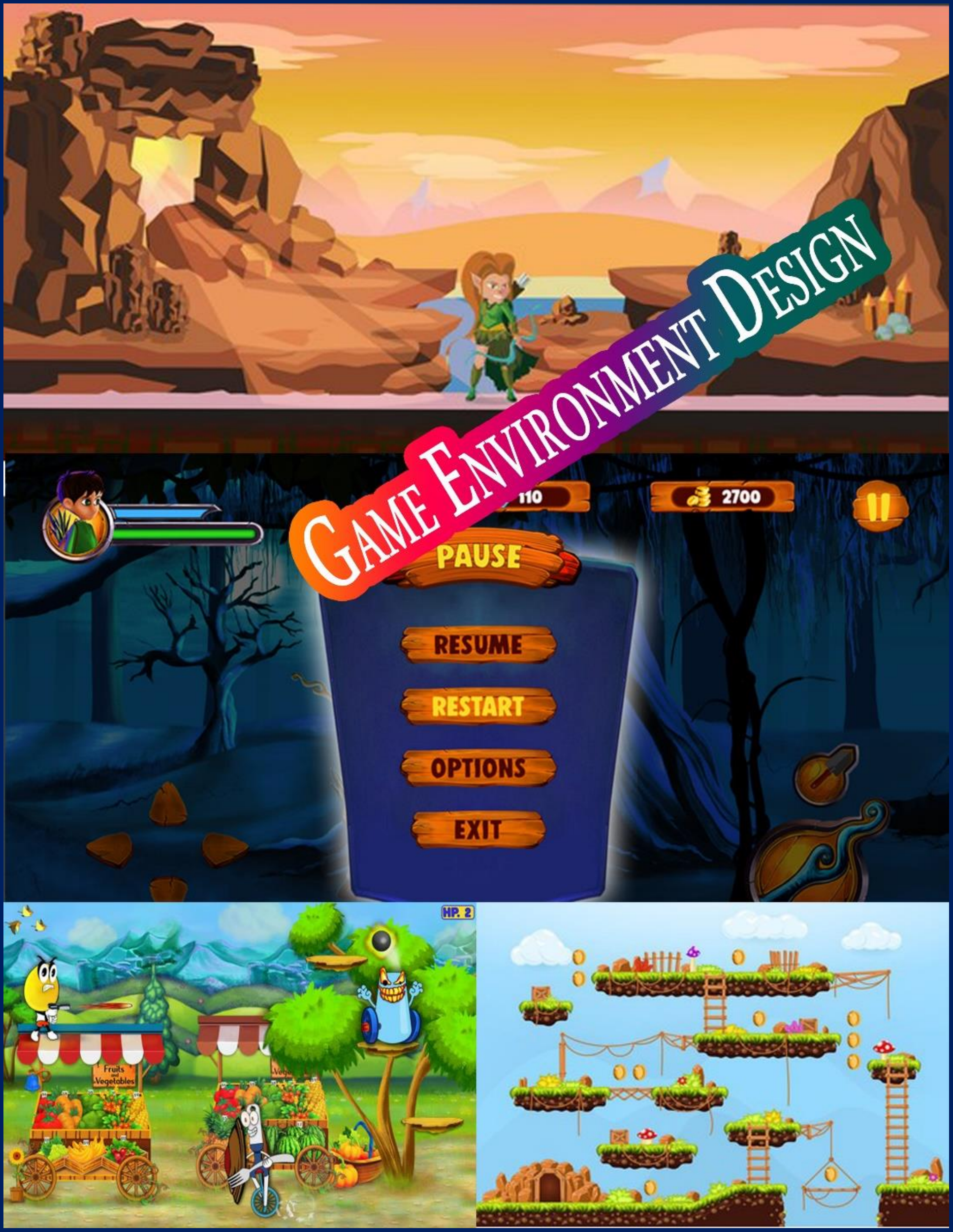
Game Environment Design



Title: Game Environment Design

By: Ali Akbary

Chapter 5 Creating Game World

Learning Outcome

Objectives of this chapter are: -

- 2D Game Environment
- 3D Game Environment
  - 3D Primitives
- Creating 3D Environment
  - Terrain

How To Create The Game World?

The game in term of dimension is divided into two categories: -

- 2D games
- 3D games

2d Game Envoronmet Developments

Here is the example of 2D game world created with Adobe illustrator: -

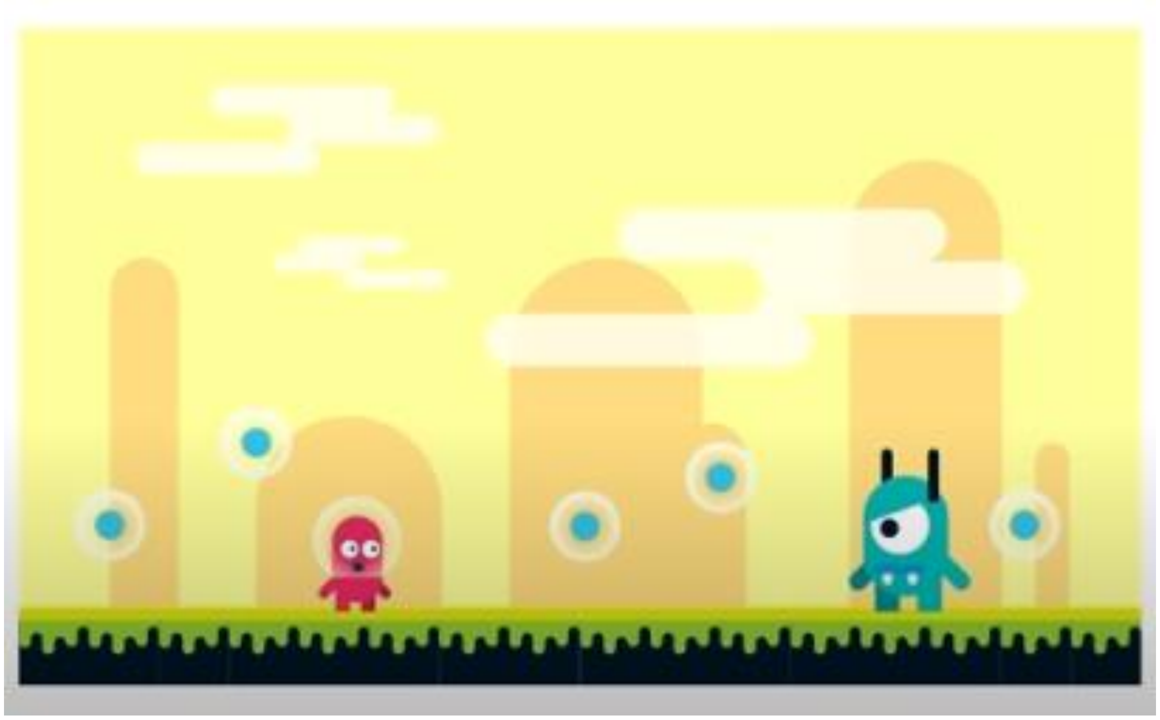


Figure 1 2D game design with character and enemy

For 2D games Environment's creation, we can use many applications software which supports it. We can use unity 3d application tool for 2 D games. Also, we can use Adobe Illustrator or Photoshop or other application which supports.

3D Game Environment developments

When we are developing the game world, we will use 3D application tools to create desired world for our game. There are many 3D application tools exist for such creation. Here we will use Unity 3D for our game world.

For creating 3D world, we need to start from somewhere. For most of the 3D creation, normally we will use primitives for starting point and will modify it and come up with desired object. First, we get familiar with unity 3D primitives' types

Primitive And Placeholder Objects

Unity can work with 3D models of any shape that can be created with modeling software. However, there are also a number of primitive object types that can be created directly within Unity, namely: -

- Cube
- Sphere
- Capsule
- Cylinder
- Plane
- And Quad

These objects are often useful in their own right (a plane is commonly used as a flat ground surface, for example) but they also offer a quick way to create placeholders and prototypes for testing purposes.

Any of the primitives can be added to the scene using the appropriate item on the menu from GameObject > 3D Object.

Cube

The cube with sides one unit long, textured so that the image is repeated on each of the six faces. As it stands, a cube isn't really a very common object in most games but when scaled, it is very useful for walls, posts, boxes, steps and other similar items. It is also a handy placeholder object for programmers to use during development when a finished model is not yet available.

For example, a car body can be crudely modelled using an elongated box of roughly the right dimensions. Although this is not suitable for the finished game, it is fine as a simple representative object for testing the car's control code. Since a cube's edges are one unit in length, you can check the proportions of a mesh imported into the scene

by adding a cube close by and comparing the sizes.

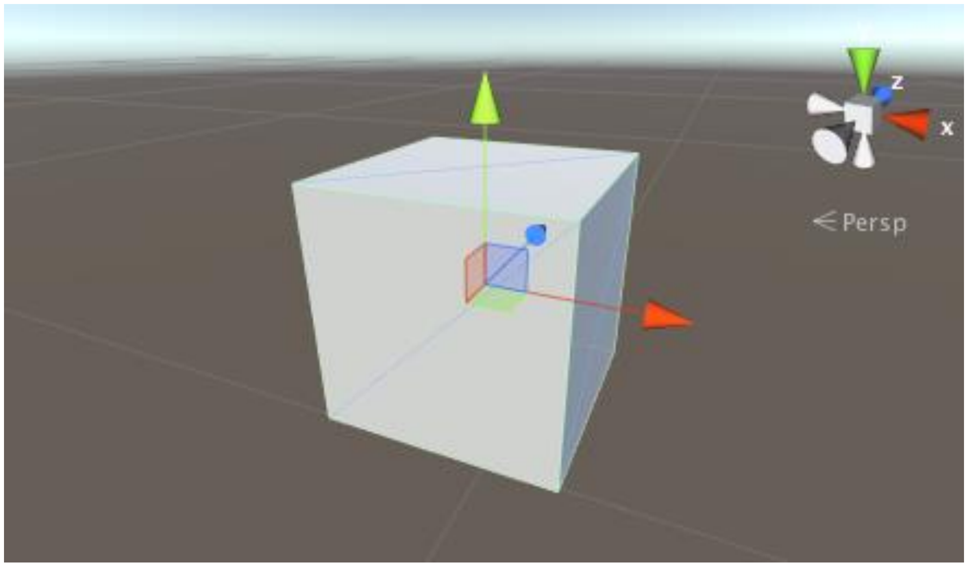


Figure 2 cube

Sphere

This is a sphere of unit diameter (0.5-unit radius), textured so that the entire image wraps around once with the top and bottom "pinched" at the poles. Spheres are obviously useful for representing balls, planets and projectiles but a semi-transparent sphere can also make a nice GUI device for representing the radius of an effect.

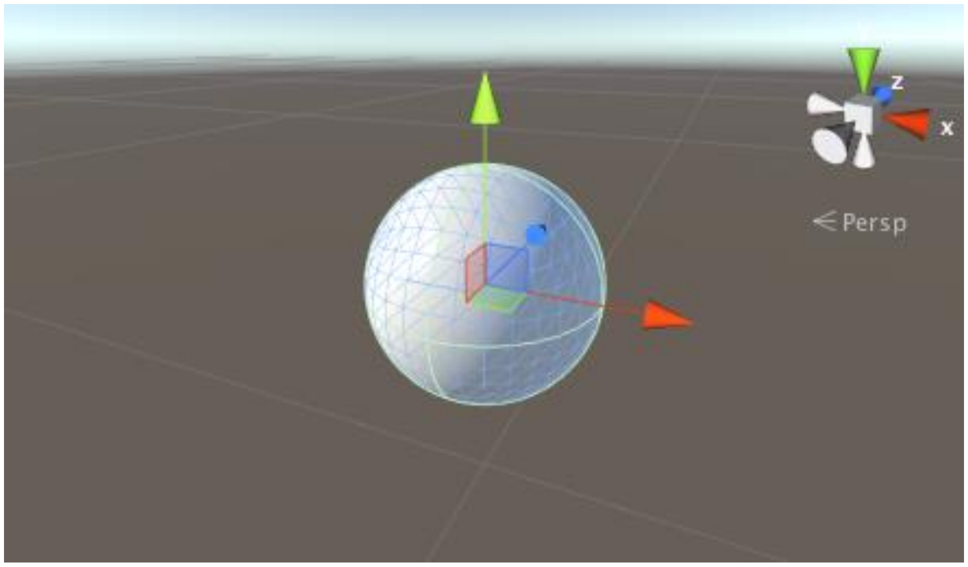


Figure 3 Sphere

Capsule

A capsule is a cylinder with hemispherical caps at the ends. The object is one unit in diameter and two units high (the body is one unit and the two caps are half a unit each). It is textured so that the image wraps around exactly once, pinched at each hemisphere's apex. While there aren't many real-world objects with this shape, the capsule is a useful placeholder for prototyping. In particular, the physics of a rounded object are sometimes better than those of a box for certain tasks.

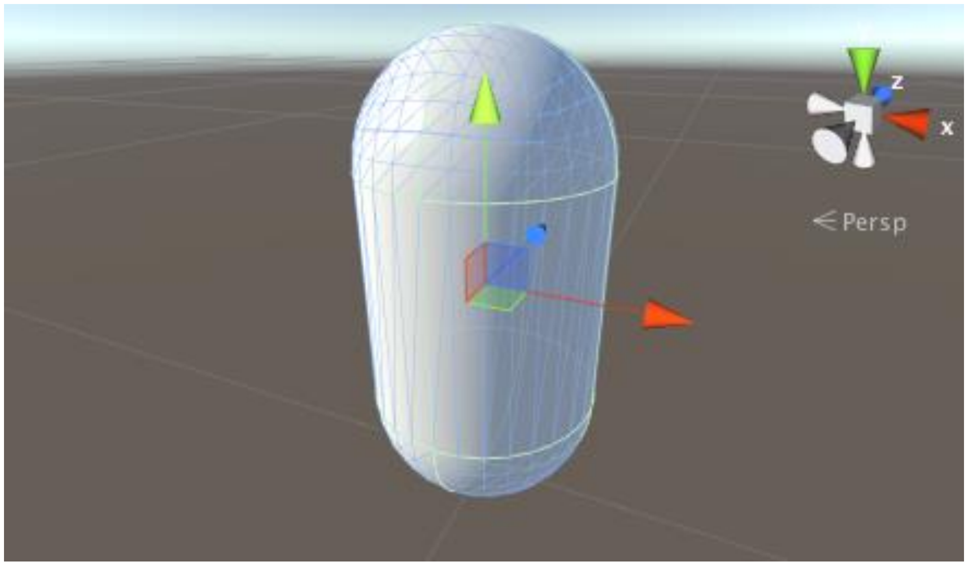


Figure 4 Capsule

Cylinder

This is a simple cylinder which is two units high and one unit in diameter, textured so that the image wraps once around the tube shape of the body but also appears separately in the two flat, circular ends. Cylinders are very handy for creating posts, rods and wheels but you should note that the shape of the collider is actually a capsule (there is no primitive cylinder collider in Unity).

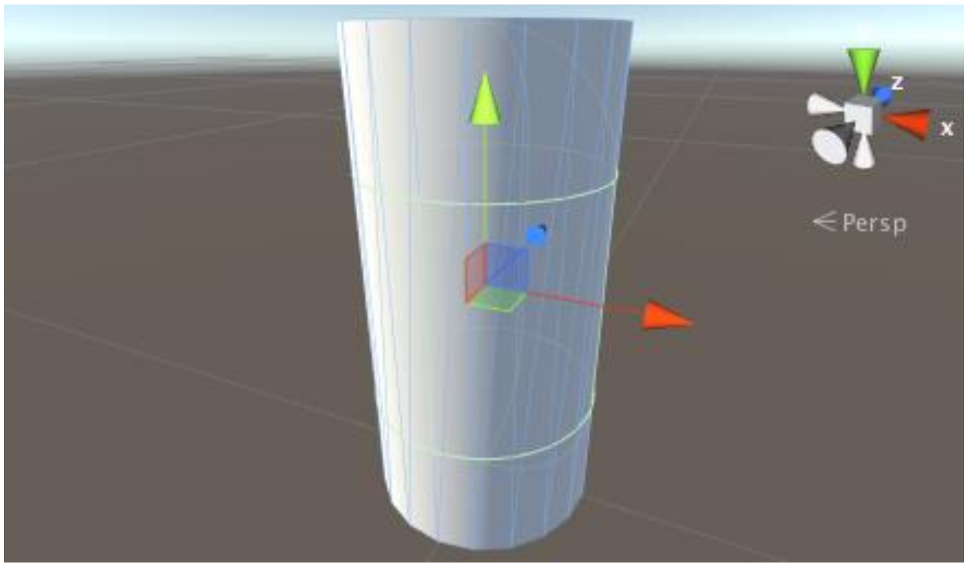


Figure 5 Cylinder

You should create a mesh of the appropriate shape in a modeling program and attach a mesh collider if you need an accurate cylindrical collider for physics purposes.

Plane

This is a flat square with edges ten units long oriented in the XZ plane of the local coordinate space. It is textured so that the whole image appears exactly once within the square. A plane is useful for most kinds of flat surface, such as floors and walls. A surface is also needed sometimes for showing images or movies in GUI and special effects. Although a plane can be used for things like this, the simpler quad primitive is often a more natural fit to the task.

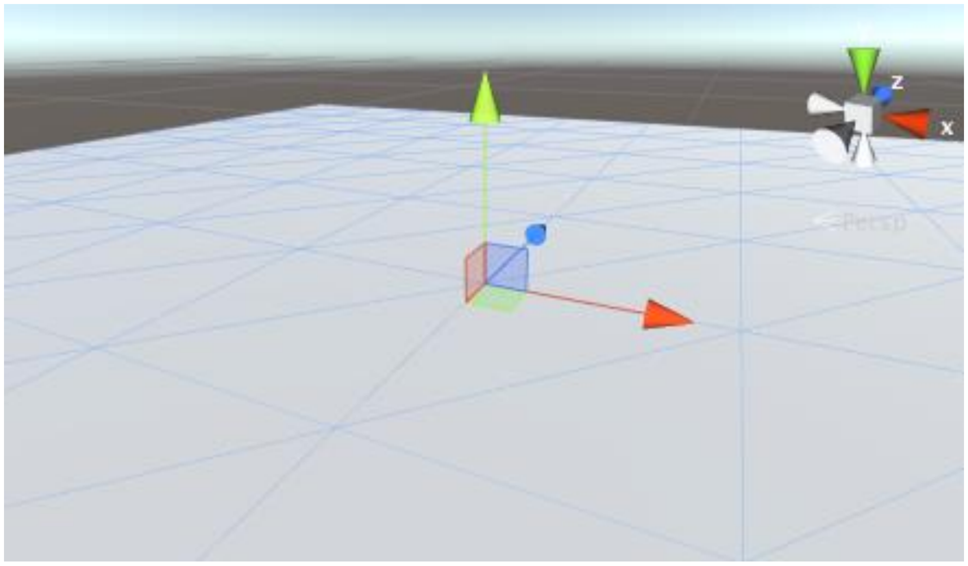


Figure 6 Plane

Quad

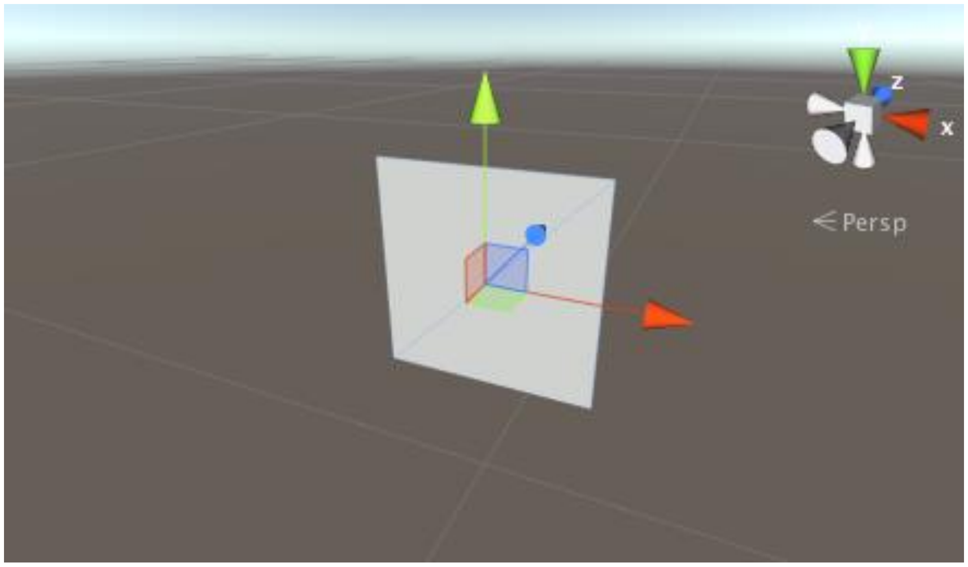


Figure 7 Quad

The quad primitive resembles the plane but its edges are only one unit long and the surface is oriented in the XY plane of the local coordinate space. Also, a quad is divided into just two triangles whereas the plane contains two hundred. A quad is useful in cases where a scene object must be used simply as a display screen for an image or movie. Simple GUI and information displays can be implemented with quads, as can particles, sprites and "impostor" images that substitute for solid objects viewed at a distance.

