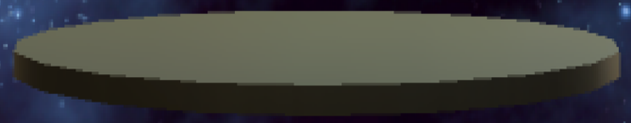
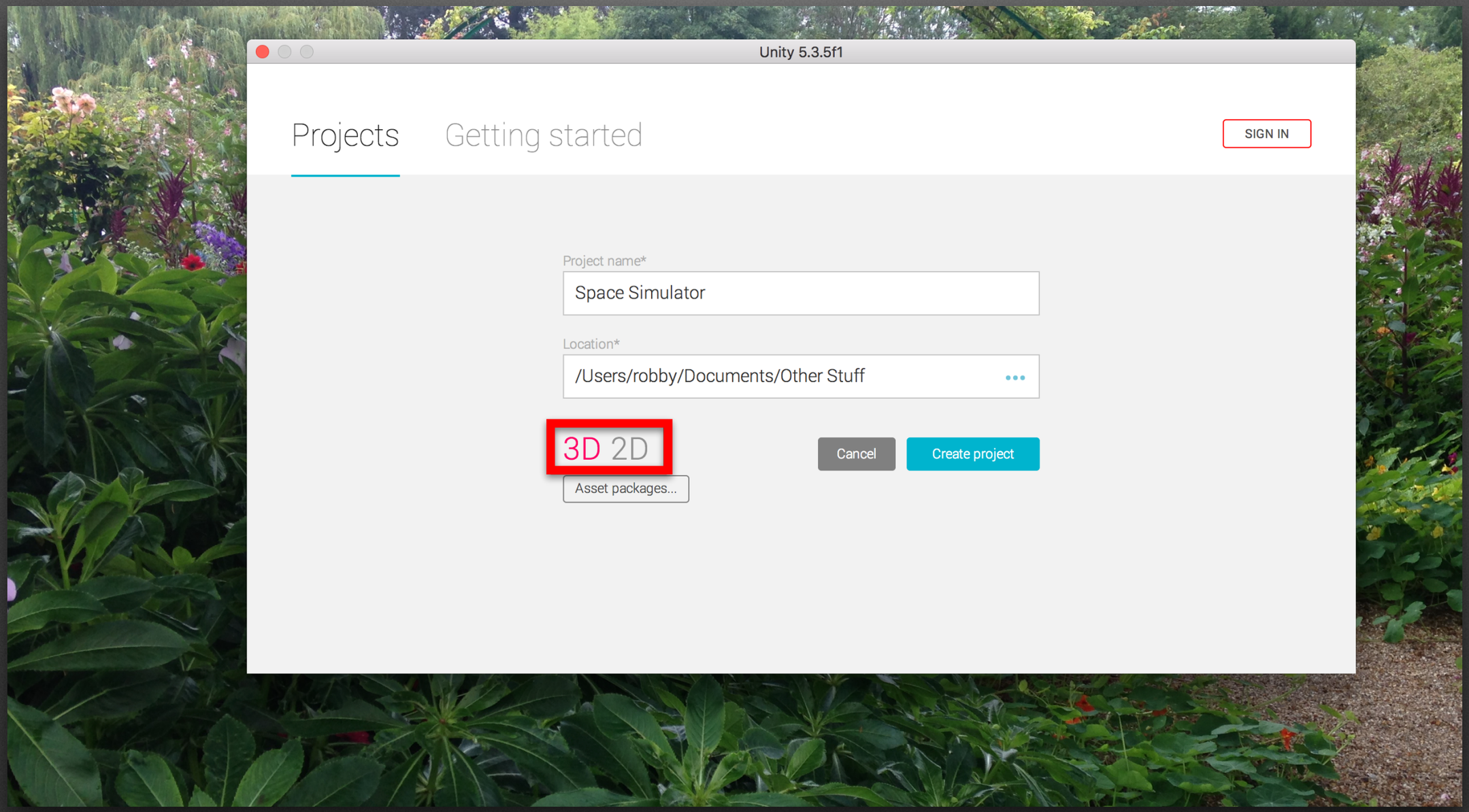
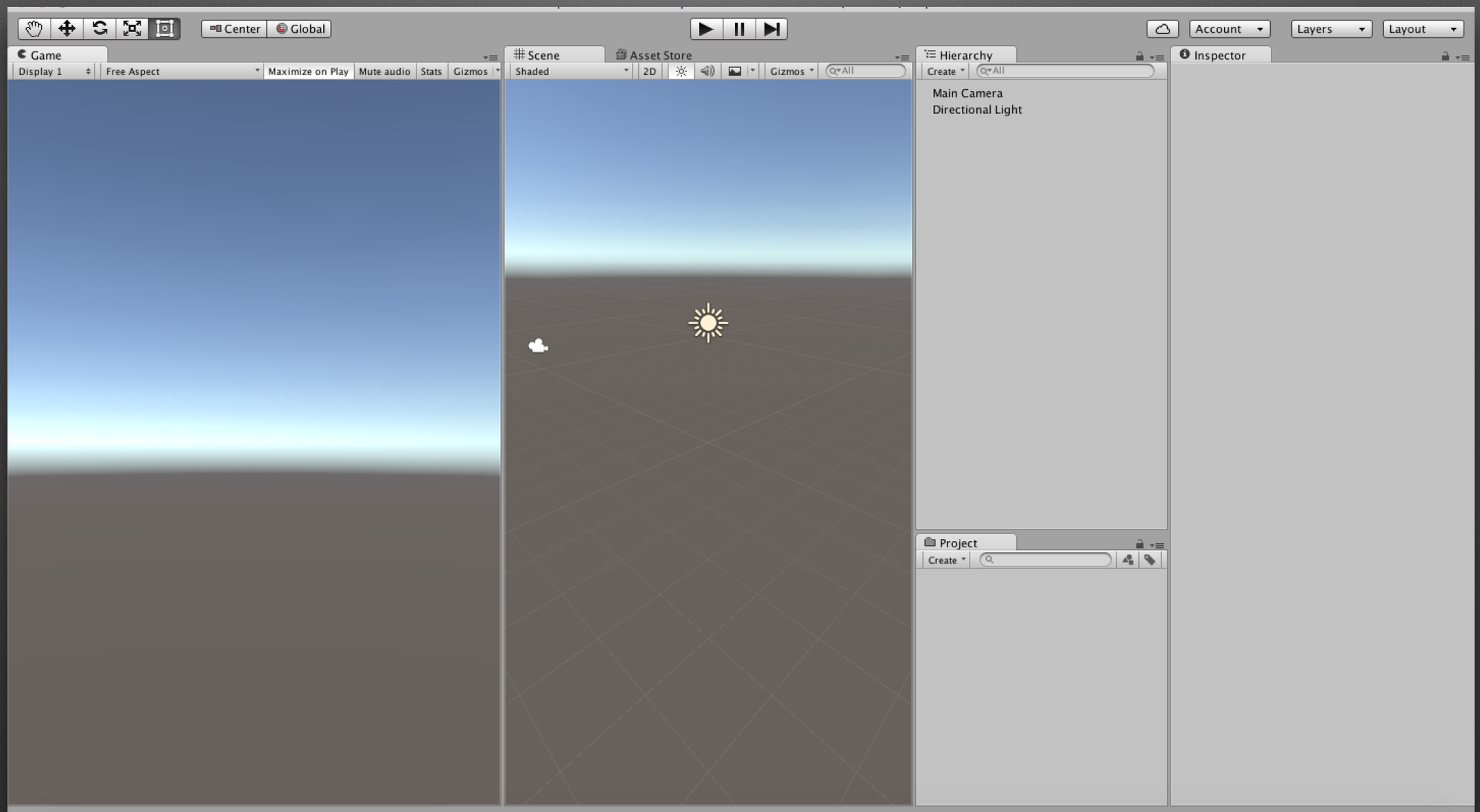
Unity Basics





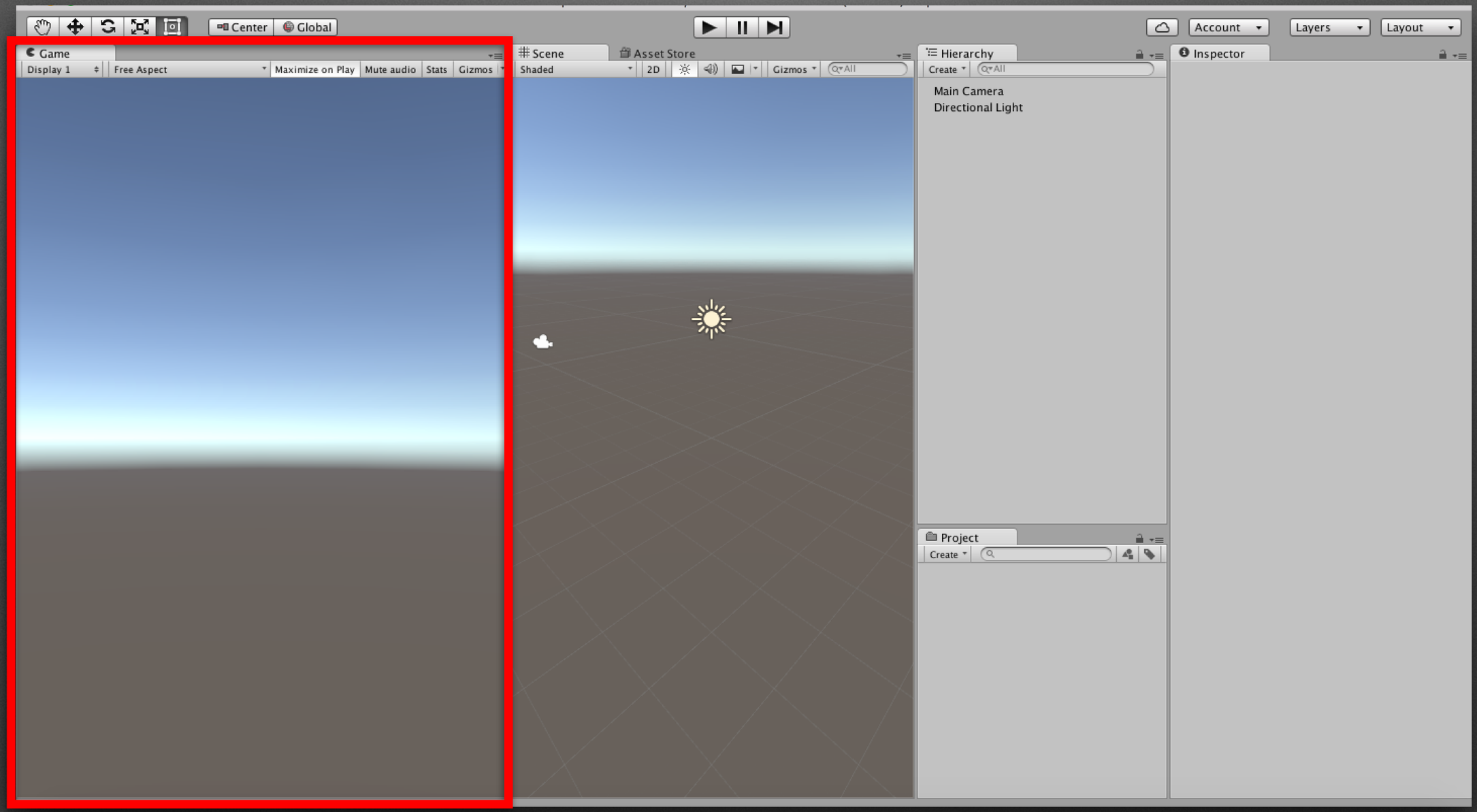
Creating a New Project

To create a new project, simply open Unity and select “File > New Project…”
Name your project, ensure 3D is selected, and click “Create Project”.



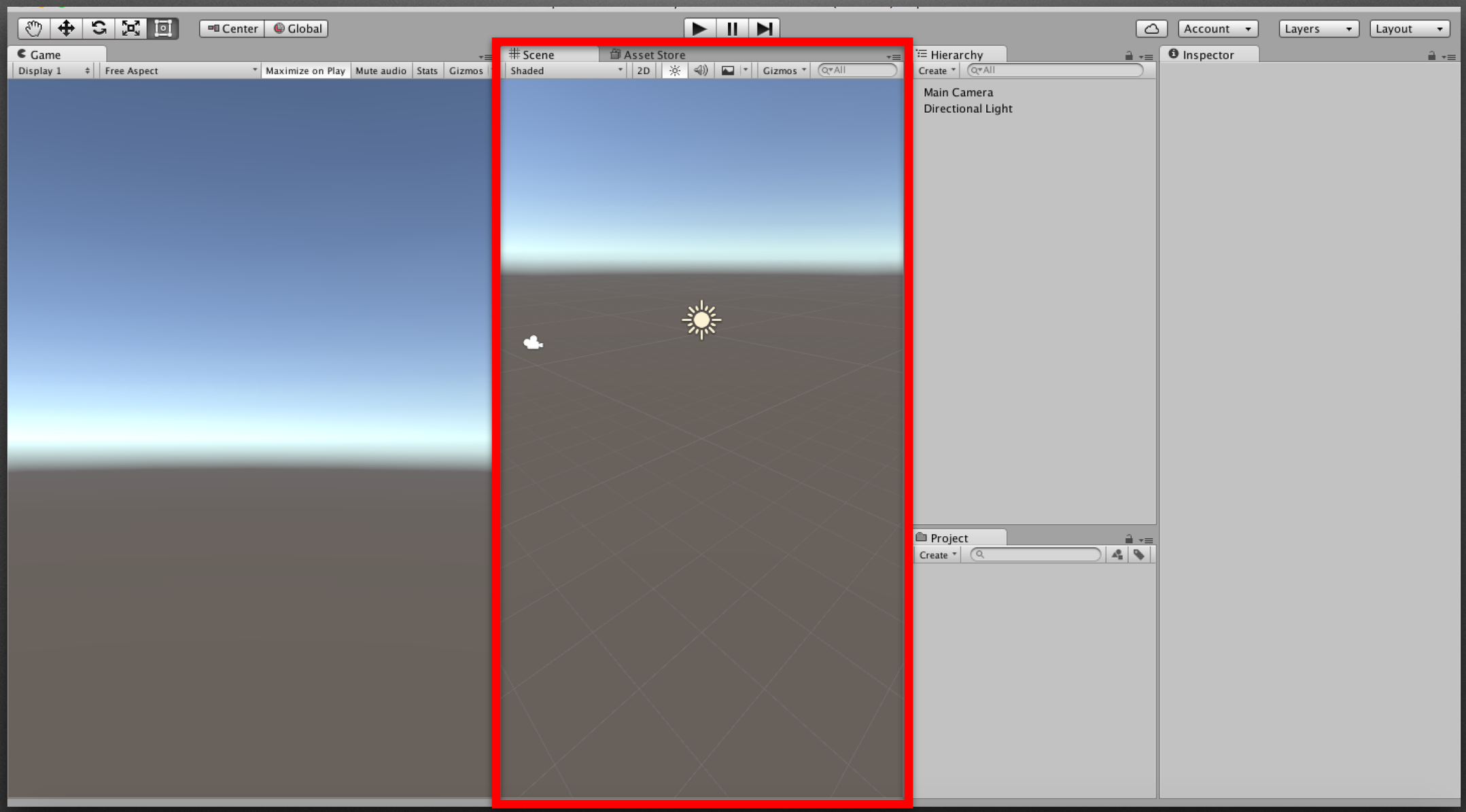
The New Project Screen

A new project will appear with many open tabs. I prefer to arrange my tabs like this, but there’s no right or wrong way.



