

Game Level Design and Intro to Unity

MSP

Prof Vasilis Argyriou

Vasileios.Argyriou@kingston.ac.uk

Today's Lecture

- ☐ Introduction to Unity3D
- ☐ Game Objects
- ☐ Models, Materials, and Textures
- ☐ Textures, Shaders, and Materials



Introduction to Unity3D

Introduction to Unity3D

Install Unity

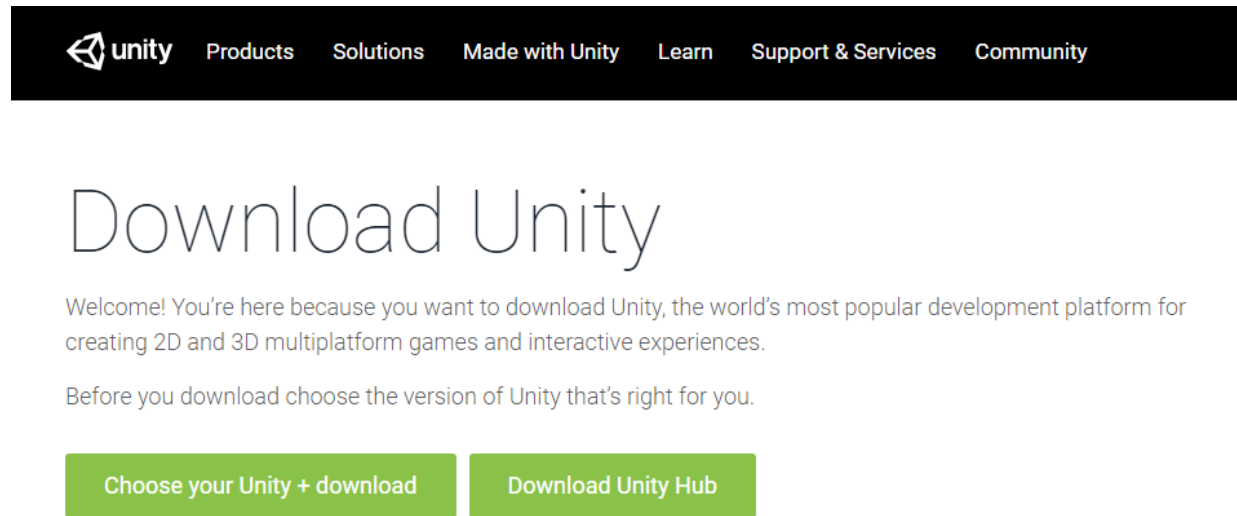
Create a new project or open an existing project

