Game Level Design and Intro to Unity

MSP

Prof Vasilis Argyriou

Vasileios.Argyriou@kingston.ac.uk

Today's Lecture

- Terrain
- Environments (Skybox, Fog, Water, etc.)
- Character Controllers
- Lights, Cameras and Layers

Terrain

Terrain

The term terrain refers to any section of land that simulates a world's external landscape.

In Unity, terrain is a flat mesh that can be sculpted into many different shapes. It may help to think of terrain as the sand in a sandbox. You can dig into the sand or raise sections of it up.

The only thing basic terrain cannot do is overlap. This means that you cannot make things like caves or overhangs. Those items have to be modeled separately.

Also, just like any other object in Unity, terrain has a position, rotation, and scale

Adding Terrain to Your Project

To add terrain to a scene, just click the menu items

GameObject > 3D Object > Terrain.

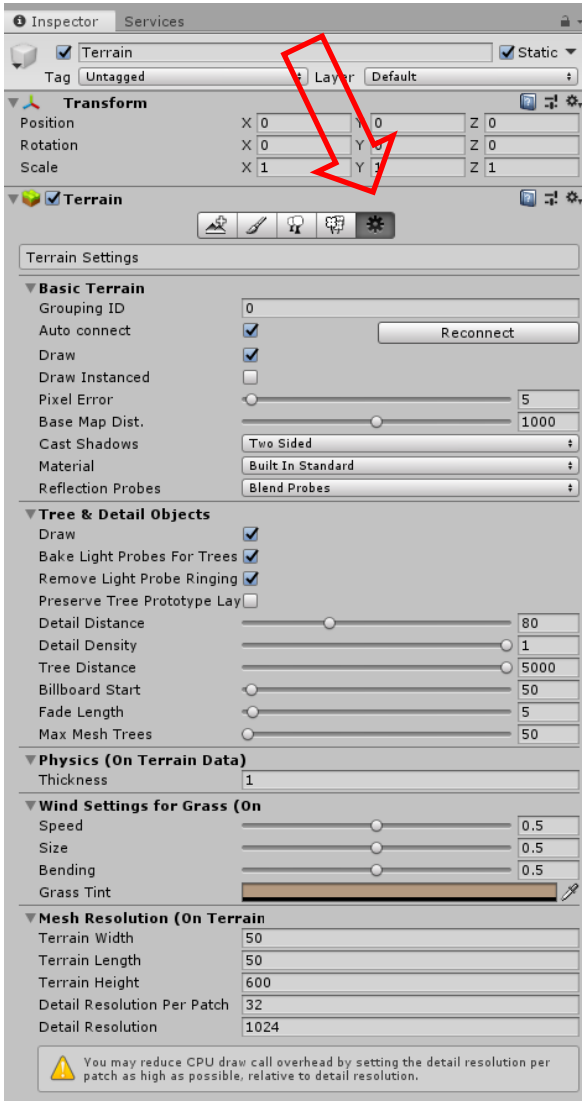
By modifying the resolution, you can change the length, width, and maximum height of your piece of terrain. The reason the term resolution is due to heightmaps used to model it.

To change the resolution of the terrain piece, follow these steps:

Adding Terrain to Your Project

1. Select your terrain in the Hierarchy view. In the Inspector view, locate and click the Terrain Settings button
2. Locate the Resolution settings.
3. Currently, the terrain width and length are set to 2000 or 500. Set these values both to 50.

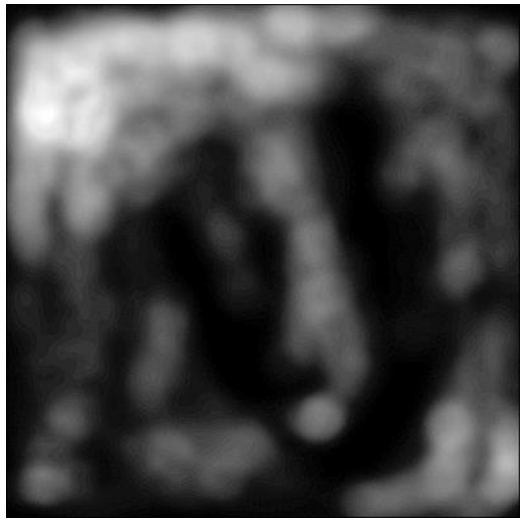
In a real game, the terrain would probably be a bigger size to fit your needs.



Heightmap Sculpting

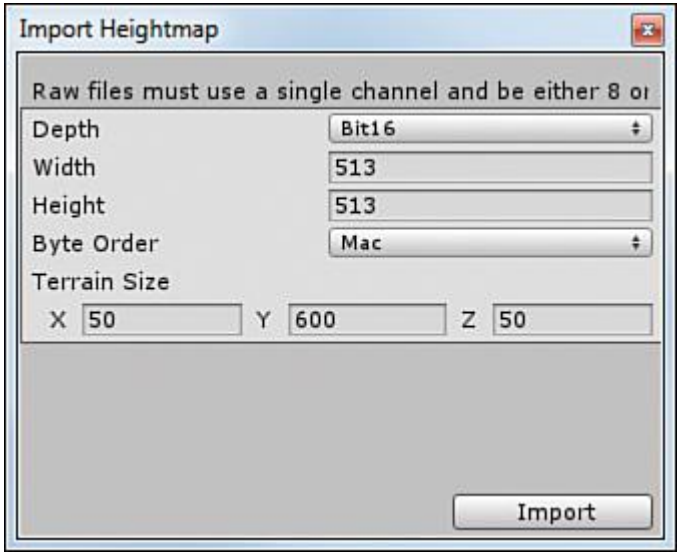
Traditionally, 256 shades of gray are available in 8-bit images. These shades range from 0 (black) to 255 (white). Knowing this, you can take a grayscale image, and use it as a heightmap.

A heightmap is a grayscale image that contains **elevation** information similar to a topographical map. The darker shades can be thought of as low points, and the lighter shades are high points.



Applying a Heightmap to Terrain

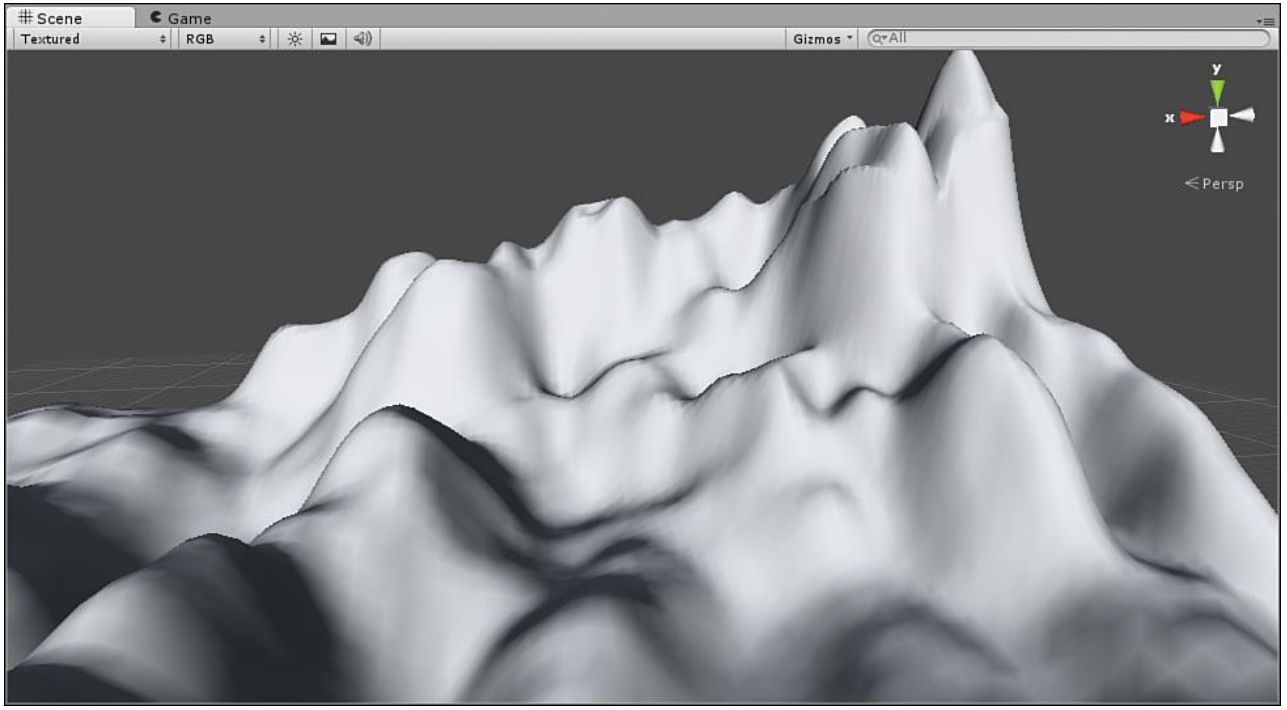
1. Locate the terrain.raw file on Canvas.
2. With your terrain selected in the Hierarchy view, click the **Terrain Settings** button. In the Heightmap section, click **Import Raw**.
3. The Import Raw Heightmap dialog will open. Locate the terrain.raw file from step 1 and click **Open**.
4. The Import Heightmap dialog will open. Leave all options as they appear and click **Import**.



You may have to adjust the values and parameters at the image

Applying a Heightmap to Terrain

5. Change the terrain resolution by going back to the Resolution section in the Terrain Settings in the Inspector view. This time, change the height value to **60**.



Unity Terrain Sculpting Tools

Unity gives you multiple tools for hand sculpting your terrain. You can see these tools in the Inspector view under the component Terrain (Script).

You use a brush with a given size and opacity to “paint” terrain. In effect, what you are doing behind the scene is painting a heightmap that is translated into changes for the 3D terrain.

