AD41700 Computer Games
Prof. Fabian Winkler/Christian Barrett
Fall 2011

Introduction to Unity3D (vers. 3.4)

Game development is a tricky thing. While there's quite a lot of people who want to be involved in amateur game development, relatively few of them can manage to create a 3D game on their own. It could be the high barrier of entry; traditionally, creating a 3D game either resulted in a large amount of coding to do the simplest of tasks or a lowering of standards to fit with the engines targeted at beginners. It could also be a resources thing; game development traditionally takes a lot of time and money. So what is the game development enthusiast to do?

The solution is in a game engine targeted at independent developers and allows for rapid testing of ideas. Luckily, there is such an engine on the market: the very capable Unity3D. Unity provides a strong combined graphics, audio, physics, and input engine that encourages an implement-test-tweak model of game development. It's easy to pick up, works well with most 3D modeling packages (including the most popular modeling package, Blender, which is completely free), runs quickly, and can deploy to Windows, Mac, the Web, and a number of hand-held devices. Another huge advantage Unity3D has is in price. For no cost at all, anyone can download a copy of Unity with a license that allows commercial distribution of any products created with it. While there are some features only available in the professional versions and licensing (such as real-time shadows and deploying to platforms other than desktops and the web), the free version of Unity3D is more than enough to introduce somebody to game development.

Unity vs. Unity Pro

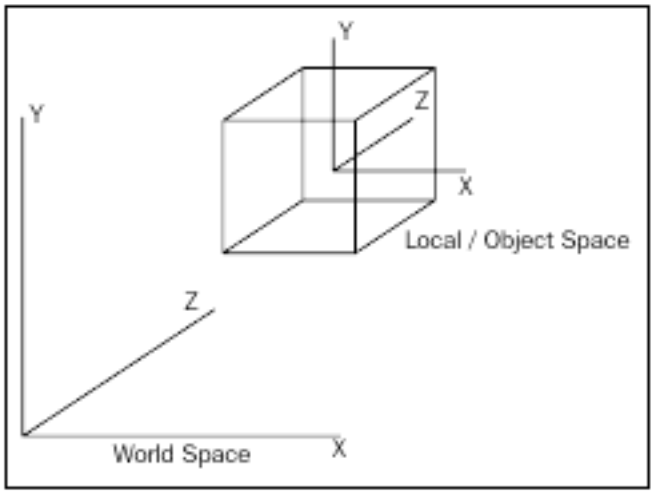
In contrast to the expensive Unity Pro version ($1,500) the free version of Unity does not have: realtime shadows, realtime audio filters, custom splash screen, video playback/streaming, development possibility for iPhone and Android devices.

But the free version of Unity still has some very powerful features:
Physics engine, positional audio, web browser and standalone deployment, shaders, terrain editor, tree creator…

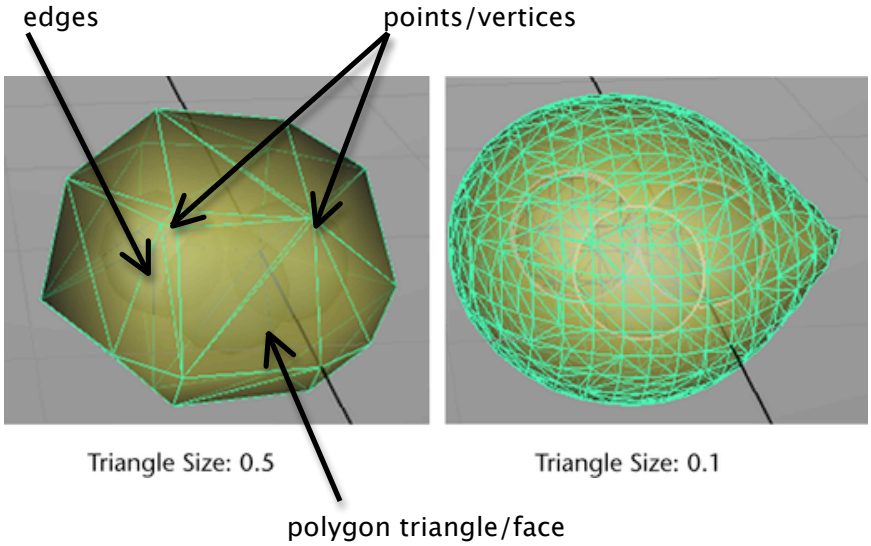
A quick note:
The goal of this workshop series is to teach students how to quickly implement a game design to be tested and tweaked. Due to the limited time we have, we will not be covering the creation of 3D or 2D art assets. This is part of the individual responsibilities of the interdisciplinary teams that are working together in the second half of the class.

Before we begin working in Unity3D, let's take a look at some important concepts used in the software:

Coordinate Space



Polygons



Textures

Textures need to be square and sized to a power of 2:

- 128 x 128
- 256 x 256
- 512 x 512
- 1024 x 1024

The larger the texture file the more processing power is needed.

Physics
A Rigid Body component is given to any object you want to be under control of the physics engine.

Objects can have the following properties:

- Mass
- Gravity
- Velocity
- Friction

Collision Detection

Assets

Scenes
Levels/areas of game content (menus, etc.)