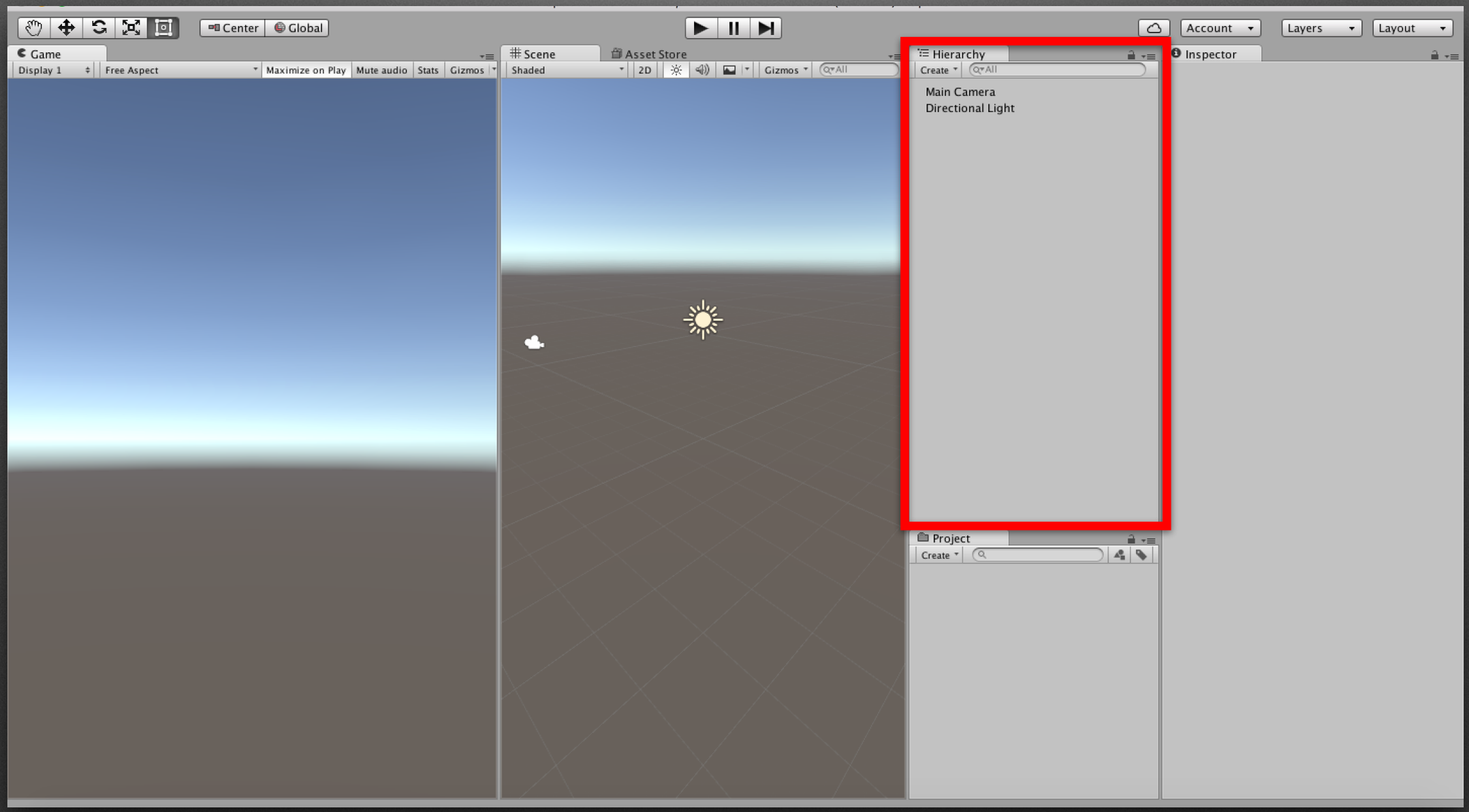
The “Game” Tab

This is where you get a preview of what your game will look like when it begins.



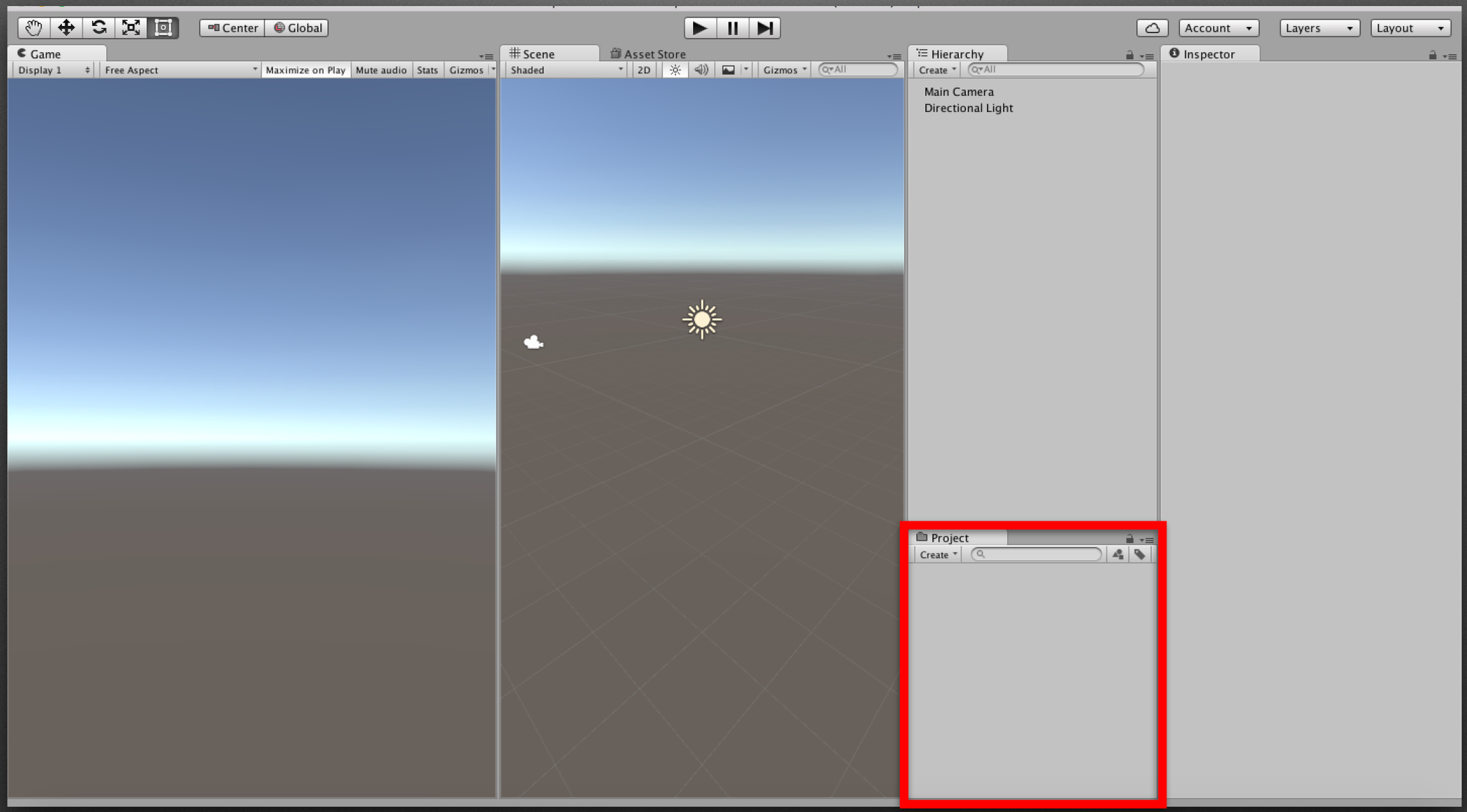
The “Scene” Tab

This is where you will arrange objects in the game world. It’s also a good way to visualize where all of your objects are.



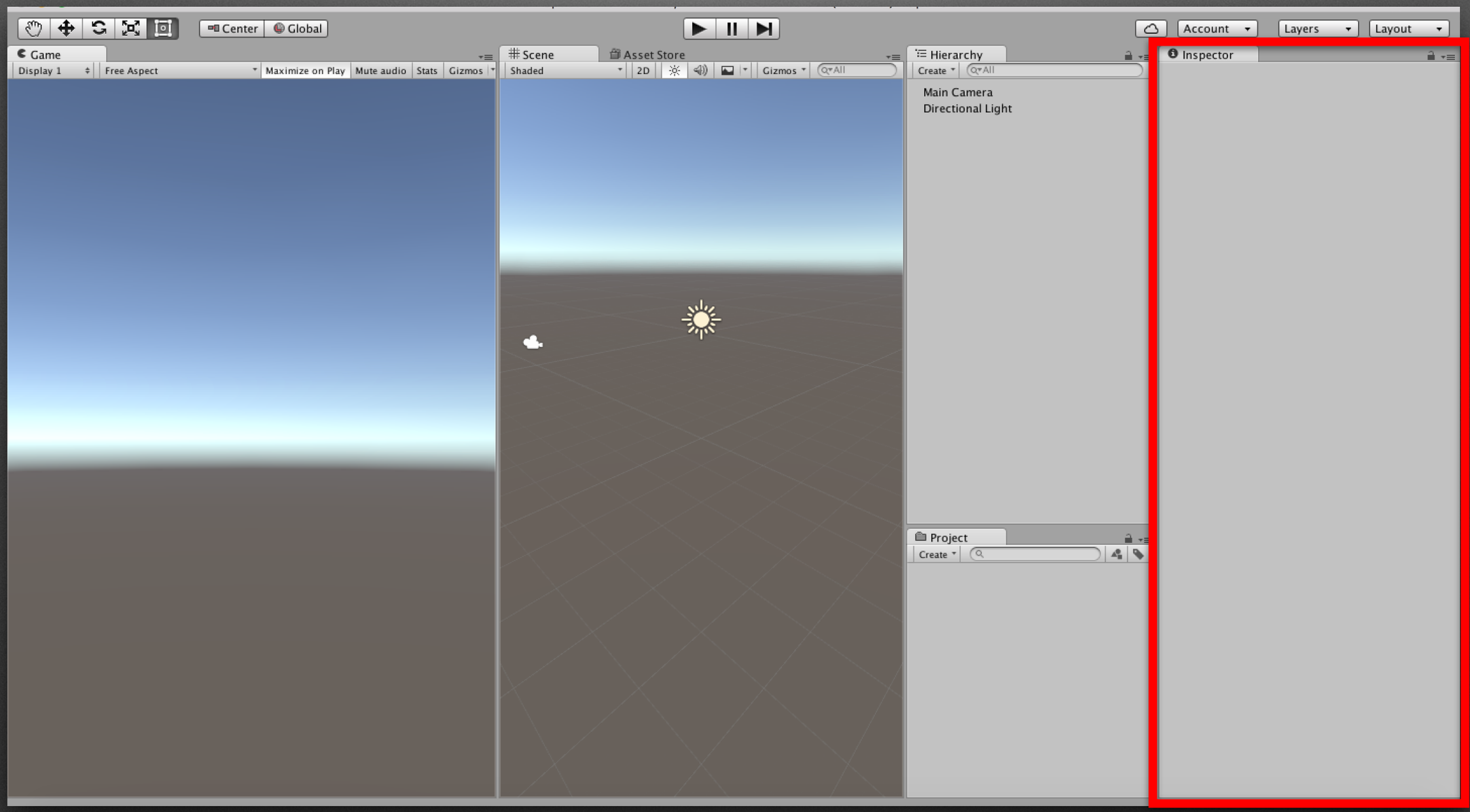
The “Hierarchy” Tab

This is where you will view all of your objects. With any new project, two objects are created by default: a camera and a light.



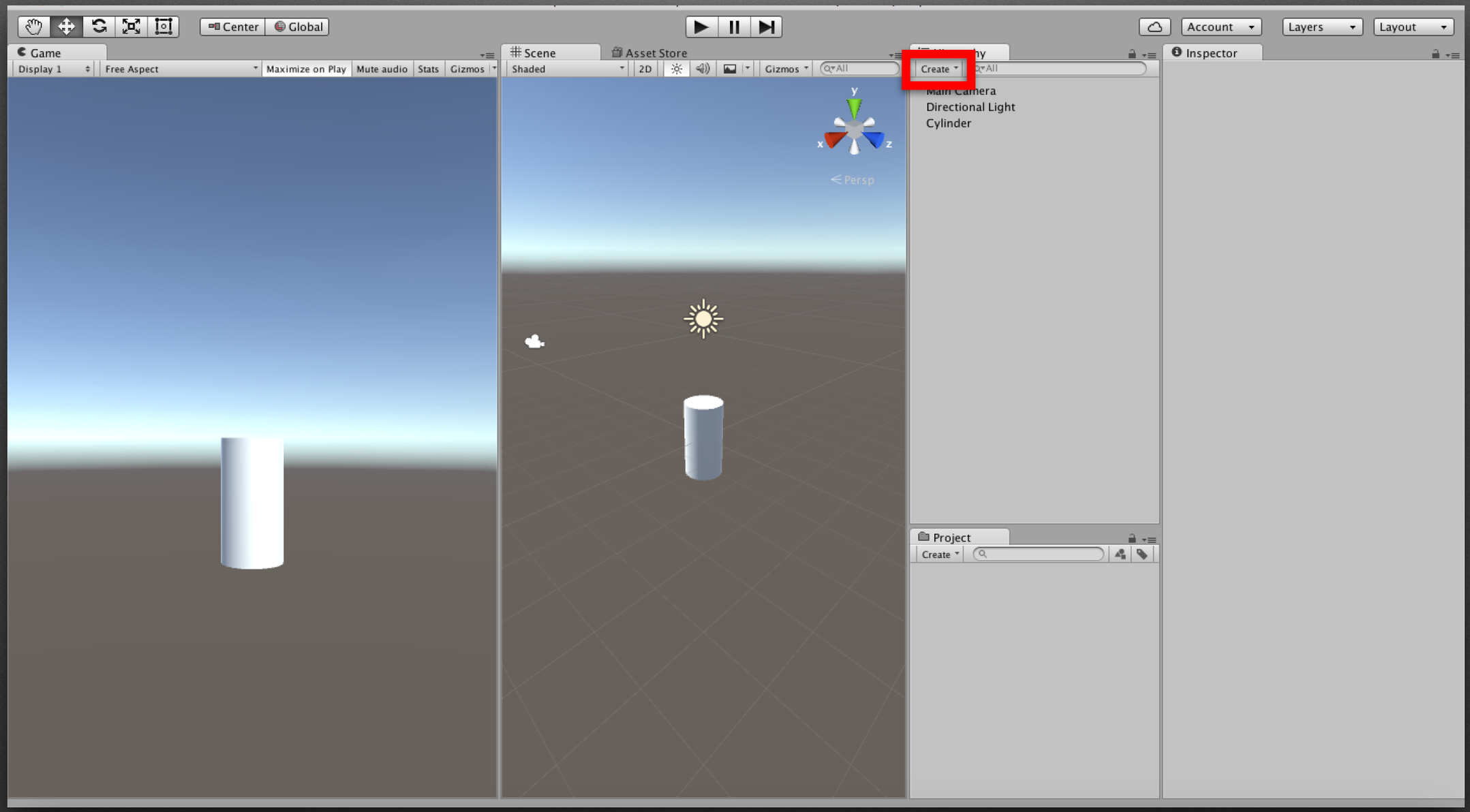
The “Project” Tab

This tab holds all assets your project contains. You will see this populate as you create and include assets for your projects.



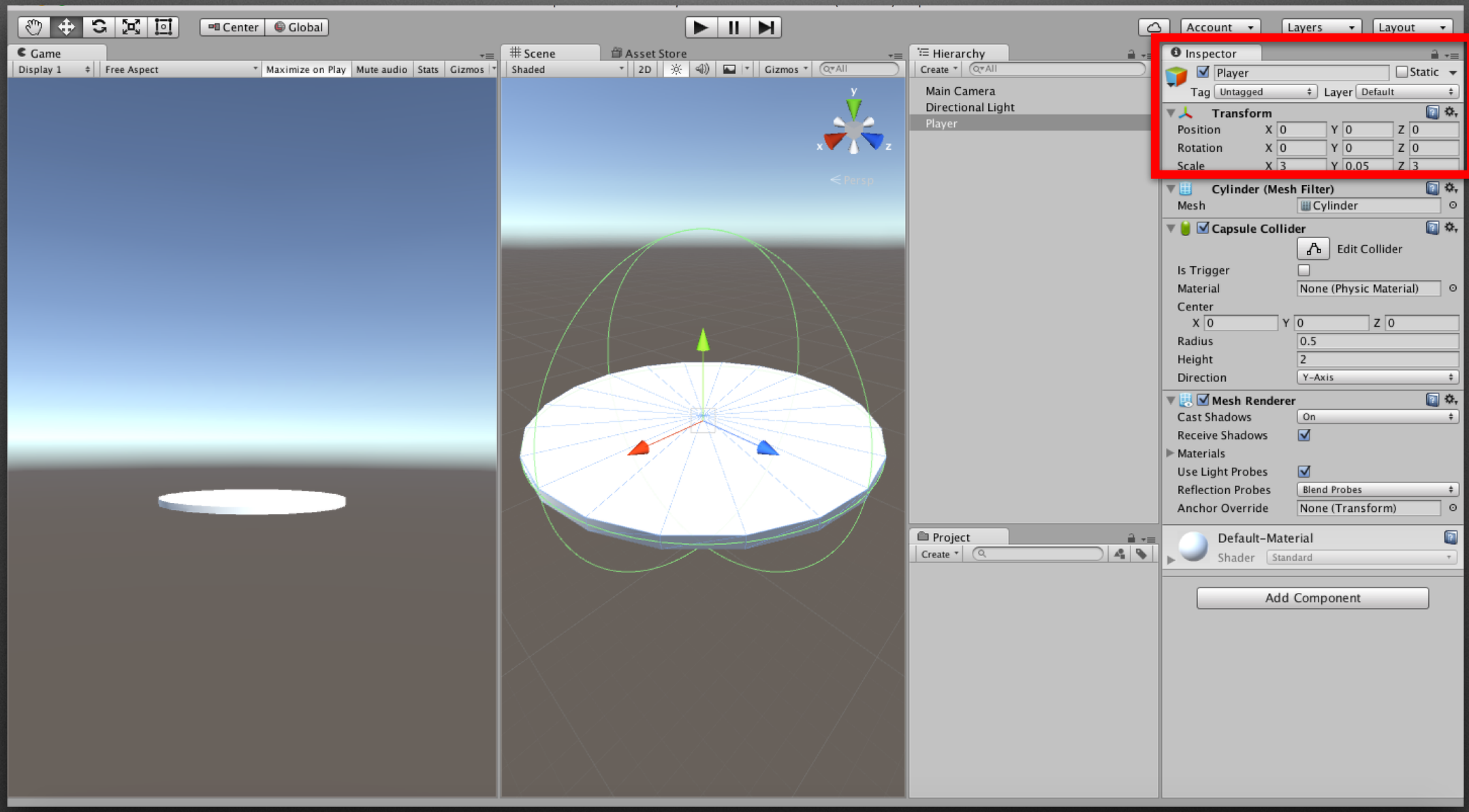
The “Inspector” Tab

The inspector tab holds all the information about whatever asset is selected.
Most of the work done on the Scene is done in this tab.



