amount: write the same into BetterContinents.cfg
```

```
# World Size + Edge Size: half the square, in metres
HALF = float(sys.argv[3]) if len(sys.argv) > 3 else 10500.0
K = HALF / 10500.0    # the layout below is for 10,500 m; a bigger world scales it
COLOURS = {'Ocean': (0, 0, 255), 'Meadows': (0, 255, 0), 'BlackForest': (0, 127, 0),
           'Swamp': (127, 127, 0), 'Mountain': (255, 255, 255), 'Plains': (255, 255, 0),
           'Mistlands': (127, 127, 127)}
# Islands: centre x, z (metres, z north), radius, highest altitude (metres above the water),
# and the biome by altitude.
ISLANDS = [(0, 0, 2600, 70, lambda a: 'Meadows' if a < 22 else 'BlackForest'),
           (3600, 3400, 2000, 260, lambda a: 'BlackForest' if a < 70 else 'Mountain'),
           (-3900, -2900, 1900, 45, lambda a: 'Swamp' if a < 7 else 'Plains'),
           (4300, -2700, 1700, 90, lambda a: 'Mistlands'),
           (-4300, 2900, 1600, 55, lambda a: 'Plains' if a > 8 else 'Meadows')]

def png(path, size, rows, depth, colour_type):  # a plain PNG: not interlaced, one IDAT
    chunk = lambda t, d: (struct.pack('>I', len(d)) + t + d
                          + struct.pack('>I', zlib.crc32(t + d)))
    header = struct.pack('>IIBBBBB', size, size, depth, colour_type, 0, 0, 0)
    data = zlib.compress(b''.join(b'\0' + r for r in rows), 6)
    with open(path, 'wb') as f:
        f.write(b'\x89PNG\r\n\x1a\n' + chunk(b'IHDR', header) + chunk(b'IDAT', data)
                + chunk(b'IEND', b''))

def noise(x, z):                                # smooth value noise, about 0..1, four octaves
    total, amp, freq = 0.0, 0.5, 1 / 1400.0
    for o in range(4):
        u, v = x * freq + o * 17.3, z * freq - o * 9.1
        i, j = math.floor(u), math.floor(v)
        fu, fv = u - i, v - j
        fu, fv = fu * fu * (3 - 2 * fu), fv * fv * (3 - 2 * fv)
        h = lambda a, b: (math.sin(a * 127.1 + b * 311.7) * 43758.5453) % 1.0
        top = h(i, j) + (h(i + 1, j) - h(i, j)) * fu
        bottom = h(i, j + 1) + (h(i + 1, j + 1) - h(i, j + 1)) * fu
        total += amp * (top + (bottom - top) * fv)
        amp, freq = amp / 2, freq * 2
    return total / 0.9375

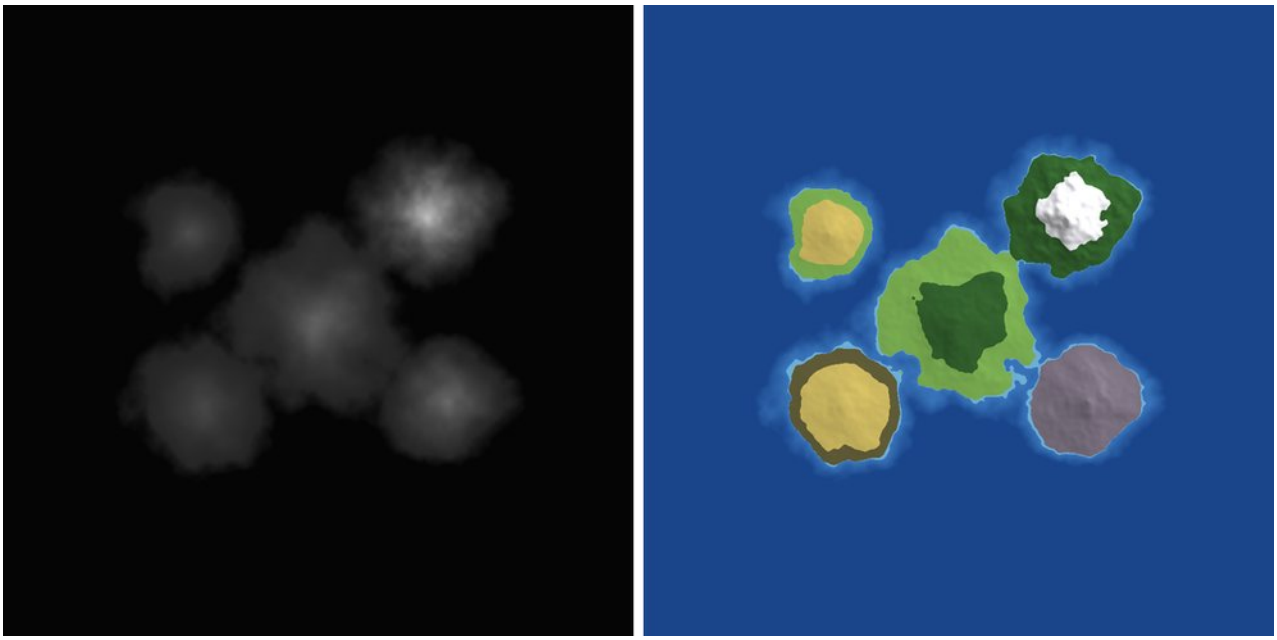
def point(x, z):                                # (altitude in metres, biome name)
    best, biome = -60.0, 'Ocean'
    n = noise(x / K, z / K)
    for cx, cz, r, peak, rule in ISLANDS:
        d = math.hypot(x - cx * K, z - cz * K) / (r * K) + 0.9 * (n - 0.5)
        if d < 1:
            a = peak * (1 - d) ** 1.4 + 6 * (n - 0.5)
        else:
            a = -50 * min(1.0, (d - 1) / 0.35) - 1
        if a > best:
            best, biome = a, rule(a) if a > 0 else 'Ocean'
    return best, biome

def main(size, folder):
    os.makedirs(folder, exist_ok=True)
    heights, biomes = [], []
    for row in range(size):                         # row 0 is north
        z = (0.5 - row / (size - 1)) * 2 * HALF
        h, b = bytearray(), bytearray()
        for col in range(size):
            x = (col / (size - 1) - 0.5) * 2 * HALF
            a, name = point(x, z)
            # altitude = (v * Amount - 0.30) * 200 at Sea Level 0.5, so:
            v = (a / 200 + 0.30) / AMOUNT
            h += struct.pack('>H', max(0, min(65535, round(v * 65535))))
            b += bytes(COLOURS[name])
        heights.append(bytes(h)); biomes.append(bytes(b))
    png(os.path.join(folder, 'heightmap.png'), size, heights, 16, 0)
    png(os.path.join(folder, 'biomemap.png'), size, biomes, 8, 2)
    pins = [bytearray(size * 3) for _ in range(size)]
    # the start temple, 300 m east of the centre
```

```
    c = round(0.5 * (size - 1)) + round(300 / (2 * HALF) * (size - 1))
    r = round(0.5 * (size - 1))
    for dr in (-1, 0, 1):
        for dc in (-1, 0, 1):
            pins[r + dr][(c + dc) * 3:(c + dc) * 3 + 3] = bytes((255, 0, 0))
    png(os.path.join(folder, 'locationmap.png'), size, [bytes(p) for p in pins], 8, 2)
    with open(os.path.join(folder, 'biomemap.txt'), 'w') as f:
        f.write(''.join(f'{k}: {r:02X}{g:02X}{b:02X}\n' for k, (r, g, b) in COLOURS.items()))
    with open(os.path.join(folder, 'locationmap.txt'), 'w') as f:
        f.write('StartTemple: 255,0,0\n')
    print(f'{folder}: {size} x {size} px, {2 * HALF / (size - 1):.2f} m a pixel, '
          f'Heightmap Amount {AMOUNT:g}')

if __name__ == '__main__':
    main(int(sys.argv[1]), sys.argv[2])
```

```
$ python3 island.py 2048 MyIslands
MyIslands: 2048 x 2048 px, 10.26 m a pixel, Heightmap Amount 2
```



What `island.py` writes, at 1,024 px: the heightmap (left, as written) and its biomes shaded by height (right, drawn for this page). The round islands and the ringed biomes are the tells of chapter 4: a sample to start from, not a finished map. It was not built in a game for this page.

The steps.

Level	What you do
Talk	Send the human the table above, the two legend files and the three config blocks, word for word. Tell them: make the folder; draw the three pictures at 2,048 x 2,048 in an editor that saves 16-bit grey PNG; save the legends beside them (plain text, UTF-8); open `BetterContinents.cfg` in a text editor and set the lines; then the New World steps below. Ask them to send back every log line that contains `[BetterContinents]`, from `New world` to `Biomemap file`
Files	Write `biomemap.txt`, `locationmap.txt` and the config lines yourself, and read the PNGs' headers if you can (a PNG's bytes 16 to 25 are width, height, bit depth and colour type). Ask the human to draw or supply the pictures
Commands	Run `island.py` (or your own generator) and `mapcheck.py`; show the human a rendered picture before going further
Game	After the world is made, read `LogOutput.log` against the proof below, and `bc info` in the world

New World, at every level: quit Valheim if it runs and start it with the profile that has Better Continents; Start, New World; a name never used before; the Better Continents dropdown beside the seed set to **From Config**; Create.

The proof (in the log, besides the shared lines above):

```
Location file (2048) <folder>/locationmap.png
Selected StartTemple for spawn position <x>, <y>
Position of StartTemple (1/1) overriden: set to (<x>, <y>, <z>)
```

The second line is the pin as read, in fractions of the map; the third is the pin as placed, in metres (x east, y up, z north). There is no `was NOT placed` line. Then the game places every other location; `Failed to place all <name>, placed N out of M` lines name the ones it could not fit, which on a small world is expected for the area-hungry ones (goblin camps, towers, villages).

In the game. You spawn at the start temple on the meadow island. The large map shows five islands. Sail to the mountain island: its peak stands about 230 m above the water. `bc locs` (Debug Mode) counts every location the world placed; check that `Eikthyrnir`, `GDKing`, `Bonemass`, `Dragonqueen`, `GoblinKing` and `Mistlands_DvergrBossEntrance1` (the bosses' altars, Eikthyr to the Queen) have at least one each.

Common mistakes.

Bad	What you see	Good
Soft (anti-aliased) biome edges	a fringe of a third biome along every coast: a Meadows and Ocean blend snaps to Mistlands	hard edges, exact legend colours
An 8-bit heightmap	terraced beaches, steps of 1.57 m	16-bit grey
Drawn for Amount 2, made at the default 1	every island 30 m lower: lowlands under water, the world mostly sea	`Heightmap Amount = 2` in the config, and `Heightmap amount 2` in the log
The dropdown left at Disabled	a vanilla world, no warning	From Config
A hill at the spawn	a plateau cut flat at 46 m within 1 km of the centre	keep the centre low, or `Mountains Allowed At Center = true`
Swamp painted on ground 10 m up	Bonemass short (it wants swamp 0 to 2 m above the water)	swamp on the flats just above the waterline
A reused world name	the old world's files and cache alias it	a new name every time

Recipe 2: a continent with planted alt biomes and pinned locations

Goal. A generated continent world with chosen alt biomes planted on chosen regions, and the bosses and vendors pinned where the story wants them: ProximaMaxi's setup, map version 13, which the human approved. Chapter 8 is the same build told as a validation; this is the recipe.

Sizes and costs (measured on one machine). Seven maps of 2,048 px over the vanilla 21 km square. The generator runs in about 40 s at 2,048 px and the four checks in a few minutes (chapter 6). The world: `Genloc duration: 18089 ms` for 183 locations; 23 planted regions; 34 `Failed to place all` lines, none of them a progression site (chapter 8).

The folder (`fire-and-ice/`, the real one):

```
fire-and-ice/
  heightmap.png      5.3 MB   16-bit grey, 2048 x 2048
  biomemap.png        61 KB   8-bit RGB         biomemap.txt     162 bytes
  heatmap.png        385 KB   16-bit grey       (exactly 0 off the Ashlands)
  forestmap.png      1.8 MB   16-bit grey
  lavamap.png         88 KB   16-bit grey
  locationmap.png     12 KB   8-bit RGB         locationmap.txt  217 bytes
  altbiomemap.png     32 KB   8-bit RGB         altbiomemap.txt  588 bytes
```

The config:

```
[00 BetterContinents.Debug]
Enabled = true
Directory = /path/to/fire-and-ice

[02 BetterContinents.Heightmap]
Heightmap Amount = 2

[04 BetterContinents.Forest]
Forest Amount = 0.5
Forest Factor Overrides All Trees = true
Forestmap Multiply = 0
Forestmap Add = 1

[07 BetterContinents.Misc]
SelectedPreset = From Config

[08 BetterContinents.AltBiomes]
Overrides = Wolf Mountain: enabled=false
```

The forest pair (Multiply 0, Add 1) makes the forest map add trees to the game's own pattern, so black is the game's forest, not a clearing (chapter 2). The override stops the game adding a random Wolf Mountain where none was painted (trap 24). Appendix B has the reasons line by line.

The legends, as used in the approved build. `biomemap.txt` lists all ten biomes in the default colours:

```
None: 000000
Meadows: 00FF00
BlackForest: 007F00
Swamp: 7F7F00
Mountain: FFFFFF
Plains: FFFF00
Mistlands: 7F7F7F
AshLands: FF0000
DeepNorth: 00FFFF
Ocean: 0000FF
```

`locationmap.txt` pins nine sites (each a 3 x 3 block of its colour, on a pixel the generator searched for: right biome, right altitude, off the river banks):

```
StartTemple: 255,0,0
Eikthyrnir: 255,153,0
GDKing: 0,255,0
Vendor_BlackForest: 0,0,255
Hildir_camp: 128,0,255
Bonemass: 0,255,255
Dragonqueen: 74,134,232
GoblinKing: 255,255,0
Mistlands_DvergrBossEntrance1: 255,0,255
```

`altbiomemap.txt` plants 19 colours (the comment lines are part of the real file):

```
# ProximaMaxi alt-biome legend (Better Continents 0.8.1): one line per planted colour.
# Black is unplanted land: the game's random placement handles it.
Fortress Mountain: B697AF
Drake Mountain: 5AAAC2
Wolf Mountain: 93C7BC
Goblin Plains: FF7622
Death Plains: 671E07
Lox Plains: CC605F
Hut Swamp: B1332D
Bat Swamp: 382B41
Abomination Swamp: 95B17E
Troll Black Forest: 2C7B76
```

```
Root Black Forest: 6F5E32
Ruin Black Forest: 9D6C4B
Dark Meadows: 2D4613
Peaceful Meadows: ADFFDA
Mushroom: FF969C
Swords Mistlands: 8A7FDB
BroodSwarm Mistlands: 8E2F8E
Rockless Mistlands: D99FE8
Lantern: F2A300
```

The other forms an alt-biome legend takes, from the legend Better Continents writes when there is none (in an alt-biome legend, a `#` with a space on each side starts a comment):

```
Dark Meadows + Lantern: 0080C0      # two alt biomes stacked on the regions of one colour
none: C030F0                        # a plain region, with no alt biome and no random one either
Troll Black Forest: at 2750, -1210  # the whole region at x 2750, z -1210 (world metres)
*Mistlands: 5050A0                  # every Mistlands alt biome that fits the land
!Wolf Mountain: 3366CC              # forced past the game's content rules
```

The steps.

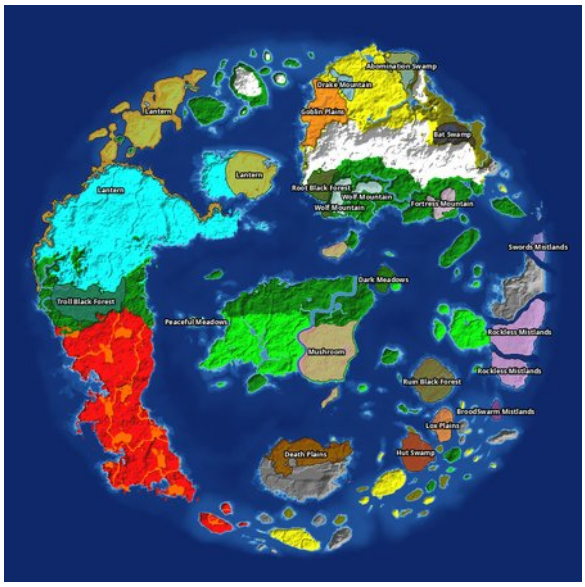
Level	What you do
Talk	Give the human the config and the three legends word for word, and the colour of every region to paint (a table: region, alt biome, colour). Tell them to paint `altbiomemap.png` black everywhere else, with hard edges and patches at least a few pixels wide (the game's grid is 12 m, a pixel 10.25 m)
Files	Write the legends and the config; read the log afterwards
Commands	Run the generator, the four checks and the previews of chapter 6 (steps 3 to 6); show the pictures; then write the config
Game	Read the log against the proof below and replay it with `locations.py --log` (chapter 6, step 11)

The proof (real lines of the approved build, chapter 8):

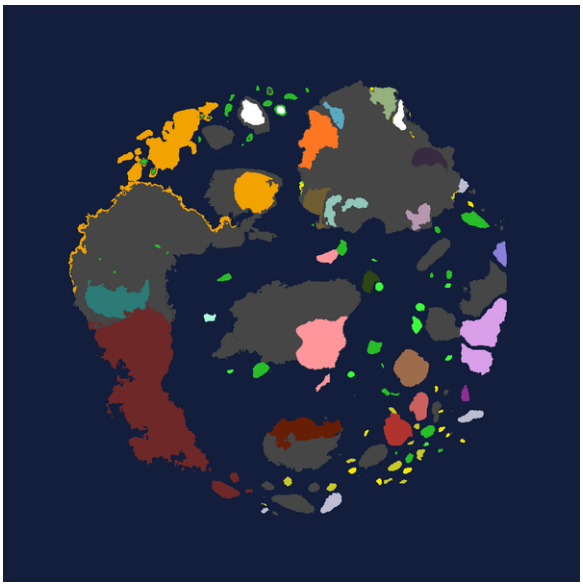
```
Alt-biome legend altbiomemap.txt: 19 colours, 0 points.
Alt-biome map .../altbiomemap.png: 2048x2048, 329922 planted pixels, decoded in 74 ms.
Selected StartTemple for spawn position 0.4941406, 0.3955078  (+ 8 more)
Alt biomes: split 23 planted region(s) out of 22 region(s) (13 fully planted and reused),
» 227235 of 4194304 grid points planted.
Alt biomes (world load): mode Random, placement seed world seed, grid 2048 x 12 m to 10500 m
» (WorldEdge), 296 sectors, 63 with alt biomes (23 planted, 40 random), 23 planted region(s)
  Mushroom [Meadows|BlackForest] 0/1-3 chance 0.2 +3 planted (counted toward the maximum): 20
» of 110 regions qualify ...
  Wolf Mountain [Mountain] 0/0-5 chance 0.2 +2 planted (counted toward the maximum): disabled
» for random placement
```

A world made with Better Continents 0.10.x also prints `Version 12`, and one `Position of <name> (1/N) overriden: set to ...` line for each of the nine pins, with no `was NOT placed` line. A painted colour that matched no legend colour logs `Alt-biome map altbiomemap.png: N pixels in K colours match no legend colour (within 20) and are treated as unplanted.` with the most common ones.

In the game. `bc_altbiomes here` (any player) names the region you stand in and its alt biome; walk into two planted regions and one unplanted one. `bc bosses` (Debug Mode) pins the bosses, the start temple and the trader on the large map: nine of them sit where the location map put them.



13g-alt-labelled.png



13h-altbiome-regions.png

The approved build: the planted alt biomes labelled over the map (left) and the regions the checker found (right: legend colour planted, bright could host a random alt biome, grey could not).

Common mistakes (chapter 7 has each in full):

Bad	What you see	Good
Ocean painted to the waterline everywhere	frozen ships 0 of 50, harbours 7 of 100 (trap 7)	Deep North water to 16 m, Mistlands water to 1 m
A multi-biome alt (Lantern, Mushroom) painted across a coast	dozens of one-point regions (trap 9)	pull its colour back 3 px from foreign shores
A thin spit painted as its own patch	a planted slot that reads as nothing (trap 10)	whole features; regions with an edge under 20 are suspect
Two pins within 64 m	`Location map: X at P was NOT placed: its zone Z already holds H` (trap 34)	sites more than a zone apart
A boss pinned in the sea	the game accepts it (trap 35)	search for a legal pixel near the intended site

Recipe 3: a big world past 16 km, with Wide Sectors

Goal. A world of World Size 24,000 + Edge Size 500 = 24,500 m: more than five times the vanilla area, made with wide sectors so that the game files every object properly. Read chapter 10's "Worlds past 16 km" first: a wide world needs Better Continents 0.10.3 or later from then on, and the human must know that before you stage it.

Sizes and costs. The maps span `2 x 24,500 = 49,000 m`. At 4,096 px a pixel is 11.97 m, just inside the game's 12 m alt-biome grid (4,084 px is the least that keeps it; chapter 9's rule). `island.py` at 4,096 px with `24500` took 116 s and 282 MB on one machine. Measured on one machine, for the costs that size brings:

What	Measured
Wide sectors	about 32 MB on each machine while the world is loaded; planning a save 21 to 31 ms instead of 3 to 4 ms
The alt-biome grid, built at every load on every machine	3,190 points a side and about 0.2 GB at 16,350 m; 6,392 points, about 0.8 GB and about 5 s at 32,764 m; 4,780 points and about 10 s at 24,500 m on the game's own terrain (both on 32 cores)

What	Measured
Starting the world	about 11 s at the vanilla size, about 18 s at 24,500 m on the game's own terrain, about 16 s at 32,764 m with five 16,384 px maps (a 32-core server)
The world file a joining player downloads	not measured for 4,096 px maps; Compact Maps took a 10,501 px set from 126 MB to 10.5 MB (chapter 10)

The folder. As recipe 1 (the same five files), at 4,096 px:

```
$ python3 island.py 4096 BigIslands 24500
BigIslands: 4096 x 4096 px, 11.97 m a pixel, Heightmap Amount 2
```

The config:

```
[00 BetterContinents.Debug]
Enabled = true
Directory = C:\Users\<you>\Maps\BigIslands

[01 BetterContinents.Global]
World Size = 24000
Edge Size = 500

[02 BetterContinents.Heightmap]
Heightmap Amount = 2

[07 BetterContinents.Misc]
SelectedPreset = From Config
Wide Sectors = Auto
Compact Maps = On
```

`Wide Sectors = Auto` is the default: a NEW world past 16,350 m is made wide, and no existing world is touched. Never write `On` unless the human asks for an existing world to be converted, after its backup (it converts every big world the profile opens, one way). `Compact Maps = On` makes the world compact although no map is over 8,192 px, so the world file and every join are smaller; a compact world cannot be opened by Better Continents 0.9, which a wide world already rules out. The legends are recipe 1's.

The steps.

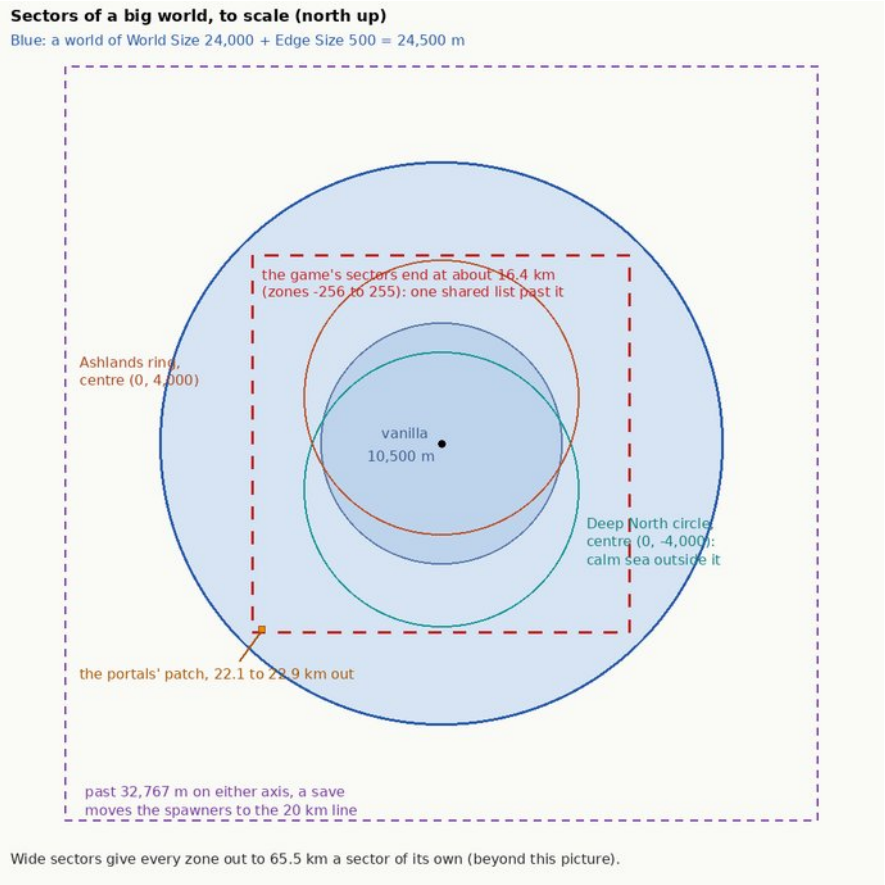
Level	What you do
Talk	Before anything, tell the human in plain words what a big world binds (below, "What to tell the players"), and get a yes. Then as recipe 1, with the World Size lines
Files	Write the legends and the config; read the log
Commands	Make the maps at 4,096 px with `island.py ... 24500` (or your generator, laid out in fractions of the radius, chapter 3); `mapcheck.py`
Game	Read the log against the proof; `bc info` in the world

The proof:

```
World size 24000
The maps span World Size and Edge Size: 49000 m across, and the world is laid out to that size
Compact Maps: the maps are held and saved as compressed tiles, each decoded when the world
» reads it
Sectors: this new world reaches 24500 m, so it is made with wide sectors (Wide Sectors is
» Auto): every zone out to 65504 m has a sector and save chunks of its own.
```

and, each time the world loads, `Sectors: every zone from -1024 to 1023 (to 65504 m) has a sector and save chunks of its own (2048 x 2048 sectors, 32 MB), where the game has 512 x 512.` and one `The minimap's <N> pixels are <M> m: they cover <N x M> m.` line. `bc info` adds `Wide sectors: every zone out to 65504 m has a sector and save chunks of its own, where the game gives everything past 16.4 km one shared sector`.

In the game. Fly or sail past 16.4 km from the centre (the large map's coordinates, or `bc info`'s numbers): the world goes on generating, and the objects there stay where they were after a restart. Far from the north the ground shows the Deep North's look: the game's terrain shader draws it by a fixed circle (chapter 3), whatever the map says. The sea has waves out to the edge, calm only where the map's Deep North is (Better Continents 0.10.5 or later, chapter 13); with 0.10.4 or older it was calm past the game's circle too.



Drawn to scale for this page from the numbers of chapter 10: the example world (blue), the vanilla world, the end of the game's own sectors at about 16.4 km, the portals' patch near their south-west corner, the 32,767 m line, and the game's fixed Deep North and Ashlands circles. Not a game picture.

What to tell the players (send this, in your words):

- Everyone needs exactly the same Better Continents version as the server, 0.10.3 or later. Another version is refused at the join: `Better Continents: world has the mod enabled, but server <version> and client <version> versions don't match`.
- The first join downloads the world's settings and maps at the server's `Settings Transfer Rate` (512 KB/s by default: 52.6 MB take about 1.7 minutes); the game keeps a copy, so the next join is quick. A `.bcworld` file handed out beforehand removes the download (recipe 6).
- Never open this world with an older Better Continents, or without Better Continents: it would come up as vanilla terrain, and a save there writes every object twice. Back up the world folder before any version change.
- The same number of dungeons, altars and villages spread over five times the area: expect long voyages, and portals.

Common mistakes.

Bad	What you see	Good
Maps drawn for 21 km used at World Size 24,000	the same picture stretched: islands 2.3 times bigger, pixels 2.3 times coarser	design in fractions of the radius, and make the maps for the size
2,048 px maps over 49 km	23.9 m a pixel: alt-biome borders off by up to two grid points, blunt coasts	at least 4,084 px
`Wide Sectors = Off`, or a big world made before 0.10.3	far objects share one list; portals or a patch lost past 22.1 km (trap 32)	Auto for new worlds; On only to convert, after a backup
A 0.10.2 machine runs the world once	two of every chest (trap 49)	0.10.3 or later everywhere, always

Recipe 4: a 16k world with tall terrain and fine heights

Goal. The largest world Better Continents 0.10.x makes well: 16,384 px maps over World Size 32,264 + Edge Size 500 (65,528 m, 4 m a pixel), heights read at Heightmap Amount 81 (white is 16,140 m above the water), the high-world rules of chapter 3, and fine heights, so that gentle slopes are smooth.

Sizes and costs. Measured on one machine (a Linux dedicated server) with a real world of five 16,383 px maps at these sizes and Amount 81:

What	Measured
Making the world	the maps read in 17 s; about 0.3 GB above the idle process while read and 0.5 GB at the highest; 1.80 GB for the whole process at its peak (Better Continents 0.10.2 needed 5.83 GB)
The game's New World screen	"Please wait" for 17 to 39 s while the maps are read; the game keeps drawing
The world file	52.6 MB, compact (352.5 MB as pictures)
Fine heights	5.7 MB more in the world file, 1.8 s more to make the world; heights within 0.9 mm of the source where the fine picture applies (up to 109 mm off with the heightmap alone)
Starting the world	about 16 s on a 32-core server
The alt-biome grid	6,392 points a side, about 0.8 GB and about 5 s on 32 cores, at every load on every machine
A join at the default rate	about 1.7 minutes for 52.6 MB (17 s at `MB3`)
One 16-bit 16,384 px map decoded	537 MB

The folder:

```
Tall16k/
  heightmap.png        16-bit grey, 16384 x 16384, with its record
  heightmap-fine.png   8-bit grey, 16384 x 16384, with its record
  biomemap.png  biomemap.txt
  forestmap.png  heatmap.png
  locationmap.png  locationmap.txt
```

The heightmap's record reads `Heightmap Amount = 81; Sea Level Adjustment = 0.5`, the fine picture's `Fine Format = 1; Fine Bits = 4; Heightmap CRC-32 = ...` (the heightmap's own CRC-32). The location map does not count toward Compact Maps' 8,192 px rule.

Save every map as a plain PNG (not interlaced; 8- or 16-bit grey, grey and alpha, 8-bit RGB or RGBA, or a palette): Better Continents reads those a few rows at a time, and any other kind whole, about 1 GB for one 16,384 px colour picture.

Making the fine heights. `fine_ref.py` (in Better Continents' source, `tools/fine_ref.py`; it needs numpy and Pillow, and about 1.5 GB of memory for a 16,384 px pair) writes `heightmap.png` and `heightmap-fine.png` together, in the right order, from float heights:

```
python3 fine_ref.py write Tall16k --from-npy heights.npy --units metres --amount 81
python3 fine_ref.py check Tall16k
```

`heights.npy` is a square float array, row 0 north, of heights in metres with the water at 30 m (the game's own height, not the altitude: add 30 to an altitude). On a 1,024 px heightmap rewritten for Amount 81 (`write tall81 --from-heightmap ... --from-amount 2 --amount 81`), the two commands printed:

```
tall81: heightmap.png (CRC-32 D39773FB) and heightmap-fine.png from island-1024/heightmap.png
» (Heightmap Amount 2, Sea Level Adjustment 0), at Heightmap Amount 81, Sea Level Adjustment 0;
» 4 bits asked, 4 used; f not 0 on 10,858 of 1,048,576 pixels
Fine heights: heightmap-fine.png refines heightmap.png with 4 bits a pixel.
  heightmap.png: 1024 x 1024, 16-bit grey, 0.10 MB, CRC-32 D39773FB
  heightmap-fine.png: 1024 x 1024, 8-bit grey, 0.02 MB, record "Fine Format = 1; Fine Bits = 4;
» Heightmap CRC-32 = D39773FB" (after the image data)
  Heightmap Amount 81 (heightmap.png's record), Sea Level Adjustment 0: a step of heightmap.png
» is 24.72 cm; with 4 bits, 15.45 mm
  under water (below 30 m): 901,481 pixels, 0 of them with f not 0 (the policy keeps 0)
```

Anything that writes `heightmap.png` afterwards (an image editor, `png_extras.py record`) changes its CRC-32, and the fine picture is then ignored. The other way to get a pair is `bc_export ... amount=81` of a world (recipe 5): `fine=auto` writes one from an Amount of about 4.92.

The config:

```
[00 BetterContinents.Debug]
Enabled = true
Directory = /path/to/Tall16k

[01 BetterContinents.Global]
World Size = 32264
Edge Size = 500

[02 BetterContinents.Heightmap]
Heightmap Amount = 81
High Terrain = Auto

[07 BetterContinents.Misc]
SelectedPreset = From Config
Max Map Size = 16384
Compact Maps = Auto
Wide Sectors = Auto
Map Memory = 1024
```

`High Terrain = Auto` gives the high-world rules to a world read above Amount 5; `Compact Maps = Auto` makes it compact (the maps are over 8,192 px, and it has fine heights); `Wide Sectors = Auto` makes it wide (32,764 m is past 16,350 m; it stays under 32,767 m, where the spawner line begins). `Max Map Size = 16384` is the default; a machine with it at 4096 or 8192 refuses the maps. `Map Memory` is the default 512 MB on a player's machine; 1,024 to 2,048 on a server with memory to spare.

The steps.

Level	What you do
Talk	Tell the human the costs first (the table) and what the world binds (recipe 3's list, plus: an older Better Continents gives a fine-heights world vanilla terrain). Give them the config. The 1.80 GB peak was a server process's; a game client's own memory comes on top of what making the world takes (not measured)
Files	Write the config; check the folder's file sizes; read the log

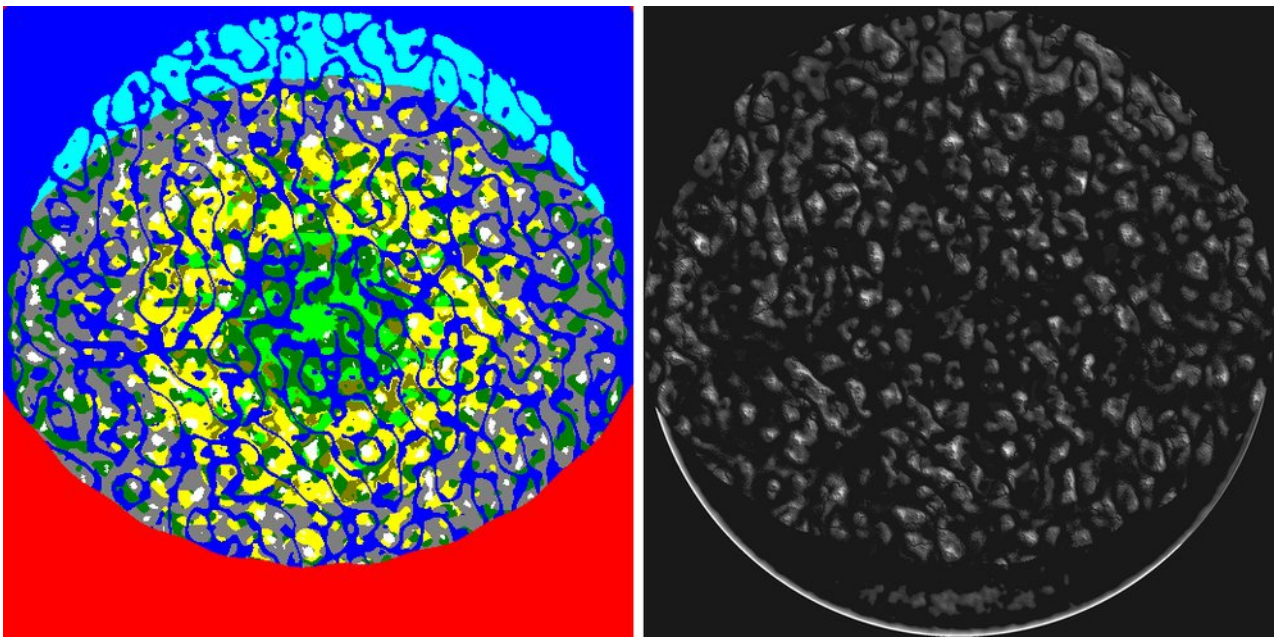
Level	What you do
Commands	Make the heights, `fine_ref.py write`, `fine_ref.py check`, `mapcheck.py`; keep every heavy command under a memory cap
Game	Read the proof; `bc info`

The proof:

```
New world <name>: making its settings from 'From Config' while the game goes on
Fine heights: heightmap-fine.png refines heightmap.png with 4 bits a pixel: ground in steps of
» 1.5 cm at Heightmap Amount 81 (heightmap.png alone: 24.7 cm), 5.7 MB of tiles, 0.17 bits a
» pixel. A Better Continents older than this one cannot read a world made with fine heights.
Compact Maps is on for this world: its heightmap has fine heights (heightmap-fine.png), which a
» world keeps as compressed tiles, and so it does its maps.
Sectors: this new world reaches 32764 m, so it is made with wide sectors (Wide Sectors is
» Auto): every zone out to 65504 m has a sector and save chunks of its own.
New world <name>: its settings are ready, in <N> s
Heightmap amount 81, blend 1, add 0, mask 0
High terrain: the land can reach <N> m, so the game's rules that hold a height are patched
» (what is inside a dungeon, the ground and the grass, altitude limits, the AI's tiles)
```

(The fine-heights line is the real world's: 5.7 MB of tiles is its number.) A map over 8,192 px with no fine heights logs `Compact Maps is Auto and Heightmap <path> is 16384 x 16384 pixels (more than 8192 across), so this world is compact. ...` instead of the first Compact Maps line. A fine picture that is not used logs one `Fine heights: ...` line that says why (chapter 2 lists the fourteen checks); the commonest is `... was made for another heightmap.png (its record names CRC-32 ...; this one is ...). If heightmap.png was edited, make heightmap-fine.png again or delete it.`

In the game. Stand on a gentle slope high up: no stairs. Above 500 m the grass grows and plants, creatures and locations appear above 1,000 m (the high-world rules, chapter 3); without them none of that would.



A real export, at 2,048 px, of a 16k test world read at Amount 81 with fine heights (World Size 15,850 + Edge Size 500, a 32.7 km square): `biomemap.png` *as written (left) and* `heightmap.png` *shaded for this page (right). Chapter 9 says what each file holds.*

Common mistakes.

Bad	What you see	Good
`heightmap.png` touched after the fine picture was made	`Fine heights: ... was made for another heightmap.png` (trap 47)	make the pair last, with `fine_ref.py write`
`Compact Maps = Off`	`Fine heights: ... is not read: Compact Maps is Off ...`; a world file of hundreds of MB	Auto
The world opened by 0.10.0 to 0.10.2	vanilla terrain (trap 48)	0.10.3 or later on every machine
A 16-bit colour or interlaced map	gigabytes while it is read; `... is a picture of a kind that is read whole` (trap 45)	plain PNGs
Made on Better Continents 0.10.2	Amount clamped to 5: the world far too low (trap 56)	0.10.3 or later

Recipe 5: export, edit and re-import

Goal. Take a world that exists (vanilla or Better Continents), write it out as maps, change something (raise an island, repaint a biome, move a boss), and make a new world from the edited maps. Chapter 9 has every option.

Sizes and costs (measured on one machine). An export holds about 100 MB of memory while it runs, at any size; 8,192 px took 43 s and 16,384 px about 3 minutes. Building the New World preset from the maps takes about 50 MB at 2,048 px, 200 MB at 4,096 px, 0.8 GB at 8,192 px and 1.4 GB at 16,384 px; the export makes the preset itself only up to 8,192 px, and `bc_import` makes it above that. Export at 2,048 px or more where alt biomes matter, and at the Amount the edit needs: 2 keeps every vanilla mountain (white 340 m above the water).

The steps.

1. **Export.** In the world, on the machine that runs it (single player or the host; a guest's export holds only the locations they have seen), open the console (F5) and type:

   ```
   bc_export 2048 amount=2
   ```

   It prints, when done:

   ```
   World export finished in <N> s: <save folder>/BetterContinents/<world>/export-<date>-<time>
   World export: ...                                  (a summary, a line each)
   World export: README.txt in the folder says how to make a world from it, in three ways.
   ```

2. **Edit.** Open the folder. Edit `heightmap.png` in 16 bits and `biomemap.png` with hard edges; keep every picture's size. After any edit of `heightmap.png`, delete `heightmap-fine.png` if there is one. A location is one pixel of its colour: move it, never smear it.

3. **Check.** `manifest.json` holds the numbers (`size`, `totalSize`, `heightmapAmount`, `seaLevel`); `mapcheck.py` the kinds and records.

4. **Import.** In the console (main menu or a world):

   ```
   bc_import
   ```

   for the newest export of the loaded or last world, or `bc_import list` and `bc_import <number>`. It prints:

   ```
   World import: Preset "<world> <date> <time>" is made and selected: create a New World to use it.
   World import: <a summary of what it holds>.
   World import: saved as <save folder>/BetterContinents/presets/<name>.BetterContinents.
   ```

5. **New World**, a new name, the dropdown already on the preset, Create.

A real export's `export.cfg` begins with the three ways and then lists the settings that rebuild the world; the lines that matter most (from an export of a test world at Amount 81):

```
[01 BetterContinents.Global]
World Size = 15850
Edge Size = 500
Skip Default Locations = true
Sea Level Adjustment = 0.5
Rivers = false

[02 BetterContinents.Heightmap]
## metres = (pixel / 65535 * amount - 0.15 + Lerp(1, -1, sea level)) * 200.
Heightmap Amount = 81

[04 BetterContinents.Forest]
Forestmap Multiply = 0
Forestmap Add = 1

[08 BetterContinents.AltBiomes]
Mode = PlantedOnly
```

and its `manifest.json` (an excerpt):

```
"size": 2048, "totalSize": 32700, "worldRadius": 15850, "totalRadius": 16350, "metresPerPixel":
» 15.974597,
"pixelToWorld": "x = (column / (size - 1) - 0.5) * totalSize; z = (0.5 - row / (size - 1)) *
» totalSize; row 0 is north",
"heightmapAmount": 81, "seaLevel": 0.5, "fineBits": 4, "fineEncoding": "signed offset, 1/256
» step, format 1",
"import": "bc_import FineEx/export-2026-10-06-23-29-58",
```

Level	What you do
Talk	Give the human the two console lines and what each prints; tell them which file to edit and how (16 bits, hard edges, same size); ask for the `World import:` lines back
Files	Read `manifest.json`, `export.cfg` and `README.txt` in the export folder; write the legend lines the edit needs
Commands	Edit the pictures with a script (a pasted area, a raised island), run `mapcheck.py`, render before and after
Game	Read the `World import:` lines and the creation lines; `bc info` in the new world

The proof. The `World import:` lines above, then the shared creation lines with `Heightmap amount 2` (the export's Amount). A heightmap read at another Amount than its record logs `<file> was made by a world export for Heightmap Amount 2 and Sea Level Adjustment 0.5, but this new world reads it at Heightmap Amount 1 and Sea Level Adjustment 0.5, so its land is not at the heights it was made for. The export's export.cfg has the settings it needs.`

In the game. The edit is there; the rest of the world is the old one's ground (not its buildings: an export holds only the generated world).

Common mistakes: chapter 9's table. The two that cost most: editing the PNGs and expecting the existing world to change (it never does: make a new world), and a 16,384 px export expected to make its own preset (it leaves that to `bc_import`).

Recipe 6: a server and its players

Goal. A dedicated server that runs a Better Continents world, and players who join it.

What is true (read from the code): a dedicated server does not make a Better Continents world from its config. Only the game's New World screen marks a world as being made, so a server that starts a world it does not have makes a vanilla one. Make the world on a client, then copy its folder to the server.

The steps.

1. On a client with the same Better Continents version as the server, make the world (any recipe above) and quit to the main menu.
2. Copy the whole world folder, `worlds_local/<World>/`, to the server's own worlds folder. It holds the world's files and an extension-less `BetterContinents` file with the settings and maps; a host that filters uploads by extension may drop it, and without it the world loads as vanilla.
3. On the server, set `[07 BetterContinents.Misc] Settings Transfer Rate` (the server's value is the one used):

   ```
   [07 BetterContinents.Misc]
   Settings Transfer Rate = MB3
   ```

   and, for a big world on a server with memory to spare, `Map Memory = 1024` to `2048`. Keep `Debug Mode = false` on a server that people play on.
4. Optionally hand out a `.bcworld`: a player whose game holds the world (they have joined once) types `bc_cache export` in the console, which writes `<save folder>/BetterContinents/<world>/<world>-<id>.bcworld` and a note. Players put it in `BepInEx/config/BetterContinents/seed/` (read at every start) or run `bc_cache import <file>`; then their join needs no download. The id changes with any change of the world's Better Continents settings.
5. Start the server, and join.

The proof (the server's log, for a join that downloads):

```
Settings transfer rate: <rate> (server setting)
The <N> MB of settings take about <T> s to send at that rate.
Settings sent: <bytes> bytes in <s> s (<KB/s> KB/s)
```

Measured on 2026-09-25, a client joining a local dedicated server with an 11.8 MB world: 78 s at `Vanilla`, 23 s at `KB512`, 3 s at `Unlimited` (chapter 9). For the bigger worlds the times follow from the rate (recipe 3 and chapter 10).

Level	What you do
Talk	Give the human the copy step with both paths, the server's config lines, and the message a player with another version sees
Files	Copy nothing yourself unless asked: the world folder is the human's (chapter 1); write the server's config lines
Commands	Check the copied folder holds `BetterContinents`; `bc_cache export` can be run by the human in game
Game	Read the server's log lines above, and a client's `bc info` (the settings the server sent)

Common mistakes.

Bad	What you see	Good
The server makes the world on its first start	a vanilla world	make it on a client, copy the folder

Bad	What you see	Good
The `BetterContinents` file left behind in the copy	the world loads as vanilla	copy the whole folder, and check the file is there
One player on another Better Continents version	`Better Continents: world has the mod enabled, but server <v> and client <v> versions don't match`	everyone on the server's version
`Settings Transfer Rate` set on a client	no change in the join time (trap 6 of chapter 9)	set it on the server
A crossplay (PlayFab) connection	the rate cannot be raised	hand out the `.bcworld`

12 Baked placements

New in Better Continents 0.10.4 (October 2026): a world can carry a **layer** of baked pieces. A baked piece looks and stands exactly as a real one, but it is not an object of the world any more: the world neither saves, sends nor updates it one by one. The pieces come from two places, and one world can have both:

1. **An in-game bake.** A player who may (single player, the host, or a server's admin) bakes the pieces players built within a circle or a box, from the export window's **Bake buildings** tab (F7) or the console's `bc_bake`.
2. **A town file.** A program outside the game, a **town compiler**, writes `placements.bcp`. In the folder of maps it builds its towns into a new world; `bc_bake load` brings a newer one into a running world.

This chapter teaches both: helping a human bake a base or a town (part 1), what a town file does in a world (part 2), writing a compiler whose files Better Continents accepts (part 3, with appendix E for every byte), and tuning and checking the result (part 4). A recipe that goes from a script to a world, and the traps, close it.

> **Human: Instance count is the measure.** A bake is judged by the objects it takes out of the world: the objects (ZDOs) the world saves and sends, the live objects in the scene, their renderers and colliders, the instances drawn. Frame time follows from those only in part, and it moves with everything else on the screen. Report what a bake removed, in those counts, before any millisecond.

Two words are easy to mix up. Elsewhere in this document "baked into the world" means what a new world does with its **maps** (chapter 2): it reads them once and keeps them in its settings. Here **bake** means `bc_bake`: pieces taken out of the world's objects and kept as records of the layer. The layer is saved in the same `BetterContinents` settings file as the maps.

What a baked piece is

	A real piece	A baked piece
what holds it	an object of the world (a ZDO), saved and sent one by one	a record of the layer, kept with the world's settings: VALtima's 702,700 records, with their towns' ground and paint, take 2.78 MB
drawn by	the game, one object at a time	Better Continents, every piece of a kind in view in one instanced batch
solid	its own colliders	colliders Better Continents builds per zone (part 3)
wear, support, repair	the game's	none: drawn as new; it holds what rests on it as firmly as the ground
the hammer	repair, move, remove	none: undo or unbake it first
what a player needs	the game	the same Better Continents version as the world (0.10.4 or later)

What stays a real piece is everything that does something: doors, chests, stations, beds, cooking fires, portals, wards, signs, anything that moves, and any part of a mod Better Continents does not know. With the **Town** option those become *protected* real pieces of the layer (nobody can damage or remove them), and comfort decor and lights are baked too (part 1).

Nothing that gives items is ever baked: a piece with a `Pickable`, `PickableItem`, `ItemDrop` or `Plant` (bushes, mushrooms, flax, crops, items on the ground). A town file's records of such a kind are placed once as ordinary objects of the game, unless the file marks the kind as decor (part 2).

At your level

Level	Baking in the game	A town file
Talk	you write the human every click and every console line, and ask for the tab's answer back (its **Copy** button copies it)	you give the writer's code and its command, and the steps to put the file in place
Files	you also read `LogOutput.log` (every `bc_bake` answer is in it)	you write the compiler and the file's folder; the human runs it
Commands	as Files; the bake itself is always the human's, in the game	you run the compiler and `bcp.py check` on its output before the human sees it
Game	the human (or you, if the game runs for you) bakes; you read the dry run before Confirm	a new world, or `bc_bake load`'s dry run, proves the file in the game

Whatever your level: **never press Confirm, or write `confirm`, on a human's world unless they asked for that change**, and read the dry run with them first. A bake is undone with one command, but only from its undo file on the machine that runs the world.

Part 1: baking in the game

The setup

- **The window.** `[09 BetterContinents.Export] Hud = true` turns on the export window. In a world, **F7** opens it and **Bake buildings** is its third tab, beside Export and Import. On a server the server's `Hud` decides for everyone on it.
- **Who may change the layer.** Single player: the player. A game hosted for friends: the host, and players on the host's admin list with `server`. A dedicated server: its admins (`adminlist.txt`), with `server`. No cheats, Debug Mode or `devcommands` are needed. On any player's game, `info`, `stats`, `hide`, `show` and `help` run locally; every other word answers `bc_bake: this runs where the world is. Ask the server: bc_bake server <the same words> (admins only).`
- **One change at a time**, and every change is a **dry run first**: nothing changes until `confirm` (or the tab's **Confirm**), which does exactly what the dry run said, at the place where the dry run was run.

The tab and the console are the same commands. The tab's sections, top to bottom: **Around you** (Radius 8, 16, 32, 64, 128, 256 m; the options **Town**, **Also pieces nobody built (ruins and the like)**, **Convert**; **Bake: dry run**, **Unbake: dry run**, **All: a town file's pieces too**), **Undo and the way out** (**Undo the last change: dry run**, **Unbake the whole world: dry run**), **A town file (placements.bcp)** (**Load the Directory's file: dry run**, **Export this world's layer**), **Look** (**Info**, **Changes**, **Check**, **Orphans**, **Destroy orphans: dry run**, **What is drawn here (stats)**, **Hide baked pieces**), and **What bc_bake said**, where every answer appears, with **Copy** and **Clear**.

The console (F5), as `bc_bake help` lists it:

```
bc_bake help | info | stats | hide | show | list | check [all | <zx> <zz>] | area <radius> [at
» <x> <z>] [town] [any] [convert] [confirm] | box <x1> <z1> <x2> <z2> [town] [any] [convert]
» [confirm] | unbake <radius> [at <x> <z>] | box <x1> <z1> <x2> <z2> | world [all] [confirm] |
» undo [<n>] [confirm] | load [<file>] [convert] [confirm] | export [<file>] | orphans
» [destroy] [items] [confirm] | drop compiler | bake <n> [confirm] | drop <world> [confirm] |
» server <the same>
```

A radius is 4 to 256 m; a box is up to 512 x 512 m. The words `town`, `any`, `convert`, `all` and `confirm` may stand anywhere after the verb. On the host, `bc bake ...` in the debug tree runs the same words.

What a bake takes, and what stays

A piece counts by its own position (its pivot), at any height: a tower's top is in the circle when its foot is. Better Continents classifies each object by its prefab's components, its creator and the words, in this order; the dry run counts each group and names its reasons.

Rule	The piece	What the bake does
1	not a piece	not touched: "not a piece"
2	has a `TerrainModifier` (it shapes the ground only while it is an object)	not touched: "ground work"
3	carries a bake id already	not touched: "already baked"
3b	has a `Pickable`, `PickableItem`, `ItemDrop` or `Plant`	not touched: "gives items when used", with any words
4	no creator (the game placed it: ruins, villages), without `any`	not touched: "not built by a player"
5	a body that moves (a cart, a ship, a creature)	not touched: "carts and ships"
6	a light, an area, comfort, connection data, or any component outside the plain drawing and colliding ones	stays a real piece; with `town`, adopted as a protected Live record (below)
7	a chair (and nothing else against it)	baked as a Seat: drawn, and a seat you can sit on near you
8	anything else	baked as a Static record

With `town`, two kinds of rule-6 piece are baked after all: one whose only reason was **comfort** (tables, banners, rugs, plants; a chair or bench becomes a Seat), which then gives no comfort; and one whose only reasons are **light and fire** (torches, lanterns, candles, braziers), which becomes a **lit copy**: it burns without fuel near the player and gives no warmth. A fire a cooking station needs (its fire area) stays real. Every other rule-6 piece (doors, chests, stations, beds, portals, wards, signs) stays a real object and becomes protected: nobody can damage or remove it, and the weather does not wear it. Without `town`, rule 6 leaves all of them exactly as they are.

`any` also takes pieces nobody built; the dry run warns that a later reset of their zone by a world tool may put the game's own pieces back over the baked ones. `convert` is for a world made without Better Continents' maps (below).

Worked example: a test village, baked, undone and unbaked

Real: Better Continents' automated game test, run Q, 8 October 2026, on one machine (BCAutoTest, a test village of 5,617 pieces around (768, -1408) in a world made from VALtima's town file). The dry run of a 128 m bake with `town`:

```
bc_bake area 128 at 768 -1408 town: a dry run, nothing changes. 'bc_bake area 128 at 768 -1408
» town confirm' bakes.
Area: 128 m around 768, -1408 (21 zones).
Found 5,617 pieces. To bake: 5,512 (stone_wall_1x1 2,500, wood_floor_1x1 2,500, wood_floor 207
» and 3 more kinds).
Town pieces: 102 (doors 15, chests 1, fires and torches 31, beds 15, stations 6, comfort 32,
» lights 2). They stay real pieces and become protected parts of the layer.
Not touched: 3 (carts and ships 1; gives items when used 2).
Objects in these zones: 9,794 now, 4,282 after. Layer: +3 KB (2.66 MB, revision 2). Undo file:
» about 629 KB.
```

Read it line by line: the area and the zones it touches; what bakes, by kind; what stays real (with `town`, adopted and protected); what is left alone and why; and the objects before and after, which is what the bake is for. Then Confirm (the log of the same run):

```
bc_bake: bake 1 started, 5,512 pieces. Undo file written.
bc_bake: layer revision 2 sent to 0 players; 0 of 0 ready in 0.1 s.
bc_bake: removing 5,512 pieces, 1,000 a frame.
bc_bake: bake 1 done in 0.5 s: 5,512 pieces baked, 102 left as pieces (town pieces, protected
» parts of the layer). Objects here: 9,794 -> 4,282.
'bc_bake undo' puts them back. The world saves as usual ('save' saves now).
```

What the test counted within 128 m of the village's middle, before and after (the same run):

	Before	After
objects (ZDOs) the world saves and sends	7,984	2,472
enabled renderers in the scene	15,713	4,689
colliders of real pieces	9,930	4,418
median frame at the test's view	14.60 ms	14.96 ms

Every baked piece left the world as an object (7,984 - 5,512 = 2,472, none left over, none removed that was not baked). The frame time did not move, because this view was not limited by objects: the instance counts are the result, which is the rule above. In a dense town it shows in frames too: in the densest block of a 700,000-piece town, 45 fps as real pieces and 81 fps baked (measured on one machine, Better Continents' changelog).

The undo of the same bake, as a dry run, and the unbake of the same circle (real, run Q):

```
bc_bake undo 1: a dry run, nothing changes. 'bc_bake undo 1 confirm' undoes it.
Bake 1 (a circle of 128 m around 768, -1408, by Autotest) is undone: its pieces come back as
» they were, with new object ids (the 30 s of grace before a piece's wear is counted start
» again), and its records leave the layer.
5,512 pieces to make, 0 to remove, 5,614 records out of it.
102 town pieces lose their bake keys.
```

```
bc_bake unbake 128 at 768 -1408: a dry run, nothing changes. 'bc_bake unbake 128 at 768 -1408
» confirm' unbakes.
Area: 128 m around 768, -1408 (21 zones).
Found 5,614 records here: 5,614 baked in game.
To unbake: 5,512 pieces (stone_wall_1x1 2,500, wood_floor_1x1 2,500, wood_floor 207 and 3 more
» kinds). 102 live pieces lose their bake keys and stay where they are.
This makes 5,512 real objects: each is a piece of the world, saved with it and sent to the
» players near it. Objects in these zones: 287 now, 5,799 after. Undo file: about 629 KB.
```

If you can only talk, the bake for the human, as one message:

```
1. In BetterContinents.cfg, under [09 BetterContinents.Export], set  Hud = true  (the game picks it up at once).
2. Load the world and stand in the middle of the base.
3. Press F7, click "Bake buildings", click the radius "64 m".
4. Leave Town, "Also pieces nobody built" and Convert unticked. Press "Bake: dry run".
5. Press "Copy" under "What bc_bake said" and paste me the lines. Do not press Confirm yet.
```

Read the lines with them (is anything they expected to bake under Stay pieces or Not touched?), then: "Press **Confirm: bake 64 m around ...**, wait for the line that starts `bc_bake: bake N done`, and type `save` in the console."

Undo, unbake and the way out

- `bc_bake undo [<n>]` puts back exactly what change `n` (the newest by default) took, from its undo file: the same kind, place, turn, health, creator and contents, with new object ids. It undoes a bake, an unbake (its pieces go and are baked again) or a drop. An older change undoes only while none of its pieces was unbaked since. The newest 20 changes of a world keep their undo files, and every one that is not finished.
- `bc_bake unbake <radius> [at <x> <z>]`, `box ...`: baked records back into real pieces, from any bake. With `all`, a town file's records too; those come back baked the next time a newer town file is loaded, so a lasting change belongs in the file. A decor record (part 2) stays in the layer: there is no object to make of it.
- `bc_bake unbake world [all]`: every record of the world. **This is the way out**: do it, save, and only then remove Better Continents from the world. Over 50,000 objects the dry run warns that the save, the loading and what each player is sent all grow with them.
- `bc_bake drop compiler`, `drop bake <n>`: take records out without making pieces (undo brings them back). `bc_bake drop <world>`, in the main menu: takes the layer out of a world's files, keeping the old file at `BetterContinents/pre-drop/<world>/BetterContinents.pre-drop` in Valheim's save folder; it refuses a layer that holds pieces baked in the game unless `confirm` is given, because those pieces are then gone for good.
- **A crash or a cut save.** Every change writes its undo file before it changes anything and notes each step. At the next load of the world, Better Continents finishes it or takes it back whole, and says which; no piece is ever lost, and none is both a real piece and a record. `bc_bake list` then shows the change as done, undone or abandoned.
- **Where the undo files are**: `BetterContinents/bakes/<world name>-<world id>/` in Valheim's save folder, on the machine that runs the world (`op-<n>.bcj`, and `layer-r<revision>.bcp` for a layer kept before a load). A world renamed later keeps the folder of its old name; since 0.10.5 Better Continents finds it by the world's id and logs `bc_bake: this world was renamed; its undo files are still in <folder>.` (0.10.4 looked only under the new name: trap 66).

> **Human: Unbake before you remove the mod.** A baked piece exists only in the layer. Before Better Continents leaves a world, or the world is opened with an older version or the game alone, run `bc_bake unbake world` (and `all` for a town file's pieces), confirm, and save. Say this whenever a human asks how to uninstall.

On a server

- The server and every player run the same Better Continents version, as for every Better Continents world.
- An admin puts `server` after `bc_bake` (`bc_bake server area 32 town confirm`); the answer comes back to the admin's console and goes to the server's log. On a dedicated server the circle is around the admin's character.
- A change reaches the players before any piece is taken out: the server writes the undo file, changes the layer, sends each player a patch of the zones that changed (or the whole layer to one who missed a revision), waits until each has the new pieces standing or has left (at most 30 s; the output names any it stopped waiting for), and only then removes the real pieces, 1,000 a frame. Nobody standing in a house falls through its floor.
- A player who joins downloads the layer after the settings (a few MB for a whole town file) and keeps a copy in `BetterContinents/cache/`; the next join with the same layer downloads nothing. A copy unused for 30 days is deleted. `Settings Transfer Rate` paces it. A `.bcworld` share file carries the settings, not the layer.

Worlds made without Better Continents' maps

Baking works in any world. A world that is not a Better Continents world (made without the mod, or with it off) becomes one at its first bake or load with `convert`: a **converted world** keeps the game's own ground, biomes, vegetation and locations, and Better Continents carries only its layer. Without `convert` the dry run says:

```
This world is not a Better Continents world. Baking makes it one: its ground, biomes,
» vegetation and locations stay the game's own, and Better Continents keeps the baked pieces
» with the world. From then on every player needs the same Better Continents version as this
» world to join it, and an older Better Continents, or the game without it, opens it without
» the baked pieces. Add 'convert' to go ahead.
```

Every connected player must run the same version, or the command names them and changes nothing. A world whose Better Continents file cannot be read (written by a newer version, or damaged) is never converted. Better Continents does not shape a converted world, so its map settings and Debug Mode do nothing there; `bc_bake drop <world>` makes it the game's own world again.

Part 2: what a town file does in a world

How a file reaches a world

1. **A new world.** `placements.bcp` in the folder that `[00 BetterContinents.Debug] Directory` names, beside the maps, is read when a new world is made From Config or from the folder, checked byte by byte, and kept in the world's settings. The log says `Baked layer: <path>: revision <r>, <N> records in <Z> zones`. A file that cannot be read makes the world without it: `The baked layer <path> cannot be read (<the reason>): the world is made without it. 'bc_bake load' adds a layer to a world later.`
2. **A running world.** `bc_bake load [<file>]` (the Directory's `placements.bcp` by default). Its dry run gives the file's counts, what it replaces, the layer's size after it and what the town's Live pieces will do (`The live pieces would follow: <N> seeded, <N> replaced, <N> removed, ..., <N> kept.`), and changes nothing. Its first line is the line to confirm with, echoing the words as typed: `bc_bake load: a dry run, nothing changes. 'bc_bake load confirm' loads it.` (since 0.10.5; 0.10.4 echoed the file's bare name, which typed back is not found: trap 68). The layer as it was is kept as `layer-r<revision>.bcp` in the undo folder; loading that file goes back.
3. **An export.** `bc_export` writes a world's layer as `placements.bcp` beside the exported maps (chapter 9), so a world made from the export has the same towns. `bc_bake export [<file>]` writes the layer alone, by default to `BetterContinents/<world name>/placements.bcp` in Valheim's save folder.

A preset saved from a running world keeps the town file's records and drops what was baked in the game, since an in-game bake fits only the world it was made in.

What a load keeps and replaces

Part	After `bc_bake load`
the town file's records (source 0)	replaced by the new file's
the records baked in the game	kept, whatever the file holds
the zones' ground, paint and clear masks	the new file's
the town's Live pieces	kept where they stand within 5 cm and 1 degree of their record (same prefab); a moved one is made again; a removed one goes
a container of the town that holds items	never taken away: it stays where it is, loses its protection and becomes an **orphan** (`bc_bake orphans`)

Part	After `bc_bake load`
consumables of the file	placed only in zones the game generates afterwards
vegetation in cells the new file clears	the game's own trees, rocks and bushes in those cells of generated zones are removed; nothing a player built, nothing planted

Live pieces, protection, orphans

A Live record is made a real object when its zone is first generated, on the machine that runs the world: the prefab at the record's pivot, with the bake keys (`bc_bake_src`, `bc_bake_id`, `bc_bake_rev`, and `bc_protect` when the palette entry has the Protect flag) and every tag of its palette entry written into its ZDO. A protected piece cannot be damaged or removed and does not wear. At every world load and after every change, Better Continents reconciles the world's Live pieces with the layer by their (source, id): so **a compiler keeps a piece's id from one file to the next**, or the piece is taken away and made again. `bc_bake orphans` lists the pieces left standing because they held items; `bc_bake orphans destroy` removes the ones that hold nothing, and `items` the rest too.

Consumables and decor

A **consumable** is a prefab with a `Pickable`, `PickableItem`, `ItemDrop` or `Plant` anywhere in it. A town file's records of such a kind, whatever their role, are never drawn, baked or protected: each is placed **once** as an ordinary object of the game when its zone is generated, with no bake keys and no creator, and from then on the game owns it (picked, regrown, rotting, put back by a zone reset as the game's own vegetation is). The log names each kind once:

```
Baked placements: <kind> is a consumable (a player picks it up or harvests it): its records are
» placed once as ordinary objects of the game when a zone generates, never baked, drawn or
» protected.
```

Decor (palette flag 32) is the compiler's opt-in for scenery: a consumable entry of role Static or Copy with the flag is drawn like any other piece of its role, and nothing places its records as objects, so they are never pickable (meads on a shelf, flax in a window box). Use collision 0 (None) or Boxes for it; Prefab would use the pickable's own colliders. Without the flag, nothing changes. On a Live or Seat entry the flag makes Better Continents refuse the whole file. The flag does nothing on a prefab that is not a consumable, and the in-game bake never writes it.