Game Object
Assets that are being used in a game scene.

Components
Games are made up of individual components or parts that can be pieced together in a modular fashion (behaviors, defining appearance, etc. also scripts), e.g. renderer component to make objects visible, transform components place them in the virtual world, etc. Components can be added to game objects.

Scripts
Javascript, C#(C sharp), Boo

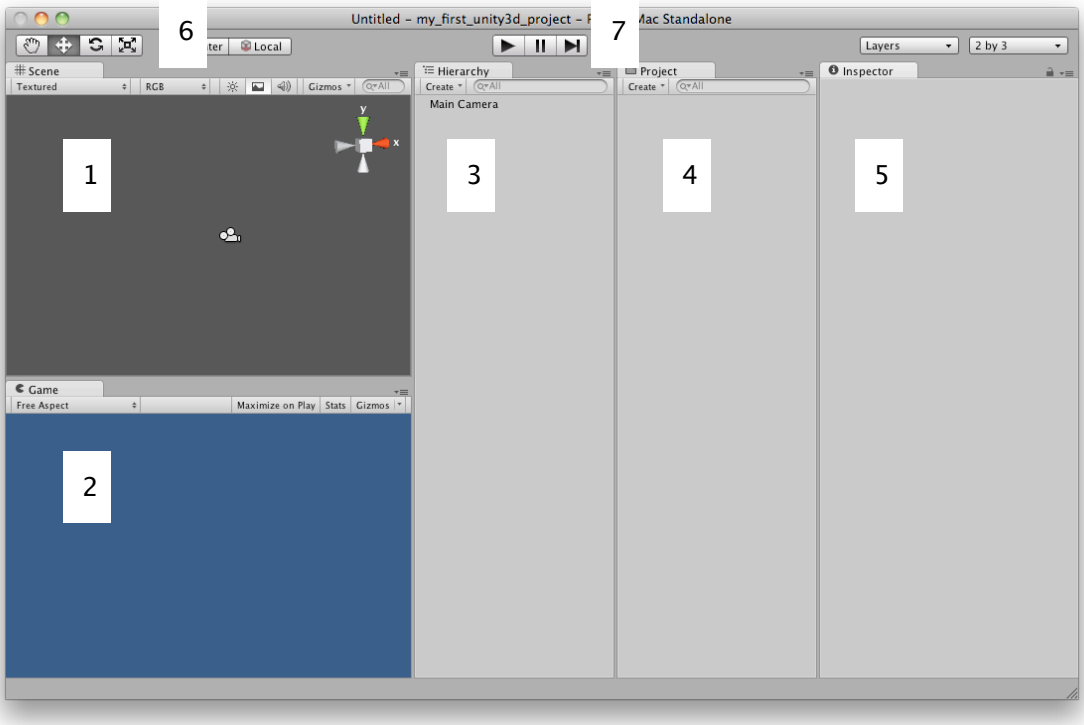
Prefabs
Store a game object together with is components + configurations for easy duplication/reuse.

User Interface

`File > New Project`

(Remember to keep all you project related assets in this location to avoid missing files and broken links alter in the game development process).

`Window > Layouts > 2 by 3`



- Scene
- Game
- Hierarchy
- Project
- Inspector

1. Scene: This is where you will place any visual assets in your Unity environment. It will update in real-time when you are previewing the game. Note the manipulator on the top right; this allows you to switch between a number of standard views. We are currently in the perspective view. Although this doesn't matter too much, it allows us to view our scene with a vanishing point, which is the standard way Unity games will display.

2. Game: When you're not actively running the game, it will show a render of how the game will look, ignoring graphical effects that need to be computed at run-time, from the point of view of the main camera. When you're previewing the game, you'll be playing through this window. Since our scene is currently empty, all this window is showing is the background color.

3. Hierarchy: This lists all the objects in the currently loaded scene, and any children they may have. **Children** are objects that can be thought of as subordinate to the parent object; wherever the top object moves, they'll follow, keeping the current offset they

have to this object. This is an important concept for Unity beginners to understand; we'll cover it more later and in the workshops.

4. Project/Assets view: This is a list of all custom assets for our game, including graphical assets, sound, scripts (more on these later), prefabs (pre- assembled game objects), and much more. Our current game is only using the Standard Assets (which come with Unity, and provide templates to quickly get going in a virtual environment), our current scene, and a custom character control script.

5. Inspector: Since we currently don't have any objects selected in the Hierarchy or the Project/Assets view, it's completely blank. The inspector allows us to look at and tweak individual settings of various game objects and assets, as well as adjust some global settings. The Inspector is content-sensitive and changes its parameters based on which game object/asset is selected. This is also a place to show you your project settings and preferences by choosing them from the Edit menu.

Number six (6) are the **graphical icons for moving the scene and its contents**. The hand allows us to pan around the scene; when combined with other scene camera controls, Unity becomes very easy to navigate (we'll cover these later). The icon on its right, which looks like four arrows, allows you to move a selected object around. We call this transforming the object. The next icon allows for rotation of the object, and the final one allows for uniform scaling of the object.

Number seven (7) is the **playback bar**. This allows us to play, pause, and stop running our game in the Unity editor. This is the quickest and easiest way to test and tweak your game.