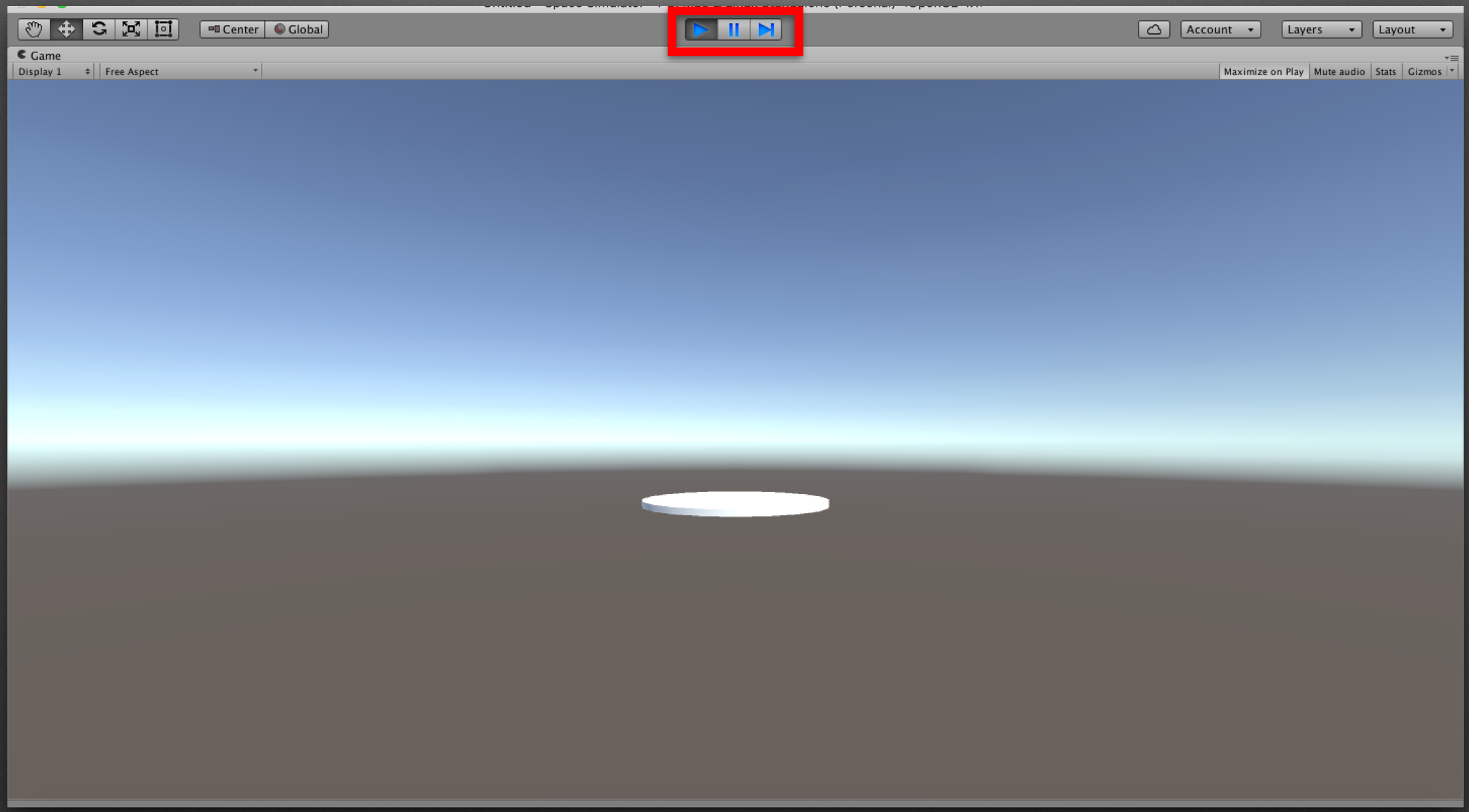
Creating an Object

An object can be created several ways. The easiest way is to simply click the “Create” button and selecting the desired object. In this case, a cylinder is created.



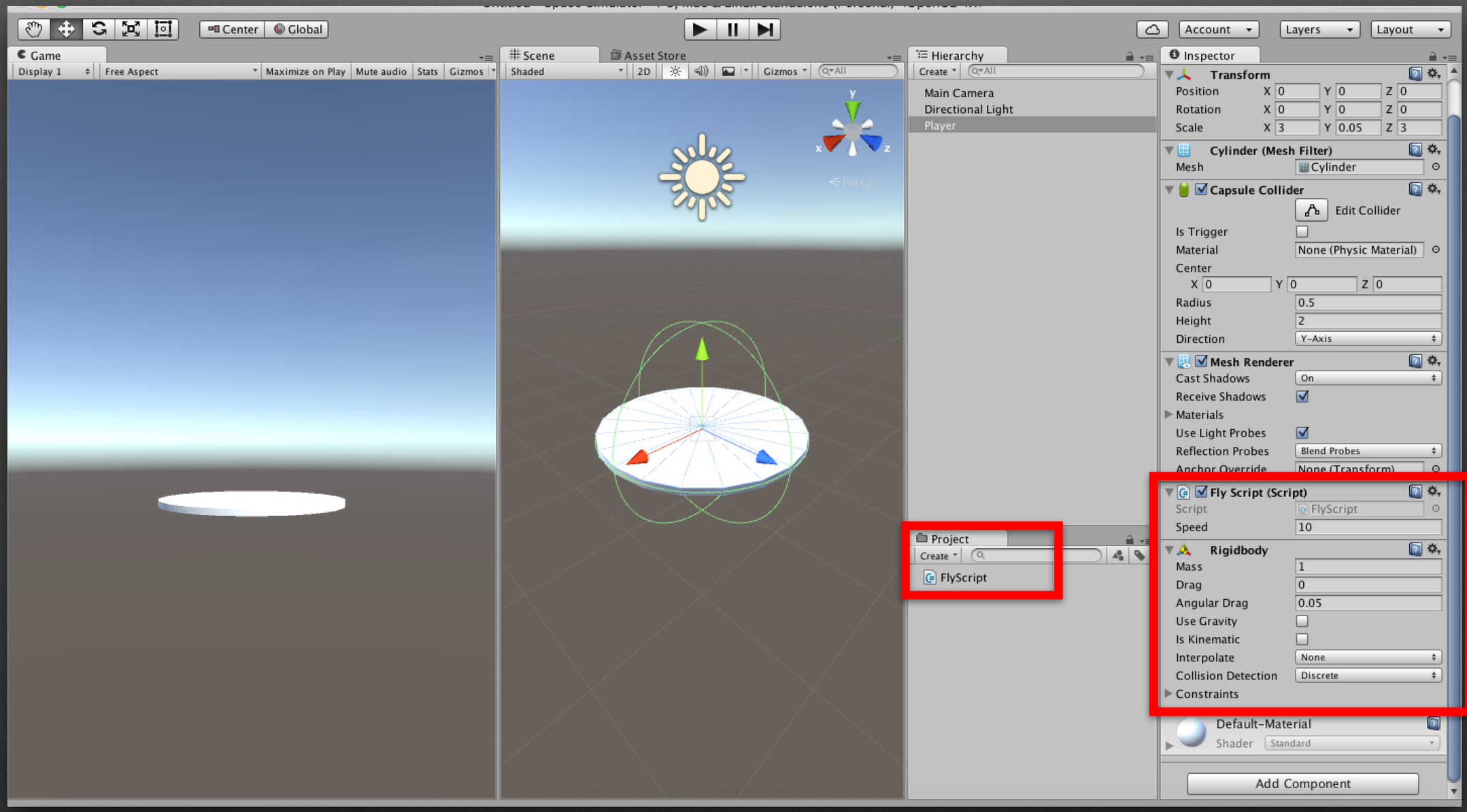
Altering Transform Values

In order to make our player character, we squash the cylinder by editing the transform values. This creates a saucer-shaped disk.



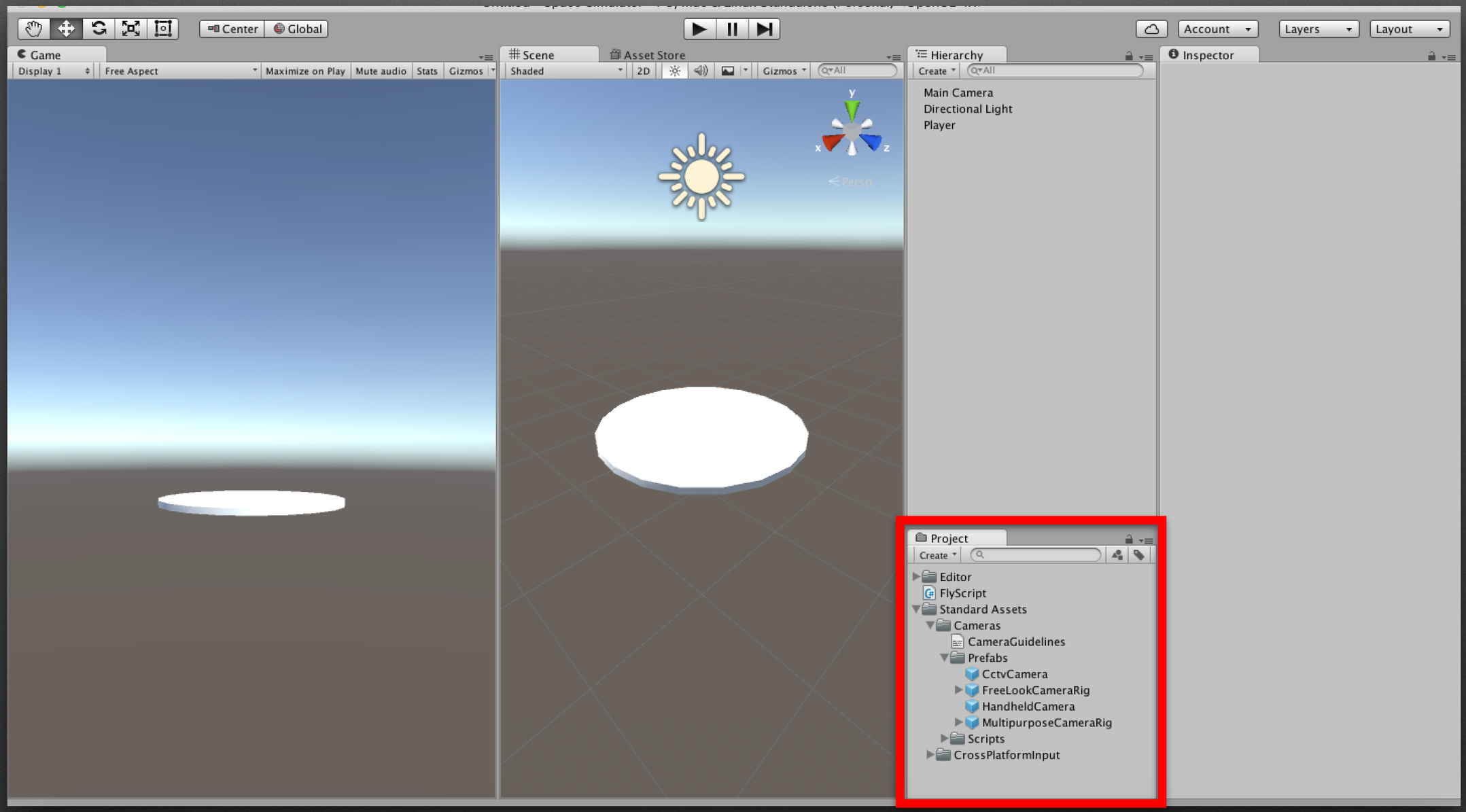
Testing

If you click the “Play” arrow, you can test everything in your game so far. In this case, there’s not much to show, but remember you can do this at any time.



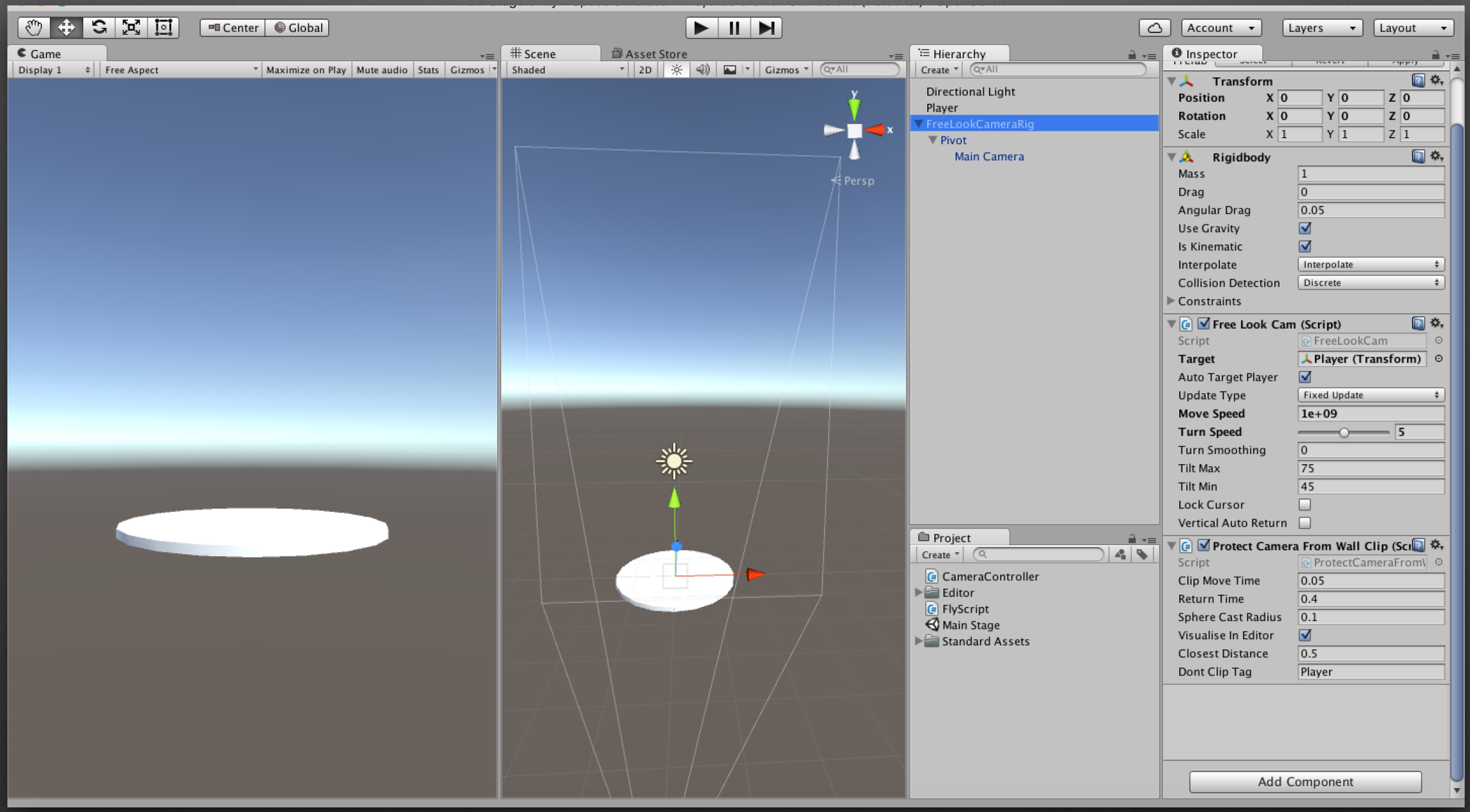
Adding Components

We add two components to our player character now: a Rigidbody, which gives the physics engine access to the object, and a script to allow it to fly. Note that (since I created this script myself) it appears in my project list.



