TESSARACTIC INFINITE PROCEDURAL TERRAIN

TECHNICAL DOCUMENTATION

Version 1.0 • Unity 2022.3 LTS • Built-in RP & URP

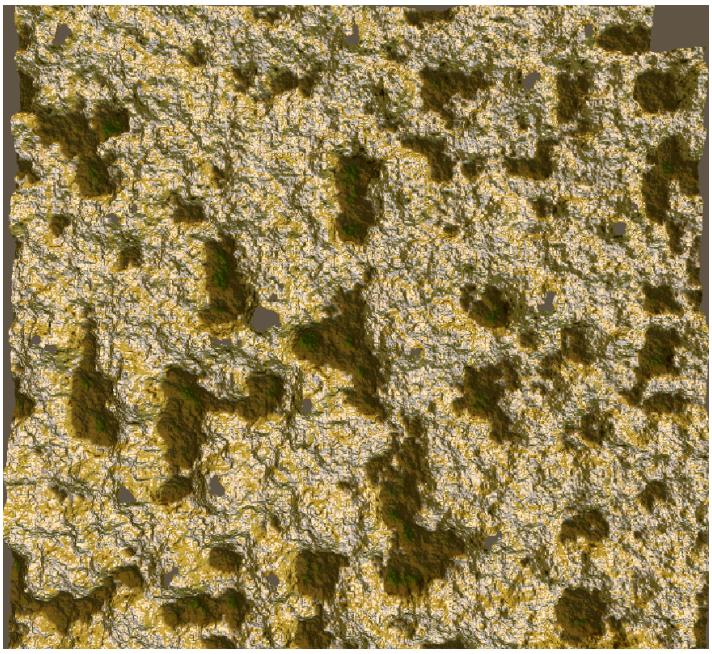
TESSARACTLAB

00 Table of Contents

01 Package Overview

Tessaractic is a fully GPU-driven, infinite procedural terrain system for Unity. It combines fractional Brownian motion (FBM) noise with a proprietary octree subdivision algorithm — Frank's Equation — to generate geologically credible landmasses at runtime with no pre-baking, no heightmaps, and no CPU mesh work. Every triangle is produced on the GPU via a compute-based Marching Cubes pipeline and streamed into the world through an infinite chunk treadmill as the player moves.

The result is terrain that looks sculpted rather than generated: rolling hills, cliff faces, rock formations, and cave-like overhangs all emerge from the same mathematical pipeline. The included terrain shader (auto-configured for both Built-in RP and Universal RP) adds triplanar Parallax Occlusion Mapping, height-band biome blending, Blinn-Phong lighting with rim and fill passes, wetness simulation, and atmospheric fog.



Bird's-Eye View of Active Voxel Chunks (Low Quality Profile) — This is the active terrain grid footprint running on the Low VRAM profile. Moving to Mid, High, or Ultra profiles dynamically increases active chunk density, rendering a drastically larger horizon grid.

Who It Is For

- Game developers — who need a high-quality, ready-to-ship infinite terrain system without writing compute shader code.
- Technical artists — who want a fully exposed shader with per-biome colour bands, slope-based material blending, and POM depth control.

⚠ Tessaractic targets the Unity Built-in Render Pipeline & URP exclusively, Auto configure according to your template HDRP are not supported in Version 1.

02 Quick Start / Setup Guide

> For the Demo scene, simply import the package, open the Demo scene, and press Play. The following steps apply only if you are setting up Tessaract from scratch in a new or existing project.

Follow these steps in order immediately after importing the package.

Step 1

Import the Package

In Unity: Assets → Import Package → Custom Package. Select Tessaractic_v1.unitypackage and ensure all files are ticked before importing.

Automatic URP Setup: On first import, Tessaractic automatically installs Universal RP shader headers if not already present. This is a 10–60 second one-time process. Your active render pipeline (Built-in or Universal) is not changed.

Step 2

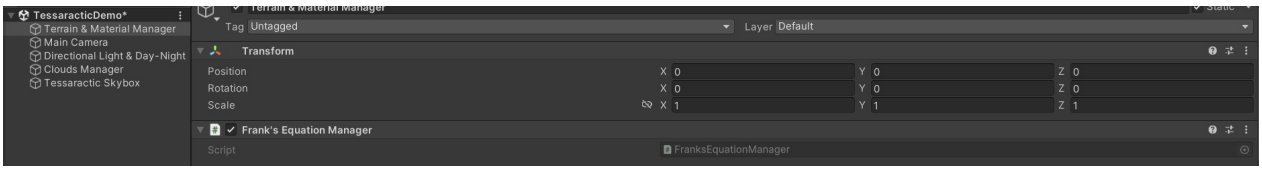
Create an Empty GameObject

In the Hierarchy, right-click → Create Empty. Name it Terrain & Material Manager.

Step 3

Attach FranksEquationManager

Add the FranksEquationManager.cs component to the empty GameObject. This is the master controller for the entire terrain pipeline.

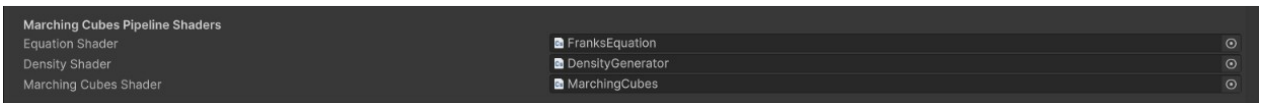


Step 3 — FranksEquationManager component attached to the TessaracticWorld GameObject in the Inspector.

Step 4

Assign Compute Shaders

In the Inspector, assign the three compute shaders to their slots: Franksequation.compute → Equation Shader, DensityGenerator.compute → Density Shader, MarchingCubes.compute → Marching Cubes Shader.



Step 4 — All three compute shader slots filled: FranksEquation.compute, DensityGenerator.compute, MarchingCubes.compute.

Step 5

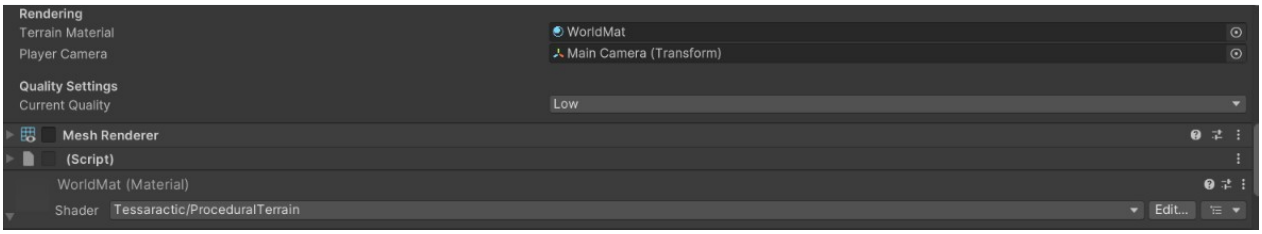
Set Scale & Octave

In the FBM Terrain Shape section of the Inspector, set Terrain Scale to 0.015 and Terrain Octave to 8. These values control the noise frequency and detail layers used for terrain generation.

Step 6

Create the World Material

Project panel → Create → Material. Name it WorldMat. Set the Shader to Tessaractic/ProceduralTerrain. Assign terrain textures (Ground, Snow albedos and their PBR maps) to the appropriate slots.



Step 6 — WorldMat material assigned in the Terrain Material slot with Tessaractic/ProceduralTerrain shader selected.

Step 7

Assign Material and Camera

On FranksEquationManager, assign WorldMat to Terrain Material and drag your Main Camera into the Player Camera field. Select Main Camera in the Hierarchy. Drag TessaracticFlyCamera.cs from the Project panel onto it in the Inspector. This enables: WASD — movement, Shift — sprint, cursor — look and mouse sensitivity - 2, Lock Cursor - Tick.

Note! Ensure that the 'Main Camera' GameObject has its Tag set to 'MainCamera' in the Inspector to prevent runtime errors.

Step 8

Add a Directional Light

The shader reads Unity's directional light automatically via AutoLight.cginc. Ensure a Directional Light exists in the scene as the primary sun. Drag this inside Environment tab in Sun source slot, sun intensity inside 'Directional Light' 3.5, Tick soft shadows, angle Rotation X 10, Y 45, Z 0., select 'Very High resolution'

Step 9

Set up Day-Night Cycle

Attach the TessaracticDayNightCycle component to your main Directional Light (or any active controller GameObject in the scene). In the Inspector, assign your primary Directional Light to the Sun Light field (if left empty, it will auto-detect). Assign the active FranksEquationManager component to the Terrain Manager field (it will auto-detect if left empty). Set the Time Progression Speed field to choose how many real-world minutes represent a full 24-hour cycle.

Step 10

Set up Procedural Skybox

Right-click in your project hierarchy → Create → Material. Name it Skybox_Material. Set the shader of Skybox_Material to Tessaractic/ProceduralSkybox. Attach the TessaracticSkybox component to an empty GameObject in your scene (e.g. name it SkyboxManager). In the Inspector, assign your Skybox_Material to the Skybox Material field. (Optional) Leave the Day Night Cycle empty to allow auto-detection.

Step 11

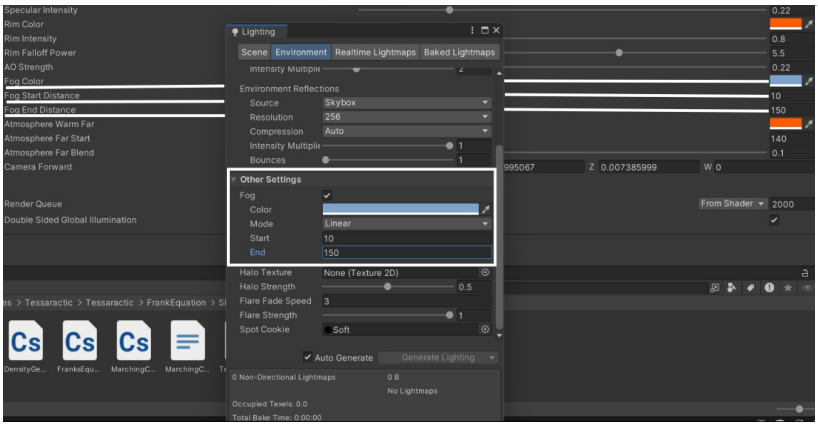
Set up Procedural Clouds

Right-click in your project hierarchy → Create → Material. Name it Cloud_Material. Set the shader of Cloud_Material to Tessaractic/ProceduralClouds. Attach the TessacticClouds component to an empty GameObject in the scene (e.g. name it CloudManager). In the Inspector, assign your Cloud_Material to the Cloud Material field. Leave the Day Night Cycle and Terrain Manager empty (the script will auto-detect them at runtime). Adjust Cloud Altitude, Cloud Density, and Wind Speed fields directly to style your sky.