Navigating the Scene Window

The scene view is what allows you to look around and move the visual assets you import into Unity. It's how you'll assemble your levels and place important things like lighting, trigger zones, audio, and much more. Being able to control the camera is important if you want to do anything at all with it.

Hand Tool (shortcut Q): drag around in the scene to pan your view. Holding down alt+drag will rotate the view, Ctrl.+drag will allow you to zoom. It is important to remember that this doesn't move anything in the scene, just your point of view.

Translate Tool (shortcut X): active selection tool, enables to drag an object's axis handles in order to reposition it.

Rotate Tool (shortcut E): using handles to allow you to rotate an object around either of its axes.

Scale Tool (shortcut R): works the same as the previous two tools, allows scaling of an object.

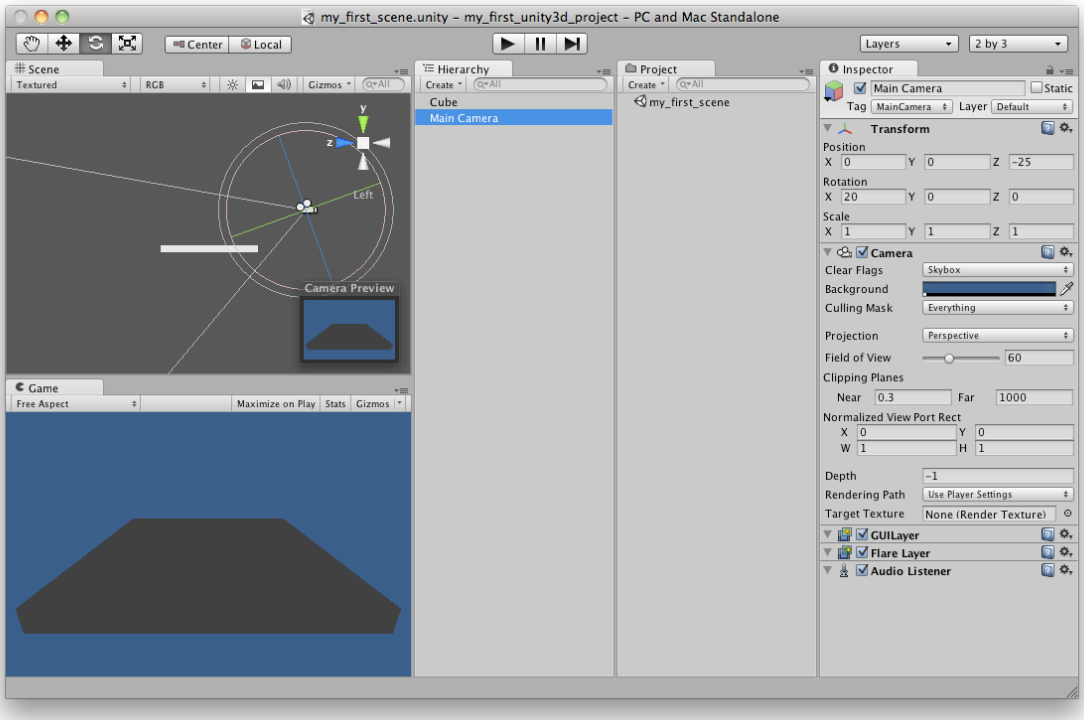
Your First Unity3D Scene

Now that you can look around the scene, let's learn a few ways we can place things in it. First we'll take a look at the basic game objects Unity can create without visual assets. Unity 3 has geometric primitives (cubes, spheres, planes, capsule, etc.), lights, particle systems, cameras, and more that it can create without needing external assets. To access these, go to the top menu bar, select Game Object ->Create Other and make a choice. To begin with, try making a simple scene with a cube (functioning as a floor), a sphere, and a light. There's three different types of light – for now, a directional should work just fine, as light travels in rays with the direction of the arrows of the light, a good way to simulate the sun in Unity.

Start by creating a cube:
`Game Object > Create Other > Cube`

You can already use the Inspector window (after selecting the cube in the hierarchy) to modify its scale properties: Scale: X: 25, Y: 2, Z: 25 and to translate it Position: X: 0, Y: -10, Z: 0

Now we just need to move our camera a little back and point it downward to see the newly created box (it looks more like a plane now)
Position: X: 0, Y: 0, Z: -25; Rotation: X: 20, Y: 0, Z: 0;



