SOROLIF RTS MAP GENERATOR

Technical Guide

Offline manual - version 1.8.0

Installation, generation workflows, runtime API, saved content, prefab integration, rendering, URP, water, grass, terrain and troubleshooting. All 26 canonical technical guides and the current integrated Editor help are included.

The online documentation contains the complete guide collection and parameter comparison gallery. Authoring sources and standalone images are maintained outside the Unity package.

Source defaults are distinct from saved scene values and tuned biome materials. Use the installed Inspector tooltips and explicit presets when configuring your project.

Contents

Contents 2

Getting Started 9

Zero-setup demo . . . 9
1. Create a generator . . . 10
2. Configure visual assets . . . 10
Triplanar terrain rendering . . . 11
3. Generate procedural maps . . . 11
4. Generate from an image . . . 15
Procedural contour smoothing . . . 15
5. Runtime generation . . . 15
6. Generate a map screenshot . . . 16
Map altitude . . . 16
7. Save and reload recipes . . . 16
8. Optional integrations . . . 16
Open the included examples . . . 17

Optional Starter package 18

What you can customize . . . 18
Scenes, updates and export . . . 18

Editable Starter biome settings 19

Select and customize . . . 19
Shared artwork . . . 19
Image-import material channels . . . 19

Compatibility and limitations 20

Announced V1 scope . . . 20
Recorded verification . . . 20
Included content . . . 20
Water, vegetation and saved content . . . 21
Runtime and performance . . . 21
Optional integrations . . . 21

Runtime API 22

Generate a map . . . 22
Select a biome explicitly . . . 23

Generation result . . . 23
Read gameplay cells . . . 23
Validate before generation . . . 24
Save a reproducible recipe . . . 24
Included runtime example . . . 24

Generated map information **25**

Captured information . . . 25
Strategic markers . . . 25
Saving and runtime access . . . 26
Previews and older scenes . . . 26

Pack Content: compact map storage **27**

What is stored . . . 27
Editing a packed map . . . 27
English quick guide . . . 28

Optional outputs and project integration **29**

Minimap . . . 29
Aerial . . . 29
Project layers . . . 29
Navigation adapter . . . 30
Bridge prefab lookup . . . 30
Bridge alternatives and barrier colliders . . . 31

Aerial navigation surface **32**

Mountain intersection . . . 32
Collider simplification . . . 33
Diagnostic examples . . . 33

Full-map symmetry **35**

Map altitude **37**

Automatic height map . . . 37
Custom height map . . . 37
Geometry and runtime data . . . 37
Water levels and banks . . . 38

Biome colors **40**

Props placement 41

Getting started . . . 41

Understanding a rule . . . 41

Standalone component . . . 43

English quick guide . . . 43

Supplied model prefabs 44

From model to generated object . . . 44

Placement and storage . . . 44

Source model corrections . . . 44

Forest models . . . 44

Tree alternatives and world-object probabilities 46

Preview alternatives in the Editor . . . 46

Configure varied world objects . . . 46

Matching and placement constraints . . . 47

Snow models 48

Preview . . . 50

Grass prefabs 52

Biome placement passes . . . 52

Recognizing legacy rules . . . 52

Map settings . . . 53

Edges and occupied tiles . . . 53

Included assets . . . 53

Ground color at the roots . . . 54

MapDataContainer and GPU instancing . . . 55

Shader . . . 55

Wind highlights . . . 55

Native screenshots . . . 56

Passes by patch size . . . 56

Other biomes and models . . . 56

Joining adjacent tiles . . . 58

Mountain rendering 60

Separate wall and mountain prefabs . . . 60

Chokepoint mountain contours . . . 60

Rock at the mountain base . . . 61

Short facets on slopes . . . 61

Plateau cliff coverage **64**

Ramp borders **65**

Triplanar Terrain Shaders **66**

Natural triplanar . . . 66
Stylized triplanar . . . 66
Instanced decoration . . . 66
Texture recommendations . . . 67
Face normals and cast shadows . . . 67
Stylized water . . . 67

URP shaders **68**

Project setup . . . 68
Included passes . . . 68
Verification . . . 69

Stylized water materials **70**

Assign a style . . . 70
Camera and pipeline . . . 70
Controls . . . 71
Native previews . . . 72
Tropical refinement - 19 October 2026 . . . 73
Tropical optics: scene color and animated bed . . . 73
Reflection polish and biome palettes . . . 75
Forest caustics and lake shores . . . 77

Glacier water **79**

Cold currents . . . 79
Ice floes . . . 79
Breaking shore ice . . . 79
Camera, animation and presets . . . 80

Autumn water and floating leaves **81**

Floating Autumn Leaves . . . 81
Water appearance . . . 82
Camera and integration . . . 82

Troubleshooting **83**

The inspector reports missing materials or prefabs . . . 83
Generation returns false but a map remains visible . . . 83
River regions cannot be connected with safe bridges . . . 83
A river crosses a path or choke corridor . . . 83
A river approaches a base or expansion . . . 83
A procedural road ends against a plateau wall . . . 84
Toggling Bridges changes choke roads or hides gizmos . . . 84
Image generation fails . . . 84
Water, paths or mountains are not visible . . . 84
River maps have no bridges or disconnected players . . . 85
Plateau ramps or surfaces are hard to read . . . 85
Ramp GameObjects use the wrong Unity Layer . . . 85
Cliff prefabs leave gaps, float or face the wrong direction . . . 85
Custom project layers are missing . . . 85
The top-down screenshot is missing . . . 86
A* Pathfinding is not installed . . . 86
Regenerating the starter demo . . . 86
Getting useful diagnostics . . . 86

Integrated Editor help **87**

Parameters and conditions . . . 87
1. Compare one setting at a time . . . 87
2. Automatic output and imported terrain . . . 87
3. Density, fairness and exports . . . 87
Your first map . . . 87
1. Create the starter scene . . . 87
2. Select the generator . . . 87
3. Make a reproducible change . . . 87
4. Check the result . . . 87
Procedural layout . . . 88
1. Choose a baseline . . . 88
2. Size, players and expansions . . . 88
3. Force strategic passages . . . 88
4. Enable the terrain features . . . 88
5. Compare at a fixed seed . . . 88
6. Make a mountainous landscape . . . 88
Import an image . . . 88
1. Try the image demo . . . 88

2. Enable Read/Write . . . 89
3. Assign the terrain palette . . . 89
4. Tune recognition, then relief . . . 89
Map altitude . . . 89
1. Enable altitude . . . 89
2. Automatic hills . . . 89
3. Custom height map . . . 89
4. First altitude stage . . . 89
Biomes and materials . . . 90
1. Create or select a preset . . . 90
2. Edit the active materials . . . 90
3. Choose triplanar shading . . . 90
4. Grass prefabs and instancing . . . 90
5. Replace props and verify rendering . . . 90
Smooth terrain contours . . . 90
1. Procedural contours . . . 90
2. Imported contours . . . 91
3. Understand protected areas . . . 91
4. Tune paths independently . . . 91
Dirt paths and bases . . . 91
1. Use curved paths . . . 91
2. Reduce visual clutter . . . 91
3. Check ground and ramp joins . . . 91
Water, plateaus and cliffs . . . 91
1. Choose a water layout . . . 91
2. Make high ground readable . . . 92
3. Fit your cliff prefabs . . . 92
4. Verify layers and shores . . . 92
5. Embed rocky mountain faces . . . 92
Export a top view . . . 92
1. Capture the current map . . . 92
2. Set framing and destination . . . 92
3. Resolve missing captures . . . 92
Optional outputs and layers . . . 92
1. Choose helper outputs . . . 92
2. Set flight clearance and layers . . . 93
3. Enable integrations deliberately . . . 93
Use it in your game . . . 93
1. Prepare the generator . . . 93
2. Generate and check success . . . 93

3. Integrate during a loading phase . . . 93
4. Regenerate deliberately . . . 93
Resolve a problem . . . 93
1. Generation fails or the map is empty . . . 93
2. Materials or props are missing . . . 94
3. There is not enough room . . . 94
4. Prepare a reproducible report . . . 94

Getting Started

English | French version

Release scope: **Unity 6000.2 or later, URP, Windows / Android / iOS**. The four supplied biomes are Forest, Tropical, Snow and Autumn. See compatibility and recorded validation.

Import Core first, then the optional **Starter unitypackage** for editable examples under `Assets/SorolifMapGenerationStarter`. Core updates do not overwrite that folder. See Starter installation.

Choose **Biome** first. **Biome preset** lists only matching BiomeSetting assets and selects a compatible default: editable Starter preset, then Core reference preset, then another matching asset. A compatible custom selection is preserved. This works in both Simple and Expert mode. **Select biome preset to edit** opens its settings. If no matching preset exists, the Inspector explains the automatic fallback. Direct material and prefab overrides retain priority.

The **Biome** menu offers only types with an available `BiomeSetting`. The supplied selection is Forest, Tropical, Snow and Autumn. **Show In Biome Selector** is on by default for new presets; turn it off for unfinished or optional content. To offer another type, create a `BiomeSetting`, choose its Biome Type and leave this option enabled. No enum entries are removed. Existing explicit settings and runtime generation continue to work even when their preset is hidden from new selections.

Each generated root has a **Generated Map Info** component with its captured seed, biome, symmetry, main settings and strategic markers. Use **Select Map Information** on the generator to find it; its **Show Gizmos** and **Show Labels** switches control the map's Scene view markers. See Generated Map Information.

For prefab placement rules and their Inspector, see Props Placement.

For grass prefabs and GPU instancing, see Grass Details: select the generator, click **Apply dense grass preset (screenshots)** in **Terrain style**, then **Generate Map**. This enables props and grass with chance multiplier 1 and height 0.5-0.8. Older biomes without grass rules use the included prefabs automatically; authored grass rules remain authoritative.

Zero-setup demo

For an in-Editor walkthrough, open `Tools > Sorolif > Map Generation > How To Use`. The guide includes eleven topics with numbered steps, English/French language selection and accent-insensitive search. English is the initial default; an explicitly selected language is remembered. The **How To Use** button in the Inspector and the **?** buttons beside Simple/Expert sections open the relevant topic. Reading the guide never changes your map settings.

Start with **Your first map**, then follow **Procedural layout** or **Import an image**. Other pages cover map altitude, biome materials, contour smoothing, dirt paths, water and cliffs, PNG export, runtime code and troubleshooting. Explicit action buttons create demos/assets or select relevant objects; generation remains an Inspector or runtime action. The runtime page provides a copyable, complete `MapBootstrap.cs` example. **Select the complete offline guide** highlights the corresponding documentation asset in Project.

Use `Tools > Sorolif > Map Generation > Create Starter Demo Scene`. Choose a scene location. The command creates independent biome settings referencing Core/Art, lighting, a camera and a complete generated map; existing settings are preserved.

See WaterRendering.md for the Forest, Snow, Tropical and Autumn water materials, camera setup and shader controls.

See MountainRendering.md for the rock transition at the foot of mountains, contour smoothing and preserved physics layers.

The same wizard prepares four editable biome presets: Forest, Tropical, Snow and Autumn. They use the package's own textures and configured source-model prefabs, with shared cube-based resource placeholders. Select **Biome preset** to change the visual starting point. Details are in StarterAssets.md.

To evaluate image-driven generation without preparing an asset first, use `Create Image Import Demo Scene`. It uses the packaged `Examples/ImageImport/StarterImageMap.png`, creates a matching editable palette under `Assets/SorolifMapGenerationStarter`, and leaves the source image and palette assigned on the generated scene for inspection.

The **Welcome & Setup** window presents this action on first import and once after each package update. Reopen it at any time from `Tools > Sorolif > Map Generation > Welcome & Setup`. Its readiness section checks the packaged shaders, biome discovery, optional Unity Layers and the currently selected generator. The same report is available through `Run Setup Check`.

The demo uses a fixed seed so its result is reproducible. Its generator remains editable: change the seed or terrain settings in the inspector and click **Generate Map** to explore variations.

1. Create a generator

Use `Tools > Sorolif > Map Generation > Create Map Generator`. The command creates a GameObject with `MapGenerator` and a default `MapDefinition`.

The generator creates its output as a child hierarchy. Re-generating replaces that hierarchy while preserving the generator component and configuration.

The inspector starts in **Simple** mode. This view is intended for normal production use and exposes the map source, players, strategic density, main terrain features, resources and seed. Use the `Balanced`, `High Ground` or `Naval` quick-start buttons to configure a coherent baseline before adjusting individual features. Select **Expert** at the top of the inspector to tune every algorithm, rendering override, water/cliff parameter, screenshot option and scene integration reference. Switching views never discards hidden values.

New recipes start with one main River layout, automatic biome bridge prefabs, forests, mountain landmarks and two contested objectives. Additional Naval Zones and artificial Choke Points start at zero: the river already provides a natural strategic passage. Increase those values explicitly for extra waterways or forced bottlenecks. `Balanced` restores this terrain profile and enables bridges after another preset was selected. `High Ground` uses strategic plateaus and their ramps without water. `Naval` uses islands and requires naval transport between separate landmasses. Preset buttons preserve the selected biome asset, map size, player count, seed and output settings. Existing saved recipes retain their own values; select a preset explicitly to adopt its settings.

The initial 70-cell map leaves room for two players and their natural expansions. Very small maps may not fit every requested expansion; generation reports `expansions.capacity` with the affected player count. Increase Map Size or reduce Expansions Per Player when this warning appears.

Expert options are grouped into separate collapsible sections, including individual plateau/ramp, forest, mountain, image-smoothing, river, bridge, island, lake and cliff-fitting groups. Section state is stored in Editor preferences and never changes the map recipe.

Generate Minimap Layer and **Generate Aerial Layer** are off by default. Find them under **Output** in Simple mode, or **Optional outputs and project layers** in Expert. Regenerate after changing either switch. Optional outputs and project integration covers their layers, graph tags and bridge prefab lookup.

When **Generate Aerial Layer** is enabled, the hidden `MeshGround/AerialGround` mesh and its collider provide an optional flight surface on the `Aerial` layer (falling back to `Ignore Raycast` when that layer is absent). It follows the final ground and water, including global altitude, with **Aerial Clearance** world units of vertical clearance (**3** by default). Mountains do not raise the flight level: the continuous surface passes through the massif, and rock protruding above it remains a physical obstacle through its own collider. A conservative margin near steep ground breaks can add clearance. Regenerate the map, then rebuild your navigation data if your game uses a separate navigation bake, including protruding rock as obstacles. The renderer stays hidden; no navigation package is required. See Aerial navigation for diagnostic views.

2. Configure visual assets

Assets can be assigned directly on `MapDefinition`, supplied through its explicit `Biome Override` reference to a preset anywhere in the project. Select a compatible preset in the Inspector, or assign `MapDefinition.BiomeOverride` in code. Leaving it empty uses directly assigned local materials and prefabs. All included art is stored under `Art`; runtime generation reads serialized asset references.

Recommended minimum setup:

- Grass Material

- Plateau Material when a distinct high-ground surface is desired; it falls back to Grass Material

- Dirt Path Material

- Water Material when water is enabled

- River Bridge Prefab with a non-trigger walkable deck collider when river bridges are enabled

- Mountain Material when mountain zones are enabled

- Cliff Prefabs when cliffs are enabled

- Gold Mine and Oil Field prefabs when those resources are requested

Direct material and prefab values on `MapDefinition` take precedence over the resolved `BiomeSetting`. When `Biome Override` is empty, the original recipe uses its explicitly assigned local assets. Assign a compatible biome preset to use the supplied complete art setup.

Plateau ramps are added outside the plateau without removing its faces. Set `Plateau Ramp Length` (default 5 cells, range 2-18) to choose a short steep descent or a longer gentle one. Ramp surfaces and colliders use the shared `Walkable` layer configured by `Walkable Layer (ramps and bridges)` on the map. No dedicated Ramp layer is needed. If the configured layer is missing, generation falls back to the configured ground layer, then `Default`.

To create only the starter visual assets, use `Tools > Sorolif > Map Generation > Create Starter Biome Assets`. It creates editable triplanar terrain materials, original stylized grass/forest/dirt/sand/rock texture copies, four `BiomeSetting` presets (Forest, Tropical, Snow and Autumn), and cube-based resources, trees, rocks, cliffs and a river bridge under `Assets/SorolifMapGenerationStarter`.

This explicit starter action reserves a free user-layer slot as `Walkable` for ramp surfaces and bridge decks. Existing layer names are never replaced. If every user-layer slot is occupied, the setup check reports the missing layer.

Bridge deck colliders also use `Walkable`. Separate barrier colliders use the configured Environment layer. Custom projects should define these layers in Tags and Layers.

Triplanar terrain rendering

The included `Sorolif/RTS Terrain/Triplanar` shader is selected automatically for newly created starter terrain materials. Use `Top Texture` for horizontal terrain and `Cliff / Side Texture` for steep faces. `World Texture Scale` controls world-space tiling and `Projection Blend` controls how sharply projections meet.

`Sorolif/RTS Terrain/Triplanar Stylized` adds banded lighting, slope accents and a rim highlight for readable low-poly or illustrated maps. The starter asset command creates `Triplanar_Stylized_Preset.mat` as an editable example. Both shaders avoid visible UV stretching on plateau walls, mountain slopes and generated cliffs. Detailed property guidance is available in TriplanarShaders.md.

3. Generate procedural maps

With `Enable Map Altitude` active, the map perimeter is closed automatically by rock faces down to a common base. `MeshGround/MapSides` follows the rendered terrain, including paths and coastal floor extensions, and has a matching collider. It uses Mountain Material, then Env Material or Grass Material as a fallback. Existing mountain border faces are continued below their lower edge.

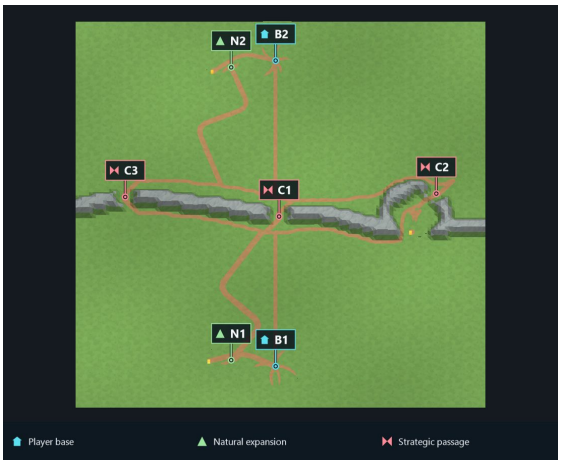
Select `Procedural` as the generation mode, choose the map size, player count, symmetry and water layout, then click **Generate Map**.

Full Map Symmetry is disabled by default: **Symmetry** controls player-start placement while the rest of the map keeps natural generation. Enable **Full Map Symmetry** to mirror the entire map, including terrain, water and decoration. See Full-map symmetry.

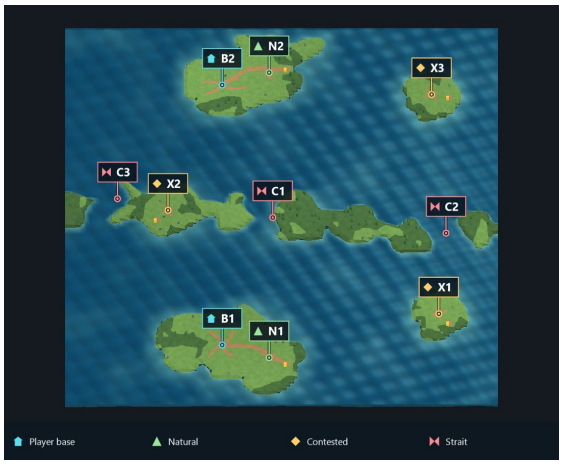
For reproducible output, disable `Use Random Seed` and provide a fixed `Seed`. The same configuration and seed produce the same logical map layout.

Set `Choke Points` above zero to funnel travel between player territories through narrow passages. Continuous dividers use forests when enabled, or rock ridges when forests are absent; existing mountains and water remain obstacles. In the `Islands` layout, irregular low islands narrow the sea into straits of varying width, including the outer sea route. Forest settings populate woodland when matching props are configured; mountains add optional rock cores. Nearby neutral expansion islands can join these formations without moving their resources or changing the expansion count. Player coastlines retain an open-water apron. Naval passages remain water and do not create dirt paths. Planning uses the final relief, ramps and bridge decks, preserves settlement clearings, and reconnects approaches through legal terrain. Constrained terrain can require additional openings to keep existing settlement access. Zero disables these strategic dividers. Regenerate the map to update its passages.

With multiple passages, openings spread along the territory boundaries and need broad approaches on both sides. Tiny isolated pockets are avoided. Land roads share a collector with a single connection from each town, rather than one long road from every base to every passage. Adjacent ramp/deck openings share one marker. If safe spacing cannot fit the requested count, optional openings are skipped; access-preserving openings can still increase the count.



Land choke



Naval choke

B marks main bases, **N** natural expansions, **X** contested expansions and **C** the actual passages.

`Contested Expansion Count` scatters neutral objectives over free terrain using the seed. Placement compares complete layouts to balance estimated access from the player bases while reducing large empty sectors. Individual sites may favor different players; they are no longer all attracted to player medians. Access is estimated from base-to-objective distances before relief and water generation, not from unit travel times. Footprint safety and spacing can reduce the achieved count on crowded maps. With a single objective, equal access takes priority.

Important density controls:

- `Has Plateaus`: creates separate role-aware high-ground platforms for starting bases, natural expansions, later player expansions and contested expansions.

- `Plateau Relief Scale`: vertical separation between strategic tiers.

- `Plateau Footprint Scale`: buildable top size around each strategic location. Values up to `2.0` are supported; nearby strategic zones and map borders can reduce the effective size to preserve valid spacing.

- `Plateau Contour Irregularity`: strength of broad lobes and recesses in the platform outline; it does not add noisy single-cell teeth.

- `Plateau Ramp Width`: width of planned accesses in grid cells.

- `Plateau Edge Thickness`: thickness of the non-walkable cliff border.

- Plateau outer walls automatically use the biome's `Cliff Prefabs`, even when `Has Cliffs` is disabled. Prefabs are fitted to the complete wall length and height, while ramp openings remain clear and traversable.

- Plateau coastlines remain port-accessible: with `Has Cliffs` disabled, the complete land/water boundary uses a continuous low shelf and sloped mesh transition. With cliffs enabled, only high shore sections can stay abrupt; low buildable beach openings remain free of vertical faces and cliff prefabs.

- `Plateau Forest Density Multiplier`: proportion of the global forest coverage allowed as natural pockets on safe plateau tops when both forests and plateaus are enabled. Zero disables summit forests; build pads, ramps and strategic paths stay clear.

- `Natural Expansion Second Access`: gives each natural a second outward ramp toward its choke or the map center when space permits.

- `Ramp Dirt Path Approach Length`: length of the visibly worn path before and after each ramp.

- `Ramp Dirt Path Approach Width`: width multiplier for these tapered traffic zones.

- `Has Cliffs`: covers generated steep terrain walls with the configured cliff prefabs.

- `Cliff Prefab Length Axis`: axis used as the wall length; `Auto` selects the largest horizontal renderer extent.

- `Cliff Tiles Per Prefab`: preferred number of connected terrain edges covered by one stretched detail. The default of twelve substantially reduces GameObject count; unsafe curvature still forces an automatic split.

- `Cliff Depth Scale`: scales the prefab only across the wall. The `0.72` default reduces overlap on plateau tops and low ground while keeping the abrupt face centered.

- `Cliff Height Scale`: enlarges cliff details around the center of the abrupt face, making their top visibly overhang the high ground while retaining low-side coverage. The safe default and runtime minimum are `1.15`; older serialized values are upgraded automatically.

- `Ramp Cliff Depth Scale`: dedicated cross-ramp depth for the two inclined rails. The `1.15` default makes ramp borders more substantial without thickening every other cliff.

- `Ramp Cliff Inset`: places each rail's inner renderer face beneath the ramp edge. The `0.32` default keeps the prefab tight against the complete descent while the remaining depth grows outward. Maps saved with the former `0.12` default are upgraded automatically.

- `Cliff Segment Overlap`: small world-space overlap that closes seams between neighboring details and at corners.

- `Cliff Vertical Overlap`: extra coverage above/below the measured height break.

Cliff prefab renderer bounds are centered on the actual steep face, independently of the prefab root pivot. The grid-to-world half-cell offset is applied automatically. Every ramp component creates exactly two continuous lateral cliff rails. Each rail uses one prefab stretched between the high and low caps and pitched in 3D to follow the ramp slope. Its inner renderer face is aligned under the ramp boundary rather than centering the whole prefab on it, so the border stays flush while most of its depth remains outside the playable ramp; the access center and both caps remain open. In the Unity Editor, generated details remain connected prefab instances; runtime generation uses ordinary instantiated GameObjects.

- `Cliff Run Height Tolerance`: maximum height variation merged into one straight wall band.

- `Mark Cliff Cells Unbuildable`: prevents construction on cells directly touching a generated wall without changing ramp walkability.

- `Island Contour Irregularity`: strength of broad coves, asymmetric coastal lobes and tapered peninsulas in the `Islands` water layout; protected strategic land and naval choke corridors are unaffected.

- `Additional Naval Zones` (serialized as the backward-compatible `NavalZones` field): secondary water features added after the selected layout's primary feature. Zero still generates water. `Lake` always starts with one broad central lake, while additional zones become smaller separated lakes at seeded random positions; `River` starts with one river and adds the requested count.

- `Has Bridges`: procedural rivers crossing a protected choke path receive a bridge perpendicular to the local river course, keeping the river continuous underneath. Oblique paths reconnect to its dry landings. The river must flow across both sides and the path must continue on both dry banks; tangencies, parallel flow and dead ends do not request a path bridge. Assign `BiomeSetting > Biome Prefabs > River Bridge Prefab` (or the map River Bridge Override) and keep clear landings within `River Bridge Max Span`.

- River channels avoid the entire open area of main bases and all expansions, using `Player Base Forest Clear Radius` and `Expansion Forest Clear Radius` even without forests. These shared radii are also exposed under **River shape**. The first river uses the center when its full channel fits; an occupied central clearing takes priority. Additional rivers and Random resolving to River use the same protection. Rivers may pass between settlements when enough space remains for their width.

- `Player Plateau Border Safety Distance`: minimum 0-20 cell clearance between a main player plateau border and the map boundary. Zero lets rotational lake layouts fit plateaus directly against the perimeter and maximizes their distance from the central lake.

- `Main Base Ramp Count`: requests 1-4 planned accesses per main-base plateau. Accesses are distributed as an inward-facing fan and the lake footprint keeps a dry land band beyond them.

- `Symmetry: None`: samples several seeded candidates, then randomly selects among those that fill unused space well enough. This keeps layouts genuinely irregular without allowing a very large unused region. Any incomplete main-base result is simulated again, up to five total attempts, before generation retains and reports the best layout.

- `Island Minimum Water Gap`: contour-to-contour clearance for unrelated islands, from 2 to 16 cells (default 6). Player-island radius adaptation and bonus/expansion placement use the same value.

- Procedural island silhouettes include compact, elongated, crescent/C, hook/J, forked, L, P, rugged and potato-like families. Player islands rotate through the strongly distinct families so a seed cannot accidentally make every coast compact, and their defining branches extend beyond the protected build core. Player forests receive an identical cell-area quota based on the most constrained island; some trees may enter the buildable ring, but the HQ core, paths and resource access remain clear.

- `Mountain Zone Count`: independent ridge or massif formations.

- `Mountain Max Area`: maximum footprint per formation, including foothills.

- `Mountain Chain Width`: width multiplier; both ridge sides vary independently.

- `Mountain Rise Sharpness`: crest sharpness without removing the foothill transition.

- `Mountain Path Clearance`: open buffer around strategic routes, choke corridors, ramps, resources and imported DirtPaths.

- `Path Connection Count`: one gives towns and expansions essential access to the road network; higher values add alternative settlement links.

- `Random Long Path Count`: extra map-spanning roads; zero is recommended for a clean layout.

