Planetary Terrain

Customizable Procedural Planets for Unity

Planetary Terrain allows you to quickly design your own procedural planets right in the Unity editor. Use a combination of procedural noise functions and terrain painting to craft exactly the kind of planets your project needs, anything from tiny planets to massive worlds.

Features:

- Procedurally generated planets with dynamic level of detail.
- Terrain painting: paint features directly on to planet surface!
- Powerful node-based editor to control the procedural generation.
- Runtime generation for minimal application size.
- Easily save planets to meshes and prefabs.
- Full source code included.
- Support for both Unity Free and Pro!
- Atmosphere and water prefabs

Getting Started

Workflow

To create a planet there are few steps to cover:

1. **Create or select a Terrain module.** Terrain modules define the procedural algorhitm that defines the planets terrain.
2. **Set mesh generation parameters**. What is the required detail level and polygon/draw call budget? Should dynamic LOD be used? These are questions best answered by trying out different settings in the planet inspector.
3. **Define surface textures and colors.** These are found in the SurfaceColor script attached to the planet gameobject by default. You may also replace the default shader alltogether.
4. **Paint heights.** Painting settings and brushes are found in the Planet inspector Paint-tab. Height brushes modify the terrain height but still preserve the procedural details.
5. **Place objects.** There is a script called PlacementHelper that can be added to the planet gameobject. This script contains the basic functionality to paint objects to the surface and automatically place objects during generation. There can be multiple PlacementHelper-scripts at the same time.
6. **Create a build.** For run-time generation to work, the terrain module files should be located inside a Resources folder. For minimal application size clear the surfaces from the planet and enable ”Generate on start”.
7. **Save meshes and prefabs.** The export button is found in the planet inspector. It will create a prefab containing everything within the planet gameobject hierarchy and save meshes and materials to files.

Creating a Planet

1. To create a new planet go to *GameObject → Create Other → PlanetaryTerrain → Planet* in the Unity editor menu. This will add a new blank Planet object to the scene.
2. Set the ”*Terrain module*” in the Inspector window. You can find ready made modules in ”PlanetaryTerrain/Resources/Modules/”-folder. Alternatively you may create a new module by clicking the ”Edit”-button
3. Set the Radius parameter in the Inspector to match your desired scale.
4. Press ”*Generate Now*”.
5. See ”Planet settings” chapter for explanation of the other settings.

Creating a Terrain Module

1. Open the Module Editor in Unity editor menu *Window → Planet Module Editor* OR in the Planet inspector by pressing ″ *Edit*″ -button next to the module file.
2. Click ″ *New*″ in the top left corner of the window.
3. This brings up the default graph which is very simple with only one FBM node and the output node.
4. Click ″ Parameters″ in the FBM node to change the settings.
5. Use ″ New Generator″, ″ New Operator″ and ″ New Macro″ to add nodes.
6. Connect nodes by clicking the small connectors in the side of the nodes. Inputs are located on the left side and outputs on the right side.
7. Connect the last node to the Output node.
8. Save the graph using the ″ Save″ button in the top left corner.
9. Now you may load the graph in the Planet inspector as instructed above in ″ *Creating a Planet*″ !

Editing planet surface (painting)

1. Make sure the planet has colliders enabled (in planet inspectors Generate-tab under Colliders click show and then tick ″ Level 0: Generate colliders″).
2. Select Paint-tab in planet inspector, you should now see a brush circle when mouse hovers over the planet in scene view. Left click on the planet will apply the currently selected brush,
3. Edit brush settings and painting resolution in the inspector.
4. The planet will maintain the changes even if surfaces are deleted, the changes will be applied to any surfaces generated later. To clear the changes click ″ Clear all″.