Step 12

Configure Scene Fog

Window → Rendering → Lighting → Environment tab. Enable Fog, set Mode to Linear, Start to 10, End to 150. Match the Fog Color to _FogColor in the WorldMat for seamless depth. Make sure to enable 'Auto generate' tick shown in Fog configuration image.



Step 12 — Fog enabled in Window → Rendering → Lighting → Environment. Mode: Linear, Start: 10, End: 150.

Step 13

Set Quality Preset

Under Quality Settings in the Inspector, select Low, High, or Ultra. Start with Low to confirm everything is working before scaling up.

Step 14

Assign Mesh Renderer Material

Under Quality Preset → click 'Add Component' → select 'Mesh Renderer' → expand the Mesh Renderer component. Under the Materials section, assign the 'WorldMat' material. Toggle the 'WorldMat' material to reveal the PBR textures and verify that all inspector settings are displayed with their pre-configured values.

Step 15

Press Play

The terrain generates chunks around the camera immediately. Use WASD + mouse with the included TessaracticFlyCamera.cs for navigation.

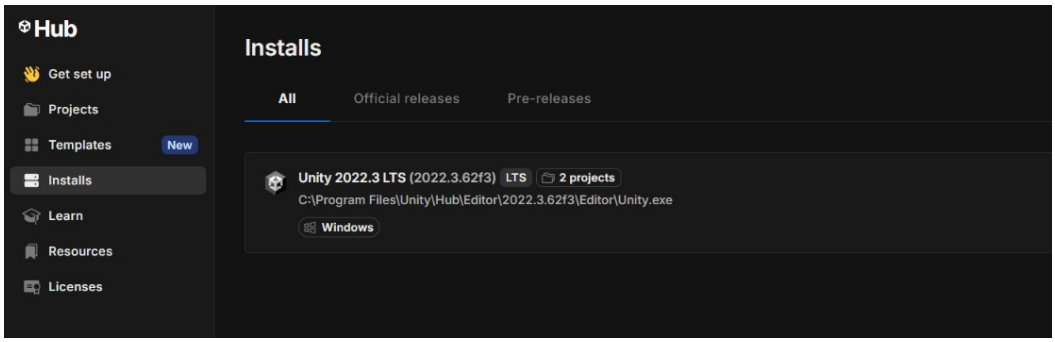
> ⓘ On first Play the GPU allocates triangle buffers. On lower-end hardware this may cause a 1-2 second hitch. This is expected and only occurs on startup.

On first import, Tessaractic automatically installs Universal RP shader headers if not already present. This is a 10–60 second one-time process. Your active render pipeline (Built-in or Universal) is not changed. Both Built-in RP and URP are fully supported via auto-configuration.

03 Technical Requirements

Unity Version

Requirement	Value	Notes
`Tested / Recommended`	Unity 2022.3 LTS	Developed and validated on 2022.3.62f1
`Minimum (expected)`	Unity 2021.3 LTS	No version-specific APIs used; earlier LTS versions likely compatible
`Render Pipeline`	Built-in Render Pipeline & Universal Render Pipeline, only	HDRP is NOT supported in v1
`Shader backend`	Legacy CG (UnityCG.cginc)	AutoLight.cginc and Lighting.cginc included
`Script IDE`	Visual Studio 2022 – 2026	For editing open-source C# files



Unity Hub — 2022.3 LTS selected as the project Unity version.

Hardware Requirements

Tier	GPU VRAM	Expected Performance	Recommended Preset
`Minimum`	4 GB VRAM	~30 FPS at Low settings. Playable; some GPU buffer stall risk under heavy load.	Low
`Recommended`	6 – 8 GB VRAM	Stable 60+ FPS at High settings with comfortable headroom.	High
`Ideal`	10+ GB VRAM (current-gen)	Smooth 60–120+ FPS at Ultra with full visual fidelity.	Ultra

> ⓘ The entire terrain pipeline runs on the GPU. VRAM is the primary hardware constraint, not CPU. A fast CPU paired with 4 GB VRAM will underperform a modest CPU with 8 GB VRAM at High preset.

Platform Support

Platform	Support	Notes
`Windows (DX11 / DX12)`	Supported	Primary development and test platform
`macOS (Metal)`	compatible	Not formally tested; CG shaders compile via Metal in Unity
`Linux (Vulkan / OpenGL)`	Likely compatible	Not formally tested
`WebGL`	Not supported	Compute shaders are unavailable in WebGL
`Console / Mobile`	Not supported	Compute shader requirements and 4 GB VRAM floor preclude these targets

Package Dependencies

Tessaractic has no Unity Package Manager (UPM) dependencies. It does not require Burst Compiler, Mathematics, Collections, or any other package beyond a standard Unity installation. All pipeline code is self-contained across the six core asset files.

Included Files

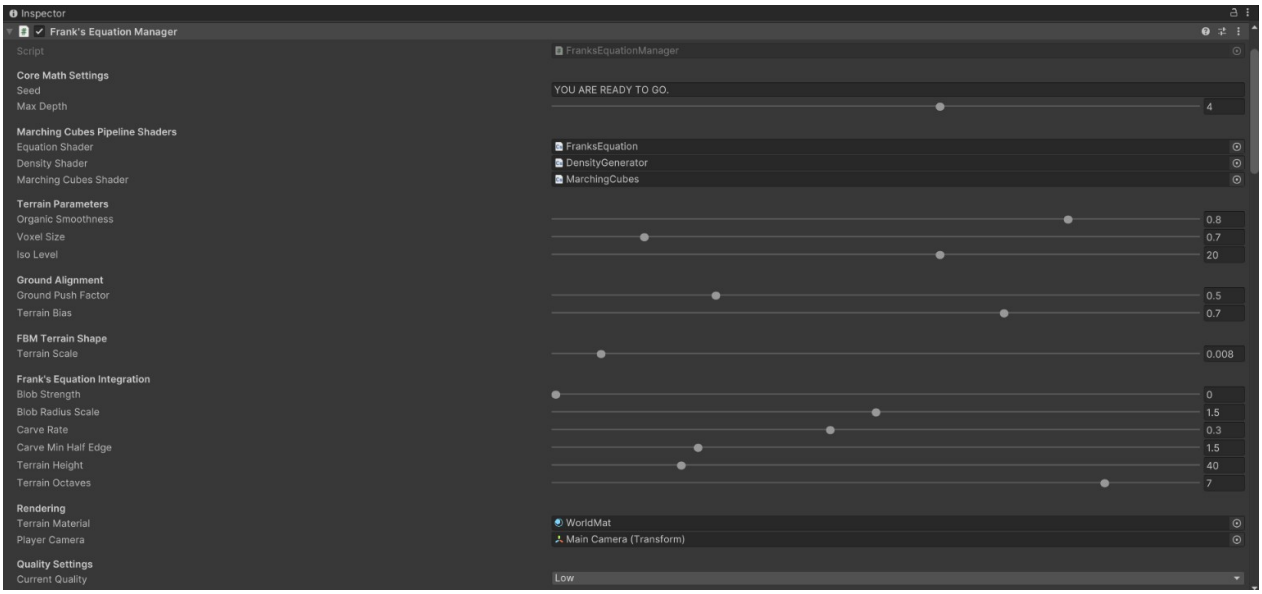
File	Type	Open Source	Role
`FranksEquationManager.cs`	C# MonoBehaviour	Yes	Master controller: chunk treadmill, quality presets, full pipeline orchestration
`Franksequation.compute`	HLSL Compute	Yes	Octree subdivision kernel (SplatCubes) implementing Frank's Equation
`DensityGenerator.compute`	HLSL Compute	Yes	FBM density field generator; feeds octree data into the voxel grid
`MarchingCubes.compute`	HLSL Compute	Yes	GPU Marching Cubes extractor; outputs 96-byte Triangle structs with smooth normals
`ProceduralTerrain.shader`	CG Shader	Yes	Full terrain shader: triplanar POM, biome blending, Blinn-Phong, atmospheric fog
`MarchingCubesTables.cginc`	HLSL Include	Yes	Edge and triangle lookup tables for Marching Cubes
`TessaracticFlyCamera.cs`	C# MonoBehaviour	Yes	Optional fly-camera: WASD movement, sprint, cursor lock, configurable speed
`TessaracticDayNightCycle.cs`	C# MonoBehaviour	Yes	Controls time progression and sun rotation for the day-night cycle
`ProceduralSkybox.shader`	CG Shader	Yes	Procedural skybox shader for dynamic environment rendering

File	Type	Open Source	Role
`TessaracticSkybox.cs`	C# MonoBehaviour	Yes	Manager component for the procedural skybox material
`ProceduralClouds.shader`	CG Shader	Yes	Procedural clouds shader with adjustable altitude and density
`TessaracticClouds.cs`	C# MonoBehaviour	Yes	Manager component for styling procedural clouds and wind

All 12 files ship as editable open source 'Customize Friendly'.

04 Inspector Field Reference

All fields belong to the FranksEquationManager component. Fields marked □ are part of the confirmed working configuration and must not be altered without understanding the cascade effects described in Section 07.



FranksEquationManager — complete Inspector view with all fields visible.