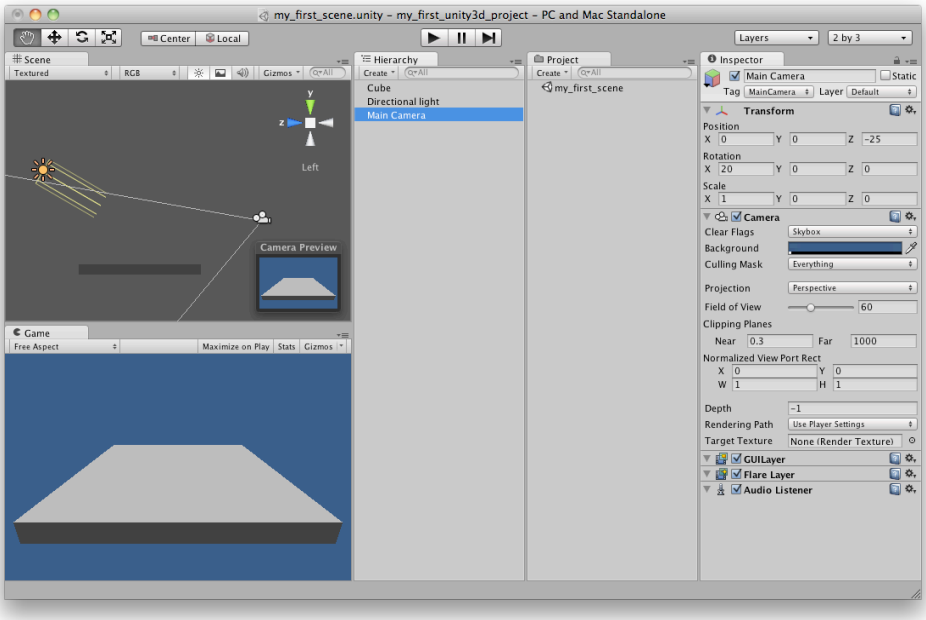
It's now a good time to save your very first scene before we continue.

Tip: If you want to remove a game objects from the hierarchy/scene – select them press command+delete or go to Edit>Delete

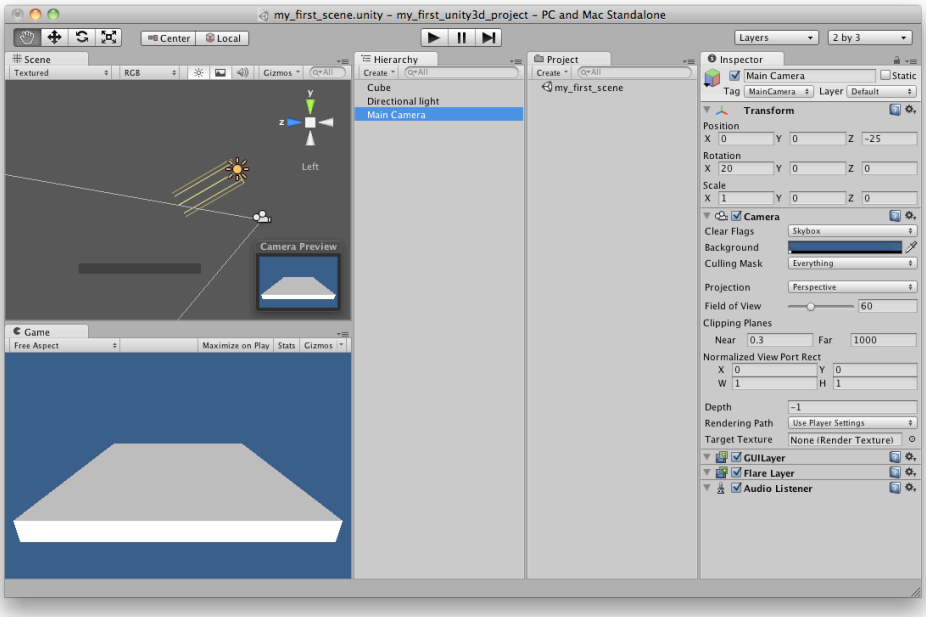
Now create the directional light source:

`Game Object > Create Other > Directional Light`

We should also move the light source a little back, up and tilt it downward to see its effect on the box object:
Position: X: 0, Y: 10, Z: 20; Rotation: X: 30, Y:180, Z: 0;



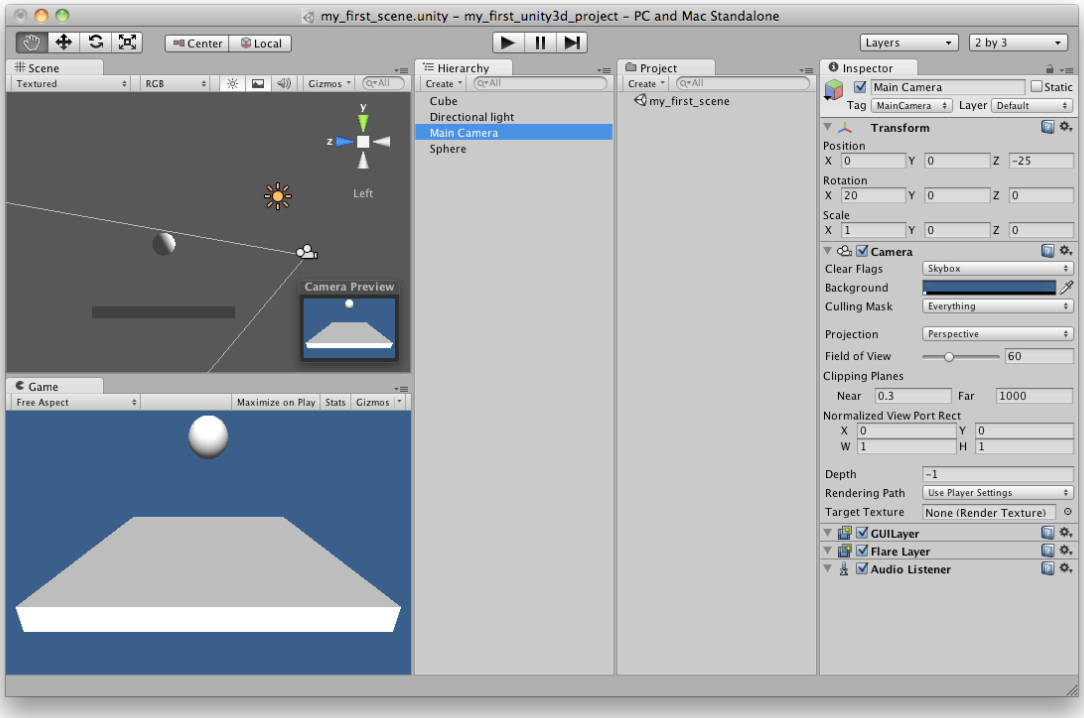
See the change in how it affects the box by experimenting with different angles and directions of your light source:
For example:
Position: X: 0, Y: 10, Z: -20; Rotation: X: 30, Y:0, Z: 0;