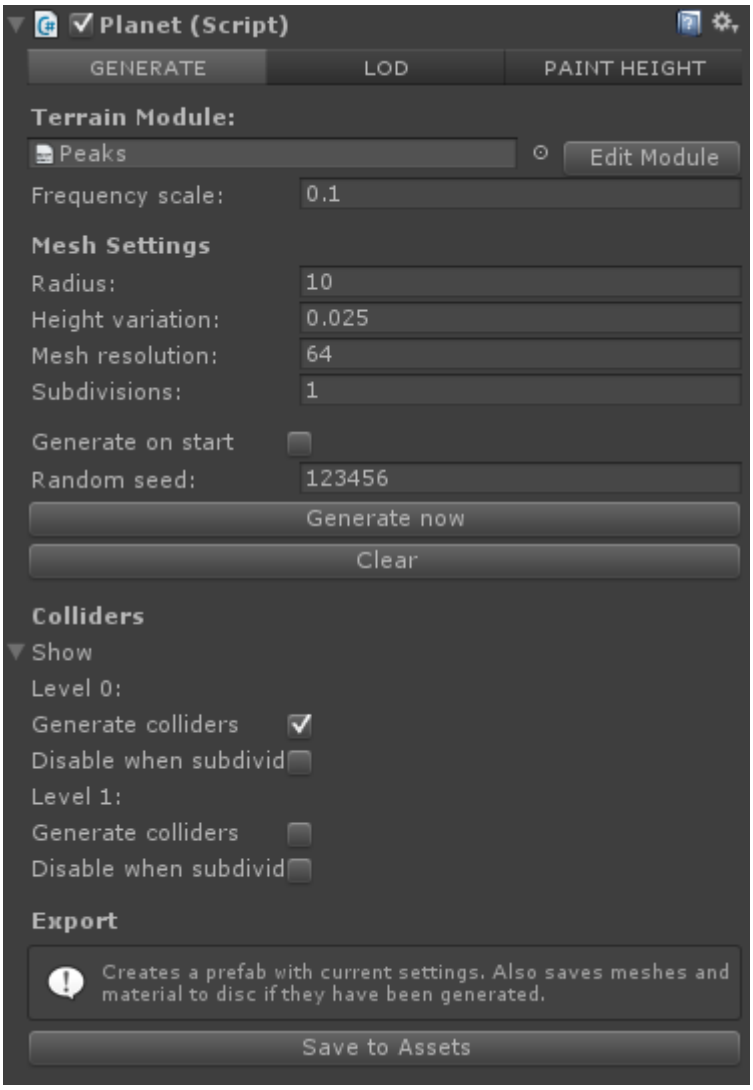
Saving Planet Meshes

1. Click ″ Save to Assets″ -button in the Planet inspector.
2. Select folder and filename for the saved assets.
3. The script will create a prefab of the whole planet and save each individual mesh in the folder.

Planet Settings

Explanations for the Planet inspector settings:

Generate



Terrain Module:

- **Terrain Module** is a file containing the instructions for generating the planet surface.
    - NOTE: For run-time generation the file should be located inside " *Resources"* folder in order to be included in the build.
- **Frequency scale** is a multiplier that affects all the generators in the module file. This override enables you to easily adjust the terrain feature scale without changes to the module file.

Mesh Settings:

- **Radius** sets the planets scale in units.
- **Height variation** defines the scale of the terrain features in relation to the radius.
- **Mesh resolution** is the size of a single surface patch of the planet. For example 32 means 32x32 vertices. Recommended to be from 16 to 64 when using LOD. Maximum is 254x254=64516 vertices (Unitys limit is 65000 for single mesh).
- **Subdivisions** means how many times the 6 original patches are divided at the beginning. This forms the minimum complexity for the model before LOD.
    - For example subdivision value of 1 means there are 6 total surfaces in the base model (before LOD). Value of 2 means each of the 6 surfaces are divided in to 4 smaller surfaces bringing the total to 24.
- **Generate on start** determines if the Planet should be generated in the *Start*-function.
- **Randomize seeds** determines if the random seeds will be shuffled when the planet is generated. Only available if " Generate on start" is enabled.
- **Generate now** generates the Planet in editor with current settings.
- **Clear** destroys any currently existing surfaces from the planet.