```
Baked placements: <kind> is a consumable that the file marks as decor: its records are drawn as
» scenery, never placed as objects of the game and never pickable.
```

Ground, paint and clear masks

A zone of the file can carry three sections, applied on every machine as the world generates:

- **Clear mask**: 16 x 16 cells of 4 m in which the game's own vegetation (trees, rocks, bushes) does not grow. Zone flag 16 keeps it out of the whole zone. Streets and squares need it; otherwise trees grow through the floors.
- **Ground**: 65 x 65 heights 1 m apart, each with a weight: 255 is this ground, 0 the world's, between them a blend. It replaces the world's height in the game's own height function, so the terrain, the minimap, the server's placement of trees and locations and every player's ground agree. A town's core is 255; a band of a few metres blends it into the world.
- **Paint**: 64 x 64 cells of 1 m: dirt, paved or cultivated, painted into the terrain's own mask (grass stops where it is painted). Players draw it; the server has no use for it.

Part 3: writing a town compiler

A compiler turns a design (a list of buildings, a blueprint, a town plan) into `placements.bcp`. Better Continents reads every byte before it uses a file and stops at the first problem, so a compiler either writes a valid file or none. Appendix E is the whole layout and every check; this part is what each field means to the game, and how to write and test one.

The file in one table

Part	Holds
header, 32 bytes	`BCPL`, format 1, header flags (1 some zones have ground, 2 paint, 4 clear masks), revision, palette, zone and record counts
producer	a short ASCII text naming the tool
palette	one entry per kind, world-wide: 1 to 8 candidate prefabs with anchors, a role, a collision kind, a layer, flags, a tint, tags
zone index	one 28-byte row per 64 m zone, sorted by z then x: where its block is, its record count, its Live count, its flags, its height range
zone blocks	one raw Deflate stream per zone: the records, column by column, then the clear mask, ground, paint and extras
trailer	the CRC-32 (zlib's) of everything before it

Palette entries

Candidates and anchors. An entry names up to 8 prefabs; each player's game draws the first it has, so a modded piece can name a vanilla stand-in after it (VALtima's file has 54 such entries). Each candidate has an **anchor**, in millimetres in that prefab's own frame: the point the record stands for. A compiler that holds pivots writes anchors of 0. When the stand-in's pivot sits elsewhere than the first prefab's, its anchor says where, so both stand in the same place.

Roles.

Role	What Better Continents does with its records
0 Static	every player's game draws them and gives them a collider as the entry's collision says; never an object
1 Live	the server makes each a real object when its zone is first generated (part 2), with the record's id and the entry's tags; Better Continents draws nothing for it, and its collision is ignored (the real piece has its own)
2 Copy	drawn, unlit, and solid as the collision says; within Light Distance, the Light Count nearest become lit copies of the piece (the game's own prefab, without its fuel, warmth and area)
3 Seat	drawn and solid as a Static; within Seat Distance, a local chair a player can sit on

Collision.

Kind	The collider
0 None	none: decor, small plants, things to walk through
1 Boxes	the entry's boxes (centre and size in mm), in the frame of its **first** candidate, positioned by that candidate's anchor. Every player collides with these boxes, also one whose game draws a later candidate: the safe choice when a player may lack a mod
2 Trunk	a 0.6 x 3 x 0.6 m box standing on the point: a tree or a post
3 Prefab	the colliders of the prefab the player's game draws, welded per zone; with no candidate there, no collider

Layer. The Unity layer of the instanced drawing and of the colliders Better Continents builds; 0 means the game's `piece` layer. Nothing else reads it (the culling, the batching, the shadow proxies and the lights ignore it). **Write 0.**

Flags.

Flag	Meaning
1 UniformScale	read and kept; Better Continents gives it no meaning of its own
2 NoShadows	the entry's pieces cast no shadow
4 Protect	a Live entry's pieces are protected (no damage, no removal, no wear)
8 LOD0Only	drawn with full detail at every distance, never the simpler model
16 NoCopyLight	a Copy's lit copy has no light (flames only)
32 Decor	a consumable entry of role Static or Copy is scenery (part 2); refused on Live and Seat

Tint. A name, a colour (three values, 1 = unchanged, up to 65.535) and a material filter: the colour is set on every material of the piece whose name contains the filter (or the part of the filter after its last _), ignoring case. A tint without a filter changes nothing. The player's `Tints` setting can replace a tint's colour by its name. VALtima's clay walls are one prefab tinted plaster, brick, sandstone or marble with the filter `clay`.

Tags. Values every Live piece of the entry gets in its ZDO when it is made: bool (stored as the game stores a bool), int, float or string, under the game's own key hash. They are how a mod finds its pieces (VALtima's are `VALtima_TownPiece`, a bool, and `VALtima_Kind`, a string). The tags `MatVar<i>` choose a piece's material variant.

Records

A record is an entry index and a point, in its zone's block:

```
zone        zx = floor((x + 32) / 64), the same for zz; zone (zx, zz) is centred on (zx * 64, zz * 64)
x, z        u16 = round((x - (zx * 64 - 32)) * 1024), 0..65535: 1/1024 m from the zone's south-west corner
y           i32 = round(y * 1000): the game's world height in mm (the water is at 30 m)
yaw         u16 = round(degrees / 360 * 65536) mod 65536
rotation    (record flag 1) only for a piece that leans: smallest three of the quaternion, x 32767
scale       (record flag 2) u16[3] = scale x 1000, 0.001 to 65.535
id          (record flag 4) u32, on every Live record, unique in the file, kept from one file to the next
pivot       = point - rotation x (scale * anchor)
```

Rounding is to the even number on a tie (.NET's `Math.Round`, Python's `round`). Write `y` as the game's world height: a tool that thinks in metres above the water adds 30.

Rules a compiler keeps

1. Little-endian; a string is a u8 length and that many ASCII bytes (255 at most).
2. Zones -1024 to 1023 on each axis; the index sorted by zz, then zx, each zone once.
3. Every Live record has an id; no (source, id) twice in the file; the index's Live count of each zone is its Live records.
4. The header's record count is the sum of the zones' counts; the zone's block count is its index count.
5. A record flag outside 1, 2, 4 is Better Continents' own (8, 16 need a registry): a compiler writes none.
6. Decor (32) only on Static or Copy; no palette flag above 32.
7. Each block inflated is at most 16 MB; the file at most 256 MB; at most 65,535 palette entries and 65,535 Live records a zone.
8. The y range of each zone row covers its records and the height of their pieces (it bounds the culling).
9. Canonical order (what Better Continents writes, and what makes two files comparable): header, producer, palette, index, blocks in index order with no gaps, CRC; records in a block sorted by palette index, then z, x, y, yaw.
10. Set the header flags 1, 2, 4 from the zones' sections, and the extras count in every block (0 when none).

The sample writer: `bcp.py`

The toolkit's `bake/bcp.py` (standard Python, no packages) writes format 1 and checks a file the way Better Continents does:

```
python3 bcp.py check <file.bcp> [--zone ZX ZZ] [--palette]
python3 bcp.py demo <out.bcp> [--at X Z] [--y Y]
```

`check` stops at the first problem with Better Continents' own words (`palette entry 0: it is marked decor (flag 32), which only a Static or Copy entry can be, and its role is Seat`), and otherwise prints a summary; `--zone` lists a zone's records decoded to world places. `demo` writes the sample town of the worked example below. As a library:

```
import bcp

layer = bcp.Layer(producer='my compiler 1.0')
# Static, collision Prefab (the defaults)
floor = layer.entry('stone_floor_2x2', height=1.0)
door = layer.entry('wood_door', role=bcp.LIVE, flags=bcp.PROTECT,
                   tags=[('mytown_piece', bcp.BOOL, True), ('mytown_kind', bcp.STRING, 'door')])
torch = layer.entry('piece_groundtorch_wood', role=bcp.COPY)
# scenery, never picked
flax = layer.entry('Pickable_Flax_Wild', collision=bcp.NONE, flags=bcp.DECOR)

# world x, y, z of the anchor (here the pivot); yaw in degrees
layer.add(floor, 481.0, 32.0, 481.0)
# a Live record needs its lasting id
layer.add(door, 484.0, 32.0, 480.0, yaw=90, id=1001)
layer.add(torch, 490.0, 32.0, 490.0)
layer.add(flax, 470.5, 32.0, 476.0, yaw=53)
# no trees through the square
for x in range(468, 496, 4):
    for z in range(468, 496, 4):
        layer.clear(x + 0.5, z + 0.5)
layer.paint(481.5, 481.5, bcp.PAINT_PAVED)
# makes every check of the mod first, and writes nothing if one fails
layer.write('placements.bcp')
```

`layer.ground(zx, zz, base, heights_cm, weights)` sets a zone's ground (65 x 65 each, row 0 south, vertex `i` at `(x0 + i % 65, z0 + i // 65)`); write the same height and weight on the edge two zones share, since each point takes the ground of the zone it is in. `entry(..., boxes=[((cx, cy, cz), (sx, sy, sz))], collision=bcp.BOXES)` gives boxes in metres; `tint=('plaster', (1.65, 1.58, 1.48), 'clay')` a tint; `anchors=[(ax, ay, az), ...]` one anchor per candidate.

Worked example: the sample town, written and checked

Real: run on Wubarrk's machine, 9 October 2026. `python3 bcp.py demo placements.bcp` writes a paved square 24 m a side, centred on (480, 480), the corner of four zones: 144 floor tiles, a low wall on three sides, two Live chests with ids and tags, four torches (Copy), two benches (Seat), a stone at twice its size (the scale run), one that leans 10 degrees (the rotation run), six berry bushes (a consumable: placed once as real bushes) and eight decor flax, a clear mask over the square, paint (paved, a dirt band) and ground at the floor's height blending into the world's over 8 m. What it prints:

```
wrote placements.bcp: 1309 bytes, 8 palette entries, 204 records in 4 zones
placements.bcp: readable (1309 bytes, CRC good)
header: format 1, flags 7, revision 1, producer 'bcp.py sample town (AI skill)'
palette: 8 entries; roles {'Copy': 1, 'Live': 1, 'Seat': 1, 'Static': 5}; collision {'None': 1,
» 'Prefab': 7}; layers {0: 8}
  palette flags {0: 6, 32: 1, 4: 1}; tinted 0; more than one candidate 0; tags
» {'sampletown_kind': 1, 'sampletown_piece': 1}
  decor (flag 32): 1 entries, 8 records {'Pickable_Flax_Wild': 8}
registry: none (a compiler's file)
```

```
zones: 4, 4 with records; zone flags {7: 4}; sections {'clear mask': 4, 'ground': 4, 'paint':
» 4}; extras 0
records: 204; by role {'Static': 196, 'Copy': 4, 'Seat': 2, 'Live': 2}; record flags {0: 200,
» 1: 1, 2: 1, 4: 2}; Live ids 2
blocks: 852 B compressed, 69921 B raw; records outside their zone's y bounds: 0
```

The same file through two other readers, the same day: the independent reader of Better Continents' build work (`bcpread.py`) found 0 errors, and Better Continents' own `BakedLayer.Read` (the mod's code of 0.10.4, commit ae63889, called from a scratch program) read it: `revision 1, 204 records in 4 zones, palette 8`, roles Static 196, Copy 4, Seat 2, Live 2, the leaning stone's quaternion as written (0.0842, 0.2578, -0.0226, 0.9623). Three broken copies were refused by both `bcp.py check` and the mod with the same words: a flipped byte (`the CRC differs: ...`), decor on a Seat entry, and a Live record without an id (`zone 0,0, record 0: a Live record has no id`). `bcp.py check` also reads VALtima's real 702,700-record town file with the values its spec lists (128 palette entries, 1,365 zones, 2,737,580 bytes of blocks).

Testing a file

> **Check:** before a human sees a town file: `python3 bcp.py check placements.bcp` prints `readable`, the role counts are the ones the design has, every Live record has an id (the `Live ids` count equals the Live records), the decor count is what you meant, and `records outside their zone's y bounds` is 0.

Then in the game, at Game level, without changing anything:

1. Put the file anywhere and type `bc_bake load <full path of the file>` in a test world: the dry run reads the file (or says `bc_bake: placements.bcp is not a layer this version can read (<the reason>). Nothing is changed.`) and gives its counts.
2. For a real test, a **new world** with the file in its Directory: the log's `Baked layer: ...` line, then `bc_bake info` (the form of its lines, from the code that prints them):

   ```
   bc_bake info: layer revision <r>, <N> records in <Z> zones, <size>, made by <producer>.
   Id <the layer's id>.
   Records by role: Static <n>, Live <n>, Copy <n>, Seat <n>. By source: the compiler's file <N>.
   Palette <n> entries. Ground in <n> zones, paint in <n>, clear masks in <n>, no vegetation at all in <n>.
   No operation in the registry.
   Live pieces seeded in this world: <n>.
   Pieces the layer names that this game lacks: none.
   ```

   It goes on with the consumable and decor lines. `Pieces the layer names that this game lacks` lists any prefab name that is wrong or belongs to a mod that is not installed; such an entry is not drawn (its boxes or trunk still collide), and the log says `none of <names> is in this game: those baked pieces are not drawn`.
3. `bc_bake check all` reads every zone and lists duplicates (the same kind within a centimetre and a degree), records of a town file and a bake in one place, real pieces standing on, under or close to a record, and Live records past 16.35 km in a world without wide sectors (chapter 10).
4. `bc_bake export` writes back what Better Continents holds, for a comparison with the file (by content: the compressed bytes of a block may differ between runtimes).

If you can only talk: send the human the commands of steps 1 and 2 with the file's full path filled in, and ask for the answer pane (Copy) or the `Baked layer` line of the log.

Part 4: tuning and checking

The settings

`[10 BetterContinents.BakedPlacements]` is each player's own: it changes how baked pieces are drawn on that player's screen, never the world, and every value applies at once (appendix B has the full descriptions).

Setting	Default	Range	What it does
`Draw Distance`	4	1 to 8 zones	zones around the camera that are drawn: 4 is a block of 9 x 9
`Draw Scale`	1.5	0.5 to 4	how far a piece is drawn, as a multiple of where the game culls the same piece
`Detail Scale`	1	0.5 to 4	how far a piece keeps its full detail, as a multiple of where the game switches it to its simpler model
`Shadows`	All	All, Near, Off	which baked pieces cast shadows (below); every one receives them
`Light Distance`	90	0 to 200 m	how far from you a baked light can burn; 0: none
`Seat Distance`	12	0 to 40 m	how far from you a baked seat can be sat on; 0: none
`Light Count`	24	0 to 128	how many baked lights burn at once, the nearest within Light Distance; the rest are drawn unlit
`Light Shadows`	4	0 to 32	how many of the burning lights cast shadows, the nearest
`Tints`	(empty)	`name=r,g,b; ...`	replaces a tint's colour by its name
`Diagnostics`	false		logs the materials of each baked kind once (for bug reports)

```
[10 BetterContinents.BakedPlacements]
# the defaults; lower Light Count and Light Shadows first when a lit town is slow
Draw Distance = 4
Draw Scale = 1.5
Detail Scale = 1
Shadows = All
Light Distance = 90
Seat Distance = 12
Light Count = 24
Light Shadows = 4
Tints =
Diagnostics = false
```

When a big town is slow on a player's machine, lower in this order: `Light Count` and `Light Shadows` (each burning light costs frame time; the setting's own text says a street of them can halve the frame rate), then `Draw Scale` to 1, then `Shadows` to Near, then `Draw Distance`. `bc_bake hide` stops drawing the layer on that screen (the pieces stay solid), for a comparison; `bc_bake show` draws it again.

Shadows, and shadow proxies

Shadows were the cost of a baked town: drawing what is in view is cheap, but every piece that casts must be drawn again into each of the sun's shadow cascades and into each shadowed light. Since 0.10.4, Better Continents draws them through **shadow proxies**:

- For each zone near the camera it merges the casting parts of the zone's pieces into a few **shadow-only meshes**: from the pieces' full-detail models for the camera's zone and its neighbours, from their last, simplest models for the other zones within the shadow distance (plus 96 m). The merging runs on **worker threads**; the main thread only hands each finished mesh to the graphics card, a mesh at a time, spread over frames. Vertices are kept compact (12 bytes each).

- The instanced pieces of a kind a proxy covers then cast nothing themselves. What a proxy cannot hold still casts by instancing, as before: parts drawn with cutout or blended materials (thatch roofs, beds, crystal walls) and parts of other shaders (rugs, banners, shrubs).
- `Shadows = All` uses both kinds of proxy. `Near` uses only the near ones: farther out, pieces in full detail cast by instancing and the simpler models cast nothing. `Off`: no proxies, and no shadows from baked pieces. Palette flag 2 keeps an entry out of every shadow.
- A tool maker's consequence: a kind made of cutout or blended materials keeps casting per instance; many of them in one view cost what they always cost. Flag 2 (NoShadows) on small, many, low pieces is the lever a compiler has.

What they did, measured on one machine (an RTX 4070 laptop GPU, 1600 x 900, VALtima's town district in Britain at its busiest view, fresh worlds, 9 October 2026): the frame went from 26.41 ms (38 fps) without proxies to 15.85 ms (63 fps) with them, and the towns' own cost at that view to 3.02 ms a frame. The proxies held 351 MB of graphics memory (169 MB near, 183 MB far), and the worst single mesh handed over cost the main thread 1.0 ms. The shadows matched pictures taken without proxies.

`bc_bake stats` shows them on a player's game: `Shadow proxies: X of Y ring zones have one (...)`, the memory by kind, the cost on the workers and on the main thread, and how many parts were merged, left to instancing as cutout or blended, of other shaders, or unreadable. Two switches in the environment exist for testing only, not as settings: `BC_SHADOW_PROXIES=0` turns the proxies off, and `BC_SHADOW_PROXY_FORMAT=plain` makes their vertices three floats each, for an A/B comparison. Players never need them.

What to read

Command	Where it runs	What it tells you
`bc_bake info`	anywhere	the layer: revision, records by role and by source, sections, operations, Live pieces seeded, missing prefabs, consumable and decor kinds
`bc_bake list`	where the world runs	every change: number, kind, time, who, area, records, state (`done`, `undone`, `abandoned`, `not finished`), `undo file kept`
`bc_bake check [all | <zx> <zz>]`	where the world runs	damage and overlaps (Testing a file, above); `No problem found.` when clean
`bc_bake stats`	a player's game	what is drawn in view: zones built and held, kinds, consumable and decor counts, lit copies, instances and batches by kind, shadow proxies
`bc_bake orphans`	where the world runs	the town's Live pieces left standing because they held items

A good result, in the measure's terms: the dry run's `Objects in these zones: A now, B after` and, after Confirm, `Objects here: A -> B` agree; `bc_bake list` says `done`; `bc_bake check` finds nothing; `info` lacks no prefab.

Recipe: a town file, from a script to a world

1. **Design.** A list of buildings with their pieces as prefab names, places in world metres (y as world height) and turns. Decide each kind's role: Static for walls, floors, roofs; Live for doors, chests and stations a player uses; Copy for torches and braziers; Seat for chairs and benches. Bushes and herbs stay consumables unless they are scenery (Decor).
2. **Write.** A compiler on `bcp.py` (or its own writer that keeps appendix E): one palette entry per (prefab, role, collision, flags, tint, tags); stable ids for Live records; clear masks under every street and floor; ground for the town's core with a blend band; paint for streets.

3. **Check.** `python3 bcp.py check placements.bcp`: `readable`, the counts as designed, 0 records outside the y bounds.
4. **Stage.** Copy the file into the Directory beside the maps. (A folder that holds only the file makes a Better Continents world at the config's settings, whose terrain is close to the game's but not the same. For the game's own terrain, make the world with the New World preset `Disabled` and bring the file in with `bc_bake load <file> convert`.)
5. **Prove.** The human makes a new world; the log has `Baked layer: <path>: revision 1, <N> records in <Z> zones`; `bc_bake info` has no missing prefab; walk the town (doors open, chests hold, torches burn near, benches seat, bushes pick, decor does not, no tree through a floor); `bc_bake check all` finds nothing.
6. **Change it later.** A newer file with the same ids into the running world: `bc_bake load` dry run, read what the Live pieces will do (kept, replaced, removed, orphaned), then confirm. Players keep what is in the chests.

Traps

Each is in chapter 7 with its sample and fix (traps 57 to 68): a world with a layer opened by an older version or the game alone; a town file refused; a whole file refused for decor on a Live or Seat entry; bushes in a town file that are real; pieces 30 m under the ground (height above the water written as y); a Live piece made again at every load (its id changed); a chest left as an orphan; a missing mod's pieces; removing the mod without unbaking; a renamed world's undo files; a bake refused for too many sets of values; and the console's `confirm` after a dry run of `load` without a path.

13 Water, lakes and clearing

New in Better Continents 0.10.5 (October 2026), three things that change what a map maker draws:

1. **Better Continents' own water.** In a world it shapes whose sea is not the game's, the sea is drawn by Better Continents' shader with the game's look, from the **world's** rules: waves to the world's edge, calm only in the world's Deep North, red only where the world's hot sea is. Boats and swimmers feel exactly what is drawn. No world data, so it reaches existing worlds in place (part 1).
2. **Lakes** (experimental): a per-world list of lakes, each with its own water level at any height, filled with the game's own water (part 2).
3. **Clearing** (experimental): a bitmask over the ground that keeps vegetation off marked pixels, such as roads and lake beds (part 3).

Lakes and clearing come from **Tarn**, the mod by Raven Iron (Wubarrk and NomadicWar), contributed to Better Continents under the LGPL-2.1. Their two file formats are Tarn's, byte for byte. Part 4 is `bc genloc`, a console command of the same release. The players' version of this chapter is `BetterContinents-Water-Guide.pdf` (*Water and lakes, the easy guide*).

> **Human: Lakes and clearing are experimental for now.** Say so plainly whenever you suggest them, and have the human **try them on a copy of a world first**: the human copies the world's folder (`worlds_local/<name>/`, appendix B) before any lake or mask goes into it. A world that carries either needs Better Continents 0.10.5 or later on every machine, and an older version opens it with vanilla terrain. As with baking, never write `confirm` on a human's world unless they asked for that change.

At your level

Level	The sea	Lakes and clearing
Talk	you tell the human where to paint Deep North and heat over the sea, and ask for `bc_water`'s answer	you give the lake lines and the mask code, and the steps to put the files in the maps folder
Files	you also read `LogOutput.log` (its `Water:` lines)	you write `lakes.txt` and `clearmask.bin` and check them with `lakekit.py`
Commands	as Files; you compute the calm and hot shares from your maps to compare	you run `lakekit.py` (and its flood over your heights) before the human sees the files
Game	`bc_water`, and `bc_water probe` with Debug Mode	a new world from the folder; `bc_lake info`, `list`, `mask at`; walk to a lake

Part 1: The sea

What the game does, and what Better Continents does instead

The game's water shader draws its calm Deep North sea and its red Ashlands sea from two fixed circles of a 10 km world (chapter 3): calm past 12,000 m (plus 100 m x `WorldAngle`) from (0, -4,000), red past 11,987.5 m (plus 100 m x `WorldAngle`) from (0, +4,000). On a bigger world nearly all the sea lies past them, so the game drew it calm and, without Better Continents' colour fix (0.10.3), red.

Better Continents 0.10.5 builds a **sea field** on every machine when a world loads, on worker threads (760 ms for a world of World Size 64,500 on 31 threads, real, one machine):

extent	the square +-R, R = World Size + Edge Size (Expand World Size's radius when it is larger), 16 to 65,536 m
grid	N x N texels, N at most 4,096, of max(8 m, 2R / 4,096)

calm (G)	the distance into the world's Deep North, `WorldGenerator.IsDeepnorth(x, z)` at each texel centre, 0 to 255 m
hot (R)	the signed distance to the border of the world's hot sea, `GetAshlandsOceanGradient(x, z) > 0`, +-127 m
waves	W = 1 - clamp01(calm / 200): full waves outside the Deep North, gone 200 m into it. The five big waves are multiplied by W, the five small ones by max(W, 0.1): a calm sea keeps a ripple
red	m = clamp01(0.5 + hot / 25): the game's 25 m edge, centred on the border, broken up by the game's foam texture
far sea	with Soft Far Sea, a second grid of 64 m holds clamp01((30 - ground) / 10) for the far plane's depth colour
where waves are geometry	in the zone water loaded around the player, out to about 130 m at the game's usual settings (the game test read 128 m); beyond it the far plane is flat with the game's moving ripple texture, as in the game

Those two functions are the ones the weather, a swimmer's heat and a ship's burn already follow, with every patch on them: Better Continents' biome map and heat map, its layout to the world's size, Expand World Size and Expand World Data.

Felt equals drawn. Two patches, on every machine including a dedicated server, make `WorldGenerator.DeepNorthWaveFade` answer 1 - W from the field and `WaterVolume.CalcWave` add the same waves on the shader's clock. Everything that floats asks those. In open sea the wave formula is the game's to the last digit; only the clock is the drawn one. Measured with the in-game probe (real, VALtima's 130 km world, at 19.5, 35.75, 50.05 and 61.75 km from the middle and in thunderstorms): the largest difference between drawn and felt surface over 81 places was 4 mm.

Which worlds get it

World	Sea
a Better Continents world (`ShapesWorld`) with a biome map, or a heat map with `Heatmap Scale` above 0, or laid out to its own size (settings version 12, World Size or Edge Size not 10,000 / 500), or with Expand World Size setting its size	a **candidate**: the field is built
a candidate whose field comes out exactly the game's two circles, texel by texel	the game's sea (nothing changes)
any other candidate	**Better Continents' sea**, drawn and felt
a Better Continents world at the game's size without maps; a version 11 world of another size without maps (its rings are not scaled); a GameTerrain world (made by `convert`); a vanilla world; the Disabled preset	the game's water, exactly

A client draws the game's water instead (and says why in the log and in `bc_water`) when: `Own Water` is off; the shader is not supported (macOS: the bundle has Direct3D 11, Vulkan and OpenGL Core programs, no Metal); the device has no RG16 texture format; the bundle does not load; or another mod has replaced the game's water shader on the sea. The felt waves stay the world's on that machine. With Own Water off on a world Better Continents shapes, the game's shader gets the zone-by-zone red colours of 0.10.3 and no bubble patches.

Settings, `[11 BetterContinents.Water]`, both live and each player's own: **Own Water** (true) and **Soft Far Sea** (true). Neither is world data; the field is never saved or sent.

Authoring for it

> **Rule: The calm sea is your biome map's Deep North over water.** Paint Deep North on the sea only where the sea must be calm, and expect the waves to fade over the first 200 m inside it. A map with no Deep North has no calm sea at all: VALtima's 130 km world reports 0.0 % calm, waves everywhere.

> **Rule: The red sea is your heat map's heat over water** (heat above 0 with `Heatmap Scale` above 0), per pixel near and far, with a 25 m soft edge and no falloff past your heat map's pixels. Fade the heat out over the sea where the red should end; keep it 0 everywhere else (trap 4). By the code a lake inside the hot sea is red and burns too, though no game test has seen that yet (part 2).

Without either map, a world of another size made since 0.10 has the game's two rings scaled to its size: a world of World Size 24,500 without maps reports 19.0 % calm and 19.0 % hot (real). The Deep North look of the **ground** (snowy mountains, soil-less farms far from the north) is the terrain shader's fixed circle and does not move (trap 52).

Check it

`bc_water` (any player, no cheats) prints what the world's sea is. Real, VALtima's world (World Size 64,500, Edge Size 500, with a biome map and a heat map):