Core Math Settings

Field	Type	Default	Range	Description
`seed`	string	"my secret dragon 123"	—	Deterministic seed string. Identical seeds produce identical terrain on any machine. Per-seed shape variation is a V2 feature; in v1 the FBM shape is calibrated to fixed terrainScale and terrainHeight values.
`maxDepth`	int	5	1 – 6	Octree subdivision depth for Frank's Equation. Higher values produce finer fractal rock detail at the cost of more GPU dispatch threads. Values above 5 rarely produce visible improvement at 0.5 m voxel resolution.

Compute Shader Slots

Drag the three compute shader assets from the Project panel into these slots.

Field	Asset to Assign
`equationShader`	Franksequation.compute
`densityShader`	DensityGenerator.compute
`marchingCubesShader`	MarchingCubes.compute

Terrain Parameters

Field	Default	Range	Description
`organicSmoothness`	0.8	0.0 – 1.0	Normal interpolation smoothness across the Marching Cubes surface. Higher values produce softer, more geological-looking terrain.
`voxelSize` ★	0.5	0.01 – 5.0	Size of each grid voxel in metres. At 0.5, a 64-grid chunk is 32 m wide. Do not change: FBM calibration and fog tuning assume this exact value.
`isoLevel` ★	0.0	−90 to 100	Density threshold for surface extraction. FBM convention requires 0. Non-zero values produce incorrect surface placement.

Set the voxel size to 0.75 for no breakouts seamless infinite world generation.

Ground Alignment

Field	Default	Range	Description
`groundPushFactor`	0.5	0.0 – 2.0	Pushes the density grid below the chunk centre. 1.0 = grid starts at chunk centre. Adjust together with terrainBias.
`terrainBias`	0.7	0.0 – 1.0	Strength of the vertical density gradient. 0.0 = floating Frank's Equation structures only. 0.7 = strong ground plane with formations growing upward. Start at 0.7 and adjust in 0.05 increments.

FBM Terrain Shape

Field	Default	Range	Description
`terrainScale` ★	0.01	0.001 – 0.1	Horizontal noise frequency. Larger = broader mountains; smaller = tighter ridges. Calibrated against terrainHeight: change both together when experimenting.
`terrainHeight` ★	40	1 – 200	Vertical noise amplitude in world units. Directly scales mountain height. Fog range and FBM calibration assume this default.
`terrainOctaves`	4	1 – 8	FBM noise layers. More octaves add high-frequency surface detail at increased GPU cost per dispatch. 4 is the quality/performance sweet spot.

Frank's Equation Integration

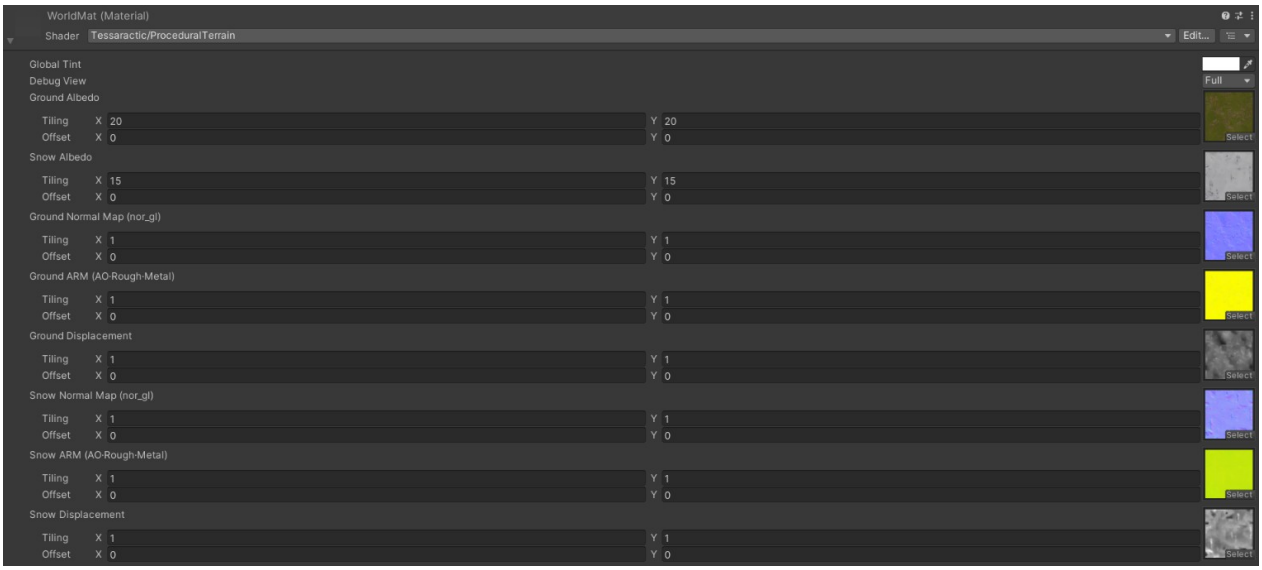
Field	Default	Range	Description
`blobStrength` ★	0.0	0.0 – 0.5	How strongly octree cubes push the terrain surface upward. 0.0 = pure FBM; Frank's Equation has no visible effect (v1 confirmed default). 0.2 = subtle rocky outcroppings. 0.5 = dramatic spires. Must be set to 0 in the Inspector for the stable configuration.
`blobRadiusScale`	1.5	1.0 – 2.0	Gaussian influence radius as a multiple of cube half-edge. 1.0 = sharp rocky edges. 1.5 = feathered blending. 2.0 = soft pillow formations.
`carveRate`	0.35	0.0 – 0.7	Fraction of octree cubes removed per subdivision level. 0.35 = 35% carved per level, producing natural cave-like sparsity. Lower = denser formations; higher = hollow or spiky results.
`carveMinHalfEdge`	1.5	0.5 – 5.0	Cubes with a half-edge below this value are never carved. Preserves fine fractal detail at the smallest scale. Keep close to voxelSize for best results.

Rendering & Quality

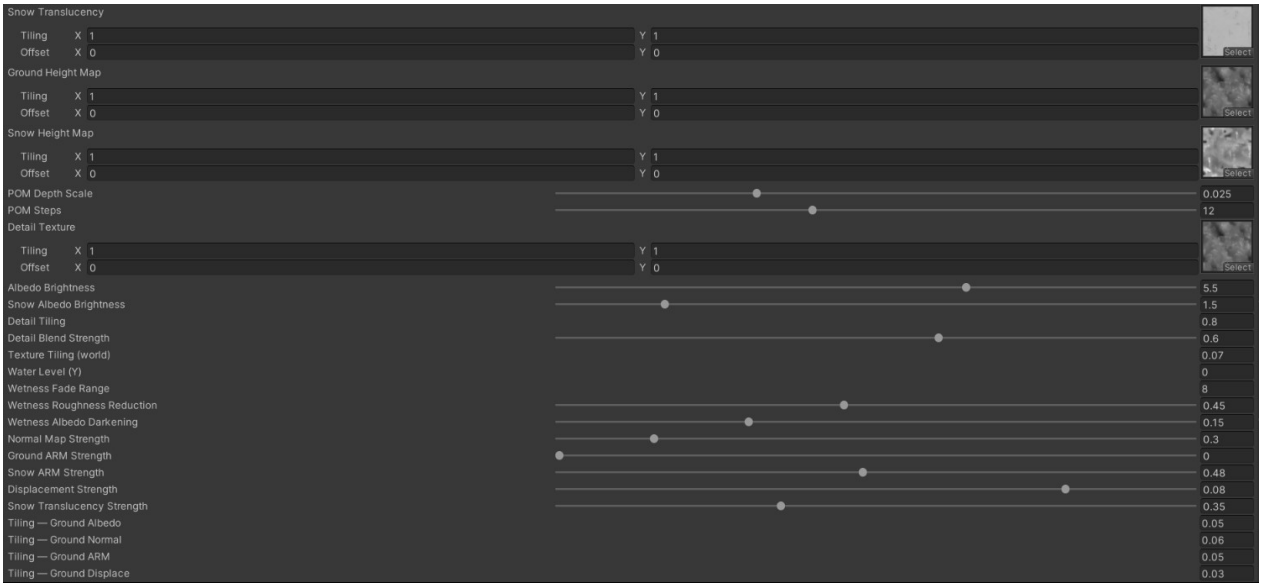
Field	Type	Description
`terrainMaterial`	Material slot	Assign the WorldMat material configured with Tessaractic/ProceduralTerrain shader.
`playerCamera`	Transform slot	Camera Transform used as the treadmill anchor. Chunks stream around this position.
`currentQuality`	Enum: Low / High / Ultra	GPU quality preset. Changing at runtime triggers full buffer reallocation and chunk re-stream. See Section 06 for full preset details.