Unity Terrain Sculpting Tools

The Raise/Lower tool enables you to raise or lower the terrain wherever you paint.

1. Select a brush. Brushes determine the size and shape of the sculpting effect.

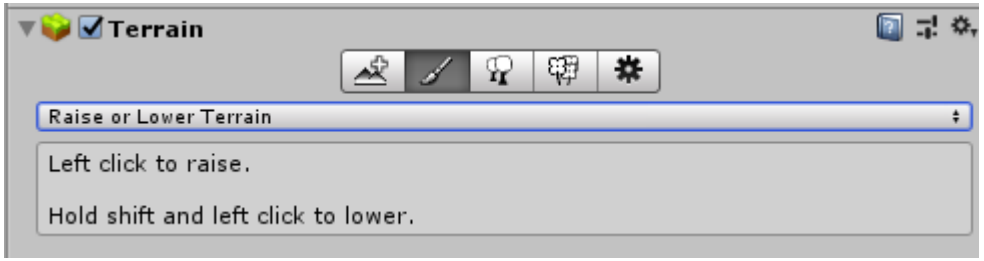
2. Choose a brush size and opacity. The opacity determines how strong the sculpting effect is.

3. Click and drag over the terrain in the Scene view to raise the terrain. Holding **Shift** when you click and drag will instead lower the terrain.

Unity Terrain Sculpting Tools

The Paint Height tool works almost exactly as the Raise/Lower tool except that it paints your terrain to a specified height.

If the specified height is higher than the current terrain, painting raises the terrain. If the specified height is lower than the current terrain, however, the terrain is lowered.



This proves useful for creating mesas or other flat structures in your landscape.

Unity Terrain Sculpting Tools

The Smooth Height tool doesn't alter the terrain in highly noticeable ways. Instead, it removes a lot of the jagged lines that appear when sculpting terrain.

If, at any time, you want to reset your terrain back to being flat, you can do so by going to the Paint Height tool and clicking **Flatten**.

If you sculpt some terrain using Unity's sculpting tools and you want to generate a heightmap for it, you can by going to the Heightmap section in the Terrain Settings in the Inspector view and clicking **Export Raw**

Terrain Textures

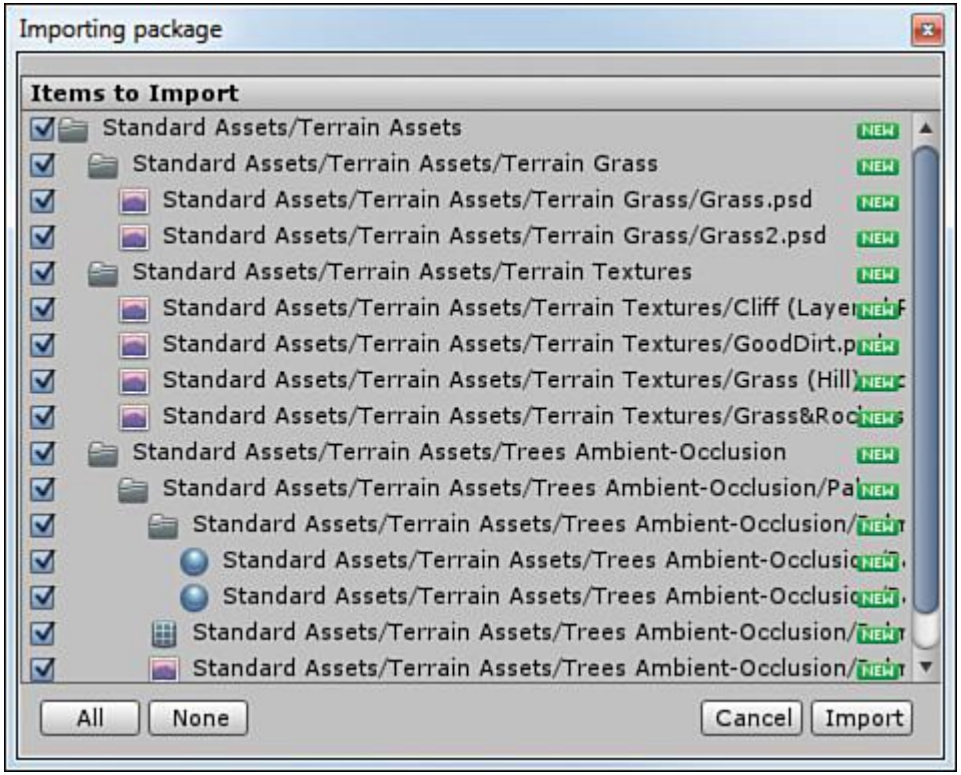
Like sculpting terrain, texturing terrain works a lot like painting. You select a brush and a texture and paint it onto your world.

Unity has some terrain assets available to you, but you need to import them first. To load these assets, download the **Standard Assets** from Asset Store. The Importing Package dialog will appear.

Just leave all options checked and click **Import**. You should now have a new folder under Assets in the Project view called Standard Assets.

Terrain Textures

The Importing Package dialog

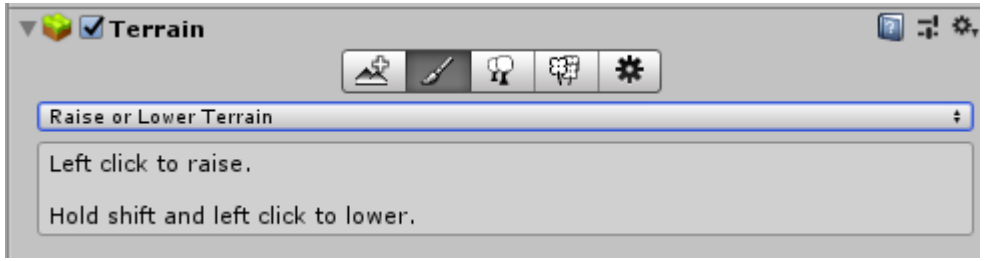


Texturing Terrain

In the Inspector pay attention to the three numeric properties: brush size, opacity, and target strength.

The target strength is the maximum opacity that it achievable through constant painting. This is a percentage, with 1 being 100%.

Use this as a control to prevent painting your textures on too strongly.

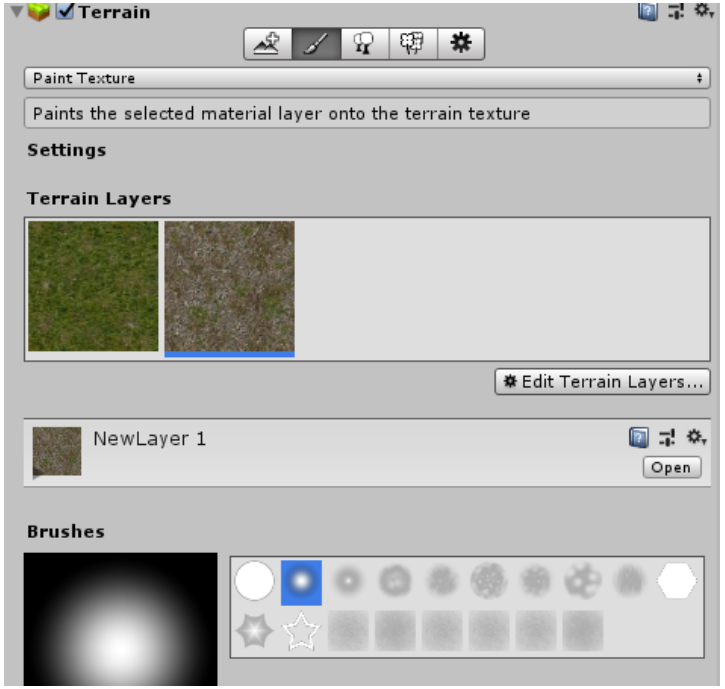


Texturing Terrain

To load a texture, follow these steps:

1. Click **Edit Textures > Add Texture**.
2. The Add Terrain Texture dialog will appear. Click **Select** in the Texture box and select **the Grass (Hill) texture**.
3. Click **Add**