```
Water: BC's sea is a candidate for this world (the world has a biome map, a heat map, a world
» laid out to its own size).
Water: the sea field: on.
Water: 4096 x 4096 texels of 31.738 m over +-65000 m, built in 760 ms on 31 thread(s).
Water: 0.0 % of the world is Deep North (calm), 0.0 % is hot sea (red), its Deep North is not
» the game's circle, its hot sea is not the game's ring.
Water: boats, swimmers and fish feel the world's waves (the field's W, the drawn clock).
Water: the far sea's depth grid: 2034 x 2034 texels of 64 m.
Water: drawn with BetterContinents/Water: 0 zone water surface(s) and 26 far sea (LOD 250).
» Soft Far Sea is on.
Water: 0 material value(s) did not survive a shader change and were put back (none: Unity keeps
» them by name); the zone depth arrays, which no shader declares, were put back 1 time(s).
```

> **Check:** the two shares are of the whole square (land and sea), so compare them with the share of your biome map's pixels that are Deep North and your heat map's pixels above 0, over the same square: they agree to about a texel's rounding. A share far from yours means the world was made from other maps, or another mod moves the Deep North or the heat. A line `Water: another mod, <name>, patches <method>: the drawn waves may not match what boats feel.` names a mod to look at.

With Debug Mode (`Debug Mode = true` in `[00 BetterContinents.Debug]`): `bc_water force on|off` draws Better Continents' water where the world's sea is the game's (to compare the two shaders at the same places; not saved; the felt waves do not change), and `bc_water probe` prints `Water probe: n=<used> median=<m> p99=<m> max=<m> (drawn minus felt, metres); camera <x> <z>` for 81 places in front of the camera. A median and a p99 of a few millimetres is what the tests measured; centimetres or more mean a foreign patch on the wave functions.

Part 2: Lakes (experimental)

What Better Continents does with a lake

A lake is one line of the world's list (below). On every machine that draws (never on a dedicated server), when the player comes within **Lake Build Range** (400 m) of the lake's box, Better Continents works out its shape on a worker, exactly as Tarn's builder did:

1. **Sample** the **generated** ground (`WorldGenerator.GetHeight`, never the edited terrain) on a 4 m grid over the box and 12 m around it.
2. **Start** at the cell nearest the lake's point; if that cell is not under the level, at the lowest cell within 3 cells (12 m); if that is not under the level either, the lake is not built (`Lakes: lake '<name>': its point (<x>, <z>) is dry: ground <g> m, level <l> m`).
3. **Flood** every cell strictly under the level joined to the start by a side (4-connected). A flood that reaches the grid's edge only warns (`... its water reaches the edge of its box - the basin is not closed there, so the shore is cut off at the box`).
4. **Grow** the wet cells by one cell (8 neighbours): that is the water's outline, cut by the ground where it rises.
5. **Build** the game's own 64 m zone-water tiles (on the zone grid) that the outline touches, nearest first, at most **Lake Build Budget** (2 ms) of main-thread time a frame, and trigger boxes that make the water swimmable; take them down 150 m beyond the range.

The water is the game's: swimming, the Wet status, floating, boats, the underwater view. Its colour, foam and wave size follow its depth: each tile's corner depths are max(0, level - ground) x **Lake Depth Scale** / 10, clamped to 1 (full colour and full waves at 10 m of scaled depth; at the default 0.5, at 20 m of real depth). The felt depth equals the drawn depth (the boxes are cut at tile borders and take the tile's own depths), and the waves follow the sea field's rules at the lake's position: **calm in the world's Deep North, red and hot in the world's hot sea** (that is how the code works; no game test has seen a lake in either yet). Measured over a lake (real): drawn minus felt median 0.000 m, p99 0.015 m, max 0.025 m at wind 0.6. Lakes are painted on the minimap and the large map (map height under the sea, no forest), and a quiet copy of the game's ocean sound plays near them.

The lake file

`lakes.txt` in the maps folder (the `Directory` setting); `tarn-lakes.txt` is read when `lakes.txt` is missing. A new world made From Config, by `bc_import` or from an export folder reads it once and keeps the lakes in its settings (DataKey 72, a blob of the depth scale and the lakes). One lake a line, nine fields separated by `;`:

```
# Tarn lakes: name;x;z;level;minX;minZ;maxX;maxZ;depth (world metres, absolute heights; x, z =
» a point in the water)
Vikedal1;-4210.5;3190.25;92;-4420;3050;-4010;3330;11.4
```

Field	Meaning
`name`	no `;`, no line break, at most 64 characters (longer is cut, empty becomes `lake<k>`); `remove <name>` removes every lake of that name
`x`, `z`	a point in the water, world metres; the flood starts there. Put it at the deepest place
`level`	the water's surface as an **absolute** world height: the sea is at 30, so `92` is 62 m above the sea
`minX`, `minZ`, `maxX`, `maxZ`	the box the water may fill; the builder floods 12 m beyond it
`depth`	metres of water at the deepest; kept and written back, not used to build (the depth comes from the ground)

The reader (LakeSet.ParseLines): invariant numbers (a dot; no thousands separator; .NET's float syntax); # lines and blank lines skipped; fields past the ninth ignored; at most 5,000 lakes. Each problem is logged as `Lakes: <path>: line <n>: ...`:

Problem	Effect
fewer than 9 fields	the line is left out
a field that is not a number, or not finite	the line is left out
an empty box (`maxX <= minX` or `maxZ <= minZ`)	kept, with a warning: the flood has nowhere to go
the point outside its box	kept, with a warning
a level under 30.5 m	kept, with a warning: the water sits in the sea's waves

Make lakes from your heights

The level must close the basin on the **generated** ground. The generated ground follows your heightmap (chapter 2's height maths, in the game's metres with the sea at 30) but is not identical to it wherever your settings let the game's own detail through, so leave a margin and prove the lake in the game. `lakekit.py` (in the toolkit zip, `toolkit/lakes/`) transcribes Better Continents' own `bc_lake at`: Tarn's priority-flood spill search on a 2 m grid (the height at which water around the point first reaches the edge of a +-radius square), the level at the spill less a margin, the box of the wet cells, then the builder's 4 m flood, lowering the level 0.2 m at a time (ten times at most) until that flood is closed:

```
import numpy as np
import lakekit                                  # toolkit/lakes/lakekit.py

grid = np.load('heights.npy')                   # world heights (the sea at 30), row 0 north,
» first and last pixels on the edges
height = lakekit.height_from_grid(grid.tolist(), half_width=10500.0)   # World Size + Edge Size
lakes = []
for x, z in [(-4210.5, 3190.25), (1700.0, 2800.0)]:      # the deepest points of your basins
    lake, words = lakekit.make_lake(height, x, z, 'lake%d' % (len(lakes) + 1), radius=300.0,
» margin=0.5)
    print(words)                                # the level, the spill, the depth and the
» area, or why there is no lake
    if lake:
        lakes.append(lake)
lakekit.write_lakes('lakes.txt', lakes)
```

`radius` must reach past the basin's rim (8 to 1,000 m; a square that ends inside the basin spills at its own edge), and `margin` is how far under the spill the water stands (Better Continents' `here` takes 0.3 m; a bigger margin is safer when the generated ground may differ from your array). Refusals: the hollow drains to the sea (its level would be within 0.2 m of 30), the point is not in a hollow, or the water still leaks after ten lowerings.

> **Check:** `python3 lakekit.py check-lakes lakes.txt --heights heights.npy --half-width 10500` prints every warning Better Continents' reader would log, with the line number, and floods each lake on your heights: it must end with `N of N lake(s) flood cleanly`. `lakekit.py selftest` proves the transcriptions on made-up ground (real: all passed). In the game: the log's `Lakes: <path>: <N> lakes` at creation, then `Lakes: lake '<name>' built at (<x>, <z>): level <l>, <A> m2 of water, <T> tile(s) (...), <B> box(es)` as you come near, and no `is dry` or `not closed` line; `bc_lake info` and `bc_lake list`.

Lakes do not clear their own bed: trees, rocks and bushes in the basin stand in the water, and grass shows through it unless the paint map paints the bed as dirt. Mask the beds (part 3) and paint them.

The console: bc_lake

`bc_lake` needs no cheats and no Debug Mode. The words that change the world run on the machine that runs it (single player, the host); a client is told `this runs where the world is. Ask the server: bc_lake server <the same words> (admins only).`, and `bc_lake server <words>` runs them on a dedicated server for an admin on its `adminlist.txt`. Words that replace or remove in bulk are dry runs until `confirm`; `here`, `at` and `depth` act at once. While the Tarn mod is installed every changing word refuses. The host's cheat tree has the same as `bc lake ...`.

Words	Who	Does
`help`, `info`	anyone	the usage; the lakes, depth scale, nearest three, Deep North / hot-sea lakes, the mask, the engine's state
`list [n]`	anyone (in a world)	the n nearest lakes (default 10, at most 1,000): name, point, level, distance, built state
`mask at <x> <z>`	anyone (in a world)	whether that point is kept clear, and its pixel
`here [radius] [margin]`, `at <x> <z> [radius] [margin]`	the world's machine	the spill search at the caller (or the point): radius 8 to 1,000 m (120), margin 0 to 20 m (0.3); adds `lake<k>`, sends it to the players
`remove <name|all>`	the world's machine	`all` needs `confirm`; the old list goes to `<save>/BetterContinents/<world>/lakes-before-<time>.txt` first
`depth [scale]`	the world's machine	the world's Lake Depth Scale, 0 to 3, live for everybody
`load [folder]`	the world's machine	the maps folder's (or the folder's) `lakes.txt` and `clearmask.bin` replace the world's; `confirm`; `convert` for a vanilla world
`import [path]`	the world's machine	Tarn's `BepInEx/config/Tarn/<world>.txt` and `.clear` (or the folder's); `confirm`; `convert`
`mask load <file>`	the world's machine	the mask alone; `confirm`; `convert`
`export [folder]`, `mask export <file>`	where exports are allowed	writes the files in Tarn's formats; never over a file without `confirm`, never inside the world saves
`regen`	the world's machine	counts the generated zones the lake boxes and the mask touch; `confirm` regenerates every unprotected generated zone, as `bc regen`

A changed lake list is saved with the world at its next save and sent to every player in it at once; the map is drawn again when the painted lakes change. `convert` makes a vanilla world a GameTerrain world (the game's own terrain, Better Continents keeping only its data), and is refused while a connected player has another Better Continents version.

Tarn's own files, and the Tarn mod

- **Import once.** On the machine that runs the world, right after its settings load, a Better Continents world with no lakes (or no mask) takes `BepInEx/config/Tarn/<world name>.txt` (or `.clear`) by itself, logs `Lakes: imported <N> lakes from <path>; they are saved with the world at the next save`, and never reads the file again. It takes Tarn's own `DepthScale` from `[Look]` of `com.raveniron.tarn.cfg` when one is set. Tarn's folder is never written to. A vanilla world is never converted by this; `bc_lake import convert confirm` does it by hand.
- **The Tarn mod installed beside Better Continents:** Better Continents' lakes and clearing step aside (no water, no map paint, no sound, no clearing; the data kept and saved), the log, `bc_lake info` and one on-screen message say `Better Continents' lakes are off while the Tarn mod is installed. Remove Tarn to use Better Continents' own.` A player with Tarn installed in a world whose lakes are Better Continents' sees dry hollows.

Part 3: Clearing (experimental)

The mask

A square bitmask centred on the world's middle, `clearmask.bin` in the maps folder (`tarn-clear.bin` when it is missing), read by a new world once and kept in its settings (DataKey 73, deflated). Tarn's `.clear` format, byte for byte:

Offset	Content
0	the ASCII bytes `TARNCLR1`
8	int32 size, little-endian: pixels across and down (Ravenrest's: 4096), at most 16,384
12	float32 half-width in metres, little-endian (Ravenrest's: 10500), at most 10,000 km
16	size x size bits, ceil(size^2 / 8) bytes: rows from the **north** edge (z = +half-width) to the south, each row from west to east, eight pixels a byte with the **first pixel in the highest bit** (numpy `packbits`); bytes after them are ignored

Pixel (r, c) is centred at x = (c + 0.5) x p - half, z = half - (r + 0.5) x p, with p = 2 x half / size, and must be at least 1 cm (a smaller pixel is refused as a mistyped half-width). A marked pixel keeps clear its square grown by 0.25 m on each side; a point is covered when it lies strictly inside one. Better Continents' maps put their first and last pixels on the square's edges (chapter 2), so a mask of the same size and half-width lies over them to within half a pixel.

What it does. A postfix on `ZoneSystem.InsideClearArea`, the game's last test before it places a vegetation spot, answers "inside a clear area" for a covered point. So in a zone generated **after** the mask is in the world nothing the game places as vegetation stands on a marked pixel (trees, rocks, bushes, pickable branches and stones), and the rest is placed as before: the test comes after each spot's rolls, so only an entry with a ground-tilt chance (one more roll for a kept spot) or with forced placement (it makes up a cleared spot elsewhere) shifts, as with the game's own location clear areas. Grass is not vegetation (paint), and locations are not affected. Only the machine that runs the world places vegetation, so only it opens the mask's bits; players receive the packed mask with the world's settings, but a mask loaded during play is not sent.

Write one

```
import numpy as np
from PIL import Image

HALF_WIDTH = 10500.0                    # World Size + Edge Size, as your other maps
picture = Image.open('roads.png').convert('L')     # white = keep clear, north at the top
size = picture.size[0]
assert picture.size == (size, size)
marked = np.asarray(picture) > 127      # rows north to south, each west to east
with open('clearmask.bin', 'wb') as f:
    f.write(b'TARNCLR1')
    f.write(np.array([size], dtype='<i4').tobytes())
    f.write(np.array([HALF_WIDTH], dtype='<f4').tobytes())
    f.write(np.packbits(marked.reshape(-1)).tobytes())
```

The same in the toolkit: `lakekit.write_mask(path, rows, half_width)` (numpy when present, else pure Python) and `python3 lakekit.py mask-from-png roads.png clearmask.bin --half-width 10500`. Ravenrest's pipeline marks its roads + 1 pixel and its lake beds + 3 pixels at 5.13 m a pixel (11.22 km2 of 441 km2), and paints the same ground as dirt.

> **Check:** `python3 lakekit.py check-mask clearmask.bin <x> <z> ...` reads the file with the checks of Better Continents' reader, prints `<size> x <size> over +-<half> m, <p> m a pixel, <N> pixels marked, <A> km2 kept clear`, and says for each point whether it is kept clear: test a point on a road and one beside it. The writer above and `lakekit.write_mask` produce

> exactly the bytes of the mod's own test fixture for the same 37 x 37 pattern (real: `cmp` found them identical). In the game: `Clearing: mask <size> x <size> over +-<half> m, <A> km2 kept free of vegetation` at load, `Clearing: the first spot kept clear of vegetation is at (<x>, <z>), in the zone (<zx>, <zz>)` once, and `bc_lake mask at <x> <z>`.

The other way to clear ground is the vegetation map: its pure white pixels disable every plant at that pixel. Use the mask for Tarn's files or for clearing finer than your vegetation map; both may be used together (either clears).

Part 4: bc genloc

The game's own `genloc` makes the world's locations again. It checks only its slot for the location run (a coroutine), so it also runs after Better Continents skipped the automatic run at world load (`Skipping automatic genloc, use the command manually if needed.`). `bc genloc` (the `bc` cheat tree, `devcommands`; it runs where the world runs: single player, the host or the dedicated server, and a joined client is told so) runs the same generation, which clears every location instance that is not placed and rolls them again in the zones that are not generated yet (placed ones and generated zones stay), and adds:

1. **A report.** At the start, what it found: `Genloc: starting. <N> location(s) now: <P> placed (kept, ...), <U> not placed (cleared, then rolled again ...)`, with a table by type in the log. At the end, what changed: `Genloc: done in <s> s: <N> -> <M> location(s), ...; types that changed: ...; short of their quantity: ...; Better Continents' own placements: ... (a location map pin or the start position that could not be placed); the location icons were sent to the clients.`
2. **The location icons** sent to every client when the run ends (the game sends none after a generation).
3. **A stuck slot freed.** When a run that threw left the game's slot taken, the game's `genloc` does nothing and says nothing. `bc genloc` sees that nothing has run in the slot for a few seconds, stops that run, frees the slot and starts a new one; a run that is alive (the game's own at load) is left alone, and it says so.
4. `bc genloc alt`: the game's own alt-biome branch (`genloc alt`).
5. `bc genloc pins`: puts only the location map's pins and the start position into the registry and generates nothing; for a registry that something else emptied (Upgrade World's `locations_restore`, chapter 6). `bc genloc help` says more.

> **Trap:** a save in the middle of a run keeps only the placed locations (real: a world left in the middle came back with 33 of its 12,308; `bc genloc` then made the rest, 33 -> 12,315). Tell the human to stay in the world until the `done` line.

Traps

In chapter 7 with their samples and fixes: the sea's calm and red (traps 54 and 55, 69 to 71), a lake world opened by an older version (72), a dry or open lake (73, 74), a lake file's warnings (75), clearing that did nothing (76), grass in a bed (77), the Tarn mod installed (78), a mask refused or shifted (79, 80), and `genloc` (81).

Appendix A

Tool reference

All in `generator/` unless stated; all standard-library Python 3 except `altmap_preview.py` (Pillow). Run each as `(ulimit -v 6000000; nice -n 10 ...)`. Each tool's docstring is its manual: `head -30` it.

Tool	Reads	Writes or prints	Time at 2048
`gen.py`	`mapscale.py`	the seven maps and three legends into `<out dir>`; a log of islands, rivers, alt plants and pinned sites	40 s, all cores
`mapscale.py` (imported)		`AMOUNT`, `SEA`, `M_PER_UNIT`, `CEILING`, `FLOOR`, `metres(v)`, `value(m)`	
`png.py` (imported)		`write_gray16`, `write_rgb8` (zlib only): plain PNGs, which Better Continents reads a few rows at a time	
`check.py`	height, biome, heat and location maps	land %, per-biome dry % and heights, the three invariants, the pinned sites	20 s
`relief.py`	height, biome	RMS height change over 50 and 250 m per biome; the coastal height profile	1 min
`preview.py`	height, biome, lava	a shaded picture; scale 2 gives 1024 px, a negative scale enlarges; the crop is in fractions	30 s
`altmap_preview.py`	the alt map and legend, a preview	the preview with the planted regions tinted, outlined and labelled	20 s
`altbiomes.py`	height, biome, alt map; `altbiomes_1.0.15_full.json`	sectors, planted regions, per-alt-biome eligibility and `reaches min` over the seeds; the regions overlay	2 to 5 min
`locations.py`	all maps; `locations_1.0.15.json`; the alt data	every location's placed/quantity with reason counts, the SHORT list, the VERDICT; with `--log`, the game's count on each line	1 to 2 min
`extract_locations.py` (in `../locations/`)	the game's bundles, read-only; needs the `lz4` CLI and `tools/`	`locations_1.0.15.json`, `_summary.txt`	1 min
`extract_altbiomes.py` (in `../altbiomes/`)	the game's bundles, read-only; the same `tools/` beside it	`altbiomes_1.0.15_full.json`, `_summary.txt`	1 min
`stage-into-gale.sh` (in `..`)	`mapscale.py`, the DLL	the plugin and `BetterContinents.cfg` in the Gale profile; refuses while Valheim runs	
`bcp.py` (in `../bake/`; on Wubarrk's machine `ai-skill/bake/`)	a `placements.bcp`	`check`: every check Better Continents' reader makes, then a summary; `demo`: the sample town of chapter 12; as a library, a format-1 writer	under 1 s, also for 702,700 records

Tool	Reads	Writes or prints	Time at 2048
`lakekit.py` (in `../lakes/`; on Wubarrk's machine `ai-skill/lakes/`)	a `lakes.txt`, a `clearmask.bin`, optionally a `.npy` of world heights	`check-lakes`: every warning Better Continents' lake reader logs, and with heights each lake flooded as the builder floods it; `check-mask`: the mask's size, pixel and marked area, and whether points are kept clear; `mask-from-png`; `selftest`; as a library, `make_lake` (`bc_lake at`'s spill search) and the two writers (chapter 13)	seconds; a lake flood of a few km2 in pure Python takes longer

How each is run:

```
python3 gen.py <size> <out dir>
python3 check.py <map dir>
python3 relief.py <map dir>
python3 preview.py <map dir> <out.png> [scale] [x0 y0 x1 y1]
~/upy/bin/python altmap_preview.py <map dir> <base.png> <out.png> [x0 y0 x1 y1]
python3 altbiomes.py <map dir> [overlay.png] [seeds]
python3 locations.py <map dir> [--amount A] [--log LogOutput.log] [--only a,b] [--seed N] [--scale f]
python3 ../locations/extract_locations.py [--game-data DIR] [--out DIR]
python3 ../altbiomes/extract_altbiomes.py [--game-data DIR] [--out DIR]
../stage-into-gale.sh "<profile>" [dll]
python3 ../bake/bcp.py check <file.bcp> [--zone ZX ZZ] [--palette]
python3 ../bake/bcp.py demo <out.bcp> [--at X Z] [--y Y]
python3 ../lakes/lakekit.py check-lakes <lakes.txt> [--heights <grid.npy> --half-width <m>]
python3 ../lakes/lakekit.py check-mask <clearmask.bin> [X Z ...]
python3 ../lakes/lakekit.py mask-from-png <roads.png> <clearmask.bin> --half-width <m>
python3 ../lakes/lakekit.py selftest
```

What each tool prints

Real output, shortened, from map version 13 (`history/out-13-2048/`) unless a line says otherwise. Compare your run with these lines: a run that prints something of another shape has failed, whatever it says last.

`gen.py 2048 <out dir>`: the islands, the progress, the rivers, every alt plant and every pinned site, then `done`:

```
  79 islands with satellites, 74 in archipelagos
  rivers: 65 mouths, 144 reaches, 43 km of water, 29 km of dry gully, 16 lakes (33 basins
» offered), 8017 stamps  (drainage 4s, rivers 0s, rest 1s)
  alt Mushroom                34795 px
  StartTemple          -> world (   -108,   -2201)  [c1meadows]
done 2048
```

`check.py <map dir>`: land, the per-biome table, the three invariants (each must be 0) and the pinned sites:

```
2048x2048   land 27.7% of the image   (reference world Era02: 33.8%)
highest 253 m   deepest -60 m
Meadows        3.04   94.9   14.5   -6.8   35.6
dry ground labelled Ocean: 0   (minimap paints it as sea)
heat > 0 on land that is not AshLands: 0   (MUST be 0 - IsAshlands is heat > 0)
AshLands pixels with no heat: 0
  Dragonqueen                     Mountain      181.4 m     6015 m from centre
```

`relief.py <map dir>`: RMS height change per biome over 50 and 250 m, and the coast profile (chapter 8 has it whole):

```
biome        rms dh 50 m   rms dh 250 m   (metres)
Meadows          4.02         9.07
Meadows     5.5  11.8  15.6  17.7  18.5  19.4  20.6  21.6  21.9  21.5 ...
```

`altbiomes.py <map dir> overlay.png 200`: the regions, the planted ones, one line per alt biome (its targets, minimum distance, edge window, minimum and maximum, planted count, valid hosts, mean total over the seeds, and how often it reaches its minimum), the random hosts, and the overlay picture:

```
309 sectors; 295 are land; 23 planted
  #290   Mushroom               Meadows     at (  1365, -3795)  4033 m  edge  159  h  38.5
alt biome                targets                     dist      edge min-max plant valid   total  reaches min
Mushroom                 Meadows+BlackForest          500 [ 66,199)   1-3       3    20    3.00  100.0%
```

`locations.py <map dir> [--log LogOutput.log]`: the Amount it reads at, every location in the game's order (`!` short, `P` prioritized, the reasons in brackets), the SHORT list with each location's rules, and the VERDICT:

```
heightmap amount 2.0: waterline pixel 0.1500
  Dragonqueen                          3/3    P Mountain
! Bonemass                            4/5    P Swamp                      [zoneTaken 90, area
» 25324, biome 97, altitude 197688, terrainDelta 1937, similar 7776]
VERDICT
  placed nothing: Mistlands_Giant2
  progression and quest sites: StartTemple 1/1, Eikthyrnir 3/3, GDKing 4/4, Bonemass 4/5,
» Dragonqueen 3/3, ...
```

With `--log`, each line also carries the game's count (`game: 0/3`, or `game: all placed`); chapter 3 has a real example.

`preview.py <map dir> <out.png> 1` ends with `wrote <out.png> (512x512)` (a 512 px map at scale 1, real); `png_extras.py record heightmap.png --set 2` prints `heightmap.png: recorded Heightmap Amount = 2; Sea Level Adjustment = 0.5`, and without `--set` prints the record it finds, or `no Better Continents record`.

Two more tools are samples in the chapters, not files of the toolkit: `pnginfo.py` (chapter 2: what a PNG is, as Better Continents sees it) and `mini_gen.py` (chapter 5: the generator's skeleton in about 120 lines). The fine-heights writer in chapter 2 is a function to paste into a generator.

Three more scripts live in `ai-skill/`, beside this document's source, and need no game data:

Tool	Usage	Reads	Writes or prints
`png_extras.py`	`png_extras.py record FILE [--set AMOUNT [SEA]]`; `png_extras.py gray-alpha OUT HEIGHT ALPHA`	a PNG	prints or writes the Heightmap Amount and Sea Level record of a heightmap (chapter 2); joins two 16-bit grey PNGs into a 16-bit grey + alpha PNG for Heightmap Alpha (chapter 10). Changes no pixel of the file it records.
`make_b_config.py`	`make_b_config.py [--repo DIR] [--schema FILE] [--out FILE] [--check]`	the mod's `SettingsSchema.cs`, `BetterContinents.MapKinds.cs`, `Legends.cs`, `DebugUtils*.cs`, `ImageMapBiome.cs`	`docs/b-config-reference.md`: every setting, map kind, legend rule and console command as the mod declares them (appendix B); `--check` instead checks that the numbers the chapters quote still hold in the source
`make_c_data.py`	`make_c_data.py`	the extracted game data, the mod's `ImageMapBiome.cs` and alt-biome palette	`docs/c-data.md`: every location and alt biome, the high-ground table, the biome legend (appendix C)

Run the two generators, and then `build_ai_skill_pdf.py`, whenever the mod's settings or the game data change. The mod's source folder is `--repo`, else `BC_REPO`, else `/home/rohan/WubarrkCODING/BetterContinents10`.

Tools live in the game itself, too, from Better Continents 0.9.0 on: chapters 9 and 10 cover them, and appendix B lists them as the mod declares them.

Tool	Where and who	Reads	Writes or does
`bc_export`	the console in a loaded world; any player (server-gated)	the world you are standing in	a folder of maps under `<save data>/BetterContinents/<world>/export-<time>/`, up to 16,384 px, plus a New World preset up to 8,192 px
`bc_import`	the console, main menu or in a world; any player	a folder of maps (an export, or one of this workspace's)	a New World preset, or `BetterContinents.cfg`
`bc_cache`	the console; any player	a world's cache entry	a shareable `.bcworld` file (up to 1.5 GB); the local cache
`bc_twins`	the console in a loaded world; any player counts, the host or a dedicated server removes	the vegetation the machine holds	counts doubled plants; with `remove confirm` deletes the extra copies
`bc_altbiomes`	the console in a loaded world; any player	the world's alt-biome data	the summary, the region you stand in, the names, the agreement hashes
`bc`	the console, cheats on (Debug Mode turns them on); single player or the host	the loaded world, if it uses Better Continents	changes its settings live, reloads maps, regenerates the world's zones except those around what players built or worked, a tombstone or another player (chapter 6)

Samples of the in-game tools. `bc info` prints the world's settings dump; three of its lines, in the forms the mod's source prints them, with ProximaMaxi's values on a new world:

```
bc info
Version 12
Heightmap amount 2, blend 1, add 0, mask 0
Biome precision 0: the ground takes its biomes from the 4 corners of each 64 m terrain zone (vanilla)
```

Type	What it does
`bc_export 4096 amount=2`	exports the loaded world at 4,096 px, heights encoded for Amount 2 (`bc_export help` lists every option)
`bc_import list`	lists your export folders, newest first, numbered; `bc_import <number>` makes a New World preset from one
`bc_altbiomes here`	describes the alt-biome region you stand in
`bc bosses`	puts pins on the large map for the bosses, the start temple and the trader
`bc scr`	saves the minimap as a PNG (2048 px unless you pass a size)

If you cannot run the toolkit

Every check has a weaker form that needs only the game and its log, or only a human's eyes. Use the strongest you have, and say which you used.

What is checked	With the toolkit	With the game and its log	With only the human
the height scale (a drowned world)	`check.py`: per-biome % dry and mean height	`Heightmap amount A, ...` in the settings dump; walk a meadow	"Is the land you start on dry, and how high are the hills?"
the files' kinds and sizes	`pnginfo.py` (chapter 2)	`Time to load <path>` for each map; no `read whole` line	the editor's image properties: size, bit depth, colour mode
the biome shares	`check.py`'s `% map` column	the `Biome map (N x N): ...` line	the preview
the heat invariant	`check.py`: `heat > 0 on land that is not AshLands: 0`	Ashlands weather or lava where there should be none	(cannot be seen in a picture: insist on the toolkit or the game)
the pinned sites	`check.py`'s site list	`Selected <name> for spawn position`, `Position of <name> ... overriden`; `bc bosses`	the map pins in the game
the locations	`locations.py`	`Failed to place all ...` lines (chapter 3)	the bosses' pins and a sail round the world
the alt biomes	`altbiomes.py`	`Alt biomes: planted: ...`; `bc ab info`; `bc_altbiomes here`	(cannot be seen in a picture)
the look	`preview.py`, `relief.py`	`bc scr` (the minimap as a picture)	the human's eye, always the judge

Which tools assume Amount 2, and which assume the 21,000 m square

The workspace's toolkit was built for Heightmap Amount 2 and the default world size. None of its scripts is changed by this edition; where one assumes either, the text says so and this table gives the Amount 1 equivalent.

Script	What it assumes	At Heightmap Amount 1
`generator/mapscale.py`	`AMOUNT = 2.0`, the one place; `SEA`, `M_PER_UNIT`, `CEILING` and `FLOOR` derive from it	set `AMOUNT = 1.0` in a copy of the generator folder: they become 0.30, 200, 140 and -60
`generator/gen.py`	`HEIGHTMAP_AMOUNT = mapscale.AMOUNT`; every height is a constant in metres through `H()`; mountains to 254 m	the same constants clip at 140 m, silently: lower the mountains first (chapter 5)
`check.py`, `preview.py`, `relief.py`	`mapscale.SEA` and `M_PER_UNIT`	nothing to change; they follow `AMOUNT`
`altbiomes.py`	`(h * mapscale.AMOUNT - 0.15) * 200` for the heights it samples	follows `AMOUNT`
`locations.py`	`mapscale.AMOUNT`, overridable with `--amount A`	`locations.py <dir> --amount 1` replays a map authored for 1 without touching `mapscale.py`
`stage-into-gale.sh`	writes `Heightmap Amount =` the value in `mapscale.py`	follows `AMOUNT`
`fire-and-ice/`	the ProximaMaxi map, authored for 2	read at 1 it drowns; give it `Heightmap Amount = 2` wherever it is read

Script	What it assumes about the square	For another World Size or image size
`gen.py`	`21000.0` for metres per pixel and for the pinned sites' coordinates, `10500.0` in the rivers and the edge	change those constants

Script	What it assumes about the square	For another World Size or image size
`check.py`	`21000.0` for the pinned sites' coordinates	change it
`altbiomes.py`	`TOTAL = 21000.0`, the 2048-point 12 m grid (`N, PIX, HALF`), the 10,500 m radius	change them (chapter 3 gives the grid of a bigger world)
`locations.py`	the grid and `TOTAL` from `altbiomes.py`; 10,000 m for the search from the centre; it models a pin as the mean position of its colour's pixels	change them with `altbiomes.py`; a colour shared by several blocks is not dealt out as the game does
`relief.py`	5 and 25 pixels as its "50 m" and "250 m"	right only for a 2048 px map over 21,000 m (about 10 m a pixel)

And what Better Continents 0.10.3 added, none of which the toolkit was built for:

Script	Tall worlds, fine heights, big maps
`mapscale.py`, `check.py`, `preview.py`, `relief.py`	any Amount: the maths is the same at 81 as at 2 (chapter 2); `preview.py`'s brightness stops growing at 120 m above the water, so a tall world's heights above that show only in its hillshade
`gen.py`	any Amount through `H()`; writes no fine picture (it truncates its doubles to 16 bits); holds every map whole, in lists and arrays of `SIZE x SIZE`, so its memory grows with the square of the size
`png.py`, `png_extras.py`	no 8-bit grey writer and no fine record: `png_extras.py record` writes the Amount record only
`locations.py`	applies every upper altitude limit as the game data gives it; a high world lifts those of 1,000 m or more, so there it under-counts the locations on ground that high (chapter 3)
`altbiomes.py`, `locations.py`	the 2048-point 12 m grid; from a radius of 42,000 m the game's grid stays at 8192 points a side, its points more than 12 m apart (chapter 3)

What the checkers reproduce, and what they cannot

`altbiomes.py` replays `AltBiomeWorldData` step for step (the grid, the flood fill, the global sectors, EdgeCount, HeightAvg, DistanceFromCenter, `CanAddModifier`) and the mod's planting (the split, the reuse, the quota). It does not reproduce Unity's random numbers, so random placement is simulated over many seeds with the same algorithm (biome order, list order, minimum then chance, maximum, incompatibility): it says how reliably each alt biome finds a home, not where a given world seed puts it.

`locations.py` replays `ZoneSystem.GenerateLocations` (the order, the attempt counts, the candidate source, zone occupancy, biome area, six points per zone, distance, biome vote, altitude, forest, terrain delta, similar and group spacing, vegetation) with the location map's sites registered first, unchecked, as the game does. It cannot reproduce Unity's random numbers (counts are typical, not the seed's), the vanilla forest and Mistlands vegetation noise under a forest map (modelled as noise of the right range), or the alt-biome-added locations and blocks (listed, not placed). Validated against real builds: it matched the game line for line on the failures that mattered (the cavern 0/3, tower ruins 0/50, frost caves 12/120, frozen ships 0/50, Deep North wrecks 22/170 on the Amount 1 map; 148 of 170 lines exactly on map version 13).

Neither checker models Expand World Data's altitude rules, lava biomes or biome numbering (chapter 10), nor the twin guard, nor a world laid out to another size, nor a high world's rules or fine heights (chapters 2 and 3).

When the game changes

A new Valheim build changes the location and alt-biome lists. Re-extract them (both extractors are read-only on the game data and need only Python, the `lz4` CLI and the small `tools/` copies of the libs-Tools asset readers), diff the summaries against the old ones, and re-run both checkers. Check the game's version by the md5 of `assembly_valheim.dll` (every Valheim assembly is stamped 0.0.0.0; compare it with the installed game's copy), and confirm any claim about the game in the whole-assembly decompile at `libs-Tools/1.0/DECOMPILED/assembly_valheim.decompiled.cs` (the per-type `Decompiled_<version>/` is faster to read; check it is the current build's, and `libs-Tools/OLD/` holds older builds).

A new Better Continents version: read its CHANGELOG and `ALTBIOMES.md`, run `make_b_config.py` and `make_c_data.py`, and rebuild. The log lines in this document are 0.10.x's. The height formula and the legend formats have not changed since 0.8.1: what 0.9.0 added (export, import, Biome precision) is chapter 9, and what 0.10 changed for a new world is chapter 10. 0.10.3 widened the formula's range (Heightmap Amount to 81) and put fine heights beside it without changing it (chapter 2).

Appendix B

The config that goes with the map

`stage-into-gale.sh` writes this into `<profile>/BepInEx/config/BetterContinents.cfg`. The comments are the reasons, each on a line of its own (the script writes longer ones). The next page, generated from the mod's own source, lists every setting of the file.