05 Shader Property Reference

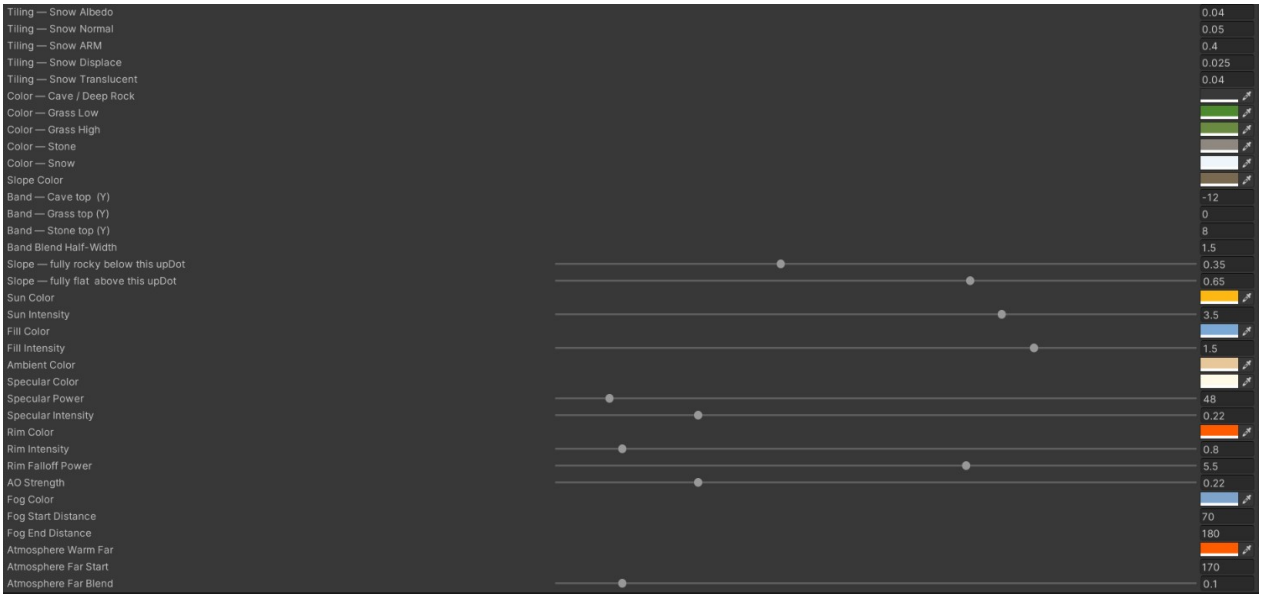
All properties belong to the Tessaractic/ProceduralTerrain shader, set on the WorldMat material. Properties are grouped as they appear in the Material Inspector.



WorldMat shader properties — Part 1 of 3: Global & Debug, Texture Layers, POM & Detail, and Biome Colours.



WorldMat shader properties — Part 2 of 3: Height Bands, Slope Blending, Lighting, Specular, and Rim Light.



WorldMat shader properties — Part 3 of 3: Wetness, PBR Blend Controls, and Fog & Atmosphere.

> Make sure to tick the Global Illumination slot at the end.

Global & Debug

Property	Default	Type / Range	Description
`_TerrainColor`	(1,1,1,1)	Color	Global tint multiplied over the entire terrain albedo output.
`_DebugView`	0 (Full)	Enum: Full / BiomeFlat / Normals / Height	Debug visualisation. Full = final render. BiomeFlat = flat biome colours. Normals = world-space normals. Height = raw height band gradient.

Texture Layers

Property	Default	Description
`_TexGround`	white	Ground albedo. Applied to flat low-slope surfaces.
`_TexSnow`	white	Snow albedo. Applied above the stone height band.
`_TexGroundNormal`	bump	Ground normal map (OpenGL convention, nor_gl).
`_TexGroundARM`	white	Ground ARM: R = Ambient Occlusion, G = Roughness, B = Metalness.
`_TexGroundDisp`	gray	Ground displacement / height map for POM.
`_TexSnowNormal`	bump	Snow normal map (OpenGL convention).
`_TexSnowARM`	white	Snow ARM: AO, Roughness, Metalness.

Property	Default	Description
_TexSnowDisp	gray	Snow displacement map for POM.
_TexSnowTranslucent	black	Snow translucency map. Controls backlit glow through snow.
_HeightMapSnow	black	Height map used for snow POM.
_TexDetail	gray	Macro variation / detail texture blended over albedo.

POM & Detail

Property	Default	Range	Description
_POMDepth	0.025	0 – 0.08	Parallax Occlusion Mapping depth scale. Higher = more pronounced surface depth illusion.
_POMSteps	12	4 – 24	POM ray-march step count. Higher = more accurate silhouette at added ALU cost.
_AlbedoBoost	5.5	1 – 8	Ground albedo brightness multiplier.
_SnowAlbedoBoost	1.5	1 – 4	Snow albedo brightness multiplier.
_DetailTiling	0.5	Float	World-space tiling frequency of the detail texture.
_DetailStrength	0.4	0 – 1	Blend weight of the detail texture over base albedo.
_TexTile	0.07	Float	Global world-space tiling scale applied across all layers.

Per-Layer Tiling Controls

Each texture layer has an independent world-space tiling value, allowing ground, snow, cliff, and displacement maps to be scaled independently for realistic variation without UV seams.

Property	Default	Affects
`_TileGroundAlbedo`	0.07	Ground albedo tiling frequency
`_TileGroundNormal`	0.07	Ground normal map tiling
`_TileGroundARM`	0.07	Ground ARM map tiling
`_TileGroundDisp`	0.035	Ground displacement map tiling
`_TileSnowAlbedo`	0.05	Snow albedo tiling
`_TileSnowNormal`	0.05	Snow normal map tiling
`_TileSnowARM`	0.05	Snow ARM map tiling
`_TileSnowDisp`	0.025	Snow displacement map tiling
`_TileSnowTrans`	0.05	Snow translucency map tiling

Biome Colours

Property	Default (RGBA)	Description
`_ColorCave`	(0.10, 0.09, 0.08, 1)	Deep rock / cave interior tint.
`_ColorGrassLow`	(0.25, 0.45, 0.15, 1)	Low-elevation grass colour.
`_ColorGrassHigh`	(0.20, 0.38, 0.12, 1)	High-elevation grass (slightly darker, cooler).
`_ColorStone`	(0.42, 0.40, 0.37, 1)	Stone / rocky midband colour.
`_ColorSnow`	(0.55, 0.62, 0.72, 1)	Snow tint (blue-grey, not pure white).
`_ColorSlope`	(0.50, 0.40, 0.30, 1)	Slope-face colour for cliff transitions.

Height Bands

Property	Default (Y)	Description
`_BandCaveTop`	−4.0	Y coordinate below which the cave/deep-rock biome is fully active.
`_BandGrassTop`	8.0	Y coordinate above which grass transitions to stone biome.
`_BandStoneTop`	16.0	Y coordinate above which stone transitions to snow biome.
`_BlendHalf`	2.0	Half-width of the crossfade zone between adjacent height bands. Larger = softer transition.

Slope Blending

Property	Default	Range	Description
`_SlopeRockBelow`	0.40	0 – 1	Dot product threshold below which the surface is fully cliff/rocky material. Dot product = surface normal vs world-up (0 = vertical face, 1 = flat ground).
`_SlopeFlatAbove`	0.72	0 – 1	Dot product threshold above which the surface is fully flat/grass. Values between the two thresholds blend linearly.

Lighting

Property	Default	Range	Description
`_SunColor` ★	(1.00, 0.92, 0.72, 1)	Color	Key light colour. Warm amber default matches a golden-hour aesthetic.
`_SunIntensity` ★	2.4	0 – 5	Key light diffuse intensity multiplier.
`_FillColor` ★	(0.40, 0.52, 0.72, 1)	Color	Fill light colour (sky bounce). Cool blue-grey default.
`_FillIntensity` ★	0.45	0 – 2	Fill light diffuse intensity. Increase to brighten shadowed faces.
`_AmbColor`	(0.18, 0.20, 0.26, 1)	Color	Ambient / shadow colour. Affects the darkest unlit surfaces.

Specular

Property	Default	Range	Description
`_SpecColor2`	(1.00, 0.96, 0.80, 1)	Color	Specular highlight colour.
`_SpecPower`	128	8 – 512	Blinn-Phong specular exponent. Higher = tighter, shinier highlight.
`_SpecIntensity`	0.15	0 – 1.0	Specular contribution multiplier.

Rim Light

Property	Default	Range	Description
`_RimColor`	(1.00, 0.75, 0.30, 1)	Color	Rim light colour. Warm amber creates a backlit silhouette glow.
`_RimIntensity`	0.12	0 – 8.0	Rim light strength.

Property	Default	Range	Description
`_RimPower`	3.0	1 – 8	Rim falloff exponent. Higher = tighter rim band at silhouette edges.

Wetness

Property	Default	Range / Type	Description
`_WaterLevel`	0.0	Float (World Y)	Y-coordinate of the simulated water surface. Terrain below this appears wet.
`_WetnessRange`	8.0	Float (metres)	Vertical fade range above water level where wetness transitions to dry.
`_WetnessRoughMul`	0.45	0 – 1	Roughness reduction multiplier for wet surfaces (more reflective).
`_WetnessAlbedoDark`	0.15	0 – 0.5	Albedo darkening amount for wet surfaces.

PBR Blend Controls

Property	Default	Range	Description
`_NormalStrength`	1.0	0 – 2	Global normal map intensity multiplier.
`_ARMGroundStrength`	1.0	0 – 1	Ground ARM map blend weight.
`_ARMSnowStrength`	1.0	0 – 1	Snow ARM map blend weight.
`_DispStrength`	0.02	0 – 0.1	Displacement warp strength applied before POM.
`_TranslucentStrength`	0.3	0 – 1	Snow translucency intensity. Controls backlit snow glow.

Fog & Atmosphere

Property	Default	Range / Type	Description
`_AOStrength` ★	0.25	0 – 1	Ambient occlusion strength blended from the ARM map's R channel.
`_FogColor` ★	(0.55, 0.65, 0.72, 1)	Color	In-shader fog colour. Should match the Unity Environment fog colour exactly.
`_FogStart` ★	128	Float (metres)	Distance at which in-shader fog begins blending in.
`_FogEnd` ★	178	Float (metres)	Distance at which in-shader fog reaches full opacity.
`_AtmosWarmFar`	(0.95, 0.86, 0.68, 1)	Color	Warm atmospheric haze colour applied at extreme distances.
`_AtmosFarStart`	400	Float (metres)	Distance at which far atmospheric warmth begins.
`_AtmosFarBlend`	0.45	0 – 1	Blend weight of the far atmospheric haze.

⚠ Properties marked □ are part of the confirmed working configuration. Altering fog distances, sun/fill intensities, or AO strength without rebalancing the others will degrade visual cohesion.

06 Quality Presets & GPU Budget

Tessaractic ships with three GPU quality presets selectable in the Inspector or via script at runtime. Changing the preset triggers a full triangle buffer reallocation and chunk re-stream.