CREATING ENVIRONMENTS

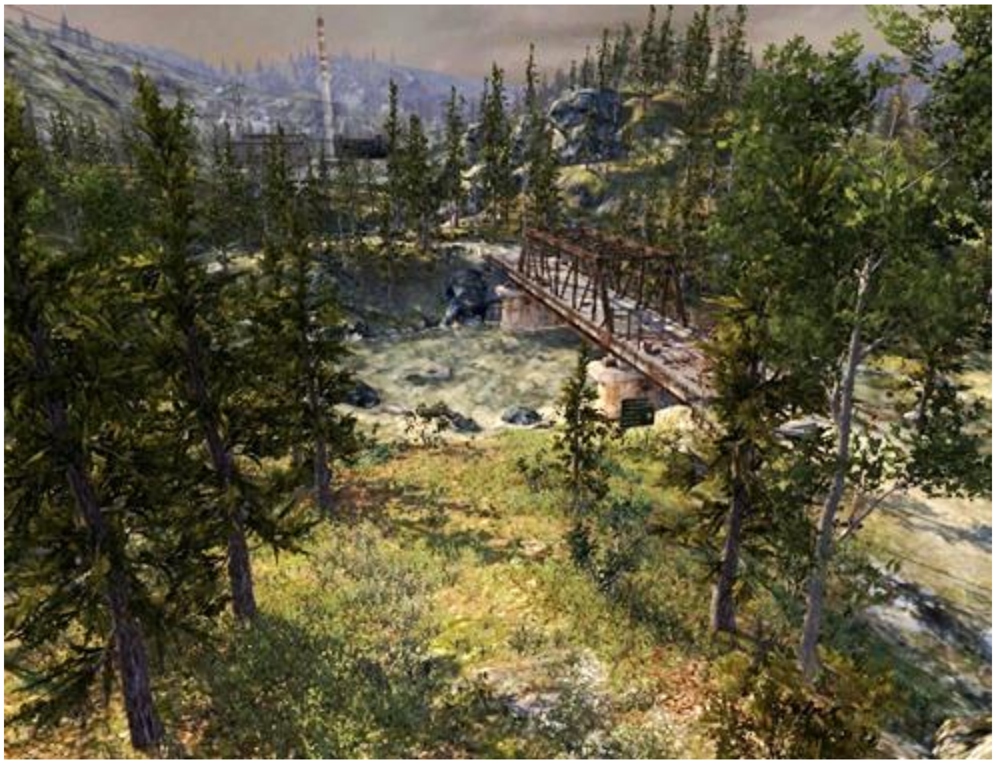


Figure 8 Unity provides a selection of tools that let you create environmental features such as landforms and vegetation.

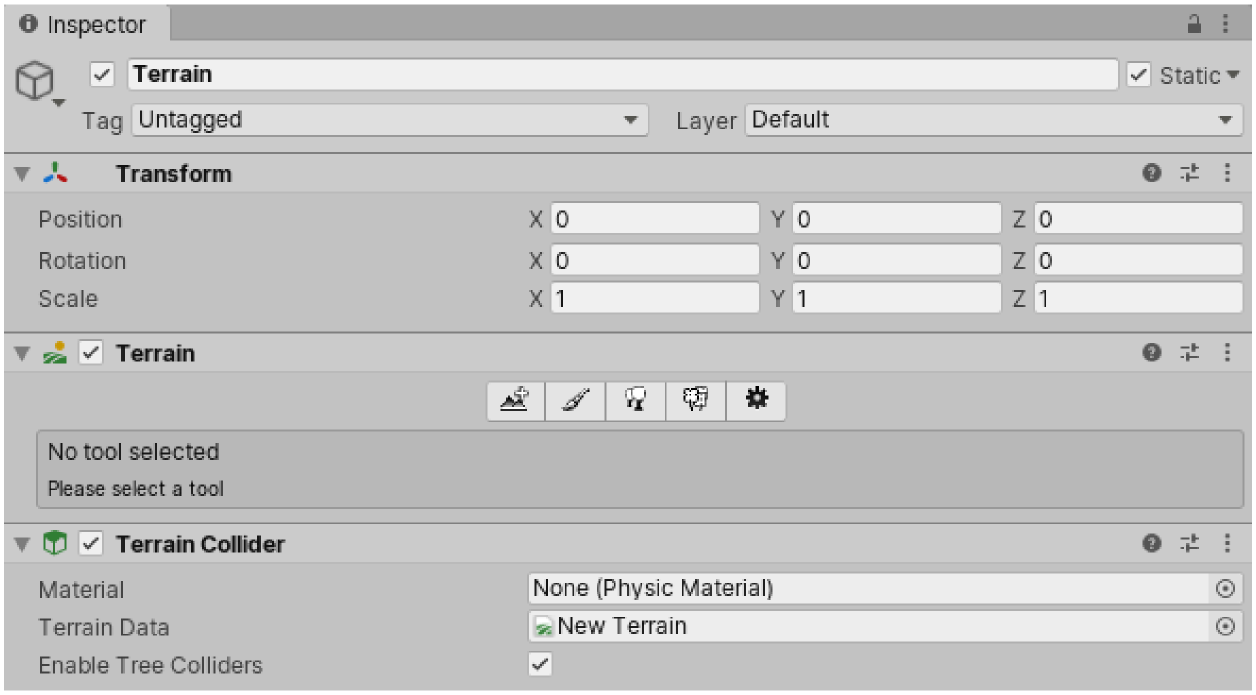
TERRAIN

The Unity Editor includes a built-in set of Terrain features that allow you to add landscapes to your game. In the Editor, you can create multiple Terrain tiles, adjust the height or appearance of your landscape, and add trees or grass to it. At runtime, Unity optimizes built-in Terrain rendering for efficiency.

The pages in this section explain the various built-in options available for Terrain, and how to use them.

Creating and editing Terrains

To add a Terrain GameObject to your Scene, select GameObject > 3D Object > Terrain from the menu. This also adds a corresponding Terrain Asset to the Project view. When you do this, the landscape is initially a large, flat plane. The Terrain's Inspector window provides a number of tools to create detailed landscape features.


Inspector

Terrain ✓ Static
Tag Untagged Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Terrain
No tool selected
Please select a tool

Terrain Collider
Material None (Physic Material)
Terrain Data New Terrain
Enable Tree Colliders ✓

Figure 9 Terrain editing tools in the Inspector

Terrain editing tools in the Inspector

The **toolbar** provides five options to adjust your Terrain:

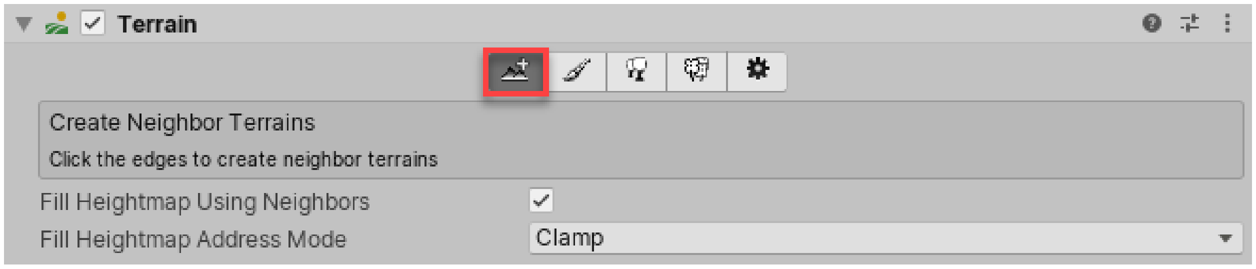
- Create adjacent Terrain tiles.
- Sculpt and paint your Terrain.
- Add trees.
- Add details such as grass, flowers, and rocks.
- Change general settings for the selected Terrain.

Select the paintbrush icon to access painting tools, which allow you to modify the Terrain. Use the cursor to sculpt the height of the Terrain, or paint texture onto the Terrain. Choose from several built-in Brush shapes, or define your own Brush using a texture. You can also change the size and opacity (the strength of the applied effect) of the Brush. Once you've defined the properties, your cursor takes the shape of the selected Brush. Click or drag on the Terrain to create different shapes and textures.

Similar to how you paint with a Brush on the Terrain, you can add textures, trees, and details like grass, flowers, and rocks. You can also create additional connected Terrain tiles, change the height of an entire tile, and even write custom Brushes with complex effects.

Create Neighbor Terrains

The Create Neighbor Terrains tool allows you to quickly create adjacent Terrain tiles, which automatically connect. In the Terrain Inspector, click the Create Neighbor Terrains icon.



Create Neighbor Terrains

Click the edges to create neighbor terrains

Fill Heightmap Using Neighbors ✓

Fill Heightmap Address Mode: Clamp

Figure 10 Create Neighbor Terrains tool in the Terrain Inspector

When you select the tool, Unity highlights areas around the selected Terrain tile, indicating spaces where you can place a new, connected tile.

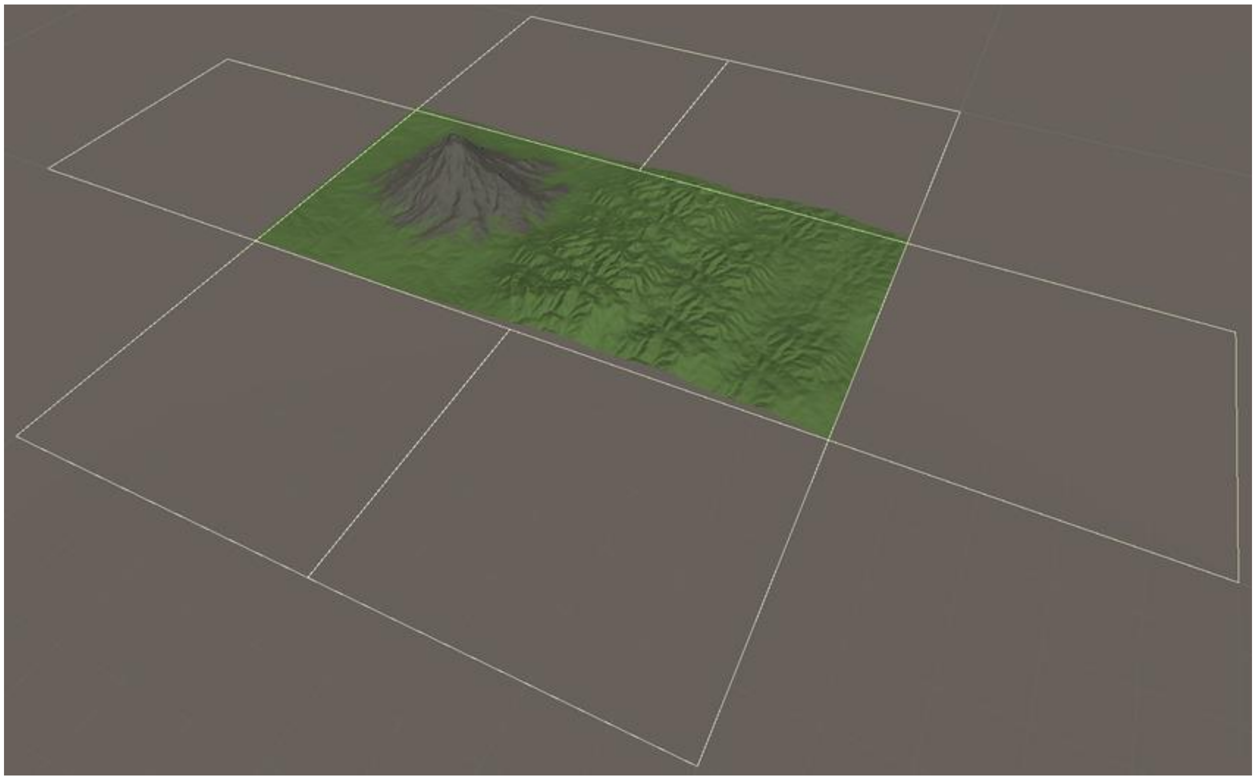


Figure 11 Spaces where you can create new Terrain tiles

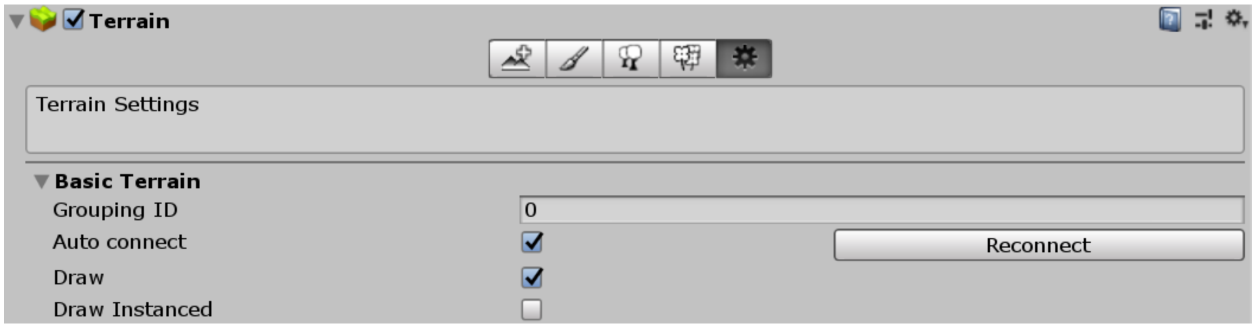
Enable the Fill Heightmap - Using Neighbors - checkbox to fill the new Terrain tile's heightmap with a cross-blend of neighboring Terrain tiles' heightmaps, which ensures that the height of the new tile's edges match up with adjacent tiles.

Choose a property from the Fill Heightmap Address Mode drop-down menu to determine how to cross-blend the heightmaps of adjacent tiles:

Property	Description
Clamp	Unity performs a cross-blend between the heights along the edges of neighboring Terrain tiles that share a border with the new tile. Each Terrain tile has up to four neighboring tiles: top, bottom, left, and right. If there is no tile in any of the four adjacent spaces, the heights along that respective border are taken as zero.
Mirror	Unity mirrors each of the adjacent Terrain tiles, and cross-blends their heightmaps to produce the heightmap for the new tile. If there is no tile in any of the four adjacent spaces, the heights for that specific tile location are taken as zero.

To create a new Terrain tile, click any of the available spaces next to an existing tile. The Editor creates a new Terrain tile in the same group as the selected Terrain, and copies over the settings of the tile it connects to. It also creates a ewTerrainData Asset.

By default, Unity enables Auto connect in the Terrain Settings of a Terrain tile. When Auto connect is enabled, the Terrain system automatically manages the connections between neighboring Terrain tiles, and a tile automatically connects to any neighbors with the same Grouping ID.



Grouping ID: 0
Auto connect: ✓
Draw: ✓
Draw Instanced: ☐

Figure 12 Terrain Settings for a Terrain tile

On rare occasions, you might lose connections between tiles if you change the Grouping ID, or disable Auto connect for one or more tiles. To recreate connections between Terrain tiles, click the Reconnect button. Reconnect only connects two adjacent tiles if they have the same Grouping ID and if both tiles have Auto Connect enabled.

Connecting Terrain tiles in a group allows you to use other tools to paint textures or adjust the heightmaps of the group so that there are no seams. At run time, the Terrain system automatically blends the tessellation and normal map of connected tiles. This ensures they appear as a single piece of Terrain, without seams or artifacts.

If you attempt to paint across two unconnected tiles in a single stroke, Unity treats them as separate tiles, so any affects you apply might appear only on one tile, or display differently on each tile.

Terrain tools

To access the **Terrain** painting tools, click on a **Terrain** object in the Hierarchy window and open an Inspector window. In the Inspector, click the Paint Terrain (paintbrush) icon to reveal the list of Terrain tools.

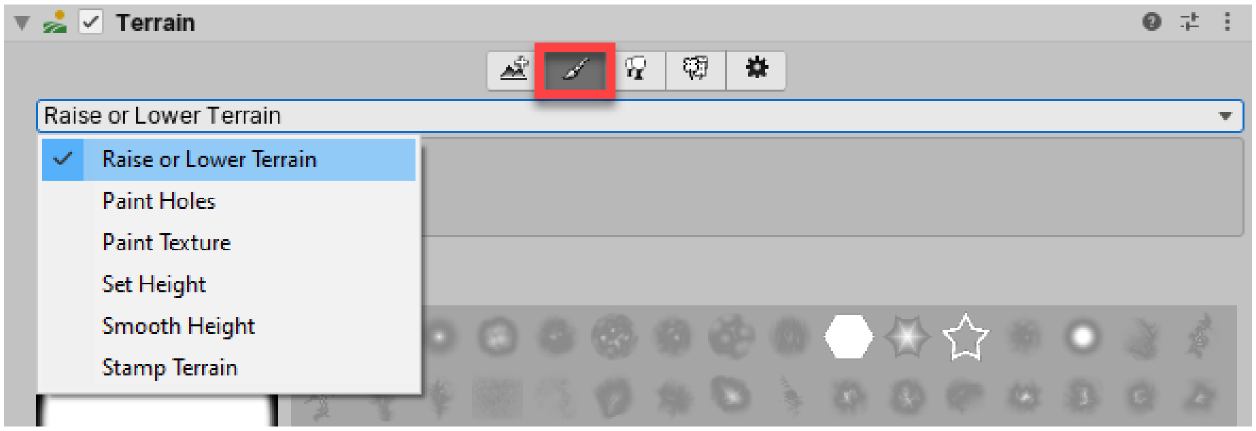


Figure 13 Terrain tools drop-down menu

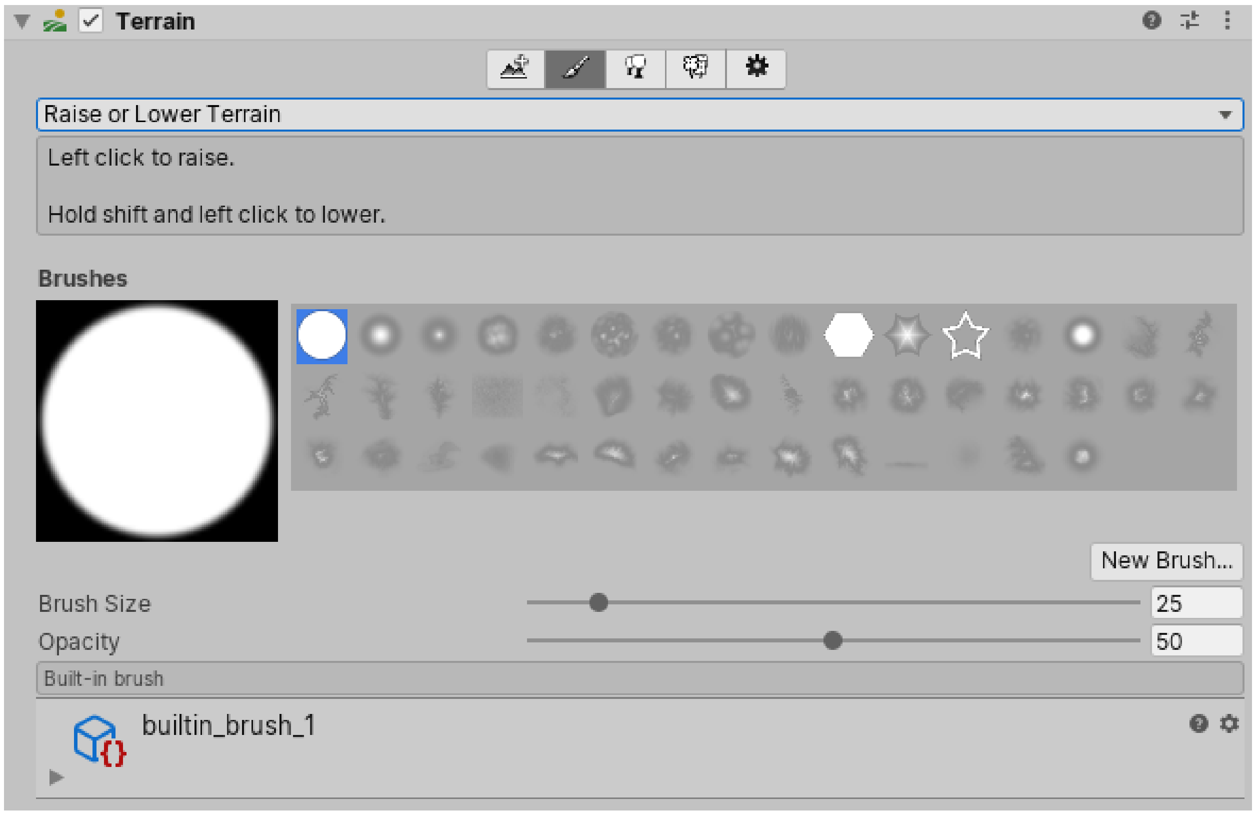
The Terrain component provides six distinct tools:

- Raise or Lower Terrain: paint the heightmap with a paintbrush tool.
- Paint Holes: hide portions of the Terrain.
- Paint Texture: apply surface textures.
- Set Height: adjust the heightmap toward a specific value.
- Smooth Height: smooth the heightmap to soften Terrain features.
- Stamp Terrain: stamp a brush shape on top of the current heightmap.

Raise or Lower Terrain

Use the Raise or Lower Terrain tool to alter the height of a Terrain tile.

To access the tool, click the Paint Terrain icon, and in the drop-down menu, select Raise or Lower Terrain. Select a brush from the palette, then click and drag the cursor over a Terrain object to raise its height. Click and drag while holding down the Shift key to lower the Terrain height.



Left click to raise.

Hold shift and left click to lower.

Brush Size 25

Opacity 50

builtin_brush_1

Figure 14 Raise or Lower Terrain tool in the Terrain Inspector

Use the Brush Size slider to control the size of your tool to create different effects, from large mountains to tiny details. The Opacity slider determines the strength of the brush when you apply it to the Terrain. An Opacity value of 100 sets the brush to full strength, while a value of 50 sets the brush to half strength.