Use the Unity editor

Navigate inside the Unity Scene view

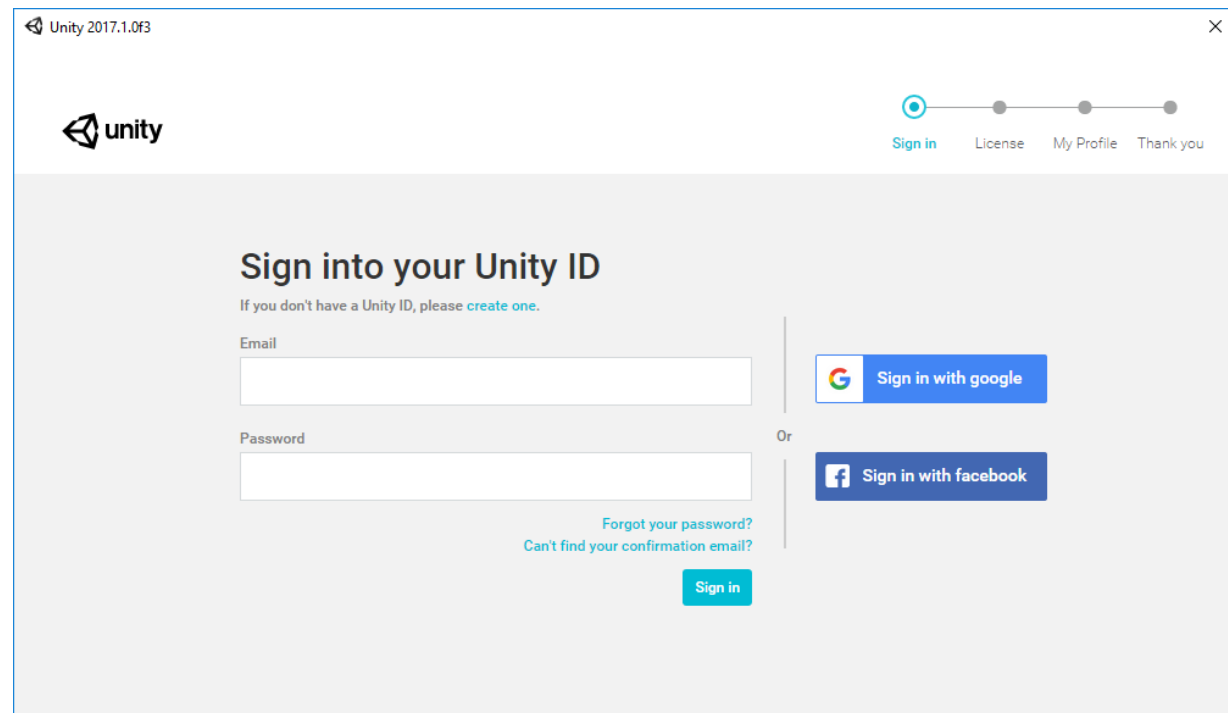
Downloading and Installing Unity

1. Download the Unity installer from the Unity3D download page at <http://unity3d.com/unity/download/> .
2. Run the installer and follow the prompts as you would with any other piece of software.



Downloading and Installing Unity

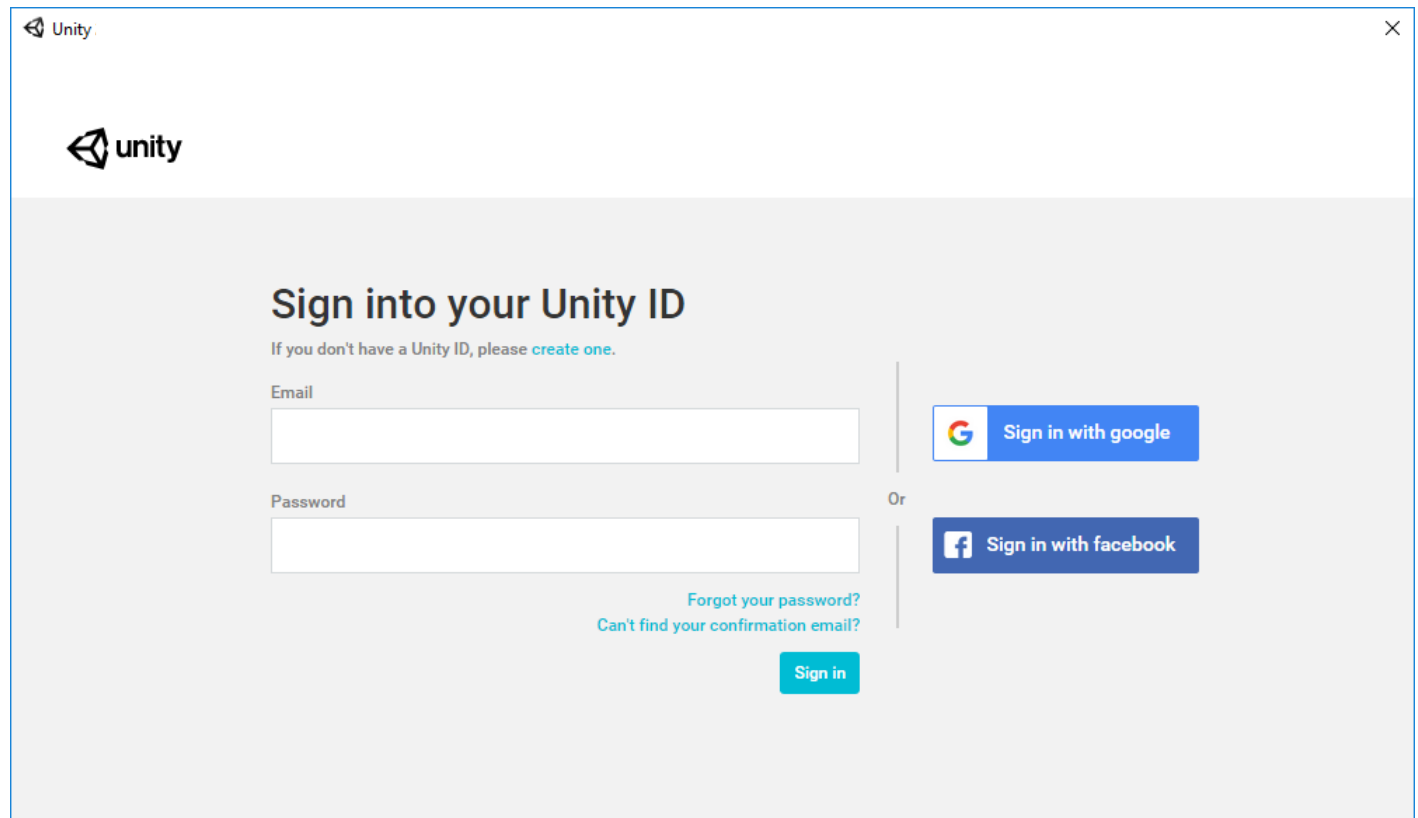
- Choose an install location for Unity and it is recommended that you leave the default
- When you run Unity for the first time, you will be asked to activate your license