Preset	Max Triangles	Render Radius	Active Chunks	VRAM Floor	Target GPU Class
`Low` ★	15,000,000	6 chunks	~169 (13×13 grid)	4 GB	GTX 1060 / RX 580 era
`High`	20,000,000	8 chunks	~289 (17×17 grid)	6 GB	RTX 2060 / RX 5700 era
`Ultra`	30,000,000	12 chunks	~625 (25×25 grid)	10 GB	RTX 3080 / RX 6800 era

Note: the three presets produce no difference in surface render quality or visual fidelity. The difference is coverage — Low renders a tighter radius with fewer active chunks, Ultra extends the visible world further with a larger triangle budget. Choose based on available VRAM, not appearance.

> Low is the confirmed production default and the only preset validated against the 4 GB VRAM development machine. High and Ultra are stable but were tested on higher-VRAM hardware. Always verify on your target hardware before shipping.

Runtime Quality Switching

Switch preset from script at any time:

```
GetComponent().currentQuality = FranksEquationManager.Quality.High;
```

The system reallocates the triangle buffer and restarts chunk streaming automatically. Expect a brief GPU stall during reallocation on lower-end hardware.

07 Do-Not-Touch Configuration

The values below represent the confirmed stable configuration after full pipeline calibration. They are deeply interdependent: FBM density calibration, Marching Cubes surface extraction, chunk treadmill timing, and shader fog range all assume these exact values. Changing any single value without understanding the cascade effects will likely produce visual artefacts, buffer errors, or incorrect terrain output.

Parameter	Value	Location	Why it must not change
`isoLevel`	0.0	FranksEquationManager	FBM convention. Non-zero values cause incorrect surface extraction.
`renderRadius (Low)`	6	Auto-set by Quality preset	Treadmill math and fog tuning are calibrated to radius 6 on Low.
`actualChunkSize`	32	Internal constant	Grid dimension in voxels. Must stay 32 for the 96-byte Triangle struct memory layout.
`maxTriangles (Low)`	15,000,000	Auto-set by Quality preset	GPU triangle output buffer size. Reducing causes pop-in; increasing risks OOM on 4 GB.
`blobStrength`	0.0 (Inspector)	FranksEquationManager	Set to 0 in the Inspector for stable terrain. Non-zero values are experimental in v1.

Internal Pipeline Constraints

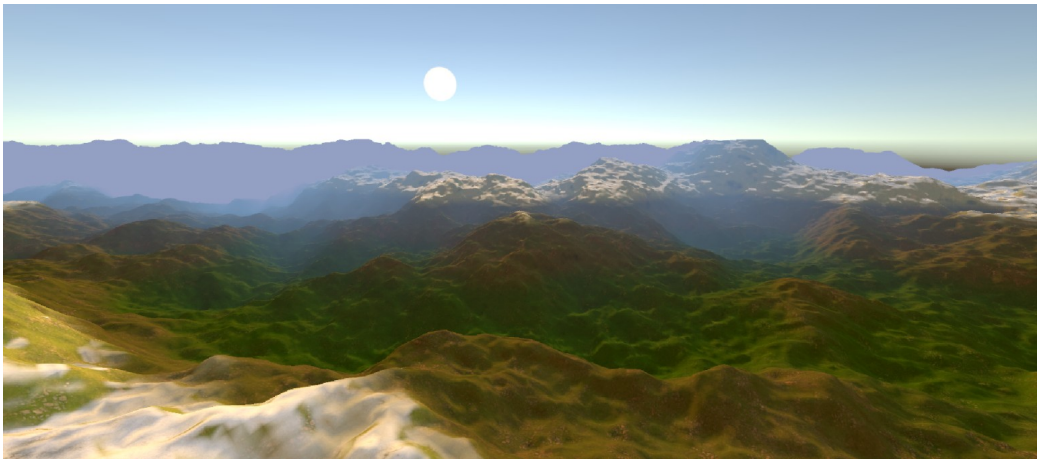
Parameter	Value	Location	Why it must not change
`_bw (SplatCubes)`	1	Franksequation.compute	Internal blob weight constant in the compute shader. Do not edit.
`cross filter threshold`	0.00001	MarchingCubes.compute	Degenerate triangle filter epsilon. Changing removes valid triangles or passes garbage geometry.
`maxEdge`	voxelSize × 6.0	MarchingCubes.compute	Maximum triangle edge length filter. Prevents stretched cross-chunk artefacts.
`bigShift threshold`	renderRadius × 4 + 1	FranksEquationManager	Treadmill shift trigger. Incorrect value causes double-shifts or missed chunk updates.
`TREADMILL_COOLDOWN_SEC`	0.8f	FranksEquationManager	Minimum seconds between treadmill shifts. Lower values cause GPU dispatch overlap.
`Fog Linear Start`	80	Unity Environment Lighting	Must match _FogStart in shader. Mismatch produces a hard visible fog seam.
`Fog Linear End`	160	Unity Environment Lighting	Must match _FogEnd in shader. Mismatch produces incorrect depth fade.
`Cull Off`	—	ProceduralTerrain.shader	Both front and back faces rendered. Required for cave interiors and overhangs.
`sunDiff / filDiff`	saturate(dot(...))	ProceduralTerrain.shader	Sun and fill diffuse use saturate variants. Do not change to max or abs.

08 Detailed Configuration Guide

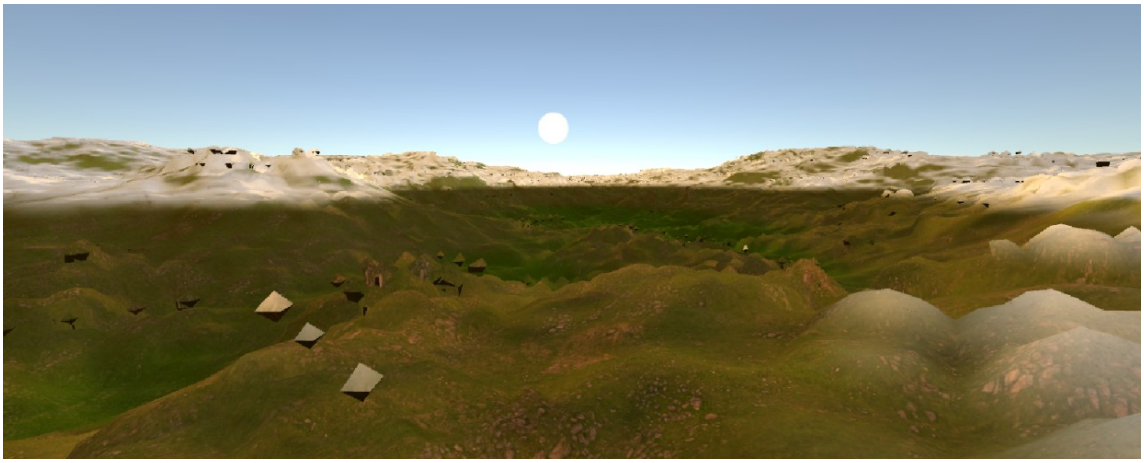
Adjusting Terrain Shape

The primary levers for terrain appearance are terrainBias, terrainScale, terrainHeight, and terrainOctaves. Common recipes:

- Broader sweeping mountains: Decrease terrainScale (0.005–0.008), increase terrainHeight (60–100).
- Tight rugged highlands: Increase terrainScale (0.015–0.03), keep terrainHeight at 40.
- Flat plains with gentle hills: Decrease terrainBias (0.4–0.5), decrease terrainHeight (15–25).
- Rocky spires and formations: Increase blobStrength (0.2–0.4), carveRate (0.5–0.6), keep blobRadiusScale at 1.5.
- Finer surface detail noise: Increase terrainOctaves to 6–8. Note GPU cost increases proportionally.



terrainScale at 0.015 — reference value showing terrain frequency and landscape spread at the shipped default.



blobStrength above 0 — surface hole artifacts visible. Shipped default is 0. Do not increase in v1.

Biome Tuning

To shift biome zones vertically relative to terrain:

- Raise _BandGrassTop to extend the grass zone higher up the slopes.
- Lower _BandCaveTop to expose more cave-colour rock near the base.
- Increase _BlendHalf (e.g. to 4.0) for softer biome transitions.
- Adjust _SlopeRockBelow and _SlopeFlatAbove to control cliff material steepness.

Lighting Adjustment

The shader uses a fully custom Blinn-Phong light model independent of Unity's PBR pipeline. To relight the scene:

- Rotate the scene Directional Light to change sun angle — the shader reads _WorldSpaceLightPos0 automatically.
- Change _SunColor for time-of-day tinting (warm noon, cool overcast).
- Increase _FillIntensity to brighten shadowed faces (reduces contrast).
- Adjust _RimIntensity for silhouette glow strength.

Texture Setup

Tessaractic expects physically-based texture sets in the following formats:

Map Type	Format	Notes
`Albedo`	RGB, sRGB	Standard diffuse colour map.
`Normal Map`	RGB, Linear, nor_gl	OpenGL convention. Set texture type to Normal Map in Unity import settings.
`ARM Map`	RGB, Linear	R = Ambient Occlusion, G = Roughness, B = Metalness. Pack manually or use Materialize.
`Displacement`	Grayscale, Linear	POM height. Mid-grey (0.5) = neutral surface.
`Translucency`	Grayscale, Linear	Snow only. White = fully translucent backlit area.