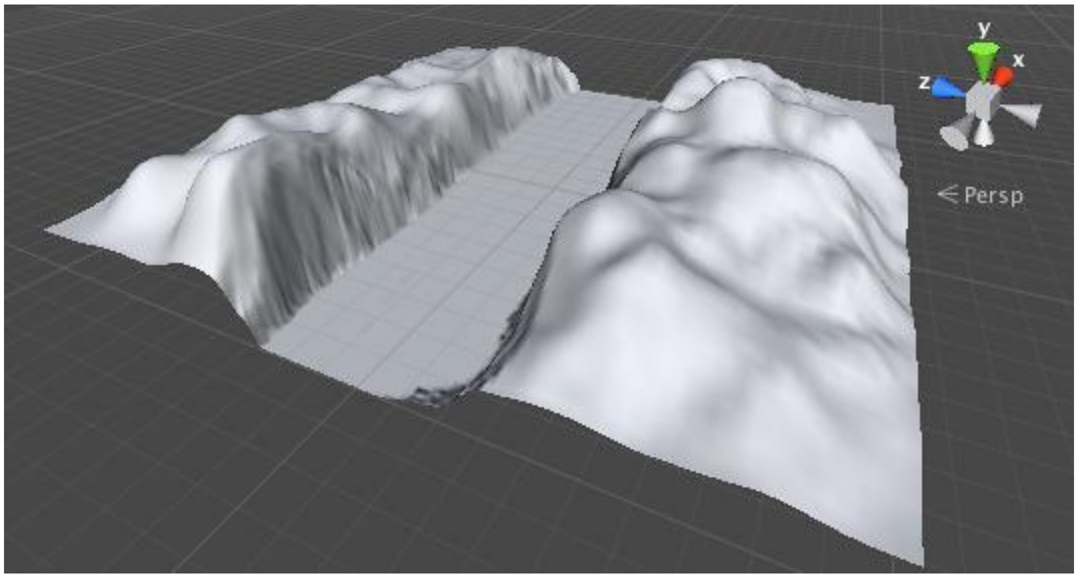


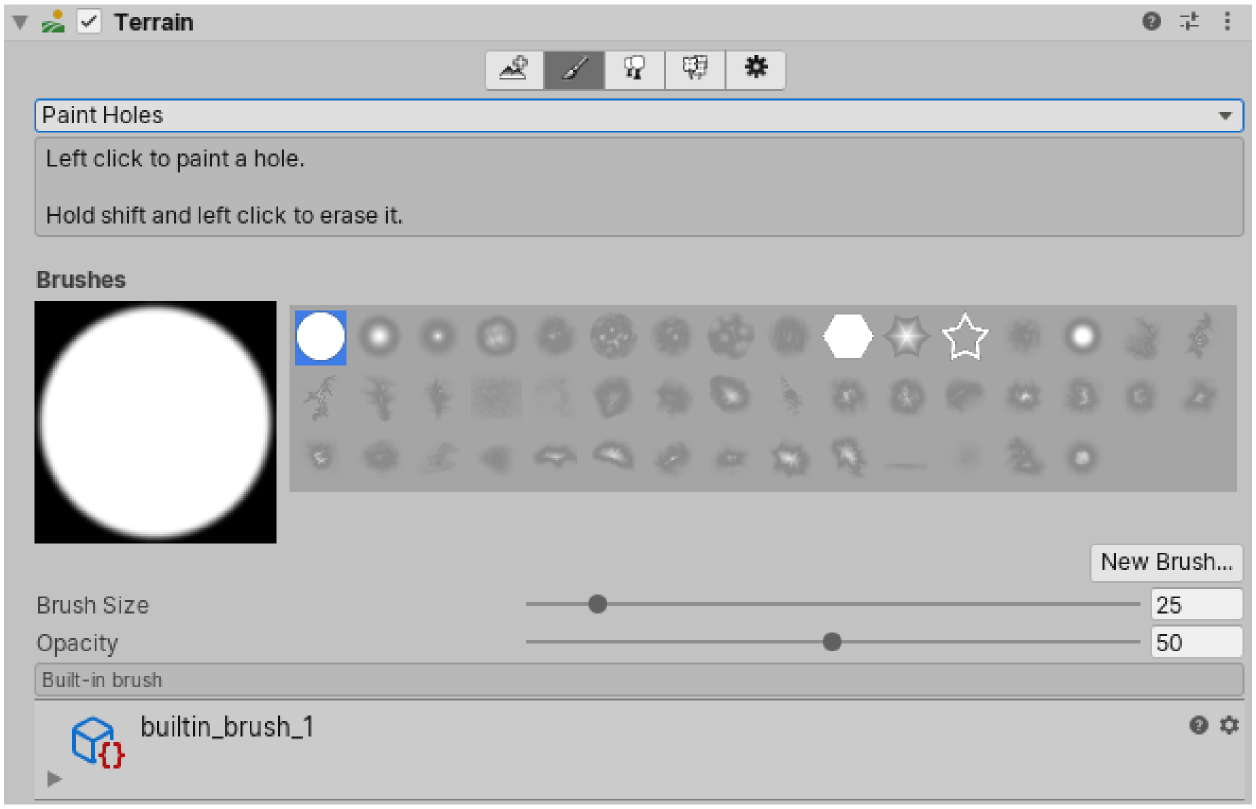
Figure 15 Rolling hills divided by a steep valley You can also use heightmaps to edit the height of your Terrain.

Use different brushes to create a variety of effects. For example, increase the height with a soft-edged brush to create rolling hills, and then decrease the height of some areas with a hard-edged brush to cut steep cliffs and valleys.

Paint Holes

Use the Paint Holes tool to hide portions of your Terrain. It allows you to paint openings in the Terrain for formations such as caves and cliffs.

To access the tool, click the Paint Terrain icon, and select Paint Holes from the drop-down menu.



Terrain

Paint Holes

Left click to paint a hole.

Hold shift and left click to erase it.

Brushes

New Brush...

Brush Size 25

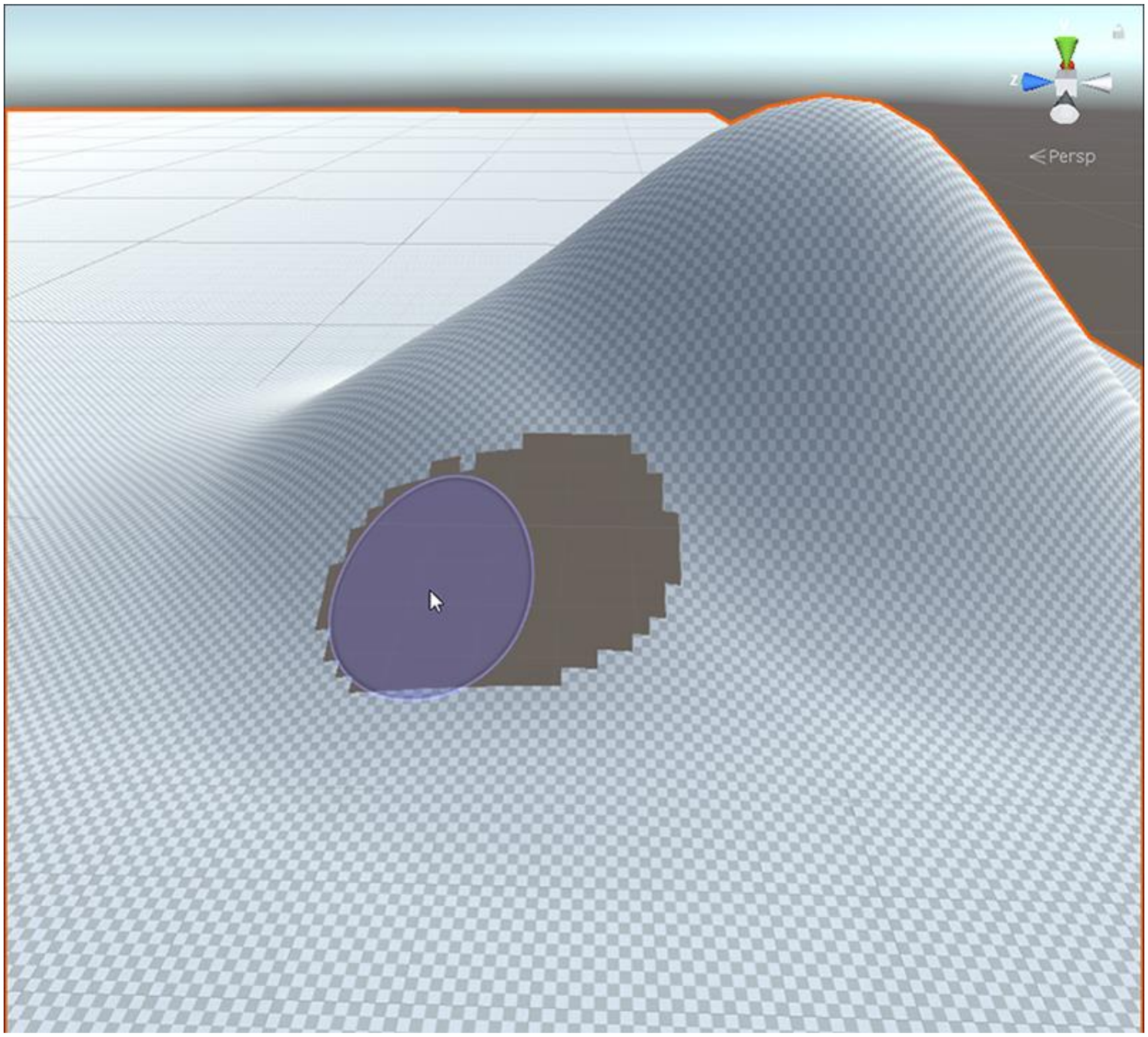
Opacity 50

Built-in brush

builtin_brush_1

Figure 16 Paint Holes tool in the Terrain Inspector

To paint holes, click and drag the cursor across the Terrain. To erase holes from the Terrain, click and drag while holding down the Shift key. Use the Brush Size slider to control the size of your tool. The Opacity slider determines the strength of the Brush when you apply it to the Terrain.



Internally, Unity uses a Texture to define the opacity mask for a Terrain surface. When you use the Paint Holes tool to paint on a Terrain, it modifies this Texture. Thus, any holes you paint are visible only if the Terrain Material you use clips or discards texels based on this mask.

Because this tool uses a Texture, you might see aliased edges surrounding holes you paint. Therefore, for example, when you make a cave, you might choose to hide the aliased edges of the hole with other geometry such as rock meshes.

Terrain holes work with lighting, physics, and NavMesh baking. Unity discards the Terrain information in areas where you paint holes to ensure accurate lighting, Terrain Colliders, and baked NavMeshes.

To support physics Colliders, the resolution of the hole’s mask Texture is equal to the resolution of the Terrain’s heightmap - 1.

Paint Texture

Use the Paint Texture tool to add textures, such as grass, snow, or sand, to your terrain. It allows you to draw areas of tiled texture directly onto the Terrain. In the Terrain Inspector, click the Paint Terrain icon, and select Paint Texture from the list of Terrain tools.

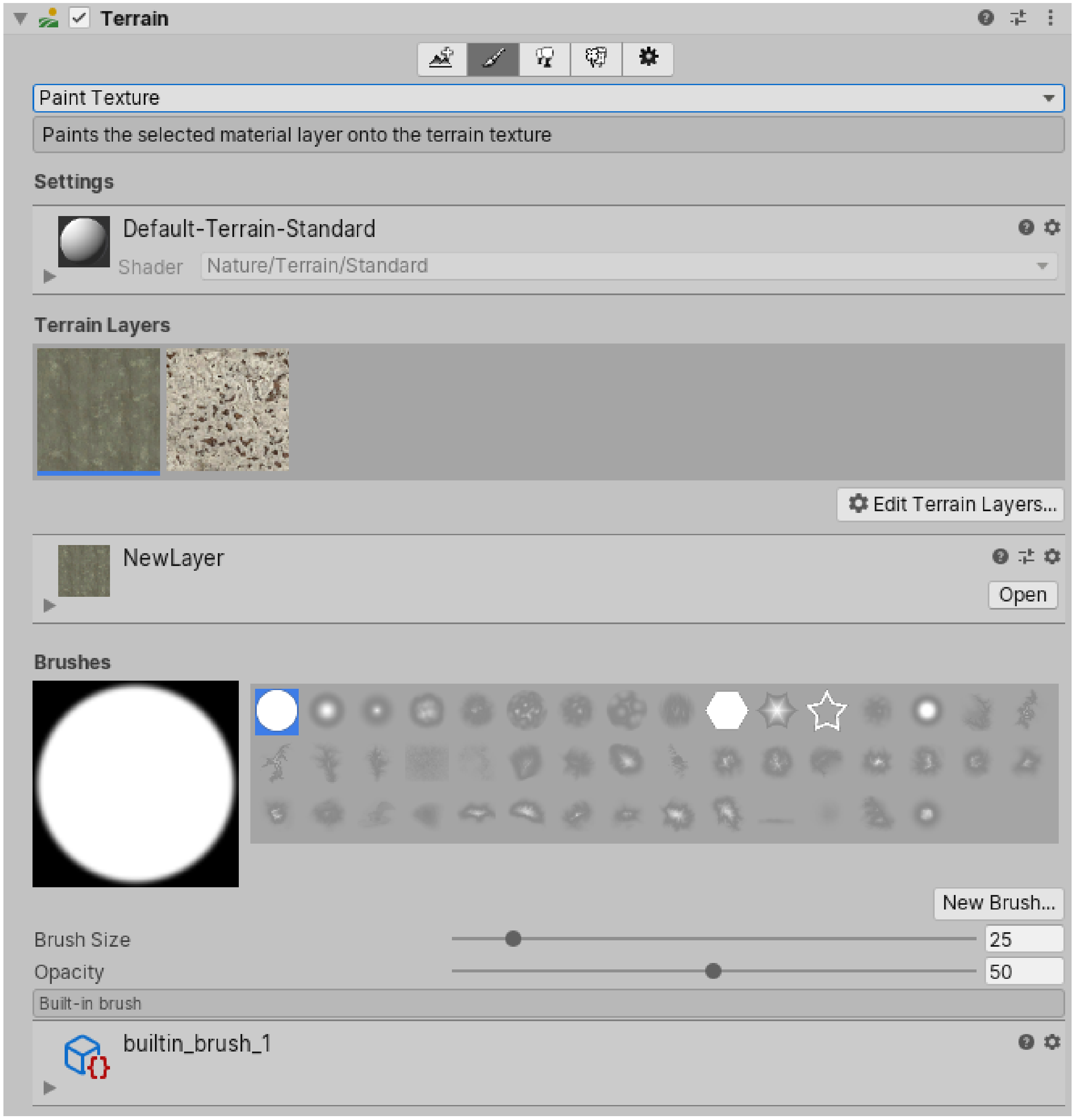


Figure 17 Paint Texture tool in the Terrain Inspector

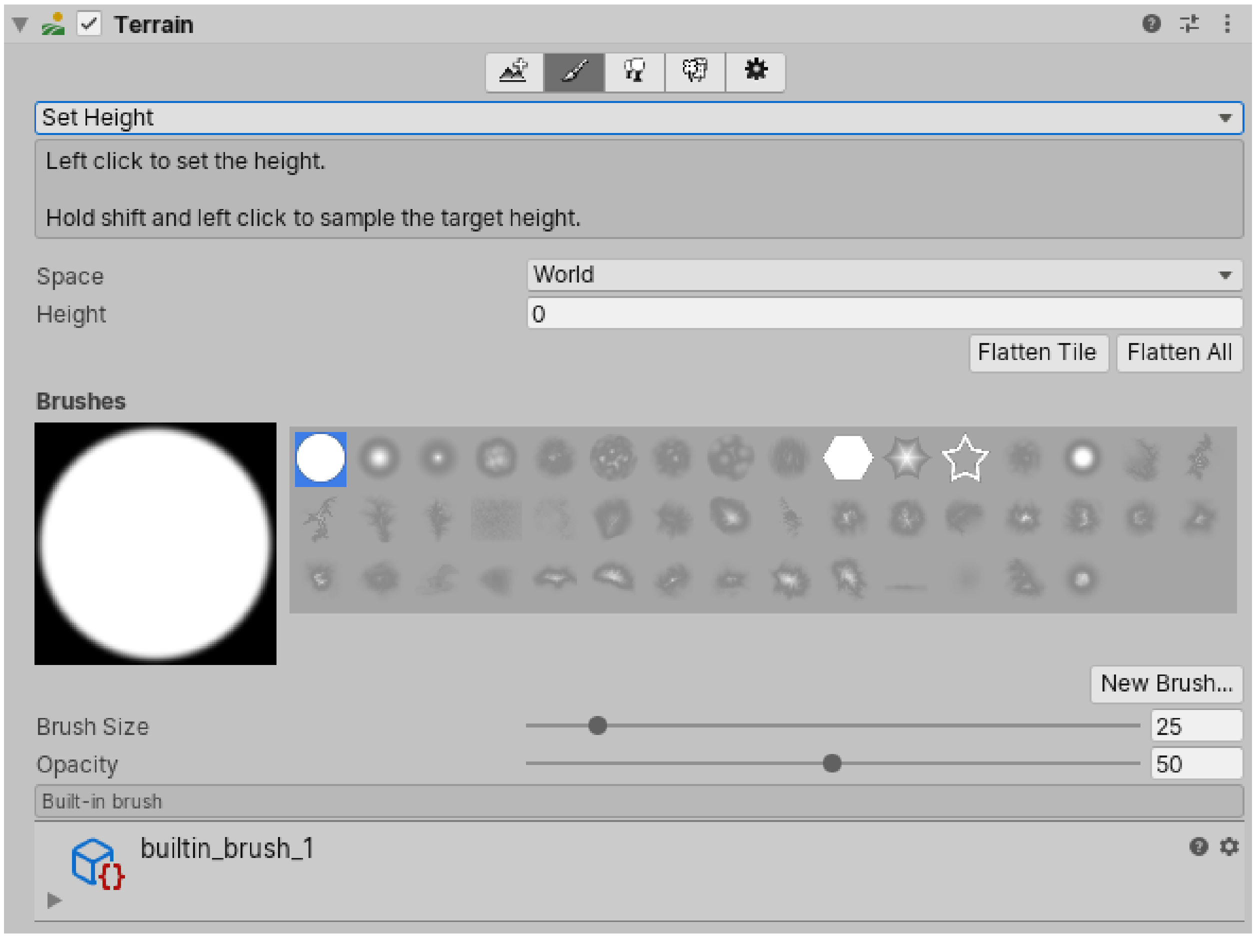
To configure the tool, you must first click the Edit Terrain Layers button to add Terrain Layers. The first Terrain Layer you add flood-fills your Terrain with the configured texture. You can add multiple Terrain Layers. However, the number of Terrain Layers each tile supports depends on your specific render pipeline. See the Rendering performance section on Terrain Layers for more information.

Next, you must choose a Brush for painting. Brushes are Assets based on Textures, which define the shape of a brush. Select from the built-in Brushes or create your own, then adjust the Brush Size and Opacity (strength of the applied effect) of the brush.

Finally, in the Scene view, click and drag the cursor across the Terrain to create areas of tiled texture. You can paint across tile boundaries to blend adjacent regions with a natural, organic look. Be aware, however, that the Terrain system adds the selected Terrain Layer to any Terrain you paint on, which might affect performance as mentioned above.

Set Height

Use the Set Height tool to adjust the height of an area on the Terrain to a specific value. To access the tool, click on the Paint Terrain icon, and select the Set Height tool from the drop-down menu.



Left click to set the height.

Hold shift and left click to sample the target height.

Space: World
Height: 0
Brush Size: 25
Opacity: 50
builtin_brush_1

Figure 18 Set Height tool in the Terrain Inspector

When you paint with the **Set Height** tool, it lowers areas of the Terrain currently above the target height, and raises areas below that height. Set Height is useful for creating flat, level areas in a Scene, such as plateaus or man-made features like roads, platforms, and steps.

Choose a property from the Space drop-down menu to specify whether the height offset is relative to Local or World space.

Property	Description
World	Select this to set the height offset to the value you enter in the Height field. However, be aware that the Set Height tool cannot lower a Terrain below its Transform Position Y coordinate, even if you enter a value lower than the Y coordinate.
Local	Select this to set the height offset relative to the Terrain. For example, if you enter 100 in the Height field, the height offset is the sum of the Terrain's Transform Position Y coordinate and 100 (terrain.transform.position.y + 100). The Height value you enter must range from 0 to the Terrain Height value in the Terrain settings.

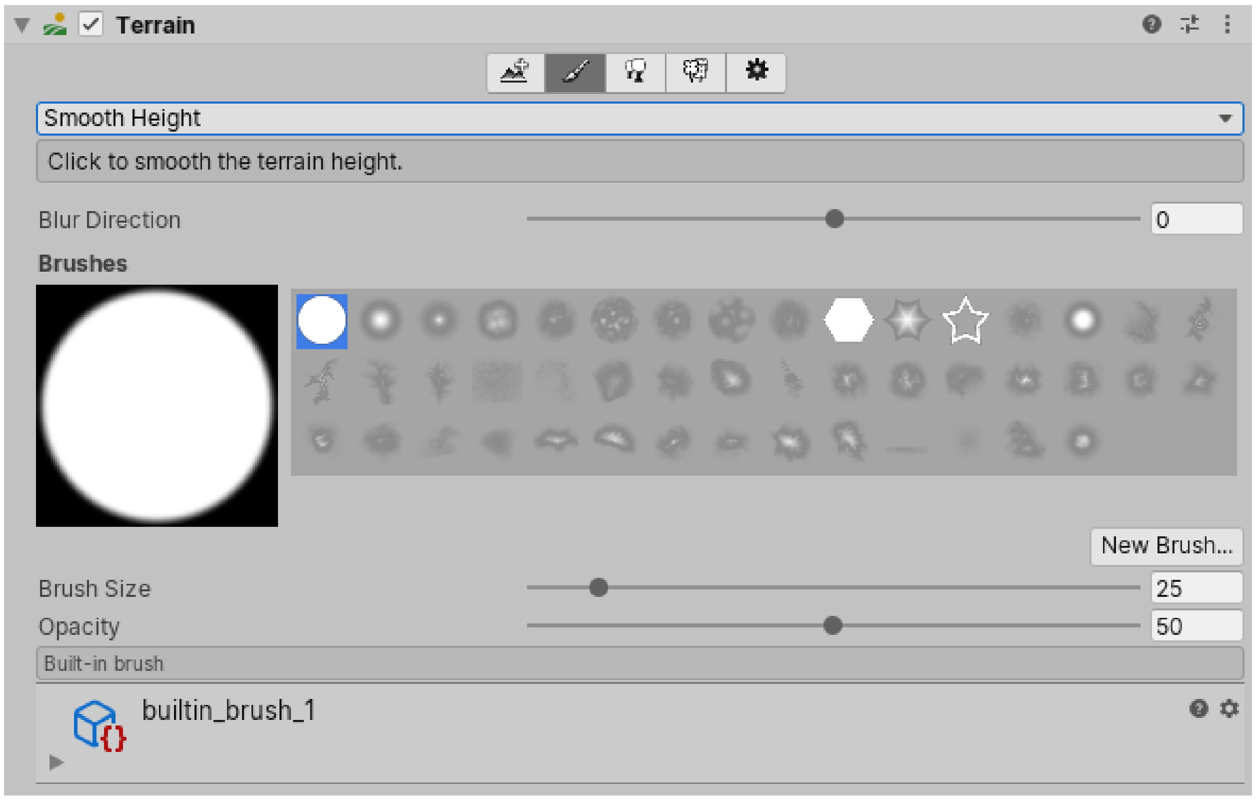
Enter a numerical value in the Height field, or use the Height property slider, to manually set a height. Alternatively, press Shift and click on the Terrain to sample the height at the cursor position, similar to how you would use the Eyedropper tool in an image editor.