The screenshot shows the Unity 2017.1.0f3 application window. The title bar reads "Unity 2017.1.0f3". The Unity logo is in the top left. In the top right, there is a progress bar with four steps: "Sign in" (active), "License", "My Profile", and "Thank you". The main content area is titled "Sign into your Unity ID" with a link "If you don't have a Unity ID, please [create one](#)." Below this are input fields for "Email" and "Password". To the right of these fields are two buttons: "Sign in with google" and "Sign in with facebook". Below the input fields, there are links for "Forgot your password?" and "Can't find your confirmation email?". A "Sign in" button is at the bottom center.

Downloading and Installing Unity

If you don't have an account, choose the Create Account option and fill out the required form



The screenshot shows the Unity login interface. At the top left is the Unity logo and name, and at the top right is a close button (X). Below the logo is the heading "Sign into your Unity ID" followed by the text "If you don't have a Unity ID, please [create one](#)." There are two input fields: "Email" and "Password". To the right of these fields are two buttons: "Sign in with google" and "Sign in with facebook". Below the input fields are two links: "Forgot your password?" and "Can't find your confirmation email?". At the bottom right is a "Sign in" button.

Unity

unity

Sign into your Unity ID

If you don't have a Unity ID, please [create one](#).

Email

Password

Sign in with google

Sign in with facebook

[Forgot your password?](#)

[Can't find your confirmation email?](#)