Now the entire terrain should be covered in patchy grass



Painting Textures onto Terrain

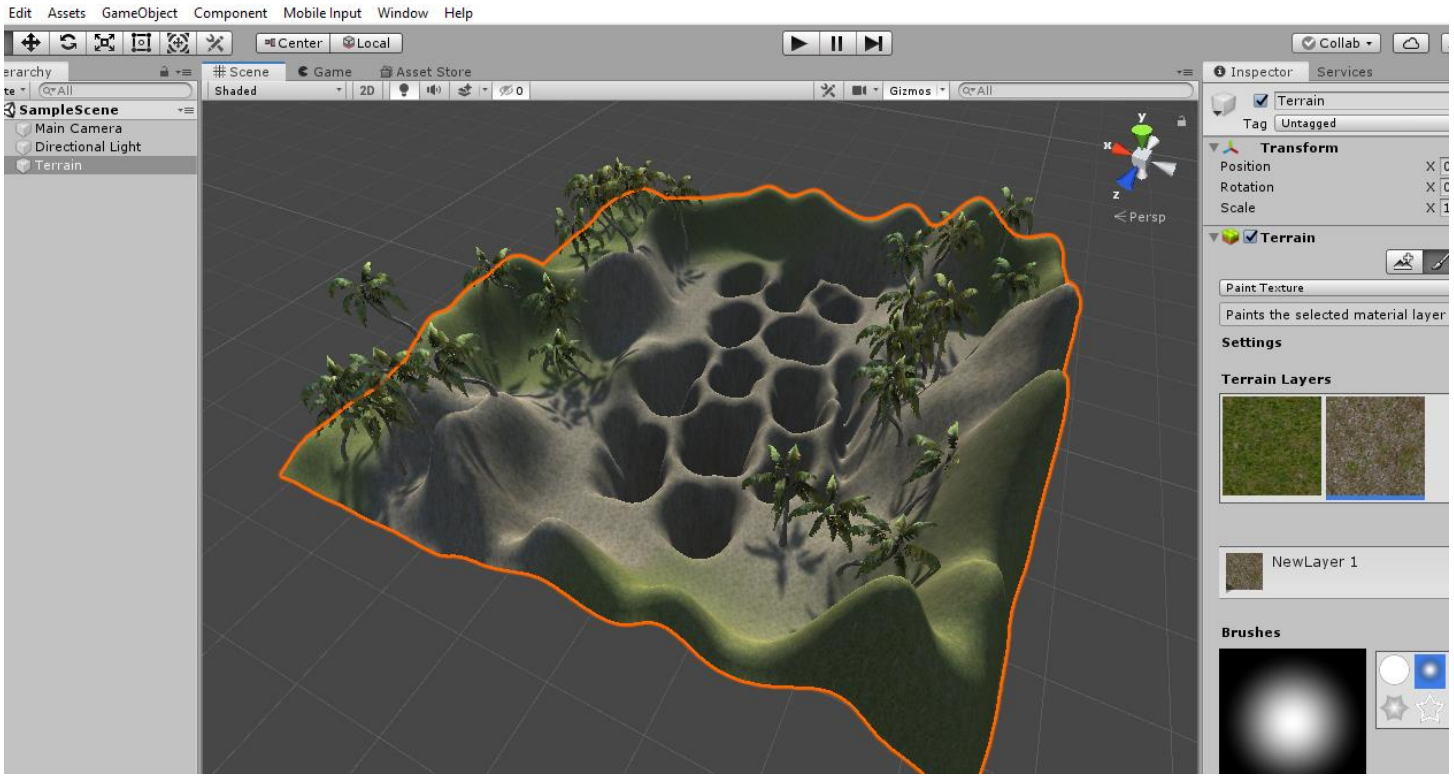
1. Using the steps listed earlier, add a new texture. This time, load the **Grass&Rock** texture. Once you have loaded it, be sure to select it by clicking it. (A blue bar appears under it if it is selected.)

2. Set your brush size to **30**, your opacity to **20**, and your target strength to **0.6**.

3. Sparingly, paint (click and drag) on the steep parts and crevices of your terrain. This gives the impression that grass isn't growing on the sides of steep grades and in between hills

Painting Textures onto Terrain

4. Continue experimenting with texture painting. Feel free to load the texture **Cliff** and apply it to steeper parts or the texture **Sand** and make a path. (or other textures that are available).



Environments

Environments

Adding trees to your terrain works just like the sculpting and texturing.

The basic premise is to load a tree model, set the properties for the trees, and then paint the area you want the trees to appear in.

Once the terrain has been selected in the scene, the Place Trees tool is accessed in the Inspector view as part of the Terrain (Script) component.

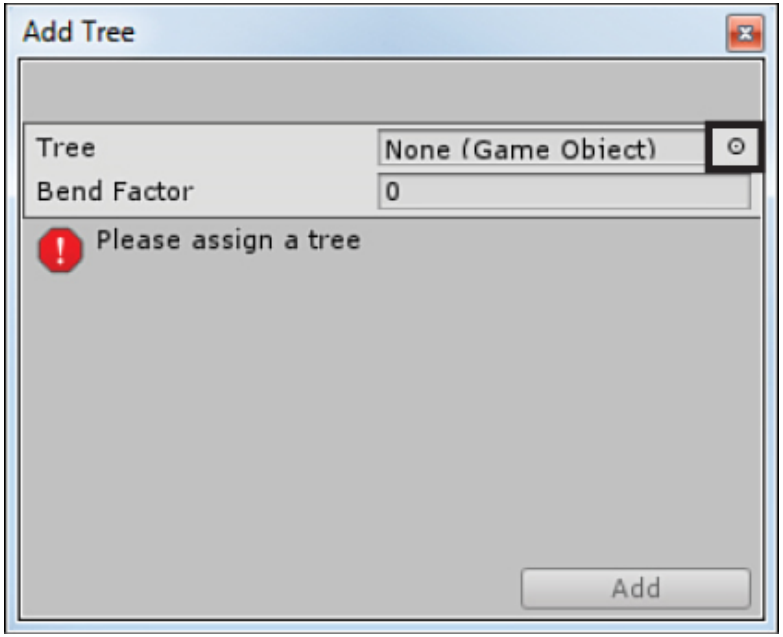
Painting Trees

Property	Description
Brush Size	The size of the area that trees are added to when painting.
Tree Density	How densely the trees will be able to be packed.
Color Variation, Tree Height/Width, and Variations	These properties allow all the trees to be slightly different from each other. This gives the impression of many different trees instead of the same tree repeated.

Painting Trees

1. Click **Edit Trees** > **Add Tree** to pull up the Add Tree dialog
2. Clicking the **circle** icon to the right of the Tree text box on the Add Tree dialog pulls up the Tree Selector dialog
3. Select the **Palm Tree** and click **Add**.

4. Set your brush size to **10**, your tree density to **70**, and your width and height to **50**. Choose whichever variation properties you want.



Painting Trees

5. Paint trees on the terrain by clicking and dragging over the areas where you want trees. Holding the **Shift** key while click-dragging removes trees.

You might notice that some trees appear black. This can happen when smaller trees are placed close to larger trees. The reason is that whatever lighting exists in your scene is calculated as hitting the taller trees around the smaller tree.

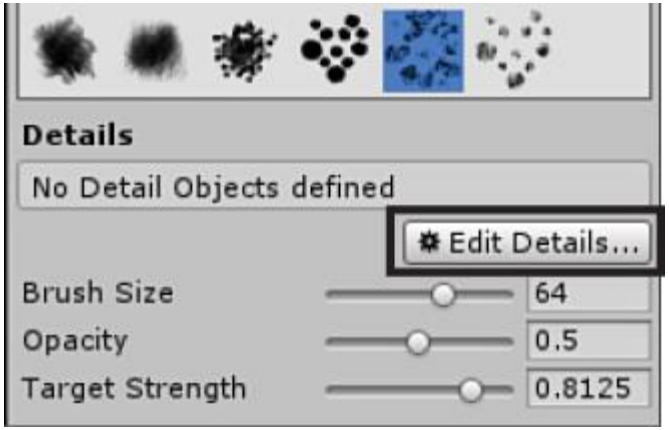
When trees are far away from the viewer, they get rendered as a much lower-quality billboard. As the viewer gets closer, the trees are swapped with higher-quality versions.

Painting Grass

Grass and other small plants are called details in Unity.

Unlike trees, which are 3D models, details are **billboards**.

Paint Details tool and some of its properties.



Billboards

Billboards are a special type of visual component in a 3D world that give the effect of a 3D model without actually being a 3D model.

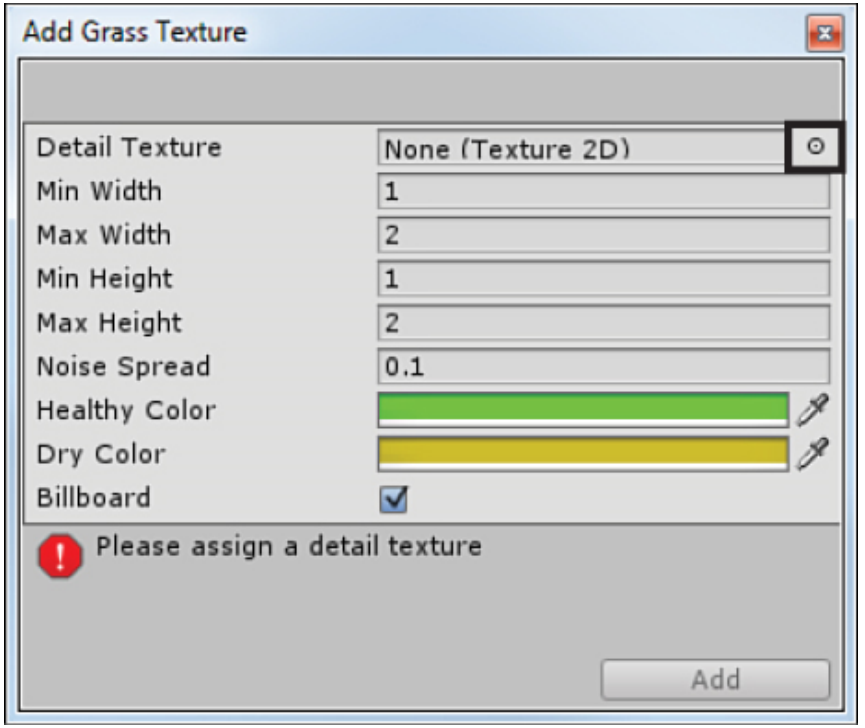
Models exist in all three dimensions. Therefore, when moving around one, you can see the different sides.

Billboards, however, are flat images that always face the camera. When you attempt to go around a billboard, the billboard turns to face your new position.

Common uses for billboards are grass details, particles, and onscreen effects.