If you press the Flatten Tile button under the Height field, it levels the whole Terrain tile to the height you specified. This is useful to set a raised ground level if, for example, you want the landscape to include both hills above the ground level and valleys below it. If you press the Flatten All button, it levels all Terrain tiles in the Scene.

The Brush Size value determines the size of the Brush to use, while the Opacity value determines how quickly the height of the area, you're painting reaches the set target height. You can also use height maps to edit the height of your Terrain.

Smooth Height

The Smooth Height tool smooths the heightmap and softens Terrain features. In the Terrain Inspector, click the Paint Terrain icon, and select Smooth Height from the list of Terrain tools.


Terrain

Smooth Height

Click to smooth the terrain height.

Blur Direction 0

Brushes

New Brush...

Brush Size 25

Opacity 50

Built-in brush

builtin_brush_1

Figure 19 Smooth Height tool in the Terrain Inspector

The Smooth Height tool averages out nearby areas, softens the landscape and reduces the appearance of abrupt changes; it does not significantly raise or lower Terrain height.

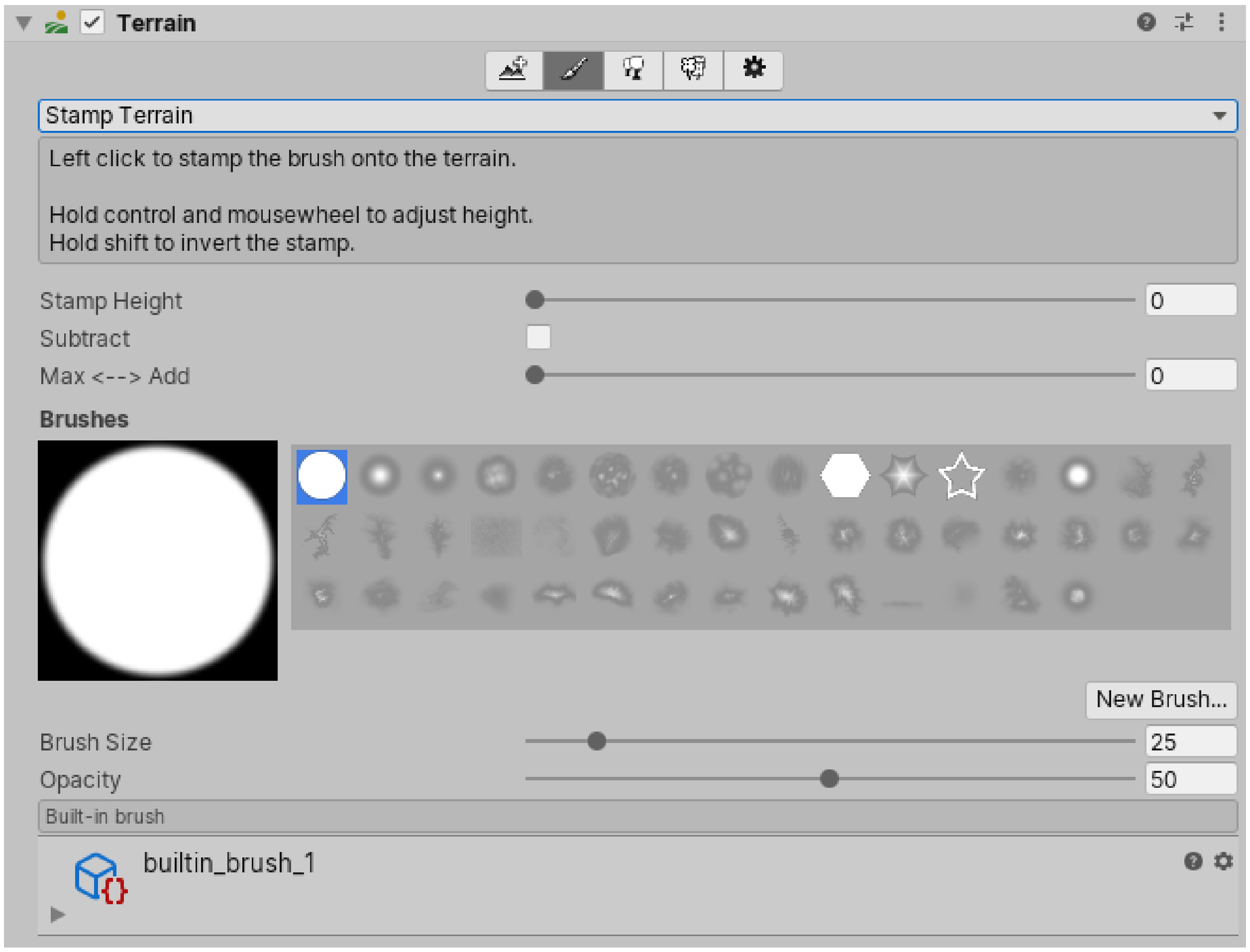
Smoothing is particularly useful after you paint with brushes containing high frequency patterns. These brush patterns tend to introduce sharp, jagged edges into a landscape, but you can use the Smooth Height tool to soften that roughness.

Adjust the Blur Direction value to control which areas to soften. If you set Blur Direction to –1, the tool softens exterior (convex) edges of your Terrain. If you set Blur Direction to 1, the tool softens interior (concave) edges of your Terrain. To smooth all parts of your Terrain evenly, set Blur Direction to 0.

The Brush Size value determines the size of the Brush to use, while the Opacity value determines how quickly the tool smooths out the area you're painting.

Stamp Terrain

Use the Stamp Terrain tool to stamp a brush shape on top of the current heightmap. In the Terrain Inspector, click on the Paint Terrain icon and select Stamp Terrain from the drop-down menu.



Terrain

Stamp Terrain

Left click to stamp the brush onto the terrain.

Hold control and mousewheel to adjust height.
Hold shift to invert the stamp.

Stamp Height 0
Subtract
Max <--> Add 0

Brushes

Brush Size 25
Opacity 50

Built-in brush

builtin_brush_1

Figure 20 Stamp Terrain tool in the Terrain Inspector

Stamp Terrain is useful if you create a custom brush using a Texture that represents a heightmap with a specific geological feature, such as a hill.

With the Stamp Terrain tool, you can choose an existing brush and apply it with a single click. Each click raises the Terrain to the set Stamp Height in the shape of the selected brush. To multiply the Stamp Height by a percentage, move the Opacity slider to change its value. For example, a Stamp Height of 200 and an Opacity of 50% sets the height of each stamp to 100.

The Max <–> Add slider lets you choose whether to pick the maximum height, or add the height of your stamp to the Terrain's current height.

- If you set the **Max <–> Add** value to 0, then stamp onto the Terrain, Unity compares the height of your stamp to the current height of the stamped area, and sets the final height to the value that is higher.
- If you set the **Max <–> Add** value to 1, then stamp onto the Terrain, Unity adds the height of your stamp to the current height of the stamped area, so that the final height is the sum of both values.

Enable the Subtract checkbox to subtract the height of any stamps you apply to the Terrain from the existing height of the stamped area. Note that Subtract works only if your Max <–> Add value is greater than zero, for example, if you set the Max <–> Add value to 1. If the stamp height exceeds the current height of the stamped area, the system levels the height to zero.

Terrain Layers

A Terrain Layer is an Asset that defines a Terrain's surface qualities. A Terrain Layer holds Textures and other properties that the Terrain's Material uses to render the Terrain surfaces. Because Terrain Layers are Assets, you can easily reuse them on multiple Terrain tiles.

You can add Textures to the surface of a Terrain to create coloration and fine detail. Terrain GameObjects are usually large, so it is best to use a base Terrain Layer with Textures that tile over the surface and repeat seamlessly. You can use multiple Terrain Layers, each with different Textures, to build up interesting, varied Terrain surfaces.

The first Terrain Layer you apply to a Terrain automatically becomes the base layer, and spreads over the whole landscape. In addition, you can paint areas with other Terrain Layers to simulate different ground surfaces, such as grass, desert, or snow. To create a gradual transition between grassy countryside and a sandy beach, you might choose to apply Textures with variable opacity.

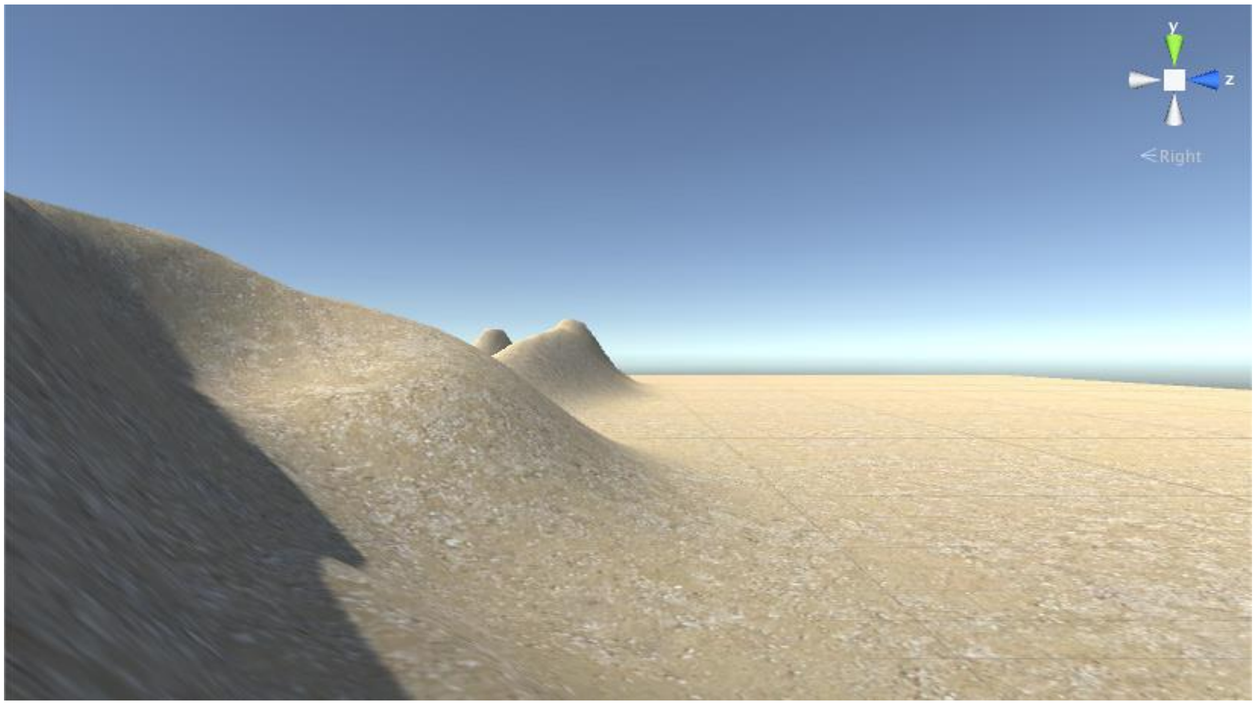
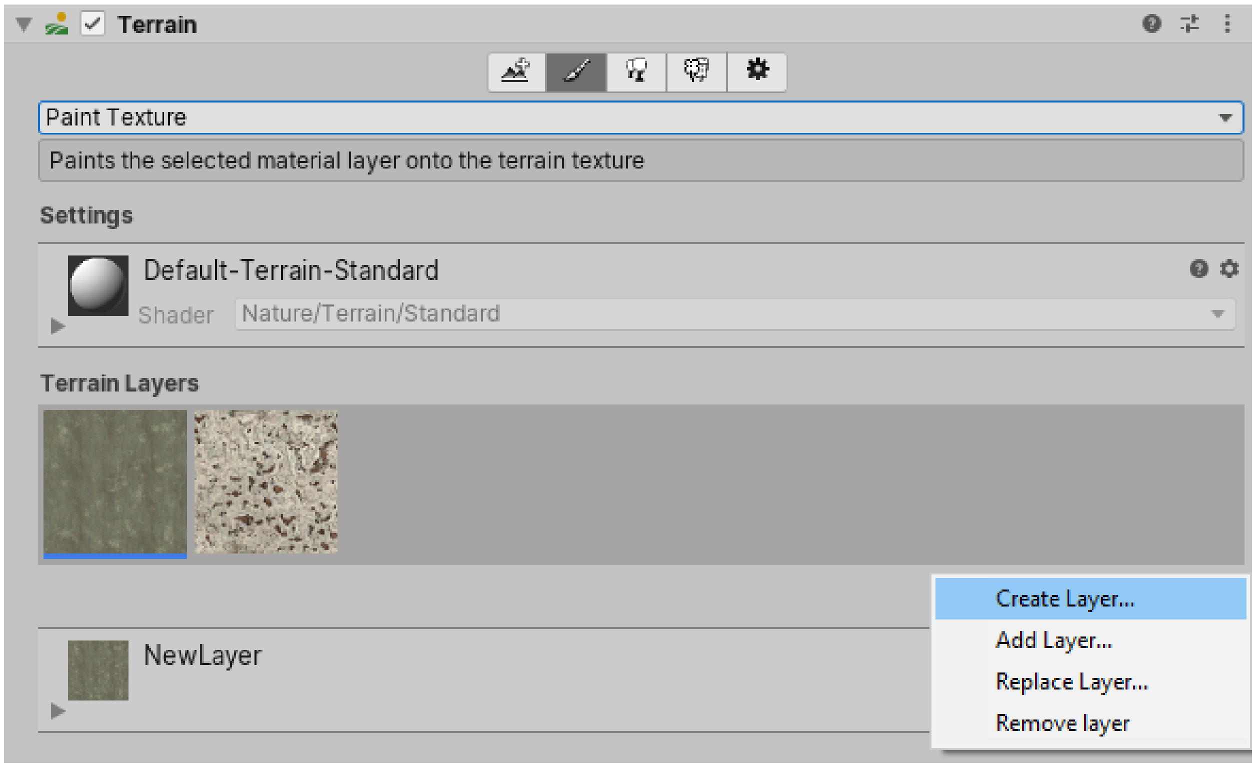


Figure 21 Terrain with sandy texture

Creating Terrain Layers

To create a Terrain Layer directly in the Terrain Inspector, click the paintbrush icon in the toolbar at the top of the Terrain Inspector, and select Paint Texture from the drop-down menu. At the bottom of the Terrain Layers section, click the Edit Terrain Layers button, and choose Create Layer.



Paints the selected material layer onto the terrain texture

Default-Terrain-Standard

Shader: Nature/Terrain/Standard

NewLayer

Figure 22 Create Layer in the Terrain Inspector

When you select Create Layer, Unity opens the Select Texture2D window. Here, choose the image to use as the Diffuse channel of the Terrain Layer. To assign a Normal Map or Mask Map Texture to your Terrain Layer, select the corresponding Terrain Layer in the Project view, and use its Inspector window.

Alternatively, to create a Terrain Layer Asset that isn't automatically associated with a Terrain, right-click the Project window, and select Create > Terrain Layer from the context menu. Then, configure the various properties in the Inspector window for your new Terrain Layer.

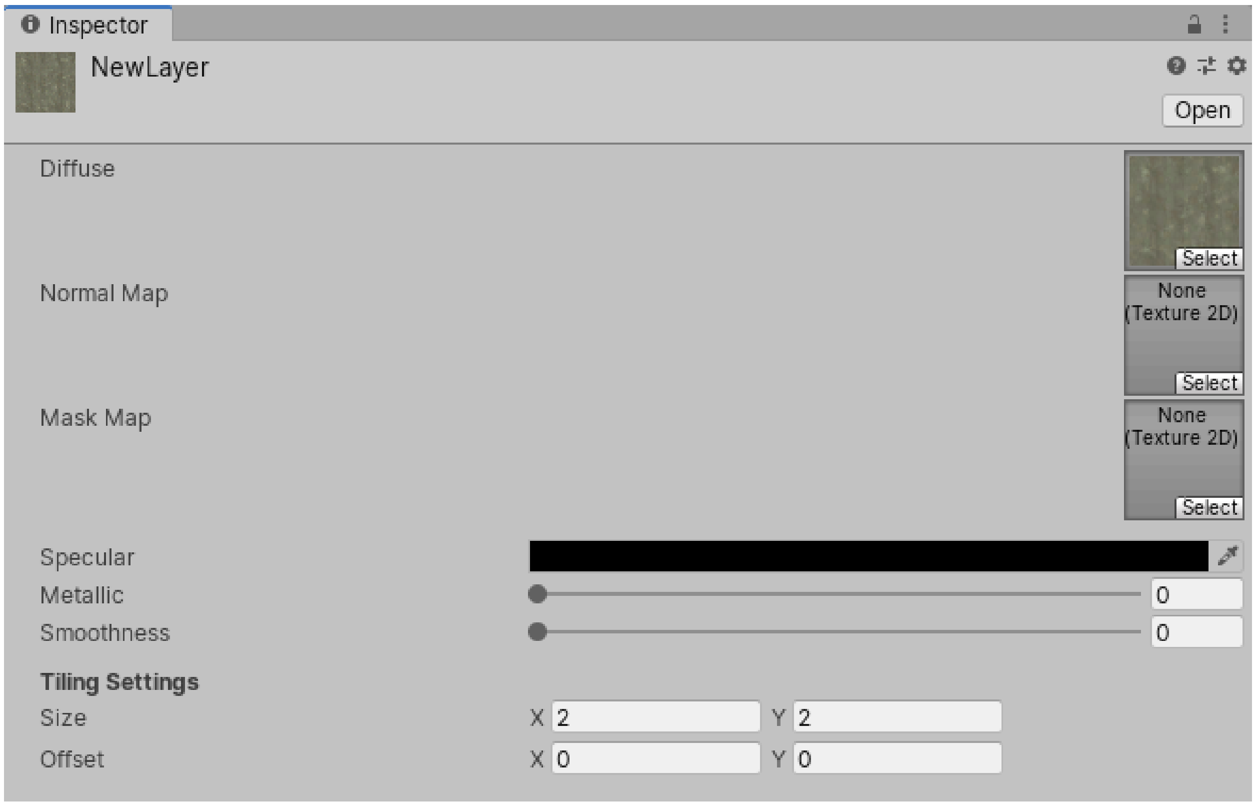
Adding Terrain Layers

Initially, a Terrain has no Terrain Layers assigned to it. By default, it uses a checkerboard Texture until you add a Terrain Layer.

After you create a Terrain Layer in your Project, click the Edit Terrain Layers button and select Add Layer to open the Select TerrainLayer window. Double-click on a Terrain Layer in this window to add it to your Terrain.

Terrain Layer properties

Depending on the Material that is set in the Terrain Settings, as well as the Render Pipeline that is currently in use, you might see different options and properties in the Inspector.


Inspector
NewLayer
Open

Diffuse — Select
Normal Map — None (Texture 2D) Select
Mask Map — None (Texture 2D) Select
Specular
Metallic 0
Smoothness 0
Tiling Settings
Size X 2 Y 2
Offset X 0 Y 0

Figure 23 Terrain Layer settings in the Inspector

Property	Description
Diffuse	The Diffuse Texture represents the base color Texture of the Terrain Layer. The Alpha channel of the Diffuse Texture has different uses, which depend on the active Scriptable Render Pipeline and Shader you use to render the Terrain. For example, the High Definition Render Pipeline (HDRP) uses the Alpha channel for Smoothness. However, if there is a Mask Map Texture on the Terrain Layer, it uses the Alpha channel of the Diffuse Texture for Density values.

Property	Description	
Normal Map	The Normal Map Texture contains the normal information for your Terrain Layer. Unity uses this information in lighting calculations. If you do not assign a Normal Map Texture and enable instancing in the Terrain Settings, the Terrain uses the normals generated from the Terrain heightmap. If you assign a Normal Map Texture and enable instancing, Unity uses the Normal Map Texture instead of the normals generated from the heightmap. If you disable instancing on the Terrain, the built-in Terrain Material uses normals generated from the Terrain geometry, even if you assign a Normal Map Texture on the Terrain Layer.	
Normal Scale	If you assign a Normal Map Texture, a new field called Normal Scale appears in the Terrain Layer settings. This value acts as a scaling factor for the normal values present in the Normal Map. A value of 0 means that the normals stored in the Normal Map have a scale of 0, while a value of 1 means that the normals are at full scale or influence. Examples and results of different Normal Scale values:	
	Normal Scale = 0	• Multiplies the unpacked normal value by 0. • The strength, and thus the length, of the normal will be 0, and has no effect on lighting calculations. The mesh triangle on the Terrain effectively uses the mesh normal for lighting calculations.
	Normal Scale = 1	• Multiplies the unpacked normal value by 1. • The strength of the normal will be 100%.
	Normal Scale = 2	• Multiplies the unpacked normal value by 2. • The strength of the normal will be 200%, and appear twice as pronounced as normals with a Normal Scale of 1.
	Normal Scale = –1	• Multiples the unpacked normal value by –1. • The strength of the normal will be at 100% but

Property		Description
		negated, making the normals point in the opposite direction from normals with a Normal Scale of 1.
Mask Map	The TerrainLit Shader, which is part of the High Definition Render Pipeline (HDRP) and Universal Render Pipeline (URP), uses this Mask Map Texture data. Custom Terrain shaders might also use this Texture for user-defined purposes, such as ambient occlusion or height-based blending. For the HDRP and URP TerrainLit Shader, the RGBA channels of the Mask Map Texture correspond to:	
	R	Metallic
	G	Ambient Occlusion
	B	Height
	A	Smoothness (Diffuse Alpha becomes Density)
Channel Remapping	If you assign a Mask Map Texture, a new heading called Channel Remapping appears in the Terrain Layer settings. Click the triangle next to that heading to display the fields for minimum and maximum RGBA values. Unity uses these ranges to remap values in each channel of the Mask Map Texture.	
Specular	The specular highlight color of the Terrain Layer.	
Metallic	The overall metallic value of the Terrain Layer.	
Smoothness	The overall smoothness value of the Terrain Layer.	
Tiling Settings	The tiling settings that apply to all Textures the Terrain Layer uses.	
	Size	The size of the Textures in Terrain space, and how often the Textures tile.

Property		Description
	Offset	A base offset that Unity applies to the sample location for each Texture in the Terrain Layer.

Texture painting

Unity applies the first Terrain Layer you add to the entire landscape. If you add multiple Terrain Layers, use the Paint Texture tool to apply subsequent Textures to your Terrain.