- `Path Branch Count`: maximum extra decorative branches beyond local town streets; the actual extra number is randomized.

Procedural roads stay curved with `Enable Path Smoothing` on or off. The option refines their contours without selecting a different road network. Towns connect to natural expansions, other expansion sites and strategic arteries through safe terrain. Roads do not automatically trace forest outlines or connect unrelated mines to one another.

Procedural bases act as towns with irregular dirt hubs and 4-6 curved local streets; expansions use 3-5 shorter arms. Their rotation, length and curvature vary with the map seed. Arms shorten or disappear near water, mountains, resources, ramps and height breaks. Imported image paths retain their authored footprint.

Free ends of curved paths and base arms taper to points. Their edge variation fades out at the tip; connected roads and ramp mouths retain their width.

Dirt paths and ramp approaches have a shallow 0.02-unit recess, with edges blending into the ground and ramp mouths. Their colliders follow the same surface. The depth remains 0.02 Unity units when `Cell Size` changes.

4. Generate from an image

1. Select `Image` generation mode.

2. Assign `Source Map Image`.

3. Enable **Read/Write** in the texture import settings.

4. Assign a `MapGeneratorImageLayer` set, or use **Auto Match** when its source-name patterns match the image family.

5. Keep related scans in one shared layer set by adding a common pattern such as `PhotoScan_2`. Matching ignores case, spaces, punctuation, copy suffixes and file extensions.

6. Use semantic layer prefixes (`Grass`, `Forest`, `DirtPath`, `Water`/`Lava`, `Mountain`) so imported cells receive the correct gameplay behavior and biome-material fallback.

7. Keep **Enable Image Ground Relief** enabled for subtle seeded altitude variation; tune its strength and scale when a flatter or broader result is preferred.

8. Leave **Import Image Resources** disabled when gold mines and oil fields will be placed manually.

9. Generate and tune tolerance/vote values only if the scan contains unusually strong lighting or compression noise.

The importer is tolerant of paint and JPEG variation, including low-saturation black, gray, silver and white regions. `Water` and `Lava` layers generate both navigable water classification and the continuous underwater-floor terrain mesh; their visible surface and seabed extend beneath at least one neighboring coastal cell to prevent shoreline gaps. This overlap is visual only and does not make ground cells navigable as water. The inspector reports a blocking validation error when the source texture is missing or not readable, and warns about duplicate prefixes or unresolved custom materials.

Procedural contour smoothing

Enable Procedural Contour Smoothing is enabled by default, with strength 0.85. It reuses the image importer's geometric relaxation on the final forest, mountain and shoreline contours. Adjust **Procedural Contour Smoothing Strength** in the Simple terrain settings or Expert scene settings; zero or disabling it restores the previous geometry, including the optional planar vertex noise.

The deformation affects shared render/collision vertices only. It preserves cell types, heights, resources, strategic plateaus, ramps, reserved land routes and bridge approaches. Curved dirt paths retain their separate path-smoothing control. Water is rebuilt from the final contour field after terrain generation, without repeating water placement or connectivity rules. Image contour controls retain their existing behavior and source-image boundary fitting.

Logical base/expansion/resource regions, plateau interiors and broad vegetation clearance masks do not freeze decorative forest boundaries. Physical resource anchors, ramps, plateau edges, bridges and narrow choke geometry remain pinned. Mountain footprints may be smoothed even though their gameplay type is Blocked; other blockers stay pinned. Coastlines and mountain outlines take priority over decorative material junctions so grass/forest changes do not interrupt the curve. Vegetation exclusions and strategic route reservations are retained unchanged.

5. Runtime generation

```
using Sorolif.RTS.MapGenerator;
using UnityEngine;

public sealed class RuntimeMapExample : MonoBehaviour
{
    [SerializeField] private MapGenerator generator;

    private void Start()
    {
        generator.Definition.UseRandomSeed = false;
        generator.Definition.Seed = 20260711;

        ProceduralMapGenerationResult result =
            generator.GenerateRuntimeMap();
```

```
        if (!result.Succeeded)
            return;

        Debug.Log($"Generated {result.Grid.Size}x{result.Grid.Size} map");
    }
}
```

`ProceduralMapGenerationResult` exposes the generated root, grid, strategic cell positions and player spawn world positions.

The complete public entry-point reference and an editable integration example are documented in RuntimeAPI.md. UPM installations expose the example in Package Manager's Samples section; the classic Asset Store package places it under `Examples/Runtime Generation`.

6. Generate a map screenshot

Enable `Generate Screenshot After Generation` in the Screenshot section. Top-down is the default view. In Expert mode, select Isometric and configure pitch and yaw when a three-quarter presentation better exposes terrain elevation. Both modes fit all eight corners of the generated renderer bounds. Configure `Screenshot Output Folder`, resolution, padding, background and the LayerMask to include. The capture uses its own temporary orthographic camera and therefore does not require a scene camera.

Relative folders are resolved from the Unity project root in the Editor and from `Application.persistentDataPath` in a player build. Absolute folders are used unchanged. The default filename is `MapName_Seed.png`; disable `Screenshot Overwrite Existing` to preserve an existing image and append a timestamp instead.

Use **Capture Map Screenshot** to capture an already generated map manually. At runtime, the resulting absolute path is returned in `ProceduralMapGenerationResult.ScreenshotPath`:

```
ProceduralMapGenerationResult result = generator.GenerateRuntimeMap();

if (result.Succeeded && !string.IsNullOrEmpty(result.ScreenshotPath))
    Debug.Log(result.ScreenshotPath);
```

Map altitude

For broad relief in either generation mode, enable **Map altitude** in the Simple or Expert Inspector. Set its minimum and maximum offsets and optionally assign a readable grayscale height map; otherwise seeded hills are generated. See MapAltitude.md for image orientation, geometry and runtime sampling. Regenerate after changing these settings.

7. Save and reload recipes

After generating in the editor, use **Save Current** to create a `MapGenerationSaveData` asset. **Reload Saved** reapplies its definition and seed, then regenerates the map.

The save asset stores the generation recipe and strategic positions; it does not duplicate all generated meshes as source assets.

8. Optional integrations

The core package has no mandatory third-party dependency.

- A* Pathfinding Project: automatically detected at runtime through reflection.

- `SOROLIF_RTS_INTEGRATION`: enables internal Sorolif GPU-instancing hooks when those types are available.

- `SOROLIF_MAPGEN_VERBOSE_LOGS`: enables detailed subsystem logs for diagnostics.

Do not enable the Sorolif RTS integration symbol in a standalone project unless you also provide its corresponding framework assemblies.

See Compatibility.md for the verified release environment, render-pipeline behavior, performance considerations and features intentionally left to the host RTS project.

Open the included examples

For a ready-made example, choose `Tools > Sorolif > Map Generation > Open Included Procedural Demo` or `Open Included Image Demo`. Unity creates a new editable scene copy under `Assets/SorolifMapGenerationExamples` and opens it. Existing copies are preserved. The terrain is visible as soon as the scene opens. Select the Map Generator to inspect its assigned recipe, or enter Play Mode to regenerate it automatically through the included `QuickStartDemo` component. No third-party content or optional runtime sample import is needed. Four presets, materials and illustrative cube-based prefabs are included under QuickStart.

The commands work for both distributions; UPM scenes cannot open directly from the read-only package. UPM materials and presets belong to the package cache; duplicate the biome or material into Assets to customize it. Remove or disable `QuickStartDemo` once your own startup code calls the runtime API. The creation wizards remain available for producing fresh editable examples in Assets/SorolifMapGenerationStarter.

Optional Starter package

English | Français

Install **SorolifRTSMapGenerator-Core.unitypackage** first. Core contains the generator, Editor tools, shaders, manuals and the complete shared **Art** folder: models, textures, materials, prefabs and reference settings for Forest, Tropical, Snow and Autumn.

The optional **SorolifRTSMapGenerator-Starter.unitypackage** installs **Examples** under **Assets/SorolifMapGenerationStarter**. Import it through **Assets > Import Package > Custom Package**, or **Tools > Sorolif > Map Generation > Import Starter Package**.

It contains:

- Four independent, prefilled `Starter...Biome` settings under `Examples/QuickStart`, initialized from the Core reference settings and linked to Core/Art.

- Two scenes demonstrating procedural and color-coded image generation, the example input image and an editable image palette.

- Eight packed RTS map prefabs for 1v1, 2v2, 3v3 and FFA, each with a portable recipe using the editable Starter biome.

- Runtime sample code and the small demonstration-specific assets used by the scenes.

What you can customize

Change probabilities, prop rules, biome options and assigned references on a **Starter setting**. These are separate assets with separate GUIDs; changes do not modify the reference settings in Core/Art. The biome selector prefers an available Starter default, while retaining a compatible explicit custom choice.

Models, textures, materials and model prefabs remain shared references to **Core/Art**. Changing a field that selects one of them is local to your setting. To change a material's own values, duplicate it into your project and assign the copy. To edit a model prefab's hierarchy, create a project-owned prefab variant. Editing a shared source directly edits Core content and that content can be replaced by a Core update. No visual meshes or textures are duplicated automatically.

Create Starter Biome Assets can create the four independent settings from Core without importing the example archive. Existing settings, including intentionally empty fields and modified rules, are preserved. It does not recreate models, materials, textures or project layers.

Scenes, updates and export

Opening a supplied demo from **Welcome & Setup** creates a scene copy under `SceneCopies`. **Select Starter Folder** locates the editable content. The supplied map prefabs and recipes reference Starter settings.

Core updates contain no Starter settings or examples and do not overwrite that folder. Shared Core artwork can receive updates. Explicitly reimporting Starter can overwrite selected Starter assets; keep project edits in version control. Starter depends on Core scripts, shaders and Art.

In the publisher workspace, `Art` and `Examples` remain in the versioned module. Only copies of **Examples** move into the Starter root in an isolated export project, preserving their GUIDs. The author's live generated scenes are not distribution sources. No `Resources` folder is shipped.

The release targets Unity 6000.2 or later, Editor-matched URP, Windows, Android and iOS. See compatibility for recorded checks and mobile validation status.

Editable Starter biome settings

Core/Art contains the source models, textures, materials, prefabs and reference biome settings for **Forest, Tropical, Snow and Autumn**. The optional Starter contains independent settings for those four biomes under `Assets/SorolifMapGenerationStarter/Examples/QuickStart`.

Run **Tools > Sorolif > Map Generation > Create Starter Biome Assets** to create missing editable settings from Core, even without the example archive. Existing settings are preserved exactly. The command does not regenerate visual assets or create project layers.

Select and customize

Choose **Biome**, then **Biome preset**. The available Starter setting is preferred over the shared Core reference; an explicitly selected compatible custom preset is preserved. **Select biome preset to edit** opens the chosen asset. Direct MapDefinition material and prefab overrides remain authoritative.

Each Starter setting has its own serialized rules, percentages and references. Change a probability, assign a different prefab or material, or add rules without modifying the corresponding Core setting. Other enum values remain available for custom BiomeSetting assets.

Forest

Shared visual starting point: Green ground, brown forest dirt, woodland props and vivid water with waves and fish silhouettes

Tropical

Shared visual starting point: Bright vegetation, sandy paths and turquoise water

Snow

Shared visual starting point: Pale ground, snow-covered trees, cold currents and coastal ice

Autumn

Shared visual starting point: Copper foliage, woodland ground and drifting bank leaves

Shared artwork

The initial references point to **Core/Art**. Models, meshes and textures stay shared. To customize a material's values, duplicate that material into your project and assign the copy. To customize a model prefab, create a project-owned variant. Editing an Art source itself changes Core content; assigning a replacement in a Starter setting does not.

Core updates preserve Starter setting files, while updating shared Art when included in the update. Explicitly reimporting Starter can replace selected Starter files. See Starter installation and ownership.

Image-import material channels

Starter image palettes map `Grass_`, `DirtForest_`, `DirtPath_`, `Water_` (seabed) and `Env_` (rock) to their terrain material channels. An explicit layer material overrides the biome material. The supplied palette and source image are demonstration assets inside Examples.

Compatibility and limitations

Announced V1 scope

The release targets **Unity 6000.2 or later** with **Universal Render Pipeline (URP)**. Unity 6000.2 uses **URP 17.2.0**; select the URP version supplied for later Unity Editors. The recorded reference Editor is **6000.2.13f1**. Unity documents this Editor-matched package policy in its URP requirements.

The runtime platform targets are **Windows, Android and iOS**. The publisher also reports the tool working in the **macOS Editor**. This editor-host report does not add macOS Player to the announced runtime platforms. Linux, WebGL, other runtime platforms, Built-in and HDRP are outside the V1 support scope.

Recorded verification

Unity 6000.2.13f1 / URP 17.2.0 / Windows Editor

Recorded status: Native generation, saved-content, prefab, Editor and rendered checks are recorded.

Unity 6000.4.1f1 / URP 17.4.0 / Windows Editor

Recorded status: Targeted shader checks and actual four-biome captures in Forward and Deferred are recorded. This is not a full-package certification.

macOS Editor

Recorded status: Working use reported by the publisher; no new automated macOS run is claimed here.

Android / iOS devices

Recorded status: Announced runtime targets; physical-device validation is not recorded in the supplied release evidence.

Final exported archives and Players

Recorded status: Require results tied to the exact submitted archives and target devices. Editor checks do not certify these files.

The eight shaders have Windows Direct3D 12 Editor rendering checks in Universal Renderer Forward and Deferred. Historical Built-in/Windows checks remain historical evidence. Later Unity versions belong to the target range, while the verification table lists the exact tested configurations. No new build or archive export is performed by this documentation update. See URP setup.

Included content

Core supplies the generator, Editor tools, shaders, manuals and all shared Art: **Forest, Tropical, Snow and Autumn** source models, textures, materials, configured prefabs and reference biome settings. The optional Starter supplies Examples and independent editable biome settings linked to Core/Art. Other enum values remain available for user-authored BiomeSetting assets; no presets or art for those extra types are advertised as included.

Two scenes demonstrate procedural and color-coded image generation. Eight pre-generated map prefabs cover 1v1, 2v2, 3v3, four-player FFA and six-player FFA, with a full MapGenerator definition and a portable saved recipe on each root. The source models stay shared. Gold/oil resource cubes are replaceable gameplay placeholders. See Starter

installation.

The tool exposes terrain and logical grid data. Units, combat, networking, economy, fog of war, team assignment and navigation integration belong to the host game. The MapEditor module is outside this release; color-coded image import remains included. No external authoring-tool account or subscription is required at runtime. A* Pathfinding is optional and is not included.

Water, vegetation and saved content

The four water styles belong to the four supplied biomes. Add StylizedWaterDepthCamera to a gameplay camera and enable the URP depth/opaque inputs. Copy Depth Mode must be **After Opaques** or **Force Prepass** for transparent water. Rendered vertex waves leave the saved CPU geometry, colliders and strategic water levels unchanged. See Water rendering.

Terrain retains authored stylized lighting with URP main-light shadows and fog. It is not a replacement for a project's full PBR lighting stack. Assign compatible materials when integrating custom lighting. Legacy pipeline branches remain in the source without expanding the announced V1 support.

MapDataContainer stores supported prefab placements and variants compactly. Visible ordinary prefab-linked objects are reconstructed in Edit Mode; GPU instancing is restricted to Play Mode. Packing preserves the shared materials, source models, layers, transforms, ground tint and supported collider state. Unsupported gameplay overrides remain ordinary saved objects. Procedural terrain and water meshes use EmbeddedMeshFilter for persistence. See packed maps and grass details.

Runtime and performance

- Map sizes range from 30 to 200 cells per side. Small maps can lack room for requested expansions; the generation diagnostics report that capacity limit.

- Runtime generation is synchronous. Generate during a loading phase for larger maps and use ProceduralMapGenerationResult to initialize the host game.

- Tune map size, props, grass and shadows for each target device. Platform targeting is not a fixed frame-rate or low-end mobile performance guarantee.

- PNG screenshot capture requires a graphics device. Headless generation does not provide rendered screenshots.

- Image-import textures need Read/Write enabled. Terrain, water and collision share fitted contours and coastal overlaps without changing logical water coverage.

- The module declares standard Unity image-conversion, JSON, physics and IMGUI modules. URP must be installed/configured for the announced release visuals; no cross-version URP pin is added to the Core manifest.

Optional integrations

- A* Pathfinding is detected through reflection when installed; it is not required.

- SOROLIF_RTS_INTEGRATION is disabled by default and requires the matching host framework.

- SOROLIF_MAPGEN_VERBOSE_LOGS enables detailed generation diagnostics.

Runtime API

This page covers the public entry points normally needed by a game. The generation subsystems remain public for advanced integrations, but most games should interact only with `MapGenerator`, `MapDefinition` and `ProceduralMapGenerationResult`.

Generate a map

Add `MapGenerator` to a GameObject, configure its `Definition`, then call one of the following methods:

```
using Sorolif.RTS.MapGenerator;
using UnityEngine;

public sealed class MatchMapBootstrap : MonoBehaviour
{
    [SerializeField] private MapGenerator generator;

    private void Start()
    {
        generator.Definition.UseRandomSeed = false;
        generator.Definition.Seed = 12345;

        if (!generator.TryGenerateRuntimeMap(
                out ProceduralMapGenerationResult result))
        {
            foreach (MapGenerationIssue issue in result.ValidationIssues)
                Debug.LogWarning(issue);

            return;
        }

        foreach (Vector3 spawn in result.PlayerSpawnWorldPositions)
            SpawnPlayerAt(spawn);
    }

    private void SpawnPlayerAt(Vector3 position)
    {
        // Connect this position to your own player-spawning system.
    }
}
```

`TryGenerateRuntimeMap` returns `false` for a blocking validation or generation failure and always assigns a result. Unexpected exceptions raised by a client prefab, material or integration are converted into a `generation.exception` error. The available terrain remains visible as a preview, and the exception is written to the Console as a warning with its stack trace. Recoverable generation failures do not trigger Unity's Error Pause. `GenerateRuntimeMap` returns the same result object when a Boolean return value is not needed.

Procedural bridge, plateau-access and player-land-route audit errors do not stop the remaining visual stages. Mountains, terrain/water, cliffs, biome props, altitude, packing, cleanup and optional screenshots still run. Missing or invalid bridge decks are skipped; invalid custom deck instantiation is reported as `bridges.prefab_invalid`. A completed visual map can therefore still have `Succeeded == false` when its gameplay connectivity is invalid.

Scene gizmos remain available on this preview and after scene reload. They use serialized visual positions belonging to the retained map, without exposing a failed gameplay grid or spawn lists. Access the independent root component through `result.MapInformation`; its `ShowGizmos` and `ShowLabels` properties control the markers. See Generated Map Information. Strategic choke roads also render from their original route plan rather than depending on a successful inter-player route audit.

For other failures, the generator finishes a terrain/water preview from the available grid, keeps the previous map if necessary, or creates a simple terrain with a collider if neither is available. Unsafe bridges are not placed. Inspect `result.PreviewRoot` to access this visible output. The preview is not a validated gameplay map: `Succeeded` remains false, `MapRoot` and `Grid` remain null, and no player spawns or gameplay score are returned. Fix the issue and regenerate before starting a match or saving a generation recipe.

Generation replaces only the hierarchy previously created below that generator. Keep custom gameplay objects outside the generated root, or spawn them after generation completes.

Select a biome explicitly

Assign `MapDefinition.BiomeOverride` when runtime code or a reusable prefab must select one exact `BiomeSetting`. Assign this reference explicitly; direct material and prefab fields on the definition still take final precedence. Leave it null to use only local settings. Biome enums filter Editor choices and do not load assets at runtime.

The included grass fallback library is `Art/Prefabs/Grass/GrassPrefabs.asset`; assign it to `MapDefinition.GrassPrefabs` when constructing definitions in runtime code without grass placement rules. Grass materials live in `Art/Materials/Grass`, and water masks in `Art/Textures/Water`. The Editor prepares missing grass fallback references on processed scene copies before Play Mode or Player scene processing; runtime-created definitions use the references you supply. For new runtime materials, pass texture references to `StylizedGrassPresets.Apply(material, biome, bladeTexture)` or `StylizedWaterPresets.Apply(material, biome, forestWaveFoam, forestCaustics)`. Existing assigned water masks are preserved.

Generation result

`ProceduralMapGenerationResult` exposes:

- `Succeeded`: whether a complete usable map was created.

- `ValidationIssues`: stable issue codes, messages and warning/error severity.

- `MapRoot`: root Transform of the editable generated hierarchy.

- `PreviewRoot`: visible generated output on both success and failure; equals `MapRoot` on success, and can contain an incomplete or retained map on failure.

- `IsPreviewOnly`: true when the last attempt failed and the visible output is only an error preview.

- `Grid`: the logical `MapGrid` used to build meshes and gameplay metadata.

- `MainBaseCells`, `PlayerExpansionCells`, `ContestedExpansionCells` and `ChokePointCells`: strategic grid coordinates.

- `PlayerSpawnWorldPositions`: spawn positions already converted to world space using the configured cell size and terrain height.

- `GameplayScore`: nullable strategic diagnostic from 0 to 100 for procedural maps. It measures base spacing, expansion fairness, rush safety and ground connectivity; it is not a visual-quality rating. It remains null when the layout was not scored, such as an image import without strategic bases.

- `ScreenshotPath`: absolute PNG path when automatic capture was enabled and succeeded; otherwise null or empty.

The generator exposes the same latest value through `LastGameplayScore`. `HasGeneratedPreview` reports whether terrain or water remains visible; `HasGeneratedGrid` requires a successful generation. Diagnostics keep their Error severity in `ValidationIssues` and the Inspector even when logged as warnings to allow generation to recover without pausing the Editor.

Read gameplay cells

`MapGrid.GetCell(x, y)` returns null outside the map. A `GridCell` contains its height and semantic flags such as `Walkable`, `Buildable`, `IsWater`, `IsMountain`, `IsRamp`, `IsMainBase`, `IsExpansion`, `IsChokePoint` and resource markers.

```
GridCell cell = result.Grid.GetCell(x, y);

if (cell != null && cell.Walkable && !cell.IsWater)
{
    // Use the cell in your own placement or navigation layer.
}
```

The generator does not impose a particular pathfinding, building or unit system. This keeps the returned grid portable. The optional A* Pathfinding integration is detected only when that package is already installed.

Validate before generation

Use the same validator as the Inspector when building a custom setup screen:

```
foreach (MapGenerationIssue issue in
         MapGenerationValidator.Validate(generator.Definition))
{
    Debug.Log(issue.ToString());
}
```

Errors block generation. Warnings describe optional or visual configuration that can safely use a fallback. Validation also rejects `NaN` and Infinity in active numeric setting groups before they can create invalid vertices. A failed validation keeps visible preview geometry available.

Save a reproducible recipe

`MapGenerationSaveData` stores the definition, fixed seed and strategic positions. After a successful generation:

```
generator.CaptureCurrentGenerationToSerializedMap(saveData);
```

To apply and regenerate the recipe:

```
generator.ReloadSerializedMap(saveData);
```

The save data stores a recipe rather than duplicating generated meshes. Keep the source image, palette, biome, materials and prefabs referenced by the definition available when reloading.

Included runtime example

UPM installations expose `Runtime Generation Example` in Package Manager's Samples section. The classic Asset Store package contains the same editable script under `Examples/Runtime Generation`.

Generated map information

English | French version

Every generated map root receives a **Generated Map Info** (`GeneratedMapInfo`) component. Select that root in the Hierarchy, or click **Select Map Information** below the generator's generation button.

The component stores the information for the visible map. Changing the generator's recipe does not change this snapshot. Regenerate to capture new settings.

Captured information

- Final seed, biome and optional biome preset, procedural/image generation mode.

- Map size, cell size, requested player count and generated start count.

- Player-start symmetry and effective full-map symmetry.

- Water layout, forest and mountain zones, plateaus, bridges and altitude settings.

- Whether optional Minimap and Aerial outputs were enabled.

- Gameplay score when available, generation diagnostics and strategic locations.

Settings are read-only in this Inspector. The saved biome preset is an asset reference; the component does not duplicate the preset, the complete generation recipe or the logical grid. Continue to save recipes through the generator when you need to reproduce or edit a map.

Strategic markers

Enable **Show Gizmos** to draw the map's locations. **Show Labels** adds their short identifiers. Both switches are saved with the map; regeneration retains their values. Unity's global **Gizmos** switch in the Scene view must also be enabled.

`P`

Location: Player start

Shape: Green sphere

`N`

Location: Natural expansion

Shape: Yellow sphere

`E`

Location: Other player expansion

Shape: Yellow sphere

`CE`

Location: Contested expansion

Shape: Yellow wire sphere

`CP`

Location: Choke point

Shape: Red cube

The Inspector lists counts and, under **Positions (map local space)**, the captured positions and known player owners. Positions include the generated terrain/water height and follow the map root when moved, rotated or reparented. These are Editor markers; they do not add rendered objects or gameplay colliders.

Saving and runtime access

The information and marker positions are serialized on the generated root, so they remain after saving/reloading the scene. If you remove the generator object, first move the generated map out of its hierarchy to retain the map itself.

```
if (generator.TryGenerateRuntimeMap(out var result))
{
    GeneratedMapInfo info = result.MapInformation;
    Debug.Log($"Seed: {info.Seed}, biome: {info.Biome}");
    foreach (GeneratedMapMarker marker in info.Markers)
        Debug.Log($"{marker.Label}: {info.GetWorldPosition(marker)}");
}
```

You can also use `mapRoot.GetComponent<GeneratedMapInfo>()` without keeping a reference to the generator. The component has no dependency on the generator remaining in the scene or on an Editor/MCP connection.

Previews and older scenes

A failed generation can leave a **Preview only** map. Its information describes the retained geometry and shows the diagnostics. A retained previous map keeps its original seed and markers; it does not inherit the failed attempt's settings. An emergency terrain preview cannot provide a reliable generation snapshot. Always check `result.Succeeded` before treating output as a validated gameplay map.

For a map saved before this component existed, **Restore Map Information (legacy map)** on the generator transfers its saved gizmo positions to the root. The original seed/settings and the natural/other expansion distinction cannot be reliably recovered from those positions. The Inspector identifies this state; regenerate once to capture complete information. The generator no longer draws these markers itself.

Pack Content: compact map storage

English | French version

The generator automatically adds the module's **MapDataContainer** to `Env/Props`, then assigns its **3D instance parents** before packing:

- `Env/Props`, `Env/Cliffs`, `Env/Bridges` and `Players/Nature`;

- other parents containing instances identified by `GeneratedPrefabSource`, including the border parent and custom groups.

Packing runs after model placement, bridge placement and altitude application. It stays enabled when **Enable Props Generation** is disabled: cliffs, bridges and resources may still be present.

What is stored

Pack Content stores prefab/material asset references, supported hierarchy variants and a compressed placement block. Positions, rotations, scales and colors retain their float values without rounding. Names, child layers, active states, materials, LODs and grass root colors are preserved. Colliders are reconstructed from unchanged prefabs.

Visible models are reconstructed when the scene opens and when the game starts. These preview objects are not saved a second time in the scene or build. Source assets remain shared and unchanged.

In the Editor, reconstructed models retain their source prefab connections, including nested model prefabs. **Unpack Content** keeps those connections and restores editable ordinary instances. Generated transforms, materials and selected alternatives are instance overrides, retained through scene reload and model reimport.

Packing is deliberately conservative. Objects with gameplay scripts, modified colliders, meshes differing from their prefab, nonpersistent materials or other unsupported state remain stored normally. This also applies to objects referenced by other scene objects. Procedural ground and water meshes continue to use their existing storage system.

Total savings depend on the map's balance of repeated prefabs and unique meshes. The component counter shows the number of stored models and compressed block size; that size excludes variant metadata and terrain.

Editing a packed map

The Inspector first shows **Editable scene objects**, **Packed content**, or **Mixed selection** for multiple selection. Stored instance count and **Placement block** describe packed content; the displayed size is not the total scene size. The main action becomes **Unpack Content** when all selected components are packed.