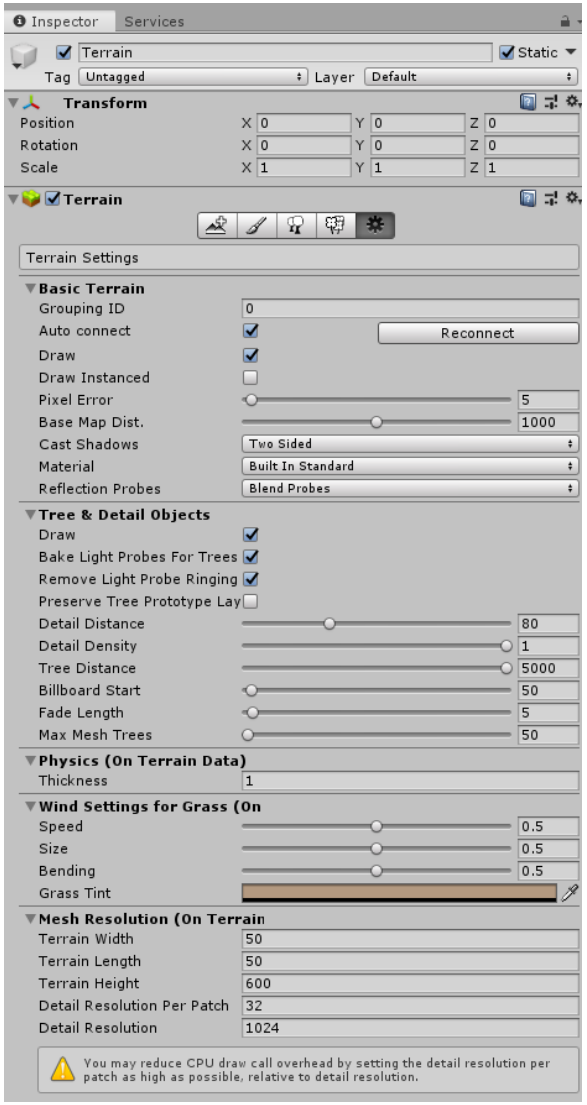
Painting Grass

1. Click **Edit Details** in the Inspector view and select **Add Grass Texture**.
2. In the Add Grass Texture dialog, click the **circle** icon next to the Texture text box. Select the **Grass** texture
3. Set your texture properties to whatever values you want. Pay special attention to the color properties because those establish the range of natural colors for your grass.
4. When done, click **Add**.



Terrain Settings

These settings control how the terrain, texture, trees, and details look and function overall



Terrain Settings

Setting	Description
Pixel Error	Number of allowable errors when displaying terrain geometry. The higher the value, the lower the detail of the terrain.
Base Map Dist.	The maximum distance that high-resolution textures will be displayed. When the viewer is farther than the given distance, textures degrade to a lower resolution.
Cast Shadows	Determines whether terrain geometry casts shadows.
Material	This slot is for assigning a custom material capable of rendering terrain. The material must contain a shader capable of rendering terrain.

Terrain Settings

Some settings directly affect the way trees and details (like grass) behave in your terrain

Setting	Description
Draw	Determines whether trees and details are rendered in the scene.
Detail Distance	The distance from the camera where details will no longer be drawn to the screen.
Tree Distance	The distance from the camera where trees will no longer be drawn to the screen.
Billboard Start	The distance from the camera where 3D tree models will begin to transition into lower-quality billboards.
Fade Length	The distance over which trees will transition between billboards to higher-quality 3D models. The higher the setting, the smoother the transition.
Max Mesh Trees	The total number of trees able to be drawn simultaneously as 3D meshes and not billboards.

Terrain Settings

Unity simulates a light wind over your terrain. This light wind causes the grass to bend and sway and livens up the world.

Setting	Description
Speed	The speed, and therefore the strength, of the wind effect.
Size	The size of the area of grass affected by the wind at the same time.
Bending	The amount of sway the grass will have due to wind.
Grass Tint	Although not a wind setting, this setting controls the overall coloration of all grass in your level.

Skyboxes

A skybox is a large box that goes around your world. Even though it is a cube consisting of six flat sides, it has inward-facing textures to make it look round and infinite.

You can create your own skyboxes or use one of Unity's standard skyboxes.

To use the standard skyboxes, you need to import the assets into your project. To import the skyboxes, use the asset store to get free examples. Then create a material of type skype box and apply it in the scene (e.g. camera)

Skyboxes

There are two ways to add skyboxes to your world: You can add the skybox to your camera or add it to the scene.

You can add a **skybox to your camera** so that whatever the camera sees beyond your game world will be replaced with sky. To add a skybox to your camera:

1. Select the **Main Camera** in the Hierarchy view.
2. Add a skybox component by clicking **Component** > **Rendering** > **Skybox**.
3. In the Inspector view, locate the Skybox component and click the **circle** icon next to the Custom Skybox field. In the Select Material dialog, select an existing skybox or download one from asset store.

Skyboxes

If you add a **skybox to the scene**, it will be present for all viewers. Another benefit of this method is that the skybox will be visible in the Scene view. This makes it easy to see how your world looks with all elements in place.

1. Click **Window > Rendering > Lighting** to open the render settings for the scene in the Inspector view.
2. Locate the Skybox Material field and click the **circle** icon to the right of it.
3. Choose another Skybox. Notice how the Scene view changes to contain the sky.

If the Scene view doesn't change, turn on the skybox, fog, and lens flare scene setting

Fog

You can use this fog to simulate many different natural occurrences, such as haze, an actual fog, or the fading of objects over great distances. You can also use fog to give new and alien appearances to your world.

1. Click **Window > Rendering > Lighting**
The render settings will open in the Inspector view.
2. Turn on fog by checking the **Fog** check box.

3. Experiment with the different fog densities and colors.

Fog

Setting	Description
Fog Color	The color of the fog effect.
Fog Mode	This controls how the fog is calculated. The three modes are Linear, Exponential, and Exp2. Linear will be a smooth fog transition and is the default mode.
Fog Density	How strong the fog effect is. This property is used only if the fog mode is set to Exponential or Exp2.
Linear Fog Start Linear Fog End	These control how close to the camera the fog starts and how far from the camera it ends. These properties are only used in Linear mode.

Lens Flares

A lens flare is a visual deformity that occurs whenever a camera looks at a bright light source.
Before you can add a flare to your scene, you need to have a light source and some flare assets. These will all be taken care of in the following steps:

1. Add a directional light to your scene by clicking **GameObject > Light > Directional Light**. A directional light is a parallel light, just like the sun.
2. Once the light is added to your scene, rotate it so that it gives the desired light effect on your terrain.

Lens Flares

3. Next, you need light flare assets. Import the Unity Effects light flare assets. In the Import dialog, leave everything checked and click Import.

4. Select the directional light in the Hierarchy view and locate the Flare property in the Inspector view.

5. Click the **circle** icon next to the Flare property and choose the **Sun** flare from the Select Flare dialog.

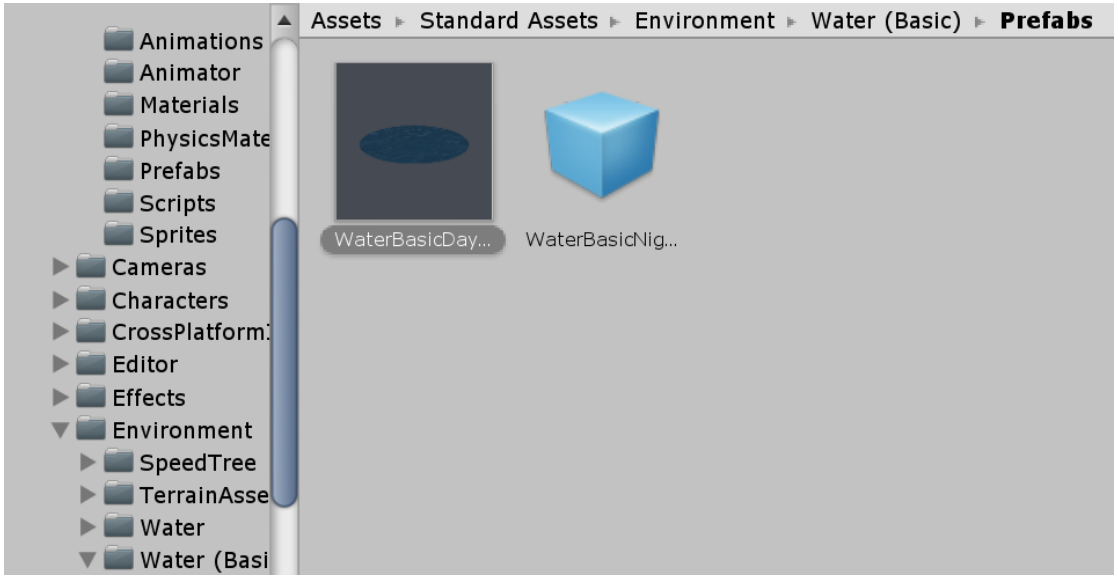
Water

Water is an effect that varies depending on whether you have the Pro or free version of Unity.
Note that the water is just an effect. If a player jumps into a lake with water, the player will fall right through the water and down into the hole that was sculpted for it.

1. Create a new terrain or work with an existing terrain. Sculpt a lakebed down into the terrain.
2. Import the water assets. (It should be part of the Standard Assets).
3. Locate the Water (Basic) folder in the Project view and locate the Daylight Simple Water asset

Water

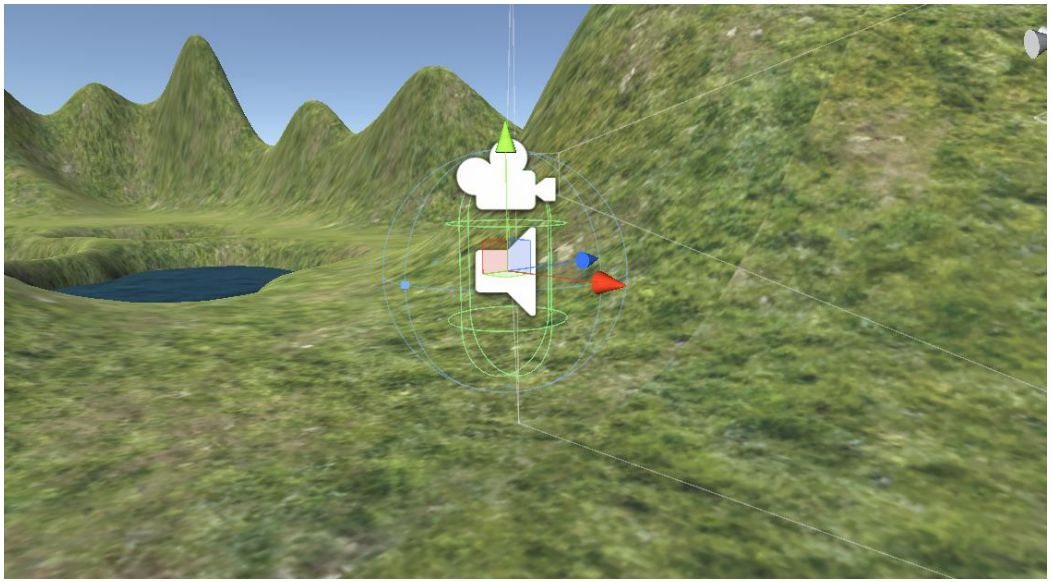
4. Drag the Daylight Simple Water asset onto the Scene view and into the lakebed you created. Scale and move the water as necessary until it fills up the lakebed properly.