Next, create the sphere and place it above the box:

Game Object > Create Other > Sphere

Position: X: 0, Y: 2, Z: 0; Scale: X: 4, Y: 4, Z: 0



Since there is a main camera that comes with every scene, you could hit the play button now and view your scene. Unfortunately, it will be entirely static. You can't control the movement of the camera and none of the scene is moving itself.

Introducing Physics
The first thing we need to do is to give the sphere a rigid body component. Select the sphere and go to: Component > Physics > Rigid Body.

Hit the play button and you see how the sphere is falling, but not bouncing.

In the next step we create a physic material which will provide the material properties to make the game object bouncy:

Asset > Create > Physic Material

Then select the material in the Project window and experiment with the bounciness value in the Inspector. The higher this value (closer to 1) the bouncier the material.

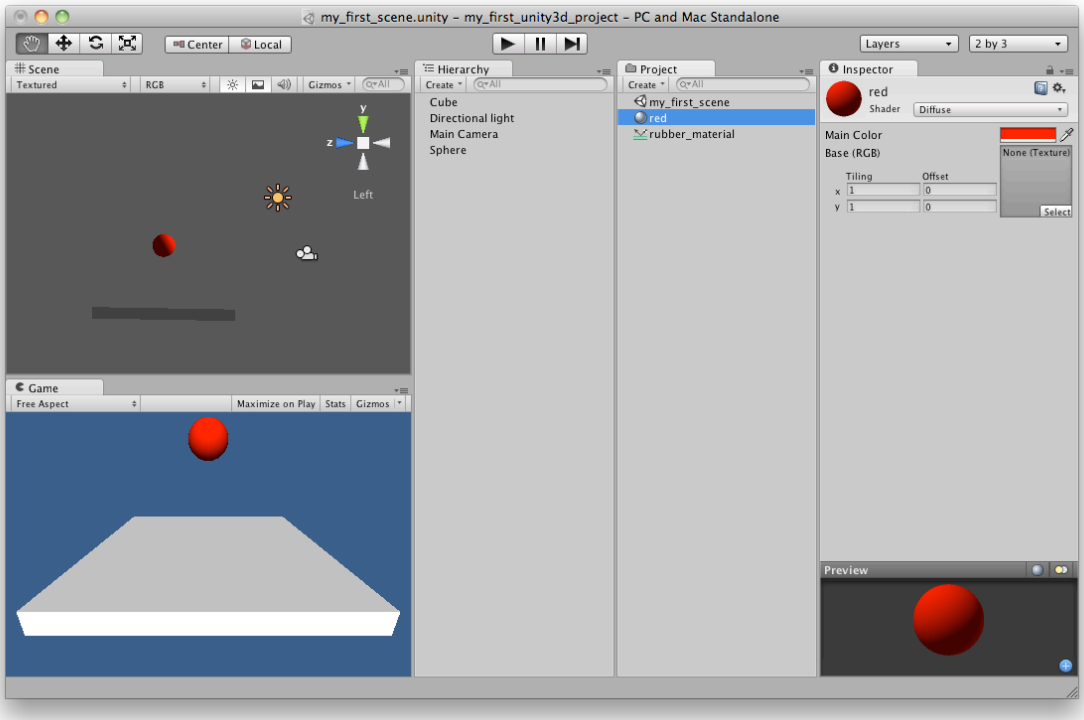
Hit the play button and see how the sphere is bouncing similar to a rubber ball.

You can now start experimenting with different heights from which the ball is falling as well as different angles of the box to see what response they create in the behavior of the sphere.

Creating Materials:

Assets->Create->Material

You will see the new material in you Project window and its properties in the Inspector window. In order to change its color double click the color swatch next to Main color and choose a different color. You apply the material to a game object by simply dragging it from the Project window onto the game object in the hierarchy window. You can then start tweaking the material's properties by experimenting with different shaders from the shader drop-down menu in the Inspector window.