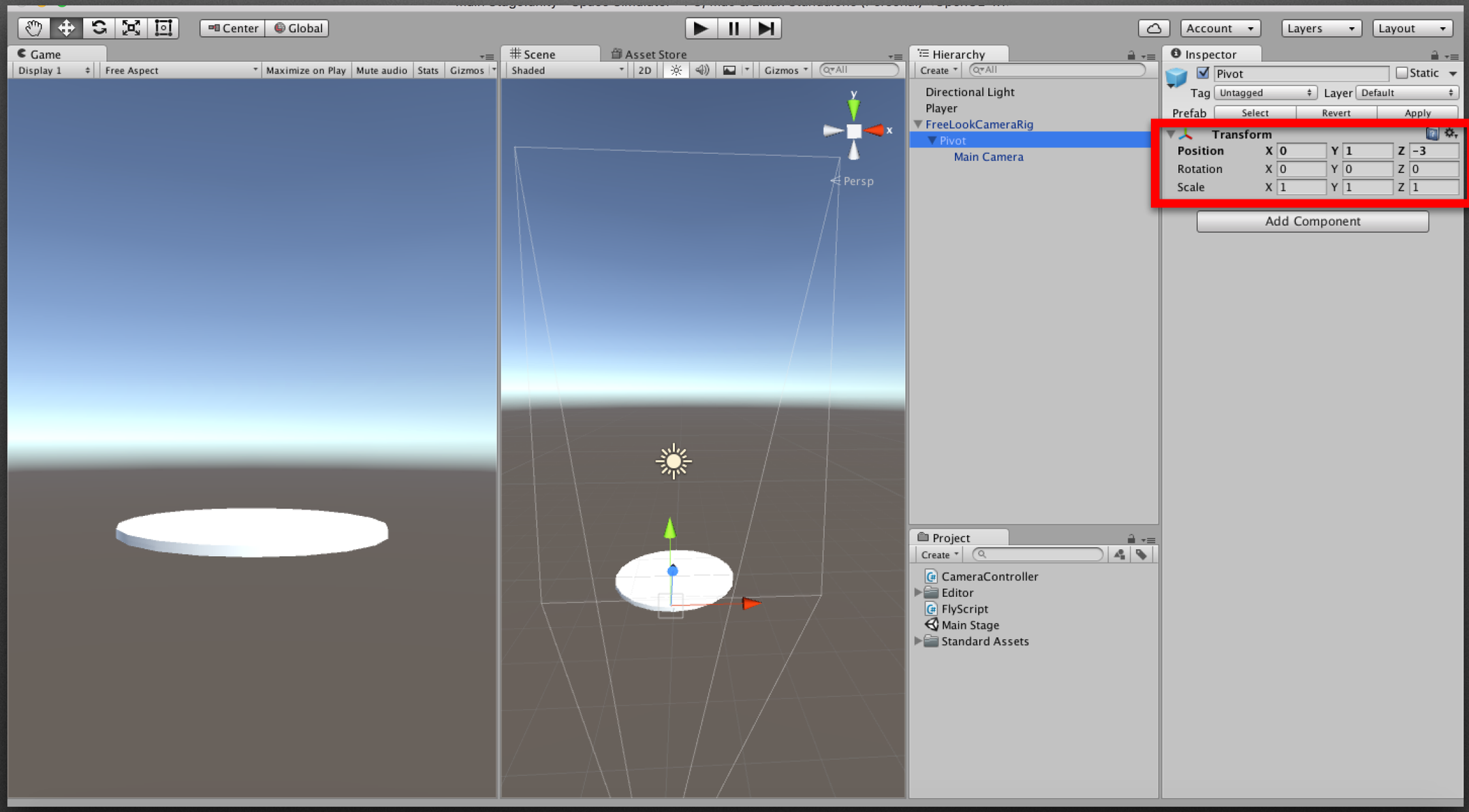
Importing Assets

Assets can be imported by clicking “Assets > Import New Assets…”
You can also import standard assets from Unity by selecting the “Assets > Import Package” option. I have imported the “Cameras” package.



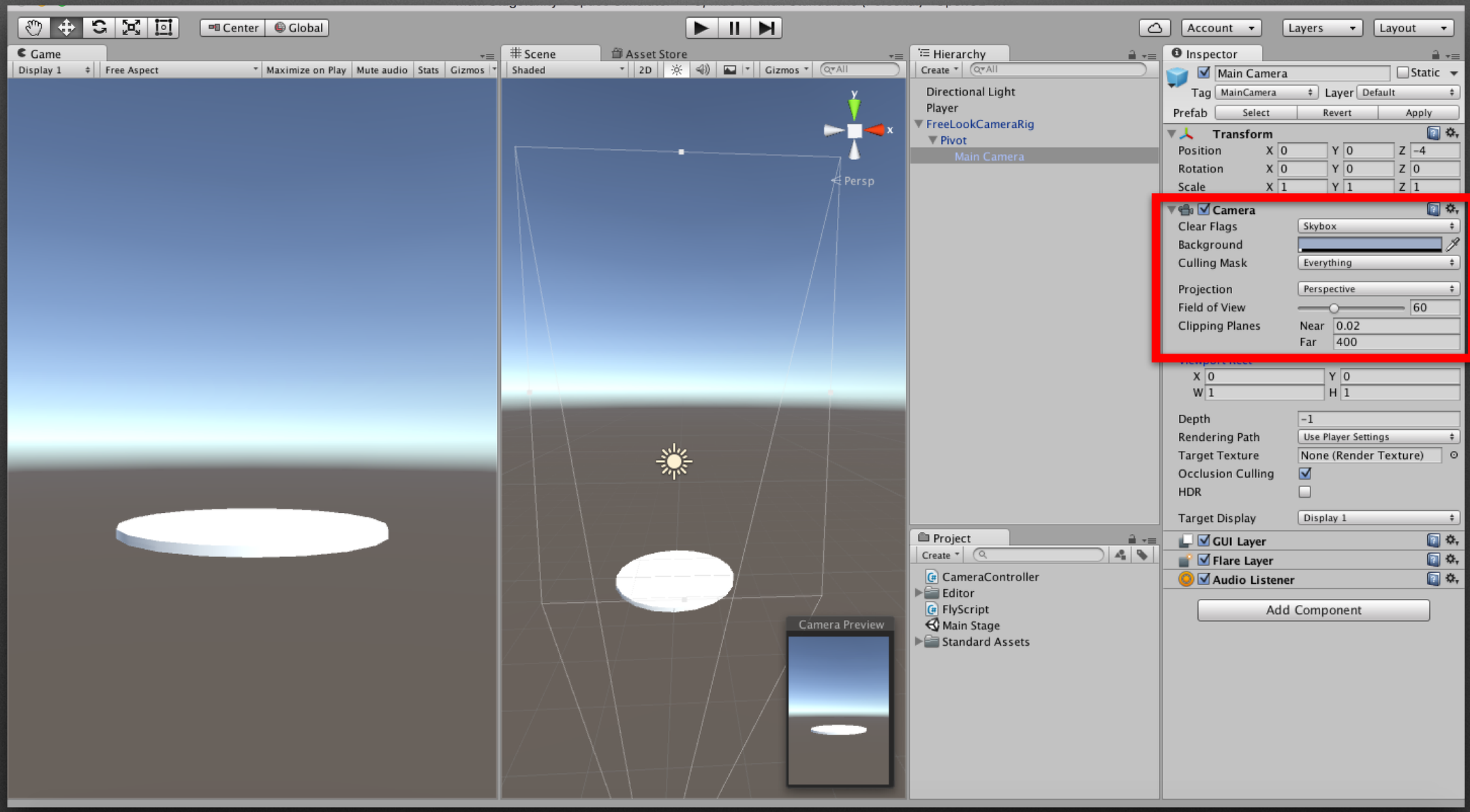
Camera Set-up

Delete the “Main Camera” object created when the project was created. Replace it with a “FreeLookCameraRig”. Ensure your player character is named “Player”.



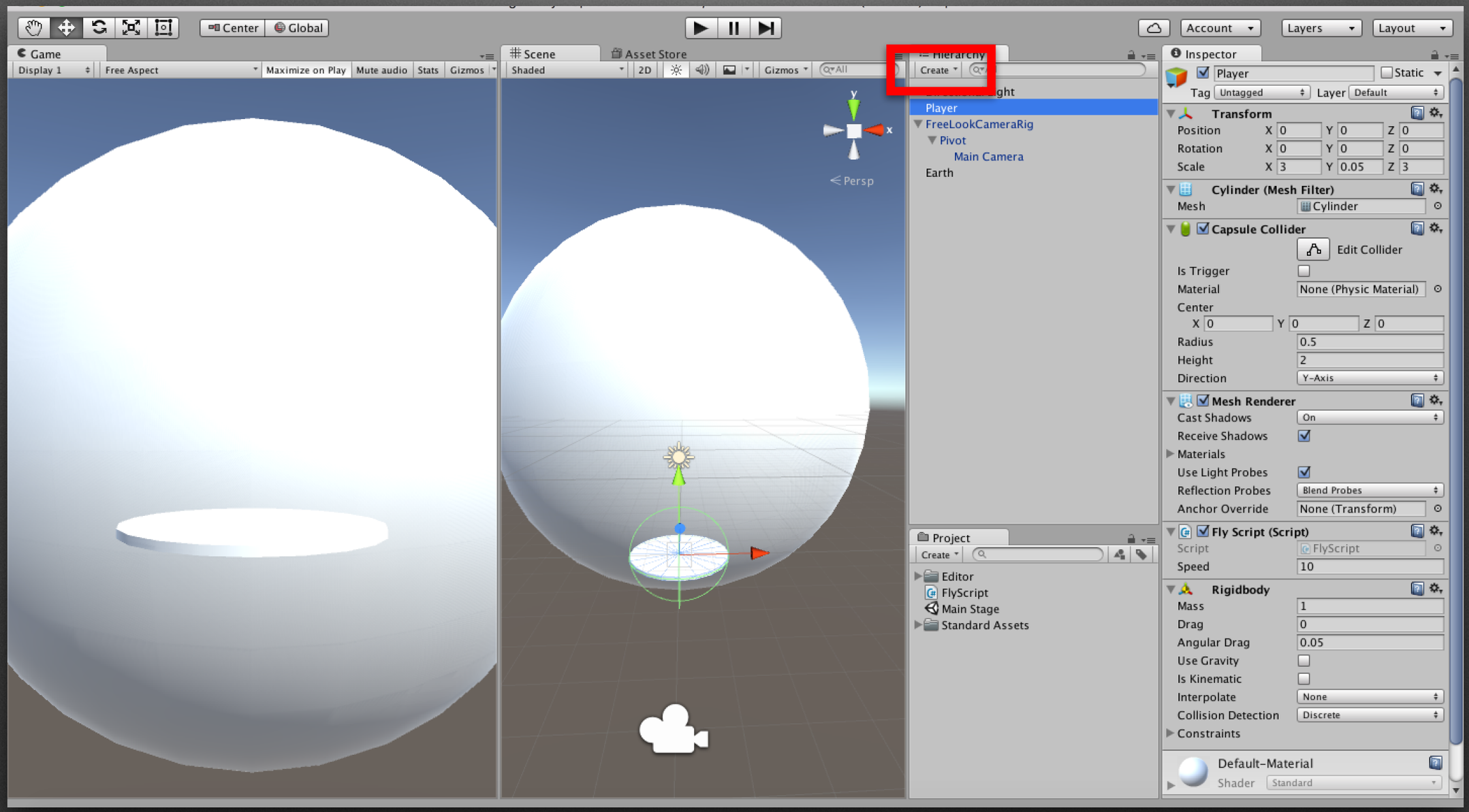
Pivot Editing

Set the position of your pivot to where you want the camera to look from. Note that this is relative to its “parent” object in the hierarchy.



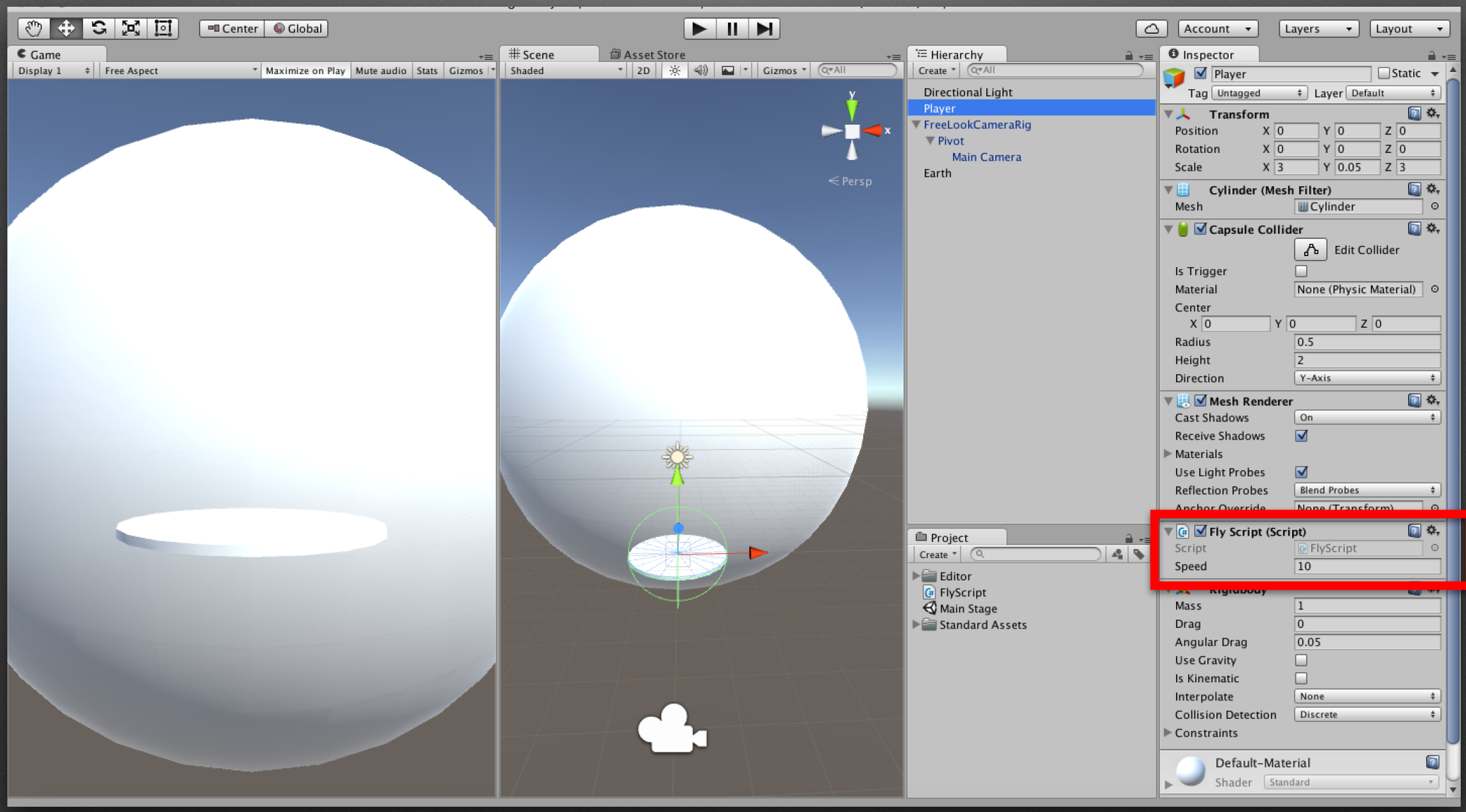
Checking Camera Settings

Making sure the clipping planes are set appropriately, and the projection is set to “Perspective.” (Note: We will edit these again later.)



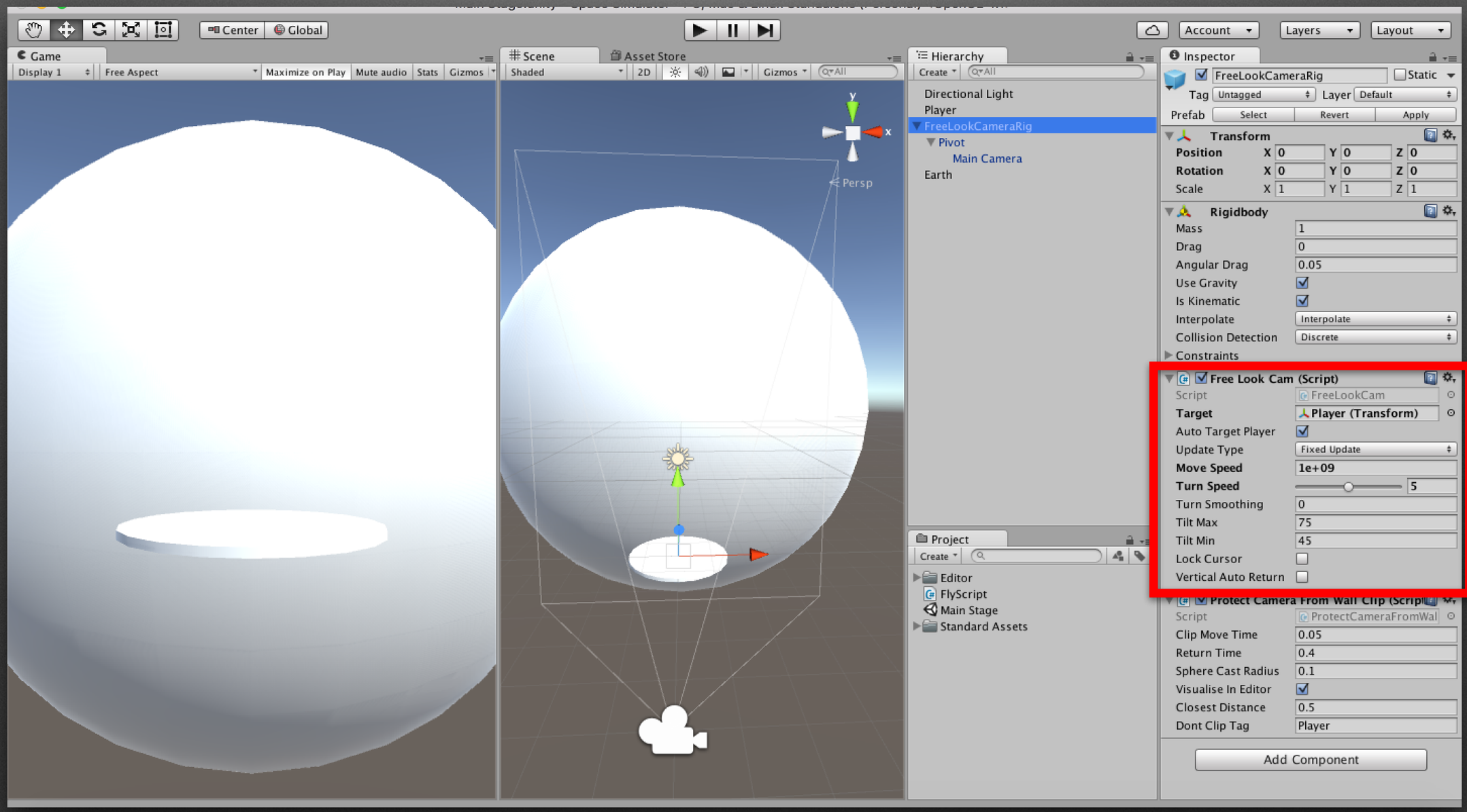
Adding Planets

We create a new Sphere from the “Create” menu. We adjust its “Scale” Transforms to taste (Mine are set to 100 for X, Y, and Z).