Sign in

Unity Editor

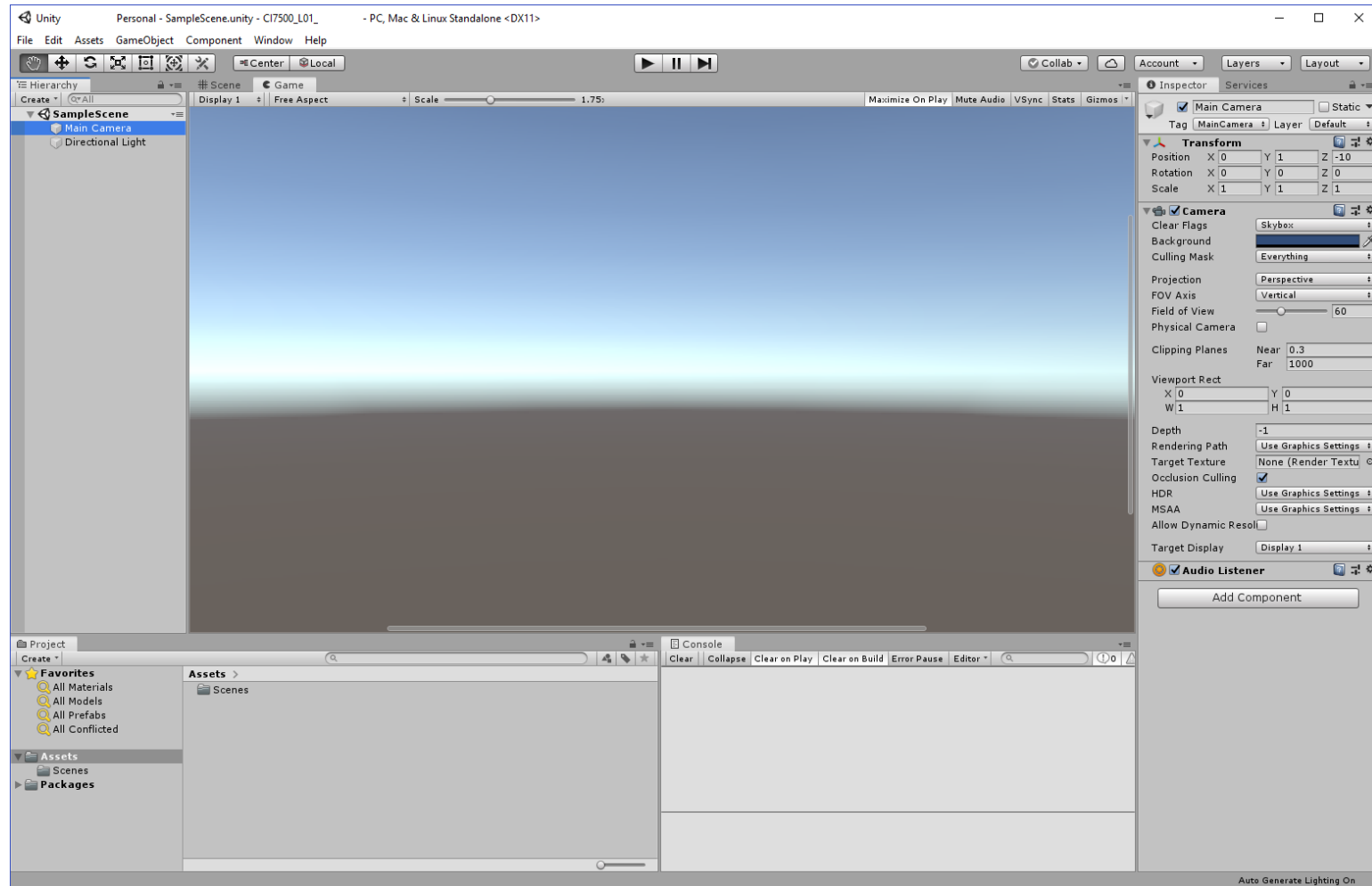
The Project Dialog

This window is what we use to open recent projects, browse for projects that have already been created, or start new projects.

If you want to open the Project dialog instead of the last project, you can do so by holding the **Alt** key while clicking the Unity icon.