```
[00 BetterContinents.Debug]
Enabled = true
# turns on cheats when you spawn, for the bc commands: a test profile; off for play
Debug Mode = true
Directory = /home/rohan/WubarrkCODING/BetterContinents10/.work/BetterContinents-TestMaps/fire-and-ice

[02 BetterContinents.Heightmap]
# from mapscale.py; MUST match the maps (mod default: 1)
Heightmap Amount = 2.0

[04 BetterContinents.Forest]
# neutral
Forest Amount = 0.50
Forest Factor Overrides All Trees = true
# the additive pair: the forest map adds trees to the game's own forest
Forestmap Multiply = 0
Forestmap Add = 1

[07 BetterContinents.Misc]
# the literal sentinel; Disabled makes a vanilla world, a preset's file that preset's world
SelectedPreset = From Config

[08 BetterContinents.AltBiomes]
# painted on the west massif; no random one elsewhere
Overrides = Wolf Mountain: enabled=false
```

Everything not listed stays at the mod's default: `Heightmap Override All = true` (vanilla's rivers and height curves are bypassed), `Sea Level Adjustment = 0.5` (offset 0), `World Size = 10000` and `Edge Size = 500` (the 21,000 m square that the generator and the checkers assume), `Heatmap Scale = 10`, alt-biome `Mode = Random` and `Grid = WorldEdge`, and `Forest Scale = 0.5`, the game's own size of forest and clearing patches. A config written by Better Continents 0.10.1 or older holds `Forest Scale = 1` (patches five times that size, the old default), and the first start of 0.10.2 sets it to 0.5 once; on 0.10.1 and older, write `Forest Scale = 0.5` yourself. The keys Better Continents 0.10.3 added stay at their defaults too (below): Wide Sectors converts no existing world, no limit below 16,384 px is set, and Compact Maps decides by the maps. The `Directory` setting auto-loads every correctly named file in the folder, `altbiomemap.png` and `lavamap.png` included. Since 0.9.4 it also reads an `export.cfg` there, whose settings then win over this file (chapter 9); `fire-and-ice/` has none, so this file is the whole story.

The world then carries a baked copy of all of it. To see what a world was built from, `bc info` prints its settings in game (chapter 6), and the fork's `tools/altbiome-harness` has an `inspect` command that reads a COPY of a world's `BetterContinents` settings file; never point it at the live one.

`SelectedPreset` and `NexusID` are hidden in configuration managers but are keys of the file. `SelectedPreset` is the New World dropdown's choice: `From Config` makes the world from this file, a preset file's path makes it from that preset, and the default, `Disabled`, makes a vanilla world with no warning (chapter 2).

On any machine

The file is `BepInEx/config/BetterContinents.cfg`, in the game folder or in the mod manager's profile, and Better Continents writes it the first time the game starts with the mod. It is a BepInEx file: `[NN Group]` section headers, `Key = value` lines under them, a dot for decimals, `##` comment lines that Better Continents writes and that you can leave. Edit the lines that are there; a key written under the wrong section is not that setting. Better Continents reads the file again whenever it changes on disk.

> **Rule: A comment is a line of its own, starting with `#`.** BepInEx skips such lines and takes everything after a line's `=` as the value, so a comment at the end of a line becomes part of the value: `Heightmap Amount = 2 # map` is not read as a number (the log says the value `could not be parsed and will be ignored`, and the Amount stays at its default of 1), and a name or a path gets the comment in it.

The lines each recipe of chapter 11 sets (everything else at its default):

Recipe	`[00 ...Debug]`	`[01 ...Global]`	`[02 ...Heightmap]`	`[07 ...Misc]`	other
1 Island world	`Enabled = true`, `Directory`	(defaults: 10,000 and 500)	`Heightmap Amount = 2`	`SelectedPreset = From Config`	
2 Continent, alt biomes	the same		`Heightmap Amount = 2`	the same	the forest pair; `Overrides = Wolf Mountain: enabled=false`
3 Big world	the same	`World Size = 24000`, `Edge Size = 500`	`Heightmap Amount = 2`	the same, `Wide Sectors = Auto`, `Compact Maps = On`	
4 16k tall world	the same	`World Size = 32264`, `Edge Size = 500`	`Heightmap Amount = 81`, `High Terrain = Auto`	the same, `Max Map Size = 16384`, `Compact Maps = Auto`, `Wide Sectors = Auto`, `Map Memory = 1024`	
5 Export round trip	set by `bc_import` (a preset, or `config`)	from `export.cfg`	from `export.cfg`	`SelectedPreset` set by the import	
6 Server	(the world is made on a client)			on the server: `Settings Transfer Rate = MB3`, `Map Memory`	`Debug Mode = false`

> **Check it:** after the world is made, the settings dump in `LogOutput.log` (or `bc info` in the world) must show what you set: `Version 12`, `Heightmap amount 2, blend 1, add 0, mask 0`, `Heightmap file (2048) <your folder>/heightmap.png`. A line that disagrees with the config means the world was made from something else: a preset, an `export.cfg` in the folder (chapter 9), or the dropdown on Disabled.

The Gale profile

`BC-Tester`, at `~/.local/share/com.kesomannen.gale/valheim/profiles/BC-Tester/`: the BepInEx pack 5.4.2350, Better Continents 0.10.x in `BepInEx/plugins/Wubarrk-BetterContinents/`, plus Server devcommands (optional; the mod's console integration with Server devcommands 1.51 or later gives autocomplete). Better Continents regenerates the zones itself after a `bc` change (chapter 6): `Debug Reset Command` is empty by default, and a console command written there, such as Upgrade World's `zones_reset start`, runs instead. A profile whose config was written by 0.9.x holds `Debug Reset Command = zones_reset start`; the first start of 0.10 sets it to empty once and says so in the log. With Upgrade World installed, `[07 BetterContinents.Misc] Upgrade World Radius` (Off by default) raises Upgrade World's own World radius (default 10,500 m) to cover a world past that size, and `bc regen dry biome=...` then `bc regen biome=...` resets by biome, which Upgrade World's `upgrade ashlands` does not on a map of its own (chapter 6). Create profiles in Gale itself (it tracks them in a database; a folder made by hand does not show up), install the BepInEx pack into the profile, then stage.

Biome precision and the Export group (0.9.0, not written by the staging script)

Two more groups exist since Better Continents 0.9.0. `stage-into-gale.sh` sets neither, so ProximaMaxi runs at their defaults; chapter 9 covers the console side of both.

```
[03 BetterContinents.Biomemap]
# 0-31 on a new world, 0-5 on an older one; 0 = vanilla
Biome precision = 0

[09 BetterContinents.Export]
# F9 status box / F7 window; on turns them on for this game (and its clients)
Hud = false
# whether a connected player may bc_export this world at all
Allow Export = true
Hud Hotkey = F9
Window Hotkey = F7
# bc_export's size when none is given, 128-16384
Default Size = 4096
# bc_export's Amount by default, 0.01-81; this workspace uses 2
Default Heightmap Amount = 1
Default Sea Level = 0.5
# 4096, 8192 or 16384: the largest export this machine makes
Largest Size = 16384
```

Fine heights are not a key of this group, nor of any: whether an export writes `heightmap-fine.png` beside its heightmap is a choice of each export, `bc_export fine=auto|off|4|8` or the export window's Fine heights choice, and `auto`, the default, writes it from a Heightmap Amount of about 4.92 (chapter 9).

Unlike the groups above, `[09 BetterContinents.Export]` applies while the game runs (as do `Settings Transfer Rate`, `Map Memory` and `Max Map Size` in `[07 BetterContinents.Misc]`; `Wide Sectors` there is read by the machine that runs a world, when the world is made and each time it loads), and on a server its `Hud`/`Allow Export` reach connected clients too. Better Continents also re-reads the whole config file whenever it changes on disk, not only once at launch (chapter 9): edit `BetterContinents.cfg` while the game is running, and a **New** world you then create in that same session reads the new values, no relaunch needed. This narrows the premise behind trap 3 (chapter 7) and behind `stage-into-gale.sh`'s own refusal to run while Valheim is running - both were written for a version that read the file only at launch. Whether the script's refusal is still needed for some other reason (such as replacing a Gale-hardlinked DLL safely) was not checked while writing this appendix; treat it as still in force and quit before staging until someone has.

Big maps, big worlds and tall worlds (0.10.3, not written by the staging script)

Better Continents 0.10.3 added these keys, and gave `Compact Maps` (a switch since 0.10.0) three values. `stage-into-gale.sh` writes none of them, so a staged profile runs at the defaults shown here, and ProximaMaxi (2048 px, the default size, Amount 2) uses nothing they govern.

```
[02 BetterContinents.Heightmap]
# high-world rules (chapter 3) for a heightmap read above Amount 5; On: every world; Off: none
High Terrain = Auto

[07 BetterContinents.Misc]
# 4096, 8192 or 16384: the largest map this machine reads from a file (live)
Max Map Size = 16384
# Auto: compact for a map over 8,192 px or fine heights; On: always; Off: never
Compact Maps = Auto
# Auto: a NEW world past 16,350 m is made wide; On: existing ones too, one way; Off: none
Wide Sectors = Auto
```

`High Terrain` and `Compact Maps` are world settings: a new world takes them when it is made and keeps them. `Max Map Size` and `Wide Sectors` are each machine's own: no world, export or import carries them (a wide world records only that it is wide). `Wide Sectors` is read by the machine that runs a world, when the world is made and each time it loads, and a joining player's game follows the world. `Wide Sectors = On` converts every existing big world the profile then opens, and that cannot be undone (chapter 10; the game saves its own copy of the old save first, `<world>_backup_<date and time>`): never write it unless the human asks for a world to be converted, after a backup of it. An older config's `Compact Maps = true` reads as On and `false` as Auto. `Map Memory` keeps its 512 MB; for a 16k world on a server with memory to spare, 1024 to 2048 (chapter 10). `Heightmap Amount` itself takes 0 to 81 since 0.10.3 (chapter 2).

Baked placements (0.10.4, not written by the staging script)

Better Continents 0.10.4 added `[10 BetterContinents.BakedPlacements]`: how baked pieces are drawn on each player's own screen. No setting there changes a world, every one applies at once, and the staging script writes none of them. The defaults, with what each does in a word (chapter 12 has the table and the order to lower them in):

```
[10 BetterContinents.BakedPlacements]
# zones around the camera that are drawn, 1-8 (4: a block of 9 x 9)
Draw Distance = 4
# how far a piece is drawn, x the game's own distance, 0.5-4
Draw Scale = 1.5
# how far a piece keeps full detail, x the game's own, 0.5-4
Detail Scale = 1
# All, Near or Off: which baked pieces cast shadows
Shadows = All
# metres within which baked lights can burn, 0-200
Light Distance = 90
# metres within which baked seats can be sat on, 0-40
Seat Distance = 12
# how many baked lights burn at once, the nearest, 0-128
Light Count = 24
# how many of them cast shadows, 0-32
Light Shadows = 4
# name=r,g,b; ... replaces a town file's tint colours
Tints =
Diagnostics = false
```

`Hud` in `[09 BetterContinents.Export]` also opens the Bake buildings tab (F7); on a server, the server's value counts.

Water and lakes (0.10.5, not written by the staging script)

Better Continents 0.10.5 added two sections (chapter 13). `[11 BetterContinents.Water]` is each player's own drawing of the sea: neither key is world data, and neither changes the waves that are felt. `[12 BetterContinents.Lakes]` (experimental) holds the one world setting of the lakes, which a new world copies, and four of each player's own, read live:

```
[11 BetterContinents.Water]
# Better Continents' own water where the world's sea is not the game's; off: the game's water
Own Water = true
# with Own Water, the far sea takes the depth of the sea under it
Soft Far Sea = true