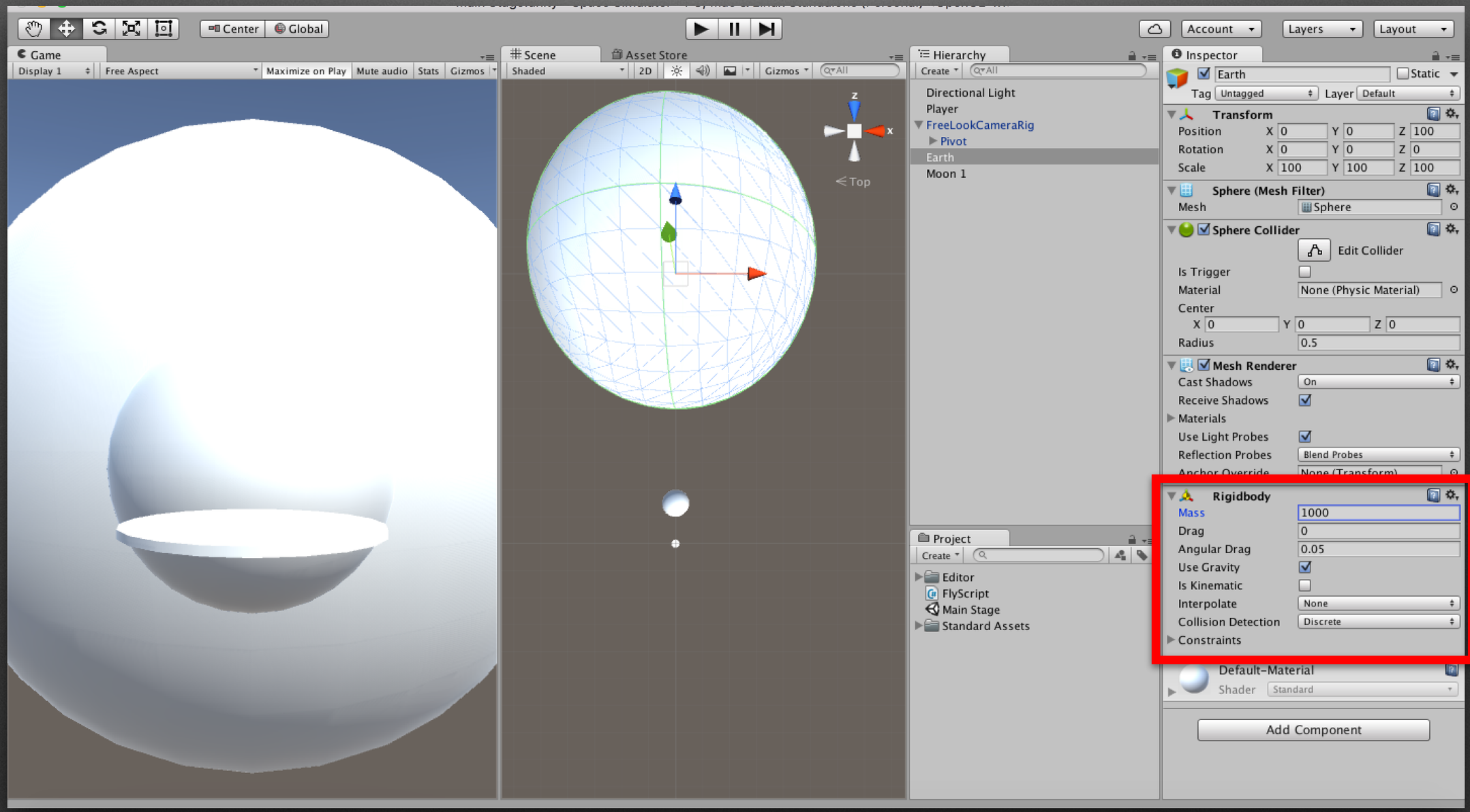
Adding Player Scripts

Attaching the “FlyScript” will allow you to move your player. You’ll notice the camera is somewhat slow to respond. We’ll fix that next.



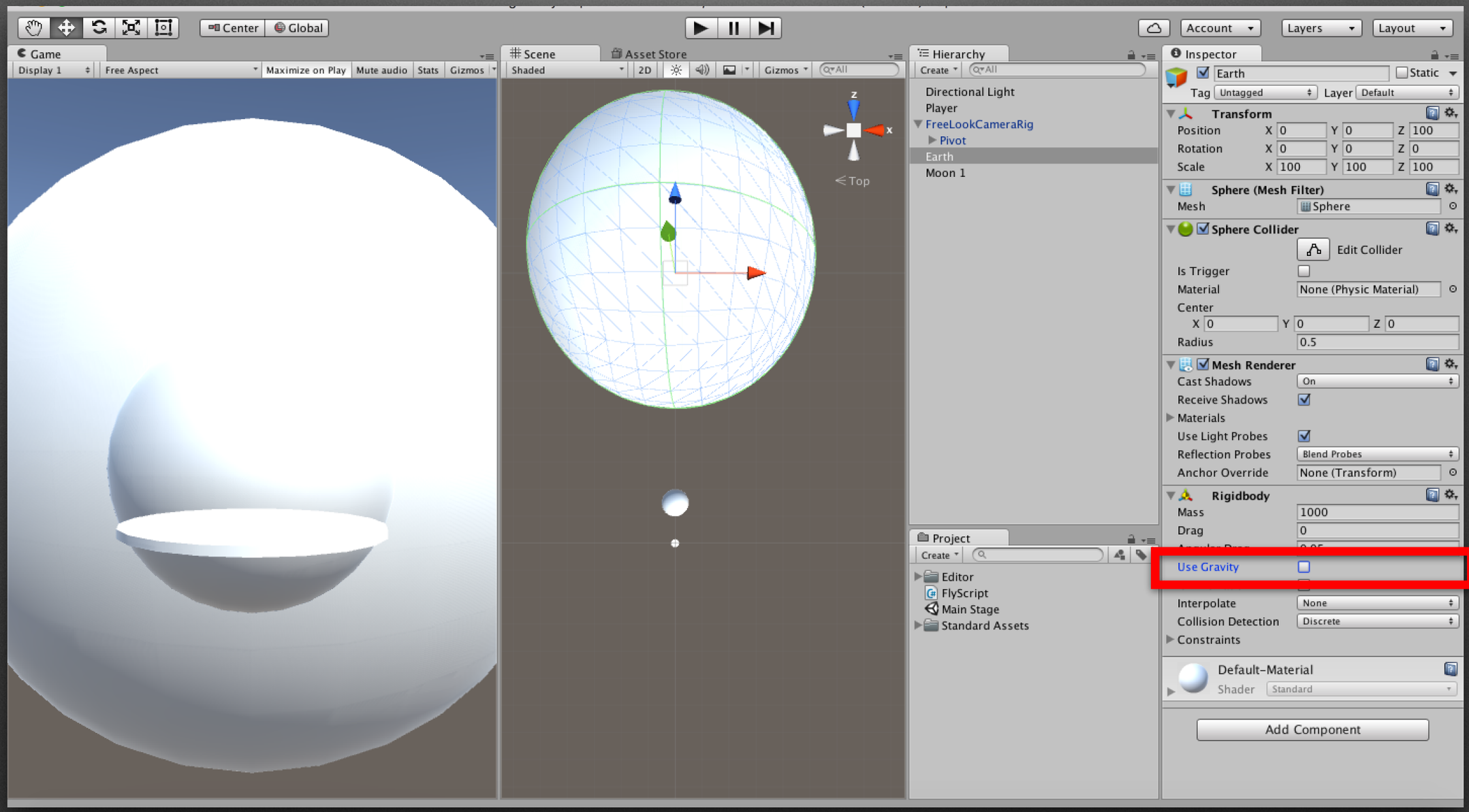
Adjusting Camera Speed

Changing the Camera Speed (and targeting the player) changes the way the camera reacts to mouse movements, making it feel more natural.



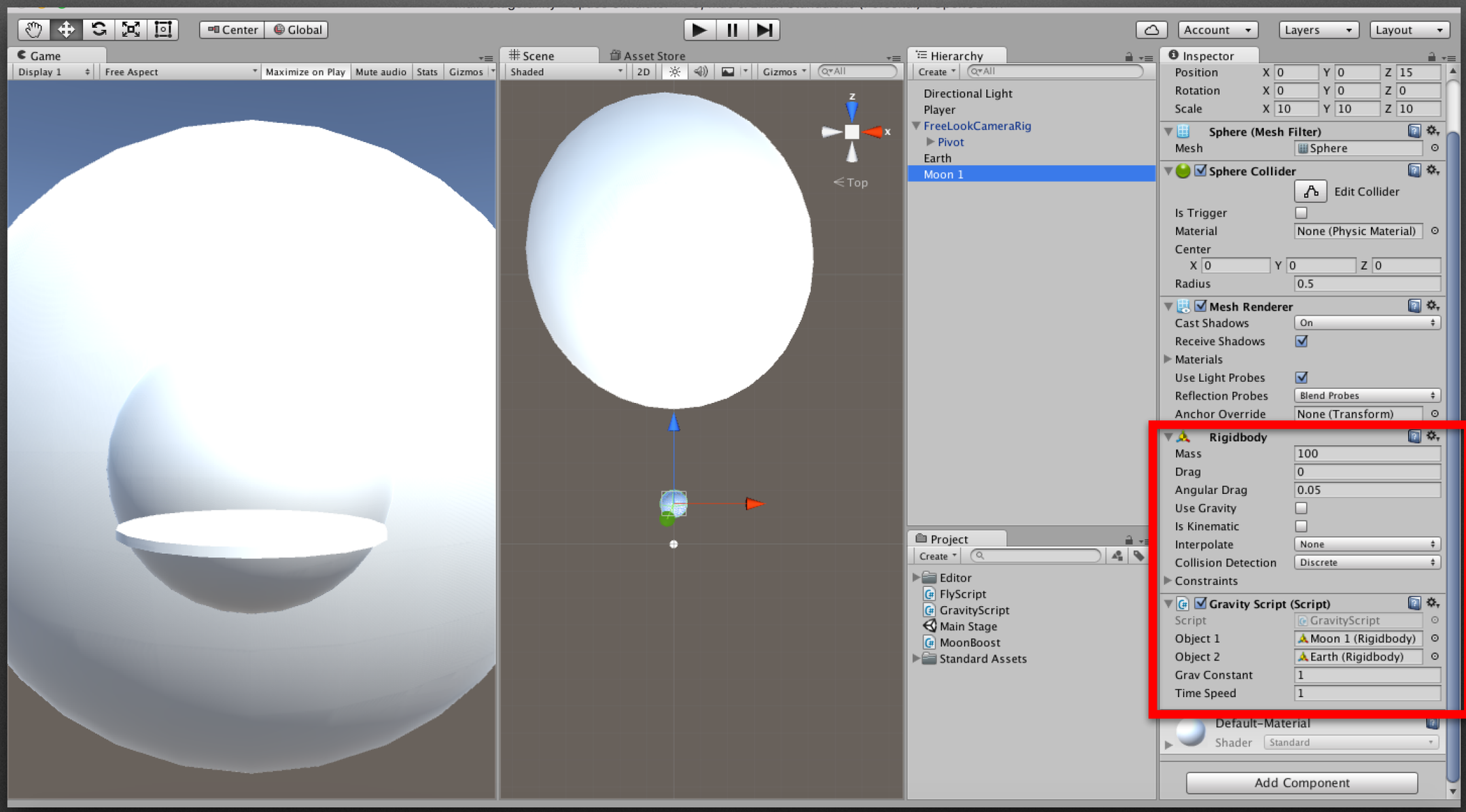
Adding the Moon (and Gravity)

Adding a Rigidbody and a high mass to our Earth is going to allow us to add gravity. We also create another Sphere, named the moon. We adjust the position transform of both to prevent all the objects from sitting on top of each other.



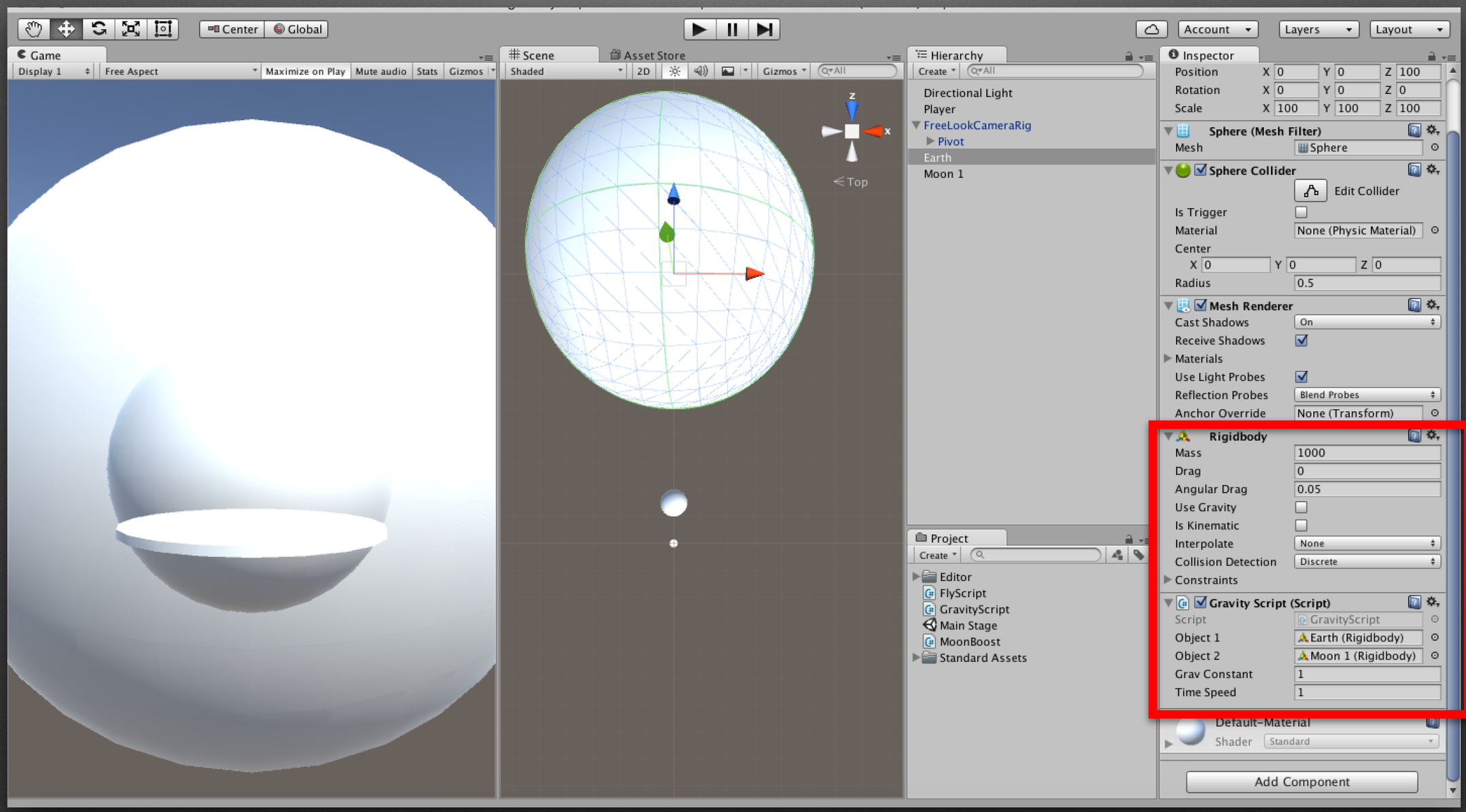
Altering Gravity

Unfortunately, the default gravity in Unity acts as though objects were on Earth, not in space. We'll have to turn theirs off and create our own. Uncheck "Use Gravity" on both the Earth and the Player object.