Unity Editor

The Unity Interface



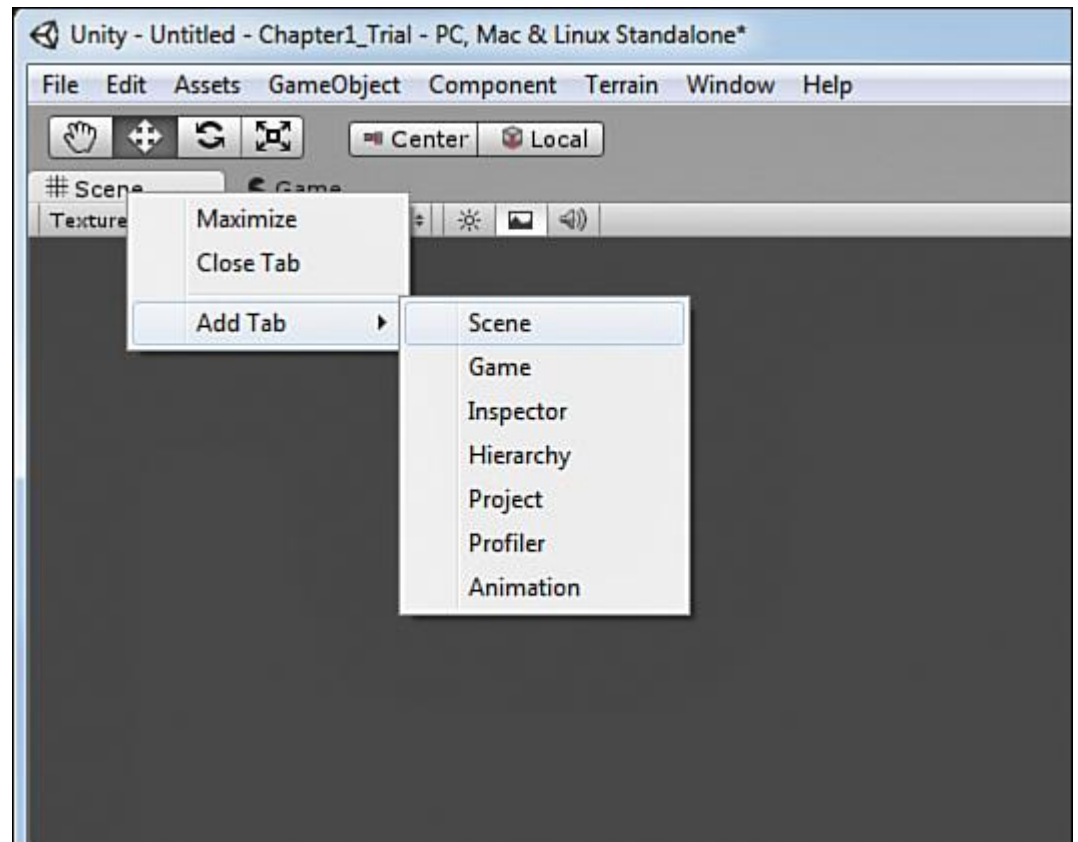
Unity Editor

The Unity Interface

- Unity allows the user to determine exactly how they want to work.
- You can quickly and easily switch back to the built-in default view by going to Window > Layouts > Default Layout.
- If you create a custom layout you like, you can always save it by going to Window > Layouts > Save Layout.

Unity Editor

You can simply right-click any view tab (the tab is the part sticking up with the views name on it), hover the mouse cursor over **Add Tab**, and a list of views will pop up for you to choose from



Unity Editor

The Project View

Everything that has been created for a project (files, scripts, textures, models, and so on) can be found in the Project view



Unity Editor

The Project View

- Unity mirrors the Project view with the folders on HD. If you create a file or folder in Unity, the corresponding one appears in the explorer (and vice versa).
- You can move items in the Project view simply by dragging and dropping.
- An **asset** is any item that exists as a file in your assets folder. All textures, meshes, sound files, scripts, and so on are considered assets. In contrast, if you create a game object, but it doesn't create a corresponding file, it is not an asset.

Unity Editor

The Project View

- Unity maintains links between the various assets associated with projects. As a result, moving or deleting items outside of Unity could cause potential problems. As a general rule, it is a good idea to do all of your asset management inside Unity.