If you add a new Terrain tile without any Terrain Layers, and paint on it, the system automatically adds the selected Terrain Layer to that new Terrain tile. Because this is the first Terrain Layer, that Texture becomes the base layer, and fills the entire Terrain tile.

In the Terrain Inspector, under Brushes, there is a box that displays the available Brushes, along with the Brush Size and Opacity options underneath. See Creating and Editing Terrains for more information about these tools.

Rendering performance

The number of Terrain Layers you assign to a Terrain tile might impact the performance of the renderer. The maximum recommended number of Terrain Layers depends on which render pipeline your Project uses.

- If your Project uses the Universal Render Pipeline (URP) or Built-in Render Pipeline, you can use four Terrain Layers per Texture pass, with no limit on the number of passes. This means that although you are allowed to use as many Terrain Layers as you want, each pass increases the time spent rendering the Terrain. For maximum performance, limit each of your Terrain tiles to four Terrain Layers.
- If your Project uses the High Definition Render Pipeline (HDRP), you can add up to eight Terrain Layers per Terrain tile, and the system renders them in a single pass. No additional passes are possible. If you add more than eight Terrain Layers, they appear in the Unity Editor, but are ignored at run time.

Brushes

When you apply a tool such as Paint Texture or Smooth Height to the Terrain, Unity uses a Brush, which is a ScriptableObject in the Terrain system. The Brush defines the tool’s shape and strength of influence.

Built-in Brushes

Unity comes with a collection of built-in Brushes. They range from simple circles for quickly sketching designs, to more randomized scatter shapes that are good for creating detail and natural-looking features.

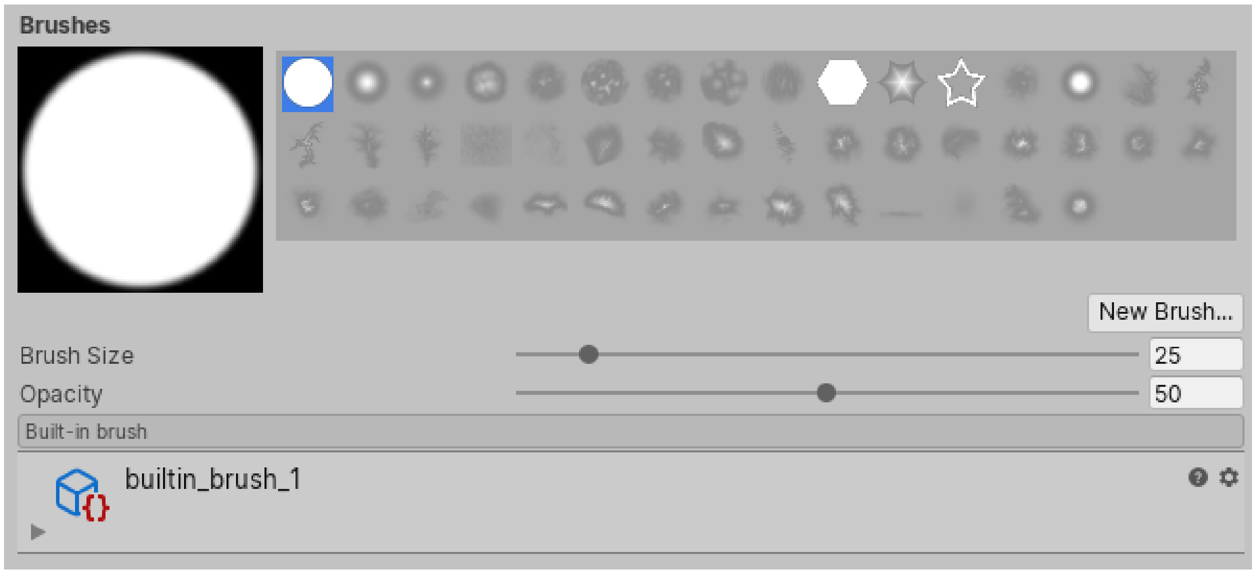


Figure 24 Built-in Brushes in the Terrain Inspector

Custom Brushes

You can create your own custom Brushes with unique shapes or specific parameters for your needs. For example, use the heightmap Texture of a specific geological feature to define a Brush, then use the Stamp Terrain tool to place that feature on your Terrain.

To create a new Brush, click the New Brush button in the Terrain Inspector window.

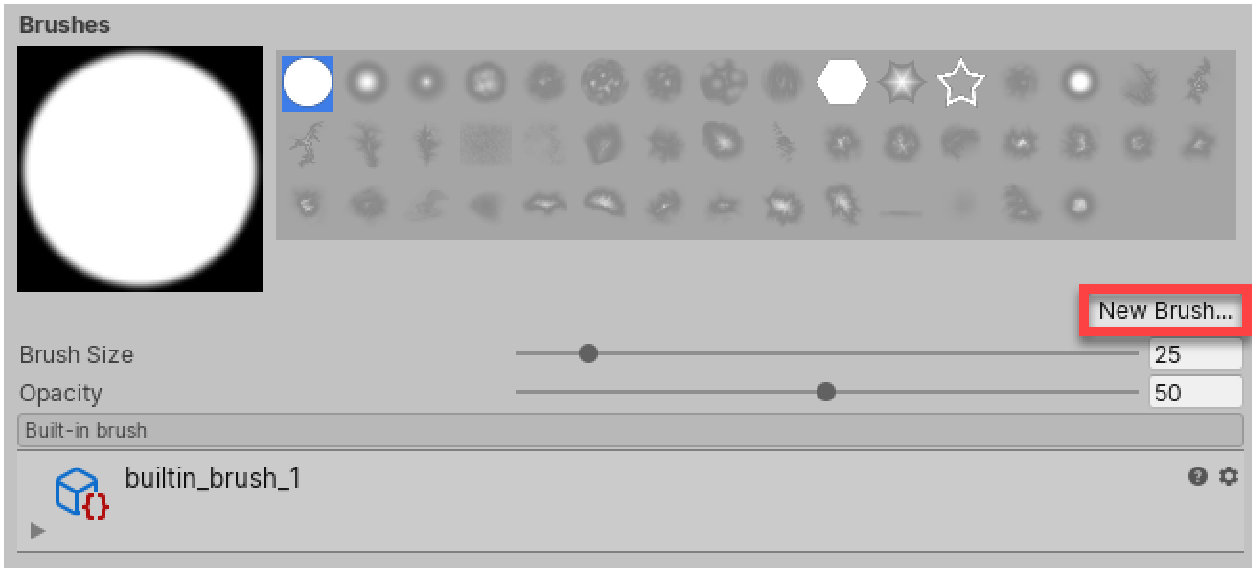


Figure 25 New Brush button in the Terrain Inspector

After you click New Brush, the Select Texture2D window appears. Choose a Texture to define the shape of your new Brush, then use the Brush Inspector to adjust the Falloff and Radius Scale values.

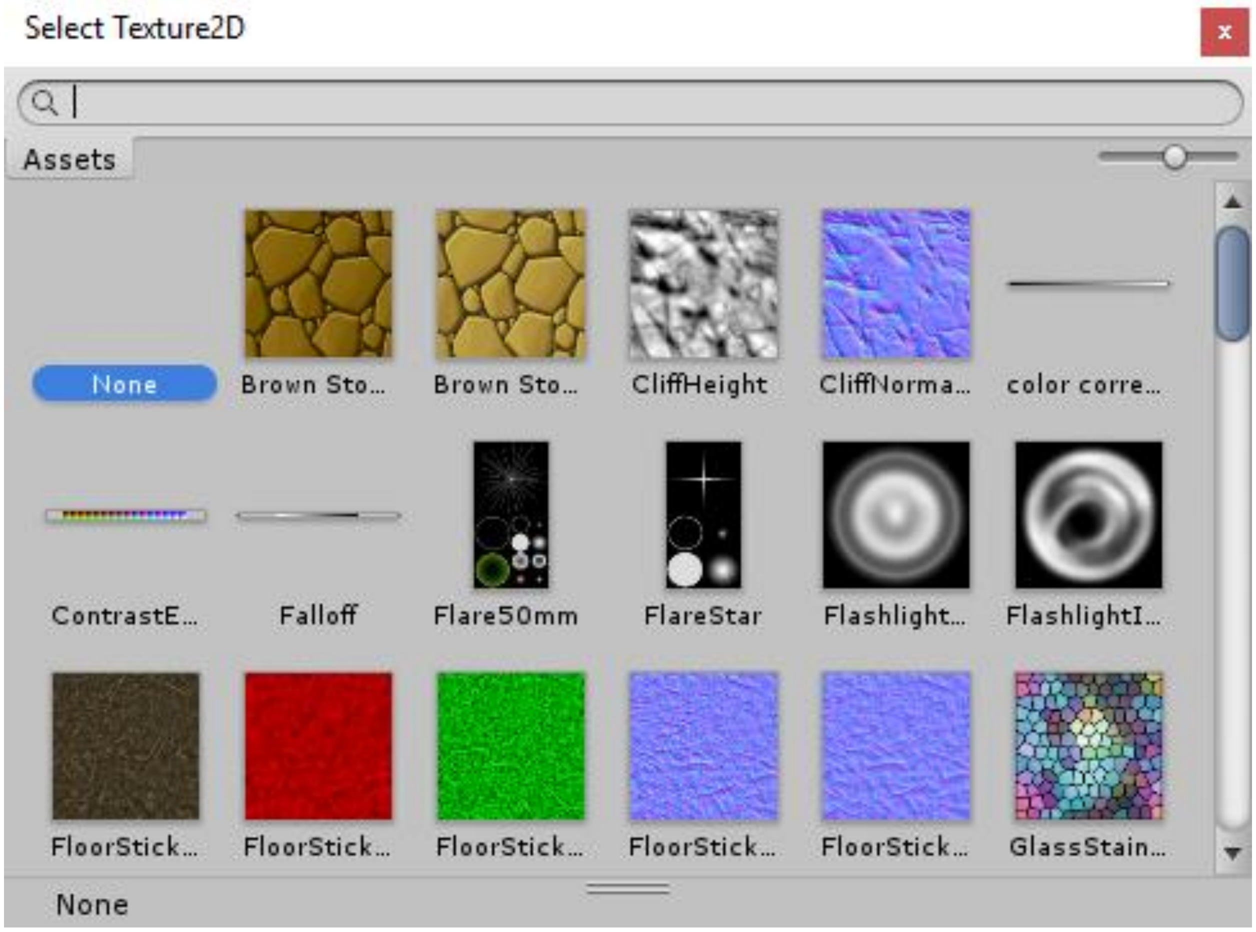
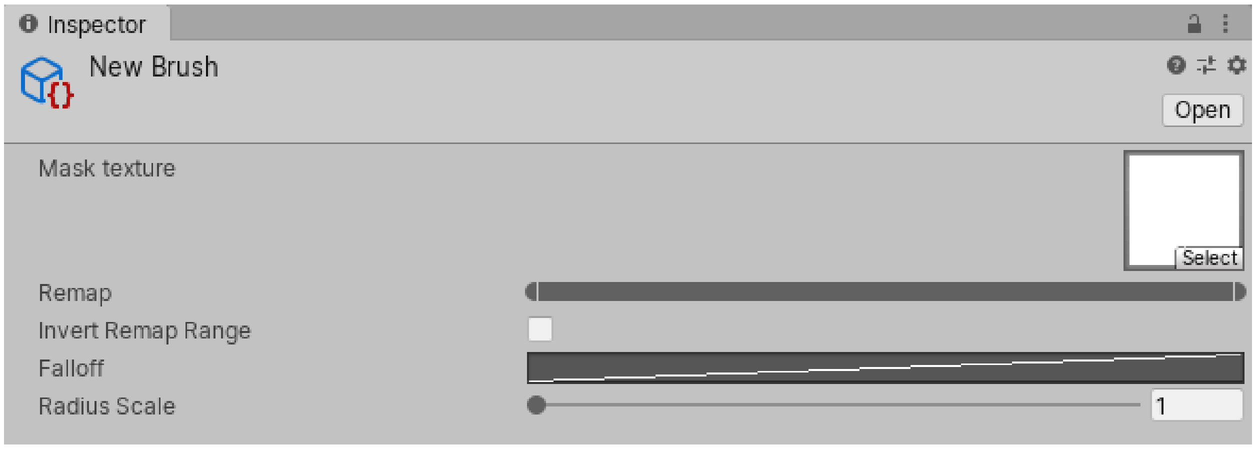


Figure 26 Window for selecting Brush Texture

Alternatively, right-click in the Project window, and choose Create > Brush to create a new Brush. The default Brush shows a simple circle defined by a white Mask Texture, a Falloff curve, and a Radius Scale of 1. Use the Brush Inspector to change these values, or set a Texture to define the shape of the Brush. You can also use the Remap slider and the Invert Remap Range option to further modify the grayscale values of the Brush Texture.


Inspector

New Brush

Open

Mask texture

Select

Remap

Invert Remap Range

Falloff

Radius Scale 1

Figure 27 Brush Inspector with default settings

Brush settings

Property	Function
Mask Texture	Defines the shape and strength of the Brush. Select a Texture in your project, and the system creates a grayscale mask from the Texture. If the selected Texture has multiple color channels, the Brush uses the Red channel as its source.
Remap	Remaps the grayscale values of the Brush mask, after applying the Falloff curve. The Editor remaps black values in the Brush mask to the value you select using the left side of the slider, and remaps white values in the Brush mask to the value you select using the right side of the slider.
Invert Remap Range	Inverts the left and right sides of the Remap slider, which basically inverts the values of the entire mask.
Falloff	Defines a curve that affects the strength of the Brush in a circular fashion. Click the Falloff curve to open the Unity Curve Editor, where you can edit the curve to create effects ranging from smooth fades to sharp edges.
Radius Scale	Affects the scale of the falloff curve. Use this option to increase or decrease the radius of the curve.

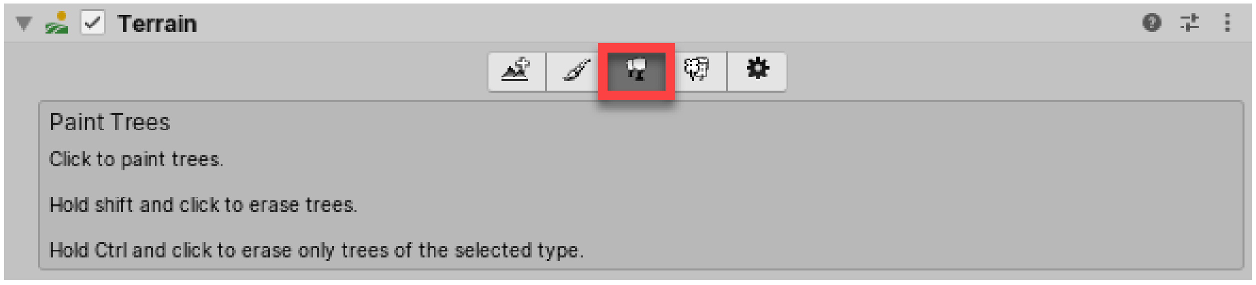
Trees

You can paint Trees onto a Terrain in a way that is similar to painting heightmaps and Textures. However, Trees are solid 3D objects that grow from the surface. Unity uses optimizations like billboarding for distant Trees to maintain good rendering performance. This means that you can have dense forests with thousands of Trees, and still keep an acceptable frame rate.

Figure 28 Terrain with Trees

Painting Trees

The Paint Trees button on the toolbar enables Tree painting:



Paint Trees

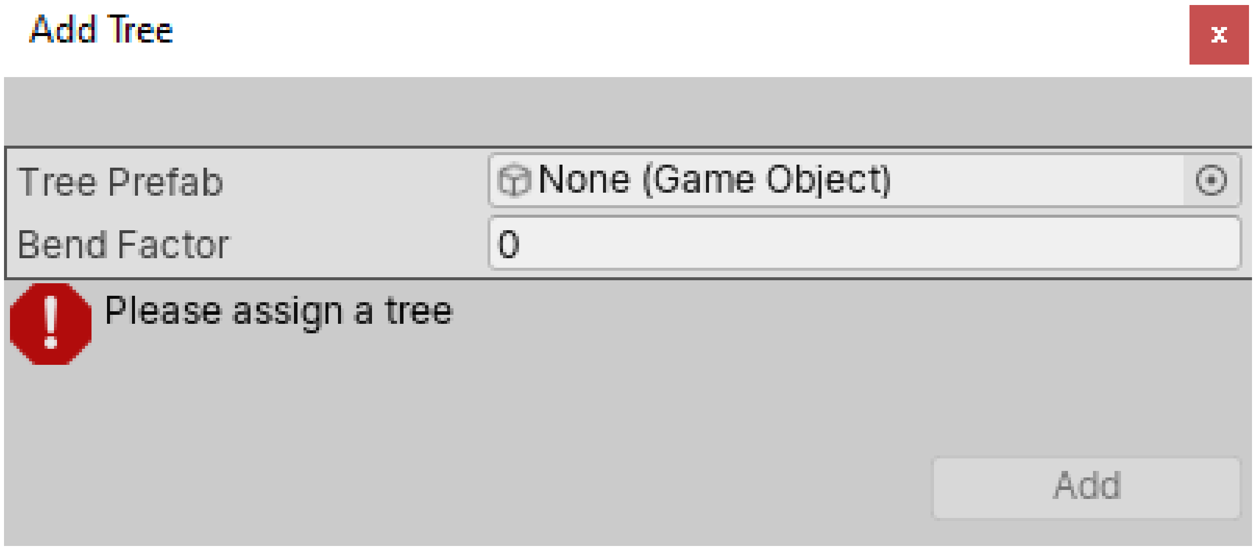
Click to paint trees.

Hold shift and click to erase trees.

Hold Ctrl and click to erase only trees of the selected type.

Figure 29 The Paint Trees button

Initially, the Terrain has no tree prototypes available. To start painting onto the Terrain, you need to add a tree prototype. Click the Edit Trees button, and select Add Tree. From here, you can select a Tree Asset from your Project, and add it as a Tree Prefab for use with the Brush:


Add Tree

Tree Prefab: None (Game Object)

Bend Factor: 0

Please assign a tree

Figure 30 The Add Tree window

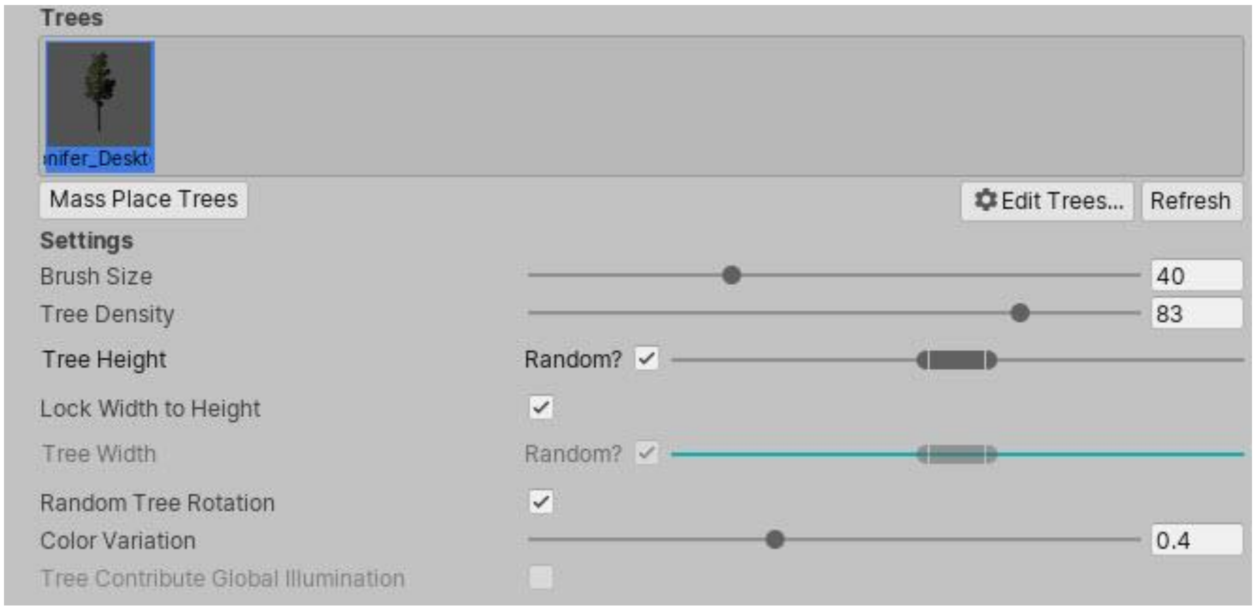
To help prototyping, SpeedTree provides four SpeedTree models in the free SpeedTrees Package on the Asset Store. Alternatively, you can create your own Trees.

If the Tree Prefab that you are importing supports Bend Factor, the Add Tree window displays a Bend Factor property for adjusting wind responsiveness. Trees created with the SpeedTree Modeler don't have a Bend Factor; only those created with Tree Editor do. See the section on Making Trees bend in the wind, below.

When you have configured your Settings (described below), you can paint Trees onto the Terrain in the same way you paint textures or heightmaps. To remove Trees from an area, hold the Shift key while you paint. To remove just the currently selected Tree type, hold down the Control key while you paint.

Settings

After you select which Tree to place, adjust its settings to customize Tree placement and characteristics.



Settings

Brush Size 40
Tree Density 83
Tree Height Random? ✓
Lock Width to Height ✓
Tree Width Random? ✓
Random Tree Rotation ✓
Color Variation 0.4
Tree Contribute Global Illumination

Figure 31 Settings for Trees

Property	Function
Mass Place Trees	Create an overall covering of Trees without painting over the whole landscape. After mass placement, you can still use painting to add or remove Trees to create denser or sparser areas.
Brush Size	Controls the size of the area that you can add Trees to.
Tree Density	Tree Density controls the average number of Trees painted onto the area defined by Brush Size.
Tree Height	Control the Tree's minimal height and maximal height using a slider. Drag the slider to the left for short Trees, and right for tall Trees. If you uncheck Random, you can specify the exact scale for the height of all newly painted Trees within the range of 0.01 to 2.
Lock Width to Height	By default, a Tree's width is locked to its height so that Trees are always scaled uniformly. However, you can disable the Lock Width to Height option, and specify the width separately.
Tree Width	If the Tree's width is not locked to its height, you can control the Tree's minimal width and maximal width using a slider. Drag the slider to the left for thin Trees, and right for wide Trees. If you

Property	Function
	uncheck Random, you can specify the exact scale for the width of all newly painted Trees within the range of 0.01 to 2.
Random Tree Rotation	If you configure the Tree with an LOD Group, use the Random Tree Rotation setting to help create the impression of a random, natural-looking forest rather than an artificial plantation of identical Trees. Uncheck this option if you want to place Trees with fixed, identical rotations.
Color Variation	The amount of random shading applied to Trees. This only works if your shader reads the _TreeInstanceColor property. For example, shaders for all trees you create with Tree Editor read the _TreeInstanceColor property.
Tree Contribute Global Illumination	Enable this check box to indicate to Unity that the Tree influences Global Illumination computations.