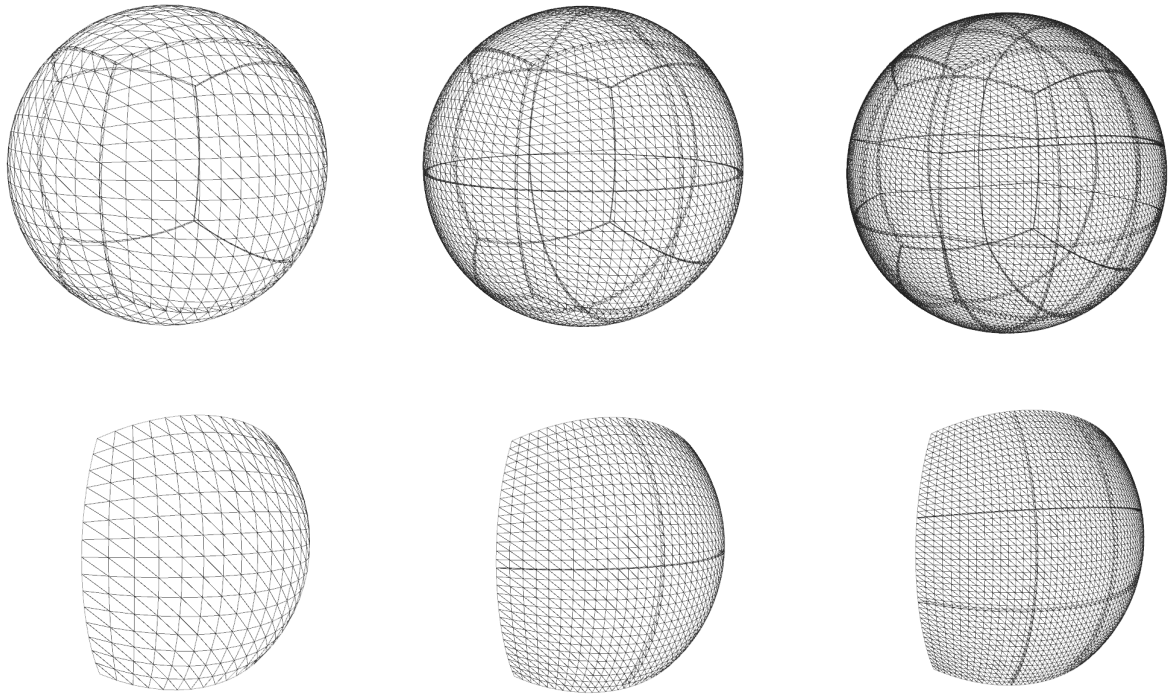


Image showing different subdivision levels on a full planet and a single surface. From left to right: subdivision 1, subdivision 2 and subdivision 3.

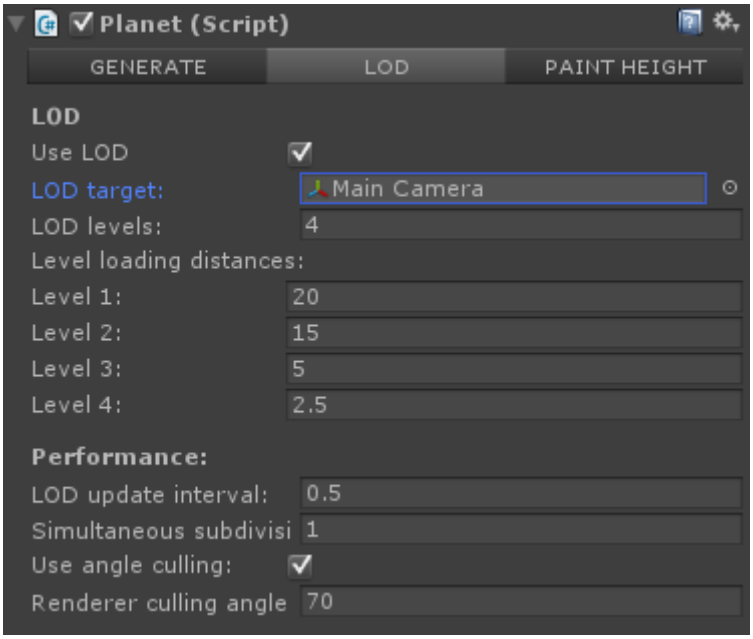
Colliders

- For each LOD level (set in the LOD tab) there are two options:
- **Generate colliders** defines if the surfaces should add a mesh collider.
    - *WARNING: Generating a lot of colliders slows performance.*
- **Disable when subdividing**, deactivates colliders in this level when higher level is generated.

Export

- **Save to Assets** saves the planet prefab, material and meshes to user defined folder.