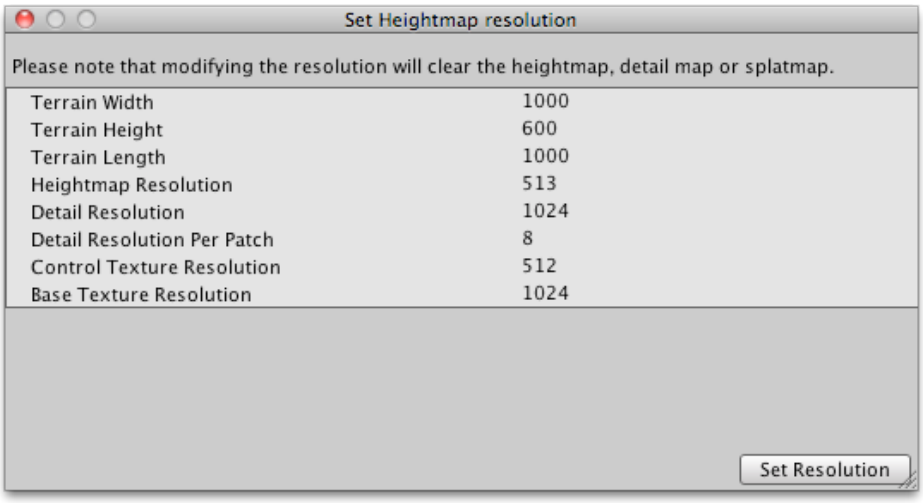
Another way to change the visual appearance of game objects is to use **2D textures**. A good overview of 2D texture features in Unity is in this chapter of the software's documentation: http://unity3d.com/support/documentation/Components/class-Texture2D.html. We will take a closer look at textures in one of the following workshops.

Using Prefabs and the Terrain Editor

Now after we have learned how to create simple primitive geometries in Unity3D and attach physics properties to them, let's explore how we can use prefabs to move around a virtual environment interactively, using a first person point of view.

But let's first create a terrain that we can move around in:
We will use Unity's Terrain editor to do this – we start with a simple plane, which can be easily turned into a topographical landscape.

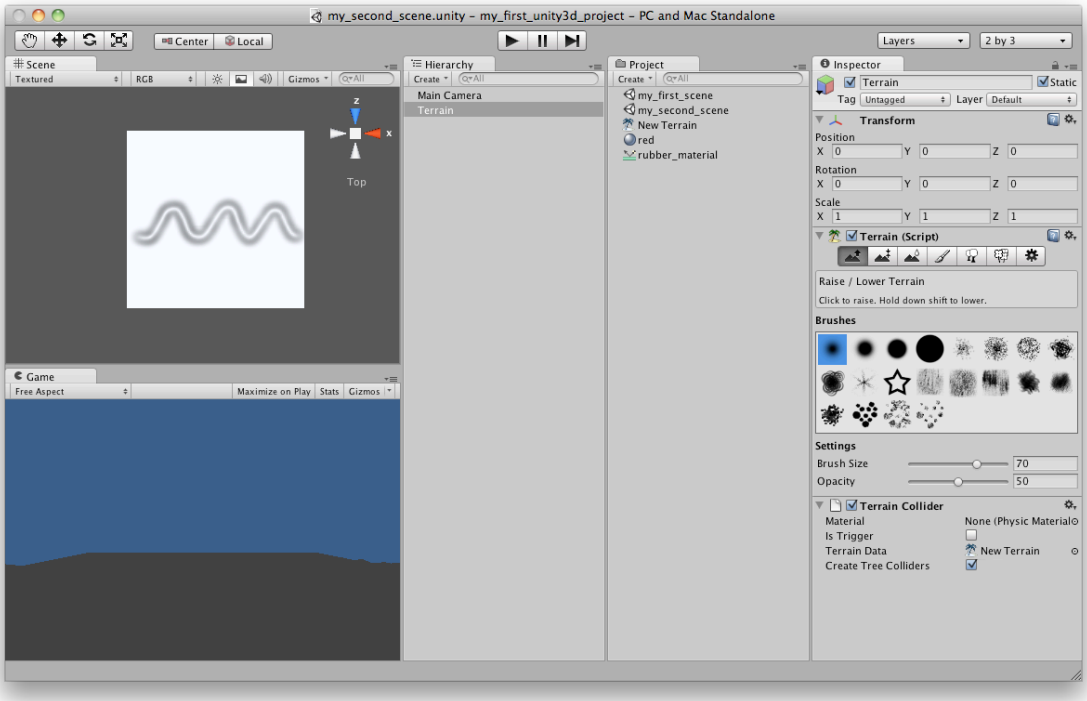
Create a terrain Terrain > Create Terrain and then access its heightmap resolution menu