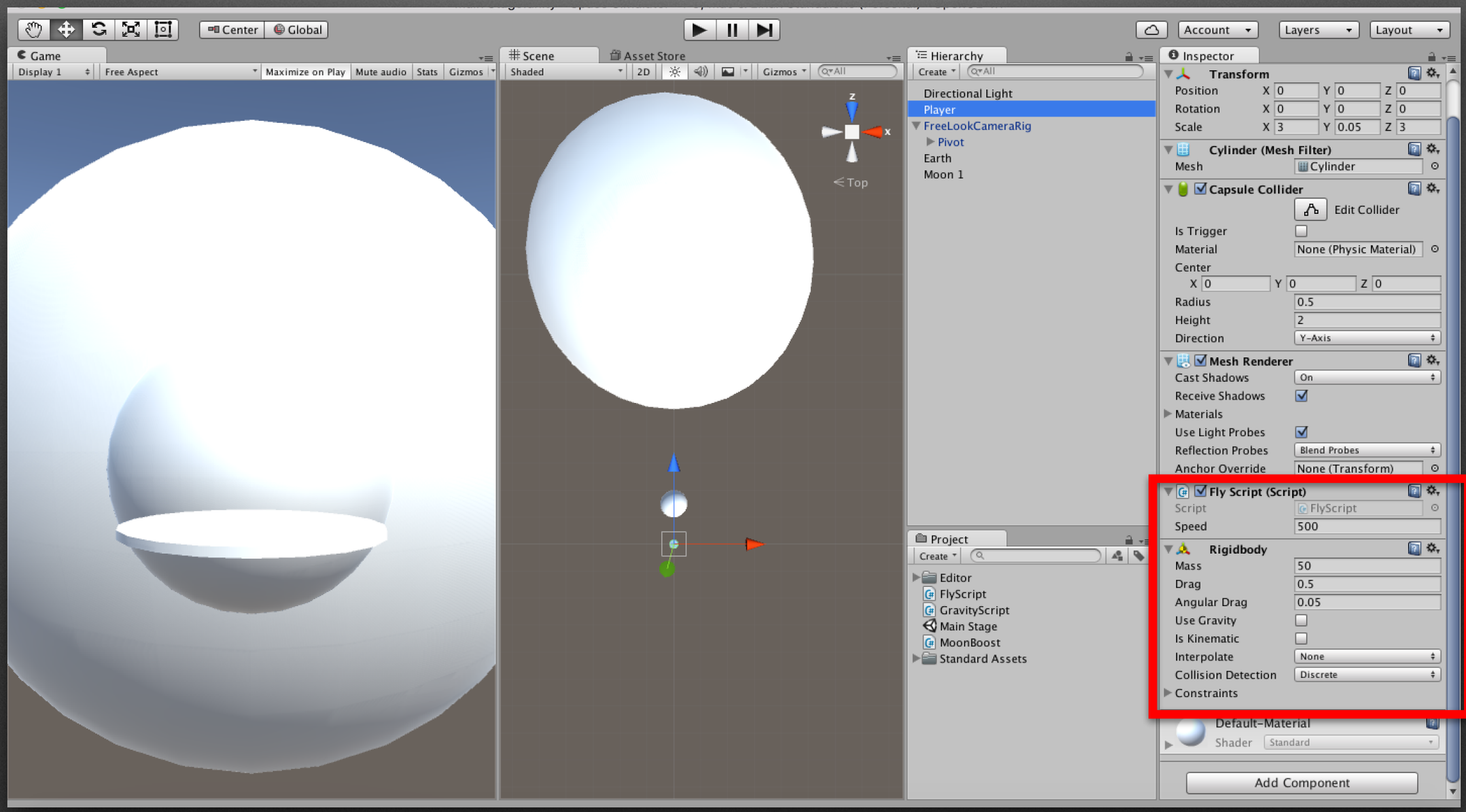
Altering Gravity (Pt. 2)

We add a Rigidbody to the Moon, uncheck “Use Gravity,” and add our new script, “Gravity Script”. This simply creates a scaling force between two objects. Make sure to drag the two objects you want to attract on another to the “Object 1” and “Object 2” fields by dragging them in.



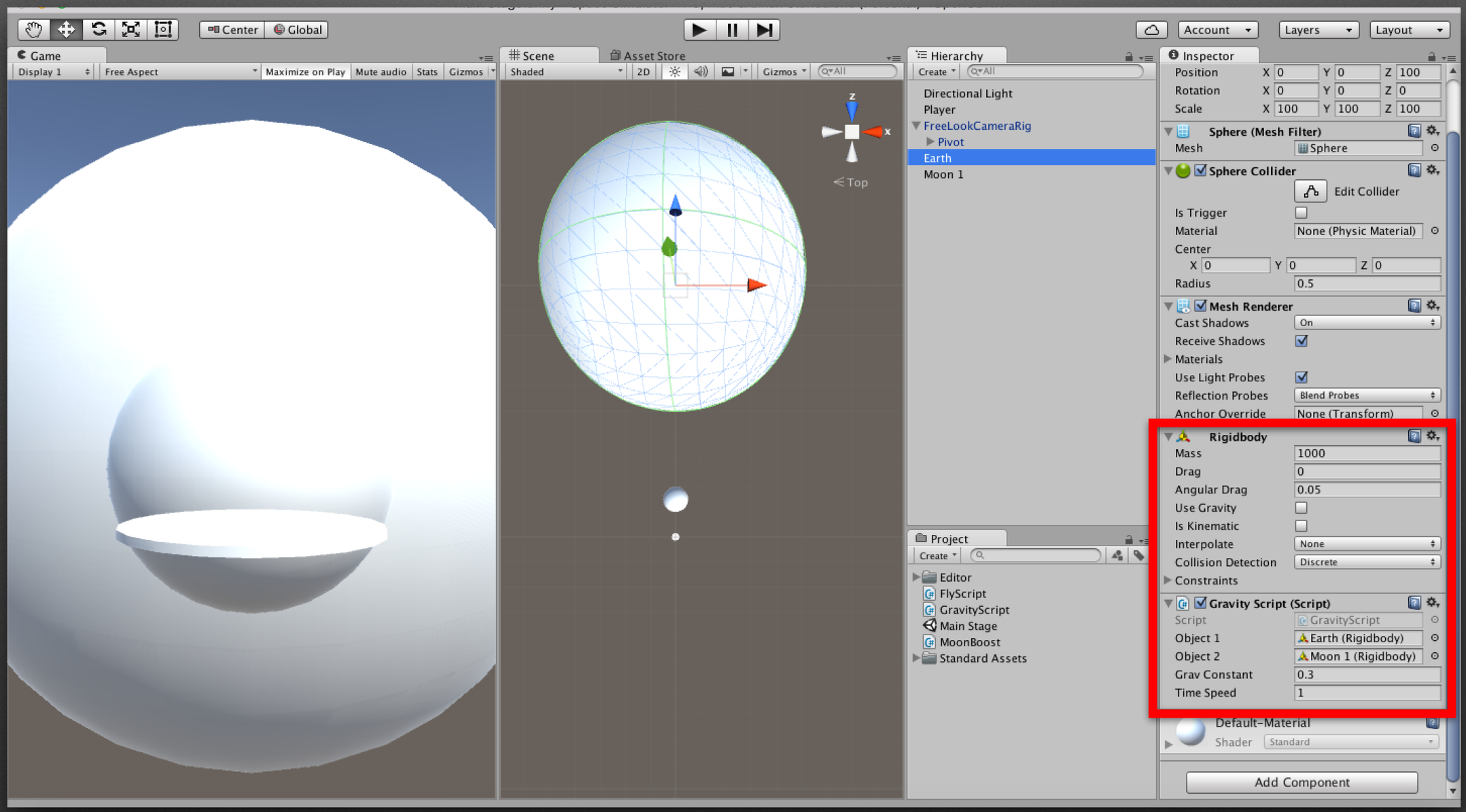
Altering Gravity (Pt. 3)

We now add the same script to the Earth. Be careful to put the object the script is attached to first.
(NB: This is a very rough script I wrote. Those of you with coding experience can try to do better!)



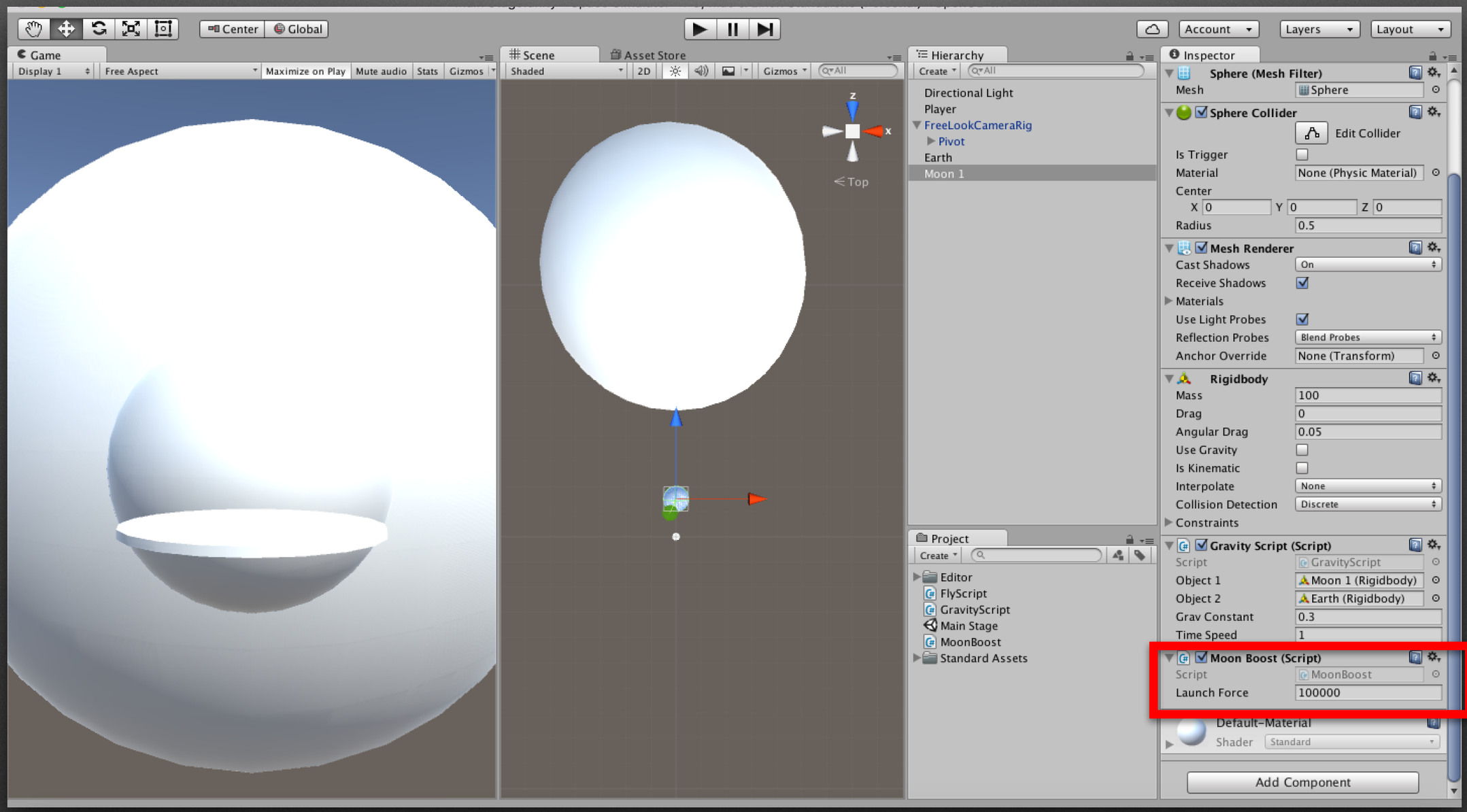
Changing Variables

Altering variables in the inspector allows you to change the way your objects interact with the world. Note that if you edit them in “Play” mode, they are edited in real time, but are not saved when you exit “Play” mode.



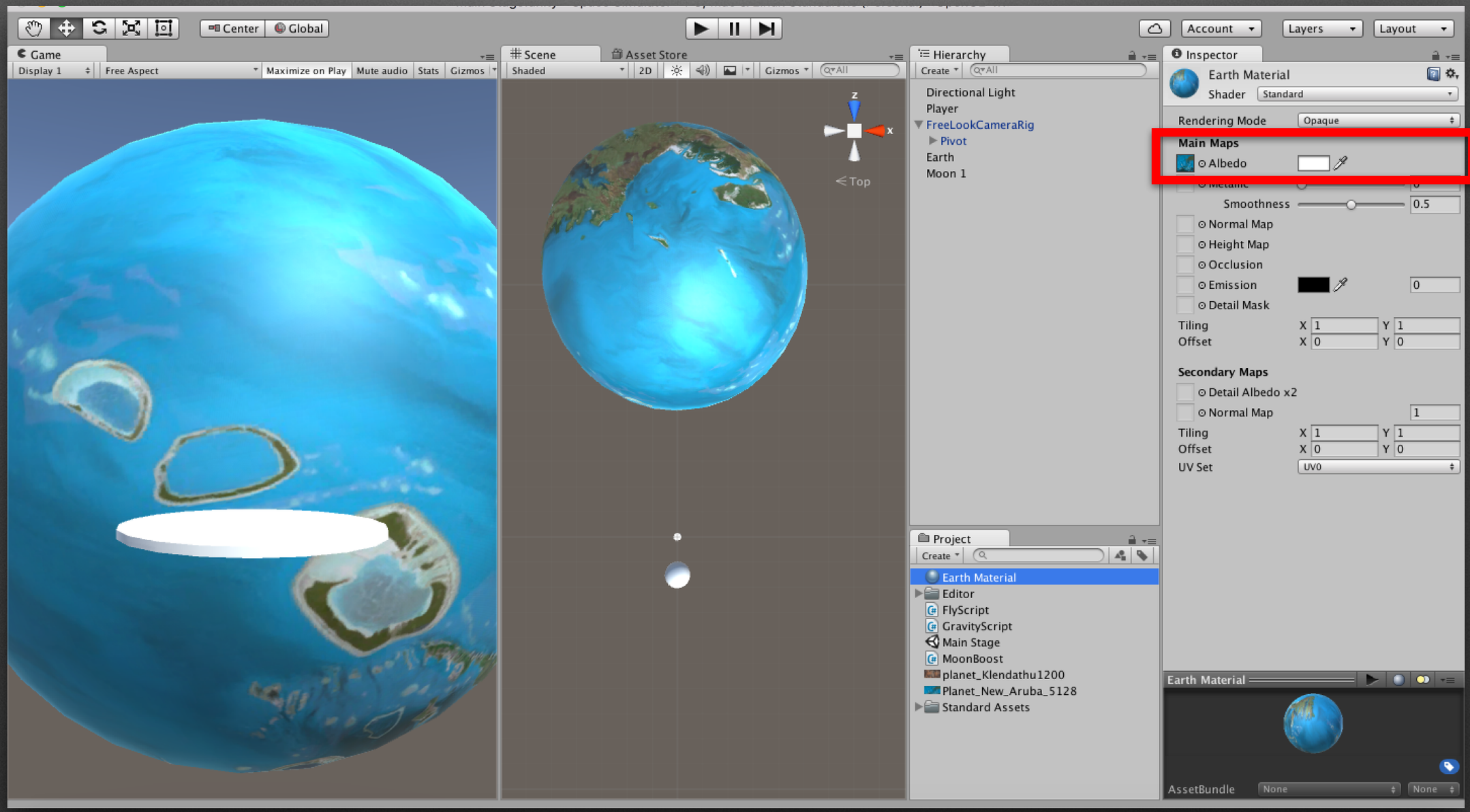
Changing Variables (Pt. 2)

Note that we've changed the Gravitational Constant. We will change it for the moon as well.



Launching the Moon

If you've tried play mode, you've noticed that the moon just falls and bumps into the Earth. We need to give it some initial velocity. So we add the "Moon Boost" script to launch it. Note that we've also altered the Gravitational Constant.

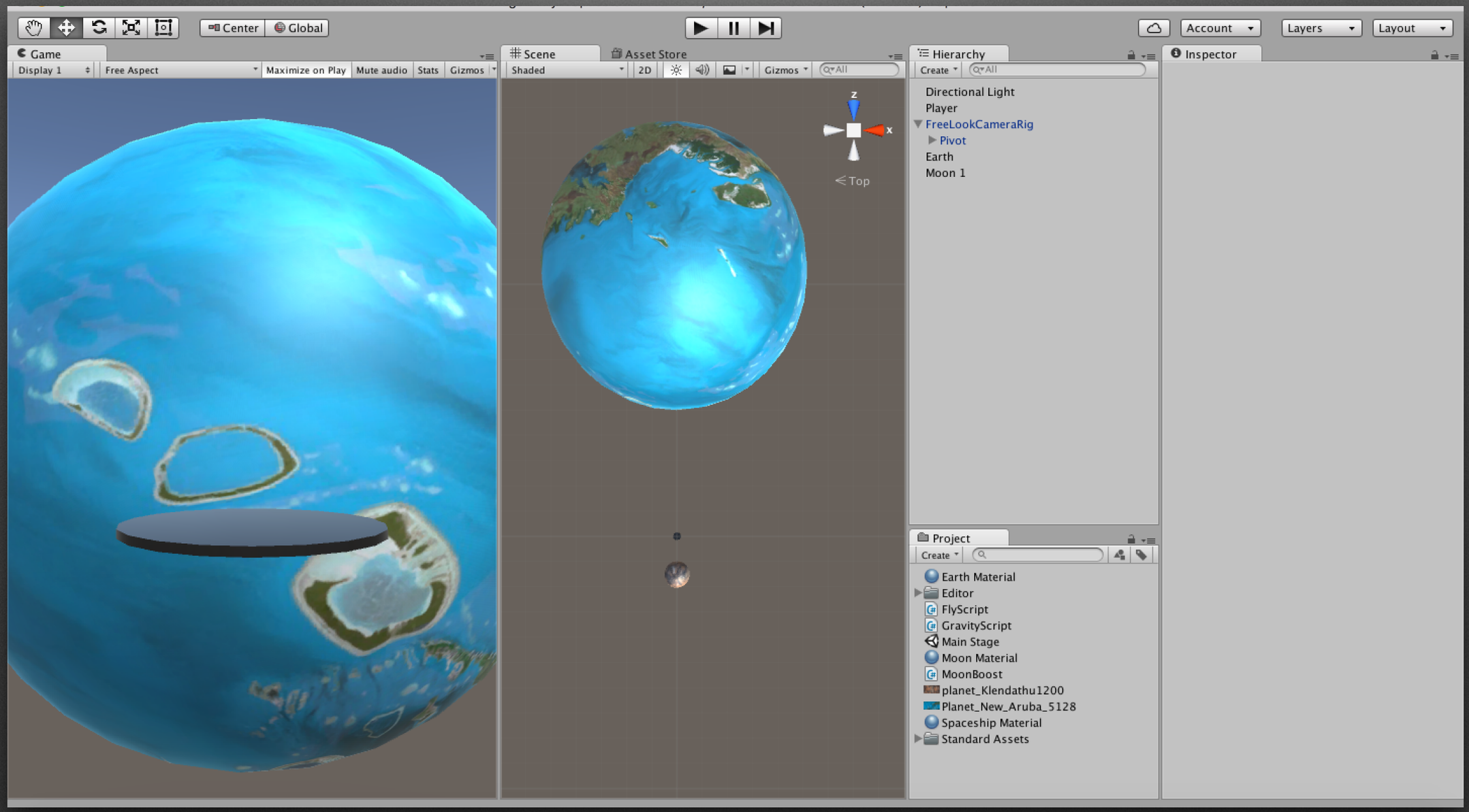


Flavor Text(ures)

Adding textures is easy. Just drag the picture you want as a texture into the “Project” tab. Then create a material using the “Create” button under the same tab. Drag the picture to the “Albedo” slot, and drag the material to the object you want to have that texture.