LOD



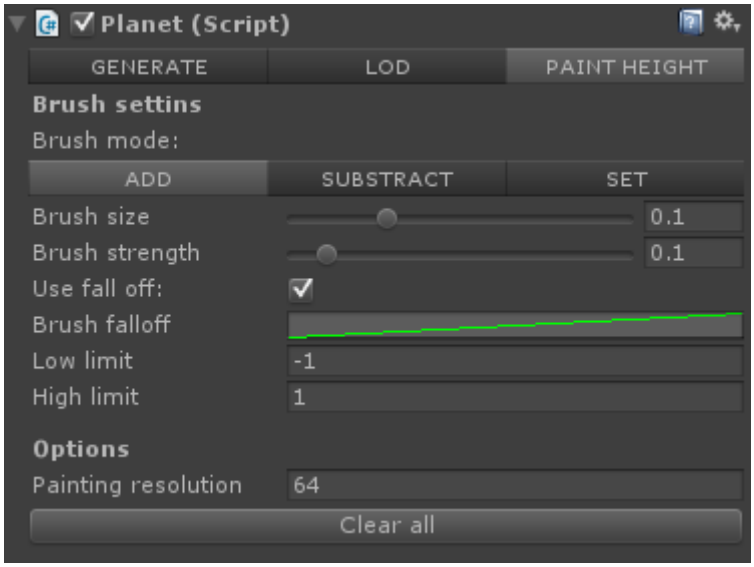
- **Use LOD**: enables / disables the dynamic level of detail system.
- **LOD target** is the transform that is used for the distances calculations when determining if new a LOD level should be loaded.
- **LOD levels** determines how deep the subdividing hierarchy will be and therefore how detailed the surface will be at maximum detail level.
- **Level loading distances** are the distances in units when the subdivision to each level will happen.

Performance

- **LOD update interval** is a time in seconds between the update checks when the LOD system check if any surfaces should be loaded or unloaded.
- **Simultaneous subdivisions** determines how many surfaces can subdivide at the same time. High number means lots of detail can be generated quickly but at the cost of performance (frames per second).

- **Use angle culling** determines if surface renderers will be disabled if outside of the culling angle.
- **Renderer culling angle** is the minimum angle between the LOD target and the surfaces closest point (relative to the planets center) when a surface will be not rendered. This helps to disable renderers that are in the camera frustum but are beyond the horizon of the planet.

PAINT HEIGHT



Brush settings

- **Brush mode** sets the brush function:
  - **Add** increases height
  - **Subtract** decreases height
  - **Set** moves the height toward set target
- **Brush size** means how big the brush is relative to the planet.
- **Brush strength** determines how powerful the brush effect will be.
- **Brush falloff** determines how the strength of the brush changes over distance.
- **Low limit / High limit**, the minimum and maximum heights the brush can make.