Creating Trees

There are two ways to create new Tree models. Use the SpeedTree Modeler from IDV, Inc. to create Trees with advanced visual effects, such as smooth LOD transition, fast billboarding, and natural wind animation. For more detailed information, refer to the SpeedTree Modeler documentation. Alternatively, use Unity's Tree Editor to create Tree models.

Internally, the Terrain Engine distinguishes between the two types of models by determining whether an LOD Group is present on the Tree Prefab's root GameObject. A SpeedTree Prefab has an LODGroup component, but a Tree Editor Prefab does not.

When creating Trees, position the anchor point at the base of the Tree where it emerges from the ground. Performance depends on the polygon count of your Tree model, so be sure to test on your platform, and create simpler Trees if necessary. Also, for Tree Editor Trees, each Mesh always has exactly two materials: one for the Tree body and the other for the leaves.

Trees you create using Tree Editor must use the Nature/Soft Occlusion leaves and Nature/Soft Occlusion Bark shader. To use those shaders, you have to place Trees in a specific folder named Ambient-Occlusion, otherwise the Trees don't render correctly.

When you place a model in this folder and re-import it, Unity calculates soft ambient occlusion in a way that is specifically designed for Trees.

If you change an imported Tree Asset in a separate 3D modelling application, you need to click the Refresh button in the Editor to see the updated Trees on your Terrain:

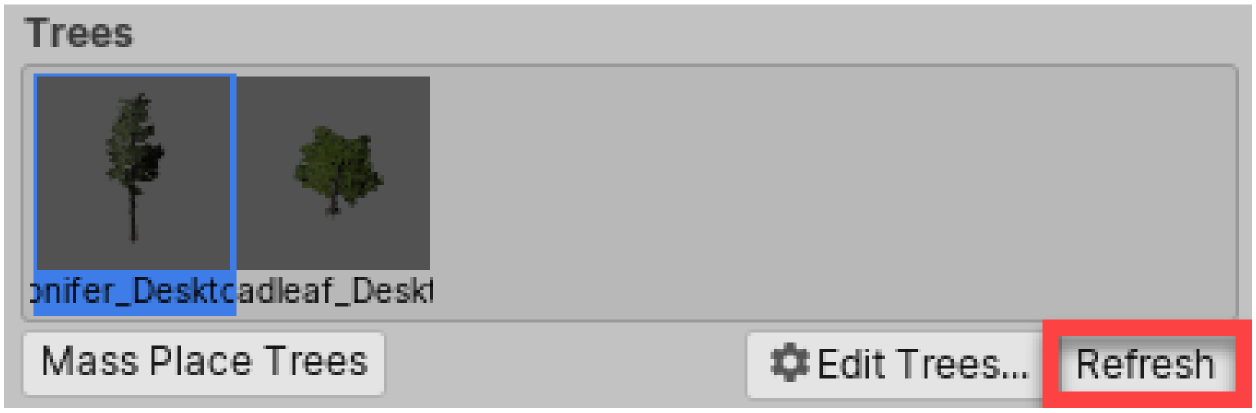


Figure 32 Refresh buton location

Note: When you import and alter a SpeedTree model in a 3D modeling program, then re-export it (as an .fbx or .obj), you might lose the natural wind animation functionality that comes with SpeedTree models.

Using Colliders with Trees

You can add a Capsule Collider to a Tree Asset. First, click > next to the Tree Asset to open the Prefab.

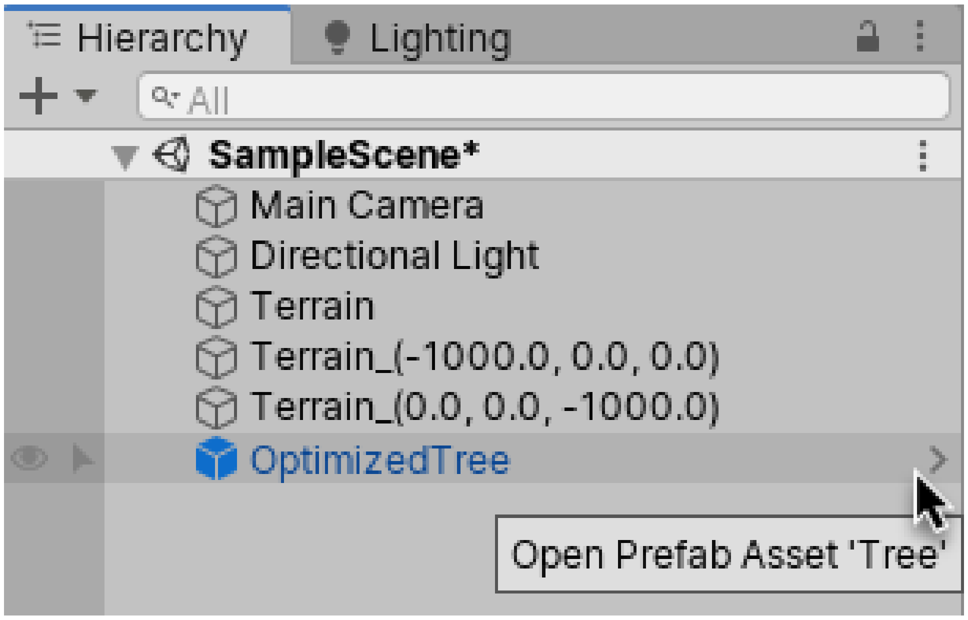
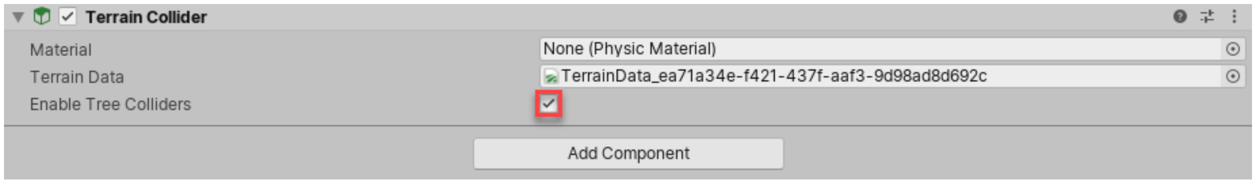


*Figure 33 Then, select **Add Component > Physics > Capsule Collider** to add the collider. To return to the Scene, click **<** next to the Prefab name.*

You must also check Enable Tree Colliders in the Terrain Collider component.


Terrain Collider

Material	None (Physic Material)
Terrain Data	TerrainData_ea71a34e-f421-437f-aaf3-9d98ad8d692c
Enable Tree Colliders	✓

Figure 34 Enable tree collider

Make Trees bend in the wind

Wind Zones can bend Trees to simulate the direct effect of wind. This feature is only available for Trees that you place through the Terrain menu. To do this, select the Terrain, click the Paint Trees button in the Inspector, then select Edit Trees > Add Tree and select your Tree Prefab. If you did not create the Tree in Unity, set the Bend Factor to 1. Click on the Terrain to place the selected Tree.

Terrain-based Trees that have a Bend Factor react to Wind Zones by bending and swaying, according to the values of the Wind Zone's Turbulence and Main properties. The Turbulence setting controls the fluttering of leaves. Reducing this value smooths the fluttering effect. The Main value controls the main force of the wind. To create a Wind Zone, select GameObject > 3D Object > Wind Zone.

If the Wind Zone's Mode is Directional, then its position relative to the Tree does not matter. The Trees will sway in the direction of the Wind Zone. However, if your Wind Zone's Mode is Spherical, then its Radius has to overlap the Tree's radius. This results in multi-directional gusts of wind, with a falloff from the center towards the edge.

With the default settings, Trees such as the Broadleaf Prefab (which is provided in the Standard Assets Environment pack) sway in a smooth and realistic way. However, if you're using your own Tree Prefab, you might need to adjust the Wind Zone property values. If your Tree Prefab trunk is bent all the way to one side, try reducing the Main value. Alternatively, if the branches of your Tree Prefab bend or stretch excessively, reduce the Turbulence value.

Tree Level of Detail (LOD) transition zone

For Tree Editor Trees, Unity's LOD system uses a 2D to 3D transition zone to seamlessly blend 2D billboards with 3D Tree models. This prevents any sudden popping of 2D and 3D Trees, which is vital in VR. Note that billboard Trees don't receive local lighting such as point lights and spot lights. They work with directional lights, but lighting on the billboards only updates when you rotate the Camera.

SpeedTree

Figure 35 SpeedTree is a third-party product by IDV Inc, which provides prebuilt tree Assets and modeling software focused specifically on trees.

Unity recognizes and imports SpeedTree Assets in the same way that it handles other Assets. If you're using SpeedTree Modeler 7, make sure to resave your .spm files using the Unity version of the Modeler. If you're using SpeedTree Modeler 8, save your .st files directly into the Unity Project folder.

Make sure that Textures are reachable in the Project folder, and that Unity automatically generates Materials for each Level of Detail (LOD). When you select an Asset, there are import settings to tweak the generated GameObject and attached Materials. Unity does not re-generate Materials when you re-import them, unless you click the Generate Materials or Apply & Generate Materials button. Therefore, it is possible to preserve any customizations to the Materials.

The SpeedTree importer generates a Prefab with the LODGroup component configured. You can instantiate the Prefab in a Scene as a common Prefab instance, or select the Prefab as a tree prototype and paint it across the Terrain.

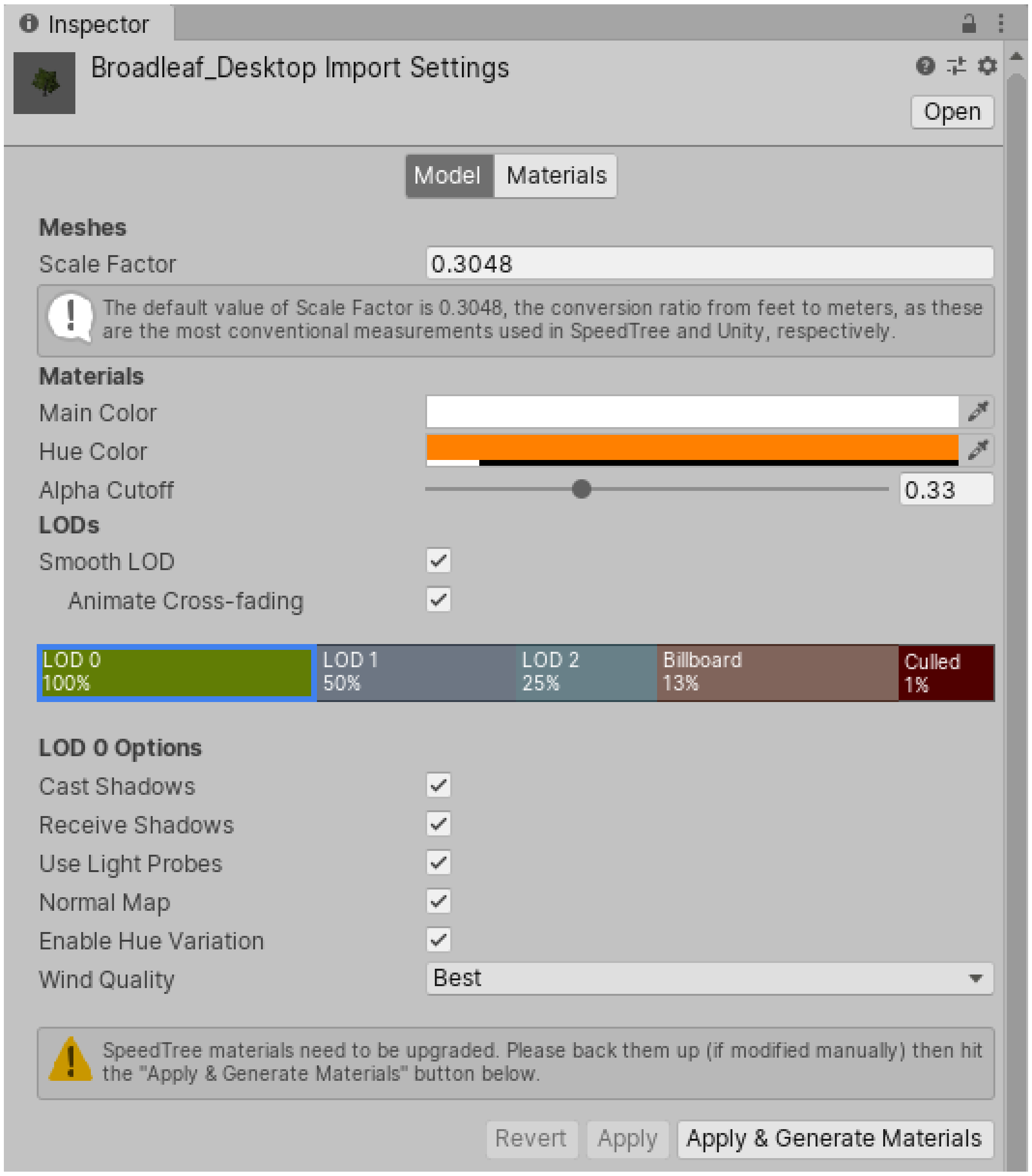
Additionally, the Terrain accepts any GameObject with an LODGroup component as a tree prototype, and does not place limitations on the Mesh size, or number of Materials used. This is different from Tree Editor trees. However, be aware that SpeedTree trees usually use three to four different Materials, which as a result, issues a number of draw calls every frame. Thus, we recommend that you avoid heavy use of LOD trees on platforms, such as the mobile platforms, where additional draw calls are relatively costly in terms of rendering performance.

Materials

The SpeedTree Model Importer has a Materials tab to improve the workflow for handling SpeedTree Material Assets. See documentation on the SpeedTree Model Importer's Material tab page for more information.

Casting and receiving real-time shadows

To make billboards cast shadows correctly, Unity rotates billboards during the shadow caster pass to make them face the light direction (or light position in the case of point light) instead of facing the camera.



Broadleaf_Desktop Import Settings

Meshes

Scale Factor: 0.3048

The default value of Scale Factor is 0.3048, the conversion ratio from feet to meters, as these are the most conventional measurements used in SpeedTree and Unity, respectively.

Materials

Main Color

Hue Color

Alpha Cutoff: 0.33

LODs

Smooth LOD ✓

Animate Cross-fading ✓

LOD 0 100% | LOD 1 50% | LOD 2 25% | Billboard 13% | Culled 1%

LOD 0 Options

Cast Shadows ✓

Receive Shadows ✓

Use Light Probes ✓

Normal Map ✓

Enable Hue Variation ✓

Wind Quality: Best

SpeedTree materials need to be upgraded. Please back them up (if modified manually) then hit the "Apply & Generate Materials" button below.

Figure 36 SpeedTree import settings

To enable these options, select the Billboard LOD level in the Inspector of a SpeedTree Asset, check Cast Shadows or Receive Shadows in Billboard Options, and click Apply Prefab.

To change the billboard shadow options of instantiated SpeedTree GameObjects, select the billboard GameObject in the Hierarchy window and tweak these options in the Inspector of the Billboard Renderer.

The trees you paint on a Terrain inherit billboard shadow options from the Prefab. Use BillboardRenderer.shadowCastingMode and BillboardRenderer.receiveShadows to alter these options at runtime.

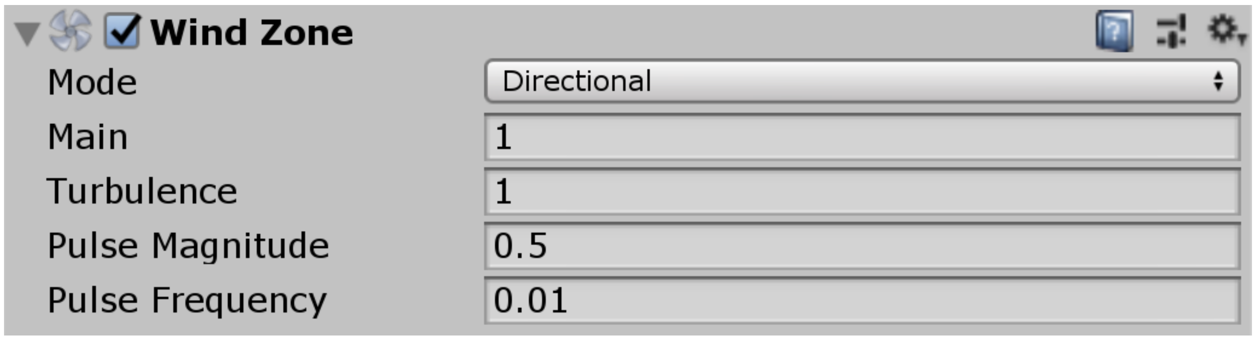
Known issues: As with any other renderer, the Receive Shadows option has no effect while using deferred rendering. Billboards always receive shadows in deferred path.

Wind Zones

SWITCH TO SCRIPTING To create the effect of wind on your Terrain and Particle Systems, you can add one or more GameObjects with Wind Zone components. Trees within a wind zone bend in a realistic animated fashion, and the wind itself moves in pulses to create natural patterns of movement among the trees.

Using Wind Zones

To create a Wind Zone GameObject directly, go to Unity's top menu and go to GameObject > 3D Object > Wind Zone. You can add the Wind Zone component to any suitable GameObject already in the Scene (menu: Component > Miscellaneous > Wind Zone). The Inspector for the Wind Zone has a number of settings to control its behavior.


Wind Zone

Mode	Directional
Main	1
Turbulence	1
Pulse Magnitude	0.5
Pulse Frequency	0.01

Figure 37 Wind Zone inspector

For details on all of the Wind Zone settings, see documentation on the Wind Zone component.

You can set the Mode to Directional or Spherical.

- In Directional mode, the wind affects the whole Terrain at once. This is useful for creating effects like the natural movement of the trees.

- In Spherical mode, the wind blows outwards within a sphere defined by the Radius property. This is useful for creating special effects like explosions.

The Main property determines the overall strength of the wind, and you can use Turbulence to give these some random variation.

The wind blows over the trees in pulses to create a more natural effect. You can control the strength of the pulses with Pulse Magnitude, and the time interval between them with Pulse Frequency.

Properties

Property:	Function:
Mode	
Spherical	Wind zone only has an effect inside the radius, and has a falloff from the center towards the edge.
Directional	Wind zone affects the entire scene in one direction.
Radius	Radius of the Spherical Wind Zone (only active if the mode is set to Spherical).
Main	The primary force applied to all objects affected by the Wind Zone.
Turbulence	The value represents the noise of the wind. A greater value creates higher variation in wind direction.
Pulse Magnitude	Defines the strength multiplier of the wind pulses.
Pulse Frequency	Defines the length and frequency of the wind pulses.

Particle Systems

The main use of wind is to animate trees, but it can also affect particles in a Particle System if that Particle System uses the External Forces module module. See the Particle System reference page for further details.

Grass and other details

A Terrain might have grass clumps and other small objects (such as rocks) covering its surface. Unity renders these objects using textured quads or full Meshes, depending on the level of detail and performance you require. Grass and other details currently only work in the built-in render pipeline and Universal Render Pipeline (URP).

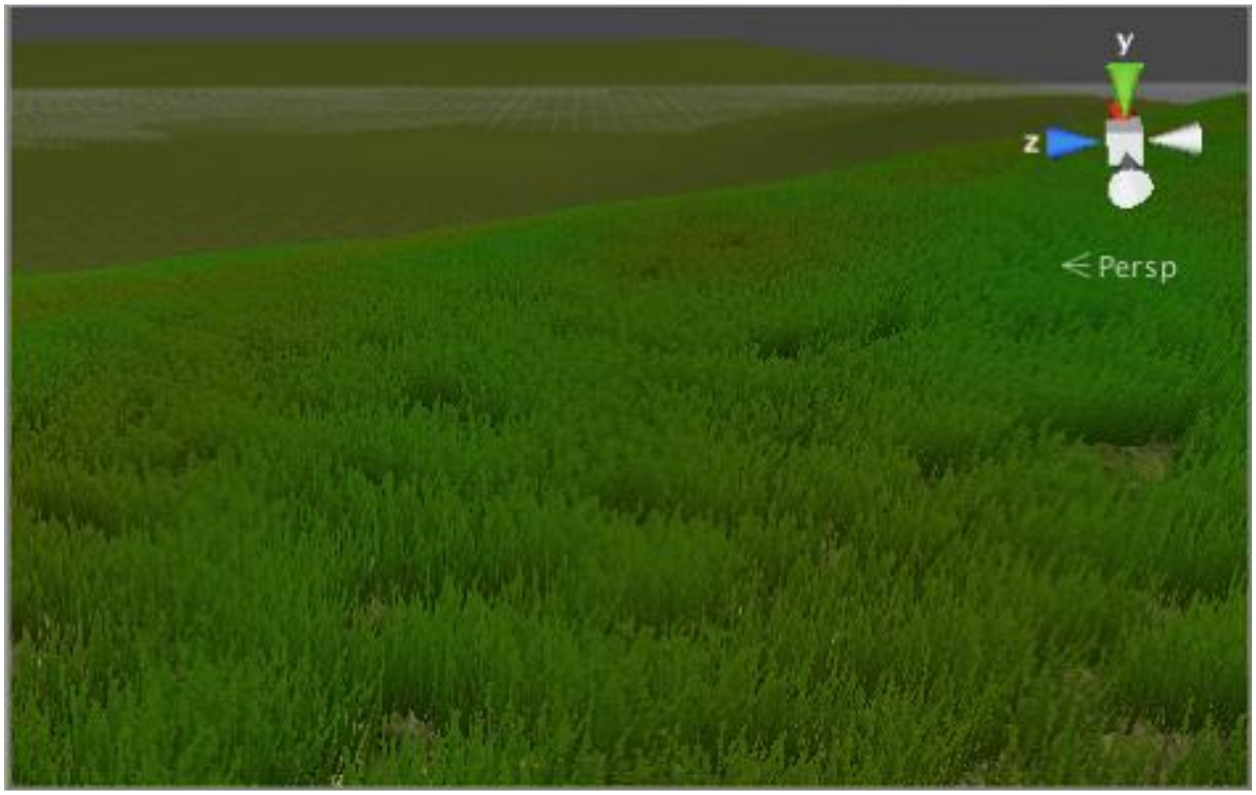


Figure 38 Terrain with grass

You can enable the Billboard property of a textured quad so that it automatically faces the camera. This is a very common way to achieve a good grass effect in game development.