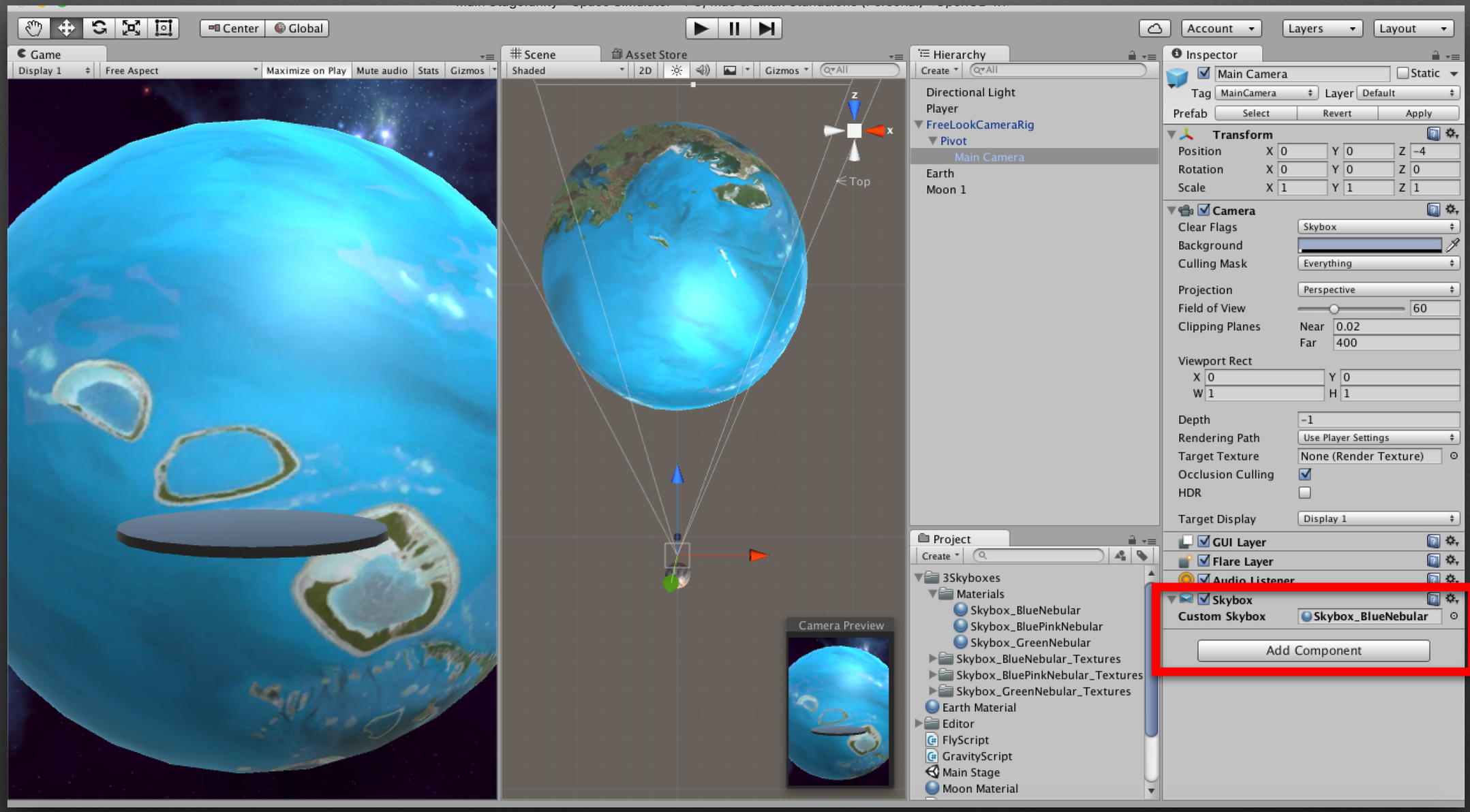
More Materials

We use the same process to create the material for the moon, creating a new material, dragging the photo into the “Albedo” slot, and dragging the material to the Moon object.



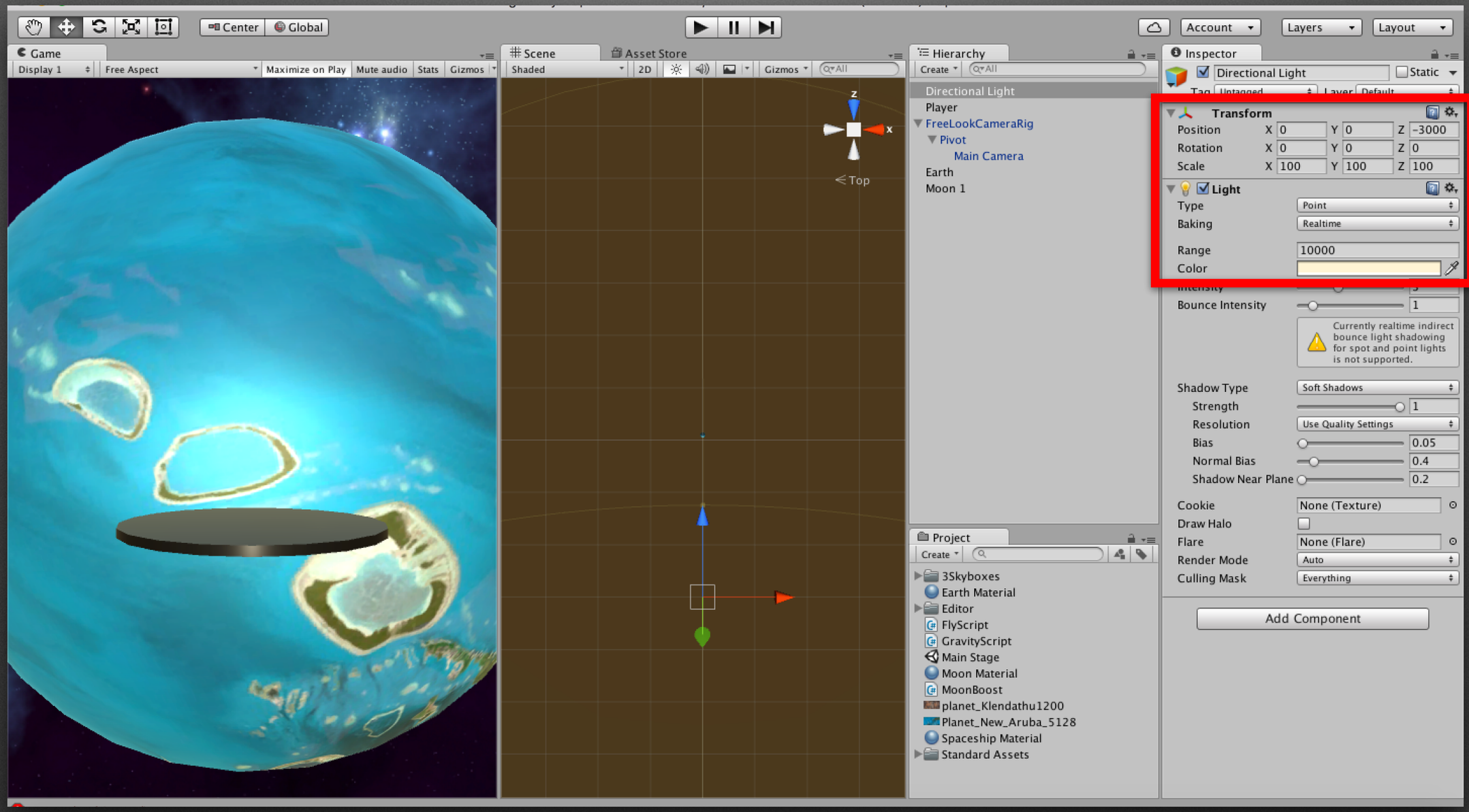
More Materials

We create another material for the spaceship. For this one, I simply made the default grey material darker, and increased its metallic slider. (NB: There are tons of free textures that look better than this. In fact, there are tons of free assets on the Unity Store. There are even spaceships that could replace our basic saucer. Look around for them!)



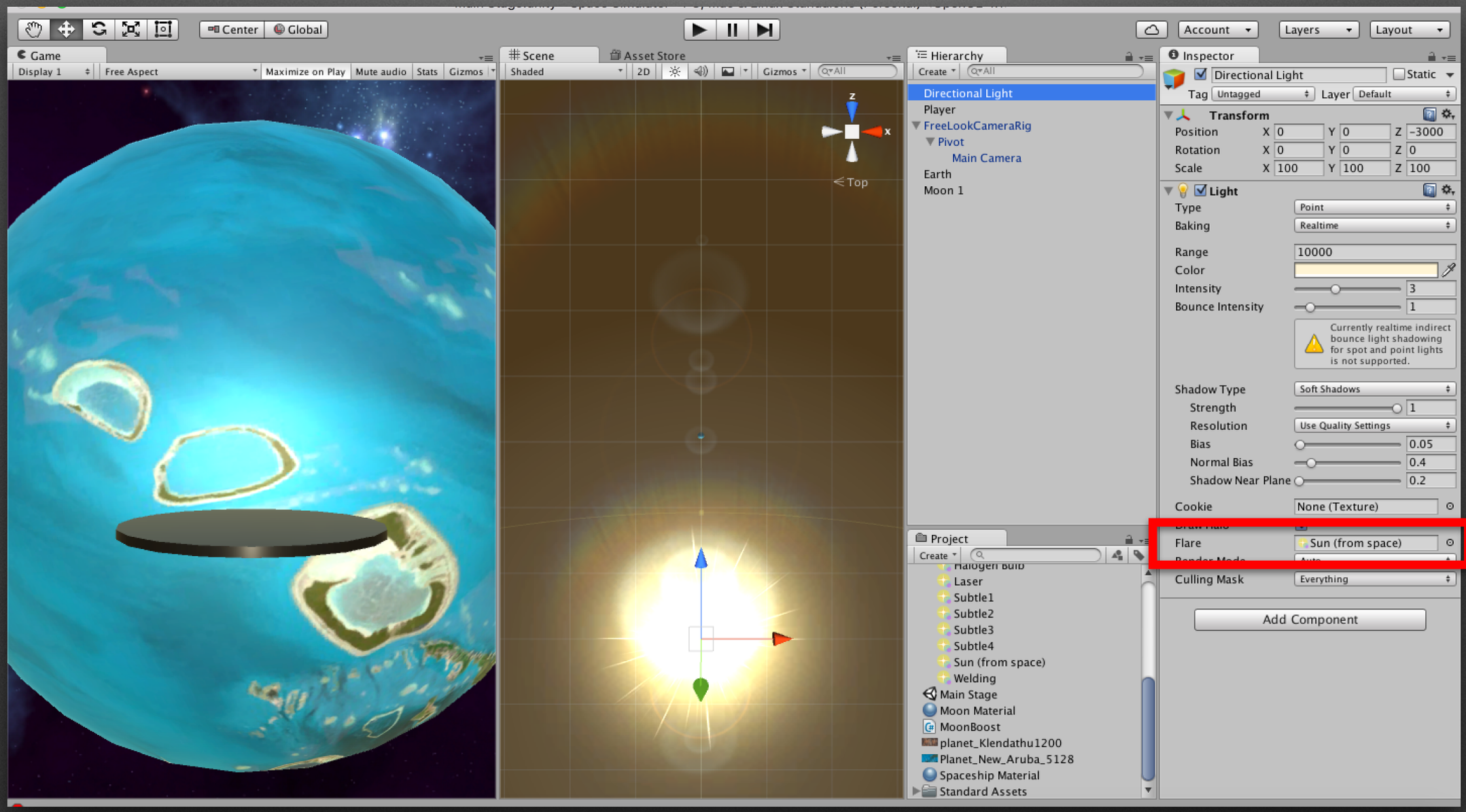
Adding a Skybox

After finding some free space skyboxes on the Unity app store, we simply import them to our camera by clicking the “Add Component” button, selecting skybox, and dragging our Skybox into the slot.



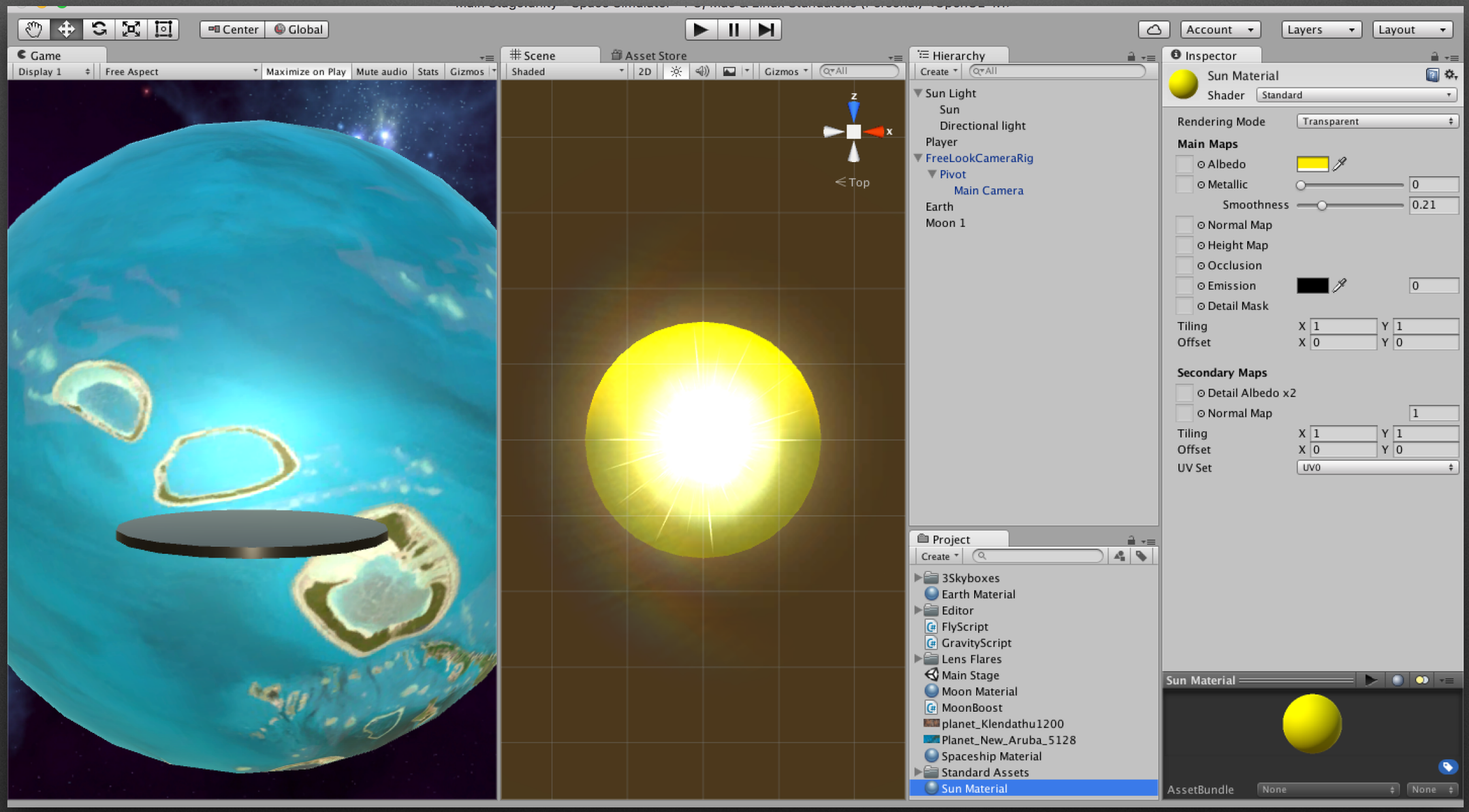
Adding the Sun

We change the “Directional Light” which we created at the beginning to a point light.