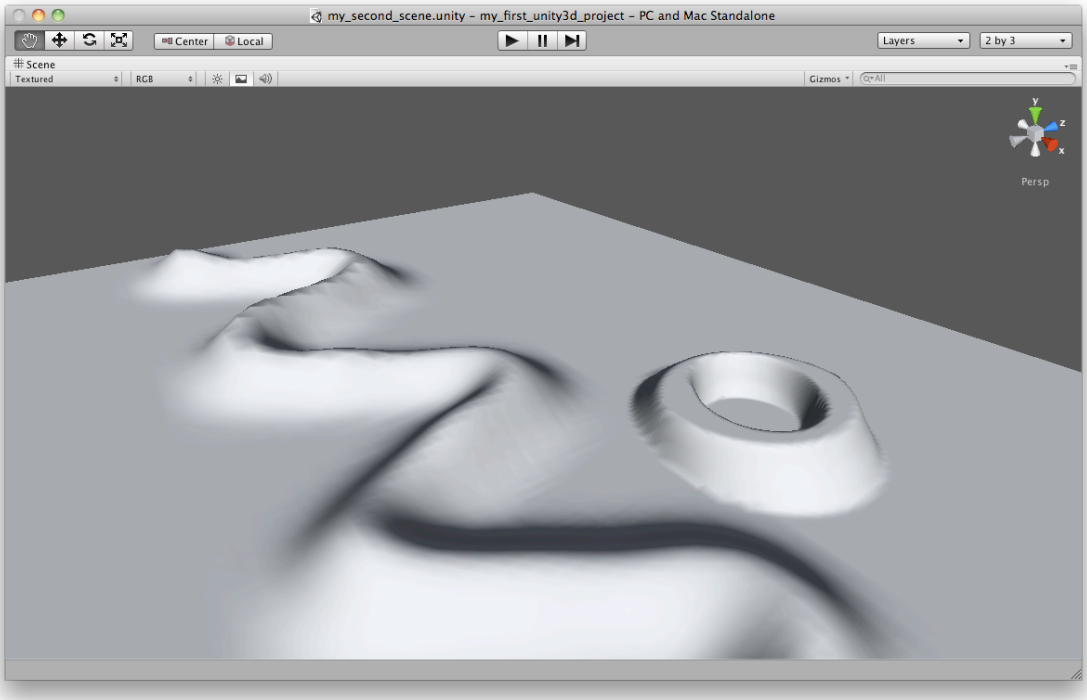
Moving the camera up and over a little helps you seeing the original plane for the terrain a little better: Position: X: 500, Y: 100, Z: 0

Change the view in your Scene to Top view by clicking on the view icon in the Scene window's top right corner.

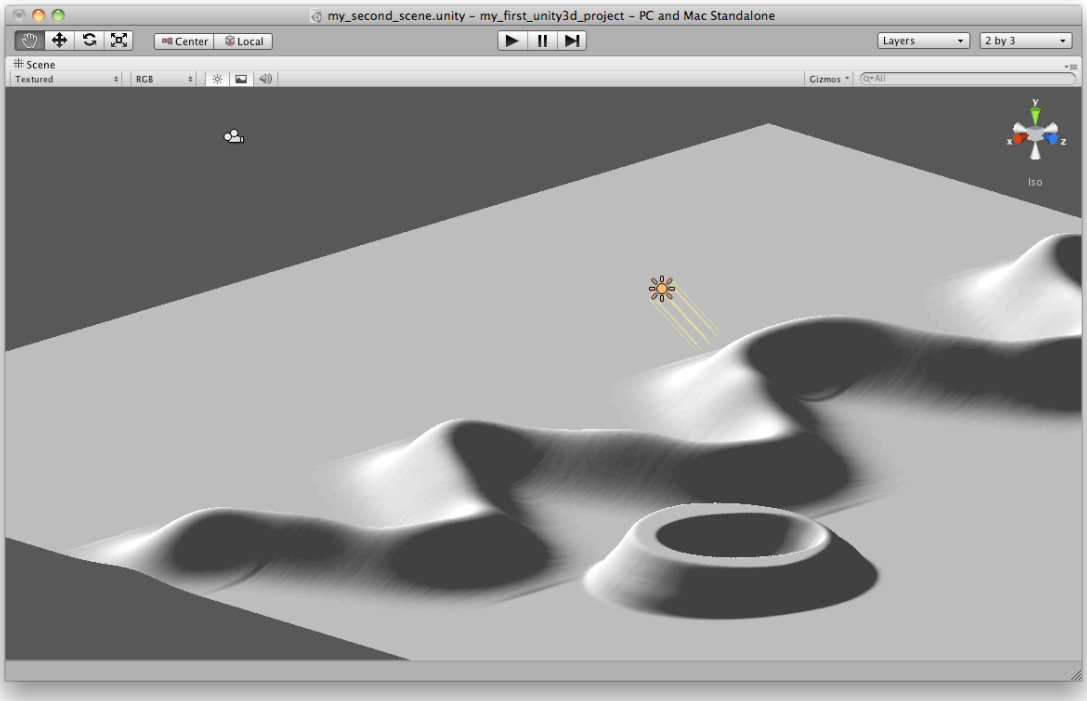
Now click on the Terrain in the Hierarchy window and choose the "raise height" tool (the first one on the right). Click and drag to create a shape on your plane.



Experiment also with some of the other terrain paint tools, such as “lower terrain height”, “set terrain height” and “smooth terrain height”.