Tip: Project Organization

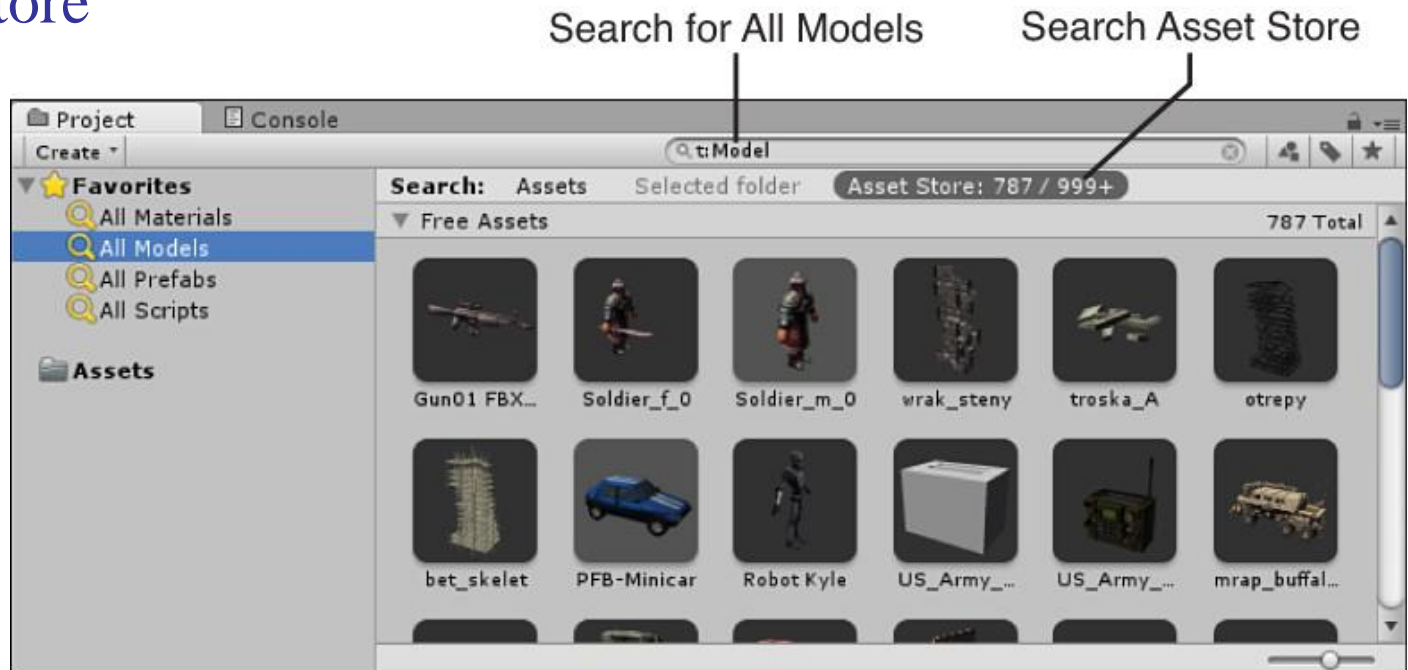
- Every asset type (scenes, scripts, textures, and so on) should get its own folder.
- Every asset should be in a folder.
- If you are going to use a folder inside another folder, make sure that the structure makes sense. Folders should become more specific and not be vague or generalized.

Unity Editor

The Project View

The Favorites buttons enable you to quickly select all assets of a certain type.

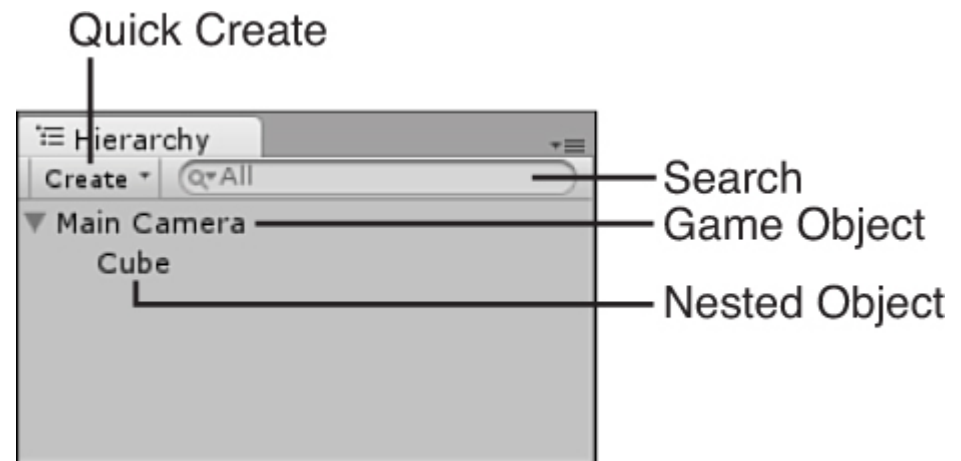
If you click Asset Store, you will be able to browse the assets that fit your search criteria from the Unity Asset Store