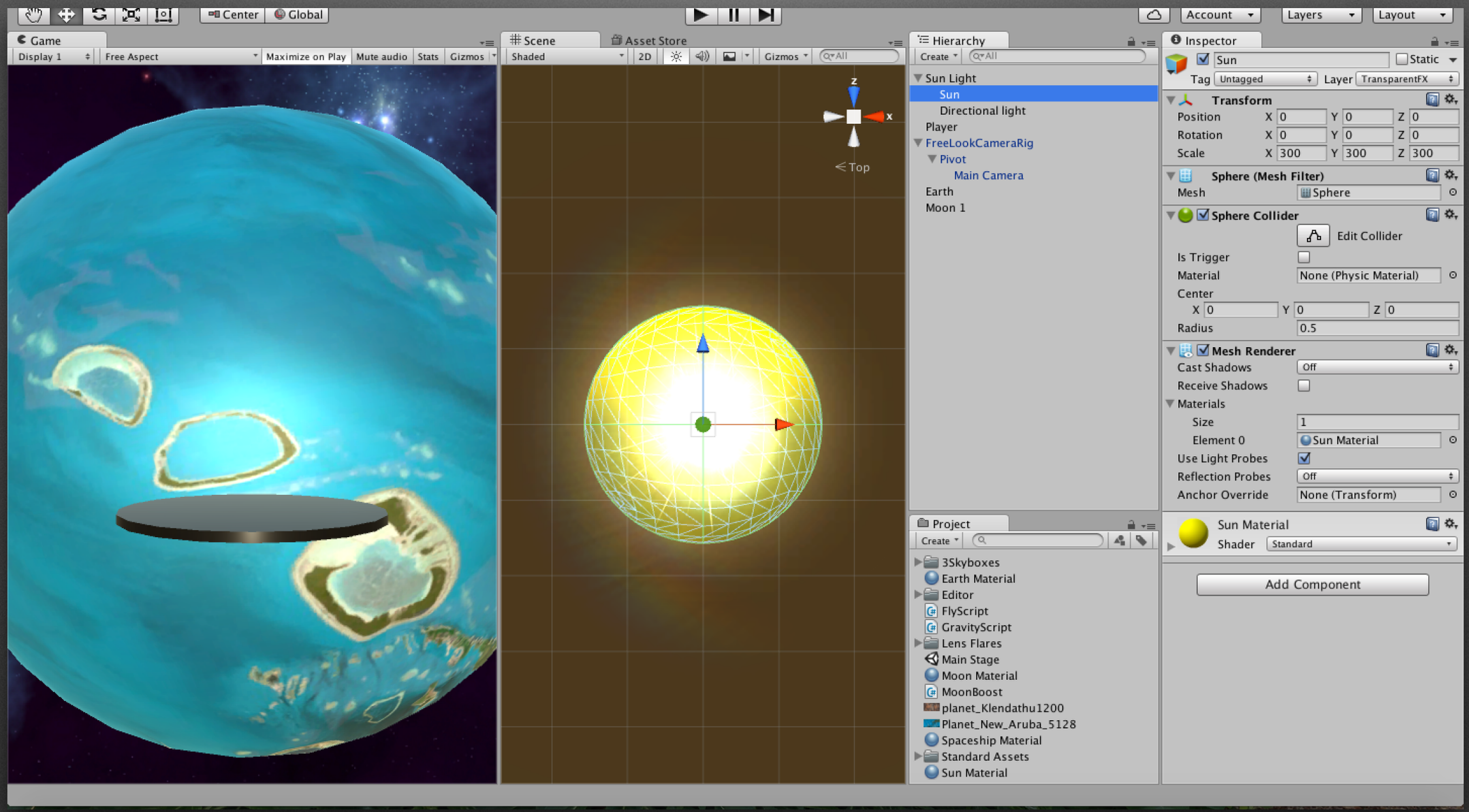
Solar Flare

After importing a set of sense flares (once again, free from the Unity store) we add the “Sun (from space)” flare to our camera.



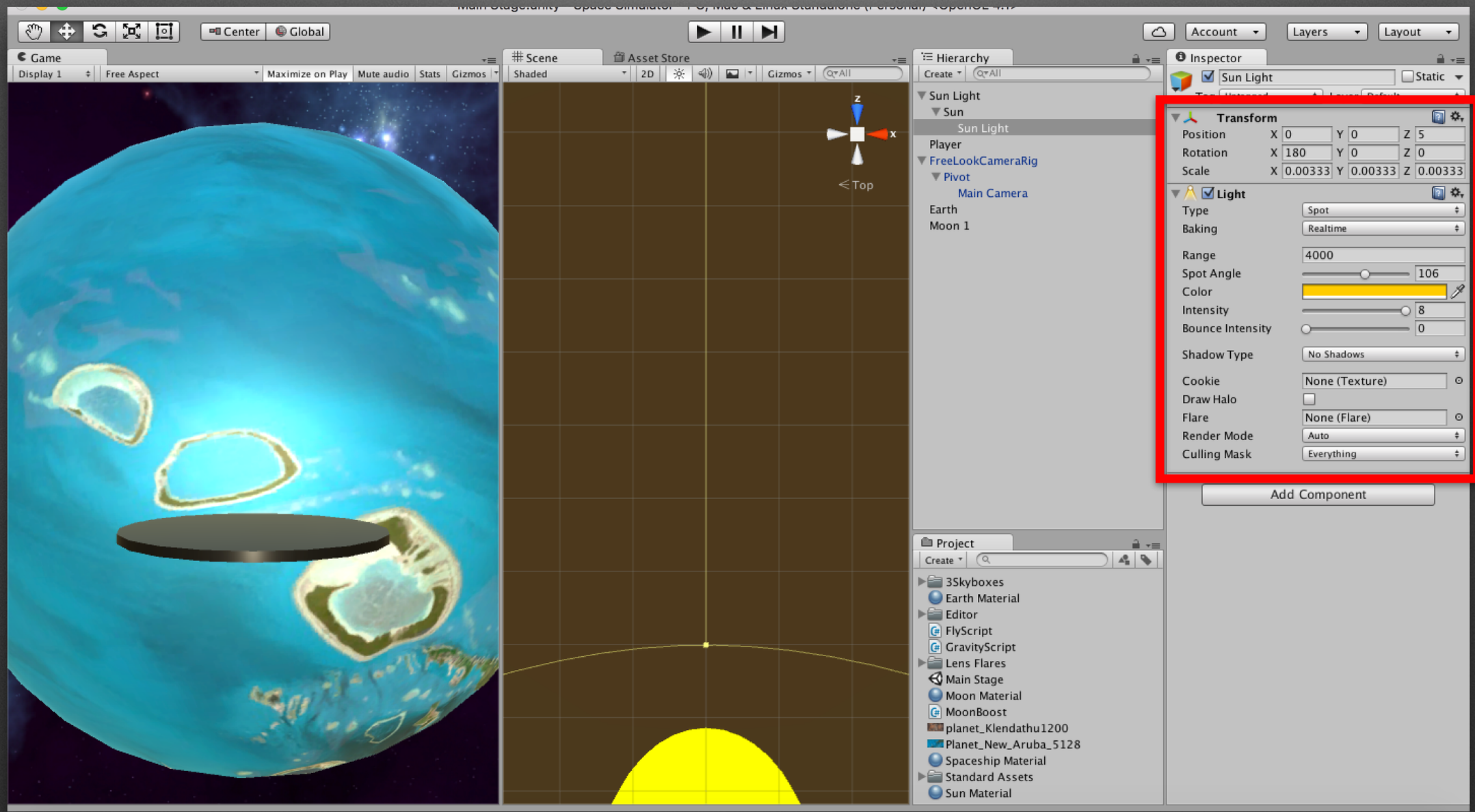
Making the Sun

We create a sphere, another default material (this time colored yellow) and attach it. This sphere will be our Sun. You'll notice as well that I've changed the name of the light we just made (I've made it "Sun Light") and made the sphere a child of it. This is done by dragging the intended child onto the parent. This has the advantage of making all transformations relative (that is, they move and scale together).



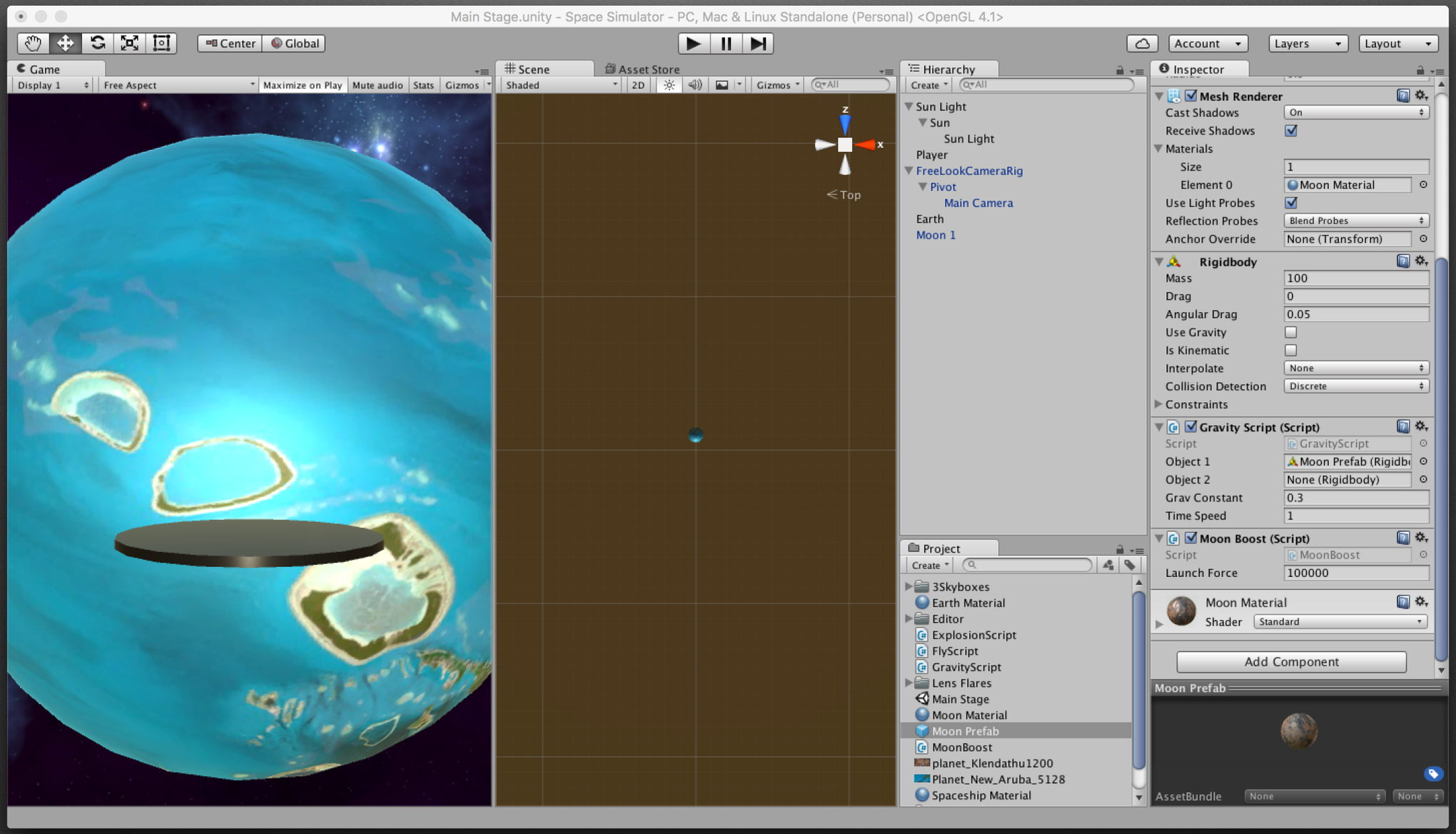
Directional Light?

That “Directional Light” you see as an additional child of the “Sun Light” is a light pointed at the sun. This is in order to give the sun a glowing appearance, in order to make the sun look as though it’s actually giving off light. This isn’t the best way to make a realistic sun, but it is easy.



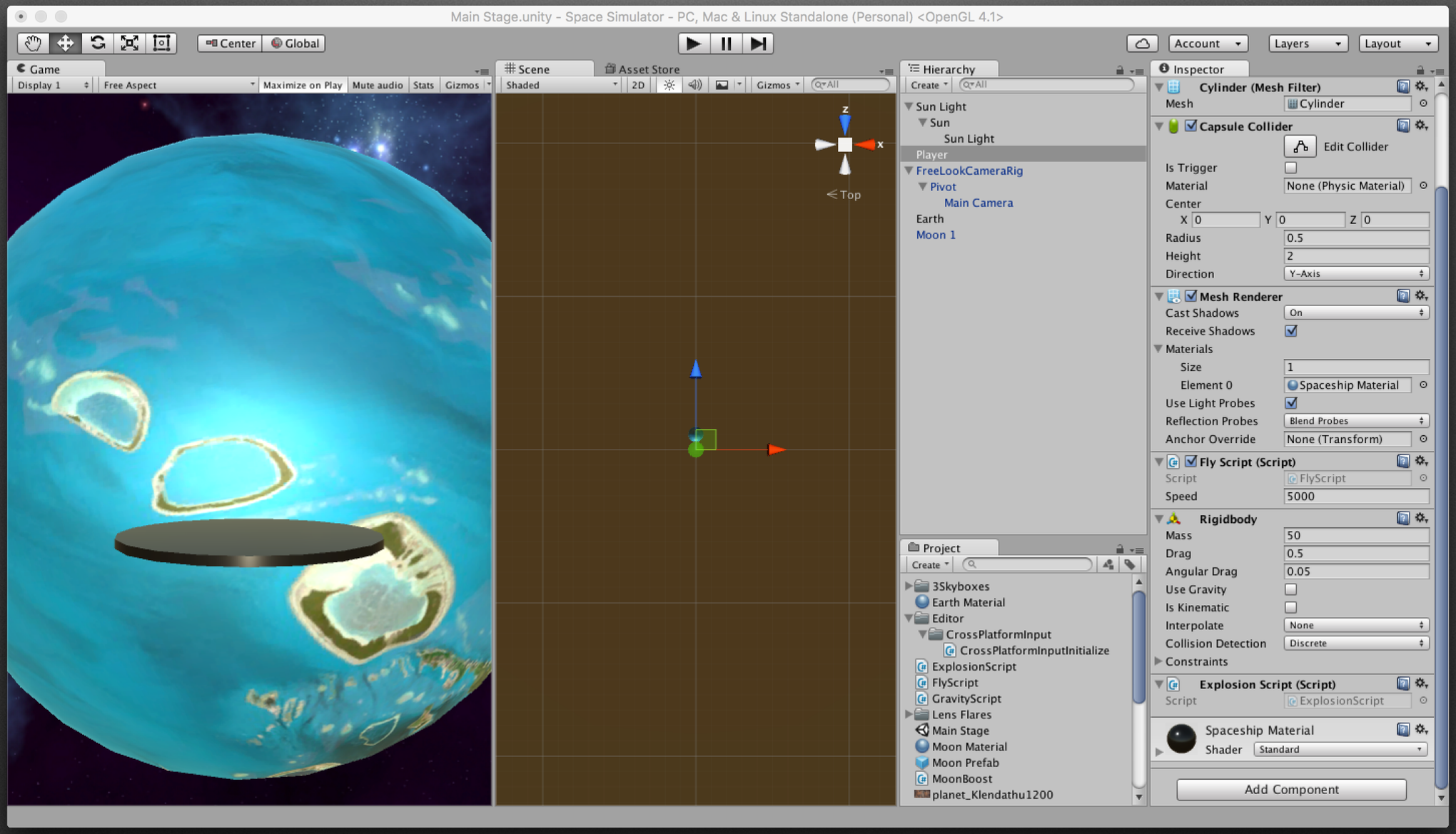
Fixing the Directional Light

The new light that we've added gets altered substantially to create the desired glow. We also make it a child of the sun in order to make sure they transform together. Please note that naming two objects the same way (as I have done) is, while technically possible, a very bad idea which can get very confusing.



Above and Beyond

Another important concept in Unity is the concept of “Prefabs” — Assets that have the same basic values that you can use over and over. I’ve made a prefab of our moon, so we can create many moons for our planet.



Above and Beyond (Pt. 2)

Triggers are another important concept in Unity. They are used to cause a script to be activated when an object enters a certain zone, or collides with another object. Triggers and collisions are treated slightly differently, but can be thought of as very similar. Here, we've added a short script to cause the player to be destroyed if he crashes into another object.

Above and Beyond (Pt. 3)
Or: Where to Go from Here

For our new friends:

- It may seem daunting, but expanding our simple Solar system can be done without writing any new code. Just add objects and gravity and create new worlds to explore.

For our more experienced friends:

- There are a lot of things in this project done “fast” rather than “well”. Fixing those (I try to mention them in the slides) is a great place to start.
- Try implementing an actual game, rather than simply something to explore. Consider setting up targets to create a race, or perhaps enemies to fight.

For our expert friends:

- Add enemies to shoot or create explosions with particle effects.

Or just use your imagination!