3D instance parents lists parents with buttons to locate them in the Hierarchy. Empty or duplicate rows are reported without changing the list automatically. The list is locked while any selected component is packed. **Rendering** contains the GPU setting, separate from disk storage. **Runtime diagnostics** counters refer to the last camera processed by this component during gameplay. **How to use** explains the steps and opens this guide.

1. Select `Env/Props`, then **MapDataContainer**.

2. Click **Unpack Content (make editable)** to restore ordinary objects.

3. Edit objects or the parent list.

4. Click **Pack Content**, then save the scene.

Packed preview components are locked to prevent edits that would not be saved to the block. Their GameObject roots stay active and do not receive the NotEditable flag; a locked preview must not be mistaken for a disabled tree. Unpack Content restores component editability. Before changing a source prefab's hierarchy, unpack the map; alternatively, regenerate the map after modifying the prefab. If a reference or packed content becomes unreadable, stored data is retained and unpacking fails without deleting it.

GPU instancing controls rendering of eligible decorations only. Cliffs retain their physical objects. Bridges always keep ordinary renderers and are excluded from GPU batches, including after packing and reconstruction. Disk compaction still applies to compatible bridge prefabs. Disabling instancing does not unpack the scene: ordinary renderers take over. During runtime generation, live objects are retained and only GPU batches are captured; disk compaction applies to maps created in the Editor. No MCP connection or external project dependency is required.

For instanced trees, the root and every collider branch stay active. Only pure visual child branches are deactivated; renderers sharing an object with physics are disabled individually. Turning GPU rendering off restores their previous states. World-object visual yaw is retained independently of collider rotation through packing, scene reload and runtime reconstruction. Different yaw angles reuse the same mesh and packed model variant. Older packed maps remain readable; regenerate them to adopt the new placement behavior.

GPU batches activate only in the playing world, including standalone builds. Edit mode and Prefab Mode always use ordinary renderers, even when **GPU instancing** is enabled or `RebuildInstAndGpu()` is called directly. Leaving Play Mode restores the authored renderer, LOD and child active states, including with scene and domain reload disabled. To edit packed instances, use **Unpack Content**; disk packing remains independent of GPU rendering.

English quick guide

The generator adds the included MapDataContainer to Env/Props, registers every generated prefab parent, then calls Pack Content after final placement and altitude. This also covers cliffs, bridges and resources when props are disabled.

Editor packing stores shared prefab variants and compressed full-precision placements. Visible previews and colliders reconstruct on scene load and in builds. Unsupported modifications, gameplay components and external scene references retain ordinary saved objects. Procedural terrain remains unchanged.

Use Unpack Content before editing instances or changing prefab hierarchies, then Pack Content and save. Regenerating a map also refreshes its packed data. GPU instancing has an ordinary renderer fallback and is independent of disk storage.

The Inspector separates storage status and actions, prefab parents, rendering, runtime diagnostics and help. Its placement block size excludes prefab metadata and terrain. Empty or repeated parent rows are reported without automatic edits. Runtime counters describe this component's latest camera callback only.

Optional outputs and project integration

English | French version

Minimap and Aerial generation are disabled by default. In the Simple Inspector, open **Output** and enable **Generate Minimap Layer** or **Generate Aerial Layer** only when your game needs them. Expert mode groups these switches and project conventions under **Optional outputs and project layers**. Regenerate the map after changing settings, including when removing outputs from an older map. Procedural generation, image import and saved-recipe regeneration use the same switches.

Minimap

`EnableMinimapGeneration` creates `MeshGround/WaterMinimap` when water exists. This is a separate rendering copy of the water surface, without colliders; a minimap camera can render it together with the normal terrain. It does not create a camera or a minimap UI. Disabling it leaves normal water, its shader and physics intact.

`MinimapLayerName` chooses an existing Unity layer, initially `BackgroundMinimap`. A missing or empty name falls back to the configured water layer. `WaterMinimapMaterial` overrides its material; otherwise the biome's minimap material is used. For a custom setup without a biome, assign Water Minimap Material explicitly. Contour refresh and altitude keep the copy aligned with the final water surface. Embedded meshes retain those final heights after scene reload.

Aerial

`EnableAerialGeneration` creates a hidden mesh and MeshCollider at `MeshGround/AerialGround`. `AerialClearance` sets the minimum clearance above ordinary ground, water and bridge decks, in world units; its default is **3**. Cell Size and generator scale are accounted for. Mountains intersect the surface: rock above the plane remains an obstacle through its existing collider. See Aerial navigation.

`AerialLayerName` chooses an existing layer, initially `Aerial`. A missing or empty name falls back to `Ignore Raycast`. The renderer stays hidden and the collider uses the existing adaptive simplification. Enabling the output does not bake navigation or configure flying units.

Project layers

Names select entries in Unity's **Tags and Layers**; generation does not create or rename project layers.

`GroundLayerName`

Initial name: Ground

Missing/empty fallback: Default

Applies to: Ground, paths and plateau surfaces

`EnvironmentLayerName`

Initial name: Environment

Missing/empty fallback: Default

Applies to: Mountain rock, seabed, cliffs and altitude side walls

`WaterLayerName`

Initial name: Water

Missing/empty fallback: Default

Applies to: Normal water surface and collider

`BridgeLayerName`

Initial name: Walkable

Missing/empty fallback: Configured ground layer, then Default

Applies to: Bridge roots and all children, including inactive colliders

`MinimapLayerName`

Initial name: BackgroundMinimap

Missing/empty fallback: Configured water layer

Applies to: Optional water rendering copy

`AerialLayerName`

Initial name: Aerial

Missing/empty fallback: Ignore Raycast

Applies to: Optional flight collider

Explicit image-palette layers retain priority. Per-prop rule layers retain their controls. Ramp surfaces share the map Walkable layer (`BridgeLayerName`), with `GroundLayerName` as fallback. The hidden legacy biome `RampLayerName` is ignored. Prop placement automatically includes Default, TransparentFX and the configured ground/environment/bridge layers. `AdditionalPropPlacementLayers` adds other collider layers without replacing this mask or changing prop instance layers.

Navigation adapter

`EnableNavigationGraphUpdates` is **false** by default. Enable it only when using a compatible installed `Pathfinding.GraphUpdateScene` API. The optional adapter assigns `WaterNavigationTag` (initially **1**) and `AerialNavigationTag` (initially **2**), each in **0-31**. These are graph node tags, not Unity GameObject tags or layers. Aerial tagging also requires Aerial generation.

No navigation package is required. An absent optional API is skipped; the module still generates its terrain and colliders. Actual movement, graph scanning and navigation baking remain your game's responsibility.

Bridge prefab lookup

Configure **BiomeSetting > Biome Prefabs > River Bridge Prefab** to choose the default deck for that biome. Use a bridge prefab with a non-trigger deck collider. Enable **Has Bridges** on a River map, then regenerate.

The map's **River Bridge Override** (`MapDefinition.RiverBridgePrefab`) takes priority when assigned. Leave it empty to follow the selected biome. For an existing map with an explicit bridge, **Use biome / default bridge** clears that override. Changing biomes then changes the bridge without copying it into the map override. New Starter presets provide their bridge through this biome field.

If neither the map nor the biome supplies a bridge, `EditorBridgePrefabSearchPath` optionally looks up a project-relative prefab in the Editor. Its initial value points to the module's generated Starter bridge. Clear the string to disable lookup,

or set an `Assets/...` / `Packages/...` prefab path. Keep the fallback reference serialized for player builds. Serialized biome references work in players directly; runtime generation never uses AssetDatabase.

Bridge alternatives and barrier colliders

A bridge prefab may use **MapPropVariants**, with inline child models or external prefab alternatives. The map seed and crossing position choose one alternative before fitting. Replanning, reversed endpoints and mirrored crossings keep the same choice. Only that model is instantiated; packing preserves its selection.

Put the walking deck's non-trigger **BoxCollider** or **MeshCollider** on **Walkable** (or the configured Bridge Layer). Keep railings and barriers on separate GameObjects. The generator fits the deck collider to both landings and assigns other physical colliders to the configured **Environment Layer Name**, the same layer used for mountains. It does not use their height or outer bounds to position the deck. Legacy plain bridge prefabs remain supported; named Fence, Barrier and Railing branches are excluded from automatic deck detection.

Create the Walkable and Environment layers in your project. Starter setup creates them when project-layer configuration is enabled; missing custom layers retain the map definition's existing fallback behavior. The validation project's Environment layer is available at slot 10. Regenerate existing maps after updating.

Aerial navigation surface

Enable **Generate Aerial Layer** in Simple > Output or Expert > Optional outputs and project layers, then regenerate. This output is **disabled by default**. When enabled, generation creates a hidden mesh and MeshCollider at `MeshGround/AerialGround`. The mesh follows the final ground and water with the configured **Aerial Clearance** in world units (**3** by default). Mountains do not lift the flight surface above their peaks: the continuous plane intersects their slopes and passes through the massif. Rock protruding above it remains a physical obstacle through the mountain's own collider. Both procedural and image maps use this behavior, including global altitude and adjusted bridge geometry.

The relief analysis still uses a conservative half-cell envelope to catch peaks between vertices. The collider then merges flat areas and regular slopes into large patches; a completely flat open map needs only four vertices and two triangles. Straight height changes use long strips; compatible neighboring rectangles are merged again across partition boundaries. More triangles remain near irregular ground relief. The plane fitting tolerance is 0.35 world units and only raises the envelope. Steep breaks may therefore leave additional clearance. Neighboring patches share all edge vertices and heights to avoid collider cracks. Mountains do not cut holes in this surface or force additional subdivisions around their outline. The mesh keeps the terrain's local alignment and accounts for Cell Size and generator scale. Large irreducible meshes still support 32-bit indices.

Rock-colored foothills do not lift the flight surface. This also applies to mountain uplift in the temporary terrain collision mesh and in renderers merged by material. Under a massif the plane extends the surrounding land height and follows the regional altitude field. Water, bridge decks, plateaus and ramps still support flight at their final height. The mountain's shape and physics layers remain unchanged. See Mountain rendering for the terrain material transition.

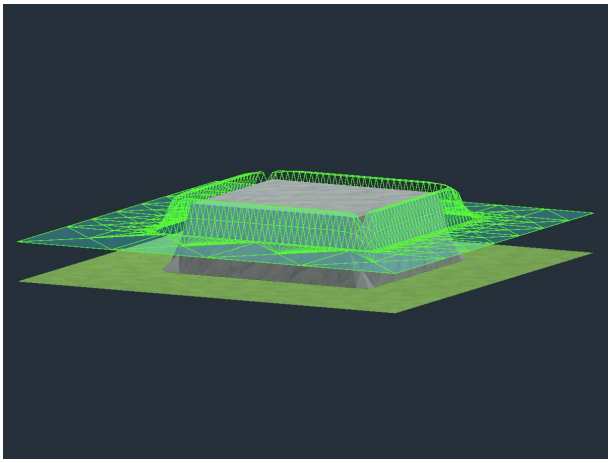
`Aerial Layer Name` selects an existing Unity layer, initially `Aerial`, falling back to `Ignore Raycast`. Its renderer stays disabled. The continuous collider and its final vertices are embedded for scene/prefab reload, including under mountain-covered areas. No navigation package is required. The graph-update adapter is disabled by default and needs an explicit opt-in. See Optional outputs for project layers and integration settings.

If your game bakes separate aerial navigation, include the protruding mountain collision geometry as obstacles as well as the Aerial surface. A collider alone does not assign navigation areas or create avoidance behavior for your units.

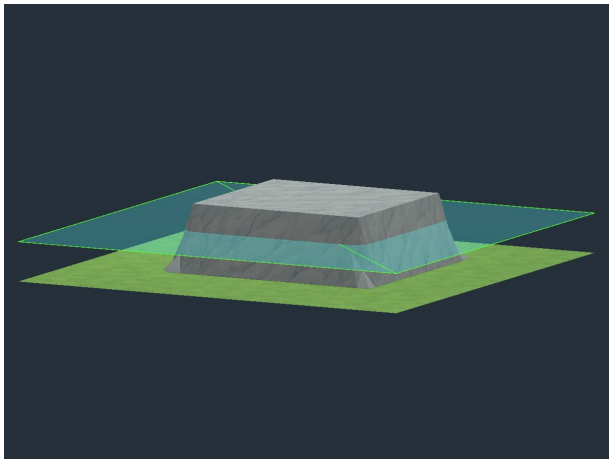
Regenerate an existing map to update this surface. Rebuild your game's navigation data afterwards if your game uses a separate navigation bake. Select `MeshGround/AerialGround` to inspect the generated MeshFilter and MeshCollider.

Mountain intersection

The same generated massif before and after the intersection correction. Cyan shows the actual Aerial surface and green lines show its triangle edges. The renderer is temporarily enabled only for these diagnostic captures.



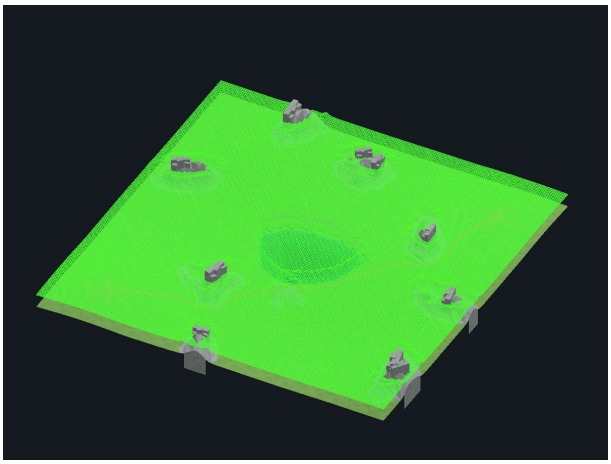
Before



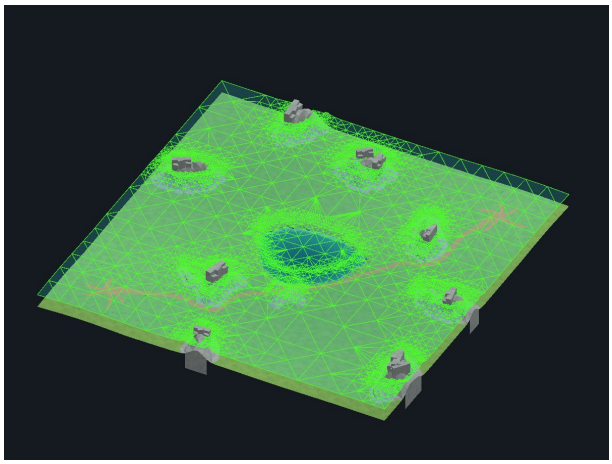
After

Collider simplification

Archived simplification comparison from before the mountain intersection change: the same procedural map, with the old uniform triangulation on the left and adaptive triangulation on the right. The green wireframe displays all edges of the actual meshes; the renderer is temporarily enabled only for these captures.



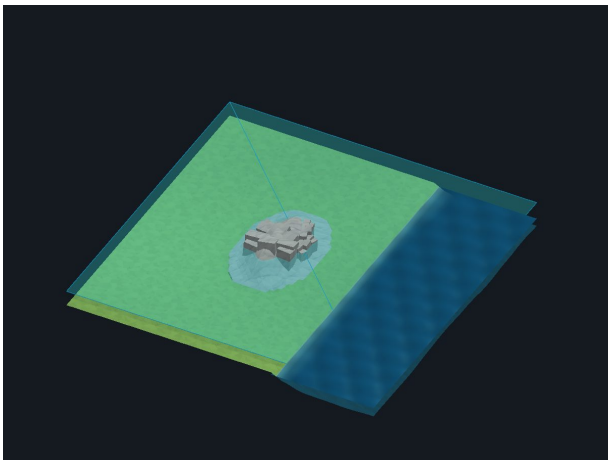
Uniform



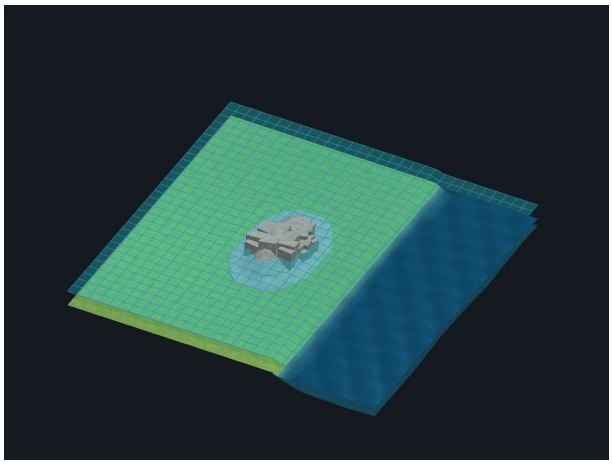
Simplified

Diagnostic examples

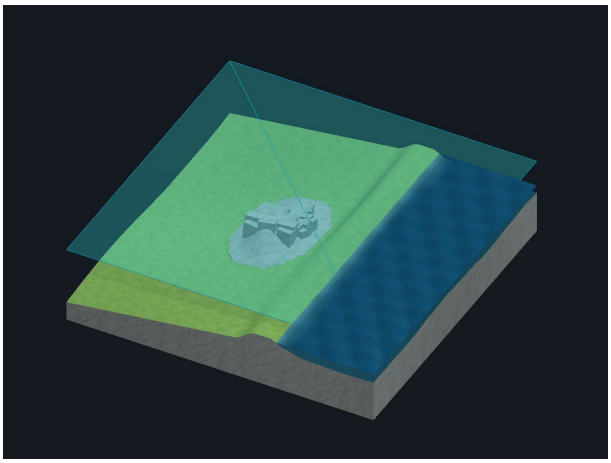
The following native captures temporarily display the actual flight mesh in transparent cyan, with a diagnostic grid. This is a documentation display only; the generated geometry and collisions are not altered.



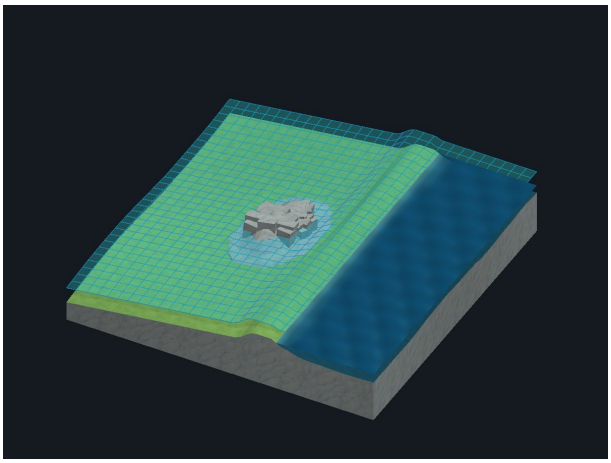
Flat before



Flat after



Altitude before



Altitude after

Both comparisons keep the same input image, palette and camera. The altitude comparison also uses the same height map. Tall mountains block the flight layer instead of moving the entire layer above the highest summit. Water remains level.

Full-map symmetry

English | French version

Full Map Symmetry (`MapDefinition.FullMapSymetry`) is **false by default**. Find it beside **Symmetry** in **Simple > Players and strategy** or **Expert > Map, players and biome**.

When disabled, the original generator is used: **Symmetry** controls player-start placement, while terrain, rivers, mountains, paths, forests and decoration retain their natural layout. The requested start count is not rounded up to complete symmetry groups. Select **None** for free player-start placement as well.

When enabled in procedural generation, **Symmetry** applies to terrain and water, mountains, plateaus and ramps, roads, forest patches, resources, strategic zones and prop placement. Corresponding terrain surfaces and their mesh colliders use the same geometry, including contour smoothing and vertex noise.

None

Relationship: Free generation

Start group size: 1

Horizontal

Relationship: Reflect across the horizontal centre line

Start group size: 2

Vertical

Relationship: Reflect across the vertical centre line

Start group size: 2

Diagonal

Relationship: Exchange the two map axes

Start group size: 2

Rotational

Relationship: Turn 180 degrees around the map centre

Start group size: 2

Quadrant

Relationship: Reflect across both centre lines

Start group size: 4

With **Full Map Symmetry** enabled, choose the mode in **Map, players and biome**. The Inspector displays the number of generated start locations. When the requested player/base count does not fill complete groups, the generator adds empty start locations while keeping the selected symmetry. For example, three players with Quadrant produce four starts; five players produce eight. The Player Count and Additional Main Bases settings are not rewritten.

`MapDefinition.ResolvePlayerStartCount()` predicts this total. The generated `MainBaseCells` and `PlayerSpawnWorldPositions` lists include the extra locations; the game can leave them unused. All start locations receive the corresponding terrain, expansions and resources.

Player expansions follow the same transformation as their base. Neutral zones are completed in symmetric groups, so a requested count can be rounded up. Rivers and mountain features are repeated as complete features to preserve their continuity across symmetry boundaries. Roads and protected passages retain their corresponding accesses; generated bridges keep water below their decks.

Altitude is also symmetric. The generated altitude field, or the samples of a custom heightmap, are combined across the selected symmetry. The source texture is not edited. Standing water stays level and river altitude constraints remain active.

Prop instances reuse the same prefab variant, scale and matching orientation at corresponding locations. Prefab art and materials remain shared. Pack Content retains these instances and their collider settings, and GPU instancing remains available. Shader lighting, wind and texture variation continue to animate normally; symmetry describes the map layout and terrain geometry.

Imported image maps keep their authored layout regardless of this switch. Existing recipes without `FullMapSymetry` use its default value, false. Enable it explicitly to retain complete symmetry when regenerating those recipes. Regenerate after changing the switch or mode; an existing mesh is not modified by changing the setting alone. The setting is saved with generation recipes.

Map altitude

In the Simple or Expert Inspector, open **Map altitude**, enable **Enable Map Altitude**, set **Altitude Min Height** and **Altitude Max Height**, then generate again. The feature is off by default and works in Procedural and Image modes. Turning it off and regenerating restores the previous relief.

The minimum and maximum are additive offsets in map units, before the generator transform's scale. Existing mountains, plateaus, seabed depth and the 0.02 path recess remain on top of this field, so they may exceed these bounds. Equal bounds produce a uniform vertical offset; negative heights are supported.

Automatic height map

Leave **Altitude Height Map** empty. Broad, seeded noise hills are generated and normalized between the minimum and maximum. **Altitude Noise Scale** controls their frequency in cell units; lower values produce broader hills. Start with 0-8 units and 0.035. Keep **Use Random Seed** disabled and **Seed** unchanged for repeatable geometry. Changing altitude does not consume Unity's random stream or change strategic feature placement. Automatic altitude does not impose symmetry on the hills.

Custom height map

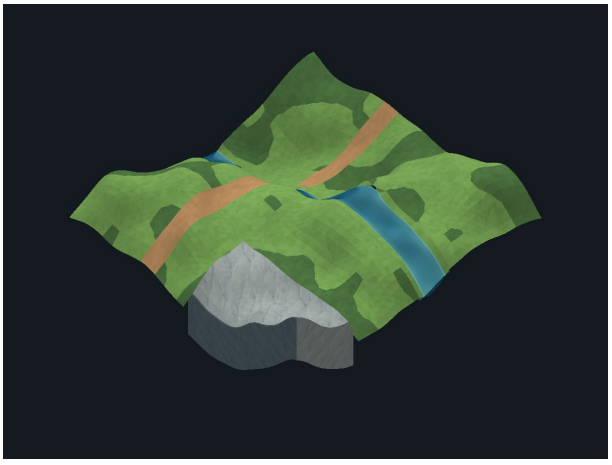
Assign a grayscale `Texture2D` to **Altitude Height Map**. This is a separate texture from **Source Map Image** and does not change terrain classification. In the texture importer, enable **Read/Write**, disable **sRGB (Color Texture)**, choose **Compression: None**, and Apply. Keep sufficient Max Size to retain the source detail. An unreadable texture stops generation with an actionable validation error.

Black is the minimum, white is the maximum, and intermediate grayscale values interpolate between them. Custom values are not auto-normalized: a uniform gray image gives a uniform intermediate offset. The entire image fits the square terrain footprint regardless of its aspect ratio. Sampling is bilinear, independent of the texture filter/wrap mode. By default the bottom-left pixel maps to minimum local X/Z; **Altitude Height Map Origin Top Left** flips the vertical axis. Samples outside the map clamp to its edge, including shoreline overlaps and cliff skirts.

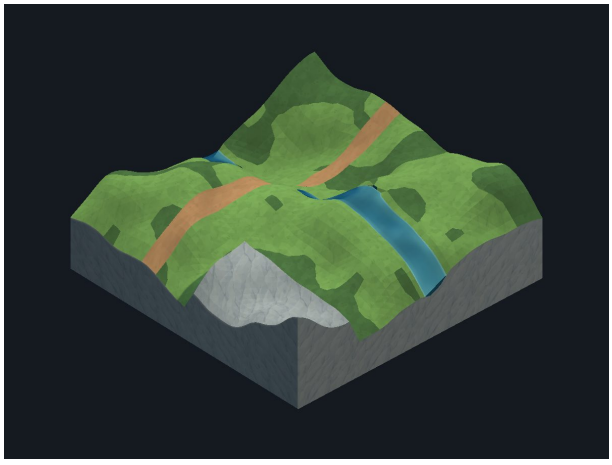
Geometry and runtime data

When altitude is enabled, `MeshGround/MapSides` closes the outer terrain down to a common base. Its rock faces follow the final perimeter, including path recesses, coastal floor extensions and the lower edges of existing mountain walls. Mountain Material is used first, then Env Material or Grass Material. The side mesh has a matching collider and embedded geometry for saved scenes. Disabling altitude and regenerating removes these additional faces.

Controlled terrain example with an automatic height map, a river and a mountain touching the map border:



Open terrain edges



Closed terrain edges

Each freshly generated terrain and water vertex receives the adjusted vertical offset at its final X/Z position, after contour and path smoothing. Mesh indices, shared vertices, UV seams and shader facet markers are retained. Normals and bounds are rebuilt and colliders are recooked from the displaced mesh. Embedded mesh data captures the final geometry, without applying the offset again when the scene reloads.

Grass is placed by MapPropsGenerator on the final displaced terrain colliders, using the same large-then-small passes and shared prefab meshes. Its placement probes follow the terrain bounds and slightly embed accepted footprints to avoid floating roots. See Grass prefabs.

Other props, resources and cliff prefab instances follow the field at their pivot and keep their original shape and orientation. Very steep custom relief can therefore leave large rigid props partially embedded or overhanging. River bridges connect the raised endpoint heights with their rigid deck. The aerial navigation mesh, when **Enable Aerial Generation** is enabled, follows the final ground and water with **Aerial Clearance** world units of clearance (**3** by default). This output is disabled by default. Mountain peaks and rock feet do not lift the flight surface: it continues through the massif at the underlying regional altitude. The rock's own collider remains a physical obstacle where it protrudes above the plane. The mesh and its collider retain this intersection when embedded geometry reloads. A separate game navigation bake must include protruding rock as obstacles. The altitude phase does not deform source prefab or texture assets.

`result.Grid.AltitudeHeightMap` is a generation snapshot. Its `Sample(terrainX, terrainZ)` method accepts X/Z in the local space of `result.MapRoot.Find("MeshGround/Terrain")`, including Cell Size. The field can be sampled without retaining a readable texture after generation. This snapshot includes the water constraints below. `MapAltitudeHeightMap.Create` returns the raw recipe field; use the result grid's snapshot to match the generated geometry around water. `GridCell.Height` and `MountainHeight` include the cell-center altitude; `GridCell.AltitudeOffset` records that offset. `ClampCellHeight` preserves the global altitude while clamping the underlying local feature height. Player spawn positions and saved generation recipes include the new relief.