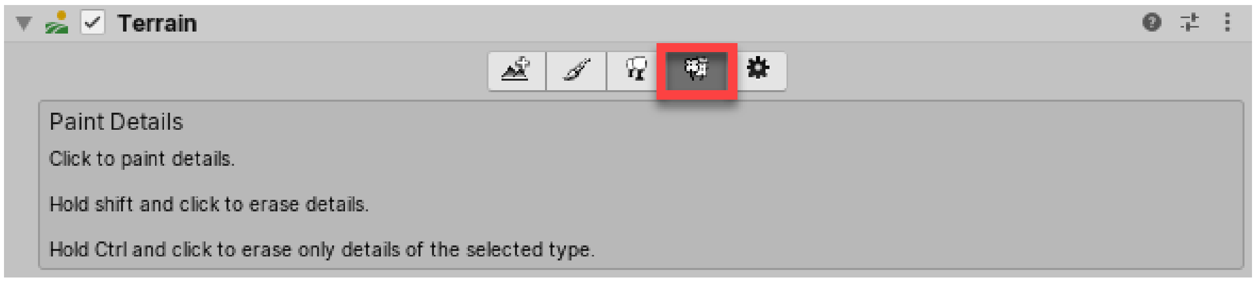
For detail Meshes, you can set the Render Mode property as either Vertex Lit or Grass.

- Select Vertex Lit to create Meshes that are vertex lit with real normals, which do not move in the wind.
- Select Grass to create Meshes that are vertex lit using Terrain normals, which do move in the wind.

Enabling details

To enable grass and detail painting, select the Paint Details button on the toolbar

.



Paint Details

Click to paint details.

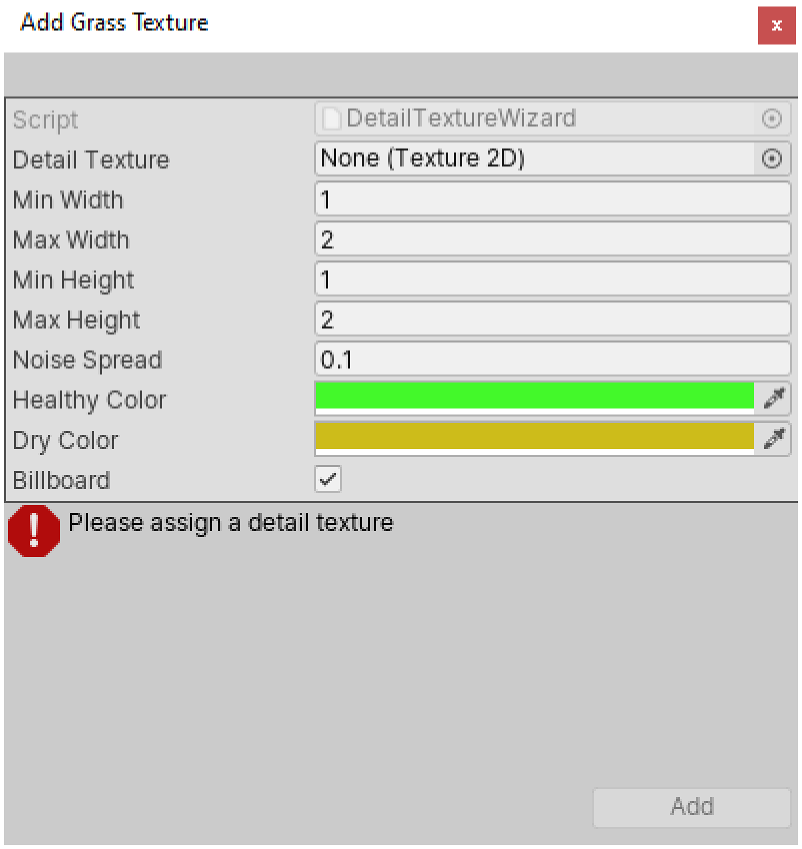
Hold shift and click to erase details.

Hold Ctrl and click to erase only details of the selected type.

Figure 39 Paint Details in the Terrain Inspector

Initially, a Terrain has no grass or details available. In the Inspector, click the Edit Details button to display a menu with the Add Grass Texture and Add Detail Mesh options. Click either option to bring up a window that lets you choose Assets to add to the Terrain for painting.

For grass, the window looks like this: -



Script	DetailTextureWizard
Detail Texture	None (Texture 2D)
Min Width	1
Max Width	2
Min Height	1
Max Height	2
Noise Spread	0.1
Healthy Color	🟩
Dry Color	🟨
Billboard	✓

Please assign a detail texture

Figure 40 The Add Grass Texture window

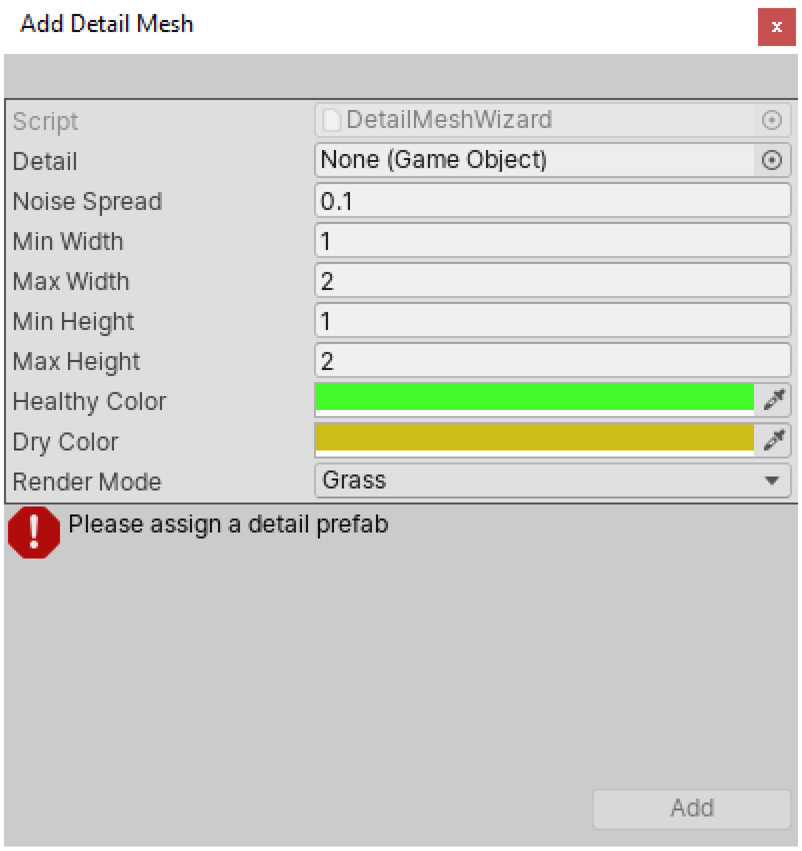
Detail Texture is the Texture that represents the grass. You can download Textures from the Asset Store, or create your own Textures. A Texture is a small image with alpha set to zero for the empty areas. Note that "Grass" is a generic term; it is possible for a Texture to represent flowers or man-made objects such as barbed wire coils.

The Min Width, Max Width, Min Height, and Max Height values specify the upper and lower size limits grass clumps that are generated. To create an authentic look, the grass is generated in random "noisy" patterns with bare patches interspersed.

The Noise Spread value controls the approximate size of the bare and grassy patches, with higher values indicating more variation within a given area. Unity uses the Perlin noise algorithm to generate noise, and Noise Spread refers to the scaling it applies between the x,y position on the Terrain and the noise image. The alternating patches of grass are considered "healthier" at the centers than at the edges, and colors set in the Healthy Color and Dry Color settings represent the health of the grass.

Finally, when you enable the Billboard option, the grass images rotate so that they always face the Camera. This is useful when you want to show a dense field of grass because clumps are two-dimensional, and not visible from the side. However, with sparse grass, the rotations of individual clumps might become apparent to the viewer, creating a strange effect.

For detail meshes such as rocks, the window looks like this: -



Script	DetailMeshWizard
Detail	None (Game Object)
Noise Spread	0.1
Min Width	1
Max Width	2
Min Height	1
Max Height	2
Healthy Color	🟩
Dry Color	🟨
Render Mode	Grass

Please assign a detail prefab

Figure 41 The Add Detail Mesh window

Use Add Detail Mesh to select a Prefab from your Project. Unity scales this randomly between the Min Width and Max Width values, and the Min Height and Max Height values. Unity uses width scaling for both the x and z axes, and height scaling for the y axis. The Noise Spread, Healthy Color and Dry Color values work the same as they do for grass.

You can set the Render Mode to Vertex Lit or Grass.

- In Vertex Lit mode, Unity renders detail objects as solid, vertex-lit GameObjects in the Scene.
- In Grass mode, Unity renders instances of detail objects in the Scene with lighting, in a way similar to grass.

Working with Heightmaps

Terrain tools that affect height, such as Raise or Lower Terrain and Set Height, use a grayscale texture called a heightmap. Unity represents the height of each point on the Terrain as a value in a rectangular array. It represents this array using a grayscale heightmap. Heightmaps are built into the Terrain, and the values stored in a heightmap define the height of each point or vertex on the Terrain.

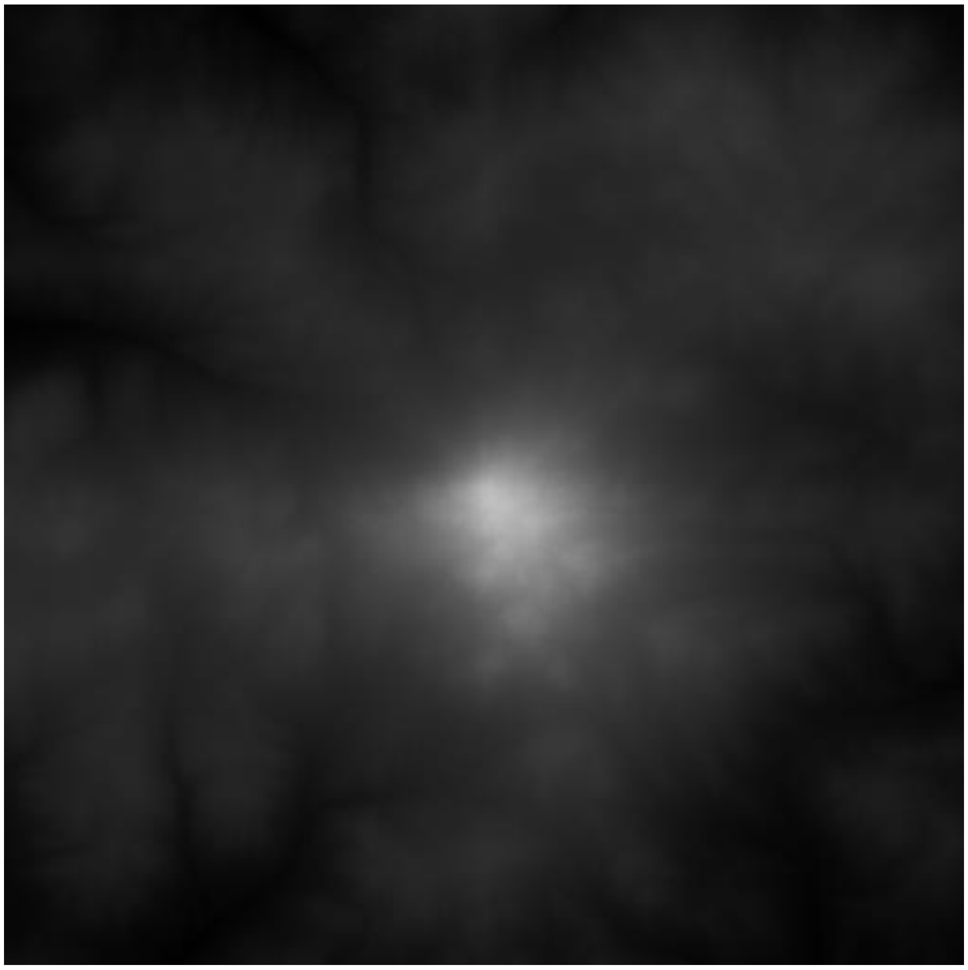


Figure 42 Example heightmap

Importing and exporting heightmaps

You can import and export heightmaps into the Unity Editor. This is useful when you want to use real world height data to replicate a landmark such as Mount Everest, or work on a heightmap image in an external editor like Photoshop. You can also use 3D modelling applications, such as Houdini and World Machine, to generate Terrain, then import your Terrain into Unity as a heightmap.

It's good practice to store heightmaps as RAW files. A RAW file uses a 16-bit grayscale format that is compatible with most image and landscape editors. The Unity Editor enables you to import and export RAW heightmap files for a Terrain.

To access the import and export settings into the Editor, select the Terrain component in the Inspector, and click the Terrain Settings button (gear icon in the toolbar).

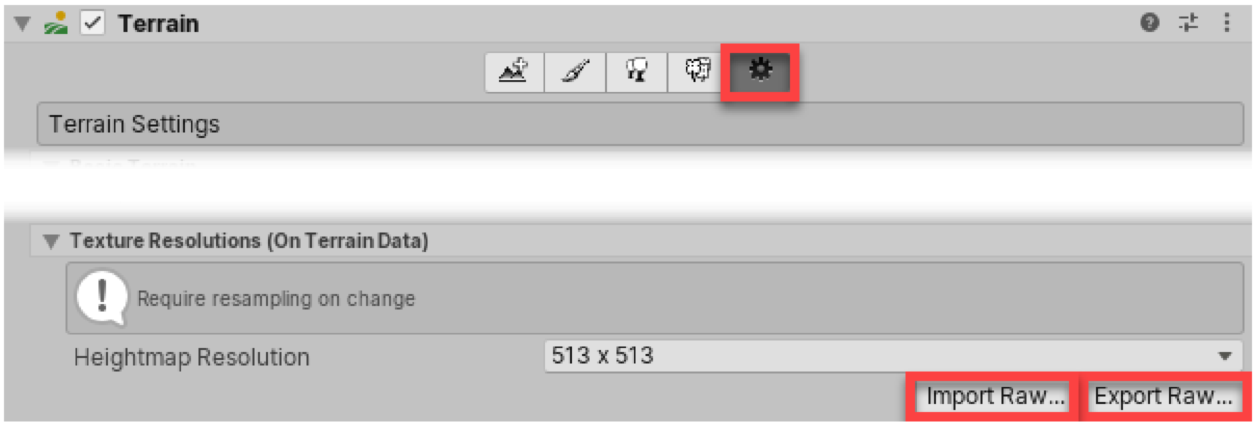
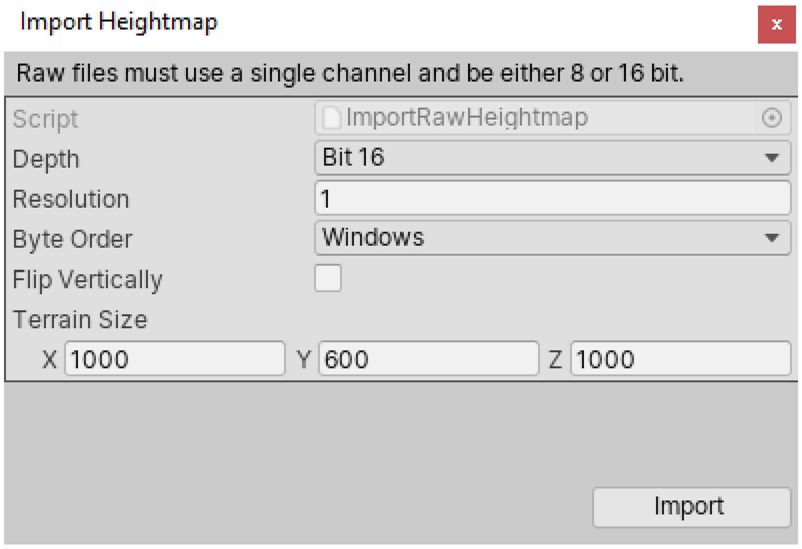

Terrain
Terrain Settings
Texture Resolutions (On Terrain Data)
Require resampling on change
Import Raw...
Export Raw...
Heightmap Resolution: 513 x 513

Figure 43 Import Raw and Export Raw buttons in the Terrain Settings Inspector

Under Texture Resolutions (On Terrain Data), there are two buttons labelled Import Raw and Export Raw.

Import Raw allows Unity to read a heightmap from the RAW file format, and generate it in the Editor.

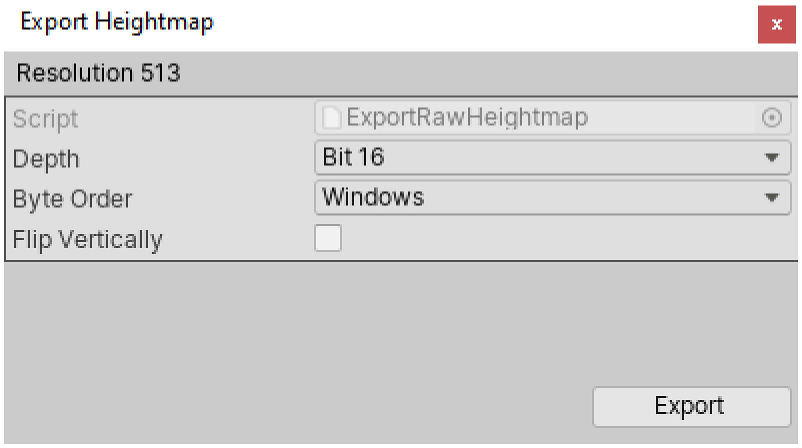

Import Heightmap

Raw files must use a single channel and be either 8 or 16 bit.

Script	ImportRawHeightmap
Depth	Bit 16
Resolution	1
Byte Order	Windows
Flip Vertically	☐
Terrain Size	X 1000 Y 600 Z 1000

Figure 44 Import Height map

Export Raw allows Unity to write a heightmap from the Editor to the RAW file format.


Export Heightmap

Resolution 513

Script	ExportRawHeightmap
Depth	Bit 16
Byte Order	Windows
Flip Vertically	☐

Figure 45 Export Height Map

Import and export options

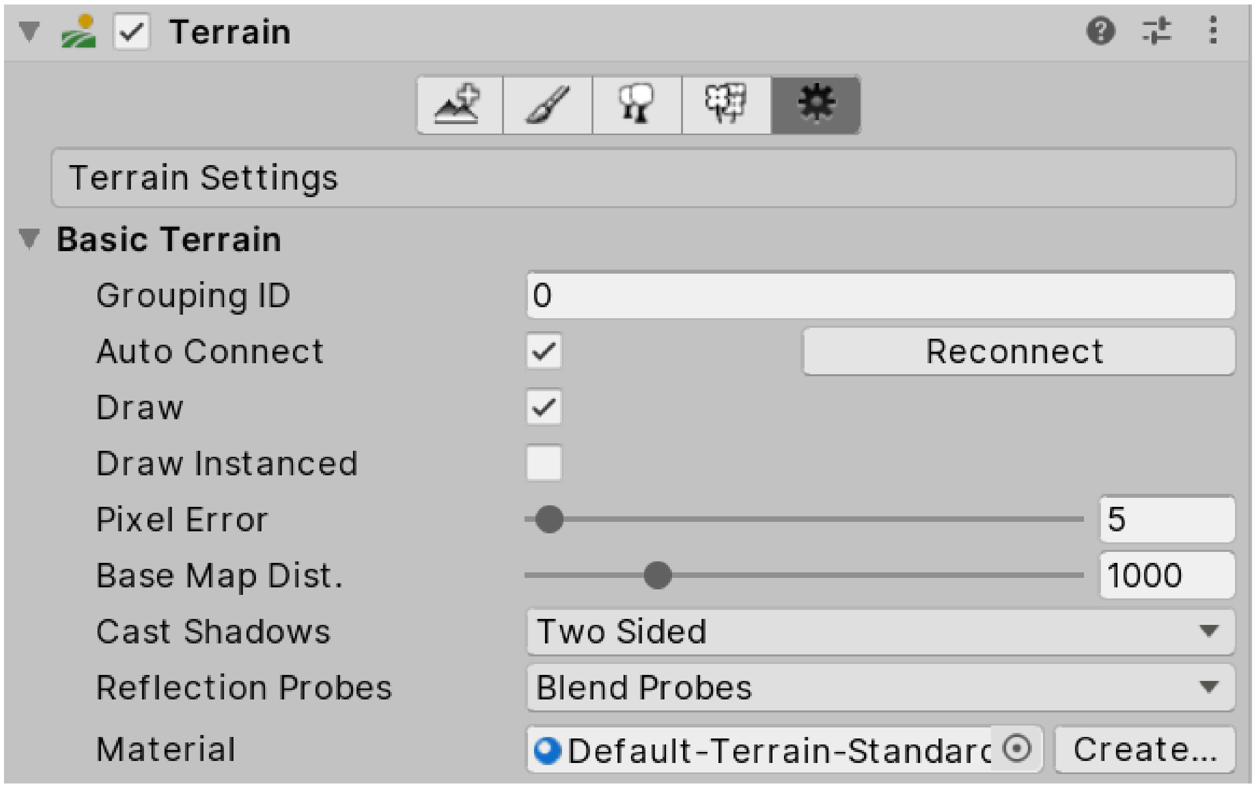
Property	Description
Depth	Determines how many bits Unity uses per pixel in the imported or exported heightmap. • Bit 16: Uses 16 bits (2 bytes) • Bit 8: Uses 8 bits (1 byte)
Resolution	The texture resolution (width and height) of the imported heightmap.