[12 BetterContinents.Lakes]
# world: how deep the water takes a new world's lakes to be, x their depth, 0-3 (bc_lake depth
» in a running world)
Lake Depth Scale = 0.5
# metres around the player within which lakes are built, 150-2000
Lake Build Range = 400
# milliseconds a frame for making lake water, 0.5-20
Lake Build Budget = 2
# the water sound near lakes, 0-1
Lake Sound = 0.35
Lakes On Map = true
```

The Directory's `lakes.txt` and `clearmask.bin` (or `tarn-lakes.txt` and `tarn-clear.bin`) are read like its maps when a world is made; there is no `*File` key for them.

The world

`~/.config/unity3d/IronGate/Valheim/worlds_local/<name>/` is Valheim 1.0's folder-per-world layout, with `_main.N.fwl2` and `.db2` files and the mod's extension-less `BetterContinents` file beside them. Copy a world by copying the whole folder. Never rename or move a modded world through the game's own save menu unless the mod version is known to carry the sidecar through it (0.8.x and later do).

Config reference, generated from the mod

This page is written by `make_b_config.py` from three files of Better Continents' own source, `SettingsSchema.cs` (every setting, declared once), `BetterContinents.MapKinds.cs` (the image maps) and `Legends.cs` (how each legend reads), plus the `bc` console's command table in `DebugUtils*.cs`. It is regenerated, never edited: when the mod's settings change, run the script again and rebuild the PDF. Generated from commit 263e929 (with uncommitted changes to the schema) of the mod's source: 98 settings in 13 sections.

The sections are `[NN BetterContinents.Group]` in `BepInEx/config/BetterContinents.cfg`. The number is part of the section's name; sections are never reordered or renamed, and a new one only ever goes last. Every key in this page is written as the config file spells it, spaces and capitals included.

Scope: who reads a setting, and when

- **World.** A default for NEW worlds: baked into a world's settings when the world is created (From Config, an import, a Directory's export.cfg) and never read again for that world. Override version is the exception among the world settings: it is read at every save and every join.
- **Live.** Applies while the game runs (LiveConfig): the export HUD and gate, the export defaults, the transfer rate. Also what a machine that runs a world decides for itself, when a world is made and when it loads (Wide Sectors); never part of a world, an export or an import.
- **Local.** This game's own bookkeeping and debug switches: never part of a world, an export or an import.

A setting with a **console** entry can be changed in a loaded world with `bc <group> <name> <value>` (the `bc` commands are cheat commands and run on the host; Debug Mode switches cheats on when you spawn). `bc <group> <name>` alone prints the current value, `bc <group> help` lists the group. The console takes the same range as the config, and a few settings a range of their own. What a console change redoes depends on the setting:

- **Regenerate.** Everything that reads the settings: noise, patches, the minimap, the alt-biome grid and the zones.
- **Precision.** Biome precision: how zones sample the biomes, not the biomes - patches and the zones only.
- **HeightmapDecode.** Heightmap Alpha: how the heightmap is read - decode it again, then everything Regenerate redoes.

[00 BetterContinents.Debug]

Setting	Value	What it does
Enabled world	`bool` default `true`	Whether this mod is enabled
Debug Mode local	`bool` default `false`	For editing a world's settings live with the bc commands. In a world that uses Better Continents, on the machine that runs it (single player or the host), it turns devcommands on when you spawn, shows "Better Continents Debug Mode Enabled!" at the top left, and adds a Better Continents button to the Esc menu and the large map that opens the settings window (Alt+F8 there does the same). Leave it off for a world you play.
Debug Reset Command local	`string` default `(empty)`	Empty (the default): Better Continents regenerates the zones itself after a change made with the bc commands and on 'bc regen'. Every generated zone is reset, and generates again with the current settings when somebody comes near, except zones within one zone of something a player built or worked on (a field, a path, levelled ground), a tombstone, a player connected from another machine (as far as their game keeps zones loaded around them), or you inside a dungeon. A location that reaches into one of those zones is kept whole. Players, what players built, tombstones and tamed animals are never removed. A console command written here (for example zones_reset start) runs instead.

Setting	Value	What it does
Override version world	`string` default `(empty)`	Empty (the default): every world is saved in its own settings version (one older than 11 as 11), and a new world gets the newest (12, made by 0.10). A number here saves every world in that version instead, and a new world made From Config gets it when it is 11 or more: 11 makes worlds as Better Continents 0.9 did. For testing only: a world saved in an older version than its own loses what its version added.
Directory world	`string` default `(empty)`	A folder of maps, each loaded by its standard name: heightmap.png, biomemap.png, terrainmap.png, locationmap.png, roughmap.png, forestmap.png, heatmap.png, coldmap.png, paintmap.png, lavamap.png, mossmap.png, vegetationmap.png, spawnmap.png, altbiomemap.png (and the legends beside them), and lakes.txt and clearmask.bin (a world's lakes and its mask of ground kept free of trees, rocks and bushes, in Tarn's formats; tarn-lakes.txt and tarn-clear.bin are taken when those are missing). When it is set, the map file settings are not used. A folder with an export.cfg in it (a world export) also brings the settings in that file, which win over this one.

[01 BetterContinents.Global]

Setting	Value	What it does
Skip Default Locations console `bc g skipdefaultlocations`, world	`bool` default `false`	Skips the default location placement. Spawn temple and location map are still placed.
Continent Size console `bc g cs`, world	`float` default `0.5` range 0 to 1	Continent size
World Size console `bc g worldsize`, world	`float` default `10,000` console range 0 to 1,000,000	The world's radius in metres (vanilla 10000). A new world is made to its size: its maps span World Size + Edge Size, and its alt-biome grid, locations, the game's own biome bands, Ashlands, Deep North, lakes and rivers, and the minimap follow it. A world made by an older Better Continents keeps vanilla's 21000 m, and World Size only moves its edge. Expand World Size, when installed, sets the world's size and lays it out itself.
Edge Size console `bc g edgesize`, world	`float` default `500` console range 0 to 1,000,000	How far the world's edge reaches past World Size, in metres (vanilla 500). On a new world the land drops away over it.
Sea Level Adjustment console `bc g sl`, world	`float` default `0.5` range 0 to 1	Modify sea level, which changes the land:sea ratio
Ashlands Gap console `bc g ag`, world	`bool` default `false`	Whether to add the Ashlands ocean gap (usually custom maps don't need this)
Deep North Gap console `bc g ng`, world	`bool` default `false`	Whether to add the Deep North ocean gap (usually custom maps don't need this)
Rivers console `bc g r`, world	`bool` default `true`	Whether rivers should be enabled or not
Map Edge Drop-off console `bc g me`, world	`bool` default `true`	Whether the map should drop off at the edges or not (consequences unknown!)

Setting	Value	What it does
Mountains Allowed At Center console `bc g mc`, world	`bool` default `false`	Whether the map should allow mountains to occur at the map center (if you have default spawn then you should keep this unchecked)
Mountain Noise console `bc g mn`, world	`float` default `1` range 0 to 5	Procedural noise scale added to Mountain biomes even with Heightmap Override All enabled (0 = perfectly flat to heightmap, 1 = Valheim's natural alpine fractal ridges, couloirs and jaggedness). Lowlands, towns and shores remain 100% faithful to the heightmap.
Mistlands Noise console `bc g msn`, world	`float` default `1` range 0 to 5	Procedural crag and stepped terrace noise scale added to Mistlands biomes with Heightmap Override All enabled (0 = flat to heightmap, 1 = Valheim's natural stepped crags).
Ashlands Water console `bc g aw`, world	`bool` default `true`	Whether the Ashlands water color follows the world's hot sea rather than vanilla's hardcoded south circle
Deep North Water console `bc g dnw`, world	`bool` default `true`	Whether Deep North calm sea follows the world's custom Deep North biome or cold map rather than vanilla's hardcoded north circle

[02 BetterContinents.Heightmap]

Setting	Value	What it does
Heightmap File world	`string` default `(empty)`	Path to a heightmap: a square image, 16-bit grey for smooth slopes, white highest (see Heightmap Amount). A heightmap-fine.png beside it (8-bit grey, the same size) adds finer steps. Not used when Directory is set: its heightmap.png is used.
Heightmap Amount console `bc h am`, world	`float` default `1` range 0 to 81	Multiplier of the heightmap's values. With Sea Level Adjustment 0.5, Heightmap Amount 1 (the default) gives heights of -30 m to 170 m (the sea is at 30 m) and clips vanilla's highest mountains; 2 gives -30 m to 370 m and keeps them all; 81, the most, gives -30 m to 16,170 m. A world whose heightmap is read at an amount above 5, the most older versions allowed, is also given Better Continents' fixes for the height rules the game assumes (what counts as inside a dungeon, where the ground and the grass are looked for, how high plants, creatures and locations may be, which heights the creatures can walk). A world export records the amount its heightmap was made for. At 81 one step of a 16-bit heightmap is 25 cm high: a heightmap-fine.png beside it makes the steps finer.
Heightmap Blend console `bc h bl`, world	`float` default `1` range 0 to 1	How strongly to blend the heightmap file into the final result
Heightmap Add console `bc h ad`, world	`float` default `0` range -1 to 1	How strongly to add the heightmap file to the final result (usually you want to blend it instead)
Heightmap Mask console `bc h ma`, world	`float` default `0` range 0 to 1	How strongly to apply the heightmap as a mask on normal height generation (i.e. it limits maximum height to the height of the mask)
Heightmap Override All console `bc h ov`, world	`bool` default `true`	All other aspects of the height calculation will be disabled, so the world will perfectly conform to your heightmap. With it off, the game's own biome heights are made over the heightmap's, and the Mountain biome doubles what is over 80 m: a peak of 1,000 m in the heightmap is about 1,900 m high

Setting	Value	What it does
Heightmap Alpha console `bc h alpha`, world	`bool` default `false`	Enables alpha channel for the heightmap file to blend vanilla generation with the heightmap. A new world reads the heightmap at full precision and blends it by its alpha: where a pixel is transparent, the game's own terrain shows. A world made by an older Better Continents keeps reading the heightmap at 8 bits, without the blend.
Roughmap File world	`string` default `(empty)`	Path to a roughmap: a square grey image of where the biomes' own rough ground shows through the heightmap (white) and where the heightmap stays smooth (black). Only used with Heightmap Override All off. Not used when Directory is set: its roughmap.png is used.
Roughmap Blend console `bc r bl`, world	`float` default `1` range 0 to 1	How strongly to apply the roughmap file
High Terrain console `bc h ht`, world	one of Auto = 0, On = 1, Off = 2 default `Auto`	Whether Better Continents adjusts the game's own rules for high ground. Those rules assume land under about 400 m: what counts as inside a dungeon (anything above 3,000 m), where the ground is looked for from, how high grass, plants, creatures and locations may stand, which ground creatures in dungeons can walk, the Valkyrie's flight height, random events, and the Deep North weather test. Auto (the default): a world whose heightmap is read at a Heightmap Amount above 5 gets the adjustments, and every other world keeps the game's own rules. On: every Better Continents world gets them, whatever its heights. Off: none does, whatever Heightmap Amount is. Off costs a world with high ground: no grass above 500 m, no plants, creatures or locations above about 1,000 m, no terrain, weather or building above 3,000 m (the game takes it for the inside of a dungeon), and no ground found above 6,000 m. A new world takes this setting when it is made (From Config, bc_import, or a Directory's export.cfg) and keeps it; 'bc h ht' changes it in a world that is running (Debug Mode)

[03 BetterContinents.Biomemap]

Setting	Value	What it does
Biomemap File world	`string` default `(empty)`	Path to a biome map: a square image with a colour per biome, named by the legend beside it (biomemap.txt, 'Biome: colour' lines, written with the default colours when missing). Not used when Directory is set: its biomemap.png is used.
Biome precision console `bc b p`, world	`int` default `0` range 0 to 31	How closely the ground follows the biome borders inside each 64 m terrain zone (ground textures, grass, vegetation and spawn points). 0 = vanilla: a zone takes its biomes from its 4 corners, so the borders follow the 64 m zone grid. 1 to 5 split every zone into (N + 1) x (N + 1) cells, with a corner every 32, 21, 16, 13 or 11 m; a new world goes on up to 31 (a corner every 2 m), and an older one stops at 5. Works with or without a biomemap, and terrain heights do not change
Terrainmap file world	`string` default `(empty)`	Path to a terrain map: a square image of the ground colour, named by its legend (terrainmap.txt, 'Ground: colour' lines). Not used when Directory is set: its terrainmap.png is used.

[04 BetterContinents.Forest]

Setting	Value	What it does
Forest Scale console `bc fo sc`, world	`float` default `0.5` range 0 to 10	Size of the game's own forest and clearing patches (a forestmap is not scaled). 0.5 (the default) is the game's own size, 0 about a third of it, 1 about five times larger. Keep it within 0 to 1: near 1.15 the game's own forest is the same everywhere, and above that the patches shrink again.
Forest Amount console `bc fo am`, world	`float` default `0.5` range 0 to 1	Adjusts how much forest there is, relative to clearings
Forest Factor Overrides All Trees console `bc fo ffo`, world	`bool` default `false`	Trees in all biomes will be affected by forest factor (both procedural and from forestmap)
Forestmap File world	`string` default `(empty)`	Path to a forest map: a square grey image, white for more forest, combined with the game's own forest by Forestmap Multiply and Forestmap Add. Not used when Directory is set: its forestmap.png is used.
Forestmap Multiply console `bc fo mu`, world	`float` default `1` range 0 to 1	How strongly to scale the vanilla forest factor by the forestmap
Forestmap Add console `bc fo add`, world	`float` default `1` range 0 to 1	How strongly to add the forestmap directly to the vanilla forest factor

[05 BetterContinents.StartPosition]

Setting	Value	What it does
Override Start Position console `bc st os`, world	`bool` default `false`	Whether to override the start position using the values provided (warning: will disable all validation of the position)
Start Position X console `bc st x`, world	`float` default `0` range -1,000,000 to 1,000,000	Start position override X value, in metres from the world's centre (vanilla's world reaches 10500 either way)
Start Position Y console `bc st y`, world	`float` default `0` range -1,000,000 to 1,000,000	Start position override Y value, in metres from the world's centre (vanilla's world reaches 10500 either way)

[06 BetterContinents.Maps]

Setting	Value	What it does
Locationmap File world	`string` default `(empty)`	Path to a location map: a square image, black with one dot of colour per location, named by its legend (locationmap.txt, 'Location: colour' lines). Not used when Directory is set: its locationmap.png is used.
Spawnmap File world	`string` default `(empty)`	Path to a spawn map: a square image of which creatures may spawn where, named by its legend (spawnmap.txt, 'r,g,b,a: +Creature, -Creature' lines; white stops every spawner). Not used when Directory is set: its spawnmap.png is used.

Setting	Value	What it does
Vegetationmap File world	`string` default `(empty)`	Path to a vegetation map: a square image of which plants may grow where, named by its legend (vegetationmap.txt, 'r,g,b,a: +Plant, -Plant' lines). Not used when Directory is set: its vegetationmap.png is used.
Paintmap File world	`string` default `(empty)`	Path to a paint map: a square image of the ground paint (the terrain mask colour), each colour as it is or as paintmap.txt maps it ('mask colour: image colour' lines). Not used when Directory is set: its paintmap.png is used.
Lavamap File world	`string` default `(empty)`	Path to a lava map: a square grey image of where the Ashlands ground is lava (white). Not used when Directory is set: its lavamap.png is used.
Mossmap File world	`string` default `(empty)`	Path to a moss map: a square grey image of the Mistlands ground cover. Not used when Directory is set: its mossmap.png is used.
Heatmap File world	`string` default `(empty)`	Path to a heat map: a square grey image of the Ashlands heat, which decides where the Ashlands' weather and lava and ship damage are (anything above black). Not used when Directory is set: its heatmap.png is used.
Heatmap Scale console `bc heat sc`, world	`float` default `10` range 0 to 100	Multiplies the heat map (white = this value). Most heat effects cap at 1; 10 covers the game's whole Ashlands ocean gradient
Location Exclusion Mask world	`string` default `(empty)`	Path to a location exclusion mask image (no_spawns_mask.png): white (or non-black) pixels prevent default random locations (crypts, dungeons, camps) from spawning in those areas. Keeps towns, roads, docks, and monuments clean while preserving natural location distribution elsewhere in the world. Not used when Directory is set: its no_spawns_mask.png is used.
Coldmap File world	`string` default `(empty)`	Path to a cold map: a square grey image of Deep North cold, which decides where Deep North weather, freezing, and calm sea are (anything above black). Not used when Directory is set: its coldmap.png is used.
Coldmap Scale console `bc cold sc`, world	`float` default `10` range 0 to 100	Multiplies the cold map (white = this value). Most cold effects cap at 1; 10 covers the full cold gradient

[07 BetterContinents.Misc]

Setting	Value	What it does
NexusID local, hidden in configuration managers	`int` default `446`	(no description in the config)
SelectedPreset local, hidden in configuration managers	`string` default `Disabled`	(no description in the config)

Setting	Value	What it does
Settings Transfer Rate live	one of Vanilla, KB256, KB384, KB512, KB768, MB1, MB1_5, MB3, Unlimited default `KB512`	How fast this machine (the host, or a dedicated server) may push a joining player's Better Continents settings and images over their Steam connection. Valheim pins every connection to about 150 KB/s, so a large world can take over a minute to join; Vanilla leaves that rate untouched; KB256, KB384, KB512, KB768, MB1, MB1_5 and MB3 set that one connection to that fixed rate for the transfer only, then restore Valheim's own; Unlimited sets 100 MB/s (the transfer itself moves about 4 MB/s at most). Steam sends at exactly the rate set, with no congestion control, so pick one the server's upload can carry. The server's value is used; a client's own value has no effect. No effect on PlayFab (crossplay) connections, whose send rate cannot be changed.
Compact Maps world	one of Auto, On, Off default `Auto`	Whether a new world keeps its maps compressed in memory, in tiles of 128 x 128 pixels that are decoded when the world first reads them (Map Memory), and saves and sends them compressed: a far smaller world file, and far less memory, for a world of large maps. Every map reads exactly as it would without. Auto (the default): a new world is compact when any of its maps is more than 8192 pixels across (as pictures such maps take gigabytes of memory, and hundreds of megabytes in the world file and in what every joining player is sent) or when its heightmap has fine heights (a heightmap-fine.png beside it), and keeps its maps as pictures otherwise. On: every new world is compact. Off: no new world is compact, not even one of 16384 pixel maps, and a heightmap-fine.png is not read. Read only when a world is made (From Config, bc_import, or a Directory's export.cfg) with the newest settings version; a world keeps it for good. Better Continents 0.9 cannot read a compact world, and would generate its terrain as vanilla. An older config's true and false read as On and Auto
Map Memory live	`int` default `512` range 64 to 65,536	For a world made with Compact Maps: how many megabytes this machine lets the decoded tiles of its maps take. Past this, the tiles read least recently are dropped, and decoded again when read again. 512 holds every tile of a world's maps up to about 8192 pixels across, so each is decoded once; the tiles of a world of 16384 pixel maps are more than that (a heightmap's are 537 MB decoded), and what players read across the whole world is decoded again when it was dropped; a server short of memory can use less (each player's surroundings take about a megabyte). No effect on any other world, which holds its maps decoded. Applies at once
Config Version local, hidden in configuration managers	`int` default `0`	Which of Migrate's one-time changes this file has had; 0 = none (a file written before the unifying refactor, or a new one).
Max Map Size live	`int` default `16,384` choices 4096, 8192, 16384	The largest map, in pixels across, that this machine reads from a picture file: 4096, 8192 or 16384. A bigger picture is refused before it is read, with a line in the log, and the world is made without that map. A world's own maps, saved with it, are never refused. Making a world from five 16384 pixel maps takes about 1.8 GB of memory (such a world is saved compact), so a machine short of memory can stop at 8192 or 4096. Applies the next time a map is read

Setting	Value	What it does
Wide Sectors live	one of Auto, On, Off default `Auto`	Whether a world that reaches past the game's own sectors (its World Size + Edge Size is past 16,350 m) gets a sector and save chunks of its own for every zone out to 65.5 km, where the game gives everything beyond 16.35 km one shared sector (every player who comes near any of it is sent all of it, and it is saved as one chunk). Auto (the default): a new world past 16,350 m is made with wide sectors, and an existing world keeps the sectors it has. On: as Auto, and an existing world past 16,350 m that still has the game's sectors is converted when it loads. The conversion goes one way: the game saves a copy of the old save first, and a Better Continents without wide sectors (or the game alone) must not save the world afterwards, or it writes objects twice. Off: a new world gets the game's sectors whatever its size (objects past 16.35 km then share one sector, past 22.1 km some of them collide with the game's portal file, and past 32.7 km the game saves each zone's spawner at 20 km); a world already made or saved with wide sectors stays wide. Read by the machine that runs the world (single player, the host or a dedicated server), when a world is made and when it loads; a joining player's game follows the world's own settings. A world of 16,350 m or less is never touched. Where Expand World Size's own save grid is wider than the game's (Expand World Size 1.44 and later, for a world past 16,384 m), that grid files the world's objects and Wide Sectors stays out of its way: a new world is not made wide, and a save made with wide sectors is not loaded while that grid is in use
Upgrade World Radius live	`bool` default `false`	Whether the Upgrade World mod, when it is installed, is told this world's size: its "World radius" setting (default 10500 m, the game's own world) is raised to cover this world's World Size + Edge Size, rounded up to a whole zone, so that its commands (zones_reset, zones_generate, locations_add, biomes_count, world_reset, upgrade) reach the outer ring of a world past 10,500 m. It is never lowered. Off (the default): Upgrade World's setting is left as it is, and the log says once, on a world it does not cover, that this setting exists. The value is written to upgrade_world.cfg by Upgrade World. Applies when a world loads

[08 BetterContinents.AltBiomes]

Setting	Value	What it does
Altbiomemap File world	`string` default `(empty)`	Path to an alt-biome map (altbiomemap.png). It plants Valheim 1.0 alt biomes with colours, the way the biome map plants biomes; the legend beside it (altbiomemap.txt, written with a default colour per alt biome when missing) says which colour plants what. Black and transparent pixels are left to the game.
Mode world	`string` default `Random` choices Random, PlantedOnly, Off	Random = the game's random alt-biome placement on unplanted land (tuned by the values below) plus every planted region; PlantedOnly = only planted regions; Off = no alt biomes at all, planted ones included

Setting	Value	What it does
Grid world	`string` default `WorldEdge` choices WorldEdge, Vanilla	WorldEdge = alt biomes and biome-based location candidates stop at the edge of the world (World Size + Edge Size) when it is smaller than vanilla's 10500 m; Vanilla = always sample the vanilla 10500 m disc. No effect on a world laid out to its World Size (made since 0.10 without Expand World Size), whose grid covers the world.
Fixed Seed world	`string` default `(empty)`	Seed for the game's random alt-biome placement. Empty = the world seed (vanilla). A number, or any text, gives the same random alt-biome layout for this map whatever the world seed is
Chance Multiplier world	`float` default `1` range 0 to 10	Multiplies every alt biome's random placement chance (vanilla 0.2, Dark Meadows 0.5)
Amount Multiplier world	`float` default `1` range 0 to 10	Multiplies every alt biome's minimum and maximum number of regions
Region Size Scale world	`float` default `1` range 0.1 to 50	Multiplies every alt biome's region size window (edge length). Vanilla only gives alt biomes to regions roughly 0.1-2.4 km across; hand-drawn maps with big regions usually need 2-10
Distance Scale world	`float` default `1` range 0 to 10	Multiplies every alt biome's minimum distance from the world centre (vanilla 500-2000 m) and its world bounds
Min Sector Thickness world	`float` default `0` range 0 to 20	Regions thinner than this (area / edge length, in 12 m cells) never get a random alt biome. 0 = vanilla. 2 filters the slivers an anti-aliased biome map leaves along its borders
Mean Sector Height world	`bool` default `false`	Measure a region's average height as the mean over the whole region instead of vanilla's (lowest + highest) / 2 of its border. Helps maps whose biome paint runs out into the sea: the border is then under water and vanilla's measure sinks below the 30 m every alt biome requires
Fix Neighbour Check world	`bool` default `false`	Use a corrected version of vanilla's require/not-neighbour test (broken in vanilla; no vanilla alt biome uses it, modded ones may)
Overrides world	`string` default `(empty)`	Per alt biome overrides of the game's random placement: 'Name: key=value, key=value; Other Name: key=value'. A name may contain * wildcards ('*Mistlands'); '*' alone applies to all. An exact name wins over a wildcard pattern, and a pattern over '*', field by field. Keys: enabled, chance, min, max, mindist, minedge, maxedge, minheight, maxheight, ignorebounds. 'Fortress Mountain: enabled=false' keeps the game from placing Fortress Mountain at random; planted Fortress Mountain still works

[09 BetterContinents.Export]

Setting	Value	What it does
Hud live	`bool` default `false`	Shows the world export HUD in game: a status box (Hud Hotkey) and a window (Window Hotkey) with an Export tab (the maps, their options, Start and Cancel), an Import tab (your exports, made into New World presets) and a Baking tab (bake what players built into the world, undo, unbake). Applies at once. On a client connected to a server that runs Better Continents 0.9.0 or later, the server's value is used instead
Allow Export live	`bool` default `true`	Whether players connected to this server may export the world (bc_export, and Start in the export HUD). The machine that runs the world always may: single player, the host, and a dedicated server's own console (where an admin's 'bc_export server' runs). Applies at once, also to connected players. On a client, the server's value is used instead of this one
Hud Hotkey live	a Unity key (None = no key) default `F9`	Shows or hides the export HUD's status box (with no Shift, Ctrl or Alt held). None = no key
Window Hotkey live	a Unity key (None = no key) default `F7`	Opens or closes the export, import and baking window, which frees the mouse while it is open (with no Shift, Ctrl or Alt held). None = no key
Default Size live	`int` default `4096` range 128 to 16,384	Pixels per side of an export when bc_export is given no size, and the export window's first choice. 2048 and up keep the alt-biome map exact; 16384 is the largest, and every player joining a world built from the maps downloads them
Default Heightmap Amount live	`float` default `1` range 0.01 to 81	The Heightmap Amount an export encodes its heights for, unless bc_export or the window says otherwise; export.cfg records it, so the import always reads the heights at the amount they were written for. 1 is Better Continents' default everywhere: with Sea Level 0.5 it spans -30 m to 170 m and clips higher ground (the export says how many pixels); 2 spans -30 m to 370 m and keeps every vanilla mountain; the largest, 81, spans -30 m to 16,170 m in steps of a quarter of a metre
Default Sea Level live	`float` default `0.5` range 0 to 1	The Sea Level Adjustment an export encodes its heights for, unless bc_export or the window says otherwise
Largest Size live	`int` default `16,384` choices 4096, 8192, 16384	The largest export, in pixels a side, that this machine makes: the export window offers sizes up to it, bc_export refuses bigger ones, and a larger Default Size is used as this. 4096, 8192 or 16384. A 16384 px export takes a dedicated server about three minutes, and every player joining a world built from it downloads its maps

[10 BetterContinents.BakedPlacements]

Setting	Value	What it does
Draw Distance live	`int` default `4` range 1 to 8	How many zones (64 m each) around the camera baked buildings are drawn: 4 is a block of 9 x 9 zones, which is what a town's buildings reach. Applies at once
Draw Scale live	`float` default `1.5` range 0.5 to 4	How far baked pieces are drawn, as a multiple of where the game culls the same piece (which follows your level of detail setting: about 140 m for a stone wall on High). 1 is the game's own; more shows more of a town from afar, at the cost of frame rate. Applies at once

Setting	Value	What it does
Detail Scale live	`float` default `1` range 0.5 to 4	How far baked pieces keep their full detail, as a multiple of where the game switches the same piece to its simpler model (about 30 m for a stone wall on High). Applies at once
Shadows live	one of Near, All, Off default `All`	Which baked pieces cast shadows. Near: the pieces in full detail cast them and the simpler models farther out do not. All: both cast, as the game's own pieces do. Off: none cast. Every baked piece receives shadows. Applies at once
Light Distance live	`float` default `90` range 0 to 200	How far from you (metres) a baked world's wall torches, braziers, fire pits and candles can burn: within it the Light Count nearest are each a real lit copy of its piece; every other one, near or far, is drawn unlit like any baked piece. 0 turns them all off (every one is drawn unlit). Applies at once
Seat Distance live	`float` default `12` range 0 to 40	How far from you (metres) a baked world's chairs, benches, stools and thrones are real, so that you can sit on them: within it each has its seat made, a few per frame; farther out it is only drawn. 0 turns the seats off. Applies at once
Light Count live	`int` default `24` range 0 to 128	How many of a baked world's wall torches, braziers, fire pits and candles burn at once: the nearest to you within Light Distance, each a real lit copy of its piece with its flames, light and sound; every other one is drawn unlit like any baked piece, so none ever vanishes in the distance. Each burning light costs frame rate (a street of them can halve it), and fewer is faster. 0 = none burns. Applies at once
Light Shadows live	`int` default `4` range 0 to 32	How many of a baked world's lit copies (see Light Distance) cast shadows: the nearest to you, which keep their lights' shadows as the piece has them (the game's own point light shadow limit still applies to them); every other copy's light casts none. A shadow from a point light costs memory (12 to 48 MB each), and a graphics mod that allows every light shadows would otherwise give every torch near you one. 0 = no copy casts any. Applies at once
Tints live	`string` default `(empty)`	Override the colours that a baked layer tints its pieces with: name=r,g,b pairs separated by semicolons, by the tint's name in the layer (a town's clay walls may be plaster, brick, sandstone and marble), for example "plaster=1.65,1.58,1.48; brick=1.25,0.72,0.56". Each colour multiplies the clay's grey texture (1 keeps it). Empty: the layer's own colours. Applies at once
Diagnostics live	`bool` default `false`	Writes, once for each kind of baked piece, the materials it is drawn with (shader, texture, colour) to the log

[11 BetterContinents.Water]

Setting	Value	What it does
Own Water live	`bool` default `true`	Draws the sea with Better Continents' own water in worlds whose sea is not the game's: waves out to the world's edge, calm only in its own Deep North, and the Ashlands' red where its hot sea is. Off: the game's water, with the red set zone by zone as earlier versions of Better Continents did; boats still float on the world's waves. If this graphics device cannot run Better Continents' water, or another mod has replaced the game's water shader, the game's water is drawn too, and the log says why. A vanilla-size world without maps, and a world that keeps the game's own terrain, always have the game's water. Applies at once
Soft Far Sea live	`bool` default `true`	With Own Water: the far sea past the loaded zones takes the depth of the sea under it, so that it looks like the near water where they meet (no light and dark strips seen from high up). Off: the far sea is drawn as deep sea everywhere, as the game draws it. Applies at once

[12 BetterContinents.Lakes]

Setting	Value	What it does
Lake Depth Scale world	`float` default `0.5` range 0 to 3	Experimental (lakes come from the Tarn mod). How deep the game's water shader takes a lake to be, as a share of its real depth, for the lakes of a new world: the game's water colour, foam and wave size all follow depth (full colour and full waves at 10 m), so lower is calmer and greener and higher bluer and rougher. It also sizes the waves that boats and swimmers feel in a lake, so every player of a world shares the one value, which is kept with the world's lakes; 'bc_lake depth' changes it in a running world (the lake set is swapped and sent to the players; no zone is regenerated). 0.5 is the value the lakes of Raven Iron's Tarn were tuned with. A new world takes this when it is made (From Config, bc_import, or a Directory's export.cfg) and keeps it
Lake Build Range live	`float` default `400` range 150 to 2000	Experimental (lakes come from the Tarn mod). Lakes within this many metres of you are built (their water and the places you can swim in it), and dropped again 150 m further out. Applies at once
Lake Build Budget live	`float` default `2` range 0.5 to 20	Experimental (lakes come from the Tarn mod). Milliseconds a frame that this machine spends making the water objects of lakes. Applies at once
Lake Sound live	`float` default `0.35` range 0 to 1	Experimental (lakes come from the Tarn mod). Volume of the water sound near a lake (the game's ocean loop, quieter). 0 = off. Applies at once
Lakes On Map live	`bool` default `true`	Experimental (lakes come from the Tarn mod). Draws a world's lakes on the large map and the minimap. The map is drawn again when the lakes change; a lake taken out stays on the picture until it is. Applies at once

The image maps

Every kind of image map, declared once in `BetterContinents.MapKinds.cs`. A **Directory** loads each map by its standard file name (and its legend, `<image name>.txt`, beside it); where the Directory is empty the map's own file setting names it. A world's maps are read once, when the world is created. In a loaded world `bc <group> fn <path>` sets a map's file (empty switches the map off) and `bc reload <command>` reads the picture and its legend again; `bc reload all` reloads every map the world has.

Map	File in a Directory	Config key	Console file	Reload
Heightmap	`heightmap.png`	[02 BetterContinents.Heightmap] Heightmap File	`bc h fn`	`bc reload hm`
Biomemap	`biomemap.png`	[03 BetterContinents.Biomemap] Biomemap File	`bc b fn`	`bc reload bm`
Terrainmap	`terrainmap.png`	[03 BetterContinents.Biomemap] Terrainmap file	`bc terrain fn`	`bc reload terrain`
Locationmap	`locationmap.png`	[06 BetterContinents.Maps] Locationmap File	`bc l fn`	`bc reload lm`
Roughmap	`roughmap.png`	[02 BetterContinents.Heightmap] Roughmap File	`bc r fn`	`bc reload rm`
Flatmap	`flatmap.png`	(none: only a world of an older version has it)		`bc reload fm`
Forestmap	`forestmap.png`	[04 BetterContinents.Forest] Forestmap File	`bc fo fn`	`bc reload fom`
Heatmap	`heatmap.png`	[06 BetterContinents.Maps] Heatmap File	`bc heat fn`	`bc reload heat`
Coldmap	`coldmap.png`	[06 BetterContinents.Maps] Coldmap File	`bc cold fn`	`bc reload cold`
Paintmap	`paintmap.png`	[06 BetterContinents.Maps] Paintmap File	`bc paint fn`	`bc reload paint`
Lavamap	`lavamap.png`	[06 BetterContinents.Maps] Lavamap File	`bc lava fn`	`bc reload lava`
Mossmap	`mossmap.png`	[06 BetterContinents.Maps] Mossmap File	`bc moss fn`	`bc reload moss`
Vegetationmap	`vegetationmap.png`	[06 BetterContinents.Maps] Vegetationmap File	`bc vegetation fn`	`bc reload vegetation`
Spawnmap	`spawnmap.png`	[06 BetterContinents.Maps] Spawnmap File	`bc spawn fn`	`bc reload spawn`
Altbiomemap	`altbiomemap.png`	[08 BetterContinents.AltBiomes] Altbiomemap File	`bc ab fn`	`bc reload ab`

The order a new world reads its maps in: Heightmap, Biomemap, Locationmap, Roughmap, Forestmap, Terrainmap, Paintmap, Lavamap, Mossmap, Vegetationmap, Heatmap, Coldmap, Spawnmap, Altbiomemap.

How each legend reads

Six kinds of legend file are read in six ways, and every way is frozen: a legend file must load exactly as it did in earlier versions, quirks included. This is the mod's own account of them (the header comment of `Legends.cs`).

- **Biome** (`biomemap.txt`). The file's lines joined with '|' (so a '|' splits a line too). An entry is "name: colour" with exactly one ':'. Blank lines and '#' comments (after any byte order mark or zero-width space) are skipped. Strict colours. A bad entry is logged, counted and skipped: a vanilla biome with a bad colour keeps its default one, and an unknown name's colour reads as None. A legend with no entry gives the default colours.
- **Location** (`locationmap.txt`). As biome, but the name is everything before the last ':' (Expand World Data's "Name:Alias"), and a name listed twice keeps its last colour.
- **Paint** (`paintmap.txt`). The lines joined with '|'. An entry is "mask: image colour" with exactly one ':'; anything else is skipped without a word (there are no comments: a '#' line with one ':' is an entry). Lenient colours. The first entry for an image colour wins. A broken number throws: read from the file, the whole legend is lost (logged); read from a world's settings, the exception reaches the world's loading.
- **Terrain** (`terrainmap.txt`). As paint, but the mask is a ground name (lower-cased in the player's culture) or a colour, and an empty legend is the default one.
- **Spawn and vegetation** (`spawnmap.txt`, `vegetationmap.txt`). The file's lines. An entry is "colour: entries" with exactly one ':'; blank lines, '#' lines (culture-aware) and lines without exactly one ':' are skipped. Lenient colours; a broken number ends the legend at that line (logged). White is always the first entry, "none". A missing legend is written empty.
- **Alt biome** (`altbiomemap.txt`). The file's text, at "\n" (and "\r\n"). An entry is "names: colour" or "names: at x, z", at the first ':'; '#' starts a comment at the start of a line, or alone after a space. Alt-biome colours. A bad line is logged with its number and skipped; black is reserved for unplanted land; a colour listed again adds its names.

Strict colours (TryParseColor32): a hex colour (ImageSharp's: #RGB, RRGGBB, RRGGBBAA...) or 3 or 4 numbers from 0 to 255, read invariantly; anything else is not a colour.

Lenient colours (ParseRGBA): the same hex, else a warning and transparent black, or 3 or 4 numbers read with byte.Parse in the player's culture, which throws on anything else.

Alt-biome colours (TryParseAltBiomeColour): RRGGBB or RRGGBBAA with an optional '#', or 3 or 4 numbers.

The default biome legend

What `biomemap.txt` holds when it is missing (`ImageMapBiome.DefaultColors`):

Biome	Colour	Biome	Colour
None	`000000`	Plains	`FFFF00`
Meadows	`00FF00`	Mistlands	`7F7F7F`
BlackForest	`007F00`	AshLands	`FF0000`
Swamp	`7F7F00`	DeepNorth	`00FFFF`
Mountain	`FFFFFF`	Ocean	`0000FF`

The bc console

The commands of the bc tree, which the host of a world can use with cheats on (bc help). Each setting with a console entry above is a value of its group; the tree also has these commands and groups.

Command	What it is
bc info	Prints current settings to console
bc reload	Reloads and reapplies one or more of the image maps
bc locs	Prints all location spawn instance counts to the console
bc bosses	Shows pins for bosses, start temple and trader
bc show	Pins locations matching optional filter on the map
bc hide	Removes the pins 'bc show' and 'bc bosses' put on the map, those matching the optional filter or, with none, all of them; your own pins stay
bc clouds	Toggles minimap clouds
bc mapds	Sets minimap downscaling factor (for faster updates)
bc reset	Redraws the minimap, rebuilds the alt-biome grid and the loaded terrain, and regenerates the zones (as every change does)
bc regen	Regenerates the zones now, the way every change here does: every generated zone is reset and generates again with the current settings when somebody comes near, except zones within one zone of something a player built or worked on (a field, a path, levelled ground), a tombstone, a player connected from another machine (as far as their game keeps zones loaded around them), or you inside a dungeon. A location that reaches into one of those zones is kept whole. Players, what players built, tombstones and tamed animals are never removed, and the loaded terrain is left as it is. A console command in Debug Reset Command runs instead. 'bc reset' does this after redrawing the minimap and rebuilding the alt-biome grid and the loaded terrain. Words after it reset only part of the world, and every rule above still holds: 'dry' counts the zones that would be reset, by biome, and changes nothing; 'biome=<names>' (comma between: Meadows,BlackForest) the zones where any corner is one of those biomes; 'min=<m>' and 'max=<m>' (or 'radius=<m>') the zones that far from 'pos=<x>,<z>' (the world's centre when there is none); 'zone=<x>,<z>' one zone; 'chance=<0..1>' each of those zones with that chance. For example 'bc regen dry biome=AshLands min=3000', then the same without 'dry'
bc scr	Saves the minimap to a png, optionally pass resolution, default is 2048
bc savepreset	Saves current world settings as a preset, including a thumbnail, pass preset name as argument (the world's name if none)
bc g	Global settings, get more info with 'bc g help'
bc h	Heightmap settings, get more info with 'bc h help'
bc r	Roughmap settings, get more info with 'bc r help'
bc b	Biomemap settings, get more info with 'bc b help'
bc terrain	Terrainmap settings, get more info with 'bc terrain help'
bc l	Locationmap settings, get more info with 'bc l help'
bc paint	Paintmap settings, get more info with 'bc paint help'
bc lava	Lavamap settings, get more info with 'bc lava help'
bc moss	Mossmap settings, get more info with 'bc moss help'
bc vegetation	Vegetationmap settings, get more info with 'bc vegetation help'
bc spawn	Spawnmap settings, get more info with 'bc spawn help'
bc heat	Heatmap settings, get more info with 'bc heat help'
bc cold	Coldmap settings, get more info with 'bc cold help'
bc fo	Forest settings, get more info with 'bc fo help'
bc st	Start position settings, get more info with 'bc st help'
bc hl	Height layer settings
bc ab	Alt-biome report, planting and per-world controls, get more info with 'bc ab help'

Command	What it is
`bc bake`	Bake pieces into the world's layer, take them out again, undo
`bc export`	Dumps the loaded world into Better Continents maps that rebuild it, get more info with 'bc export help'
`bc import`	Makes a New World preset (or the config) from an export folder, get more info with 'bc import help'
`bc cache`	A shareable world cache, for pre-seeding clients of big worlds
`bc twins`	Two copies of one vegetation prefab on one spot, found and removed
`bc genloc`	Makes the world's locations again: every location that is not placed yet is cleared and rolled again in the zones that are not generated yet; placed ones and generated zones stay. Does what genloc does also when the game's own run is stuck or the world was loaded with its locations marked as made, says what it found and what changed, and sends the new icons to the clients. 'bc genloc alt' passes to the game's alt-biome branch exactly as 'genloc alt' does. 'bc genloc pins' only puts the location map's pins and the start position into the registry and generates nothing. 'bc genloc help' says more
`bc lake`	The world's lakes above the sea and its mask of ground kept clear of plants, from Raven Iron's Tarn
`bc water`	Whether Better Continents' own sea is on and why, its field, and what draws the water; force it on, probe drawn against felt

`bc reset` and `bc regen` (chapter 6): `bc reset` redraws the minimap (and reveals all of it) and rebuilds the alt-biome grid, and when the new grid is in place it rebuilds the terrain that is loaded (the distant terrain and the grass too) and does what `bc regen` does. That is: run the console command in `Debug Reset Command` if one is set and, when the setting is empty, start the zone regeneration. `bc regen` on its own leaves the loaded terrain as it is, since nothing changed. Most changes made through the tree run all of `bc reset`, and the terrain rebuild stalls the game for a second or two; a change to Biome precision and the `bc ab` edits skip the minimap.

The commands the game's console registers, as the mod declares them. `bc` is the root of the tree above and a cheat for whoever runs the world (devcommands on, host or single player only: the Server devcommands mod lets a joined player type it on their own machine too, where a change alters only that player's copy of the settings, chapter 6). The others are commands of their own for every player, not only the host; they report to the console that asked, and to the admin who ran them on a dedicated server (`bc_export server ...`, `bc_twins server ...`).

Command	Who	Its own help line
`bc`	the host, with cheats on	Root Better Continents command
`bc_altbiomes`	any player	[here | names [filter] | hash | list [filter]] - Better Continents alt biomes: summary, the region you stand in, alt-biome names and colours, agreement hashes, every alt-biome region (list needs devcommands on a client)
`bc_bake`	any player	[help | info | list | area ... | box ... | unbake ... | undo | load | drop ... | server ...] - Better Continents: bake pieces into the world's layer, take them out again, undo (bc_bake help)
`bc_cache`	any player	[export | import <file or folder> | list | help] - Better Continents: a shareable world cache, for pre-seeding clients of big worlds (bc_cache help)
`bc_export`	any player	[status | cancel | server ... | size and options] - Better Continents: export the loaded world as a map set that rebuilds it (bc_export help)
`bc_import`	any player	[list | <number> | <folder> | <world>] [config] - Better Continents: make a New World preset (or the config) from an export folder (bc_import help)
`bc_lake`	any player	[help | info | list | here | at | remove | depth | load | import | export | mask ... | regen | server ...] - Better Continents: the world's lakes above the sea and its mask of ground kept clear of plants (bc_lake help)

Command	Who	Its own help line
bc_twins	any player	[radius | remove [confirm] [radius] | server ... | help] - Better Continents: find and remove vegetation twins, two copies of one prefab on one spot (bc_twins help)
bc_water	any player	[force on|off | probe] - Better Continents: whether its own sea is on in this world and why, the sea field, and what draws the water; with Debug Mode: force it on, probe the drawn waves against the felt

The usage line each command prints for `<command> help`:

Command	Usage
bc_altbiomes	bc_altbiomes [here | names [filter] | hash | list [filter]]
bc_bake	bc_bake help | info | stats | hide | show | list | check [all | <zx> <zz>] | area <radius> [at <x> <z>] [town] [any] [convert] [confirm] | box <x1> <z1> <x2> <z2> [town] [any] [convert] [confirm] | unbake <radius> [at <x> <z>] | box <x1> <z1> <x2> <z2> | world [all] [confirm] | undo [<n>] [confirm] | load [<file>] [convert] [confirm] | export [<file>] | orphans [destroy] [items] [confirm] | drop compiler | bake <n> [confirm] | drop <world> [confirm] | server <the same>
bc_export	bc_export [status | cancel | server <options> | <options>] - options: [size] [amount=1] [sealevel=0.5] [heatscale=10] [edge=auto|on|off] [forest=additive|exact] [fine=auto|off|4|8] [noheight] [nobiomes] [nolocations] [noforest] [noheat] [nocold] [noaltbiomes] [nolava] [nomoss] [nopaint] [nosources] [nopreset] [perpoint]
bc_cache	bc_cache [export | import <file or folder> | list | help]
bc_import	bc_import [list | status | help | edited | <number> | <folder> | <world>] [config]
bc_water	bc_water [force on|off | probe]
bc_lake	bc_lake [help | info | list [n] | here [radius] [margin] | at <x> <z> [radius] [margin] | remove <name|all> | depth [scale] | load [folder] | import [path] | export [folder] | mask at <x> <z> | mask load <file> | mask export <file> | regen | server ...] (load, import, remove all, regen, and writing over a file need 'confirm')
bc_twins	bc_twins [radius] | remove [radius] | remove confirm [radius] | server <the same> | help

Appendix C

The game's own numbers

Generated from the extracted game data (`locations/locations_1.0.15.json`, `altbiomes/altbiomes_1.0.15_full.json`) and the mod's source by `make_c_data.py`. Regenerate it after re-extracting for a new game build, or when the mod's defaults change.

Every enabled location, in the game's placement order

Prioritized locations first (the `_ZoneSystem` prefab's own list, then the add-on lists), then the rest. Area is the biome-area rule (Median, Edge or Any); altitude is metres above the water; delta is the terrain delta window; centre is the distance from the world centre in metres, in vanilla's numbers: the window of `m_minDistance` and `m_maxDistance`, which a new world laid out to its own World Size scales by World Size / 10,000 (chapter 10), and, marked fixed, the window of `m_minDistanceFromCenter` and `m_maxDistanceFromCenter`, which nothing scales (chapter 3); sim is the minimum distance from similar locations, with the group's name where the rule is by group; veg is the vegetation (lava mask) window where it is not 0..1. Names are the prefab asset names, which the game log uses.

Location	Biome	Area	N	Altitude	Delta	Centre	Sim	Notes
StartTemple	Meadows	Median	1	3..1000	0..3	0..10000		prioritized, centre first, forest 1..5
Eikthyrnir	Meadows	Median	3	1..1000	0..3	0..1000		prioritized
GoblinKing	Plains	Median	4	1..1000	0..4		3000	prioritized
GDKing	BlackForest	Median	4	1..1000	0..5	1000..7000	3000	prioritized
Bonemass	Swamp	Median	5	0..2	0..4	2000..10000	3000	prioritized
SunkenCrypt4	Swamp	Median	175	0..2	0..4		64 (SunkenCrypt)	prioritized
Vendor_BlackForest	BlackForest	Median	10	1..1000	0..2	1500+	512	prioritized, unique
Dragonqueen	Mountain	Median	3	150..500	0..4	0..8000	3000	prioritized
DN_Bossroom	DeepNorth	Any	3	80..5000	0..6		1024 (dn_boss)	prioritized
FaderLocation	AshLands	Median	3	1..300	0..40		2048 (FaderBoss)	prioritized, veg 0..0.1
PlaceofMystery1	AshLands	Median	1	0..1000	0..5		2048 (PlaceofMystery)	prioritized, unique, veg 0..0.1
PlaceofMystery2	AshLands	Median	1	0..1000	0..5		2048 (PlaceofMystery)	prioritized, unique, veg 0..0.1
PlaceofMystery3	AshLands	Median	1	0..1000	0..5		2048 (PlaceofMystery)	prioritized, unique, veg 0..0.1
CharredFortress	AshLands	Any	20	20..1000	0..10		256 (FaderBoss)	prioritized, veg 0..0.1, not near group zigg
BogWitch_Camp	Swamp	Median	10	1..1000	0..2	3000..8000	1000	prioritized, unique
Mistlands_DvergrTownEntrance1	Mistlands	Median	120	12..1000	5..40		256 (DvergrDungeon)	prioritized
Mistlands_DvergrTownEntrance2	Mistlands	Median	120	1..20	0..40		256 (DvergrDungeon)	prioritized
Mistlands_DvergrBossEntrance1	Mistlands	Median	5	1..20	0..40		2048 (DvergrBoss)	prioritized
AncientUpgradeStation	Mountain	Median	10	0..5000	0..40	200+, fixed 500+	400 (AncientUpgradeStation)	prioritized, unique
Hildir_cave	Mountain	Median	3	200..5000	0..40	1000..8000	2000	prioritized
Hildir_camp	Meadows	Any	10	1..1000	0..2	3000..8000	1000	prioritized, unique
Hildir_crypt	BlackForest	Median	3	1..1000	0..2	3000..8000	3000	prioritized

Location	Biome	Area	N	Altitude	Delta	Centre	Sim	Notes
Hildir_plainsfortress	Plains	Median	3	8..1000	0..4		3000	prioritized
StoneCircle	Meadows	Any	25	1..1000	0..3		200	
Greydwarf_camp1	BlackForest	Median	300	1..1000	0..3		128	
Runestone_Greydwarfs	BlackForest	Any	25	1..1000	0..3	0..2000	128 (Runestones)	
Grave1	Swamp	Median	200	0.5..1000	0..3			
SwampRuin1	Swamp	Median	30	-0.5..1000	0..3		256 (SwampRuin)	
SwampRuin2	Swamp	Median	30	-0.5..1000	0..3		256 (SwampRuin)	
FireHole	Swamp	Median	75	-1..1000	0..5		16 (FireHole)	
Runestone_Draugr	Swamp	Any	50	0.5..1000	0..3		128 (Runestones)	
Crypt2	BlackForest	Any	200	1..1000	0..2		128	
Ruin1	BlackForest	Any	200	1..1000	0..3			
Ruin2	BlackForest	Any	200	1..1000	0..3			
StoneHouse1_heath	Plains	Any	0	1..1000	0..3			
StoneHouse2_heath	Plains	Any	0	1..1000	0..3			
StoneHouse5_heath	Plains	Any	0	1..1000	0..3			
StoneHouse3	BlackForest	Any	200	1..1000	0..3			
StoneHouse4	BlackForest	Any	200	1..1000	0..3			
Ruin3	Plains	Any	50	1..1000	0..3		512 (Goblintower)	
GoblinCamp2	Plains	Any	200	2..1000	0..3		250	
StoneTower1	Plains	Any	50	1..1000	0..3		512 (Goblintower)	
StoneTower3	Plains	Any	50	1..1000	0..3		512 (Goblintower)	
StoneHenge1	Plains	Any	5	5..1000	0..3		1000 (Stonehenge)	
StoneHenge2	Plains	Any	5	5..1000	0..3		1000 (Stonehenge)	
StoneHenge3	Plains	Any	5	5..1000	0..3		1000 (Stonehenge)	
StoneHenge4	Plains	Any	5	5..1000	0..3		1000 (Stonehenge)	
StoneHenge5	Plains	Any	20	2..1000	0..3		500 (Stonehenge)	
StoneHenge6	Plains	Any	20	2..1000	0..3		500 (Stonehenge)	
WoodHouse1	Meadows	Any	20	1..1000	0..4			
WoodHouse2	Meadows	Any	20	1..1000	0..4			
WoodHouse3	Meadows	Any	20	1..1000	0..4			
WoodHouse4	Meadows	Any	20	1..1000	0..4			
WoodHouse5	Meadows	Any	20	1..1000	0..4			
WoodHouse6	Meadows	Any	20	1..1000	0..4			
WoodHouse7	Meadows	Any	20	1..1000	0..4			
WoodHouse8	Meadows	Any	20	1..1000	0..4			
WoodHouse9	Meadows	Any	20	1..1000	0..4			
WoodHouse10	Meadows	Any	20	1..1000	0..4			
WoodHouse11	Meadows	Any	20	1..1000	0..4			
WoodHouse12	Meadows	Any	20	1..1000	0..4			
WoodHouse13	Meadows	Any	20	1..1000	0..4			
WoodFarm1	Meadows	Any	10	1..1000	0..4	500..2000	128 (woodvillage)	
WoodVillage1	Meadows	Any	15	1..1000	0..4	2000..10000	256 (woodvillage)	
TrollCave02	BlackForest	Median	200	3..1000	5..10		256	
Dolmen01	Meadows, BlackForest	Any	100	1..1000	0..2			
Dolmen02	Meadows, BlackForest	Any	100	1..1000	0..2			
Dolmen03	Meadows, BlackForest	Any	50	1..1000	0..2			
Crypt3	BlackForest	Any	200	3..1000	0..2		128	
Crypt4	BlackForest	Any	200	1..1000	0..2		128	
InfestedTree01	Swamp	Any	700	-1..1000	0..3			
SwampHut1	Swamp	Any	50	-2..1000	0..3		128 (Swamphut)	

Location	Biome	Area	N	Altitude	Delta	Centre	Sim	Notes
SwampHut2	Swamp	Any	50	2..1000	0..3		128 (Swamphut)	
SwampHut3	Swamp	Any	50	2..1000	0..3		128 (Swamphut)	
SwampHut4	Swamp	Any	50	-1..1000	0..3		128 (Swamphut)	
SwampHut5	Swamp	Any	25	-1..1000	0..3		128 (Swamphut)	
SwampWell1	Swamp	Any	25	-1..1000	0..3		1024	
StoneTowerRuins04	Mountain	Any	50	150..1000	6..40		128 (Mountainruin)	
StoneTowerRuins05	Mountain	Any	50	150..1000	6..40		128 (Mountainruin)	
StoneTowerRuins03	BlackForest	Any	80	2..1000	0..2		200 (Stonetowerruins)	
StoneTowerRuins07	BlackForest	Any	80	2..1000	0..2		200 (Stonetowerruins)	
StoneTowerRuins08	BlackForest	Any	80	2..1000	0..2		200 (Stonetowerruins)	
StoneTowerRuins09	BlackForest	Any	80	2..1000	0..2		200 (Stonetowerruins)	
StoneTowerRuins10	BlackForest	Any	80	2..1000	0..2		200 (Stonetowerruins)	
ShipSetting01	Meadows	Any	100	1..1000	0..3		128	
DrakeNest01	Mountain	Any	200	100..2000	0..3	0..10000	100	
Waymarker01	Mountain	Any	50	100..1000	0..2			
Waymarker02	Mountain	Any	50	100..1000	0..2			
AbandonedLogCabin02	Mountain	Any	33	100..1000	0..3		128 (Abandonedcabin)	
AbandonedLogCabin03	Mountain	Any	33	100..1000	0..3		128 (Abandonedcabin)	
AbandonedLogCabin04	Mountain	Any	50	100..1000	0..3		128 (Abandonedcabin)	
MountainGrave01	Mountain	Any	100	100..1000	0..2		50	
DrakeLorestone	Mountain	Any	50	100..1000	0..3		128 (Runestones)	
MountainWell1	Mountain	Any	25	100..1000	0..3		256	
ShipWreck01	Swamp, BlackForest, Plains, Ocean	Any	25	-1..1	0..10		1024 (Shipwreck)	
ShipWreck02	Swamp, BlackForest, Plains, Ocean	Any	25	-1..1	0..10		1024 (Shipwreck)	
ShipWreck03	Swamp, BlackForest, Plains, Ocean	Any	25	-1..1	0..10		1024 (Shipwreck)	
ShipWreck04	Swamp, BlackForest, Plains, Ocean	Any	25	-1..1	0..10		1024 (Shipwreck)	
Runestone_Meadows	Meadows	Any	100	1..1000	0..3		128 (Runestones)	
Runestone_Boars	Meadows	Any	50	1..1000	0..3		128 (Runestones)	
Runestone_Swamps	Swamp	Any	100	0..1000	0..3		128 (Runestones)	
Runestone_Mountains	Mountain	Any	100	100..1000	0..3		128 (Runestones)	
Runestone_BlackForest	BlackForest	Any	50	1..1000	0..3		128 (Runestones)	
Runestone_Plains	Plains	Any	100	1..1000	0..3		128 (Runestones)	
CombatRuin01	Meadows	Any	5	5..1000	0..1.5	1500..10000, fixed 1500+	2000	forest 0.5..1
BigRockClearing	BlackForest	Median	10	5..1000	0..2	1000..10000, fixed 1000+	2000	unique, forest 0.8..1
TheHole01	DeepNorth	Median	40	2..1000	0..3		256	not near group thehole
NorthVillage	DeepNorth	Median	135	2..1000	0..3		70, at most 85	not near group thehole
MorkBorg	DeepNorth	Any	40	30..10000	0..300		275 (morkborg)	not near group 0
NorthMemorialPlace	DeepNorth	Median	15	2..1000	0..3		400 (memorialplace)	not near group 0
FrozenShip01_DN	DeepNorth	Any	50	-15..-5	0..100	fixed 8000..9750	64 (FrozenShip)	snaps to water
FrozenShip02_DN	DeepNorth	Any	50	-15..-5	0..100	fixed 8000..9750	64 (FrozenShip)	snaps to water

Location	Biome	Area	N	Altitude	Delta	Centre	Sim	Notes
FrozenShip03_DN	DeepNorth	Any	50	-15..-5	0..100	fixed 8000..9750	64 (FrozenShip)	snaps to water
IcePond1	DeepNorth	Median	40	10..200	0..1.5		128 (icepond)	forest 0..1
DN_hut01	DeepNorth	Median	40	40..1000	0..4		100 (northvillage)	not near group 0
ShipSetting02	DeepNorth	Any	100	2..1000	0..2		64 (shipsetting)	
ShipSetting03	DeepNorth	Any	50	2..1000	0..2		64 (shipsetting)	
LumberCamp	DeepNorth	Median	50	4..1000	0..3		100 (thehole)	forest 0..1
ShipWreck01_DN	DeepNorth	Any	170	-0.5..0.5	0..2			
ShipWreck02_DN	DeepNorth	Any	120	-0.5..1	0..2			
Runestone_DeepNorth	DeepNorth	Any	70	0..1000	0..10		128 (Runestones)	veg 0..0.1
DN_gammeltrollFrac01	DeepNorth	Any	30	5..1000	0..10			
DN_gammeltrollFrac02	DeepNorth	Any	30	10..1000	0..10			
BearCave	BlackForest	Median	50	5..1000	5..10		256	
FortressRuins	AshLands	Any	100	0..1000	0..10		at most 70	veg 0..0.8, not near group zigg
CharredRuins1	AshLands	Median	75	-5..1000	0..6		256 (zigg)	
Runestone_Ashlands	AshLands	Any	70	0..1000	0..10		128 (Runestones)	veg 0..0.1
LeviathanLava	AshLands	Any	100	10..1000	0..10			veg 0.6..1
SulfurArch	AshLands	Median	100	10..1000	0..6		40	veg 0..0.8
CharredStone_Spawner	AshLands	Any	300	0..1000	0..10		100	veg 0..0.2
VoltureNest	AshLands	Any	350	0..1000	0..5		100	veg 0..0.1
CharredTowerRuins1	AshLands	Any	30	0..1000	0..4		100 (towerruins)	veg 0..0.5
CharredTowerRuins1_dvergr	AshLands	Any	30	0..1000	0..4		100 (towerruins)	veg 0..0.1
CharredTowerRuins2	AshLands	Any	40	0..1000	0..4			veg 0..0.3
CharredTowerRuins3	AshLands	Any	30	0..1000	0..4		500	veg 0..0.3
AshlandRuins	AshLands	Median	100	0..1000	0..4		16	veg 0..0.2
MorgenHole1	AshLands	Median	40	0..1000	0..10		300 (MorgenHole)	veg 0..0.3
MorgenHole2	AshLands	Median	40	0..1000	0..10		200 (MorgenHole)	veg 0..0.3
MorgenHole3	AshLands	Median	40	0..1000	0..10		200 (MorgenHole)	veg 0..0.3
CharredRuins2	AshLands	Median	100	-5..1000	0..30		128	
CharredRuins3	AshLands	Median	100	-5..1000	0..30		128	
CharredRuins4	AshLands	Median	100	-5..1000	0..30		128	
Mistlands_GuardTower1_new	Mistlands	Median	75	2..1000	0..10		128 (Dvergr)	
Mistlands_GuardTower1_ruined_new	Mistlands	Median	80	2..1000	0..10		128 (Dvergr)	
Mistlands_GuardTower1_ruined_new2	Mistlands	Median	20	2..1000	0..10		128 (Dvergr)	
Mistlands_GuardTower2_new	Mistlands	Median	75	4..1000	0..10		128 (Dvergr)	
Mistlands_GuardTower3_new	Mistlands	Median	50	12..1000	0..10		128 (Dvergr)	
Mistlands_GuardTower3_ruined_new	Mistlands	Median	50	11..1000	0..10		128 (Dvergr)	
Mistlands_Lighthouse1_new	Mistlands	Median	100	1..20	0..40		128 (Dvergr)	
Mistlands_Excavation1	Mistlands	Median	40	4..100	0..4		128 (Excavation)	
Mistlands_Excavation2	Mistlands	Median	40	4..100	0..4		128 (Excavation)	
Mistlands_Excavation3	Mistlands	Median	40	4..100	0..5		96 (Excavation)	
Mistlands_Harbour1	Mistlands	Edge	100	-1..-0.25	2..20		64 (Harbour)	snaps to water
Mistlands_Viaduct1	Mistlands	Median	100	-1..15	2..40		128 (Harbour)	
Mistlands_Viaduct2	Mistlands	Median	150	-1..25	4..40		64 (Harbour)	
Mistlands_RockSpire1	Mistlands	Median	200	-10..1000	2..99		60	
Mistlands_Giant1	Mistlands	Median	250	-1..1000	0..3		180 (Giant)	
Mistlands_Giant2	Mistlands	Median	85	-1..1000	0..4		256 (Giant)	
Mistlands_RoadPost1	Mistlands	Any	500	1..1000	0..10			
Mistlands_StatueGroup1	Mistlands	Any	200	2..1000	0..10		32	
Mistlands_Statue1	Mistlands	Any	200	2..1000	0..10			

Location	Biome	Area	N	Altitude	Delta	Centre	Sim	Notes
Mistlands_Statue2	Mistlands	Any	200	2..1000	0..10			
Mistlands_Swords1	Mistlands	Median	33	-1..1000	0..10		256 (GiantArmor)	
Mistlands_Swords2	Mistlands	Median	33	-1..1000	0..10		256 (GiantArmor)	
Mistlands_Swords3	Mistlands	Median	33	-1..1000	0..10		256 (GiantArmor)	
Runestone_Mistlands	Mistlands	Any	50	-1..1000	0..10		128 (Runestones)	
TarPit1	Plains	Median	100	5..60	0..1.5		128 (tarpit)	
TarPit2	Plains	Median	16	5..60	0..1.5		128 (tarpit)	
TarPit3	Plains	Median	100	5..60	0..1.5		128 (tarpit)	
MountainCave02	Mountain	Any	120	100..5000	0..40		200 (mountaincaves)	

170 location types. Disabled entries (the game ships some switched off) and the duplicate entries of the add-on lists are left out, as the game leaves them out.

Locations that ask for high ground, and what a heightmap needs

A heightmap pixel p (0 to 1) gives ground at altitude `(p * A - 0.30) * 200` metres above the water at Heightmap Amount A and the default Sea Level (chapter 2). So ground at altitude h needs the pixel `(h / 200 + 0.30) / A`. These are the enabled locations that ask for 30 m or more. The last two columns give the darkest pixel that is still high enough, at Amount 1 (Better Continents' default; ceiling 140 m above the water) and at Amount 2 (ceiling 340 m). "Out of reach" means that no pixel is bright enough: an Amount 1 world has no legal ground for that location, apart from a site pinned on the location map, which the game takes as it is.

Location	Biome	Quantity	Altitude (m)	Pixel at Amount 1	Pixel at Amount 2
Hildir_cave	Mountain	3	200..5000	out of reach	from 0.650
Dragonqueen	Mountain	3	150..500	out of reach	from 0.525
StoneTowerRuins04	Mountain	50	150..1000	out of reach	from 0.525
StoneTowerRuins05	Mountain	50	150..1000	out of reach	from 0.525
AbandonedLogCabin02	Mountain	33	100..1000	from 0.800	from 0.400
AbandonedLogCabin03	Mountain	33	100..1000	from 0.800	from 0.400
AbandonedLogCabin04	Mountain	50	100..1000	from 0.800	from 0.400
DrakeLorestone	Mountain	50	100..1000	from 0.800	from 0.400
DrakeNest01	Mountain	200	100..2000	from 0.800	from 0.400
MountainCave02	Mountain	120	100..5000	from 0.800	from 0.400
MountainGrave01	Mountain	100	100..1000	from 0.800	from 0.400
MountainWell1	Mountain	25	100..1000	from 0.800	from 0.400
Runestone_Mountains	Mountain	100	100..1000	from 0.800	from 0.400
Waymarker01	Mountain	50	100..1000	from 0.800	from 0.400
Waymarker02	Mountain	50	100..1000	from 0.800	from 0.400
DN_Bossroom	DeepNorth	3	80..5000	from 0.700	from 0.350
DN_hut01	DeepNorth	40	40..1000	from 0.500	from 0.250
MorkBorg	DeepNorth	40	30..10000	from 0.450	from 0.225

4 of the 18 are out of reach at Amount 1.

The 32 alt biomes and their placement windows

In the game's list order (which is also the order the game places them in, per base biome). Count is the number of regions per world, minimum to maximum; the chance applies above the minimum; edge is the `EdgeCount` window [min, max); centre is the minimum distance from the world centre; the colour is Better Continents' default legend colour (`palettes/altbiomemap.txt`). Every alt biome wants `HeightAvg` 30 to 10,000 (absolute), and none sets a neighbour rule or world-position bounds.

Alt biome	Base biomes	Count	Chance	Edge	Centre	Incompatible with	Adds	Colour
Mushroom	Meadows, BlackForest	1 to 3	0.20	66 to 199	500			FF969C
Lantern	Meadows, Swamp, Mountain, BlackForest, Plains, DeepNorth, Mistlands	3 to 5	0.20	66 to 199	500			F2A300
Bones	Swamp, Plains, Mistlands	1 to 3	0.20	66 to 199	2000			F8E1B6
Menhir	Meadows, Plains	1 to 3	0.20	33 to 199	750	Birch Meadows		DDA776
Dark Meadows	Meadows	2 to 5	0.50	66 to 299	1900		1	2D4613
Peaceful Meadows	Meadows	1 to 2	0.20	99 to 299	1000	Bones, Dark Meadows		ADFFDA
Dandelion Meadows	Meadows	1 to 3	0.20	66 to 199	500			E5C54B
Raspberry Meadows	Meadows	1 to 3	0.20	66 to 199	500			C2185B
Smalltree Meadows	Meadows	1 to 3	0.20	33 to 199	500			6EC03D
Birch Meadows	Meadows	1 to 3	0.20	66 to 199	500			CCFFAC
Troll Black Forest	BlackForest	1 to 3	0.20	66 to 199	750			2C7B76
Root Black Forest	BlackForest	1 to 4	0.20	99 to 299	2000	Rock Black Forest, Kalhygge Black Forest		6F5E32
Ruin Black Forest	BlackForest	1 to 5	0.20	33 to 199	500		4	9D6C4B
Rock Black Forest	BlackForest	1 to 5	0.20	33 to 199	750	Pinetree Black Forest, Root Black Forest		5A5264
Pinetree Black Forest	BlackForest	1 to 5	0.20	33 to 199	750	Rock Black Forest		004754
Blueberry Black Forest	BlackForest	1 to 5	0.20	33 to 199	500			4A5CB6
Hut Swamp	Swamp	1 to 3	0.20	33 to 199	1500	Bog Swamp	3	B1332D
Bog Swamp	Swamp	1 to 5	0.20	33 to 145	1500			6E2143
Bat Swamp	Swamp	1 to 3	0.20	33 to 145	2000			382B41
Abomination Swamp	Swamp	1 to 3	0.20	66 to 145	2000			95B17E
Wolf Mountain	Mountain	1 to 5	0.20	199 to 299	2000	Drake Mountain, Fortress Mountain		93C7BC
Drake Mountain	Mountain	1 to 3	0.20	199 to 299	2000	Wolf Mountain, Fortress Mountain		5AAAC2
Fortress Mountain	Mountain	1 to 2	0.20	199 to 299	2000	Wolf Mountain, Drake Mountain	1	B697AF
Lox Plains	Plains	0 to 1	0.20	99 to 299	2000	Goblin Plains, Death Plains		CC605F
Goblin Plains	Plains	0 to 2	0.20	66 to 199	2000	Lox Plains, Death Plains	4	FF7622

Alt biome	Base biomes	Count	Chance	Edge	Centre	Incompatible with	Adds	Colour
Death Plains	Plains	0 to 1	0.20	99 to 299	1000	Lox Plains, Goblin Plains	3	671E07
Kalhygge Black Forest	BlackForest	1 to 2	0.20	20 to 80	500	Root Black Forest		A9AE34
Rockless Mistlands	Mistlands	2 to 5	0.20	99 to 599	2000			D99FE8
Trees Mistlands	Mistlands	1 to 5	0.20	59 to 299	2000			472577
Swords Mistlands	Mistlands	1 to 3	0.20	59 to 199	2000			8A7FDB
Hare Mistlands	Mistlands	1 to 3	0.20	59 to 199	2000			D56EBB
BroodSwarm Mistlands	Mistlands	1 to 3	0.20	59 to 199	2000			8E2F8E

The biome legend

The ten biome names of the game, with the default colours (`biomemap.txt` holds these when it is missing; read from the mod's source). With Expand World Data installed a legend may also name the biomes its yaml adds.

Biome	Colour	Biome	Colour
None (let vanilla decide)	000000	Plains	FFFF00
Meadows	00FF00	Mistlands	7F7F7F
BlackForest	007F00	AshLands	FF0000
Swamp	7F7F00	DeepNorth	00FFFF
Mountain	FFFFFF	Ocean	0000FF

ProximaMaxi's alt-biome legend (map version 13)

Nineteen colours, all from the default palette, 23 planted regions (Mushroom three times, Rockless Mistlands twice, Wolf Mountain on two summits that count as two regions, Lantern on the lobe and every Deep North isle, counted once):