Character Controllers

To add a character controller to your scene, you first need to import the asset. (It should be part of the Standard Assets).

Locate the First Person controller asset in the Character Controllers folder and drag it onto your terrain in the Scene view.



Character Controllers

When you added the character controller to the scene, you might have noticed a message at the bottom of the editor that said, "There are 2 audio listeners in the scene." This is because the Main Camera (the camera that exists by default) has an audio listener component and so does the character controller that you added.

Because the cameras represent the player's perspective, only one can listen for audio.

You can fix this by removing the audio listener component from the Main Camera.

Lights and Cameras

Lights and Cameras

In Unity, lights are not objects themselves. Instead, lights are a component.

This means that when you add a light to a scene, you are really just adding a game object with the Light component.

This light component can be any of the types of light you can use.

Point Lights

The first light type you will be working with is the point light. Think of a point light as a light bulb. All light is emitted from one central location out in every direction. The point light is also the most common type of light for illuminating interior areas.

To add a point light to a scene, click **GameObject > Light > Point Light**. Once in the scene, the point light game object can be manipulated just like any other.

Point Lights

Property	Description
Type	The Type property is the type of light that the component gives off. Because this is a point light, the type should be Point. Changing the Type property changes the type of light it is.
Range	The Range property dictates how far the light shines. Illumination will fade evenly from the source of light to the range dictated.
Color	The color the light shines. Color is additive, which means that if you shine a red light on a blue object, it will end up purple.
Intensity	The Intensity property dictates how brightly a light will shine. Note that the light will still shine only as far as the Range property dictates.
Cookie	The Cookie property accepts a cubemap (like a skybox) that dictates a pattern for the light to shine through.
Shadow Type	The Shadow Type property is how shadows are calculated for this source in a scene. Hard shadows are more accurate and more performance intensive. All shadows require Unity Pro to function. If you are using Unity Free, the only way to have shadows is if you manually bake them into your textures.
Draw Halo	The Draw Halo toggle determines whether a glowing halo will appear around your light.
Flare	The Flare property accepts a light flare asset and simulates the effect of a bright light shining into a camera lens. You have worked with light flares in previous hours to simulate a sun effect.

Adding a Point Light

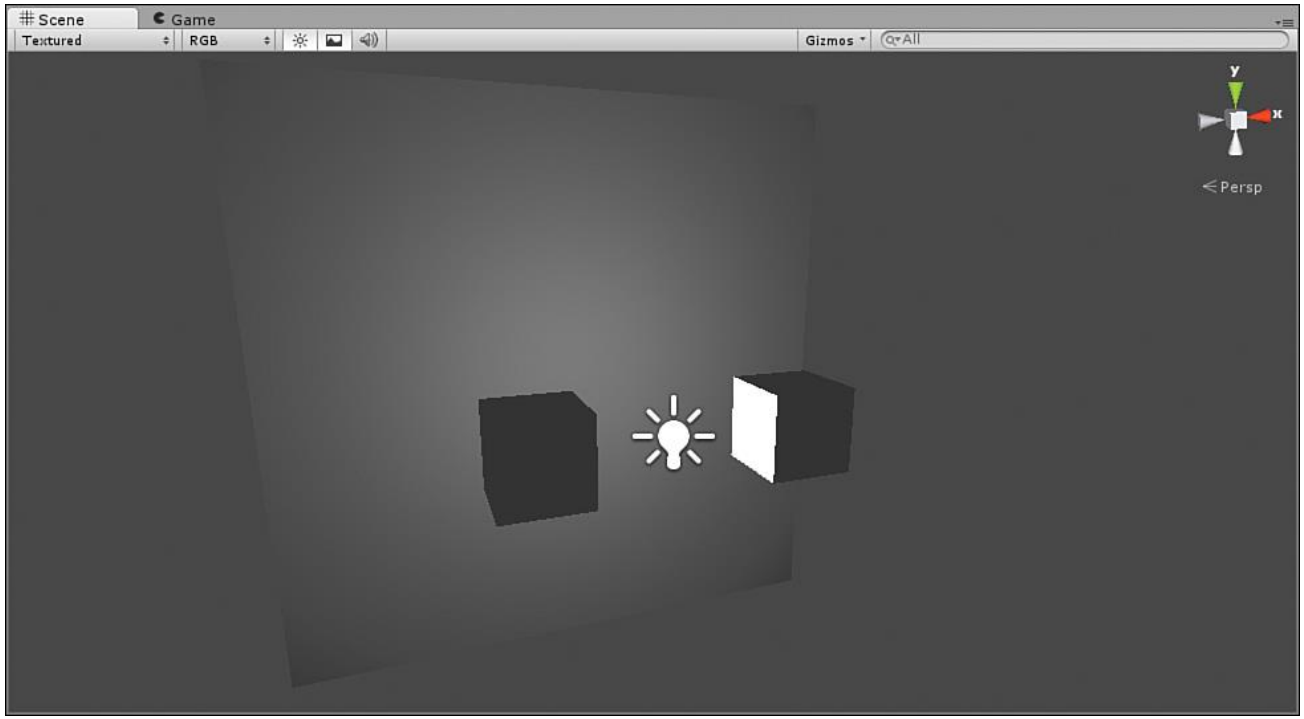
1. Create a new scene or project.

2. Add a plane to the scene (**GameObject > 3D Object > Plane**). Ensure that the plane is positioned at (0, .5, 0) and it rotated (270, 0, 0). The plane should be visible to the camera.

3. Add two cubes to the scene. Position them at (–1.5, 1, -5) and (1.5, 1, –5).

Adding a Point Light

4. Add a point light to the scene (**GameObject > Light > Point Light**). Position it at (0, 1, –5). Notice how the light illuminates the inner sides of the cubes and the background plane. Be sure to experiment with the light color, range, and intensity.



Spotlights

Spotlights work a lot like the headlights in a car or flashlights. The light of a spotlight begins in at a central spot and then radiates out in a cone. In other words, spotlights illuminate whatever is in front of them while leaving everything else in the dark.

To add a spotlight to your scene, click **GameObject > Light > Spotlight**.

The **Spot Angle** property determines the radius of the cone of light emitted by the spotlight.

Directional Lights

The directional light is similar to the spotlight in that it can be aimed. Unlike the spotlight, though, the directional light illuminates the entire scene.

To add a directional light to your scene, click **GameObject > Light > Directional Light**.

Area light

An area light is a Unity feature that exists for a process called lightmap baking.

Creating Lights out of Objects

Because lights in Unity are components, any object in a scene can be a light. To add a light to an object, first select the object. Then in the Inspector view, click the **Add Component** button. A new list should pop up. Select **Rendering** and then **Light**. Now your object has a light component.

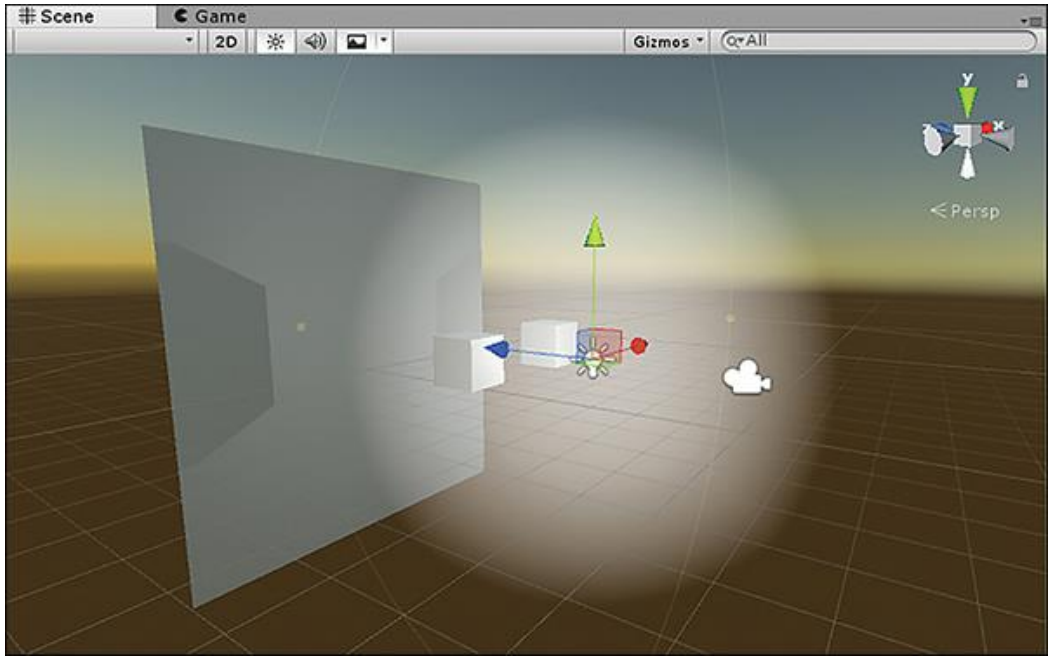
The object will not block the light.