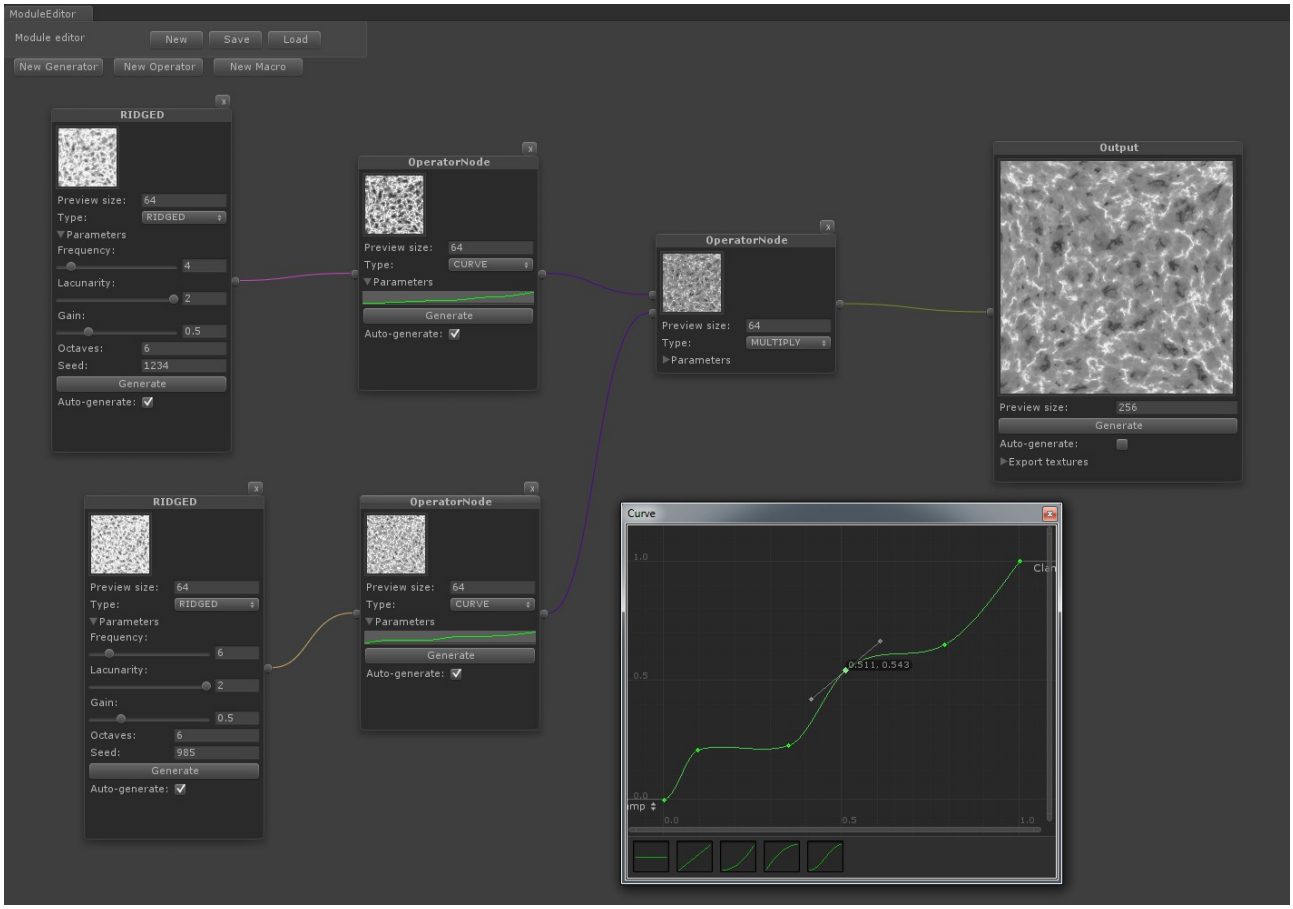
Options

- **Painting resolution** determines the size of the height buffer and therefore how detailed the painting will be.
- **Clear all** clears all details painted so far and resizes the buffer to the painting resolution.

Defining the Terrain

PlanetaryTerrain comes with the Module Editor extension that is used to create terrain modules and save them to files. The modules define the terrain shapes and the outputs are displayed as heightmaps.

Connecting nodes. To connect nodes just click the little button at the side of the node and connect it to another. The connectors at the left side of the node are inputs and on the right side there are outputs. Outputs can be only connected to inputs and vice versa.



There are 4 types of nodes: *Generators, Operators, Macros and Output.*

- **Generator node** generates noise or other source values.
  - **FBM** generates Fractal Brownian Motion based on 3D Simplex noise.
  - **Ridged** generates Ridged multifractal noise based on the FBM.
  - **Billow** generates Billow noise. (Equal to absolute Ridged noise).
  - **Const** generates a constant value.
- **Operator node** modifies the values coming from generators or other operator nodes.
  - **Abs** returns absolute value of given input.
  - **Add** adds two inputs together.

    - **Blend** linear interpolation of two inputs using a third input.
    - **Clamp** truncates the input to given range.
    - **Exponent** raises the input to given exponent.
    - **Invert** returns inverted value.
    - **Max** selects the higher of two inputs.
    - **Min** selectes the lower of two inputs.
    - **Multiply** multiplies two inputs.
    - **Power** raises the first input to the power of the second input value.
    - **Substract** subtracts the second value from the first.
    - **Terrace** creates a terrace between given range.
    - **Translate** translates the noise in xyz direction.
    - **Divide** divides the first input by the second.
    - **Curve** displaces the input with a user defined curve.
    - **Weight** weights the input towards given target value.
- **Output node** shows the final output of the graph and can also be used to export textures.
- **Macro node** loads a full graph to be used as a single node. Useful for creating complex graphs.

Planet generation in-depth

This chapter details how the terrain generation works under the hood. You are not required to know these in order to use the tools to generate your own planets, but it might be useful.

Here is a step-by-step breakdown of the main generation procedure.