Finally create a directional light, so the topography looks more dramatic:

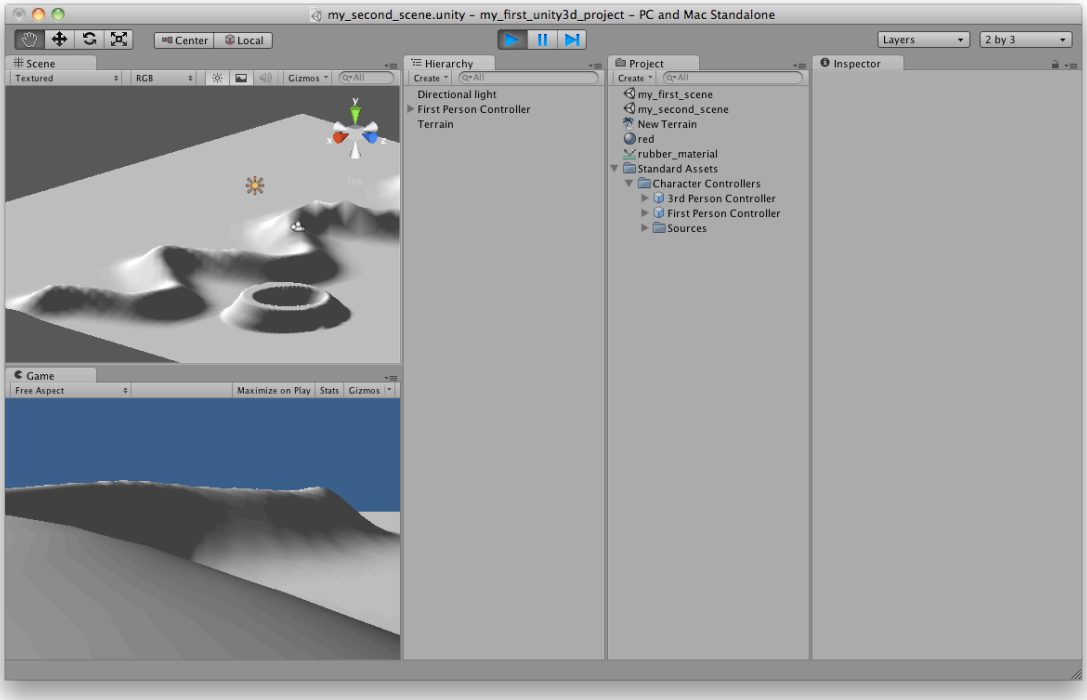


In the interest of time we will stop our experimentations with Unity’s Terrain Editor here and move on to creating a first person perspective that allows you to navigate the

terrain interactively. (For a more in depth discussion of terrains and Unity's Terrain Editor see chapter 2 in Will Goldstone's book *Unity Game Development Essentials*).

If you haven't included the Standard Assets Package while you created the new project you should import it now, specifically the Character Controller package: Assets > Import Package > Character Controller.

In the Project Window open the Standard Assets folder, open the Character Controller folder and drag the First Person Controller onto your scene. Delete the main camera, hit the play button and you are ready to explore the terrain you have just created. You can look around with your mouse and move with WASD or the arrow keys.



Further Resources:

Unity beginner's video tutorials: http://unity3d.com/support/documentation/video/

Unity User guide: Basics
http://unity3d.com/support/documentation/Manual/Unity 20Basics.html

Creating 3D Content externally for Unity3D – freeware modelers for Apple computers

- Blender (http://www.blender.org/) extremely powerful freeware but steep learning curve.
- Wings3D (http://www.wings3d.com/) less powerful but also easier accessible.
- Sketchup (http://sketchup.google.com/), easiest to use but hard to export content for use in Unity3D – only possible with the free OBJexporter v1.6 20110402plugin:
  http://forums.sketchucation.com/viewtopic.php?f=323&t=33448

Here is a list of currently supported software for external content:
http://unity3d.com/unity/features/asset-importing

3D formats	Meshes	Textures	Anims	Bones
Maya .mb & .ma[1]	✓	✓	✓	✓
3D Studio Max .max[1]	✓	✓	✓	✓
Cheetah 3D .jas[1]	✓	✓	✓	✓
Cinema 4D .c4d[1] [2]	✓	✓	✓	✓
Blender .blend[1]	✓	✓	✓	✓
Carrara[1]	✓	✓	✓	✓
Lightwave[1]	✓	✓	✓	✓
XSI 5.x[1]	✓	✓	✓	✓
SketchUp Pro[1]	✓	✓		
Wings 3D[1]	✓	✓		
3D Studio .3ds	✓			
Wavefront .obj	✓			
Drawing Interchange Files .dxf	✓			
Autodesk FBX .fbx	✓	✓	✓	✓

[1] Import uses the application's FBX exporter. Unity then reads the FBX file.
[2] Cinema4D 10 has a buggy FBX exporter. Please see here for workarounds.

Image Formats

- Photoshop .psd and .tiff are imported with layers automatically flattened.
- JPEG, PNG, GIF, BMP, TGA, IFF, PICT and many other image formats are supported.

Video and Audio Formats

- Ogg Theora video is natively supported.
- Ogg Vorbis .ogg audio files are natively supported, and are ideally suited for soundtracks.
- Video MOV, AVI, ASF, MPG, MPEG, MP4VIDEO files are recoded by Unity with a configurable bitrate.
- Audio AIFF, WAV, MP3 and most other audio format are stored uncompressed, ideally suited for sound effects.

Other File Formats

- XML and text files with .xml and .txt extensions can be referenced at runtime.
- Any other file types, such as RTF and DOC, can be used for project notes and to-do lists.