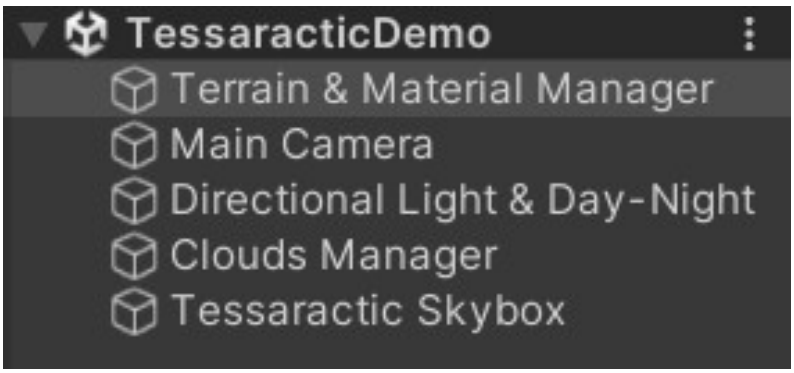
Free PBR textures are included for Ground & Snow. For more high-quality materials, you can check PolyHaven (polyhaven.com)

09 Demo Scene Guide

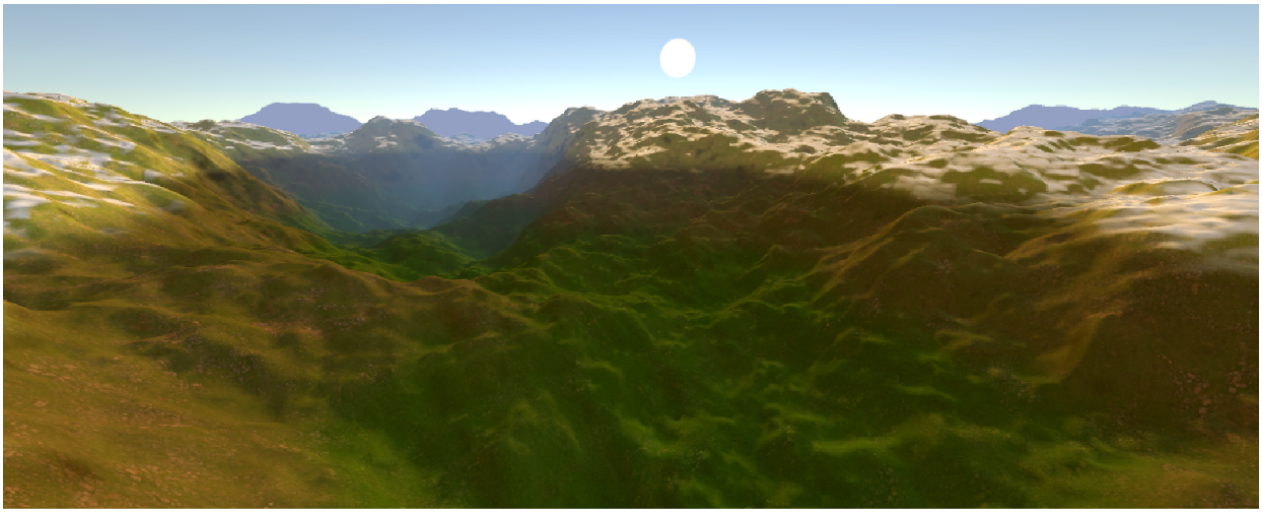
The included demo scene is a clean, distraction-free showcase of the core terrain system and its integrated high-performance environmental systems. It contains no decorative props, particle effects, or post-processing beyond what the native terrain shader and new procedural environmental scripts provide.

Scene Contents

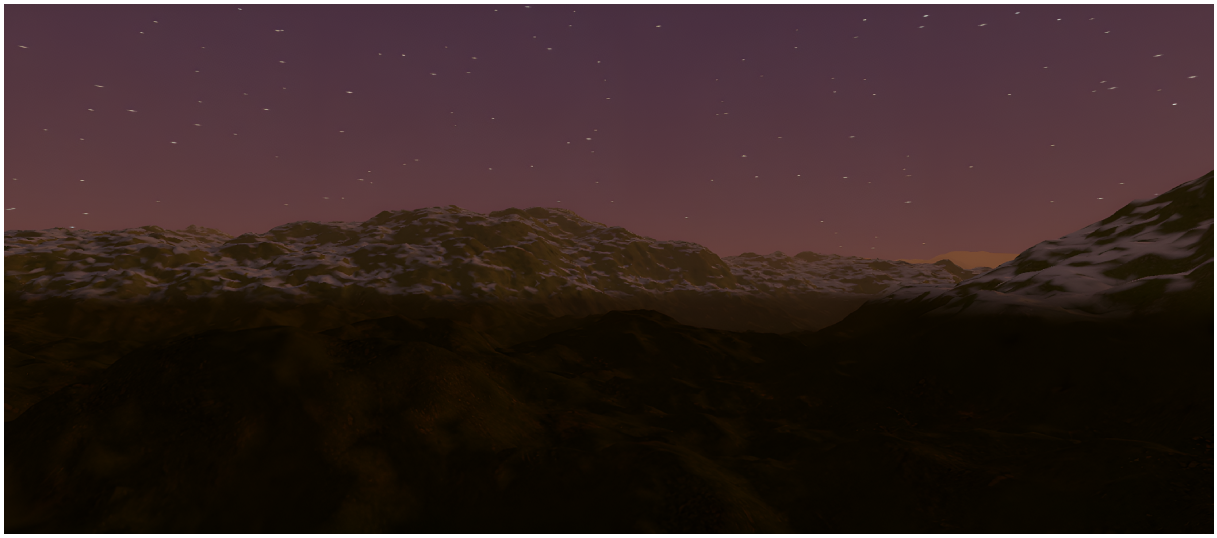
- Terrain & Material Manager: Empty GameObject with FranksEquationManager and the confirmed working Inspector configuration pre-set.
- Main Camera: Fly camera using TessaracticFlyCamera.cs. WASD to move, mouse to aim, Shift to sprint. Cursor locks on Play; press Escape to release.
- Directional Light: Single sun light angled to catch slopes and cliff faces. Matched to _SunColor and _SunIntensity shader defaults. Features the TessaracticDayNightCycle.cs script to drive the dynamic day-night cycle, which interpolates 14 distinct lighting and shading parameters in the terrain shader from sunrise to midnight.
- WorldMat: Pre-configured material on Tessaractic/ProceduralTerrain with all PBR slots assigned to the included sample textures. It's inside 'Terrain & Material Manager', scroll down.
- CloudManager: Empty GameObject containing the TessaracticClouds.cs script and pre-assigned Cloud_Material (using ProceduralClouds.shader) to render lightweight, dynamic Fractal Brownian Motion cloud cover.
- SkyboxManager: Empty GameObject containing the TessaracticSkybox.cs script and pre-assigned Skybox_Material (using ProceduralSkybox.shader) to drive the procedural skybox gradient, twilight star field, solar disc, and moon phases.



Demo scene Hierarchy — Directional Light (with Day-Night Cycle), Terrain & Material Manager, Main Camera, CloudManager, and SkyboxManager.



Demo scene — in-game terrain render with Tessaractic Day-Night Cycle, Procedural Clouds, and Procedural Skybox, alongside rolling hills, POM slope detail, and height-band biomes.



Demo scene — night-time terrain render demonstrating the Procedural Skybox with a twilight horizon glow and twinkling star fields softly illuminating the snowy mountain peaks.

What the Scene Demonstrates

- Rolling hills and open terrain from the calibrated FBM pipeline.
- POM depth on inclined surfaces, visible at low angles across rock faces.
- Height-band biome transitions: grass through stone through snow.
- Atmospheric fog depth pulling distant geometry into the skybox naturally.
- Infinite chunk treadmill: walk in any direction and terrain generates ahead.
- Blinn-Phong rim lighting creating silhouette depth at the horizon.
- Tessaractic Day-Night Cycle: Fully dynamic time-of-day progression synchronizing 14 terrain material parameters (sun direction/intensity/color, ambient/fill/fog colors, specular settings, and dynamic

albedo boosts).

- Procedural Cloud System: Lightweight, textureless GPU FBM infinite cloud plane with continuous wind speed animation, camera tracking, and sun rim-light glow.
- Procedural Skybox: 3-way color gradient sky shifts, twinkling star fields with atmospheric extinction, volumetric solar disc, and crater-detailed moon with phase shading.

> 💡 Asset Store screenshots should be captured from this scene without modifications. The confirmed configuration produces the most visually representative output for the store listing.

10 🎁 Bonus Experimental Files

Tessaractic ships with six additional experimental files as a free bonus, separate from the seven core asset files. These implement a GPU-instanced vegetation and surface-detail system for placing trees, grass, moss, rocks, and similar objects procedurally on the terrain surface.

These files are approximately 80% production-ready: functional, but likely requiring project-specific debugging before full integration. They are provided as open source, as-is, as a significant time-saving starting point.

File	Type	Role
`ForestDebugger.cs`	C# MonoBehaviour	Debug visualiser for vegetation instance placement. Draws Gizmos showing where instances will spawn relative to terrain chunks.
`ForestCulling.compute`	HLSL Compute Shader	GPU frustum and distance culling for indirect-draw vegetation instances. Discards out-of-view instances before the draw call to save GPU bandwidth.
`ForestInstances.compute`	HLSL Compute Shader	Procedural instance placement kernel. Scatters objects across terrain chunks based on slope angle and height band rules.
`InstancedIndirectLit.shader`	CG Shader (BiRP & URP)	GPU-instanced indirect draw shader matching the terrain lighting model. Supports Blinn-Phong sun / fill / rim lighting from ProceduralTerrain.shader.
`terrain.cs`	C# MonoBehaviour	Terrain surface query helper. Provides world-space height and normal sampling for placing instances flush with the terrain surface.
`TerrainHeightBaker.cs`	C# MonoBehaviour / Editor	Editor utility for baking a heightmap representation of a terrain region for offline LOD or collision generation workflows.

> ⚠ Bonus files are not covered by core asset support. They are provided to save significant development time, not as finished drop-in features. Expect debugging effort when integrating them into your project.

11 Known Limitations & V2 Roadmap

The following features are intentionally absent from Version 1. They are documented here with full technical explanations so the limitations are understood as deliberate architectural boundaries, not overlooked features.

Shadow Receive from External Objects

To achieve maximum GPU throughput and infinite terrain generation on low-end hardware, the custom pipeline focuses 100% of its computing power on terrain geometry and terrain self-shadowing. It does not allocate draw calls to casting or receiving external object shadows. This is a deliberate performance trade-off, not an oversight.