```
# ProximaMaxi alt-biome legend (Better Continents 0.8.1): one line per planted colour.
# Black is unplanted land: the game's random placement handles it.
Fortress Mountain: B697AF
Drake Mountain: 5AAAC2
Wolf Mountain: 93C7BC
Goblin Plains: FF7622
Death Plains: 671E07
Lox Plains: CC605F
Hut Swamp: B1332D
Bat Swamp: 382B41
Abomination Swamp: 95B17E
Troll Black Forest: 2C7B76
Root Black Forest: 6F5E32
Ruin Black Forest: 9D6C4B
Dark Meadows: 2D4613
Peaceful Meadows: ADFFDA
Mushroom: FF969C
Swords Mistlands: 8A7FDB
BroodSwarm Mistlands: 8E2F8E
Rockless Mistlands: D99FE8
Lantern: F2A300
```

Appendix D

Glossary and sources

Glossary

- **Talk, Files, Commands, Game**: the four levels of what an agent can do (chapter 1). **Talk**: write messages only; the human does every step. **Files**: read and write the human's files. **Commands**: run commands (Python, the toolkit, image tools). **Game**: Valheim with Better Continents can be run, by the agent or by the human, and its log read. The first three add up; the notes "If you can ..." in every chapter say what changes at each.
- **Directory**: `[00 BetterContinents.Debug] Directory`, the folder of maps a new world made From Config reads, each map by its standard name (`heightmap.png`, `biomemap.png`, ...). A map file that is not there is not used. A folder with an `export.cfg` also brings that file's settings, which win over the config's.
- **From Config**: the New World preset that makes a world from `BetterContinents.cfg` and its Directory (`SelectedPreset = From Config` pre-selects it). `Disabled` makes a vanilla world, with no warning.
- **Legend**: the text file beside a colour map (`biomemap.txt`, `locationmap.txt`, `altbiomemap.txt`) that names what each colour means. Each kind reads its own way (appendix B): only the alt-biome legend allows a comment after an entry.
- **Snap**: the biome map's rule that a pixel takes the biome of the legend colour nearest to it in RGB. A blended edge therefore grows fringes of other biomes (chapter 2's picture). The location map does not snap: a pin's colour must match exactly. The alt-biome map counts a pixel within 20 RGB units of a legend colour as it.
- **Preview**: a picture of a map set rendered for people (`preview.py`): biome colours shaded by height, water by depth. It is not what the game draws; the minimap picture (`bc scr`) is.
- **Recipe**: a complete setup to copy, with its config, legends, commands, expected log lines and pictures (chapter 11).
- **Amount (Heightmap Amount)**: the config multiplier on the heightmap pixel; sets the waterline pixel (0.30 / Amount) and the ceiling ((Amount - 0.30) * 200 m above the water). Better Continents' default is 1 (a 140 m ceiling); ProximaMaxi's is 2.0 (340 m); since 0.10.3 it goes up to 81 (16,140 m). One 16-bit step of the heightmap is 200 * Amount / 65,535 m.
- **High world**: a world whose heightmap is read at an Amount above 5. Better Continents applies its own versions of the game's height rules there: inside a dungeon means more than 3,000 m over the ground below, the ground is found at any height, and upper altitude limits of 1,000 m or more are lifted (chapter 3). `High Terrain`, a world setting, decides: Auto, the default, as above; On, every Better Continents world; Off, none.
- **Fine heights, fine picture**: `heightmap-fine.png`, an optional 8-bit grey picture beside the heightmap that refines each height by a signed 1/256 of a 16-bit step (format 1), tied to its heightmap by the CRC-32 in its record. A world made with it is compact and needs Better Continents 0.10.3 or later (chapter 2). A world export writes one when its `fine=` option asks (`bc_export fine=auto|off|4|8`, or the export window's choice): an option of each export, not a config key (chapter 9).
- **Plain PNG**: a PNG without interlacing, in 8- or 16-bit grey, grey with alpha, 8-bit RGB or RGBA, or a palette: Better Continents reads it a few rows at a time. Any other picture is read whole, about 1 GB for a 16,384 px colour one.

- **Waterline**: pixel 0.30 / Amount; finalHeight 0.15; height 30 m.
- **Height**: the game's y, in metres; the water surface is at 30 m. The export's `README.txt` writes `(pixel x Amount - 0.15 + Sea Level Adjustment) x 200`.
- **Altitude** (in the game's location rules): height minus 30, i.e. metres above the water.
- **Settings version**: a world's version number, stored in its `BetterContinents` file; it decides which rules the world follows (chapter 10). 12 marks a **new world**: what Better Continents 0.10 makes From Config or from a folder of maps, except in the cases below. 11 marks an **older world**: every world made by 0.9.x, and, in 0.10, every world made from a preset that ships with the mod (version 7 files) or one saved by 0.9.x, from an export of a world resized under 0.9.x or made under Expand World Size at another size, or with `Override version` 11. A world from a much older Better Continents shows 1 to 10 until it is loaded again after its first save (which writes 11), and the creation log of a world made from a shipped preset says 7 although its file is saved as 11. `bc info` shows the version. Nothing upgrades a world from 11 to 12. It is not the map's version number (`gen-13`, "map version 13" of ProximaMaxi), which counts the generator's snapshots.
- **Override version**: `[00 BetterContinents.Debug] Override version`, empty by default, for testing. A number of 11 or more is the version a From Config world gets, and the version every world is saved in and a host sends to joining players while it is set; a number below 11 writes the old fixed layout and drops data. Leave it empty.
- **Layout**: what a new world places by its distance from the centre at its own World Size: the alt-biome grid, the locations (all but four fixed windows, such as the frozen ships' 8,000 to 9,750 m; chapter 3), the game's biome bands and rings, lakes and rivers, the minimap.
- **Big world**: a world whose World Size + Edge Size is past 16,350 m, where the game's own filing of objects goes wrong (chapter 10). A **16k world** is one made from 16,384 px maps.
- **Object sector**: the game's 64 m zone as it files objects (trees, rocks, creatures, buildings), in a 512 x 512 array around the centre that stops at about 16.4 km. Not the alt-biome **sector, region** below, which the game also calls a sector.
- **Wide sectors, wide world**: Better Continents giving every zone out to 65.5 km an object sector and save chunks of its own. A new world past 16,350 m is made wide by default (`Wide Sectors = Auto`); `On` converts an existing one, one way; a wide world needs Better Continents 0.10.3 or later from then on (chapter 10). `Wide Sectors` is each machine's own setting, read by the machine that runs the world; a joining player's game follows the world.
- **Pin**: a site on the location map. A new world puts it exactly on its pixel, before the game's random pass, and the game does not check it.
- **Tile**: 128 x 128 pixels of a map. Maps are held as tiles, a tile of one value as that value; Compact Maps keeps them compressed.
- **Compact world**: a world whose maps are kept, saved and sent as compressed tiles. `Compact Maps = Auto`, the default, makes a new world compact when a map is more than 8,192 px across or the heightmap has fine heights; 0.9 cannot read one (chapter 10).
- **Heightmap Alpha**: a heightmap with an alpha channel that blends it with the game's own terrain (a new world).
- **Twin**: two copies of one vegetation prefab within a metre of each other. The twin guard keeps new ones out of the zones that are generated from now on (vegetation only), in every Better Continents world; `bc_twins` removes the ones a world has.
- **Debug mode**: `Debug Mode = true`: in a world that uses Better Continents, on the machine that runs it, cheats switched on when you spawn (which the host's live `bc` tree needs) and the settings window (chapter 6). It does not reveal the map: most `bc` changes do.

- **Zone**: the game's 64 m square; one location per zone. The game fills a zone once and keeps it.
- **Zone regeneration**: Better Continents resetting the generated zones after a `bc` change (and on `bc regen`), in single player or on the host and in a world that uses Better Continents, so that they fill again with the new settings, except the zones around anything a player built, a tombstone, ground work, the zones a player on another machine has loaded, or you inside a dungeon; a location that reaches into more than one zone is kept or reset whole (chapter 6). It comes with the `bc` commands, which need cheats, not with Debug Mode itself.
- **Ground work**: what the hoe, the cultivator and the pickaxe change (fields, paths, roads, levelled or dug ground). It is stored in the zone's `_TerrainCompiler` object, has no creator, and keeps its zone like a player's building.
- **Grid, points**: the game's sampling of biome and height at 12 m, regenerated at every load: 2048 x 2048 at the default size, more on a bigger new world, at most 8192 a side (chapter 3).
- **Sector, region**: a 4-connected area of one biome on that grid; Ashlands, Deep North and Ocean are one each. Not the object sector above.
- **EdgeCount**: a region's border cell count; the alt-biome size window.
- **HeightAvg**: (lowest + highest) / 2 of a region's border heights, absolute; must reach 30.
- **DistanceFromCenter**: the distance of a region's mean border position from the origin.
- **Alt biome**: one of Valheim 1.0's 32 biome variants, placed per region; changes spawns, vegetation, locations, weather and music, not the biome's name.
- **Planting**: Better Continents 0.8.1 assigning alt biomes to painted regions (`altbiomemap.png`) before the game's random placement; planted regions count toward the maxima (the quota rule).
- **Prioritized**: a location placed first, with 60,000 attempts (bosses, vendors, key dungeons).
- **Unique**: a location of which only one is spawned; several candidates are placed and the rest are removed when the first is reached.
- **Group, minDistanceFromSimilar**: spacing rules between locations of one kind or group.
- **Biome area (Edge, Median, Everything)**: whether the zone centre must be inside the biome (Median: the eight points 64 m around it agree), on its edge, or either.
- **Terrain delta**: the highest minus the lowest of ten random heights within a location's exterior radius; a flatness window.
- **Vegetation window**: in the Ashlands, the lava mask value; the lava map.
- **FBM**: fractional Brownian motion, summed octaves of noise; the gain is the amplitude ratio per octave.
- **Domain warp**: sampling a field at displaced coordinates to bend its outlines.
- **Leash**: capping a warped landmass field against its unwarped shape plus slack.
- **Priority-Flood**: the drainage algorithm that fills pits and gives every cell a downhill route to the sea.
- **ROLL, ROLL_K**: per-biome RMS relief in metres, and the 1 / std factor that makes an FBM's output unit-RMS.
- **Staging**: writing the mod and the config into the Gale profile (`stage-into-gale.sh`).
- **Baking (maps)**: the mod copying the maps and settings into the world at creation (chapter 2). Not the next entry.
- **Bake, baked placements (pieces)**: since 0.10.4, `bc_bake` taking pieces out of the world's objects and keeping them as records of the world's **layer**, drawn and solid on every player's screen (chapter 12). The layer is saved in the world's `BetterContinents` settings file and needs Better Continents 0.10.4 or later to be drawn.

- **Layer, record, palette entry**: the layer is format 1 of `placements.bcp` (appendix E): a palette of kinds (prefabs with a role, a collision kind, flags, a tint, tags) and, per 64 m zone, records (an entry, a place, a turn) with the zone's clear mask, ground and paint.
- **Role**: what a palette entry's records are. **Static**: drawn and solid, never an object. **Live**: a real, usually protected object the server makes when its zone is first generated (doors, chests, stations). **Copy**: drawn unlit; the nearest within Light Distance become lit copies. **Seat**: drawn, and a chair near the player.
- **Town compiler, town file**: a program outside the game that writes `placements.bcp`, and the file it writes. In a Directory it builds its towns into a new world; `bc_bake load` brings a newer one into a running world.
- **Source, operation**: who made a record: source 0 is the town file, source n the in-game operation n (a bake, an unbake, a load, a drop), listed by `bc_bake list`.
- **Undo file**: the journal an operation writes before it changes anything, in `BetterContinents/bakes/<world>-<id>/`; `bc_bake undo` and the check at world load read it.
- **Dry run, confirm**: every `bc_bake` change first says what it would do and changes nothing; `confirm` (or the tab's Confirm) does it.
- **Protected**: a real piece of the layer (with the Protect flag, or adopted by `town`) that nobody can damage or remove.
- **Consumable**: a prefab with a `Pickable`, `PickableItem`, `ItemDrop` or `Plant`. Never baked; a town file's records of one are placed once as ordinary objects, unless its entry has the **Decor** flag (32), which makes them drawn scenery.
- **Orphan**: a town's Live container that a newer file removed or moved while it held items: left standing, unprotected.
- **Converted world**: a world made without Better Continents' maps that a bake or load with `convert` made a Better Continents world keeping the game's own terrain; it carries only its layer.
- **Shadow proxy**: a shadow-only mesh Better Continents merges, on worker threads, from the casting parts of a zone's baked pieces, so that the pieces need not cast one by one.
- **Instance count**: the measure of a bake: the objects (ZDOs), live objects, renderers and drawn instances it removes.
- **Better Continents' own water, sea field**: since 0.10.5, a shader of Better Continents' (`BetterContinents/Water`) put on the game's water materials, and the grid every machine builds at world load of where the world's sea is calm (its Deep North) and hot (red); the felt waves follow the same grid (chapter 13). **Own Water**, **Soft Far Sea**: its two client settings.
- **Candidate**: a world whose sea may differ from the game's (a biome map, a heat map, its own size, Expand World Size); only a candidate's field is built.
- **Lake, lake line**: since 0.10.5 (experimental), one line `name;x;z;level;minX;minZ;maxX;maxZ;depth` of a world's lake list (`lakes.txt`, DataKey 72): a basin filled to an absolute level with the game's water (chapter 13).
- **Clearing mask, TARNCLR1**: since 0.10.5 (experimental), a square bitmask (`clearmask.bin`, DataKey 73) that keeps vegetation off its marked pixels in zones generated after it; `TARNCLR1` is its file's first eight bytes.
- **Tarn**: Raven Iron's mod (Wubarrk and NomadicWar) that lakes and clearing come from; its two file formats are kept. With the Tarn mod installed, Better Continents' lakes and clearing step aside.
- **Spill, spill height**: the height at which water rising in a hollow first runs out of it; `bc_lake here` and `lakekit.make_lake` set a lake's level a margin under it.

- **Gale**: the mod manager; profiles hold BepInEx and the plugins; DLLs are hardlinked from its cache.
- **BC-Tester**: the Gale profile used for these builds.
- **ProximaMaxi, Fire and Ice**: the map this document was learned on; the realm of Wubarrk's Proxima Mod Pack. Fire and Ice was its first name, and the folder keeps it.

Sources, and how to check a claim

Every number in this document has one of these origins:

1. **The mod's code**, Better Continents 0.10.x (`/home/rohan/WubarrkCODING/BetterContinents10`): the height formula (`BetterContinents.WorldGeneratorPatch.cs`: `GetBaseHeightV3`, `GetBiomeHeightWithHeight`; the export's `WorldExportMath`), the forest mapping (`ApplyForest`), heat (`IsAshlands`), the alt-biome map (`ImageMapAltBiome.cs`, `BetterContinents.AltBiomePlanting.cs`, documented in `ALTBIOMES.md`), the save layout (`MIGRATE-1.0.md`), every setting (`SettingsSchema.cs`, the source of appendix B), the image maps (`BetterContinents.MapKinds.cs`), the legends (`Legends.cs`), which settings version a world gets and what each version changes (`Serialize.cs`: `NewWorldVersion`, `UnifiedVersion` and the properties that read it; `Presets.cs`; `WorldImport.cs`: `ExportVersion`), the version rule of a join (`BetterContinents.ZNetPatch.cs`), the world's size and layout (`WorldGeometry.cs`, `BetterContinents.WorldSizePatch.cs`), the location pins (`ImageMapLocation.cs`, `BetterContinents.ZoneSystemPatch.cs`), the twin guard (`VegetationTwins.cs`), the water of rebuilt zones (`BetterContinents.WaterPatch.cs`) and zone regeneration (`ZoneReset.cs` for the rules, `ZoneRegen.cs` for the game's side, `ZoneRegenPatch.cs` for its three hooks into saving and building, `GameUtils.cs` and `DebugUtils.cs` for `bc reset` and `bc regen`). What 0.10.3 added: the largest map and `Max Map Size` (`ImageMapBase.cs`), reading a picture a few rows at a time (`PngRows.cs`), Compact Maps (`CompactMapsMode.cs`; `CompactForLargeMaps` in `BetterContinents.BetterContinentsSettings.cs`), fine heights (`FineHeights.cs`, the checks in `ImageMapFloat.cs`, and `tools/fine_ref.py`, the format's executable spec), the high-world rules (`HighTerrain.cs`, `BetterContinents.HighTerrainPatch.cs`), wide sectors (`WorldSectors.cs`, with the design and the game's code in `tools/sector-tests/DESIGN.md`), the lake merge (`LakeMerge.cs`), the export's sizes and preset (`WorldExport.cs`), the `.bcworld` limit (`WorldCacheShare.cs`), and the colour of the Ashlands sea and the calm sea of the Deep North (`AshlandsWater.cs`). What 0.10.4 added, baked placements: the commands and their words (`BakeCommandLine.cs`, `BakeCommands.cs`, `DebugUtils.Bake.cs`), the tab (`ExportHud.Bake.cs`), the settings (`SettingsSchema.cs`, section 10), the format (`BakedFormat.cs`, `BakedLayer.cs`), what bakes and what stays (`BakeClassifier.cs`), consumables and decor (`BakedConsumables.cs`), the undo files and the check at load (`BakeJournal.cs`, `BakeRunner*.cs`, `BakeSettle.cs`), the server's Live pieces and reconciliation (`BakedServer.cs`, `BakedReconcile.cs`, `BakedProtect.cs`), the join download and cache (`BakedTransfer.cs`, `BakedCache.cs`, `BakedPatch.cs`), ground and paint (`BakedGround.cs`, `BakedPaint.cs`), the drawing and the shadow proxies (`BakedKinds.cs`, `BakedDraw.cs`, `BakedProxy*.cs`, `BakedMeshWrite.cs`), and the read-only API for other mods (`BakedApi.cs`). What 0.10.5 added: the sea field and the felt waves (`SeaField.cs`, `SeaWaves.cs`, `SeaPatches.cs`), the drawing and its fallback (`SeaDraw.cs`, `AshlandsWater.cs`, the shader's sources in `unity/Sea/`), `bc_water` and the probe (`DebugUtils.Water.cs`, `SeaProbe.cs`), `bc genloc` (`Genloc.cs`, `DebugUtils.Genloc.cs`), the lake file and the mask (`LakeSet.cs`, `ClearMask.cs`, `LakeFiles.cs`), the clearing (`ClearVegetation.cs`), `bc_lake` (`LakeCommandLine.cs`, `LakeCommands.cs`), the lake engine (`LakeFlood.cs`, `LakeSpill.cs`, `LakeStreamer.cs`, `LakeBuild.cs`, `LakeWater.cs`, `LakeMap.cs`, `LakeSound.cs`), the push of a changed lake list (`LakeTransfer.cs`), the Tarn mod beside it (`compatiblity/TarnCompat.cs`), and the two bake fixes (`BakeJournal.cs`: `FolderFor`; `BakeRunner.Load.cs`: the echo). What 0.10.7 added: the export for BetterWorlds (`WorldExport.Interop.cs`: the cold map, `world.json`, the manifest keys, the noise pins), the editor pack and its way back (`EditorPack.cs`, `EditorIngest.cs`,

`EditorReaders.cs`, `EditorRowSinks.cs`, `ExportHud.Editor.cs`), the Upgrade World support (`compatiblity/UW.cs`, the `pins` word of `bc genloc` in `Genloc.cs`, `ZoneRegenFilter.cs`, the regen words in `ZoneRegen.cs`), and Expand World Data's and Expand World Size's current releases (`compatiblity/EWD.cs`, `compatiblity/EWS.cs`, `WorldSectors.cs`).

2. **The game's code**, Valheim 1.0, decompiled in `libs-Tools/1.0/DECOMPILED/assembly_valheim.decompiled.cs` (the client; `_SERVER` for the server): `ZoneSystem.GenerateLocationsTimeSliced` (the attempt counts, the six points, the error categories, the unique rule), `AltBiomeWorldData.VerifyBiomeData` (`RemoveCache`, `GenerateBiomePoints`, `GenerateSectors` on every load), `BiomeSector.CanAddModifier`, `WorldGenerator.GetBiomeSector`, `Player.UpdateBiome` (the GetBiome warning), `EnvMan` (the Deep North weather bug), `Player.PlacePiece` and `Piece.SetCreator` (the only place an object gets a creator, which is what zone regeneration counts as built by a player), `Plant.Grow` (the tree that grows from a sapling is a new object without one), `TerrainOp` and `TerrainComp` (ground work lives in the zone's `_TerrainCompiler`), `SimulationDistance` (how far around a player their game loads), `ZNet.SaveWorld` and `ZoneSystem.PrepareSave` (what a save stores), `WaterVolume.Start` and `Heightmap.Regenerate` (the water's depth), `ZoneSystem.SpawnZone` (a zone is filled once), and for big and tall worlds `ZoneSystem.SectorToIndex` and `ZDOMan.FindObjects` (the shared list past 16.4 km), `Utils.SmallPosition` (a spawner past 32,767 m on either axis saved at 20,000 m), `ZoneSystem.ChunkPortal` (the portals' file), `Character.InInterior` (inside a dungeon above 3,000 m), and `WorldGenerator.GetAshlandsOceanGradient` and `WorldGenerator.DeepNorthWaveFade` (the hot sea and the calm sea). The per-type files in `libs-Tools/Decompiled_<version>/assembly_valheim/` are quicker to read; check the folder is the current build's; `altbiomes/RESEARCH-vanilla-behaviour.md` cites them by file and line.
3. **The game's data**, extracted read-only from the bundle `d59cfac` (Unity 6000.0.75f1) by the two extractors: `_ZoneSystem.prefab` and the `_LocationList_*` prefabs (340 location entries, 170 enabled distinct types), `AltBiomes_Erik.prefab` (32 alt biomes).
4. **Real builds**: `Proxima` (map version 11, Amount 1.0), `ProximaEra01` (map version 12), `ProxEra01` (map version 13), each with its game log replayed by the checker. The times, memory and object counts of 0.10.3's big and tall worlds were measured by its work on one machine, a Linux dedicated server, and are gathered with the rest of its facts in `BetterContinents-Guide/v010/facts/`: `FACTS-0103-MERGED.md`, checked against the merged code, wins over the older `FACTS-0103.md` and `wide-sectors.md`; the join times follow from the transfer rate. 0.10.4's numbers were measured on one machine during its build and are kept in `~/WubarrkCODING/BetterContinents10/.work/bc-0104/`: the build spec (`BUILD-SPEC-0104-BAKE.md`; its section 2 is format 1, its section 2.7 the counts of VALtima's real town file), the handoff (`HANDOFF-0104.md`, section 000: the shadow proxies' frame times and memory), the agents' notes in `handoff/`, and the automated game test's run Q (`client/qq5/`: the test village's dry run, bake, undo and unbake, and its object counts); the Unity facts the shadow proxies rely on are in `libs-Tools/SHADOW-PROXY-AND-MESH-THREADING-FACTS.md`. 0.10.5's numbers come from the same folder: the plans (`BUILD-SPEC-0104-WATER.md`, `../bc-0106-tarn/BUILD-SPEC-TARN.md`), the agents' notes (`handoff/water-*.md`, `handoff/lakes-*.md`), and the automated game test's phases W (`client/w-valtima*/`, `client/w-a3/`, `client/w-genloc2/`: the sea of VALtima's world, the parity with the game's shader, `bc genloc`) and L (`work-lakes-b/l-final/`: a lake built, swum, sailed and probed); the water shader's facts are in `libs-Tools/WATER-SHADER-FACTS.md`.
5. **The human's words**, quoted where they set a rule.

To check a claim about the game: find the method in source 2, read it, and note the line. To check a claim about the mod: the fork's source, or `ALTBIOMES.md` section 2 for anything alt-biome. To check a claim about a map: run the checker. Write what you learn into the workspace's notes (`HANDOFF.md`, a

`RESEARCH-*.md`, or a `libs-Tools/*-FACTS.md` with a dated banner and file:line citations), so that the next agent does not re-derive it.

Where to look a fact up

The question	Where the answer is	What to search for
a setting's section, default, range or scope	the fork's `SettingsSchema.cs`; appendix B is generated from it	the key as the config spells it: `"Heightmap Amount"`
what a log line means, and when it is printed	the fork's source	the line's fixed words: `grep -rn "was NOT placed" *.cs`
a `bc` command and its help	`DebugUtils*.cs`, `AltBiomeCommands.cs`; appendix B's tables	`AddCommand("info"`
how a map or a legend is read	`ImageMap*.cs`, `Legends.cs`	the file name: `biomemap`, `locationmap`
a location's rules (biome, altitude, delta, quantity)	appendix C, from `locations/locations_1.0.15.json`	the prefab name: `Hildir_cave`
an alt biome's window and limits	appendix C, from `altbiomes/altbiomes_1.0.15_full.json`	the alt biome's name: `Mushroom`
what the game itself does	`libs-Tools/Decompiled_1.0.17/assembly_valheim/<Type>.cs`	the method: `GenerateLocationsTimeSliced`
a measured number of 0.10.3 (times, memory, sizes)	`BetterContinents-Guide/v010/facts/FACTS-0103-MERGED.md`	the subject: `Compact Maps`, `Fine heights`
whether the chapters still match the mod	`make_b_config.py --check`	(it prints what no longer holds)
what a `bc_bake` line means, or a format-1 rule	`BakeRunner*.cs`, `BakedFormat.cs`, `BakedLayer.cs`; appendix E	the line's fixed words: `grep -rn "a Live record has no id" *.cs`
what a `Water:`, `Lakes:` or `Clearing:` line means	`SeaDraw.cs`, `SeaPatches.cs`, `LakeStreamer.cs`, `LakeFiles.cs`, `ClearVegetation.cs`; chapter 13	the line's fixed words: `grep -rn "the basin is not closed" *.cs`

Worked example: checking the waterline. Chapter 2 says the waterline is pixel `0.30 / A`. In the fork's `BetterContinents.WorldGeneratorPatch.cs`, `GetBaseHeightV3` reads the heightmap (`Settings.ApplyHeightmap`, the pixel times the Amount at the default Blend), then `finalHeight -= 0.15f` and `finalHeight += Settings.SeaLevelAdjustment` (0 at Sea Level 0.5); `GetBiomeHeightWithHeight` multiplies by `200f`. The game's water is `m_waterLevel = 30f` in `ZoneSystem`. So the water is at finalHeight 0.15, where `v x A - 0.15 = 0.15`, `v = 0.30 / A`: 0.30 at Amount 1, 0.15 at Amount 2. The same reading gives the centre cap (chapter 2): below it, `finalHeight > 0.28f` within `m_minMountainDistance` (1,000 m in `WorldGenerator`, 1,500 m only for the oldest world-generator version).

> **Check:** a number you cannot trace to one of the five sources above is not a fact yet. Say so in the text ("not measured", "not checked"), or leave it out.

The map's versions, as a study

Version	What changed	What the human said	Its pictures in this document
01	the layout, drawn as ellipses; islands parked in a row	"we want a realistic feeling map"	chapter 4 (`01-parked-before.png`)
02 to 05	natural fronts, ragged fronts, island groups, the Ashlands arm pivoted 45 degrees south-west (it came within 1.2 km of spawn), a Plains and Mistlands landmass south of spawn		
06	two archipelagos, satellites, skerries, windward cliffs, alt-biome homes	"i like it 06", "refine and work off of it"	chapter 4 (`06-archipelagos.png`)
07	refined alt-biome homes (the random-placement tuning)		chapter 3 (`07b-altbiome-hosts.png`)
08	alt biomes planted with colours (0.8.1)		
09	rivers as drainage, lakes	"the river looks too sharp and unnatural"	chapter 4 (`09d-spawn-closeup.png`)
10	Deep North isles with Lantern, Wolf Mountain moved, the starter river redrawn	"make various alt deep north islands, move wolf mountain"	chapter 4 (`10d-spawn-river.png`)
11	Mushroom three times, streams wetter, natural-basin lakes	"we def want some mushroom meadows"; built as `Proxima`: the cavern 0/3	
12	Amount 2.0, icefields, the Ashlands fattened with burning isles and lava, rim chains, Mistlands lowlands, biome-labelled shallows, the location checker	"get everything right this time"; built as `ProximaEra01`: "reads too flat"	chapters 3 and 4 (`12d-mistlands.png`, `12e-deepnorth.png`, `12i-start-coast.png`)
13	rolling relief, the short beach, the low bog, sunk Mistlands valleys	built as `ProxEra01`: "looks good on my end"	chapters 2, 3, 4 and 8 (`13-full.png`, `13c-ashlands.png`, `13h-altbiome-regions.png`, `13i-start-coast.png`, ...)

Appendix E

placements.bcp, format 1

The whole layout of a baked layer, as Better Continents reads and writes it since 0.10.4 (`BakedFormat.cs` and `BakedLayer.cs` in the mod's source are the reference; this page was checked against them at commit ae63889, the code of 0.10.4). Chapter 12 says what each part means in the game. Every number is little-endian. A **string** is a u8 length and that many ASCII bytes; a **str16** (the registry only) is a u16 length and that many UTF-8 bytes.

The layout