Water levels and banks

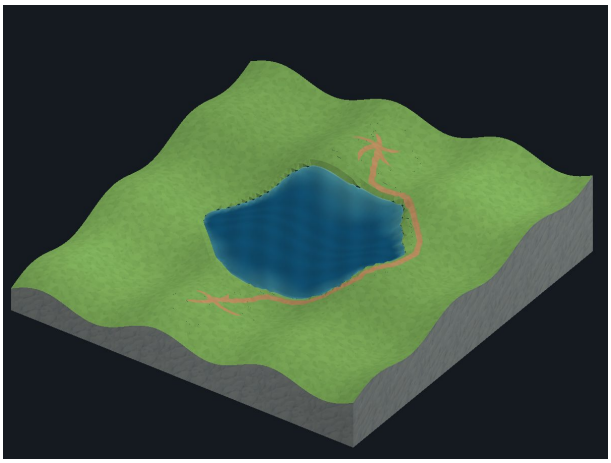
Water constraints apply automatically when both altitude and water are enabled, with automatic or custom height maps. Lakes, seas and closed pools use one horizontal level per connected water surface. Disconnected basins can have different levels. Their level uses the mean height-map offset over that basin.

In the `River` layout, channels reaching separate map-boundary mouths use the lowest mean mouth as their outlet. The surface follows a common distance-to-outlet profile, so branches join continuously and water can descend toward that outlet. The grade is capped at 8 percent. Closed pools without through-flow stay level. This removes the height map's repeated uphill/downhill waves from the water.

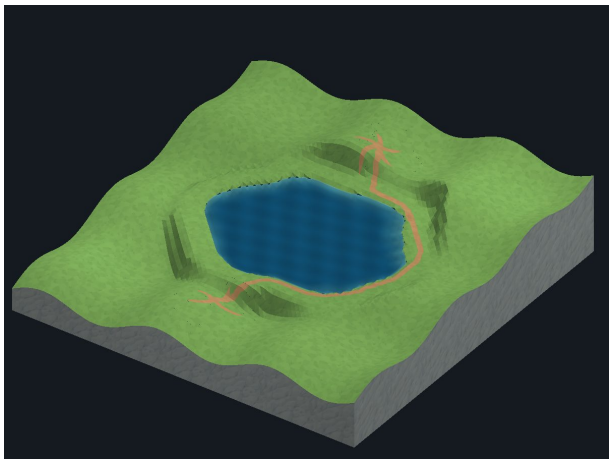
The seabed receives the same correction, retaining its local depth/noise. Banks follow the water over the visible shoreline and three further cells, then blend back into the height map over four cells. This keeps the water attached to its banks. Bridge decks use the adjusted landing heights; water collision and optional minimap meshes retain their final levels when embedded geometry reloads. The source height-map texture is unchanged. Relief far from water keeps its original field.

Controlled custom height-map examples, identical recipes and cameras:

Lake

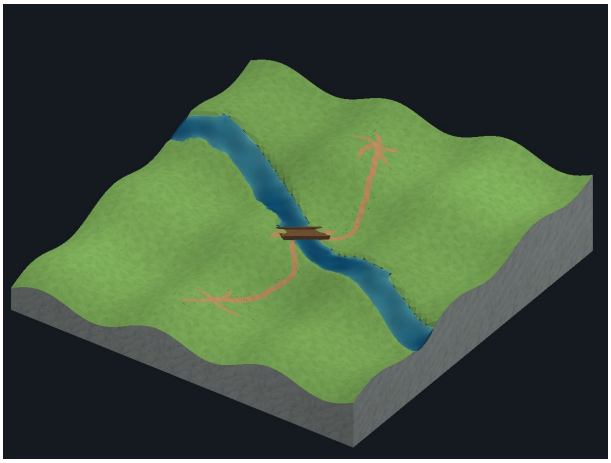


Hilly lake

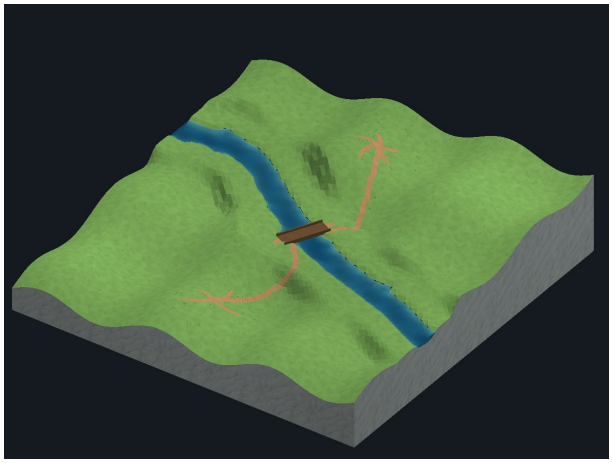


Horizontal lake

River



Undulating river



Downstream river

Strong custom height-map differences can still produce steep banks within the transition band.

This stage retains the strategic flags and layout score. It does not yet recompute walkability/buildability from slope, level base footprints, simulate time-dependent fluid flow or create waterfalls, terrain holes or elevated land bridges. Bake or update your game's navigation from the generated colliders as appropriate.

Biome colors

The Tropical starter palette uses deep jungle greens, olive foliage highlights and pale sand paths. The supplied tropical models use the same palette through their **Sorolif/RTS Props/Atlas Lit** materials. In **Foliage Palette**, adjust **Foliage Deep Green**, **Foliage Sunlit Olive** and **Foliage Palette Strength**. Only green pixels are remapped; bark, rocks and flowers retain their atlas colors. Set the strength to zero to show the original foliage colors.

The base grass remains the shared Medium/Large grass prefabs placed by `MapPropsGenerator`. Imported grass accents still have their occasional 4% rule. Placement, density and geometry are independent of this palette change.

All eight non-Autumn grass materials use colors closer to their biome soil, less color variation, and a soil blend covering the lower 80% of the blades. The sampled soil color fades towards the blade color, with full strength at the foot. The user-edited Autumn material retains its existing settings.

Snow additionally follows the sampled terrain across the blades with **Blade Soil Tint Strength = 0.9**, neutral soil-tint lighting and subtle wind highlights. This keeps its snow grass close to the underlying ground color in Gamma and Linear projects. Other biomes retain their palette-driven blade colors and root gradient.

After updating the module, regenerate the map with **Generate Map** so the grass roots sample the new ground colors. Shared material colors update on existing instances, but previously packed soil colors retain their saved value until the props are generated again. Custom material overrides remain explicit.

Props placement

English | French version

For disk storage of props, cliffs and other models, see Pack Content.

Editor placement keeps scene objects connected to their source prefabs, including inline model alternatives and symmetry copies. Unselected alternatives are removed as instance overrides; their source prefab is unchanged. Resources and borders keep their source connections too. Regenerate older maps to replace objects created without a prefab connection.

Getting started

1. On the map generator, enable **Enable Props Generation**, then select the biome or its explicit preset. Regenerate the map after changes.

2. Edit the biome preset's **Decoration > configs** rules. A nonempty **Prop Configs** list on the map overrides biome rules.

3. Assign a **Prefab**, **Surface prefix** and **Placement chance (%)** to each rule. For example, `DirtForest` targets forest dirt, `Grass` targets grassy ground and `Plateau` targets plateaus. Match case exactly and omit the _ suffix used by generated mesh names. The field's right-hand arrow suggests common surfaces; custom names remain supported.

4. Use **Random yaw** to vary orientation. Other settings are under **Advanced placement / legacy**.

The **MapPropsGenerator** Inspector identifies the active rule source and lets you select the biome to edit its rules. **Local fallback rules** is used only when neither the in-memory rules resolved during generation nor the biome supplies rules. In-memory rules may include biome rules and automatically added default grass. Biome rules are never copied automatically into the component.

On a generated map, the component is a helper: the next generation fills its area and seed from the map definition. To retain a change, edit the generator or the preset supplying its rules.

Understanding a rule

Prefab

Existing field: `prefab`

Purpose: Shared model to instantiate.

Surface prefix

Existing field: `Prefix`

Purpose: Exact surface name; custom names are supported.

Placement chance (%)

Existing field: `tilePercent`

Purpose: Probability from 0 to 100 per eligible location for ordinary props.

Chance / weight (%)

Existing field: `tilePercent` with `IsWorldObject`

Purpose: Highest matching value controls group coverage; all positive values weight the selected model.

Grass placement chance

Existing field: `tilePercent`

Purpose: Independent probability per rule; larger sizes are placed before smaller ones.

Grass detail

Existing field: `IsGrassDetail`

Purpose: Identifies a grass rule and enables the map's height/material style. Legacy Grass/TransparentFX rules are also recognized.

Random yaw

Existing field: `RandomRotation`

Purpose: Random rotation around the vertical axis.

Patch size (tiles)

Existing field: `Scale`

Purpose: Placement pass 1, 2, 3 or 4; 0 means 1; -1 selects the final pass. Does not resize the prefab. Grass sizes run largest first and reserve tiles; 0/-1 infer prefab coverage.

World object

Existing field: `IsWorldObject`

Purpose: World-object placement conventions, overlap checks and relief tolerance.

Instance layer

Existing field: `Layer`

Purpose: Layer of the placed instance's root.

Limit altitude / Maximum altitude

Existing field: `HasMaxAltitude` / `MaxAltitude`

Purpose: Relief limit, evaluated before continuous map altitude during generation.

Legacy scan color

Existing field: `color`

Purpose: Reference color for the standalone Scan By Color action.

World-object rules share one placement roll using the highest matching percentage, then select a model using all matching percentages as weights. For 80 and 50, placement chance is 80%, and the model ratio is 80:50. Other ordinary rules retain their order. A 100% chance does not bypass path exclusions, prefab footprints or slope checks. See Tree alternatives and probabilities, or Grass Details for the separate grass passes.

For **World object** rules, **Random yaw** rotates the visible model while the instance root and collider branches retain their alignment with the map grid. This also applies to tree alternatives. Imported child rotations and offsets are retained beneath the generated yaw. A prefab with a renderer on its collider object receives a separate visual child; its shared mesh is reused. Regenerate existing maps to apply this placement behavior.

Demo biome trees use **DirtForest**, a **1-tile** pass and **Placement chance = 80%**. This increases tree coverage on brown areas without filling them with grass. It remains a probability per location; slopes, protected paths and model dimensions affect final coverage. New wizard presets also use 80%. Refreshing a custom preset preserves its values, including a different density.

Standalone component

Placement area groups the instance parent, area size in tiles, tile size in world units and X/Z origin. **Randomness** groups the seed and reproducible-roll setting for ordinary props; grass always uses its own seed-derived stream.

Advanced: slopes and footprint contains height tolerances under the footprint. **Legacy: color scan and border** contains the image and mask used only by color scanning, plus the optional border box. Existing standalone actions remain in the component's context menu.

English quick guide

Enable props on the map generator, choose a biome preset, edit its Decoration rules, then regenerate. A non-empty map Prop Configs override takes priority over biome rules; the helper's Local fallback rules are used last. The helper shows the active source and provides a button to select the biome asset.

Expand a rule to choose its prefab, exact surface prefix (such as `DirtForest` or `Grass`), placement chance and yaw variation. Grass also uses per-rule chances, placing larger patch sizes first and filling free planting tiles with smaller ones. Advanced settings retain patch pass, world-object handling, instance layer, altitude limit and legacy scan color. Patch size selects the placement pass; it never scales the prefab.

The Inspector keeps serialized names, types and rule order. Opening it does not apply presets, clamp existing values or generate props. Included starter forest tree rules now use an 80% placement chance; refreshing custom presets preserves their authored rules.

Supplied model prefabs

English | Français

The examples demonstrate **placing configured prefabs automatically**. The generator does not rebuild or repair imported models.

From model to generated object

1. Keep the FBX in `Art/Biomes/<Biome>/Meshes` and its textures in `Textures`.

2. Create a prefab with the linked FBX as its visual child. Assign its material and set any offset or scale on the prefab. Add colliders to the root when needed.

3. Reference the prefab in `BiomeSetting.configs`, with a surface prefix and probability. `MapPropVariants` chooses between complete models with the map seed.

4. Select the biome preset and generate. `MapPropsGenerator` places the configured prefabs.

Visual MeshFilters point directly to meshes inside the source FBX. There are no Prepared, Generated or detail mesh copies. Reimporting a model updates its geometry. Geometry, UV and scale issues should be corrected in the authoring tool.

Open a prefab to see its linked model. Its MeshFilter mesh locates the FBX. The `MapPropVariants` Inspector also provides **Source FBX files** buttons. No source-tracking component is required.

Placement and storage

The Snow, Tropical and Autumn model and Starter presets demonstrate trees and small decorations. See Snow models and props placement. Existing probabilities, base-grass filling passes, materials and collider settings remain configured in the presets and prefabs.

Autumn includes five trees in two seeded groups (forest rules 80/50), two bushes (combined 3% rule), flowers/berry variants (1.5%), branches/stumps (4%), two rocks (4%), fallen leaves (4%) and mushrooms (1.5%). Decorations populate Grass and Plateau; paths use only the small rock at 2%. Forest can receive fallen leaves at 4%. The existing two-pass base grass remains separate. Tree roots keep their aligned colliders; vegetation uses the shared wind material and rocks/ dead branches use the shared static atlas material. All visuals reference their FBX meshes directly, including the four alternatives inside the berry FBX.

A tree prefab has a physical root and one linked visual model. World-object rotation affects the visual branch while the root collider stays aligned with the grid. Inline alternatives keep their shared root; external palettes instantiate the selected prefab directly.

`MapDataContainer` packs prefab instances for disk storage. GPU instancing runs during **Play Mode only**. Edit Mode keeps ordinary visible objects and active colliders. Geometry and materials remain shared assets.

Bridges display their source FBX, with a Walkable deck and Environment side barriers. They remain excluded from GPU instancing. Three small separate meshes describe arched deck collision surfaces and contain no visual model copies. **Prepare Biome Bridges** creates missing bridge prefabs and necessary collision geometry while preserving authored prefabs.

Source model corrections

The corrected Snow birch is included in `Snow_BareTrees`. Correct geometry, UV or scale issues in the source FBX; no substitute geometry is generated.

Forest models

`Art/Biomes/Forest/BiomeForestModels.asset` and Starter Forest use all fifteen non-bridge FBX models: five trees and ten decorations. The three existing bridges remain connected. Each model has a prefab under `Forest/Prefabs/Trees` or `Forest/Prefabs/Details`, with its linked FBX and the shared atlas materials.

Two tree palettes populate DirtForest at 80/50: oak/young tree, then birches/dead tree. Grass and Plateau receive grass clumps at 6%, bushes at 3%, plants/ferns at 1.5%, branches at 4%, rocks at 4% and mushrooms at 1.5%. DirtPath only receives small rocks at 2%. The four existing base-grass rules are preserved. Imported grass clumps are decoration, separate from the base-grass system.

New prefab roots, model-placement rules and importers use scale 1. The publisher confirmed the correction of `Biome_Forest_Tree_Birch1_Textured.fbx` on 6 November 2026; the earlier transform warning is resolved. Correct source models in the authoring tool while preserving their dimensions and linked prefab references. The large fern measures 3.35 × 1.84 × 3.23 units; check that this size is intended.

Tree alternatives and world-object probabilities

English | French version

Preview alternatives in the Editor

An instance of a prefab with **Map Prop Variants** automatically displays one alternative when placed in a scene, including ordinary Editor instantiation. Inline alternatives keep their common root and collider. A palette referencing separate prefabs displays one child preview. Repeated refreshes do not add copies.

Select the scene instance and use **Preview Next Alternative** to inspect another model. The selected preview is saved with the scene. Previewing a scene instance does not change its source prefab or consume the generator's random stream.

Author the palette in **Prefab Mode**: all source alternatives remain available for editing there. The automatic scene preview does not run on prefab assets or temporary prefab-authoring scenes.

MapPropsGenerator makes its own choice with the map seed. Generated trees already display that final choice in Edit Mode; inline alternatives that were not selected are removed from the generated instance. The shared root collider and authored model transforms are retained. External alternatives instantiate directly under Props. Compact packing, scene reconstruction and GPU instancing retain the selected model.

Configure varied world objects

On the map's **Prop Configs**, biome **Decoration** rules or the standalone helper:

1. Add a rule for each tree prefab or alternative palette.

2. Use the same **Surface prefix** and **Patch size** for trees competing at the same eligible position.

3. Enable **World object** (`IsWorldObject`) for these rules.

4. Set **Chance / weight (%)** using the existing `tilePercent` field.

5. Keep the existing overlap/layer, maximum-altitude and footprint settings as appropriate for the models, then regenerate.

Matching world-object rules use two seeded rolls:

- The **highest percentage** sets the chance of placing any object.

- If placement succeeds, **all positive percentages** weight the model choice.

For example:

Tree A

Chance / weight: 80

Tree B

Chance / weight: 50

The group has **80% placement chance**. Among placed trees, A receives `80 / 130 = 61.5%` and B receives `50 / 130 = 38.5%`. Before footprint rejections, that means about 49.2% A, 30.8% B and 20% empty positions. This replaces the first-rule bias and cumulative 90% coverage of separate 80% then 50% rolls.

Coverage and model choice use separate seed-derived streams. Adding another model without raising the highest percentage does not raise the density roll. The same seed, rules and terrain reproduce the positions, models, nested alternatives and random yaw.

Matching and placement constraints

Groups only include eligible world-object rules for the current surface and placement pass. Patch size 0 still aliases pass 1. The legacy color scan groups by its matching color instead of Prefix. Missing prefabs, empty palettes and zero/negative/non-finite percentages do not contribute. Finite percentages above 100 are treated as 100 during generation; reading the Inspector does not rewrite stored values.

A successful density roll still has to pass protected-path, overlap, terrain footprint, slope and altitude checks. A rejected model does not trigger repeated rolls until something fits. Ordinary non-world rules keep their existing ordered selection. The world group is evaluated at its first matching rule, once per position/pass. Grass keeps its separate large-then-small planting passes.

Use Props Placement for the complete rule Inspector and Packed Map Content for compact storage.

Snow models

Français

Select **Snow** and the **BiomeSnowModels** or **StarterSnowBiome** preset. Generate the map again to populate it with the supplied models. Existing generated maps keep their saved props until regenerated.

The supplied content includes five complete trees and ten small props. Each FBX remains a complete model, including its attached decorations. `MapPropVariants` selects an alternative using the map seed. Tree colliders remain aligned with the tile grid; random rotation applies to their visuals.

Ancient and young pines

Surface: DirtForest

Tile Percent: 80

Two bare birches and dead tree

Surface: DirtForest

Tile Percent: 50

Existing medium grass

Surface: Grass / Plateau

Tile Percent: 90

Existing large grass

Surface: Grass / Plateau

Tile Percent: 75

Imported grass accents

Surface: Grass / Plateau

Tile Percent: 4

Mushrooms

Surface: Grass / Plateau

Tile Percent: 3

Branches, fallen log, stump

Surface: Grass / Plateau

Tile Percent: 1.5

Rocks and crystals

Surface: Grass / Plateau

Tile Percent: 4

Small rocks

Surface: DirtPath

Tile Percent: 2

The tree percentages follow Tropical: the highest chance sets forest coverage, then 80:50 weights the choice between families. Small-prop probabilities apply to a palette, not separately to every model. The two base-grass passes retain sizes 1 and 4, with rotation only on the small pass. Imported grass is an occasional prop and does not replace the grass filling system. Snow bridges and terrain colors remain configured independently.

Prefabs are in `Art/Biomes/Snow/Prefabs/Trees` and `Details`. Each prefab contains one linked FBX child. Its MeshFilter references the mesh inside the FBX directly; position, rotation and scale adjustments live in the prefab. Trees retain their authored dimensions and use the Tropical collider dimensions, 0.75 × 2 × 0.75. Small decorations are normalized to useful tile-scale footprints. Their proportions are preserved.

The new FBX UVs use `Art/Textures/Biomes_Shared_Cutout_1024.png`, the same atlas as Tropical. The older `Biome_Snow1..4` polygon atlases use a different UV layout and remain available without being assigned to these models. Prepared materials use **Sorolif/RTS Props/Atlas Lit** with world-space wind on trees and grass accents. Rocks, crystals, mushrooms and wood remain still.

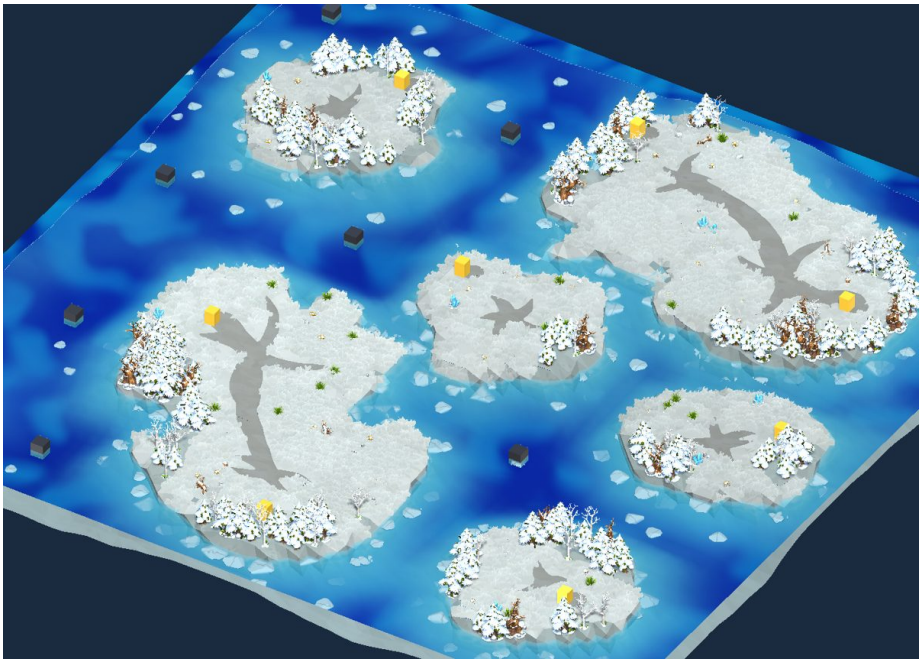
The corrected `Biome_Snow_Tree_Birch_Bare2_Textured.fbx` is included through **Snow_Tree_Birch_Bare2** in **Snow_BareTrees** → **Variants**, alongside the first bare birch and the dead tree. Both Snow presets use this palette. Corrected tree dimensions come directly from the authored FBX/import and prefab transforms; regeneration does not normalize them again.

There is no model-baking or mesh-rebuilding step. Reimporting a FBX updates the geometry used by its prefabs. Select a palette and click a button under **Source FBX files** to locate the model in Project. Inspect and fix source geometry in the 3D authoring tool; the generator places the configured prefabs.

`MapPropsGenerator` performs placement. `MapDataContainer` packs the props for disk storage and enables GPU instancing during Play Mode. Edit Mode retains ordinary visible objects and active tree colliders.

Snow's base grass follows the sampled ground color across its blades (**Blade Soil Tint Strength = 0.9**) and uses gentle wind highlights. Regenerate after changing ground materials to refresh the stored per-instance soil samples. Imported green grass accents remain separate authored props.

Snow water uses a clear azure glacier style: saturated blue depths, icy-blue transparent shallows, soft animated seabed caustics and broken white shore foam. Both presets reference `Snow_Water`; `Art/Materials/Water/WaterOnSnow` provides the same style for direct assignment. An explicit custom water-material override remains authoritative.



Azure water in a generated Snow map

Preview

The model galleries below use URP with the actual prepared prefabs and materials.



Snow tree variants in URP



Snow ground props in URP

Grass prefabs

English | French version

Grass uses the same **MapPropsGenerator** as other props. Each patch is a prefab instance with a shared mesh and material. Grass meshes are neither created nor merged during generation. Instances sit directly under `Env/Props`, with their `GeneratedPrefabSource` reference.

Biome placement passes

Configure placement in **BiomeSetting > Decoration > configs**, as for other props. A nonempty **Prop Configs** list on the map takes priority. The generator processes grass sizes from largest to smallest, regardless of their order in the list.

Example for models that actually cover 3 × 3 and 1 × 1 tiles:

GrassCards_Large3

Prefix: Grass

Tile Percent: 75

Scale / Patch size: 3

Random Rotation: Disabled

Layer: TransparentFX (1)

GrassCards_Medium

Prefix: Grass

Tile Percent: 90

Scale / Patch size: 1

Random Rotation: Enabled

Layer: TransparentFX (1)

Large patches are placed on the grid without random offsets. Each placement reserves its tiles. The small-patch pass then fills only unoccupied tiles, using the rotation set by its rule. Multiple variants can share a size: they are tried in biome order, each with its own probability, until one placement succeeds.

Tile Percent is a probability, not a relative weight between sizes. A zero value is never used. **Scale selects the area to reserve** without resizing the model. Match it to the prefab's actual footprint: the included models currently use **Medium = 1** and **Large = 4**. **GrassCards_Large3** also provides a **3 × 3** footprint at the same height, using shared meshes and materials. A custom 3 × 3 prefab uses **Scale = 3**. Legacy values 0 and -1 infer the size from GrassPrefab, or use 1 without metadata. A patch supplied for four tiles cannot reserve just one tile: reservations always cover at least the footprint declared by GrassPrefab.

Recognizing legacy rules

Prefabs with **GrassPrefab**, **Is Grass Detail** rules, and legacy prefabs named Grass on the TransparentFX layer are recognized as grass. They use the MapPropsGenerator passes without adding a second set of default rules. Rule resolution does not modify biome assets.

A legacy prefab without GrassPrefab or Is Grass Detail keeps its original material and height. **Is Grass Detail** enables the map's style for a custom prefab. An explicitly assigned **Grass Detail Material** on the map takes priority, including for legacy rules. Included prefabs and prefabs with GrassPrefab use the module's biome height and material.

The Autumn material and its preset use the olive palette saved in the editor, with **Root Tint Gradient Height = 0.8** to tint the roots toward the ground color.

The Snow material also uses **Blade Soil Tint Strength = 0.9**. The whole blade follows its sampled terrain color, retaining texture detail and gentle variation instead of a fixed white/green tip. Its wind highlights are reduced to **0.045**. Other biome materials default to **Blade Soil Tint Strength = 0**, keeping their existing palette and root gradient. Without a sampled soil color, the shader uses its Root/Tip Color palette. These settings affect color only; cutout, depth, shadows and wind bending keep their existing behavior.

When no grass rule is configured, the module adds its Medium/Large rules for Grass and Plateau in memory. Any configured grass rule, even at 0%, prevents this automatic addition. Disabling Enable Grass Details also disables recognized legacy grass rules.

Map settings

Enable **Enable Props Generation**, then **Enable Grass Details**, and regenerate after making changes. **Apply dense grass preset (screenshots)** enables both options and resets **Density = 1** and **Height = 0.5 / 0.8**, while preserving the biome, materials and custom rules.

- **Grass Detail Density** multiplies each rule's probability, from 0 to 4, capped at 100%. **1 uses the biome percentages exactly**. A legacy scene set to 3 therefore increases placement chances up to the cap, without creating multiple patches on the same tile.

- **Grass Detail Height** sets the minimum/maximum blade height for the module's stylized prefabs. Only vertical scale changes; ground coverage remains model-defined and follows Cell Size. Legacy prefabs without a style marker keep their height, scaled only by Cell Size.

- **Grass Detail Material** explicitly overrides the shared LOD material. Without an override, legacy prefabs without a style marker keep their materials; the module's patches use the biome material.

Edges and occupied tiles

Planting tiles remain reserved throughout grass generation. A second patch cannot occupy the same tiles. Rotated large patches also reserve tiles touched by their rotated footprint. Reservations reset when the generator clears props.

Ground checks use a slightly inset planting square, **0.25 tile per side**, instead of a circle enclosing every leaf tip. This inset follows the approach used by the old temporary volumes. Blades can meet along neighboring tile edges without stacking whole patches. Intermediate probes also detect narrow paths inside a large patch. For a compact custom model, **GrassPrefab.FootprintRadius** can reduce probe coverage without reducing reserved tiles or extending planting beyond its configured size.

Ground must match the rule's Prefix. Water, mountains, ramps, resources and protected passages remain excluded. Altitude is checked under the planting footprint. Seeded generation uses a grass random stream separate from other props. The ground-tint cache is shared for the pass and released afterward.