Technical context. The terrain mesh is generated and submitted to the GPU via Graphics.DrawProceduralNow inside the OnRenderObject callback. Unity's shadow collection pipeline runs before OnRenderObject in the frame, meaning the terrain geometry is not yet available when the shadow map is built. Rather than introducing a secondary pass that would consume GPU budget and compromise the infinite-streaming architecture, the pipeline prioritises raw terrain throughput — ensuring stable frame rates even on 4 GB VRAM hardware.

> Terrain self-shadowing is fully functional. The pipeline intentionally reserves all GPU resources for terrain generation and self-shadow computation, delivering consistent performance across all supported hardware tiers.

Per-Seed Terrain Variation

The seed string field is functional: it is read at runtime and the binding infrastructure is in place. However, changing the seed does not produce a meaningfully different terrain shape in Version 1.

Technical explanation. The FBM density pipeline is numerically calibrated against exact values of terrainScale (0.01) and terrainHeight (40). The noise octave weights, vertical density gradient, isoLevel convention, and Frank's Equation blob integration all assume this specific numeric range. A seeding system that produces qualitatively distinct terrain shapes — different mountain profiles, valley widths, plateau distributions — requires the FBM pipeline to be redesigned from the ground up around seed-parameterised noise functions rather than calibrated constants. This cannot be added as a patch and is a V2 redesign target.

> The seed binding system is in place. The limitation is that FBM calibration must be redesigned to be seed-driven before meaningful variation is achievable.

Feature Status Summary

Feature	V1 Status	Root Cause	V2 Target
`Shadow receive (external objects)`	By design	GPU budget reserved for terrain throughput and self-shadowing	Under consideration
`Per-seed terrain variation`	Not available	FBM calibrated to fixed terrainScale / terrainHeight constants	Yes — density pipeline redesign required
`URP support`	Working	Auto-configured via automatic shader header installation on first import	—
`HDRP support`	Not available	HDRP requires a full shader rewrite to HLSL	Under consideration
`Terrain self-shadowing`	Working	—	—
`Infinite chunk streaming`	Working	—	—
`Runtime quality switching`	Working	—	—
`Deterministic seed binding`	Working	—	—

⚠ All of the 6 core scripts ship as fully open source. If you are a senior developer or an experienced engineer who requires external object shadow casting, the source code is clean, well-documented, and fully ready for you to extend.

12 Revision History

Version	Date	Summary
`1.0`	May 2026	Initial release. Full GPU terrain pipeline: FBM + Frank's Equation octree subdivision, GPU Marching Cubes with 96-byte Triangle structs and per-vertex smooth normals, triplanar POM shader, Blinn-Phong + rim + fill lighting, height-band biome blending, wetness simulation, atmospheric fog, and infinite chunk treadmill. Three quality presets (Low / High / Ultra). Seven open-source core files. Six bonus experimental vegetation files. Known limitations fully documented.

13 ☼ Tessaractic Environment Features: Documentation Guide

This section details the technical features, inner workings, and inspector controls for the high-performance environmental systems integrated with Tessaractic: the Tessaractic Day-Night Cycle, Procedural Clouds, and the Procedural Skybox.

13.1 Tessaractic Day-Night Cycle (TessaracticDayNightCycle.cs)

The Day-Night Cycle acts as the core lighting controller for the entire environment. It dynamically rotates the scene's primary directional light to represent the time of day, and smoothly interpolates 14 distinct lighting and shading properties in the terrain shader (ProceduralTerrain.shader) to guarantee perfect visual consistency from sunrise to midnight.

Key Features

- Dual Operating Modes: Easily toggle between Automatic (real-time passing driven by a configurable day length) and Manual (fixed time controlled directly by a 0–24 hour slider).
- Multi-Property Synchronization: Automatically updates the following terrain material parameters:
  - Sun color, intensity, and direction vectors.
  - Ambient, fill, and fog colors.
  - Specular colors, specular power, and rim light thresholds.
  - Dynamic albedo boosts (_AlbedoBoost and _SnowAlbedoBoost) to keep snow and peaks visible and atmospheric under weak twilight/midnight light.

13.2 Procedural Cloud System (ProceduralClouds.shader & TessaracticClouds.cs)

This lightweight, textureless system generates a dynamic, mathematically driven cloud layer on an infinite horizontal plane. It synchronizes with the active Day-Night Cycle to inherit natural, atmospheric color shifts.

Key Features

- GPU Fractal Brownian Motion (FBM): Uses a highly optimized 4-octave 3D noise generation algorithm to create realistic, infinite cloud formations with zero texture overhead.
- Continuous Wind Animation: Animates clouds smoothly using a configurable speed and direction vector.
- Camera Tracking: Automatically snaps the XZ coordinates of the cloud mesh to follow the camera, providing seamless sky coverage as the player explores the infinite world.

- Horizon Fade & Alpha Test: Incorporates a soft horizontal falloff to fade out clouds cleanly before they hit the horizon sky, using optimized alpha-testing (clip()) to ensure excellent fill-rate performance.
- Interactive Sun Rim Tinting: Features custom lighting math that makes the cloud edges realistically 'glow' with warm light when passing directly in front of the sun.

13.3 Procedural Skybox (ProceduralSkybox.shader & TessaracticSkybox.cs)

A fully procedurally generated skybox featuring a dynamic sky gradient, a soft solar disc, an automated twilight/midnight star field, and a realistic cratered moon with phase lighting.

Key Features

- Dynamic Sky Gradient: Blends a 3-way color system (Top, Horizon, and Ground) that shifts beautifully between daylight, vibrant golden sunsets, and deep midnight.
- Procedural Twinkling Star Field: Generates a sparse, realistic star field across two independent depth layers. Includes built-in atmospheric extinction (stars naturally fade out near the horizon) and dynamic twinkle speeds.
- Volumetric Solar Disc: Renders a clean solar disc with a soft, warm atmospheric halo.
- Crater-Detailed Moon & Phase Lighting: Generates a highly stylized moon with crater noise details. Moon phases automatically shift (lighting up or casting shadows across the moon face) depending on the sun's relative angle.
- Auto-Opposite Alignment: The moon automatically tracks opposite the sun light's forward vector, rising realistically as the sun sets.
- Inspector Slider Overrides: Features fine-grained slider controls directly on the C# script to easily customize Sun, Moon, and Star sizes on the fly.

13.4 Skybox Inspector Customization Reference

You can customize the proportions of your sky elements directly from the TessaracticSkybox component inspector:

Setting	Range	Default Value	Description
`Sun Disc Size`	0.001 - 0.05	0.005	Adjusts the radius of the hard solar disc (smaller = realistic).
`Sun Halo Size`	0.01 - 0.30	0.08	Sets the radius of the soft glow surrounding the sun.
`Star Density`	30 - 800	80	Sets the grid subdivision count (lower = sparser, more natural stars).
`Star Brightness`	0.0 - 3.0	0.8	Controls how intensely the stars twinkle and shine at night.

Setting	Range	Default Value	Description
`Moon Disc Size`	0.001 - 0.05	0.003	Adjusts the physical size of the moon disc.
`Moon Glow Size`	0.005 - 0.10	0.02	Extent of the soft glowing halo surrounding the moon.
`Moon Glow Intensity`	0.0 - 2.0	0.3	Controls how bright the moon glow/halo appears against the night sky.

14 🔧 Troubleshooting & FAQ

This section covers every issue and edge case encountered during development, with the confirmed cause and resolution for each. Always open the Unity Console first — Tessaractic prefixes every status message with [Tessaractic]. The startup chain logs kernel caching, buffer allocation, and world generation in sequence. The first red error in that chain identifies the exact failure point.

14.1 Setup Errors

These errors appear immediately on entering Play Mode and are always Inspector assignment issues. No code changes are needed to resolve them.

Compute Shader Slots Not Assigned

All three compute shader slots on the FranksEquationManager component must be filled before entering Play Mode. The three files are FranksEquation.compute, DensityGenerator.compute, and MarchingCubes.compute. If any one slot is empty, the following error chain fires in the Console:

```
[Tessaractic] Kernel cache FAILED: The variable equationShader of FranksEquationManager has not been assigned.
You probably need to assign the equationShader variable of the FranksEquationManager script in the inspector.
[Tessaractic] Kernels failed — skipping buffer allocation and world gen.
[Tessaractic] OnRenderObject: triangleBuffer null. GenerateTessaracticWorld() may have failed.
```

> 💡 The first error line changes slightly depending on which slot is empty (equationShader, densityShader, or marchingCubesShader), but the fix is always the same: assign all three compute shaders in the Inspector. Kernel names are case-sensitive — Subdivide, ClearGrid, SplatDensity, SplatCubes, ApplyTerrainBias, MarchingCubes.

WorldMat Material Not Assigned

If the WorldMat material is not assigned on the TessaracticWorld component, the terrain pipeline initialises but has no surface to render to. The following errors appear:

```
[Tessaractic] StartGeneratingWorld: triangleBuffer is null — GenerateTessaracticWorld failed.
[Tessaractic] OnRenderObject: terrainMaterial null.
```

> ℹ️ Drag the WorldMat material from the Materials folder into the Terrain Material slot on the TessaracticWorld component in the Inspector. This is the single material that drives the entire surface shader. Without it, nothing renders.

14.2 What Happens If You Change Locked Parameters

Certain parameters are locked at their documented values for v1 because the entire pipeline is calibrated around them. The table below documents exactly what each change produces, so developers understand the consequences before experimenting.

Parameter	Locked Value	If Changed — Observed Result
`isoLevel`	0 (slider centre)	Decreasing (e.g. to −50): camera spawns below the terrain surface, overall size shrinks, consistent gaps appear, output is mostly snow. Increasing toward 45–47: significantly larger mountains with snow peaks. Grid lines appear only inside impact craters in this range. Best results near 45–47, with minor peak stretching at extremes. Range 1–50 is usable. The zero-crossing convention is the FBM calibration baseline — 0 is the locked shipping value.
`renderRadius`	6	Increasing to 12 with the same triangle budget: terrain is cut in a hexagonal silhouette when viewed from above. Fix: switch to High preset to raise the triangle buffer alongside the radius. Decreasing to 3: terrain footprint shrinks. Infinite streaming functions only in the camera's direction of travel — the world loses its 360° infinite nature and expands only forward.