Adding a light to an object does not make it glow

Halos

Halos are glowing circles that appear around lights in foggy or cloudy conditions

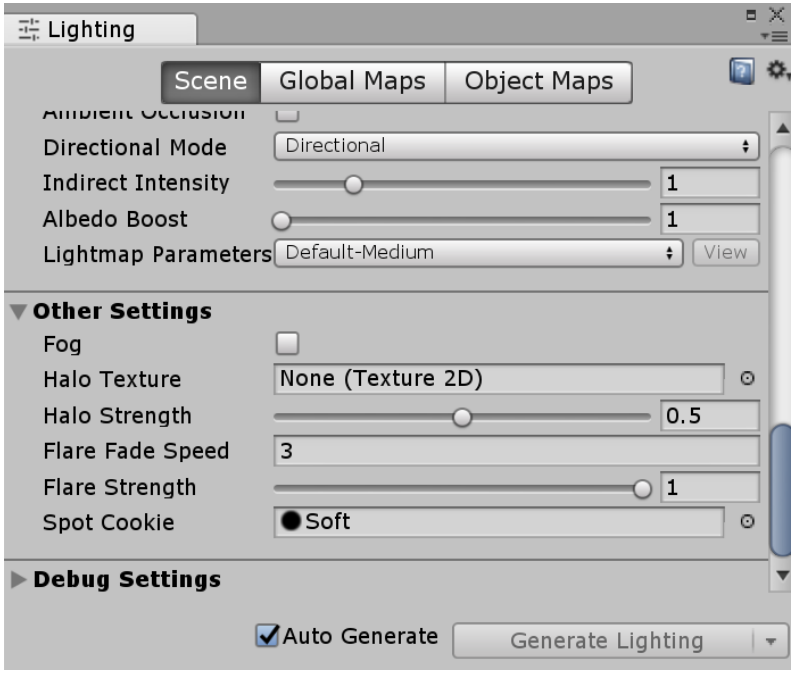
In Unity, you can easily add halos to your lights. Each light has a check box called Draw Halo. If it is checked, a halo will be drawn for the light.



Halos

The size of a halo is determined by the light's range. The bigger the range, the bigger the halo. Unity also provides a few properties that apply to all halos in a scene. You can access these properties by clicking **Window > Rendering > Lighting**

The Halo Texture property allows you to specify a different shape for your halo by providing a new texture.



Cookies

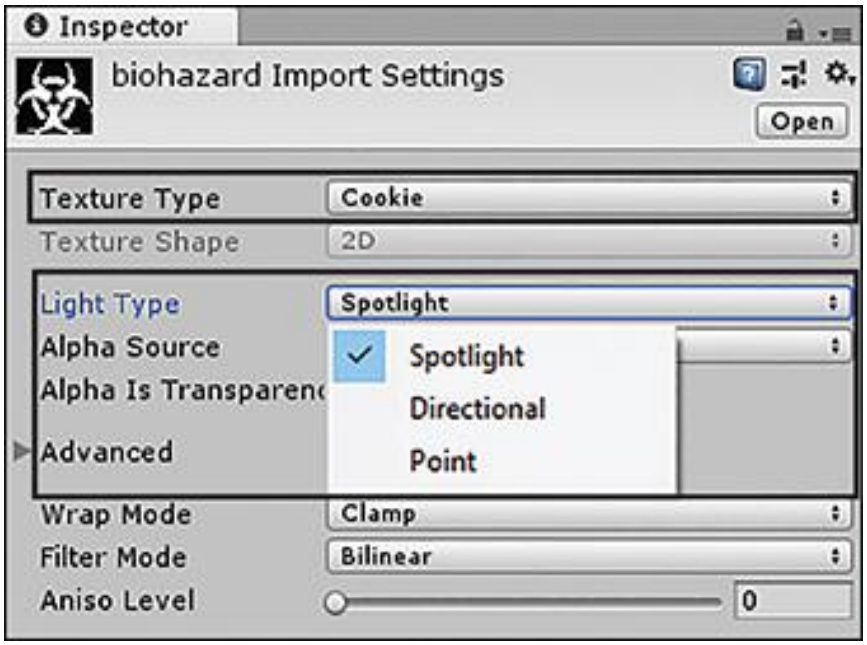
Cookies are special textures that you can add to lights to dictate how the light radiates. Cookies differ a little for point, spot, and directional lights.

Spotlights and directional lights both use black-and-white flat textures for cookies.

Point lights also use black-and-white textures, but they must be placed in a cubemap. A cubemap is six textures placed together to form a box (like a skybox).

Cookies

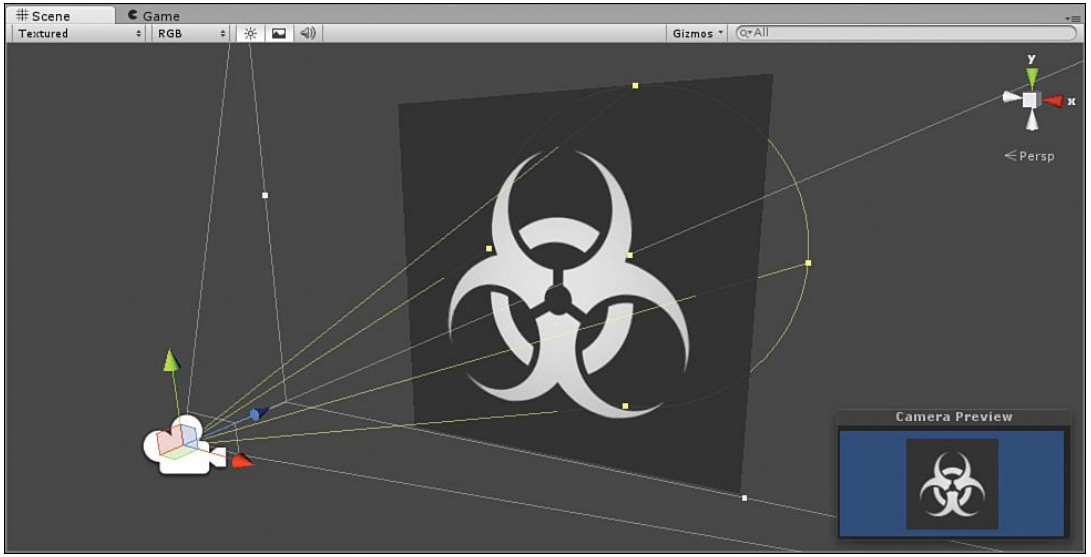
Select a texture in Unity, and then change its properties in the Inspector window for a point cookie, a spot cookie, and a directional cookie.



Select the texture, and in the Inspector view, change the texture type to **Cookie**, set the light type to **Spotlight**, and set the alpha source to **From Grayscale**.

Adding a Cookie

1. Select the texture, and in the Inspector view change the texture type to Advanced. Check the Alpha from Grayscale check box. Check the Border Mip Maps check box. Finally, change the wrap mode to Clamp. Click Apply.
2. Click and drag the biohazard texture into the Cookie property of the light component.



Baking

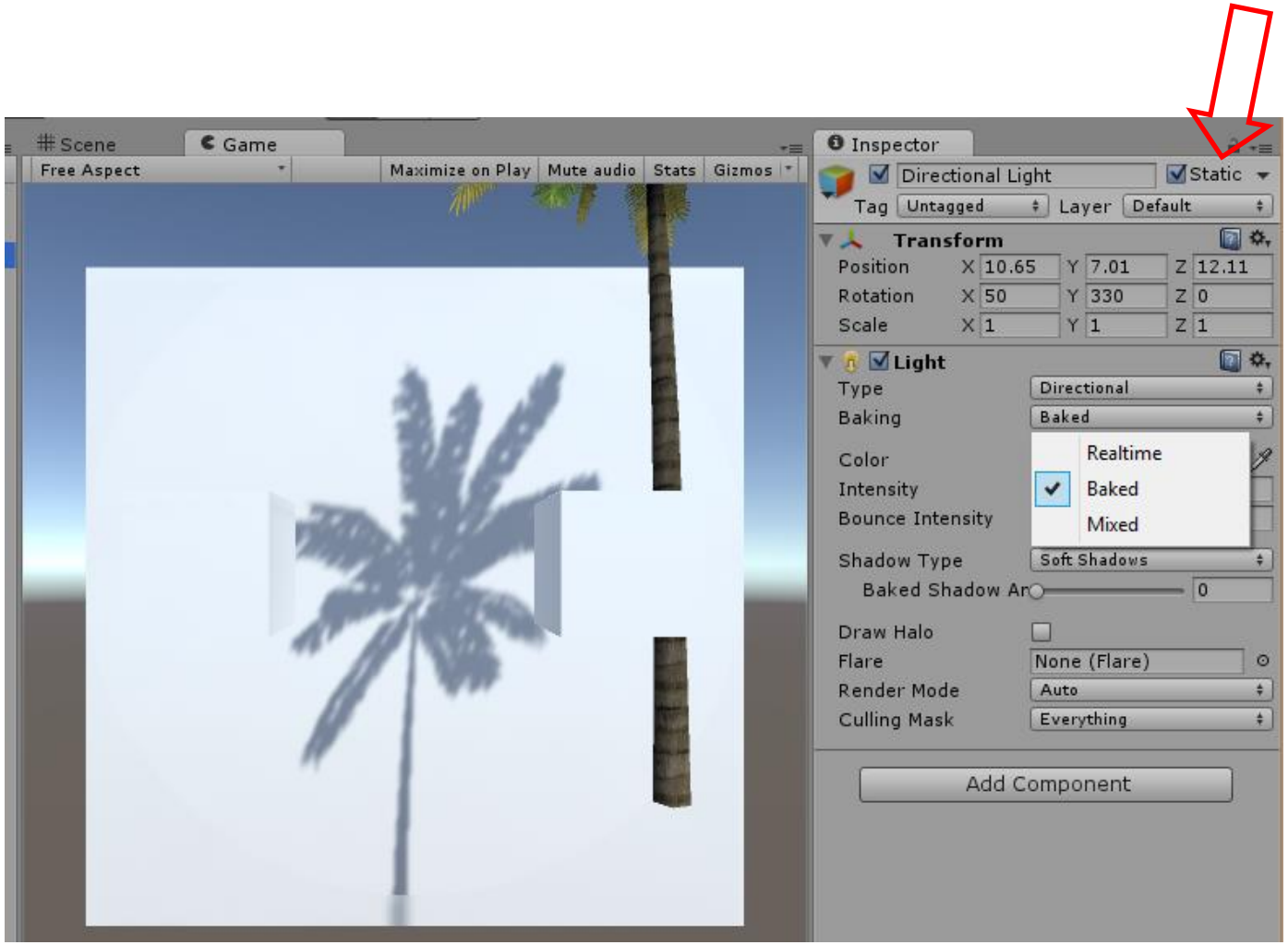
Baking refers to the process of adding light and shadow to textures and objects during creation. You can do this with Unity or with a graphical editor.