With map altitude enabled, MapPropsGenerator places grass after the final terrain colliders have been displaced. Ray origins follow the terrain bounds, including maps above 20 units or below -180 units. Roots use the lowest accepted footprint sample and a small planting inset, so a flat patch cannot float above a gentle slope. Large patches rejected by the footprint altitude tolerance leave their tiles available to the smaller pass. Shared prefab meshes and GPU packing remain unchanged. Regenerate existing maps to update their grass positions.

Brown **DirtForest remains reserved for trees** in the supplied rules. Trees keep an 80% chance per eligible placement. No forest-grass rule is added implicitly.

Included assets

- `Art/Prefabs/Grass/GrassCards_Medium.prefab` and `GrassCards_Large.prefab`.

- `Art/Prefabs/Grass/GrassCards_Large3.prefab`, a 3 × 3 variant at the same height.

- Six static meshes in `Art/Meshes/Grass`: two variants and three LODs.

- Original RGBA texture `Art/Textures/GrassTuft_Original.png`, imported as 2D with mipmaps that preserve alpha coverage.

- Nine materials in `Art/Materials/Grass`.

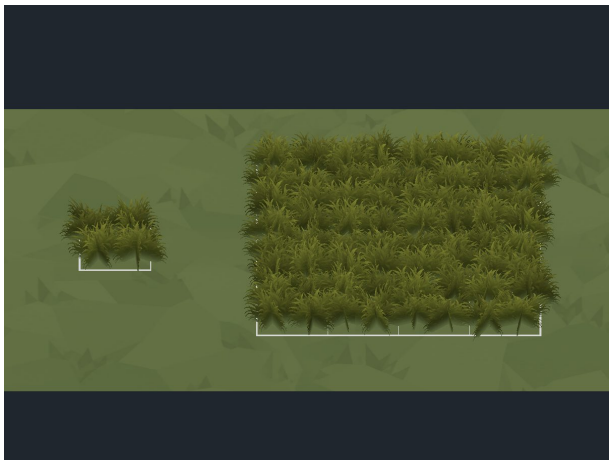
- `Art/Prefabs/Grass/GrassPrefabs.asset`, a library of shared references.

Patches use several oriented, curved cards to create a dense volume. **Medium fills a nominal 1 × 1 unit tile** and **Large a nominal 4 × 4 unit tile** at scale 1. Large distributes 64 small three-card tufts (192 cards in LOD0) across this area, with blades comparable in size to Medium; it does not simply enlarge the same cards. Large's reference height stays below 0.70 units. Tufts follow a slightly offset 0.5-unit grid; edge blades deliberately extend beyond the tile to join adjacent patches. Space centers 1 unit apart for Medium or 4 for Large, without increasing this spacing to match the bounds. Actual geometry therefore extends beyond the nominal area. The protection radius includes this overhang. Cell Size scales these footprints to one or four tiles. Grass Detail Height controls height independently. Renderer bounds include a small additional wind margin; the dimensions above describe geometry at rest.

LOD0/1/2 reduces the number of cards with distance while retaining Large's sixteen sectors. Meshes contain **72/48/8 triangles for Medium** and **1152/768/128 for Large** respectively. QuickStart presets and newly created starter assets include the corresponding rules. Refreshing the wizard preserves existing custom configurations; in-memory fallback rules support older biomes without a grass rule.



Before



After

Ground color at the roots

MapPropsGenerator samples the color of the surface hit by its raycast and stores it on each instance in `GrassGroundTint`. For the module's triplanar ground, it also samples the top texture pattern and global variation. The standard Triplanar material's height grading, slope darkening and diffuse lighting are included. The small texture copy used for sampling is temporary, shared during generation and then released; import settings remain unchanged.

The shader uses a simple color gradient at the foot while preserving the tip palette. **Root Tint Strength** controls intensity (default 1), and **Root Tint Gradient Height** controls the affected blade-height fraction (default 0.38). This changes color only; grass remains alpha-tested with depth writing. It does not sample background objects, scene color or scene depth for this tint. Existing material values are retained under the same `_GroundBlendStrength` and `_GroundBlendHeight` property names. Materials remain shared: color uses a per-instance GPU property and also works with ordinary renderers. MapDataContainer preserves it during capture and reload.

The sampled soil contribution is applied after blade lighting, so a warm scene sun cannot turn blue-grey Snow roots cream. The optional whole-blade soil tint uses the same treatment. Tip palette lighting, wind highlights, dynamic shadows and fog remain active. Regenerate older maps to refresh the stored soil tones.

A custom ground shader using `_BaseColor` or `_Color` provides its main tint. Textures and lighting specific to another shader are not reproduced exactly. If no color is recognized, the grass material's root palette is used. An external instancer must provide the instanced `_GroundColor` property to preserve this transition.

Regenerate the map to update existing instances' footprints and colors.

MapDataContainer and GPU instancing

After generation, **MapDataContainer is placed on Props** and captures prefab references, materials and transforms.

The generator uses the `MapDataContainer` included in this module. It assigns Props, Cliffs, Bridges, Nature and additional prefab groups before **Pack Content**. Bridges retain their colliders and child layers. Compact storage is independent of GPU instancing; objects with unsupported state remain ordinary instances. See Packed Map Content.

The standalone package includes its own `Sorolif.RTS.MapGenerator.MapDataContainer`. Packed maps display reconstructed instances; use **Unpack Content** before editing them, then **Pack Content** to save changes. In Play Mode, the container uses shared meshes and materials for GPU-instanced draws, selects LODs by camera and disables eligible individual renderers. Transforms and other props remain available. Disabling Use Gpu Instancing restores renderers; a GPU without instancing support keeps ordinary rendering. The legacy **Capture & Pack** context menu only refreshes GPU data in memory; **Pack Content** handles compact disk storage. `RebuildInstAndGpu()` remains available for integration code.

Rendering follows parent movement. Models with colliders can be stored compactly while retaining reconstructed physical instances; they are not drawn by the decoration instancer.

Shader

Sorolif/RTS Terrain/Stylized Grass uses the RGBA texture for blade cutout and detail, a Root Color / Tip Color gradient and per-instance color variation. **Wind Direction X Z** is a world-space direction, independent of patch rotation. **Wind Bending**, **Wind Speed** and **Gust World Scale** control the strength, travel speed and spacing of gusts. Gusts sweep along that direction according to world position; neighboring patches share their phase, including when GPU-instanced. Color variation does not randomize wind timing. Roots stay fixed. Alpha Cutoff controls cutout; Fade Start / End controls distance fading. Animation Time = -1 follows the clock; a nonnegative value freezes motion.

Built-in and URP passes are included without a mandatory URP dependency. The shader is double-sided with depth writing and receives shadows. Matching animated/cutout depth and shadow passes prevent background shadows from appearing on foreground blades; URP also includes a depth-normal pass. Included prefabs keep shadow casting disabled, while their depth still occludes background surfaces. UV0 covers the texture; UV2 provides variation and height for included meshes. Fallback Grass Height adjusts wind for external meshes without these UV2 data.

Wind highlights

The upper blades brighten as they lean into a stronger gust, then return toward their normal color when the gust eases. Color uses the same world-space gust as vertex bending, before displacement, so the highlight sweeps with the motion across neighboring patches and GPU instances. It follows Wind Direction X Z, including a reversed direction; it is a stylized color effect rather than a screen-space reflection.

Under **Wind Highlights** on the grass material:

- **Wind Highlight Color** controls the highlight tint. Existing Gust Highlight colors are retained through the same `_WindTint` property.

- **Wind Highlight Strength** controls intensity (default 0.4); 0 disables it.

- **Wind Highlight Start** sets the blade-height fraction where the smooth tip gradient begins (default 0.45).

- **Wind Highlight Contrast** concentrates the effect in stronger gusts (default 2).

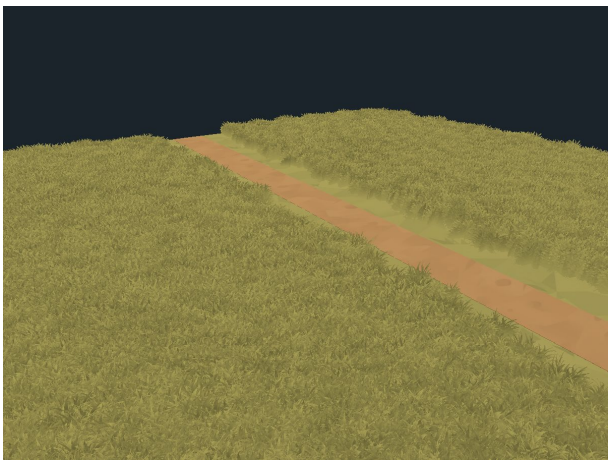
The root soil tint, alpha cutout and shadow/depth passes are retained. Highlights receive the same lighting and shadows as the blades and stop when Wind Bending is zero. Existing material palettes are preserved; the shader

defaults enable the effect without reapplying a biome preset or regenerating grass.

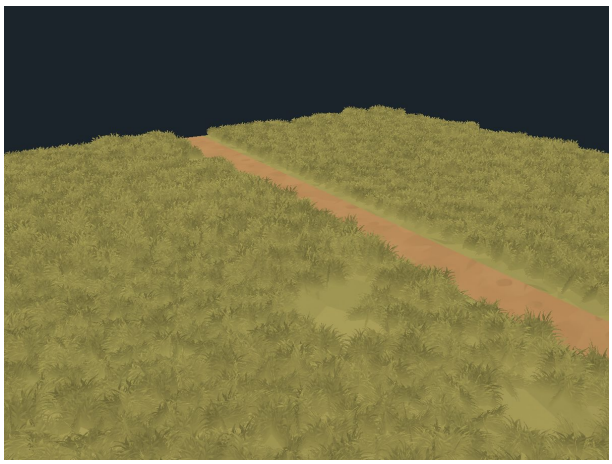
Dense prefabs increase overdraw and instance counts. Adjust density, LODs and distances for the target GPU.

Native screenshots

Passes by patch size

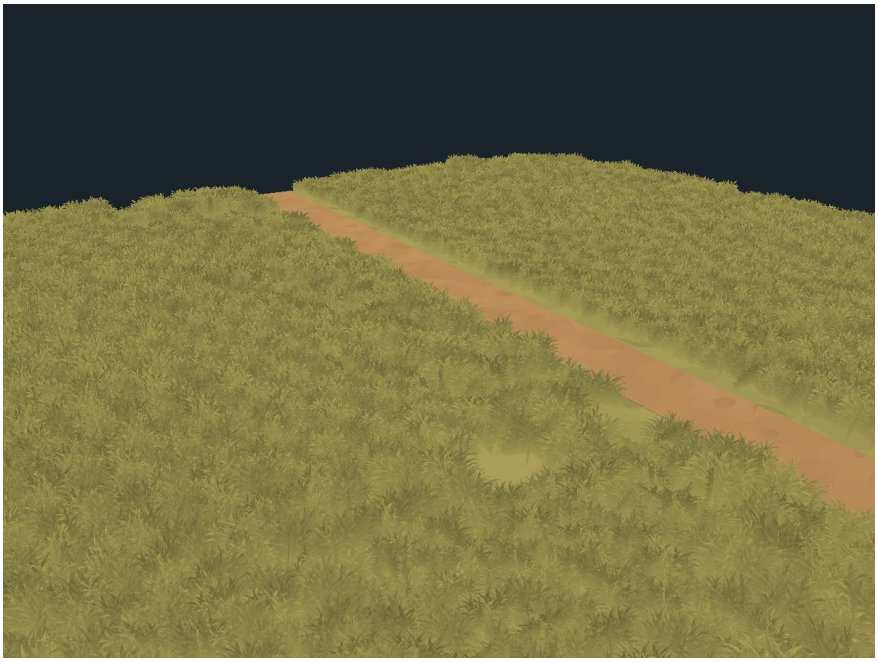


Before



After

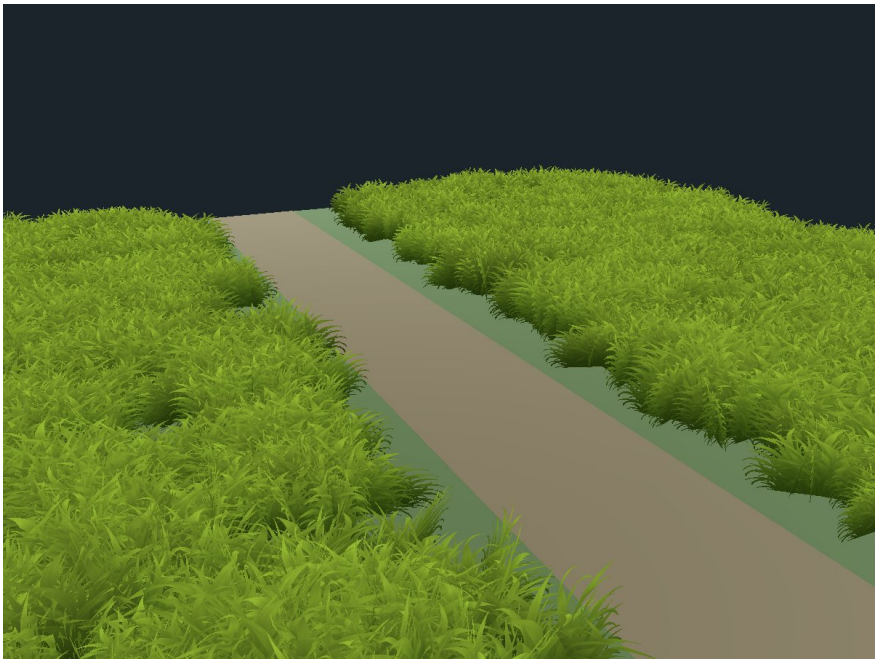
The validation scene compares the same meshes and path. The following view uses a large 3 × 3 rule at 75% without rotation and a 1 × 1 rule at 90% with rotation, without the new Is Grass Detail marker.



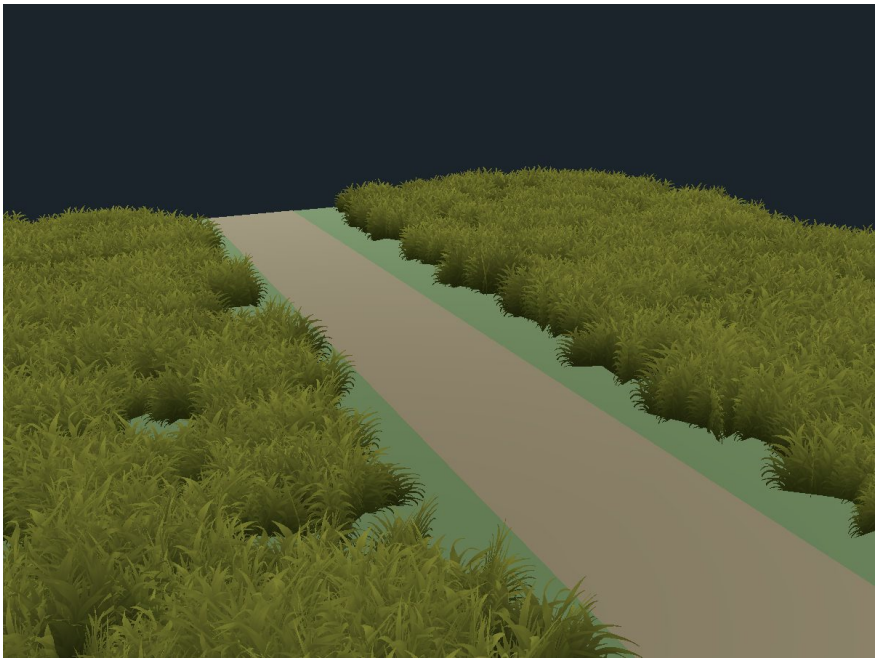
Legacy size 3 and size 1 rules

The following model, placement and color captures remain historical rendering examples. The current configuration uses the size-based passes described above; patch counts do not indicate ground coverage when patches overlap.

Other biomes and models



Forest patches



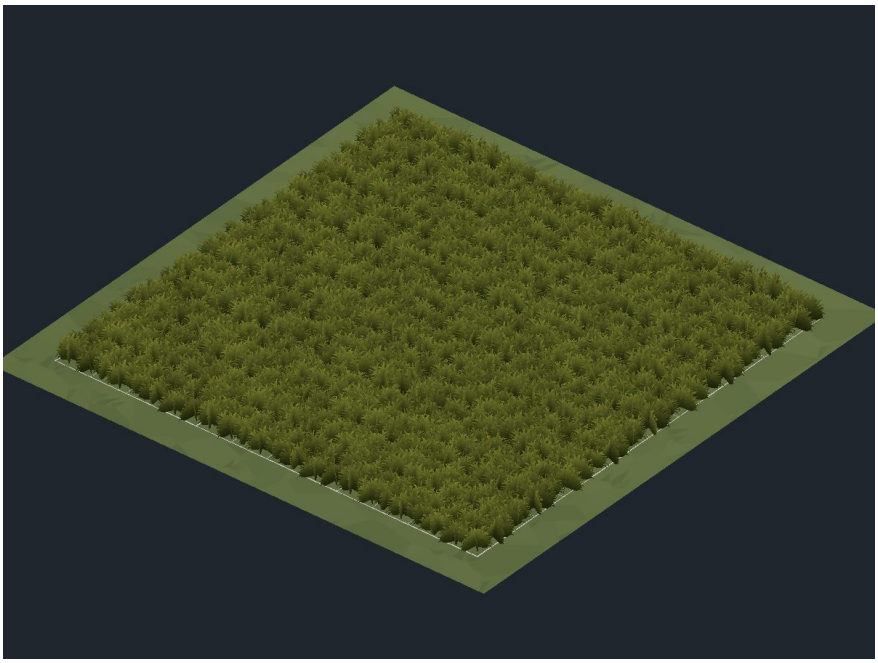
Swamp patches



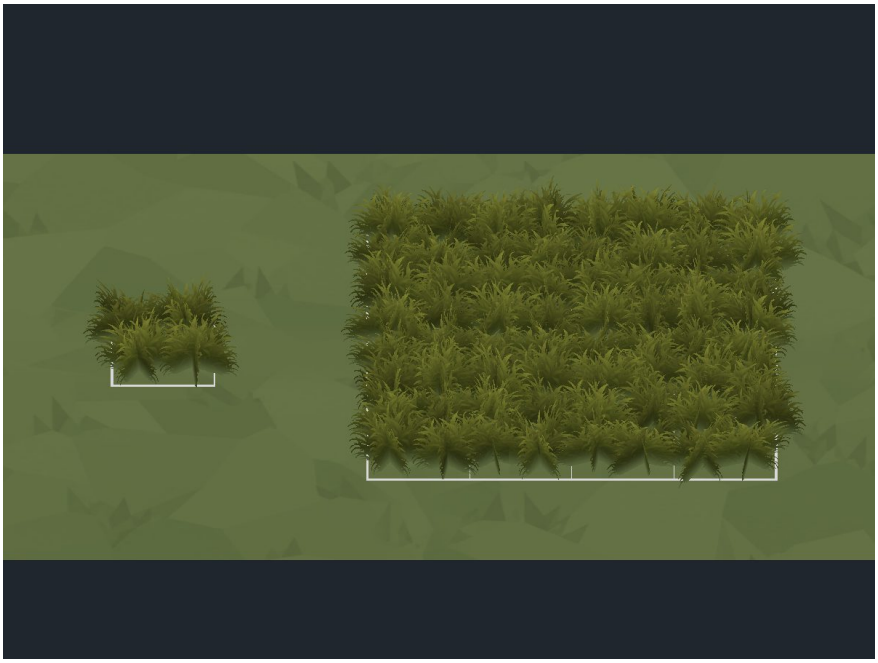
Map with grass prefabs

Joining adjacent tiles

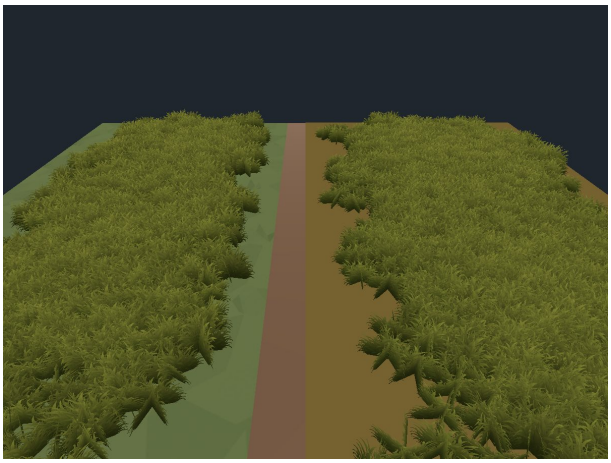
Space supplied tiles **1 unit apart (Medium)** or **4 units apart (Large)** at scale 1. Their fringes overlap to avoid empty strips; small natural gaps between blades remain. Do not use bounds width as placement spacing.



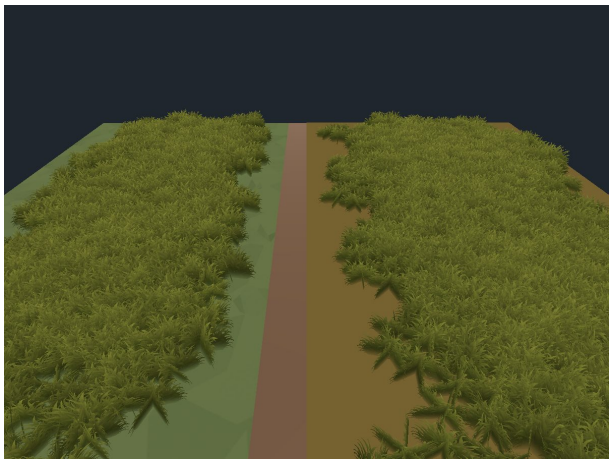
Nine adjoining Large tiles



Large with an overlapping fringe



Before



After

Mountain rendering

English | French version

At the edge of a massif, seabed corners use neighboring water cells' depth without averaging mountain heights. Rock faces close the join down to the bed, including when shared heights are smoothed. Image-mode coastal extensions use the same corners. Regenerate existing maps to rebuild these surfaces and their colliders.

Mountains use **Mountain Material**, supplied by the biome or a map override. Rock extends to the foot of slopes instead of stopping at logical cell boundaries midway up a still-green face. Rock and ground triangles share a continuous boundary without overlapping surfaces. This material distribution leaves the gameplay grid unchanged.

Rocky foothills keep their original surface's collision layer, including custom image-palette layers. They may appear as **MountainFoot_…** objects, separate from the main mountain mesh to retain that layer. Materials remain shared. Paths, ramps, plateaus and seabeds keep their own handling; regenerating without mountains removes old foothills.

To round contours, enable **Enable Procedural Contour Smoothing** in procedural mode or **Enable Image Contour Smoothing** for image imports. The faceted rock style is intentional. **Mountain Max Height** controls procedural massifs; **Imported Mountain Height** controls imported mountains.

Mountains forming chokepoint barriers use the same contour smoothing as ordinary massifs, both inland and along water. Previously, the logical barrier marker pinned every vertex, so parts of the same map could remain stair-stepped. Smoothing preserves blocked cells and creates no additional passage. Actual chokepoint passages, ramps, bridge approaches, resource anchors and other obstacles retain their protections. Outer map cuts remain straight to close the side faces correctly.

Steep slopes now have their own facet density, independent of the base grid. Large faces are subdivided and slightly varied to preserve the rocky style without stretching triangles over the full height. Before the heightmap is applied, a facet spans no more than 0.8 × Cell Size vertically, with a minimum of 0.35 units. Shallow slopes retain their original density.

New join vertices are shared by neighboring materials and colliders. Ground, path and water joins stay on their original boundary. Map-edge cuts remain on the clipping plane and close down to the bottom of the map. Mountain slopes along water use rock material down to the seabed instead of stretching underwater material toward the summit.

This finish adds triangles only to steep slopes and their immediate joins. Logical cells, configured heights, passages and masks remain in place. The Aerial mesh retains its simplification and still intersects massifs; rock colliders above it remain obstacles.

After updating the module, click **Generate Map** to rebuild existing surfaces.

Separate wall and mountain prefabs

BiomeSetting.CliffPrefabs supplies wall, plateau and ramp-side details. **BiomeSetting.MountainCliffPrefabs** supplies the crests embedded in mountain ridges. The map exposes independent overrides with the same names. An empty override follows the biome; an older biome without mountain prefabs retains its previous cliff-list fallback. Disable **Enable Mountain Cliff Details** to omit crests entirely.

The supplied presets use **Biome_Cliff_Block_LowPoly** for walls and **Biome_Mountain_Crest_LowPoly** for mountain outcrops, under **Art/Prefabs/Cliffs**. Both prefabs retain their nested FBX source and a MeshCollider referring to that same source mesh. No prepared mesh copies are created. Their shared neutral rock material is **Art/Materials/Cliffs/Cliff_Rock**; mountain instances follow **Mountain Cliff Material**, then **Mountain Material**. Newly created Starter settings reference the shared Core/Art prefabs. Create a project-owned prefab variant before editing its hierarchy. Existing Starter settings and prefab lists are preserved when creating missing settings.

Chokepoint mountain contours



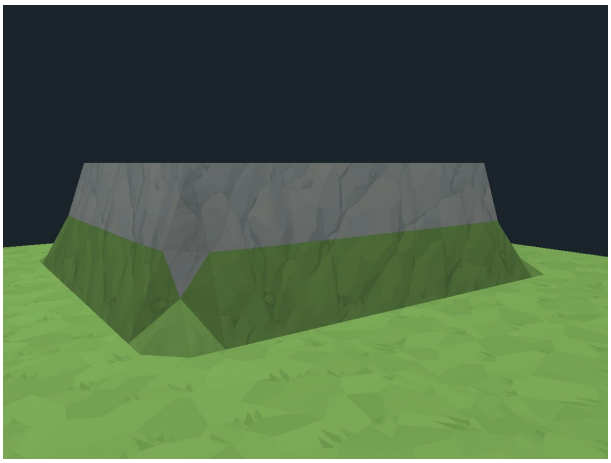
Before



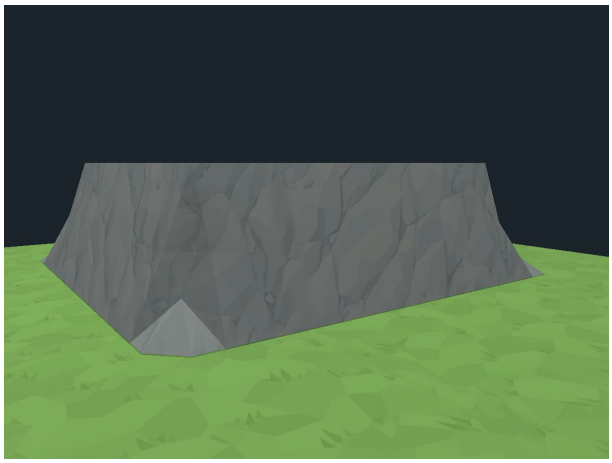
After

Native Unity captures of the demo scene, seed 1346378419, with identical settings, framing and frozen wind. Only the smoothing exclusion for mountains marked as barriers changes. Smoothed corners increase from 21 to 190 out of 194, while 174 protected anchor vertices remain unchanged. The scene retains its pre-existing connectivity diagnostics; these images validate its 3D preview, not its suitability for gameplay.