🛑 CRITICAL — actualChunkSize (32), blobStrength (0), and the bigShift formula (renderRadius × 4 + 1) are all derived from voxelSize and renderRadius. Changing any one value in isolation breaks the pipeline interdependencies. These are DO NOT TOUCH for v1.

14.3 Common Issues — Quick Reference

Symptom	Cause	Fix
`Zero triangles / black scene on Play`	terrainMaterial not assigned, triangle buffer failed silently, kernels not cached before world gen fired, or isoLevel changed from 0.	Check Console for the [Tessaractic] startup chain. Assign WorldMat. Confirm all three compute shaders are in Inspector slots. Restore isoLevel to 0 and press R to regenerate.
`Buffer allocation failed / out of memory`	GPU rejected the triangle buffer. Occurs on cards below 4 GB VRAM on High or Ultra preset.	Switch to Low preset (15 M triangles). The allocator retries at half capacity automatically and logs a warning.
`Kernel not found on Play`	Compute shader not assigned in Inspector, or shader file was renamed.	Assign all three compute shaders. Kernel names are case-sensitive: Subdivide, ClearGrid, SplatDensity, SplatCubes, ApplyTerrainBias, MarchingCubes.
`Chunk seam lines / grid visible at borders`	voxelSize or actualChunkSize was modified, or the _bw = 1 value inside DensityGenerator SplatCubes was removed.	Restore voxelSize to 0.5. Never remove _bw = 1 from SplatCubes — it exists specifically to suppress seam geometry at chunk borders.
`Terrain spikes / exploding geometry`	blobStrength above 0, or isoLevel changed from 0.	Set blobStrength back to 0. Restore isoLevel to 0. Press R to regenerate.
`Sprint stutter / 8-second freeze while moving fast`	Triangle buffer hitting capacity during sustained movement.	Low preset (15 M triangles) is the confirmed safe value for 4 GB VRAM. Do not reduce renderRadius to compensate — a smaller radius triggers more frequent full regenerations and makes the stutter worse.
`Terrain disappears after pressing R`	material.SetBuffer("rawTriangles") loses its reference after SetCounterValue(0) following a code change.	Ensure RecreateTriangleBuffer() calls SetBuffer immediately after allocation. This is handled internally in the shipped build.
`Fog not visible despite correct settings`	Unity's Environment fog toggle can be off even when fog is correctly configured in RenderSettings via code. Confirmed issue from development.	Window → Rendering → Lighting → Environment — confirm the Fog checkbox is enabled there.
`Tiny surface holes at chunk borders`	Residual pixel-level stitch at chunk edges when blobStrength is at 0. Known accepted artifact inherent to the v1 border exclusion logic.	No action required. This artifact protects overall terrain quality and is documented as won't-fix for v1.
`Albedo too dark`	AlbedoBoost set too low for the current lighting environment.	Slide AlbedoBoost up toward 5. Snow brightness is adjusted independently via SnowAlbedoBoost (default 1.5).

Symptom	Cause	Fix
`Lighting looks mismatched / two-tone`	The scene Directional Light is not matched to the shader's _SunColor and _SunIntensity values.	Match the Directional Light colour, intensity, and shadow type (Soft) in the Hierarchy exactly to _SunColor and _SunIntensity on WorldMat. A mismatch produces a two-source conflict that cannot be corrected in post.
`Terrain not visible in Scene View`	OnRenderObject fires only during Game View rendering. Terrain is not a standard Renderer and is invisible to the Scene View camera.	Expected behaviour. Use Game View for all terrain work.
`Quality preset change causes blank world briefly`	Changing Quality at runtime disposes and reallocates the triangle buffer and re-streams all chunks from scratch.	Expected behaviour. The world fills in from the closest chunk outward over a few seconds. Not a bug.
`HDRP selected in Project Settings`	The shader is a CGPROGRAM with dual Built-in RP and URP SubShader blocks. HDRP is not supported and throws errors.	Keep the project on Built-in RP or Universal RP. HDRP requires a full shader rewrite and is not supported in v1.
`Accidental edit to MarchingCubes.cginc causes errors`	This file is a sensitive shared include. A single mis-click can break compilation silently with no obvious line number.	Treat MarchingCubes.cginc as read-only. Restore from the original package if any error appears after touching this file.

14.4 Known v1 Limitations

The following are architectural constraints of the v1 pipeline. They are not bugs. Each was thoroughly investigated during development. They are documented here so that developers do not spend time re-attempting approaches that have already been exhausted.

Shadow Receive — Deferred to v2

Tessaractic terrain does not receive shadows from scene objects in v1. The root cause is architectural: terrain renders via OnRenderObject at step 7 of Unity's Built-in pipeline, after screen-space shadow collection completes at step 3. Terrain depth never reaches _CameraDepthTexture, so no shadow map sampling is possible regardless of the approach taken. Nine separate solutions were attempted and exhausted during development:

#	Approach Attempted	Error / Result
1	URP HLSL includes inside CGPROGRAM	Couldn't open include file 'Lighting.hlsl'. URP ShaderLibrary is not accessible from a CGPROGRAM block.
2	AutoLight.cginc macros: SHADOW_COORDS, TRANSFER_SHADOW, SHADOW_ATTENUATION	Compiled clean. No shadow visible. LightMode=ForwardBase tag missing. TRANSFER_SHADOW uses v.vertex which does not exist in the SV_VertexID shader.
3	Full CGPROGRAM → HLSLPROGRAM conversion	Couldn't open include file 'Core.hlsl'. Built-in pipeline does not have URP ShaderLibrary. Reverted immediately.
4	Manual URP uniform declarations (_MainLightShadowmapTexture etc.)	Compiled clean. No shadow. URP uniforms have no effect on Built-in pipeline.
5	Screen-space shadows via _ScreenSpaceShadowmapTexture + screenPos	invalid subscript 'screenPos'. screenPos not yet in struct at point of frag access. Also wrong pipeline.
6	UnitySampleShadowmap() via UnityShadowLibrary.cginc	undeclared identifier 'UnitySampleShadowmap'. Function not resolving despite explicit include.
7	tex2D(_ShadowMapTexture) via ComputeScreenPos	invalid subscript 'screenPos'. screenPos field missing from v2f struct at that point.
8	tex2D(_ShadowMapTexture) via existing shadowCoord.xy/w	redefinition of 'shadow'. Leftover shadow variable from a previous attempt still present in frag.
9	UNITY_LIGHT_ATTENUATION + float4 _ShadowCoord : TEXCOORD2 + CommandBuffer depth prepass at AfterDepthTexture + Shader.EnableKeyword(SHADOWS_SCREEN)	4–5 FPS. Sunlight permanently off when looking −X. No shadow. EnableKeyword is global — corrupted keyword state for all scene materials. CommandBuffer ran a full geometry pass every frame. Root cause confirmed: OnRenderObject runs after screen-space shadow collection, so terrain depth was never present in _CameraDepthTexture.

ⓘ Shadow casting from terrain onto other scene objects works correctly via the ShadowCaster pass. Only shadow receiving is affected by this limitation. Shadow receive is a planned v2 feature.

Per-Seed Terrain Variation — Deferred to v2

The seed field drives Frank's Equation octree variation only. FBM terrain shape — hills, height distribution, and scale — is intentionally identical across seeds. The density generation pipeline is calibrated to exact terrainScale and terrainHeight values and cannot be safely varied per-seed without a ground-up redesign of that stage.

⚠ Since the asset ships with open-source scripts (only FranksEquation.compute is compiled to .dll), some developers may attempt seed-based terrain variation by modifying the density shader SetFloat calls inside GenerateMMOChunk. The two lines that must remain unchanged are:
densityShader.SetFloat("_Scale", terrainScale);
densityShader.SetFloat("_Height", terrainHeight);
Replacing these with seed-derived values was tested and produces broken terrain: spikes, stretched PBR textures, grid lines, and large gaps. Seed-driven FBM variation is a v2 feature.

Terrain Not Visible in Scene View — By Design

The terrain will never appear in the Scene View camera. OnRenderObject is a Game View callback and does not fire for the Scene View camera pass. This is not a rendering error — it is an inherent property of how the procedural draw call is issued. All terrain work, tuning, and visual review must be done in Game View.

14.5 Quick Tips

Practical notes gathered directly from the development process.

Tip	Detail
`Press R to regenerate`	Pressing R during Play Mode disposes the current buffer, reallocates, and re-streams the entire world from scratch. Use it after any parameter change to see the result cleanly without restarting Play Mode.
`Match Directional Light to shader`	The scene Directional Light colour, intensity, and shadow type (Soft) must match _SunColor and _SunIntensity on WorldMat exactly. A mismatch creates a two-source lighting conflict visible as an unnatural split between sunlit and shaded faces.
`Fog not appearing`	Window → Rendering → Lighting → Environment — confirm the Fog checkbox is ticked. Unity's Environment toggle can be disabled even when fog is set correctly in RenderSettings via code. This was a confirmed issue during development.
`Albedo too dark`	Slide AlbedoBoost up toward 5. Snow brightness is controlled separately by SnowAlbedoBoost (shipped default 1.5). The two sliders are independent.

Tip	Detail
`isoLevel experimentation`	The locked value is 0. If experimenting, values in the 40–47 range produce very large mountain ranges with snow peaks. Grid lines appear only inside impact craters at these values. Do not go below 0 — the camera spawns below the terrain surface.
`Do not edit MarchingCubes.cginc`	This is a sensitive shared include file. A single accidental edit can silently break compilation with no clear line number reported. Treat it as read-only and restore from the original package if any error appears after it was open.
`Built-in RP or URP only`	Tessaractic supports both Built-in RP and Universal RP via automatic shader header configuration on first import. HDRP is not supported in v1. Switching to HDRP throws shader errors immediately on entering Play Mode.

For support, contact Tessaract Lab via the Unity Asset Store publisher page.