For instance, if you were to make a wall texture with a dark spot on it that resembled a human shadow, and then put a human model next to the wall it was on, it would seem like the model was casting a shadow on the wall. The truth is, though, that the shadow was “baked” into the texture.

Baking can make your games run much more quickly because the engine won’t have to calculate light and shadow every single frame.

Baked Scenes

Object that are static should be selected as static



Cameras

Cameras

The camera is the player's view into the world. All games in Unity have at least one camera.

Property	Description
Clear Flags	The Clear Flags property determines what the camera displays in the areas where there are no game objects. The default is Skybox. If there is no skybox, the camera defaults to a solid color. Depth Only should be used only when there are multiple cameras. Don't Clear causes streaking and should be used only if writing a custom shader.
Background	The Background property dictates the background color if there is no skybox present.
Culling Mask	The Culling Mask property determines what layers are picked up by the camera. By default, the camera sees everything. You can uncheck certain layers and they won't be visible to the camera.
Projection	The Projection property determines how the camera sees the world. The two options are Perspective and Orthographic. Perspective cameras perceive the world in 3D, where closer objects are larger and farther objects are smaller. This is the setting to use if you want depth in your game. The Orthographic camera setting ignores depth and treats everything as flat.
Field of View	The Field of View property specifies how wide of an area the camera can see.

Cameras

Clipping Planes	The Clipping Planes property dictates the range where objects are visible to the camera. Objects that are closer than the near plane or farther than the far plane will not be seen.
Normalized View Port Rect	The Normalized View Port Rect property establishes what part of the actual screen the camera is projected on. By default, the X and Y are both set to 0, which causes the camera to start in the upper left of the screen. The width and height are both set to 1, which causes the camera to cover 100% of the screen vertically and horizontally.

The GUI Layer allows the camera to see GUI elements

The Flare Layer allows the camera to see the lens flares of lights.

The audio listener allows the camera to pick up sound.

Multiple Cameras

1. Create a new project or scene and add two cubes. Place the cubes at (–2, 1, –5) and (2, 1, 5). Add a directional light to the scene.
2. Move the Main Camera to (–3, 1, –8) and change its rotation to (0, 45, 0).
3. Add a new camera to the scene (click **GameObject > Camera**) and position it at (3, 1, –8). Change its rotation to (0, 315, 0).
4. Run the scene. Notice how the second camera is the only one displayed. This is because the second camera has a higher depth than the Main Camera.

Change the Main Camera to 1 and then run the scene again.

Split Screen

The viewport basically treats the screen as a simple rectangle. The upper-left corner of the rectangle is (0, 0), and the lower-right corner is (1, 1).
Think of the coordinates as percentages of the size.

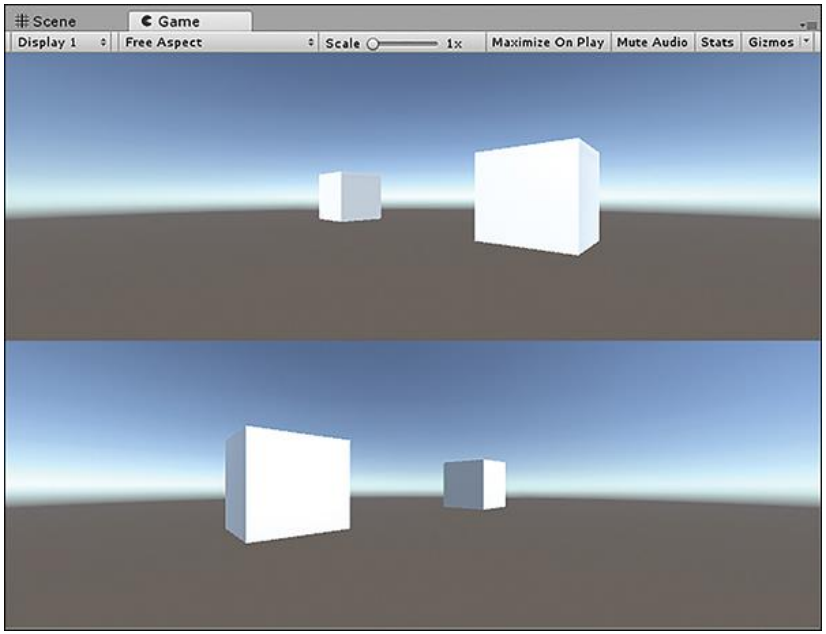
A split-screen camera system is common in two-player games where the players have to share the same screen.

1. Ensure that the Main Camera has a depth of –1. Ensure that the X and Y properties of the camera's ViewPort Rect property are both 0.
2. Set the W and H properties to 1 and .5, respectively (100% of the width and 50% of the height).

Split Screen

3. Ensure that the second camera also has a depth of –1. Set the X and Y properties of the view port to (0, .5). This will cause the camera to begin drawing halfway down the screen. Set the W and H properties to 1 and .5, respectively.

4. Run the scene and notice how both cameras are now projecting on the screen at the same time. You can split the screen like this as many times as you want.

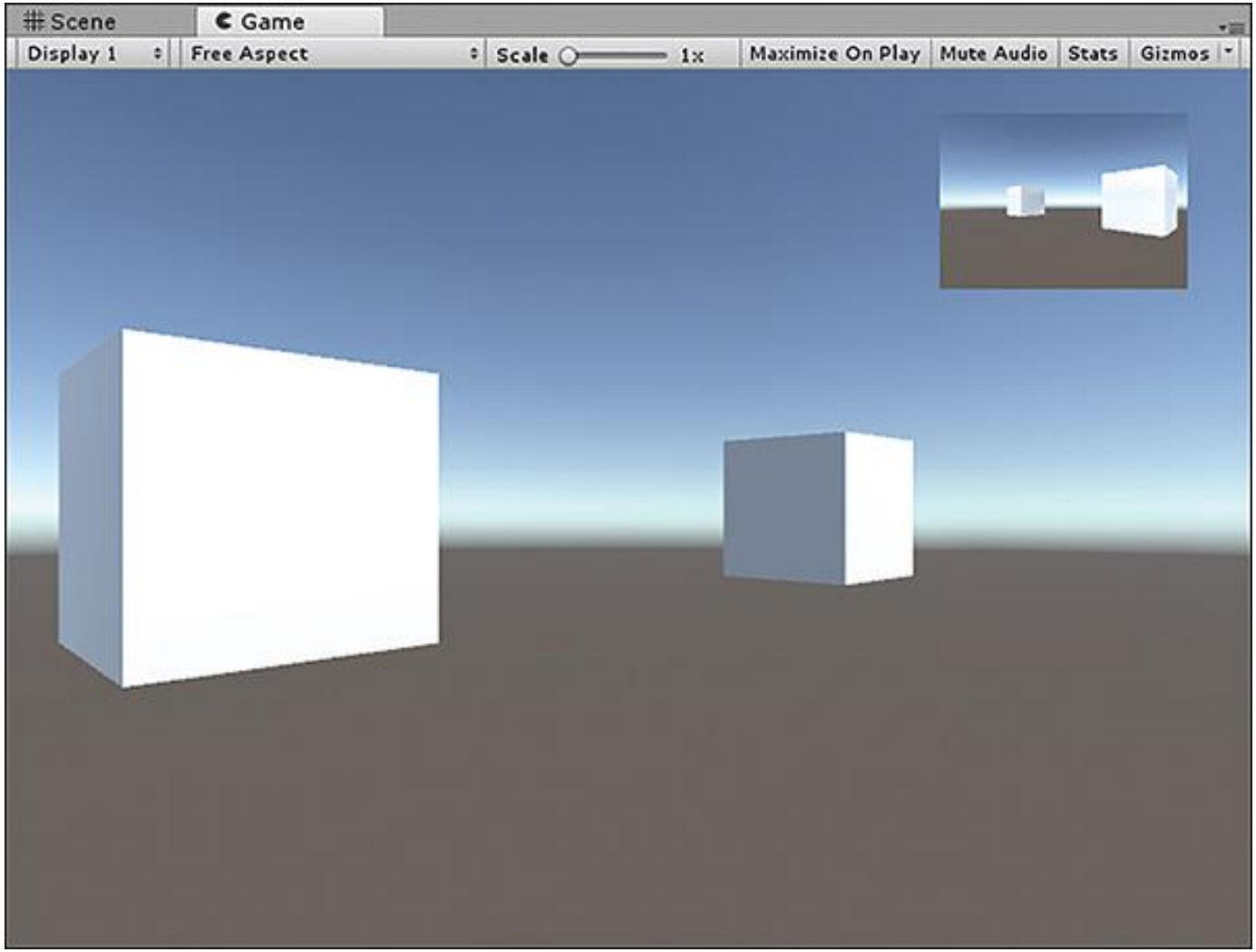


Picture-in-Picture Effect

Picture in picture is a common way to create effects like minimaps. One camera is going to draw over another one in a specific area.