Rock at the mountain base



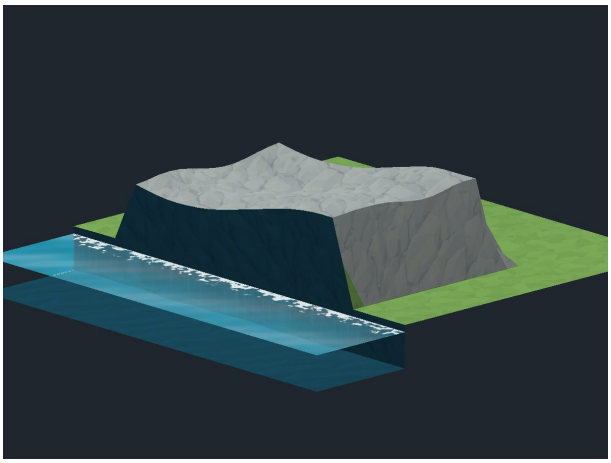
Before



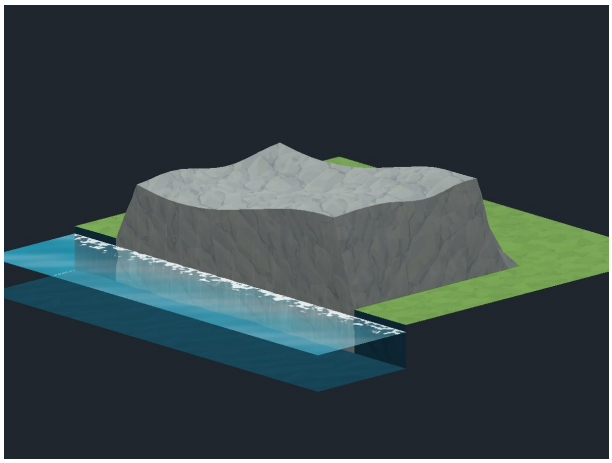
After

Both Unity captures use the same test relief, materials, lighting and camera. Only slope material distribution changes. The correction also applies to maps with altitude enabled.

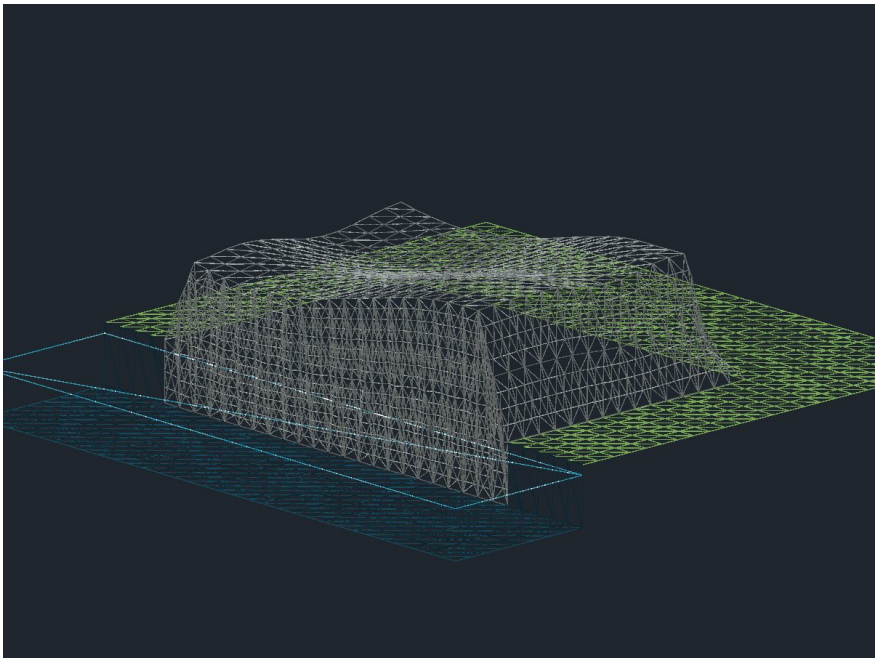
Short facets on slopes



Before



After



Slope mesh

These native views show a test massif and a simple water plane with identical grids and cameras before/after. Wireframe view verifies vertical subdivision independently of the material. Materials and compatible vertices remain shared when **Enable Mesh Optimization** is enabled.

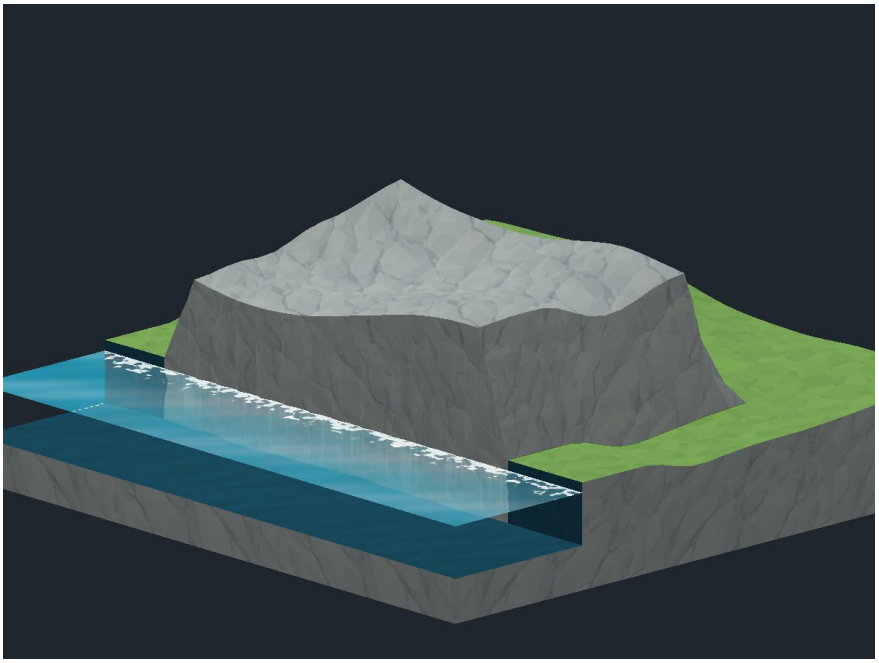


Image import with altitude

Plateau cliff coverage

Local repairs use the actual intersection with a neighbouring ramp or walking volume when trimming a corner. Projecting the entire neighbouring volume onto the wall would remove safe portions beyond the overlap.

Cliff instances and coverage checks use the terrain surface coordinate frame, including the half-cell offset used by the generated scene hierarchy. This keeps the rocks aligned with the actual plateau walls and ramp sides.

Cliff placement preserves the plateau border while leaving paths and ramp entrances clear. When a broad prefab conflicts with a nearby path, the generator first reduces its depth while retaining enough rock around the terrain face and its configured edge offset. If a long chord across a curve still cannot fit, shorter pieces follow the original contour. Remaining unsupported joins receive a local prefab placement through the same clearance checks. Reserved choke openings remain clear.

Walking areas on the plateau do not require an empty border: the rock's upper overlap is lowered beneath the walking surface when needed. With map altitude enabled, wall coverage is checked again after the relief is applied, and only unsupported joins receive a local fitting. Passage checks follow the slope instead of treating uphill paths as flat volumes.

`Cliff Depth Scale`, `Cliff Edge Offset` and the overlap settings remain the preferred fitting values. Tight areas can require narrower or shorter pieces. Unconstrained contours keep the usual merged placement. Ramp borders retain their own thin fitting and use the existing ramp settings.

Generated objects remain connected to their source prefabs in Edit mode. No modified model assets or prepared mesh copies are created. Regenerate an existing map to apply the updated placement.

Ramp borders

Retained plateau triangles are subdivided along the upper landing outline before altitude displacement. This preserves their original surface and UVs while making the terrain meet the landing plane; no additional mesh asset files are created.

Added plateau ramps use a continuous prefab border on each side of the descent. A separate flat segment covers the upper landing from the actual plateau edge on each side. It does not extend across retained flat walking ground. Both segments keep their source prefab links.

The exposed upper inner edge is fitted to the actual ramp endpoints and width, rather than fitting the complete inclined bounding box to a horizontal interval. The border stays mostly embedded, with a small lip above the surface. Its thickness follows `Ramp Cliff Depth Scale`; `Ramp Cliff Inset` is limited near the walking surface to preserve the usable width.

With altitude enabled, borders are fitted again from the final ramp planes, including the slope across the ramp. Placement uses the terrain coordinate frame, including the generated hierarchy's half-cell offset. Offset prefab pivots do not move the visible edge away from the ramp.

Path masks do not shorten these explicit side borders. The border is attached to the edge of its own ramp, while ordinary plateau cliffs retain their passage checks. Ramp surfaces remain on the configured Walkable layer and border colliders use Environment.

Regenerate an existing map to apply these changes.

Triplanar Terrain Shaders

Sorolif RTS Map Generator includes two world-space terrain shaders. They are designed for generated terrain where conventional UVs stretch across steep plateau walls, mountain slopes and coast drops.

Natural triplanar

Shader name: `Sorolif/RTS Terrain/Triplanar`

- `Top Texture`: texture projected on upward-facing terrain.

- `Cliff / Side Texture`: texture projected on steep faces.

- `Top Tint` and `Side Tint`: independent color grading for horizontal and vertical surfaces.

- `World Texture Scale`: world-space texture frequency. The texture remains consistent across separately generated meshes.

- `Projection Blend`: sharpness of the transition between the three projections.

- `Texture Detail Strength`: blends the texture pattern with the biome tint; zero gives an untextured flat tint. The default 0.65 keeps ground readable at an RTS camera distance.

- `Texture Brightness`: balances pale source textures against the material tint without changing the biome palette.

- `Repeat Seam Blend`: blends in a half-period sample near repeat boundaries. Zero retains the original authored repeat; 0.05 is the default. It adds one texture sample per projection when enabled.

- `Broad Color Variation`: subtle world-space grading shared across generated chunks; zero disables it.

- `Slope Darkening`: adds depth to cliffs without requiring a second material.

- `High Ground Tint`, `High Ground Start` and `High Ground Blend`: optional altitude-based variation for plateaus and mountains.

- `Light Direction`, `Ambient Light` and `Lighting Strength`: portable lighting controls that do not require a render-pipeline-specific material upgrade.

Stylized triplanar

Shader name: `Sorolif/RTS Terrain/Triplanar Stylized`

The stylized shader uses the same world-space top/side projection and adds:

- `Slope Accent`: a controlled darker or colored cliff treatment.

- `Light Bands`: quantized lighting for illustrated and low-poly art directions.

- `Light Band Softness`: softens band boundaries while retaining broad lighting planes. Both shaders use wrapped diffuse to keep side faces readable.

- `Rim Color` and `Rim Strength`: silhouette separation at an RTS camera angle.

`Tools > Sorolif > Map Generation > Create Starter Biome Assets` creates `Triplanar_Stylized_Preset.mat` so the stylized controls can be evaluated without modifying the active Forest biome.

Instanced decoration

Both shaders include an instancing variant and initialize the vertex instance ID before reading the world transform. This keeps each repeated prop's texture anchored to its own world position. Newly created starter materials using these shaders enable GPU Instancing; refresh preserves an existing material's choice. Enable the option on your own materials when using an instanced renderer. Ordinary terrain MeshRenderers do not require it. The Built-in ordinary variant retains target 3.0, while `INSTANCING_ON` requests target 3.5. The URP passes use target 3.5. Unity documents

conditional targets in Shader model requirements. Real-device Editor tests compare two instanced objects against individual draws; they are skipped, not passed, when the device cannot support instancing.

Texture recommendations

- Use seamless textures with matching visual scale.

- Start with `World Texture Scale` between `0.08` and `0.25` for maps using the default one-unit cell size.

- Keep side textures slightly darker and less saturated than top textures for readable height changes.

- Increase `Projection Blend` for harder low-poly transitions; lower it for soil, snow and eroded rock.

- Built-in and URP rendering retain authored directional lighting and realtime main-light shadows. Native URP Editor checks cover Unity 6000.2 / URP 17.2 and Unity 6000.4 / URP 17.4 in Forward and Deferred; see URP setup and verification. HDRP and a full PBR lighting stack require suitable project shaders.

Face normals and cast shadows

With Enable Mesh Optimization, coincident vertices and triangle edges are shared within each material. Surfaces using the same material and Unity layer can be combined; surfaces with custom transforms remain separate. The generated object keeps the name of the first surface in its group. Do not use that name to infer all terrain categories in an optimized mesh; use the logical grid instead.

Both terrain shaders reconstruct the face normal from world-position derivatives on optimized generated meshes, retaining the faceted style without duplicating each triangle's vertices. UV channel 3 (Unity's uv4) carries the generated-mesh marker and is preserved by EmbeddedMeshFilter. Existing unmarked meshes continue to use their authored normals. Conventional customer shaders retain UV and hard normal seams. The owned water shader uses continuous UVs and shared cell corners; custom water materials retain their repeated cell UVs. Meshes with at most 65535 vertices use 16-bit indices; larger meshes use 32-bit indices.

Disable Enable Mesh Optimization to retain separate, flat-shaded surface meshes.

Both terrain shaders include their own ShadowCaster pass and receive the main directional light's realtime shadows. Use Cast Shadow Strength and Cast Shadow Tint for readable, soft-colored contact shadows. The material's Light Direction still controls its authored terrain lighting. Shadow rendering also follows your directional light and the project's quality and distance settings.

New starter materials use Texture Detail Strength 0.48, World Texture Scale 0.085 and Broad Color Variation 0.055 to keep large surfaces quieter. Refresh preserves existing custom materials and colors.

Stylized water

`Sorolif/RTS Terrain/Stylized Water` provides animated world-space color waves, a restrained glint, deep/shallow colors and a narrow shore highlight. Rendered wave displacement does not change CPU mesh or collision geometry. Adjust Deep Water, Shallow Water, Shallow Depth Range, Shore Highlight Depth/Strength, Wave Color Strength, Wave Speed and Light Glint Strength on the material.

The shader includes Built-in and URP passes. Add the included `StylizedWaterDepthCamera` component to a custom camera; in Built-in, Depth in `Camera.depthTextureMode` also enables depth-only effects. Starter and included demo cameras request depth automatically. Perspective, orthographic and PNG capture cameras are covered. Without a usable depth texture, the deep palette and wave shading remain available. For URP, configure Copy Depth Mode before transparent rendering as described in URP setup.

URP shaders

The announced release is Unity 6000.2 or later with URP, targeting Windows, Android and iOS. Use the URP version supplied for the Editor (17.2.0 on Unity 6000.2). macOS Editor operation is publisher-reported; mobile-device validation is not recorded. See compatibility.

The eight included shaders have native Editor rendering checks on Windows Direct3D 12 with Unity **6000.2.13f1 / URP 17.2.0** (Gamma) and **6000.4.1f1 / URP 17.4.0** (Linear). Both the Universal Renderer's Forward and Deferred modes are covered. The stylized opaque materials use `UniversalForwardOnly` in either mode, preserving their authored lighting. This verification covers shaders in the Editor; no Player build or package publication is included in this shader validation.

Project setup

1. Install the URP package matching your Unity version and create a URP asset with a **Universal Renderer**.

2. Assign the pipeline in Graphics and in any Quality level that overrides it.

3. For water, set the Renderer asset's **Copy Depth Mode** to **After Opaques** or **Force Prepass**. The default After Transparents copies depth too late.

4. Add **StylizedWaterDepthCamera** to your gameplay camera. It requests depth and opaque color through URP camera overrides. The included demo/capture cameras already request these inputs.

5. Enable main directional-light shadows on the pipeline and light, with a shadow distance suitable for the map and camera.

The module does not assign a project pipeline or change renderer settings. Package requirements keep its Built-in fallback available when URP is absent. HDRP, the URP 2D Renderer and other graphics APIs require separate validation.

Included passes

Triplanar and Triplanar Stylized

URP passes: Forward, DepthOnly, DepthNormalsOnly, ShadowCaster

Props / Atlas Lit

URP passes: Forward, alpha-tested depth and normals, wind-aware ShadowCaster

Stylized Grass

URP passes: Forward, alpha-tested depth and normals, wind-aware ShadowCaster

Stylized Water, Snow Water, Autumn Water, Forest Water

URP passes: Transparent Forward, camera depth and opaque-color sampling

The terrain retains its world-space projections, optimized-mesh face normals, top/side tints, texture grading and material-controlled **Light Direction**. URP supplies realtime main-light shadows and fog. This is authored stylized lighting, rather than URP Lit PBR or additional-light shading.

Vegetation keeps its alpha cutout, wind and per-renderer sampled soil tint in Forward and Deferred. Material constants use `UnityPerMaterial`; per-object property blocks can opt a renderer out of SRP batching, as usual in Unity. This shader layout does not enable GPU instancing in Edit Mode. Runtime instancing remains controlled by the existing

map renderer.

Existing shader names, GUIDs, material properties and biome palettes are retained. The five basic QuickStart resource/tree/rock materials formerly using Standard now reference the included Atlas Lit shader; their serialized colors and other settings are retained. No source mesh or prefab is duplicated.

Verification

Native GPU regression tests cover terrain tint, required passes, shadow casting and reception, vegetation cutout and per-renderer soil tint. The tests explicitly skip when URP or a real graphics device is unavailable. Separate map captures use the actual Forest, Tropical, Snow and Autumn presets, including terrain, grass, cliffs, props and animated water, in both renderer modes.

See Triplanar terrain, water setup and compatibility scope for the individual controls and remaining release validation.

Stylized water materials

The owned `Sorolif/RTS Terrain/Stylized Water` shader provides the general biome styles. Snow uses the separate `Sorolif/RTS Terrain/Snow Water` shader; see Glacier water. Autumn uses `Sorolif/RTS Terrain/Autumn Water`, with clear woodland water and floating bank leaves; see Autumn water. The release supplies four biome water styles, referenced by the corresponding biome presets. Forest uses Sorolif/RTS Terrain/Forest Water; Tropical uses the general Stylized Water shader; Snow and Autumn have their dedicated shaders. Their materials and optional light/foam masks live under Art and remain editable.

Forest

Included water appearance: Waves, crest foam, moving seabed light and fish silhouettes

Tropical

Included water appearance: Turquoise shallows, caustics and broken shoreline foam

Snow

Included water appearance: Cobalt/cyan depth, cold currents and breaking coastal ice

Autumn

Included water appearance: Woodland water with softly appearing floating bank leaves

The shaders combine procedural patterns with optional supplied foam/light masks. Ripples affect shading and the presets add subtle rendered vertex displacement, up to 0.035 world unit in the tropical style. CPU mesh vertices, water levels, colliders and bridge fitting retain their generated positions.

The dedicated Snow preset uses cobalt depths and transparent cyan shallows. Its currents, ice and shore breakup are independently adjustable. `Snow_Water` and `WaterOnSnow` share the style; the Tropical shader and materials remain independent. Snow's depth transition spans **2.2 units**. Its own seabed light uses soft noise patches, rather than the general shader's Voronoi controls.

Assign a style

Choose a matching **Biome preset**, or assign an included material to `MapDefinition.WaterMaterial`. A directly assigned material takes priority over the biome. Duplicate a material into your own folder to customize it.

The starter wizard creates the new styles for new materials. Refresh preserves existing user shaders, colors and settings. An existing project can assign one of the included materials or explicitly reset a material using the preset API:

```
var water = new Material(Shader.Find(StylizedWaterPresets.ShaderName));
StylizedWaterPresets.Apply(water, BiomeType.Tropical);
definition.WaterMaterial = water;
```

Manage the lifetime of a runtime-created material in your project.

Camera and pipeline

Add **StylizedWaterDepthCamera** to your gameplay camera. It requests depth and opaque scene color in Built-in and configures the public camera data of an active URP installation. Demo and screenshot cameras already request depth.

Built-in makes one opaque-color copy before transparent rendering, shared by all water draws on that camera, and releases its temporary target after rendering. Disabling the helper removes only its own command buffers. URP uses

its opaque texture through camera overrides; the helper does not edit a pipeline asset. Scene view cameras receive a transient helper when a map generator is present; this does not save a component in the user's scene.

In the URP Renderer asset, set **Copy Depth Mode** to **After Opaques**, or use **Force Prepass**. Depth must be available before transparent water is drawn. After Transparents copies it too late for these effects. The camera helper preserves your shared renderer asset and does not change this project setting.

The shader contains Built-in and optional URP passes. Package requirements skip the URP code when URP is absent; the module does not require that package. Native Editor rendering is checked in isolated projects on Windows Direct3D 12, Unity 6000.2.13f1 / URP 17.2.0 and Unity 6000.4.1f1 / URP 17.4.0, in Forward and Deferred. Terrain, vegetation and the four water shaders are covered together; see URP shader setup and verification scope. Other pipelines require separate validation.

Opaque seabed and banks provide the depth for shore effects. Without depth, water retains its deep palette and ripples. `_UseDepth` disables depth effects explicitly. Orthographic and perspective views use physical water depth along the surface normal, keeping the palette consistent when the camera tilts.

Controls

For **Forest Water**, the **Open Sea Waves** section controls the larger crest waves and their foam. **Open Sea Start Depth** is the resting distance from the water surface to the seabed, in world units. Waves start beyond that depth and reach full strength over **Open Sea Depth Transition**. For example, start 2 and transition 1 fades them in between depths 2 and 3. Start 0 preserves the previous appearance at every depth. The same mask affects wave displacement, normals, refraction and crest foam; the shallow/bank effects retain their controls.

Open Sea Distance From Shore is a separate horizontal clearance from land. Both conditions must be satisfied. **Crest Foam Height Threshold** controls which elevated parts of a wave produce foam; it is not a seabed-depth threshold. Depth gating uses the generated water mesh's UV1 depth data, without an extra texture or camera-depth read. Meshes without that metadata retain the previous behavior; generate water through `WaterGenerator` to supply it.

- **Deep/Shallow Water** set colors and opacity. **Shallow Depth Range** controls their transition in world units; **Edge Transparency Depth** fades the contact.

- **Shore Foam Depth/Strength/Breakup Scale** shape an irregular foam rim. **Foam Lace Contrast** opens small water holes between denser white patches; zero retains the softer original rim. **Rolling Shore Foam** adds travelling bands that move from deeper water towards the bank. **Shore Wave Speed** controls their arrival rate; zero freezes this shoreward motion.

- **Flow Direction X Z** and ripple scale/speed/normal strength set motion. This is a shared artistic flow direction for the material.

- **Visual Wave Height** adds small rendered swells without changing water physics or CPU geometry. Zero keeps the previous flat rendered surface.

- **Refracted Bed Blend** composites the visible opaque bed and props through the water. **Bed Distortion** displaces that image with the moving normals; the tropical default is restrained. Distortion fades at contact and rejects samples above the surface to keep banks from bleeding into the water. Refraction needs the camera helper (or URP opaque/depth textures); it falls back to ordinary alpha transparency when these buffers are absent. Objects rendered only in the transparent queue are not part of the opaque-color copy.

- **Sun Glint**, **Fresnel Reflection** and **Sky Reflection Tint** control the stylized light response. The sky reflection is a color-based Fresnel effect.

- **Shallow Caustics** adds light patterns over the visible shallow bed and fades with depth. **Caustics Visible Depth** extends their reach into clearer water; **Caustics Flow Distortion** bends the light network into animated filaments. **Caustics Diffusion** softens the highlights. With refraction, caustics illuminate the captured bed rather than covering it with a bright surface lattice. Set strength to zero to disable this per-pixel work.

- **Seabed Voronoi Blur**, under **Underwater Light**, blurs only the caustic light network projected on the depth-reconstructed seabed. Its radius is in world units: **0** retains the sharp pattern, **0.10** is the Tropical default, and

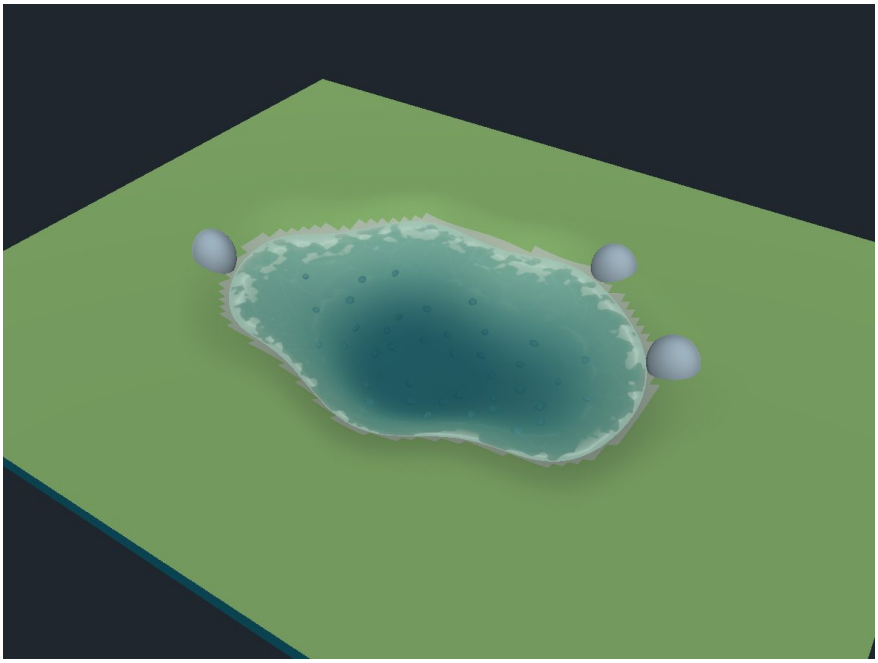
0.20-0.30 gives a more diffuse glow. Bed textures and shore foam stay sharp. This is independent of **Caustics Diffusion**, which also bends the animated pattern. The filter broadens and reduces the light peaks in a single Voronoi evaluation; it does not add a full-screen blur pass.

- **Surface Film** adds drifting colored patches. Custom styles can use olive variation with caustics disabled.

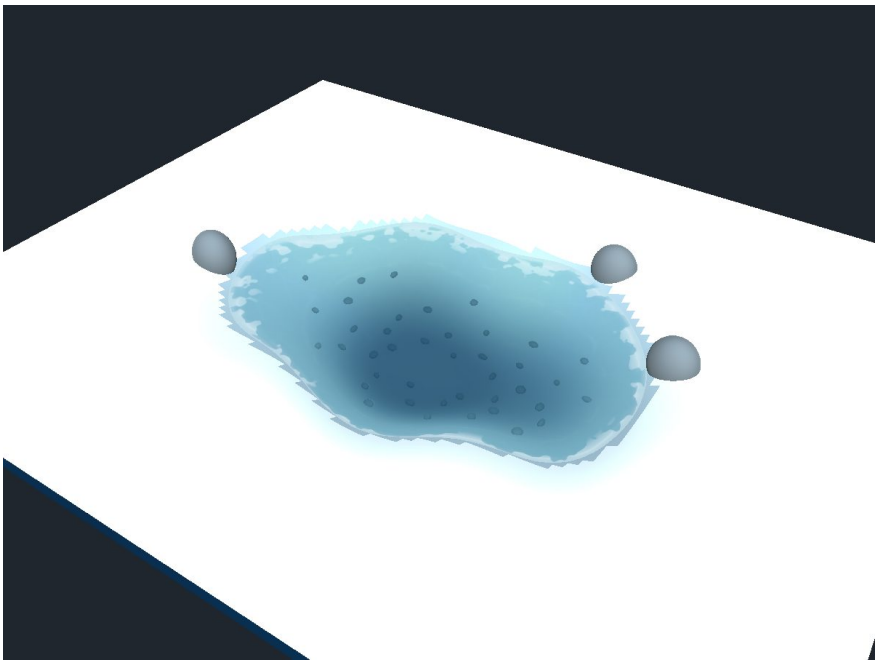
- **Animation Time** uses the running clock at -1. Nonnegative values freeze animation for comparisons and screenshots.