```
HEADER (32 bytes)
  0   char[4]  "BCPL"
  4   u16      format = 1
  6   u16      header flags: 1 some zones have ground, 2 some have paint,
               4 some have clear masks, 8 a registry follows the palette (Better Continents' own)
  8   u32      revision (1 for a new layer; Better Continents adds 1 at every change)
  12  u32      palette entries
  16  u32      zones
  20  u64      records (placements)
  28  u32      0 (reserved)
PRODUCER     string: the tool that wrote the file
PALETTE      one entry per kind, world-wide, in order (a record names its entry by this index)
  u8       candidates, 1 to 8; per candidate:
             string prefab name; i16[3] anchor in mm, in that candidate's own frame
  u8       role: 0 Static, 1 Live, 2 Copy, 3 Seat
  u8       collision: 0 None, 1 Boxes, 2 Trunk, 3 Prefab
  u8       layer: 0 = the game's piece layer, else a Unity layer index (0 to 31)
  u8       flags: 1 UniformScale, 2 NoShadows, 4 Protect, 8 LOD0Only, 16 NoCopyLight,
             32 Decor (Static or Copy only)
  only when collision = 1:
           u8 boxes; per box: i16[3] centre in mm, i16[3] size in mm, in the frame of the
             FIRST candidate, positioned by that candidate's anchor
  string   tint name ("" = none); when not empty: u16[3] colour x 1000, string material filter
  u8       tags; per tag: string key, u8 type (0 bool, 1 int, 2 float, 3 string), value:
             bool u8 (0 or 1), int i32, float f32, string a string
REGISTRY     only with header flag 8 (below)
ZONE INDEX   one row per zone, sorted by zz, then zx, strictly increasing (28 bytes a row)
  i16 zx, i16 zz          the zone, -1024 to 1023 on each axis
  u32 offset, u32 length  the zone's block, from the start of the file
  u32 records
  u16 live                the zone's records whose entry has role Live
  u8  zone flags          1 ground, 2 paint, 4 clear mask, 8 protect footprints (kept, no
                          effect), 16 no vegetation anywhere in the zone
  u8  0
  f32 y min, f32 y max    covering the records and the pieces' heights (bounds whole pieces)
ZONE BLOCKS  each one raw Deflate stream (no zlib header); inside, column by column:
  u32      count (0 for a zone that holds only a mask, ground or paint)
  u16[n]   palette index
  u8[n]    record flags: 1 full rotation, 2 scale, 4 id;
             Better Continents' own: 8 source and value set, 16 seed
  u16[n]   x: (world x - (zx * 64 - 32)) * 1024, rounded, clamped to 0..65535
  u16[n]   z: the same for z
  i32[n]   y: world y in mm (the water is at 30,000)
  u16[n]   yaw: 65,536 steps of 360 degrees (with flag 1, the rotation's yaw, not used)
  the runs, each in record order and only for the records with its flag, in this order:
    flag 1   u8 index of the largest quaternion component (0 x, 1 y, 2 z, 3 w; the
               quaternion is first negated if that component is negative),
               then i16[3] the other three in index order, x 32767
    flag 2   u16[3] x, y, z scale x 1000 (0.001 to 65.535)
    flag 4   u32 id
    flag 8   u16 source (the in-game operation, 1 to 65535) for each such record;
               then, as a second run, u16 value set
```

```
    flag 16  u16 RandMatSeed, 0 to 12344
  the zone's sections, as its flags say, in this order:
    clear mask  u8[32]: 16 x 16 cells of 4 m from the zone's south-west corner, row 0
                  south; cell (row, col) is bit row * 16 + col, in byte bit >> 3,
                  value 1 << (bit & 7); a set bit grows no vegetation
    ground      f32 base in m; i16[65 x 65] height above base in cm; u8[65 x 65] weight
                  (0 the world's ground, 255 this ground); vertex i at
                  (x0 + i % 65, z0 + i / 65), row 0 south
    paint       u8[64 x 64] cells of 1 m from the south-west corner, row 0 south:
                  0 none, 1 dirt, 2 paved, 3 cultivated
  u8 extras, always written (0 when none); per extra: string tag, u32 length, the bytes
TRAILER      u32 CRC-32 (the zlib / IEEE polynomial, reflected) of every byte before it
```

Zone `(zx, zz)` has its centre at `(zx * 64, zz * 64)` and its south-west corner at `(zx * 64 - 32, zz * 64 - 32)`, the game's own zones. A point is in zone `floor((x + 32) / 64)`. A record's point is the anchor of its entry's candidate, and the piece's pivot is `point - rotation x (scale * anchor)`; without flag 1 the rotation is the yaw alone, `Quaternion.Euler(0, yaw, 0)`.

What the reader checks

Better Continents reads every byte before it uses a layer and stops at the first problem, naming it and where it is (the header, a palette entry, a zone, a record). These are its words; `bcp.py check` gives the same.

Part	Refused when	The message
size, trailer	under 36 bytes; over 256 MB; the CRC differs	`the file has N bytes, under the 36 of a header and a CRC`; `the CRC differs: the file says X, its bytes come to Y`
header	not `BCPL`; format not 1; an unknown header flag; the reserved word not 0; over 65,535 palette entries	`header: the file does not start with BCPL`; `header: the format is N, not 1 (made by a newer Better Continents or tool?)`; `header: made by a newer Better Continents or tool: header bit N`; `header: the reserved word is N, not 0`
palette	candidates outside 1..8; role, collision or tag type above 3; layer above 31; an unknown flag; Decor on a Live or Seat entry; a string running past the end or holding a byte outside ASCII	`palette entry N: it has N candidates (1 to 8)`; `palette entry N: made by a newer Better Continents or tool: palette flag bit N`; `palette entry N: it is marked decor (flag 32), which only a Static or Copy entry can be, and its role is Live`
registry	its format not 1; a kind outside 1..5 or a state outside 1, 2; an operation twice; bytes left over	`registry: its format is N, not 1`
index	it runs past the file; a zone outside -1024..1023; rows not strictly sorted by (zz, zx); an unknown zone flag; the pad byte not 0; a block outside the area after the index	`zone X,Z: the rows of the index are not in strictly increasing order (after zone X,Z)`; `zone X,Z: made by a newer Better Continents or tool: zone bit N`
block	it cannot be inflated, or inflates past 16 MB; its count differs from the index; a palette index out of range; an unknown record flag; flag 8 without a registry, or naming an operation or value set it does not list; a Live record without an id; a rotation component index above 3; a scale of 0; a seed above 12344; an extra running past the end; bytes left over	`zone X,Z: the block holds N records, the index says M`; `zone X,Z, record K: a Live record has no id`; `zone X,Z: N bytes are left over after the extras`
across zones	a block's Live records differ from the index's live count; a (source, id) of a Live record twice in the layer; the records do not add up to the header's	`zone X,Z: it holds N Live records, the index says M`; `zone X,Z: a Live record repeats an id already in the layer (source S, id I)`; `totals: the zones hold N records, the header says M`

The reader does not check the header flags 1, 2 and 4 against the zones, the y ranges, the order of the records in a block, or gaps between blocks: Better Continents' own writer keeps all four, and so should a compiler (chapter 12's rules).

What a refusal does: in the Directory of a new world, the world is made without the layer, with a log line naming the file and the problem; in `bc_bake load`, nothing changes; for the world's own layer at load, the load stops (the game refuses to save over it); for a player's download or cached copy, the player is disconnected with the reason and the copy is deleted.

Better Continents' additions

A compiler's file sets none of these, except Decor if it chooses. Better Continents writes them when it changes a layer, so a layer it exports may hold them, and a reader of exported layers must handle them.

Addition	Where	What it holds
header flag 8	a registry between the palette and the zone index	the in-game operations, and the value sets their records point to
record flag 8	two runs after the id run	the operation that owns the record (1 to 65535) and its value set
record flag 16	a run after flag 8's	the piece's `RandMatSeed` as its object held it
palette tags `MatVar<i>` (int)	ordinary tags	the material variant of the piece's material slot `i`
palette flag 32 Decor	the flags byte	the compiler's word that a consumable kind is scenery

```
REGISTRY
  u32     length of the Deflate stream that follows
  Deflate of:
    u16   registry format = 1
    u16   next operation number (one counter for all; a number is never used twice in a world)
    u16   operations; per operation:
      u16   number (for a bake, also its records' source)
      u8    kind: 1 bake (area), 2 bake (box), 3 unbake, 4 load, 5 drop
      u8    state: 1 in the layer, 2 undone
      i64   time, Unix seconds, UTC
      str16 who: the character's name, or "server console"
      f32[4] bounds x1, z1, x2, z2; f32 radius (0 for a box, a load or a drop)
      u32   records added, u32 records removed, u32 live pieces adopted
      str16 Better Continents version
      u16   value sets; per value set:
        u16   values; per value: i32 ZDO key hash, u8 type, then the value:
              0 float f32 | 1 vec3 f32[3] | 2 quat f32[4] | 3 int i32 | 4 long i64
              | 5 string str16 | 6 bytes u32 length + bytes
```

A value set is what a baked piece's object held that Better Continents gives back when the record becomes a real piece again (its health, its creator, ...); pieces that held the same values share a set. The registry keeps at most 1,000 operations.

The look of a record without stored values. A piece whose prefab has a `MaterialVariation` or a `RandomMaterialValues` gets a variant and a seed from its place, the same on every machine: `h` is FNV-1a 32 over the record's point in world mm, three little-endian i32 (`round(x * 1000)`, `round(y * 1000)`, `round(z * 1000)`, to even on a tie); the variant of slot `i` is the first whose running weight passes `u x` the total weight, with `u = ((h xor (i * 0x9E3779B9)) mod 65536) / 65536` in u32 arithmetic; the seed is `h mod 12345`, drawn in 8 buckets (`seed mod 8`). A compiler that wants one variant writes the tag `MatVar<i>`; one that wants another look makes another palette entry.

How Better Continents writes a layer

- The canonical layout: header, producer, palette, registry (if any), zone index, the blocks in index order with no gaps, the CRC.
- Records in a block sorted by palette index, then z, x, y, yaw, flags and the run values. A block it does not change is copied byte for byte; a change re-encodes only the zones it touches. `bc_bake load` rebuilds the palette (the file's entries in its order, then the entries the in-game records still use) and re-encodes every block.
- A new world keeps the file's revision; every change Better Continents makes sets the old revision plus 1. The producer is kept from the compiler's file, or `Better Continents <version> (bc_bake)` for a layer made in the game only.
- An entry made by an in-game bake: one candidate (the piece's prefab), anchor 0, its role from the classifier, collision 3 (Prefab), layer 0, flags 0 (4 Protect on an adopted Live piece), no tint, tags `MatVar<i>` only.
- Raw Deflate at the best level. Two runtimes may compress the same block to different bytes, so compare layers by their decoded content, and by bytes only when one program wrote both.

Appendix F

GNU General Public License, version 3

The GNU General Public License, version 3, reproduced verbatim from LICENSE.

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.
```

```
  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
```

```
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
```

"keep intact all notices".

c) You must license the entire work, as a whole, under this
License to anyone who comes into possession of a copy. This
License will therefore apply, along with any applicable section 7
additional terms, to the whole of the work, and all its parts,
regardless of how they are packaged. This License gives no
permission to license the work in any other way, but it does not
invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display
Appropriate Legal Notices; however, if the Program has interactive
interfaces that do not display Appropriate Legal Notices, your
work need not make them do so.

A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit. Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

a) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by the
Corresponding Source fixed on a durable physical medium
customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by a
written offer, valid for at least three years and valid for as
long as you offer spare parts or customer support for that product
model, to give anyone who possesses the object code either (1) a
copy of the Corresponding Source for all the software in the
product that is covered by this License, on a durable physical
medium customarily used for software interchange, for a price no
more than your reasonable cost of physically performing this
conveying of source, or (2) access to copy the
Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the
written offer to provide the Corresponding Source. This
alternative is allowed only occasionally and noncommercially, and
only if you received the object code with such an offer, in accord
with subsection 6b.

d) Convey the object code by offering access from a designated
place (gratis or for a charge), and offer equivalent access to the
Corresponding Source in the same way through the same place at no
further charge. You need not require recipients to copy the
Corresponding Source along with the object code. If the place to
copy the object code is a network server, the Corresponding Source
may be on a different server (operated by you or a third party)
that supports equivalent copying facilities, provided you maintain
clear directions next to the object code saying where to find the
Corresponding Source. Regardless of what server hosts the
Corresponding Source, you remain obligated to ensure that it is
available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided
you inform other peers where the object code and Corresponding
Source of the work are being offered to the general public at no
charge under subsection 6d.

A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or

```
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
```

receives a license from the original licensors, to run, modify and
propagate that work, subject to this License. You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations. If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License. For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based. The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version. For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement). To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients. "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License. You may not convey a covered

work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all. For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work. The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time. Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number. If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation. If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions. However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

```
  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<https://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<https://www.gnu.org/licenses/why-not-lgpl.html>.
```