1. The planet starts as a cube, each surface being stored as an array of points. Number of surfaces is determined by the subdivision level, starting at 6 (one for each side of the cube) and subdividing each one to 4 more until subdivision level is reached.
2. Each point is transformed to a point in sphere surface with uniform distance between the points.
3. Each point is displaced by a height value given by the coherent noise algorithm for the given points coordinate in 3D space.
4. The points are scaled to the planets desired radius.

Code locations:

- Subdivision happens in the *Planet* classes *Generate* function.
- Transformation of the points happens in *Surface* class under the function *CalculateGeometry*.

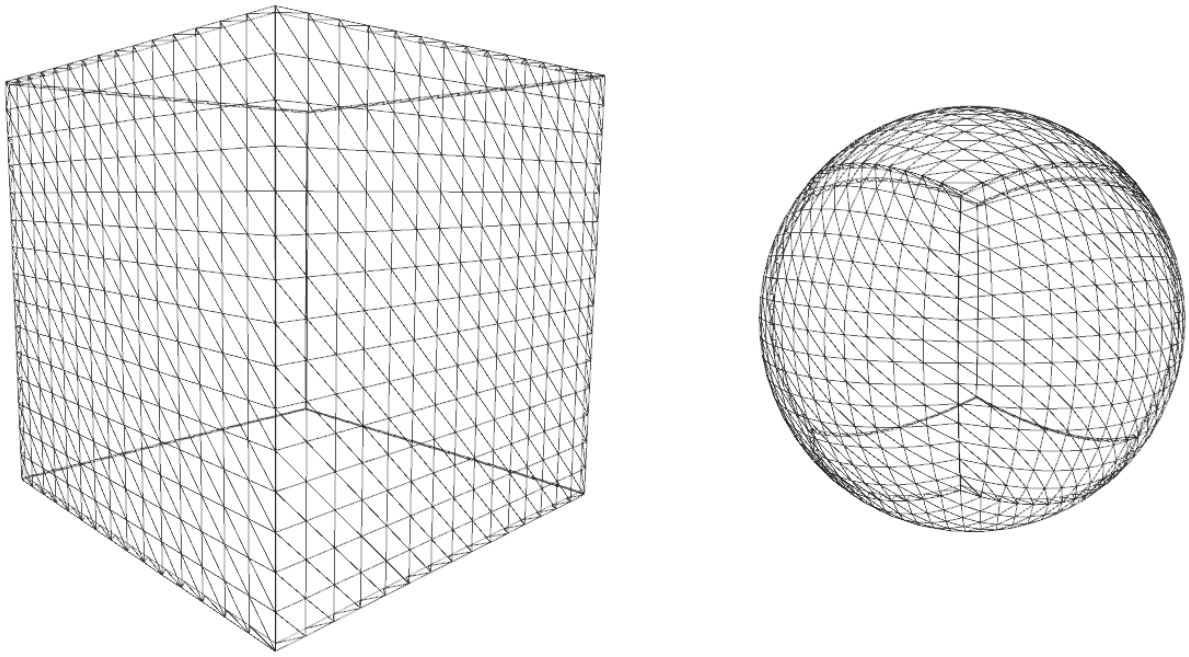


Image 1. The mesh before and after the sperifying.

LOD works by further subdividing each surface to 4 smaller ones and generating each new one with the same procedure.

Each surface has additional vertices directly below the border points, to eliminate visible seams formed when surfaces of different subdivision levels are aligned.

Data such as the height value from noise algorithm and the polarity (distance in 0-1 range from either pole) is stored to the vertex color channel for use in the shader.