Native previews

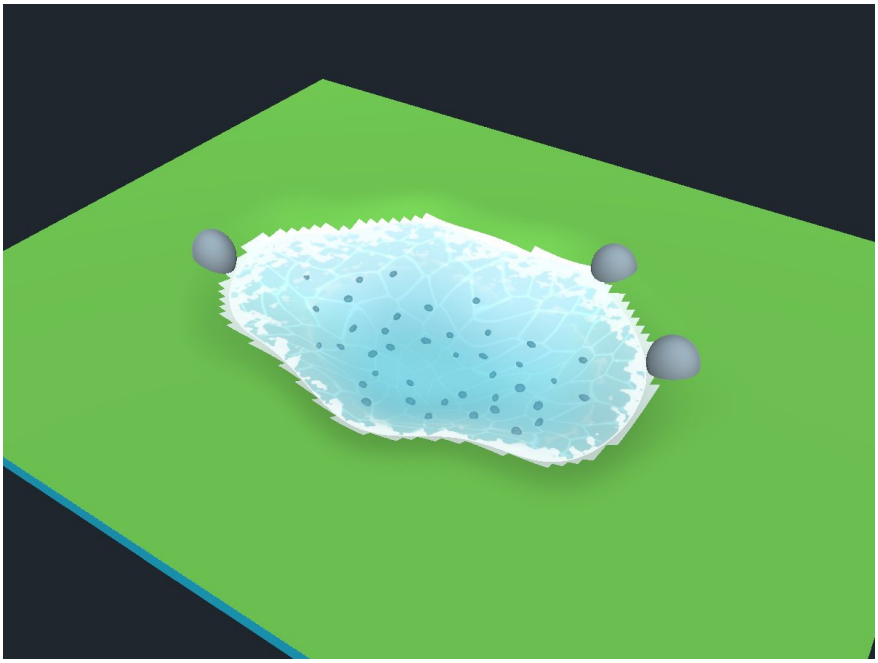
The calibration basin and illustrative rocks show the water independently of the final environment props.



Classic water



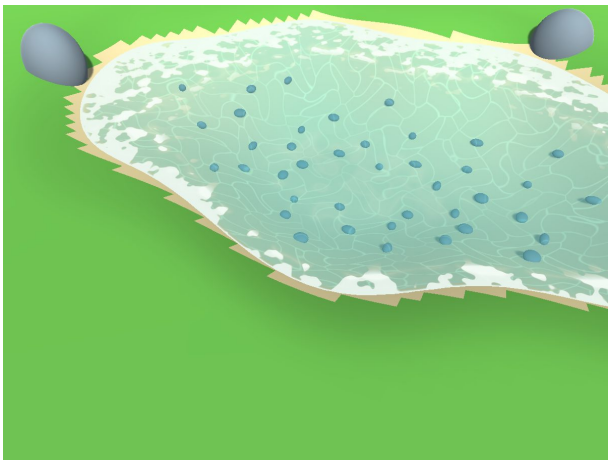
Snow water



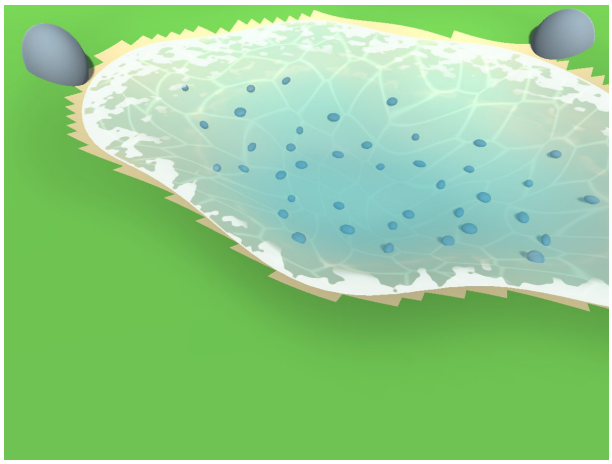
Tropical water

Tropical refinement - 19 October 2026

The tropical preset uses a lighter palette, lower shallow opacity, a longer depth-color transition and more fragmented white foam. The blade/terrain meshes, water height and collision are unaffected. The light network is entirely procedural, with no copied reference textures or external shader dependency.



Before



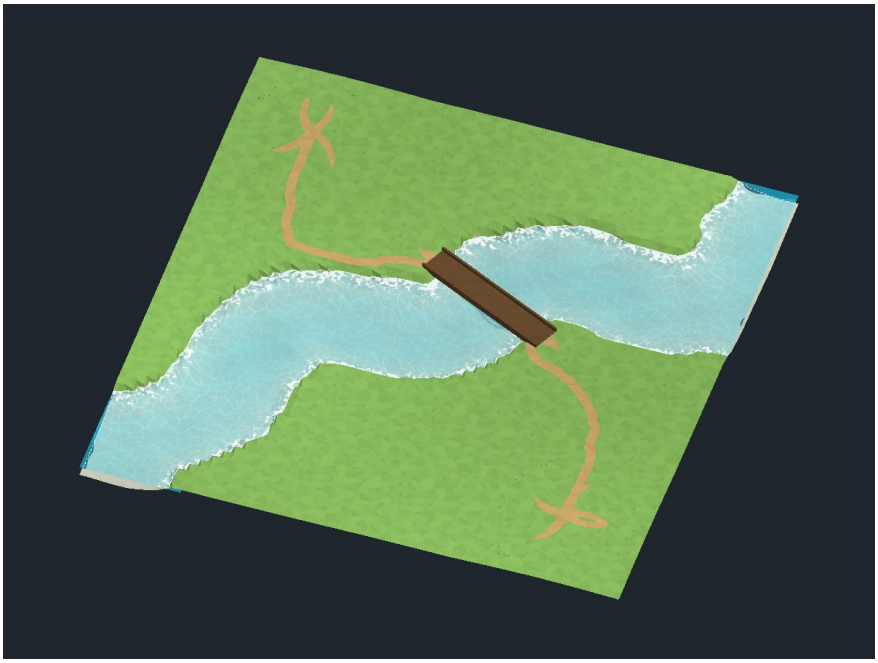
After

These native Unity views use the same calibration basin, camera and lighting. The terrain and spheres are illustrative test geometry. Generated-map views and Built-in/URP validation are included in the publisher report. Existing custom materials keep their values: assign `Art/Materials/Water/WaterTropical` to **Water Material**, or explicitly apply the tropical preset to a duplicate. The tropical Starter material in this working project has also been updated. The camera still needs depth before transparent water, as described above. The old Starter demo scene in this workspace now includes the existing StylizedWaterDepthCamera component on Main Camera; reopen that scene if it was already open during the update. Other custom scenes should add the component to their gameplay camera explicitly.

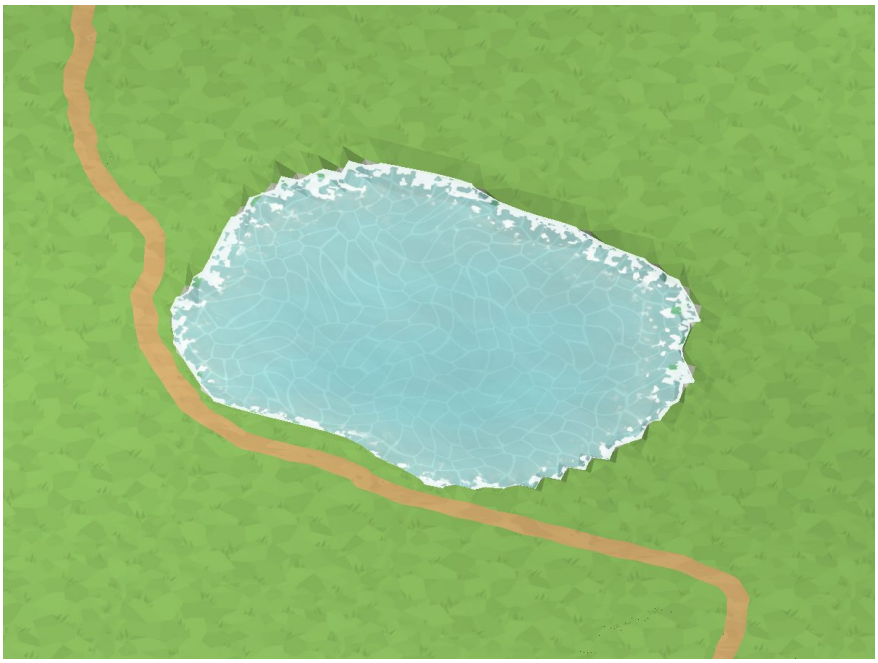
Tropical optics: scene color and animated bed

The tropical preset now refracts the actual opaque scene, including submerged props. Bed Distortion is 0.014, Refracted Bed Blend is 1 and Visual Wave Height is 0.035 world unit. Caustics Diffusion softens the light network.

Distortion fades near the shore and rejects above-water depth samples to reduce bleeding. The neutral tropical seabed keeps the water blue instead of olive.

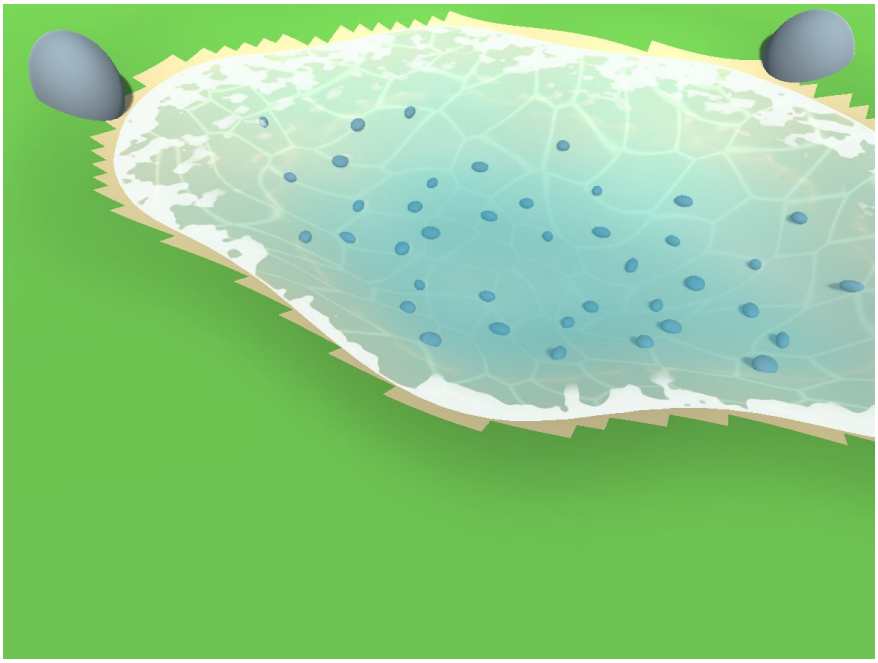


Generated tropical river



Generated tropical lake

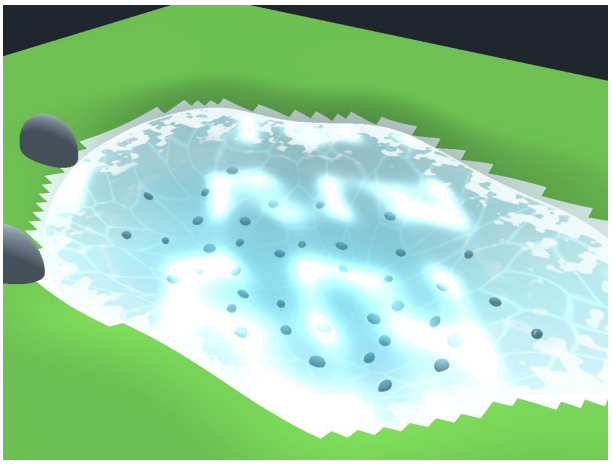
The native views use the real map generator. The shore comparison above uses the same basin and lighting before and after. The second animation frame below shows the moving foam, ripples and bed highlights. CPU water geometry and colliders retain their physical heights.



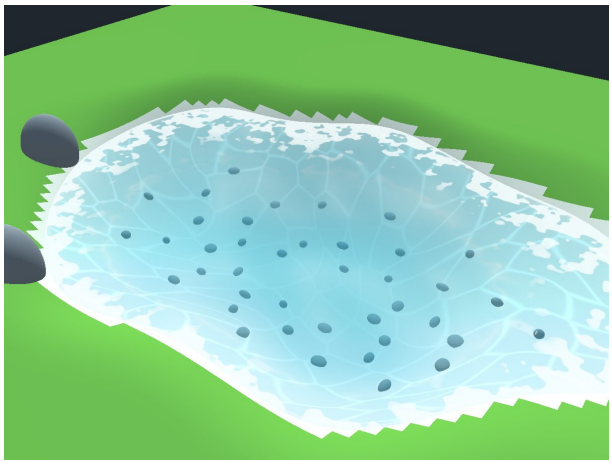
Second animation frame

Reflection polish and biome palettes

The sky tint and sunlight are composed as a separate surface coating after transmission. They are not amplified by dividing by shallow-water opacity. Sun glints are filtered at a distance, and the sky response uses gentler ripple normals to avoid bright bands. Built-in and URP both reject refraction samples above the water surface, including the colour filter footprint around them. This prevents displaced outlines when the opaque colour is downsampled. This is a stylized sky tint and direct sun highlight; it does not trace mirrored scenery or add a reflection camera.



Before



After

Forest

Character: Blue-green water, neutral muddy bed, broken cream foam

Snow

Character: Cold blue depths, pale clear shallows, slow ripples

Tropical

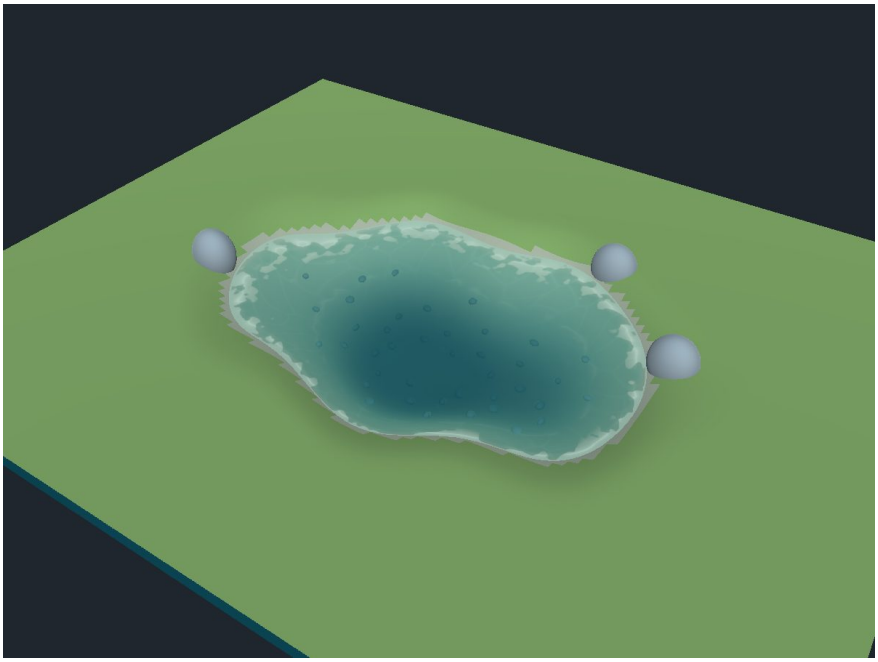
Character: Existing turquoise palette, gentler reflections, animated bed light

Autumn

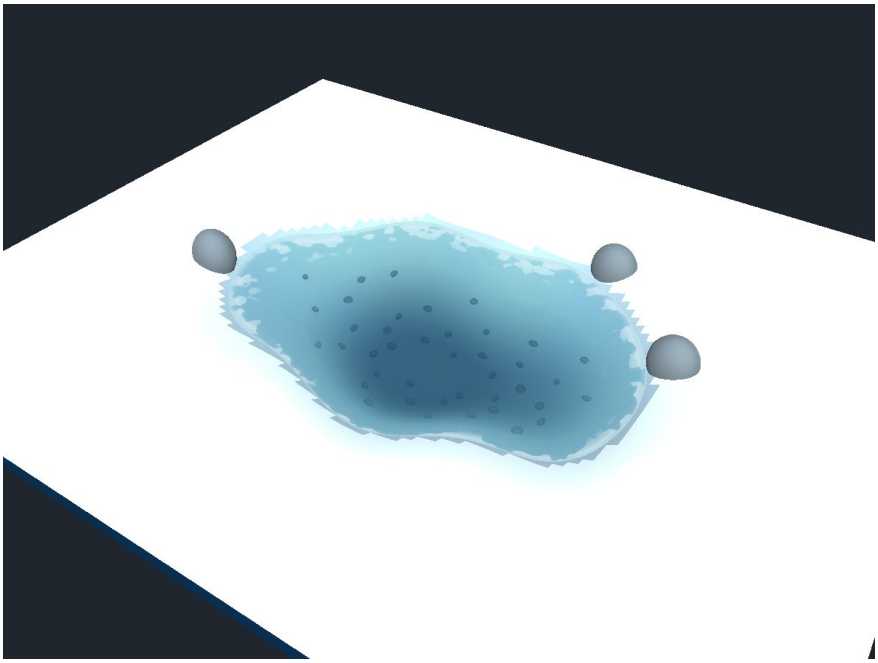
Character: Muted teal, warm shore foam and earthy bed

Fluid Emission defaults to zero. A custom warm-fluid style can use it for a visible fluid with no blue foam or caustics. It does not illuminate nearby geometry and does not require bloom or a postprocessing package. Surface film masks part of this emission to keep the cooling patches dark.

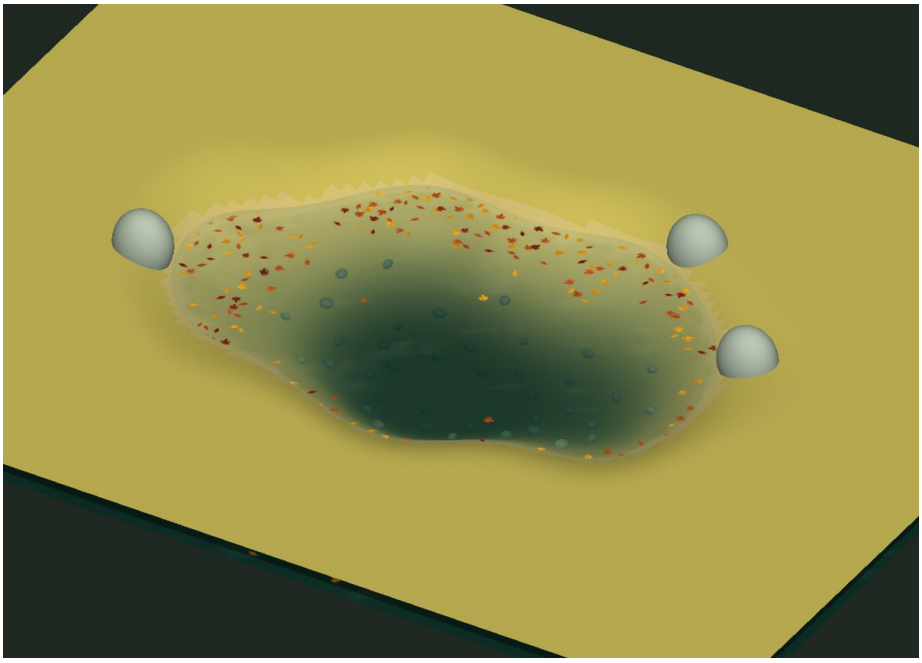
Only Forest, Tropical, Snow and Autumn presets are included in the release. Other enum values remain available for custom settings. Existing owned water/seabed material GUIDs are retained. Future starter materials use the same palettes. Refresh still preserves existing custom materials. Assign a matching included material when a direct override currently points to a custom older one. Regenerate existing maps to refresh any generated material copies.



Forest



Snow



Autumn

Forest caustics and lake shores

Forest water uses the included `ForestCaustics` light mask for moving seabed filaments. **Seabed Caustics Mask** accepts a replacement texture; **Seabed Light Scale**, **Soft Seabed Light** and **Seabed Light Visible Depth** control its world size, intensity and depth fade. The included mask is imported at 512 × 512 with mipmaps. Warped, overlapping patches avoid visible tiling; independent wave directions and wavelengths break up the surface pattern. Existing custom masks survive preset application.

In procedural Lake mode, contested islands use their own organic coastlines. The Lake group exposes **Contested Island Radius Min/Max**, **Island Contour Irregularity** and **Island Minimum Water Gap**. Placement reserves space for nearby player ramp landings before carving the lake. Beach shelves preserve the water between islands, and resource working pads remain on connected land. Other island layouts retain their existing generation rules.

Submerged Forest shoreline triangles use the seabed material. Grass placement also rejects roots below the local water level, including dry cells covered by the visual water border. Regenerate a saved map to update its terrain, island

positions and grass placements.

The views use an owned calibration basin, simple stones and the same lighting. They demonstrate the materials independently of the final biome props.

Glacier water

English | French version

The Snow biome uses **Sorolif/RTS Terrain/Snow Water**. Its shader is separate from Tropical water. The included **Snow_Water** and **WaterOnSnow** materials provide cyan shallows, cobalt depths, cold underwater currents, drifting ice and breaking coastal ice. Select the Snow biome preset or assign either material as the map's Water Material. An explicit custom material remains authoritative.

Cold currents

Current Volume Strength controls soft cyan billows below the surface. Four samples through the depth-reconstructed water column give the effect depth and parallax. It uses a moving three-dimensional noise field, without repeated sinusoidal bands. It is a view from the water surface, rather than an underwater camera fog system.

- **Current World Scale** sets the size of the billows; smaller values make them larger.

- **Current Travel Speed** and **Current Direction X Z** control their movement.

- **Current Visible Depth** limits sampling to a shallow column in world units.

- Set strength to **0** to disable the volume work.

Depth affects the palette and visibility of these effects. A very shallow pond stays cyan; deeper sea areas reveal cobalt water and more visible currents. **Shallow Depth Range** defaults to **2.2 world units**. The seabed's independent soft light uses broken noise patches, without a sinusoidal light network.

Ice floes

The shader intersects the viewing ray with small irregular ice slabs. Their raised caps, blue side faces and fine fractures remain readable at low camera angles. Floes drift and rotate slowly, independently of the underwater currents.

- **Ice Floe Density** controls how many cells contain open-water ice (default **0.06**).

- **Ice Cell World Scale** controls cell spacing; lower values make cells larger.

- **Ice Floe Size** controls the footprint inside a cell.

- **Ice Height Above Water** sets the apparent relief in world units (default **0.16**).

- **Ice Drift World Units per Second** and **Ice Rotation Speed** control motion.

- **Ice Color and Opacity** controls tint and transparency.

These are visual shader features: they create no GameObjects, meshes, colliders or navigable surfaces. They do not change the map's physical water levels. Disable open-water ice with **Ice Floe Density = 0**.

Breaking shore ice

Shore Ice Coverage creates denser, overlapping ice plates on the shallow, sloping banks. Over each cycle, cracks open between plates and individual pieces move away from the coast before fading. Different plates use different phases.

- **Shore Ice Depth Band** sets the shallow-water depth covered by the bank ice (default **0.75 world unit**). This is a depth band, not a fixed horizontal width.

- **Shore Ice Break Cycle Speed** controls the breaking cycle. Zero freezes it.

- **Breakaway Travel in Ice Cells** sets outward travel as a fraction of the ice cell size (default **0.35**, about **0.78 world unit** at the default scale).

- Set **Shore Ice Coverage = 0** to disable the coastal effect independently.

Coastal ice uses the reconstructed water depth and its local slope to find the bank and outward direction. It fades at contact and does not cover flat shallow areas indiscriminately. Without camera depth, coastal ice and underwater currents are disabled; open-water floes and the deep palette remain available.

Camera, animation and presets

Add **StylizedWaterDepthCamera** to gameplay cameras. Under URP, depth and opaque color must be available before transparent rendering. See Water rendering. The shader includes Built-in and optional URP passes.

Animation Time = -1 uses the clock; any nonnegative value freezes all effects for comparisons. Updating or duplicating an existing custom material does not reset its settings automatically. To explicitly reset an owned water material:

```
StylizedWaterPresets.Apply(material, BiomeType.Snow);
```

This selects the dedicated Snow shader on an owned water material. Applying the Tropical preset explicitly selects the shared water shader again. An unrelated custom shader is never replaced by this API.

Autumn water and floating leaves

English | French version

Sorolif/RTS Terrain/Autumn Water is a dedicated shader for the Autumn biome. The supplied **Autumn_Water** material is referenced by the model and Starter Autumn presets. It combines deep forest-green water, clear amber shallows, restrained reflections, gentle ripples and small patches of cream-colored foam. Tropical and Snow keep their own shader implementations.

Floating Autumn Leaves

Pointed and lobed leaves have small stems, veins and color variation. They use gold, copper and russet tones. Leaves cluster near shallow banks; a few can drift over deeper water. The shapes and motion are procedural shader effects, with no texture dependency, additional GameObjects, meshes or colliders.

Golden / Copper / Russet Leaves

Effect: Three editable leaf colors

Shore Leaf Density

Effect: Spawn probability near the bank; default 0.78

Open Water Leaf Density

Effect: Much sparser leaves away from the bank; default 0.015

Leaf Shore Depth Band

Effect: Depth range for bank accumulation; default 1.2 world units

Leaf Pattern World Scale

Effect: Higher values make cells and leaves smaller and closer together

Leaf Size in Cell

Effect: Leaf size relative to its cell, independent of spawn probability

Leaf Drift World Units per Second

Effect: Translation along Flow Direction X Z; default 0.025

Leaf Turn Speed

Effect: Slow individual rotation and tiny eddies

Leaf Cluster Strength

Effect: Irregular groups with open water between them

Leaf Color Variation

Effect: Small differences between individual leaves

Leaf Opacity Fade

Effect: Soft appearance/disappearance as leaves drift through the bank density band; default 0.2. Higher values widen the transition

Density is a per-cell probability, adjusted by clustering and water depth; it does not describe the percentage of the surface covered by leaves. At the default scale and size, leaves are approximately 0.25-0.35 world unit long. Putting both leaf densities at zero disables the effect. Setting drift and turn speed to zero freezes leaves while the water can continue moving. The opacity fade smooths the density threshold for each complete leaf, rather than dissolving its silhouette. Its width is relative to density, not seconds; the transition duration also depends on drift speed and the submerged slope.

The depth band follows the submerged bank, rather than a fixed horizontal distance. Local depth gradients keep leaves attached to their sampled surface as the camera moves. They remain a surface coating instead of being tinted or refracted with the seabed. Opaque geometry still occludes them normally.

Water appearance

Deep Forest Water and **Clear Amber Shallows** set both color and absorption opacity. **Shallow Depth Range** defaults to 2.6 world units. The visible bed contributes much more in shallow water. **Bed Distortion** stays subtle by default; **Soft Underwater Light** adds restrained moving light on the bottom.

Broken Current Foam adds sparse elongated flecks along the artistic flow direction. Shore foam and small rolling bands remain independently adjustable. These are surface effects: they do not generate waterfalls, alter water levels or simulate hydraulic flow. A material has one shared artistic flow direction.

Camera and integration

Add **StylizedWaterDepthCamera** to gameplay cameras. In URP, camera depth and opaque color must be available before transparent rendering. See Water rendering. Without depth, the water retains its deep palette and ripples; bank leaves are disabled instead of being guessed.

Select the Autumn biome preset, or assign **Autumn_Water** as the map's Water Material. An explicitly assigned custom material retains priority. Existing custom materials are not reset automatically. To reset an owned water material explicitly:

```
StylizedWaterPresets.Apply(material, BiomeType.Autumn);
```

This selects the dedicated Autumn shader and its defaults. An unrelated custom shader is preserved. **Animation Time = -1** follows the clock; a nonnegative value freezes all animation for screenshots and comparisons.

Troubleshooting

The inspector reports missing materials or prefabs

Run `Tools > Sorolif > Map Generation > Create Starter Biome Assets` for independent settings initialized from Core/Art, or assign production materials and prefabs directly on `MapDefinition` or through an explicitly assigned `BiomeSetting` anywhere in the project.

Missing visual assets are warnings unless the related feature cannot operate without them. Use `Validate Selected Generator` to see all configuration issues before generation.

Generation returns false but a map remains visible

Inspect `ProceduralMapGenerationResult.ValidationIssues` or the validation section in the inspector. Blocking causes include a null definition, an unsupported map size, an invalid player count, a non-positive cell size, or an unreadable image source.

A failed attempt leaves a 3D preview: terrain and water from the current grid, the previous map if a new preview cannot be built, or a simple terrain with a collider for an invalid initial configuration. The Inspector shows the error and a `generation.preview` warning. Console messages use warnings so Unity's Error Pause does not interrupt recovery. Unexpected generation exceptions are caught and included as `generation.exception` with their stack trace.