Property	Description
Byte Order	Determines how Unity orders the bytes for each pixel in the imported or exported heightmap. This mainly applies to bit–16 depth heightmaps, and is platform-dependent.
Flip Vertically	Determines whether Unity flips the exported heightmap vertically across the x-axis.
Terrain Size	The size of Terrain that Unity will apply the imported heightmap to.

Terrain settings

The final tool on the five-icon toolbar is for settings. In the Inspector, click the gear icon to reveal the Terrain settings.

Basic Terrain



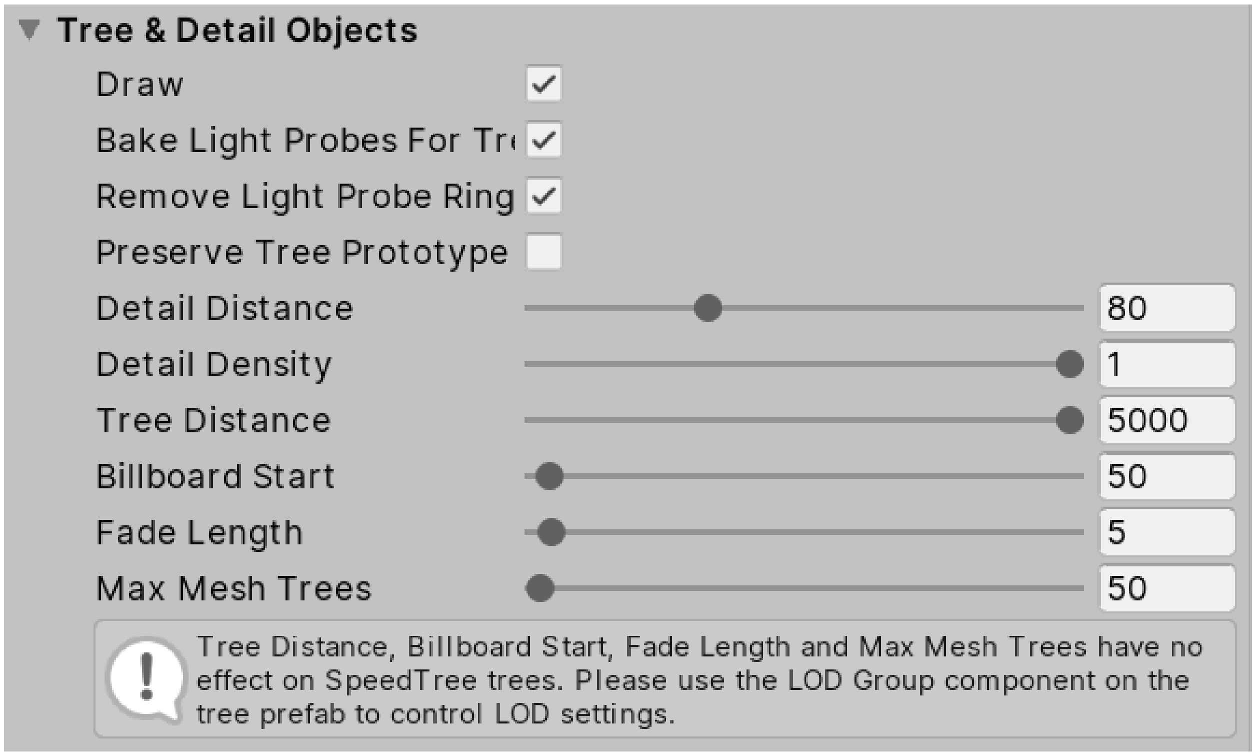
Grouping ID: 0
Auto Connect: ✓
Draw: ✓
Draw Instanced: ☐
Pixel Error: 5
Base Map Dist.: 1000
Cast Shadows: Two Sided
Reflection Probes: Blend Probes
Material: Default-Terrain-Standard

Figure 46 Terrain setting

Property	Function
Grouping ID	The grouping ID for the Auto connect feature.
Auto Connect	Check this box to automatically connect the current Terrain tile to to neighboring tiles sharing the same Grouping ID.
Reconnect	On rare occasions, you might lose connections between Terrain tiles if you change the Grouping ID, or disable Auto connect for one or more tiles. To recreate connections between tiles, click the Reconnect button. Reconnect only connects two adjacent tiles if they have the same Grouping ID and if both tiles have Auto Connect enabled.
Draw	Check this box to enable rendering of the Terrain.
Draw Instanced	Check this box to enable instanced rendering.
Pixel Error	The accuracy of the mapping between Terrain maps (such as heightmaps and Textures) and generated Terrain. Higher values indicate lower accuracy, but with lower rendering overhead.
Base Map Dist.	The maximum distance at which Unity displays Terrain Textures at full resolution. Beyond this distance, the system uses a lower resolution composite image for efficiency.
Cast Shadows	Use this to define how the Terrain casts shadows onto other objects in the Scene. Rendering.ShadowCastingMode controls how the Terrain shadow interacts with Scene objects.
Off	The Terrain does not cast shadows.
On	The Terrain casts shadows.
Two Sided	Casts two-sided shadows from either side of the Terrain. Enlighten and Progressive Lightmapper do not support two-sided shadows.
Shadows Only	Shadows from the Terrain are visible, but the Terrain itself is not.

Property		Function
Reflection Probes		Use this to set how Unity uses Reflection Probes on Terrain. This setting only has an effect when Material is set to Built In Standard, or if you use a custom Material (with Material set to Custom) that supports rendering with reflection.
	Off	Disables Reflection Probes, and uses a skybox for reflection.
	Blend Probes	Enables Reflection Probes. Blending occurs only between probes. Uses default reflection if there are no Reflection Probes nearby, but no blending between default reflection and probe.
	Blend Probes And Skybox	Enables Reflection probes. Blending occurs between probes, or between probes and default reflection.
	Simple	Enables Reflection probes, but no blending occurs between probes when there are two overlapping volumes.
Material		Lets you specify the Material to use for rendering Terrain.
	Create	The Create button only appears when you select a default Terrain Material. It does not appear when you select a custom Material. When you click Create, Unity creates a copy of the Material in your Project folder, which you can modify, and then automatically selects this new copy.

Tree and Detail Objects



Tree & Detail Objects

Draw ✓

Bake Light Probes For Tre ✓

Remove Light Probe Ring ✓

Preserve Tree Prototype

Detail Distance 80

Detail Density 1

Tree Distance 5000

Billboard Start 50

Fade Length 5

Max Mesh Trees 50

Tree Distance, Billboard Start, Fade Length and Max Mesh Trees have no effect on SpeedTree trees. Please use the LOD Group component on the tree prefab to control LOD settings.

Figure 47 Tree and Detail Objects

Property	Function
Draw	Check this box to draw trees, grass, and details.
Bake Light Probes For Trees	If you check this box, Unity creates internal Light Probes at the position of each tree, and applies them to tree renderers for lighting. These probes are internal, and don't affect other renderers in the Scene. If you don't check this box, trees are still affected by Light Probe Groups. This option is only effective for trees with Light Probe enabled on their prototype Prefab.
Remove Light Probe Ringing	If you check this box, Unity removes visible overshooting, which often appears as ringing on GameObjects that are affected by intense lighting. This setting reduces contrast, and is dependent on Bake Light Probes for Trees. For more information, see Light Probe Groups: Ringing.

Property	Function
Preserve Tree Prototype Layers	Check this box if you want your tree instances to take on the layer values of their prototype Prefabs, instead of the Terrain GameObject's layer values.
Detail Distance	The distance from the camera beyond which details are culled.
Detail Density	The number of detail/grass objects in a given unit of area. Set this value lower to reduce rendering overhead.
Tree Distance	The distance from the camera beyond which trees are culled.
Billboard Start	The distance from the camera at which Billboard images replace 3D Tree objects.
Fade Length	The distance over which Trees transition between 3D objects and Billboards.
Max Mesh Trees	The maximum number of visible Trees that are represented as solid 3D meshes. Beyond this limit, Billboards replace Trees.

Wind Settings for Grass

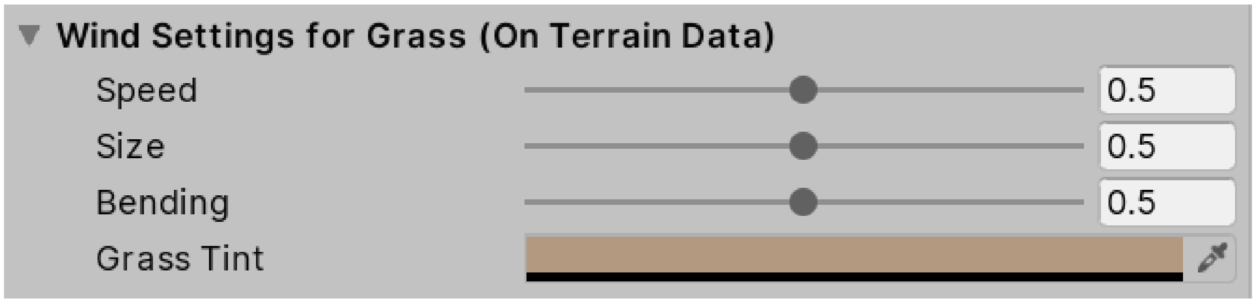
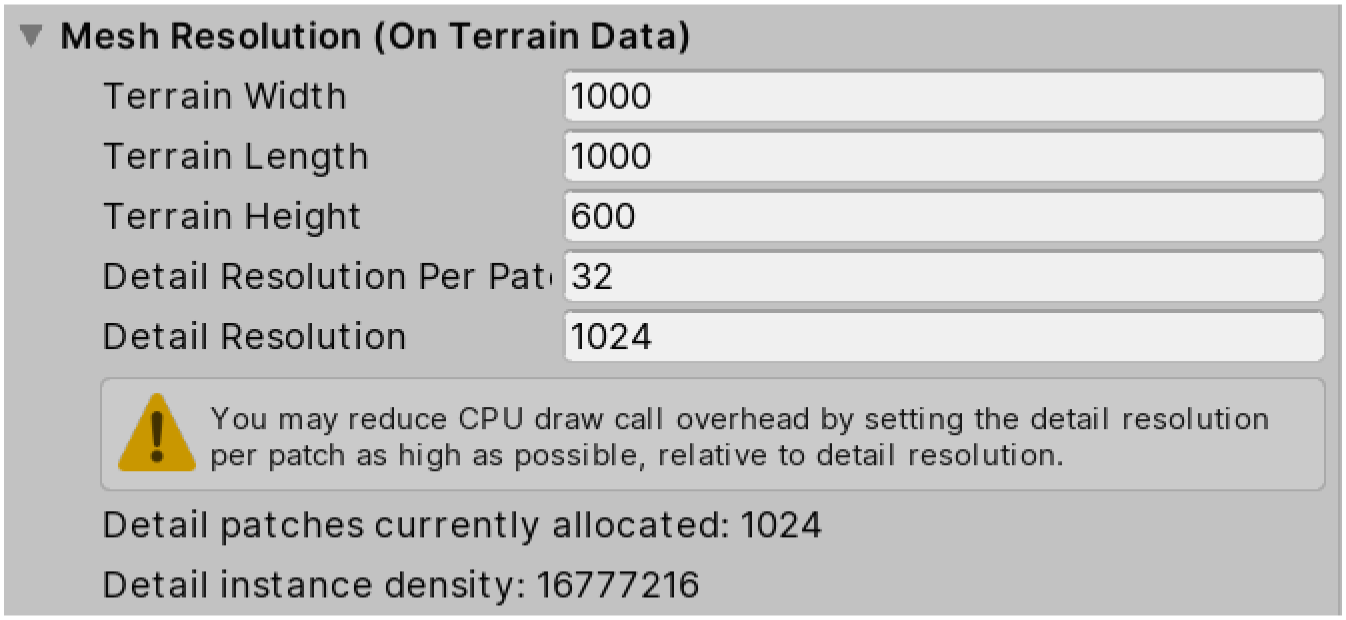

Wind Settings for Grass (On Terrain Data)
Speed 0.5
Size 0.5
Bending 0.5
Grass Tint

Figure 48 wind setting

Property	Function
Speed	The speed of the wind as it blows across the grass.
Size	The size of ripples on grassy areas as the wind blows over them.
Bending	The degree to which the wind bends over grass objects.

Property	Function
Grass Tint	The overall color tint applied to grass objects.

Mesh Resolution


Mesh Resolution (On Terrain Data)

Terrain Width: 1000
Terrain Length: 1000
Terrain Height: 600
Detail Resolution Per Pat: 32
Detail Resolution: 1024

You may reduce CPU draw call overhead by setting the detail resolution per patch as high as possible, relative to detail resolution.

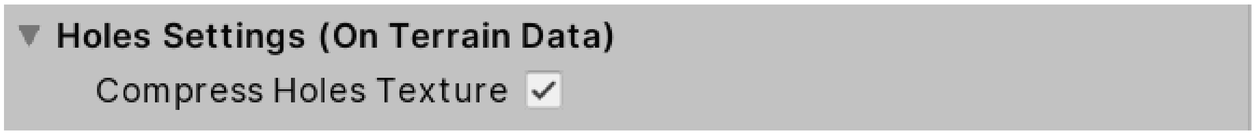
Detail patches currently allocated: 1024

Detail instance density: 16777216

Figure 49 Mesh resolution

Property	Function
Terrain Width	The size of the Terrain GameObject in its X-axis, in world units.
Terrain Length	The size of the Terrain GameObject in its Z-axis, in world units.
Terrain Height	The difference in Y-coordinate between the lowest possible heightmap value and the highest one, in world units.
Detail Resolution Per Patch	The number of cells in a single patch (mesh). This value is squared to form a grid of cells, and must be a divisor of the detail resolution.
Detail Resolution	The number of cells available for placing details onto the Terrain tile. This value is squared to make a grid of cells.

Holes Settings

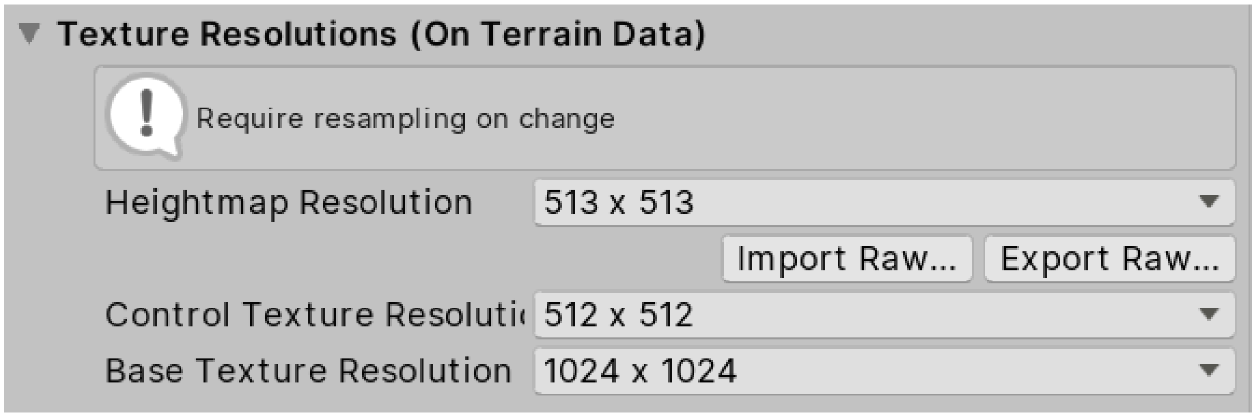

Holes Settings (On Terrain Data)

Compress Holes Texture ✓

Figure 50 Holes setting

Property	Function
Compress Holes Texture	If you check this box, Unity compresses the Terrain Holes Texture to the DXT1 graphics format in the Player during runtime. If you don't check this box, Unity does not compress the Texture.

Texture Resolutions


Texture Resolutions (On Terrain Data)

Require resampling on change

Heightmap Resolution: 513 x 513

Control Texture Resoluti: 512 x 512

Base Texture Resolution: 1024 x 1024

Figure 51 Texture resolutions

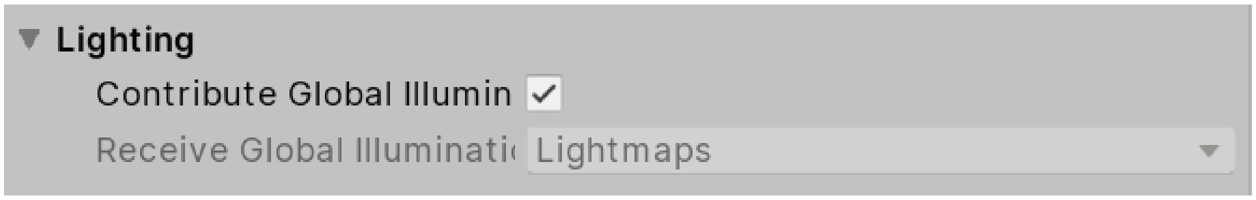
Property	Function
Heightmap Resolution	The pixel resolution of the Terrain's heightmap. This value must be a power of two plus one, for example, 513, which is 512 + 1.
Control Texture Resolution	The resolution of the splatmap that controls the blending of the different Terrain Textures.
Base Texture Resolution	The resolution of the composite Texture to use on the Terrain when you view it from a distance greater than the Basemap Distance.

Require resampling on change indicates that when you change properties under Texture Resolutions, the Editor resizes the Terrain tile's content to the new size you specify, which can potentially affect the quality of your content.

Heightmap Import/Export buttons

The Import Raw and Export Raw buttons allow you to set or save the Terrain's heightmap to an image file in the RAW grayscale format. You can create RAW format files in third-party terrain editing tools (such as Bryce), and you can open, edit, and save them in Photoshop. This allows for sophisticated generation and editing of terrains outside Unity.

Lighting

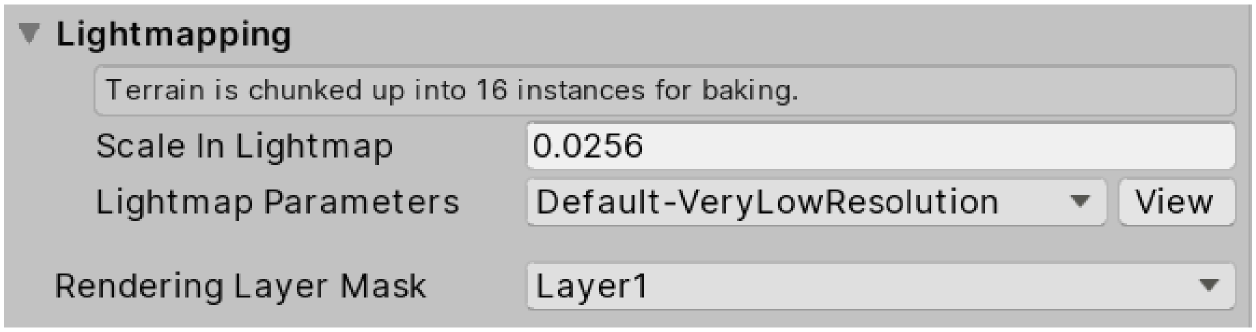


Contribute Global Illumin ✓
Receive Global Illuminati Lightmaps

Figure 52 Lighting

Property	Function
Contribute Global Illumination	Enable this check box to indicate to Unity that the Terrain influences Global Illumination computations. When you enable this property, Lightmapping properties appear.
Receive Global Illumination	You can only configure this option if you've enabled Contribute Global Illumination above. If you don't enable Contribute Global Illumination, the Terrain registers as non-static, and receives Global Illumination from Light Probes .

Lightmapping



Terrain is chunked up into 16 instances for baking.
Scale In Lightmap 0.0256
Lightmap Parameters Default-VeryLowResolution
Rendering Layer Mask Layer1

Figure 53 Light Mapping

Property	Function
Scale in Lightmap	Let’s you specify the relative size of an object’s UVs within a lightmap. If you set this to zero, the object is not lightmapped, but still contributes lighting to other objects in the Scene. A value greater than 1.0 increases the number of pixels (the lightmap resolution) used for this GameObject, while a value less than 1.0 decreases it.
Lightmap Parameters	Let’s you adjust advanced parameters that affect the process of generating a lightmap for an object using global illumination. See Lightmap Parameters for more information about these settings.
Rendering Layer Mask	Determines which rendering layer this Terrain lives on. When you use a scriptable render pipeline, you can specify an additional rendering-specific layer mask. This filters the renderers based on the mask the renderer has and the mask passed to the DrawRenderers command.

Terrain Collider

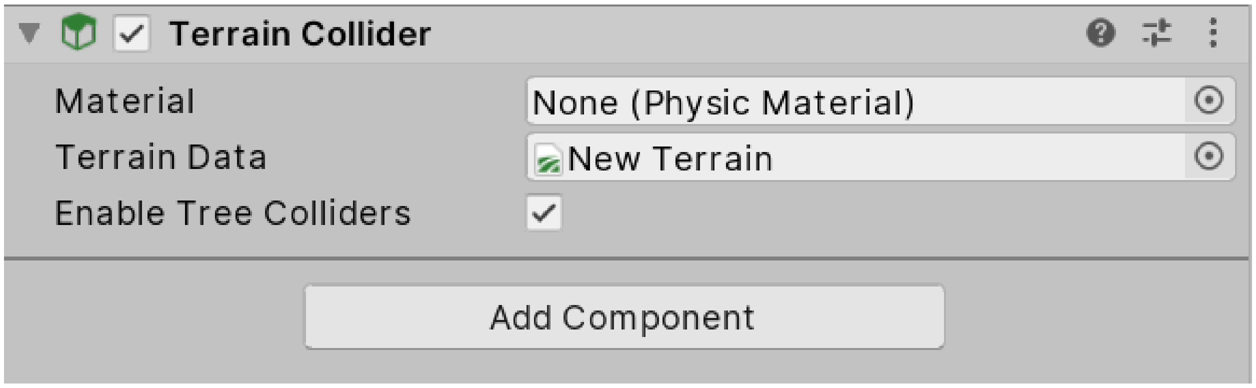

Terrain Collider
Material: None (Physic Material)
Terrain Data: New Terrain
Enable Tree Colliders: ✓

Figure 54 Terrain Collider

Property	Function
Material	A reference to the Physic Material that determines how this Terrain’s Collider interacts with other Colliders in the Scene.
Terrain Data	The TerrainData Asset that stores heightmaps, Terrain Textures, detail Meshes, and Trees.
Enable Tree Colliders	Check this box to enable Tree Colliders.