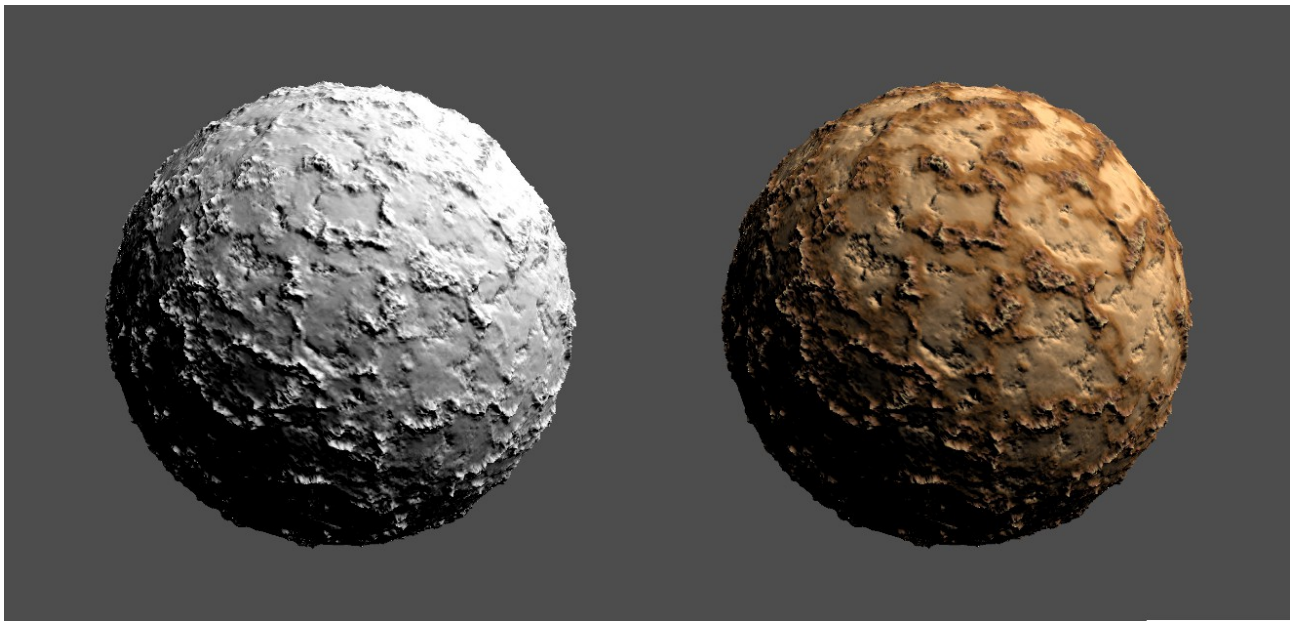


Image 2,. A dense planet mesh (32x32, subdivision level 8) before and after coloring.

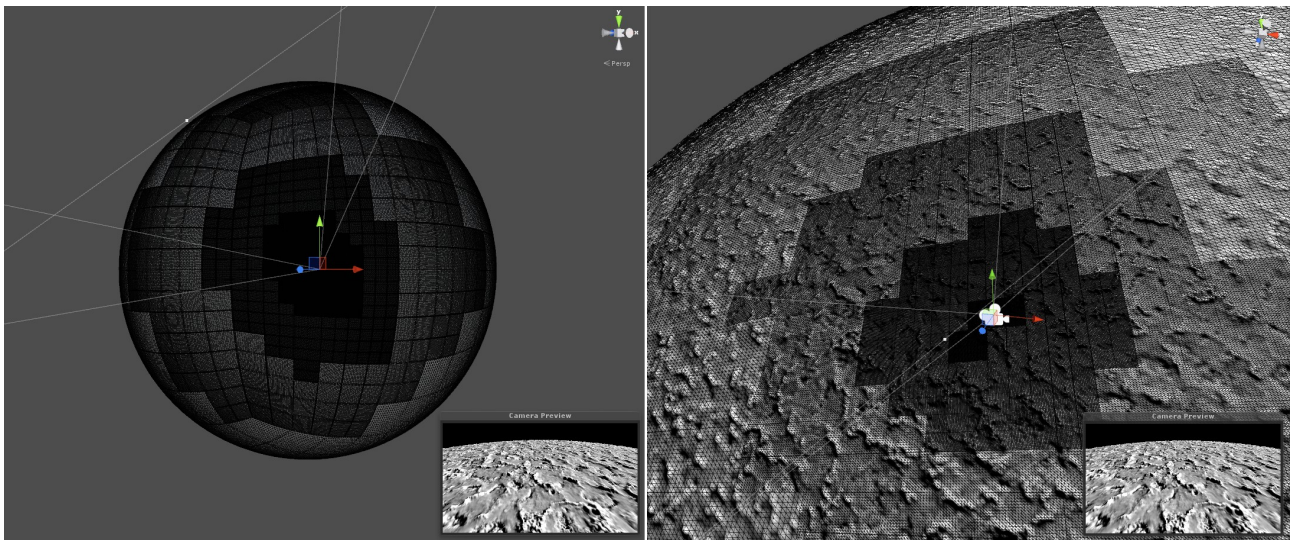


Image 3. Dynamic LOD in action. The settings are subdivision level 2, 16x16 meshes and LOD level 8.