The generator does not return an earlier successful grid after a failed attempt. Always check `Succeeded` or the Boolean returned by `TryGenerateRuntimeMap`. Use `PreviewRoot` for the visible output; `MapRoot`, `Grid`, gameplay score and spawn lists are unavailable on failure. Saving a generation recipe remains disabled until a successful regeneration.

River regions cannot be connected with safe bridges

`bridges.connectivity` means at least one land region cannot receive a safe crossing with the configured span and landing constraints. Generation continues through the remaining visual stages: mountains, terrain, water, cliff details, biome props, altitude, packing, cleanup and optional screenshots. Plateau-access and player-land-route audit failures also keep their diagnostics without stopping these stages. The finished map remains an unvalidated preview; independently validated path bridges can be shown, but unsafe or missing bridge decks are skipped. A malformed custom bridge prefab is reported as `bridges.prefab_invalid` and does not stop later visual stages. Increase **River Bridge Max Span**, reduce **River Width**, or keep wider, flatter banks clear of cliffs and resources, then regenerate. An error preview does not guarantee that all land regions are playable or reachable.

A river crosses a path or choke corridor

With **Has Bridges** enabled on a procedural **River** layout (including a Random layout that resolves to River), a protected choke corridor no longer cuts a continuous river into separate pieces. The generator reopens the river and reserves a bridge perpendicular to the local river course, including when the land regions already connect elsewhere or several paths cross the same river. Water, its bed and naval accessibility remain underneath the deck; land traversal uses the deck height. Assign a usable **River Bridge Prefab** and regenerate.

A path crossing requires the same river to flow through both lateral sides and dry path continuations on both banks. Oblique paths reconnect to the perpendicular deck over dry terrain. A river that merely touches or follows the path, ends inside it, or meets a path dead end does not request a path bridge. Bases, resources, ramps and plateau reservations keep their existing protection. Lakes, island naval corridors and disabled bridges retain their existing behavior. If the crossing exists but its bridge span, landings or prefab are invalid, the river remains visible in the error preview instead of being filled with land.

A river approaches a base or expansion

Procedural rivers skirt settlement clearings using **Player Base Forest Clear Radius** and **Expansion Forest Clear Radius**, including with forests disabled. The same values are exposed in **River shape**. Existing base, expansion and plateau footprints remain dry. The full channel routes around the clearing, rather than stopping at its boundary and resuming on the other side.

An occupied central clearing takes priority over the primary river's central anchor. A free central passage is retained, and rivers can pass between settlements when their full width fits. If clearings occupy every possible passage, increase map size, reduce River Width or space the settlements farther apart. A river is not carved through a protected clearing to force a crossing.

If a different land region still blocks the global connectivity audit, the error preview can display independently validated path bridges with a usable deck collider. The whole map remains an error preview until a successful regeneration; unsafe or missing decks are not displayed.

A procedural road ends against a plateau wall

Regenerate the map with the current generator. Choke corridors are reserved before the terrain heights are known; their blocked dry segments are now recalculated over the final terrain through legal ramps. The road curves respect plateau edges and cannot cut across a cliff corner. If no legal access exists, the unreachable link is omitted instead of painting a direct road against the wall. River crossing reservations remain unchanged.

Toggling Bridges changes choke roads or hides gizmos

Disable **Use Random Seed** and keep the same **Seed** when comparing settings. With a fixed seed, the planned choke points and strategic corridor routes are independent of **Has Bridges**. The renderer retains their river crossing reservations even if a later bridge or player-route audit fails. On plateaus, blocked dry portions are recalculated through the final ramps; reachable portions remain visible. True river crossings use the bridge deck, and unsafe crossings keep the water and diagnostic. Other player trails can follow different available land connections, so the entire finished map need not be identical.

Bases, expansions and choke gizmos belong to the visible generated map, including an unvalidated preview. Their cached positions survive scene reload and follow the generated root transform, cell size and altitude. Enable **Show Gizmos** on the root's **Generated Map Info** component and the global **Gizmos** switch in the Scene view to display them. **Select Map Information** on the generator selects that component; see Generated Map Information. An invalid initial fallback has no strategic markers; retaining an earlier map also retains its markers.

Image generation fails

- Enable **Read/Write** in the source texture import settings.

- Assign the source image before selecting `Image` generation mode.

- Verify that color tolerances and region vote thresholds match the source artwork.

- Start from a high-contrast image without compression artifacts, then tune thresholds gradually.

Water, paths or mountains are not visible

Confirm that the corresponding feature is enabled and that its material is assigned. Procedural roads remain curved with `Enable Path Smoothing` disabled; this switch refines contours rather than choosing another network. Essential town and expansion access is generated first. Increase `Path Connection Count` for alternative links, `Path Branch Count` for extra branches, or `Wandering Path Count` for independent trails. Local streets may deliberately end in pointed dead ends around towns. A road cannot cross an unbridged river or climb a cliff outside a safe ramp.

Mountain formations avoid water, bases, expansions, resources, ramps, choke corridors and reserved strategic routes. A densely occupied or very small map can therefore contain fewer massifs than requested. Reduce `Mountain Path Clearance`, `Mountain Max Area` or `Mountain Zone Count` if the map does not have enough legal space.

River maps have no bridges or disconnected players

Enable **Has Bridges** and assign a **River Bridge Prefab** with a non-trigger deck collider. Bridges apply to the resolved **River** water layout. The planner searches the river banks even if an earlier road hint cannot reach a safe deck. Conflicting road crossings yield to a connected bridge network; final player routes use its walkable crossings. Water remains underneath the bridges.

Generated plateau walls close single-cell ground pinholes completely surrounded by cliff edges. These wall artifacts no longer count as isolated river banks. Real islands, open passages and strategic areas still have to remain reachable.

When bases and expansions already share a dry route, a river layout can have no bridge if no safe shortcut is available. The important check is strategic accessibility, rather than the number of visible decks.

An impossible span or banks blocked by resources, mountains or steep cliffs still produces a validation issue. Increase **River Bridge Max Span** or leave room for safe landings, then regenerate. A retained 3D preview with a connectivity error is not a validated gameplay map.

Plateau ramps or surfaces are hard to read

Assign a distinct `Plateau Material` on `MapDefinition` or the active `BiomeSetting`; when it is missing, plateau tops intentionally fall back to the grass material. Increase `Plateau Relief Scale` for a stronger height separation, or lower `Plateau Contour Irregularity` if the outline is too pronounced for the map size.

Plateau cliff borders are non-walkable by design. Dirt paths may change tier only through planned ramps. If a very dense setup leaves too little room for clean access, reduce the expansion count or choke count, reduce `Plateau Footprint Scale`, or increase the map size.

Ramp GameObjects use the wrong Unity Layer

Set `Walkable Layer (ramps and bridges)` on the map and make sure a Unity Layer with exactly that name exists under **Project Settings > Tags and Layers**. Ramp meshes and colliders are generated under `Ramp_*` GameObjects on this shared layer. When it is missing, the package falls back to the configured ground layer, then `Default`, and reports a validation warning. The hidden legacy biome `RampLayerName` field is ignored.

The ramp surface deliberately reuses `Dirt Path Material`. The worn areas immediately before and after it are separate `DirtPath_RampApproach_*` meshes so they remain visible even when curved path smoothing replaces the regular grid path mesh.

Cliff prefabs leave gaps, float or face the wrong direction

Cliff assets should contain at least one `Renderer`. Their root pivot may be offset: placement uses all child-renderer bounds and aligns the rendered top to the high ground automatically. Set `Cliff Prefab Length Axis` to the local axis that runs along the wall, or keep `Auto` when that axis is simply the prefab's largest horizontal extent.

Increase `Cliff Segment Overlap` slightly if a very irregular rock mesh exposes tiny corner seams. `Cliff Vertical Overlap` hides the joins above and below the terrain wall. `Cliff Height Scale` is a coverage multiplier and should stay at one or higher; older serialized values below one are clamped to full coverage.

Generated cliff instances are direct children of the configured `Cliffs` parent. Straight terrain edges are tiled continuously; cliff sides follow ramps without closing their passages. Mountain outcrops remain inside their massif. Enable `Mark Cliff Cells Unbuildable` when gameplay construction must respect walls.

Custom project layers are missing

The package can use the optional `Ground`, `Environment` and `Aerial` layers when they exist. Missing layers are safely ignored and do not block generation. Add them to the project only when your gameplay collision or rendering setup needs them.

The top-down screenshot is missing

Enable `Generate Screenshot After Generation`, verify that the output folder is writable, and make sure `Screenshot Layer Mask` includes at least one generated renderer. A relative output folder targets the project root in the Editor and `Application.persistentDataPath` in a build.

Automatic screenshots require a graphics device. Unity sessions launched with `-nographics` skip the capture and report a warning while map generation remains successful. Use a normal Editor session or a rendered player build when an image is required.

The generator exposes the final absolute path through `LastScreenshotPath` and `ProceduralMapGenerationResult.ScreenshotPath`. If the output is inside the project's `Assets` directory, Unity imports the PNG automatically.

A* Pathfinding is not installed

A* Pathfinding Project is optional and is not bundled. When compatible A* components are present, the package configures them through their public component names; otherwise map generation continues without pathfinding setup.

Regenerating the starter demo

`Create Starter Demo Scene` is safe to run again. Existing starter materials, prefabs and biome references are preserved rather than reset, so buyer customizations are not overwritten.

Getting useful diagnostics

Configuration errors and important warnings are always logged. Add `SOROLIF_MAPGEN_VERBOSE_LOGS` to **Project Settings > Player > Scripting Define Symbols** only when detailed subsystem traces are needed.

If the issue remains reproducible, contact `support@sorolif.com` with the Unity version, render pipeline, package version, seed, relevant `MapDefinition` settings and the complete Console error. A minimal project or source image is especially useful for import-related issues.

Integrated Editor help

Current English help topics exported from the compiled Editor. These workflows complement the technical chapters and mirror the steps shown in the How To Use window.

Parameters and conditions

Know when each setting applies and compare its effect.

1. Compare one setting at a time

Disable Use Random Seed, keep the seed and source assets fixed, change one setting and regenerate. Feature switches, generation mode and safety constraints determine whether a setting can affect the result. The parameter reference lists every recipe and asset field with its consumer and conditions.

2. Automatic output and imported terrain

Water Height stays at -0.5 before Map Altitude. Cliff Parent and Resource Parent display generated output locations. Imported Mountain Height controls painted peaks; procedural mountain count, width and footprint settings apply only to Procedural mode. Random water layouts use their resolved layout for bridge and oil generation.

3. Density, fairness and exports

Forest Density is a percentage, including 1-5%. Forest Zone Count Multiplier changes fragmentation at comparable density: 1 creates broad woods, 10 creates many smaller groves. Older serialized density multipliers migrate once. Resource Fairness Tolerance changes the starting-resource score, not placements. Mesh Optimization preserves appearance. Screenshot settings affect the PNG. Prop Scale selects patch size; Color belongs to the standalone legacy color scanner.

Your first map

Start with the supplied demo, then change one setting at a time.

1. Create the starter scene

Click Create Starter Demo Scene below and choose a scene location. The wizard prepares four independent biome settings (Forest, Tropical, Snow and Autumn) referencing Core/Art, a camera and a generated map. Unity asks about saving modified scenes before replacing the active scene.

2. Select the generator

Select the GameObject with MapGenerator in the Hierarchy. Its Simple Inspector shows Map Size, Generation Mode and Biome preset. The Select Generator button finds it in the open scene; if several exist, select the intended one first.

3. Make a reproducible change

Keep Use Random Seed off while learning. Change Seed or Biome preset, then click Generate RTS Map in the Inspector. Regeneration replaces the output below this generator. Save any hand-edited generated objects separately before regenerating.

4. Check the result

Read the Inspector validation messages and inspect bases, access ramps and resources. Top View exports a PNG after generation. Use Expert only when you need finer control; switching views preserves the recipe.

Procedural layout

Build the strategic layout before adding visual detail.

1. Choose a baseline

Set Generation Mode to Procedural. In Simple, Balanced favors mixed land, High Ground emphasizes plateaus, and Naval starts an island layout. These buttons change several recipe values: use Undo to revert a preset.

2. Size, players and expansions

Map Size supports 30-200 cells per side; Cell Size controls their world-space size. Adjust Player Count, Expansions Per Player, Contested Expansion Count and Choke Points. Contested expansions vary with the seed across free terrain: the complete set balances estimated access from player bases and fills underused sectors. Individual sites may favor different players instead of all lying on medians. Dense settings need more free space: enlarge the map or reduce the requested features when validation reports missing placements.

3. Force strategic passages

Set Choke Points above zero to separate player territories with continuous forest or rock barriers according to the terrain settings. Roads must pass through their narrow openings. Islands forms straits of varying width between irregular low islands, wooded when forests and matching props are enabled, with optional rocky cores when mountains are enabled. Nearby neutral expansion islands may join these formations without moving their resources or adding expansions. Player coastlines stay open. Planning runs after water, ramps, bridges and mountains; clearings, plateau tops and existing settlement access are preserved. Multiple openings spread along broad approaches, avoiding tiny pockets. Land paths share a collector with one town connection instead of separate roads to every gate. Adjacent ramp/deck openings share one marker. Optional gates are skipped when safe spacing is insufficient; access preservation may still require extra openings. Markers follow the final passages. Zero leaves the map unrestricted by these dividers.

4. Enable the terrain features

Enable water, forests, mountains, plateaus and cliffs as needed. Expert groups expose their densities, clearance and relief controls. A requested count is constrained by bases, routes, ramps and resources; mountains cannot occupy protected space.

5. Compare at a fixed seed

Disable Use Random Seed, keep Seed unchanged and modify one group at a time. Generate again to compare. The gameplay score measures strategic constraints; it does not rate visual quality.

6. Make a mountainous landscape

Enable Has Mountain Zones, then adjust Mountain Density Multiplier (3 by default). Mountain Zone Count is the base count for 80x80 land cells at density 1; the total follows land area. Use 0.5-1 for sparse ridges or 3-4 for many massifs. Mountain Max Area caps each formation. Strategic buffers and connectivity checks preserve passages; constrained terrain can reduce the achieved density.

Import an image

Turn a readable color map into semantic terrain using a palette.

1. Try the image demo

Create Image Import Demo provides a source image and matching palette. To use your own image, set Generation Mode to Image and assign Source Map Image.

2. Enable Read/Write

Select the source texture in Project, enable Read/Write in its import settings and Apply. Without readable pixels, generation stops with a validation error. Prefer a lossless image with clear, separated colors.

3. Assign the terrain palette

Use Image Layer Set (MapGeneratorImageLayer). Name semantic layers with Grass, Forest, DirtPath, Water or Lava, and Mountain prefixes. Auto Match uses the layer set's image-name patterns; assign the palette explicitly if it selects the wrong one. Without a set, Expert exposes legacy fallback colors.

4. Tune recognition, then relief

Generate first. Adjust Image sampling and color recognition only if regions are misclassified. Image contours and tile cleanup control boundaries; Imported ground relief adds seeded height variation. Import Image Resources is optional. Imported DirtPaths retain the authored footprint.

Map altitude

Add broad hills or use your own grayscale height map in either generation mode.

1. Enable altitude

Open Map altitude in Simple or Expert and enable Enable Map Altitude. Set Altitude Min Height and Altitude Max Height, then regenerate. These are offsets in map units: existing mountains, plateaus, seabed depth and recessed paths keep their own relief. Disable the mode and regenerate to restore the previous terrain.

2. Automatic hills

Leave Altitude Height Map empty. Altitude Noise Scale controls the size of hills; lower values give broader forms. Disable Use Random Seed and keep Seed fixed to reproduce the terrain. Automatic samples are normalized across the configured minimum and maximum.

3. Custom height map

Assign a grayscale Texture2D to Altitude Height Map. Enable Read/Write, disable sRGB and use Compression None in its import settings. Black maps to the minimum, white to the maximum, with bilinear sampling between pixels. The entire image fits the terrain footprint; Origin Top Left flips it vertically. This texture is independent of the color image used for terrain classification.

4. First altitude stage

All terrain and water vertices and their colliders follow the field. Props, resources and cliff prefabs move vertically while keeping their shape; bridge decks connect the elevated landings. Embedded meshes and saved recipes retain altitude. Walkable/buildable flags and the fairness score still describe the strategic layout, without a new slope audit. Lakes, seas and closed pools remain horizontal. Through-flow rivers share a gentle slope toward the lowest boundary mouth, capped at 8 percent; branches join continuously. The seabed and banks follow those levels, with a smooth transition into the surrounding height map. Water collision and optional minimap geometry retain their final levels on reload. Aerial is off by default; when enabled, its hidden mesh follows final ground and water with the configured Aerial Clearance (3 world units by default). Mountains and their rock feet do not raise its flight level: the continuous surface passes through them. Rock protruding above it remains a physical obstacle through the mountain colliders. Include that rock in any separate aerial navigation bake. The continuous collider and embedded geometry survive reload.

Waterfalls, terrain holes and slope-dependent land access come later.

Biomes and materials

Use a biome preset, then replace its illustrative assets with your art.

1. Create or select a preset

Create Starter Assets prepares four editable BiomeSetting presets (Forest, Tropical, Snow and Autumn) in Assets/SorolifMapGenerationStarter. Choose Biome preset in the Inspector. Assign your own trees and props in the preset's Decoration configs. Biome Override explicitly references the selected preset; directly assigned definition materials and prefabs take final precedence. Included art is shared under Core/Art; editing a Starter setting does not modify the Core setting. Additional enum values remain available for your own biome settings.

2. Edit the active materials

Select the active biome asset to find Grass, Forest, Dirt Path, Mountain, Plateau and Water materials. Duplicate a shared material into your project before changing its values, then assign the copy to the Starter setting. Create a project-owned prefab variant before editing a shared prefab. Plateau falls back to grass if no dedicated material is provided. Existing starter assets are preserved when the wizard runs again.

3. Choose triplanar shading

The supplied Natural and Stylized shaders project textures in world space. Top Texture covers horizontal surfaces; Cliff / Side Texture covers steep faces. World Texture Scale changes tiling. Start with subtle texture strength, then tune slope tint or stylized lighting on the material. Cast Shadow Strength controls directional shadows in Built-in. Water, WaterOnSnow, WaterTropical and WaterSwamp are included in Art/Materials/Water. Assign one to Water Material or choose a matching biome preset. Add StylizedWaterDepthCamera to your custom camera for depth colors, broken shore foam and shallow caustics in Built-in or an installed URP. Swamp uses a slow olive surface film instead of caustics.

4. Grass prefabs and instancing

In Terrain style, click Apply dense grass preset (screenshots), then Generate Map: chance multiplier 1, height 0.5-0.8. Older scenes retain their saved settings. Maps without grass rules use the included Medium/Large prefab rules in memory; custom grass rules remain authoritative. MapPropsGenerator places the largest Patch sizes first, then fills free tiles with smaller patches. Tile Percent is a probability per rule and Random Rotation is respected; disable yaw for grid-aligned large patches. Legacy TransparentFX prefabs named Grass are recognized without rewriting the biome. Keep TransparentFX on prefab and rule. Medium covers one by one cell; Large distributes 64 small tufts across four by four cells. Border blades overlap adjoining tiles; space their centers by one/four cells for Medium/Large. Soil color blends into the foot of each tuft and is retained by the included GPU instancer. Height changes only the blades, preserving ground coverage. Shared meshes, three LODs and an original RGBA texture supply the grass. Instances stay under Props with MapDataContainer for GPU instancing. See the grass chapter in the Technical Guide PDF.

5. Replace props and verify rendering

The included trees, rocks, resources and bridge are cube-based placeholders. Replace the biome prefab references with your own renderable assets and regenerate. This candidate is verified in Built-in; check your materials separately in URP/HDRP. Presets change visuals, not strategic rules.

Smooth terrain contours

Contour smoothing, path smoothing and ground relief serve different purposes.

1. Procedural contours

In Simple Terrain style or Expert Mesh and decoration, keep Enable Procedural Contour Smoothing on. Procedural Contour Smoothing Strength controls forest, mountain and shoreline geometry. Start from 0.85; disabling it or setting strength to zero restores the unsmoothed contour behavior.

2. Imported contours

In Image mode, open Image contours and tile cleanup. Enable Image Contour Smoothing and its strength relax the outline. Image surface smoothing cleans isolated tile regions and can affect classification; check it separately when preserving fine painted details.

3. Understand protected areas

Contour deformation shares render and collision vertices. Strategic plateau edges, ramps, bridge approaches and physical resource anchors can remain pinned to keep access valid. Grid gameplay flags are still cell-based; do not expect every protected corner to become rounded.

4. Tune paths independently

Procedural roads stay curved with Enable Path Smoothing off. The option refines their contours without changing the network. Towns connect to expansions and strategic arteries; local branches may end in pointed dead ends. Imported path footprints remain authored.

Dirt paths and bases

Control road density, curved base arms and their relationship to ramps.

1. Use curved paths

Open Expert > Mesh and decoration > Dirt paths and wandering routes. Curved roads and town streets are generated in both smoothing modes. Enable Path Smoothing rounds their contours more finely. Local streets vary with Seed and taper at free ends; junctions and ramp mouths keep their width.

2. Reduce visual clutter

Path Connection Count starts with essential settlement access; higher values add alternative links. Random Long Path Count adds map-spanning routes. Path Branch Count adds decorative branches beyond the local town streets. Wandering Path Count adds independent meandering trails; zero disables those trails.

3. Check ground and ramp joins

Dirt paths and ramp approaches are recessed by 0.02 Unity units with matching colliders. This depth stays fixed when Cell Size changes. Base arms shorten near water, mountains, resources and height breaks. Missing arms in a crowded area can be intentional clearance.

Water, plateaus and cliffs

Keep routes and ramps readable while tuning height and water layout.

1. Choose a water layout

Enable Has Water and choose Water Layout. Expert exposes river, island and lake settings. Additional Naval Zones adds secondary features; zero keeps the main lake or river. Rivers avoid main-base and expansion clearings using Player Base Forest Clear Radius and Expansion Forest Clear Radius, even without forests. The primary river keeps the center when free and skirts it when occupied. Bridges require Has Bridges and a usable River Bridge Prefab. On River layouts, a river crossing both sides of a choke path is preserved beneath a bridge perpendicular to its local

course. Oblique paths reconnect to the dry landings. A river following the path, ending against it or meeting a path dead end does not request a path bridge.

2. Make high ground readable

Use a distinct Plateau Material and tune Plateau Relief Scale. Ramps are planned passages between tiers; paths cannot climb a cliff elsewhere. If access placement fails, reduce plateau footprints, expansion density or choke count, or enlarge the map.

3. Fit your cliff prefabs

Enable Has Cliffs and supply prefabs with Renderers. Cliff Prefab Length Axis defines the axis along the wall. Tune segment and vertical overlap for joins. Cliff instances, including mountain details, are direct children of Cliffs. Ramp openings stay clear.

4. Verify layers and shores

Ramps and bridge decks use the map Walkable layer (Bridge Layer Name). No dedicated ramp layer is needed. Missing layers fall back safely. Shore smoothing aligns ground and water geometry; the image import's visual underwater overlap does not turn neighboring ground cells into water.

5. Embed rocky mountain faces

Enable Mountain Cliff Details and assign Mountain Cliff Prefabs in the biome or recipe, independently of the wall/plateau Cliff Prefabs. Rocks keep their authored proportions with uniform scaling, follow ridge directions and lean into the slope. Smaller pieces fill safe gaps and narrow massifs; bases stay buried. Mountain Cliff Material overrides the mountain terrain material fallback; empty sources preserve prefab materials. Detail Scale controls size. Complete tilted renderer and collider bounds stay on mountain cells, clear of roads, ramps and reserved passages.

Export a top view

Capture the generated map and locate the saved PNG.

1. Capture the current map

Generate a map, then click Top View in the Inspector. The button becomes available when a logical grid exists. For automatic captures, enable Generate Screenshot After Generation.

2. Set framing and destination

In Expert > Screenshot output, set output folder, dimensions, padding, view mode and layer mask. Relative folders are inside the project root in the Editor and persistentDataPath in a player. A PNG saved under Assets is imported into Project.

3. Resolve missing captures

Check the Console path, folder permissions and Screenshot Layer Mask. Rendering needs a graphics device; -nographics skips screenshot capture. Runtime code can read result.ScreenshotPath or generator.LastScreenshotPath.

Optional outputs and layers

Generate only the helper meshes your game needs and configure its project conventions.

1. Choose helper outputs

In Simple > Output, enable Generate Minimap Layer or Generate Aerial Layer, then regenerate. Both are disabled by default, including on recipes without these new fields. Expert > Optional outputs and project layers exposes the same settings. Minimap creates a separate water rendering copy without colliders; normal water stays present when it is disabled.

2. Set flight clearance and layers

Aerial Clearance defaults to 3 world units above ordinary ground and water. Mountains intersect the hidden flight collider and remain obstacles. Select existing Unity layers using Ground, Environment, Water, Bridge, Minimap and Aerial Layer Name. Missing names use documented safe fallbacks. Image palette and biome prop/ramp layers keep their explicit overrides; Additional Prop Placement Layers extends the placement mask.

3. Enable integrations deliberately

Enable Navigation Graph Updates is off by default. It opts into an installed compatible A* GraphUpdateScene API, using configurable Water and Aerial Navigation Tags (0-31). No navigation package is required. The map River Bridge Override takes priority over BiomeSetting > Biome Prefabs > River Bridge Prefab, then the optional Editor Bridge Prefab Search Path. Use biome / default bridge clears the map override; clear the path to disable automatic lookup. Keep a serialized prefab reference for player builds.

Use it in your game

Generate from code and connect the returned grid to your gameplay.

1. Prepare the generator

Configure a MapGenerator in your scene and keep a serialized reference in your bootstrap script. Assign a Biome Override explicitly when multiple assets share a biome type. The runtime sample is in Examples for classic imports or in Package Manager > Samples for UPM installs.

2. Generate and check success

Call TryGenerateRuntimeMap(out ProceduralMapGenerationResult result). Check its Boolean before using Grid, MapRoot or player spawn positions. On failure, read result.ValidationIssues. Save the example below as MapBootstrap.cs, attach it to a GameObject and assign its generator in the Inspector.

3. Integrate during a loading phase

Generation is synchronous and cost grows with map size. Use a loading phase for larger maps. Grid.GetCell(x, y) exposes Walkable, Buildable and semantic terrain flags. Unit control, combat and navigation are your game's responsibility; A* is optional.

4. Regenerate deliberately

A new generation replaces this generator's previous output. Do not retain old MapRoot or cell references for a new match. Use Source Map Image and an Image Layer Set to generate from an image prepared by your own tooling.

Resolve a problem

Start with validation and change the smallest relevant part of the recipe.

1. Generation fails or the map is empty

A failure leaves a visible 3D preview, but Succeeded remains false. Connectivity audit errors let all remaining visual stages finish, including mountains, cliffs, props and altitude. Read the Inspector errors and Console warnings; fix the

recipe and regenerate before playing or saving. For bridges.connectivity, increase River Bridge Max Span, reduce River Width or clear wider, flatter landings. PreviewRoot gives runtime access to the preview. Check Map Size, Player Count, Cell Size and, in Image mode, the source and Read/Write. Validate Selected Generator checks the recipe; Refresh Check checks project readiness.

2. Materials or props are missing

Create Starter Assets or assign your own biome and references. Check feature switches and direct material overrides, which take priority over the biome. For pink materials, inspect the shader and pipeline compatibility in the Console.

3. There is not enough room

Lower expansion/choke counts, mountain footprints or clearance, or increase Map Size. Keep the seed fixed to isolate the setting responsible. Some requested decorative details can be omitted to preserve bases, resources and legal routes.

4. Prepare a reproducible report

Keep the Unity and package versions, pipeline, seed, recipe and complete Console message. Include the palette and source image for import problems. The offline Troubleshooting guide lists feature-specific fixes.