1. Ensure that the Main Camera has a depth of –1. Ensure that the X and Y properties of the camera's ViewPort Rect property are both 0 and the W and H properties both 1.
3. Ensure that the depth of the second camera is 0. Set the X and Y property of the view port to (.75, .75) and set the W and H values to .2 each.
4. Run the scene. Notice how the second camera appears in the upper-right corner of the screen.

Picture-in-Picture Effect



Layers

Layers

With so many objects in a project and in a scene, it can often be difficult to organize them.

Sometimes you want items to be viewable by only certain cameras or illuminated by only certain lights.

Sometimes you want collision to occur only between certain types of objects.

Unity's answer to this organization is layers.

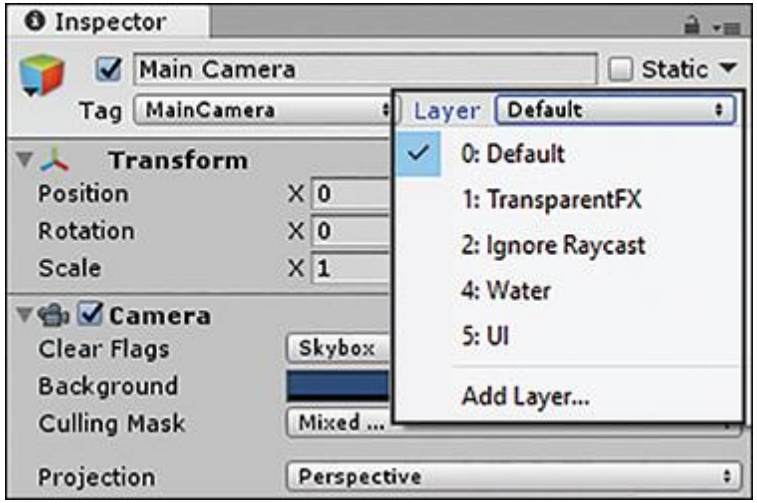
Layers are groupings of similar objects so that they can be treated a certain way.

Layers

Every game objects starts in the Default layer.

You can easily add an object to a layer in the Inspector view. With the object selected, click the Layer drop-down in the Inspector and choose a new layer for the object to be a part of.

Although the current built-in layers aren't exactly useful to you, you can easily add new layers.



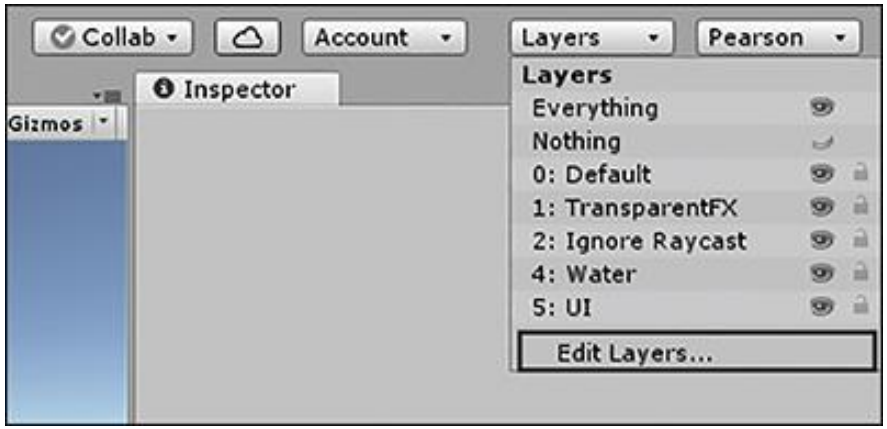
Layers

You add layers in the Tag Manager, and there are three ways to open the Tag Manager:

>With an object selected, click the **Layer** drop-down and select **Add Layer**.

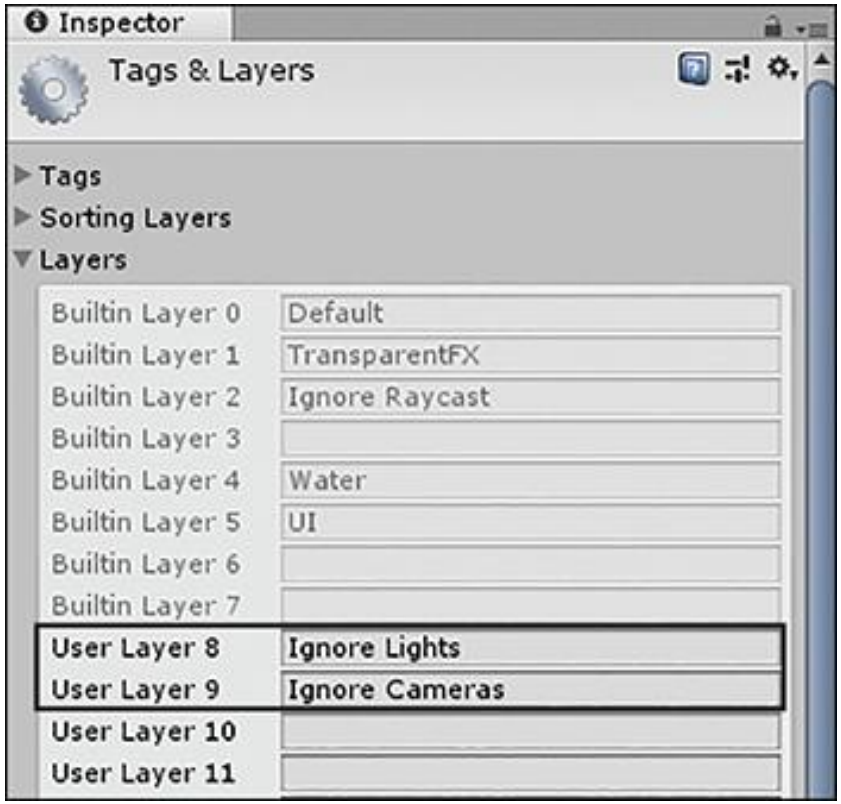
>In the menu at the top of the editor, click **Edit > Project Settings > Tags**.

>Click the **Layers** selector in the scene toolbar and choose **Edit Layers**



Layers

Once in the Tag Manager, just click to the right of one of the user layers to give it a name



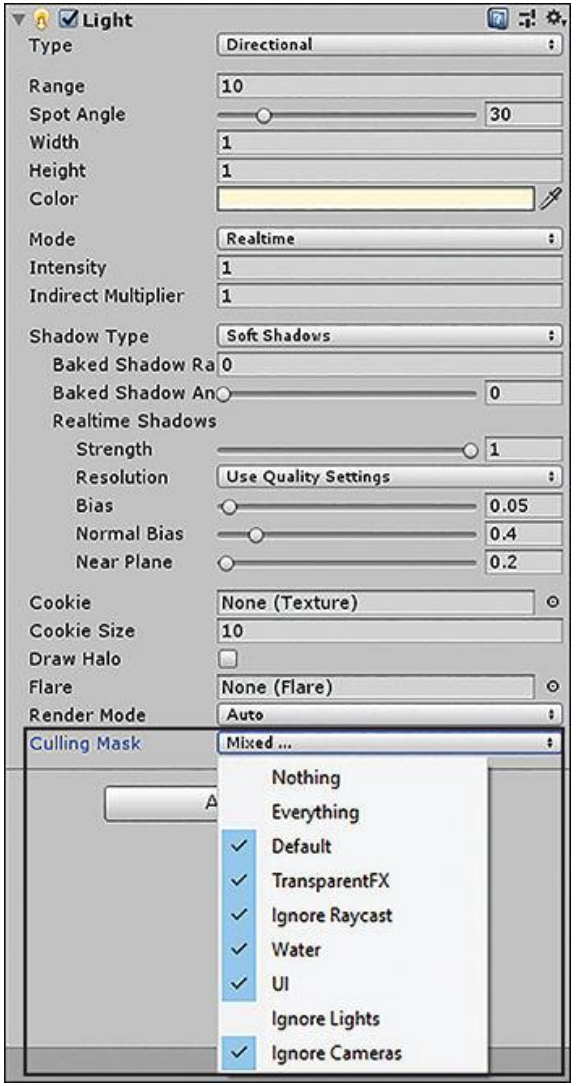
Layers

The first is the ability to hide layers from the Scene view. By clicking the Layers selector in the Scene view toolbar

The second utility of layers is to use them to exclude objects from being illuminated by light. This can prove useful if you are making a custom user interface, shadowing system, or are using a complex lighting system.

Layers

To prevent a layer from being illuminated by a light, select the light. Then, in the Inspector view, click the **Culling Mask** property and deselect any layers that you want ignored



Layers

The last thing to know about layers is that you can use them to determine what a camera can and cannot see.

This is useful if you want to build a custom visual effect using multiple cameras for a single viewer.

Just as previously described, to ignore layers simply click the **Culling Mask** drop-down on the camera component and deselect anything you don't want to appear.

Layers

1. Create a new project or scene. Add two cubes to the scene and position them at (–2, 1, –5) and (2, 1, –5).
2. Enter the Tag Manager using any of the three methods listed earlier and add two new layers: **IgnoreLights** and **IgnoreCameras**.
3. Select one of the cubes and add it to the IgnoreLights layer. Select the other add it to the IgnoreCameras layer.
4. Add a point light to the scene and place it at (0, 1, –7). In the Culling Mask property for the light, deselect the **IgnoreLights** layer.
5. Select the Main Camera and remove the IgnoreCameras layer from its Culling Mask property. Run the scene and notice how only one nonilluminated cube appears.

Summary

- Terrain
- Environments (Skybox, Fog, Water, etc.)
- Character Controllers
- Lights, Cameras and Layers