Unity Editor

The Hierarchy View

- The Hierarchy view shows all the items in the current scene instead of the entire project
- Just like with the Project view, you can use the Create menu to quickly add items to your scene, search using the built-in search bar, and click and drag items to organize and “nest” them.



Unity Editor

The Hierarchy View

- A scene is the term Unity uses to describe what you might already know as a level. As you develop a Unity project, each collection of objects and behaviors should be its own scene. Therefore, if you were building a game with a snow level and a jungle level, those would be separate scenes.
- The first thing you should do when working with a new Unity project is create a Scenes folder under Assets in the Project view. This way, all your scenes (or levels) will be stored in the same place. Be sure to give your scenes a descriptive name.

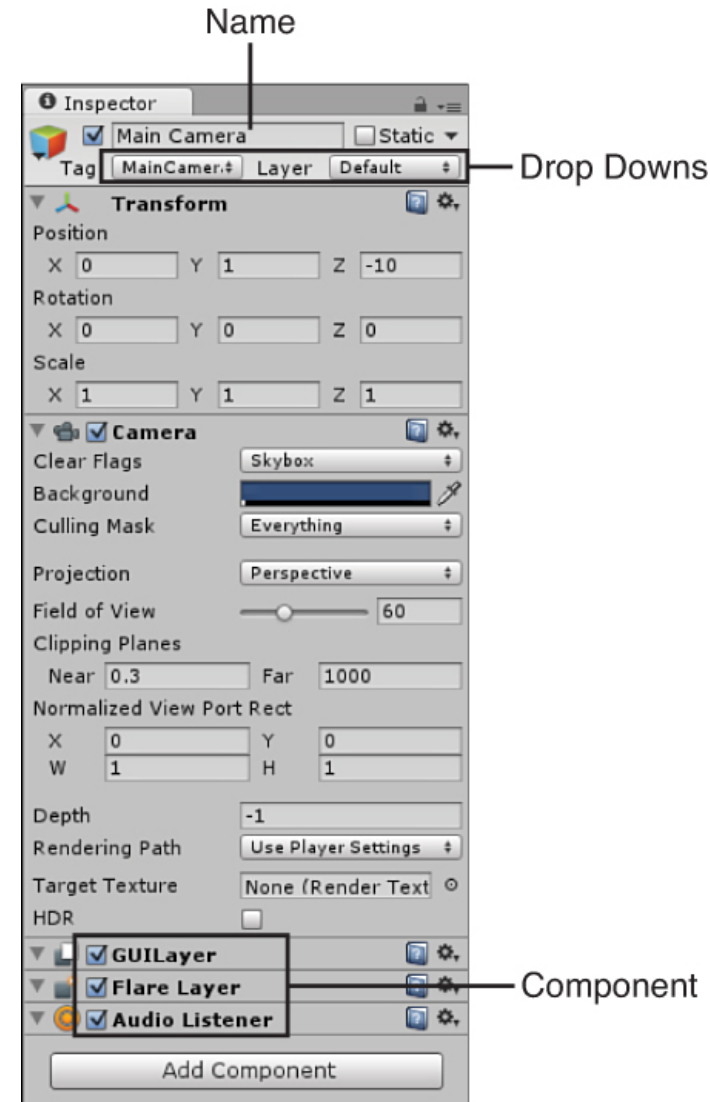
Unity Editor

The Inspector View

The Inspector view enables you to see all of the properties of a currently selected item.

Caution:

Changing Properties
While Running a Scene



Unity Editor

The Inspector View

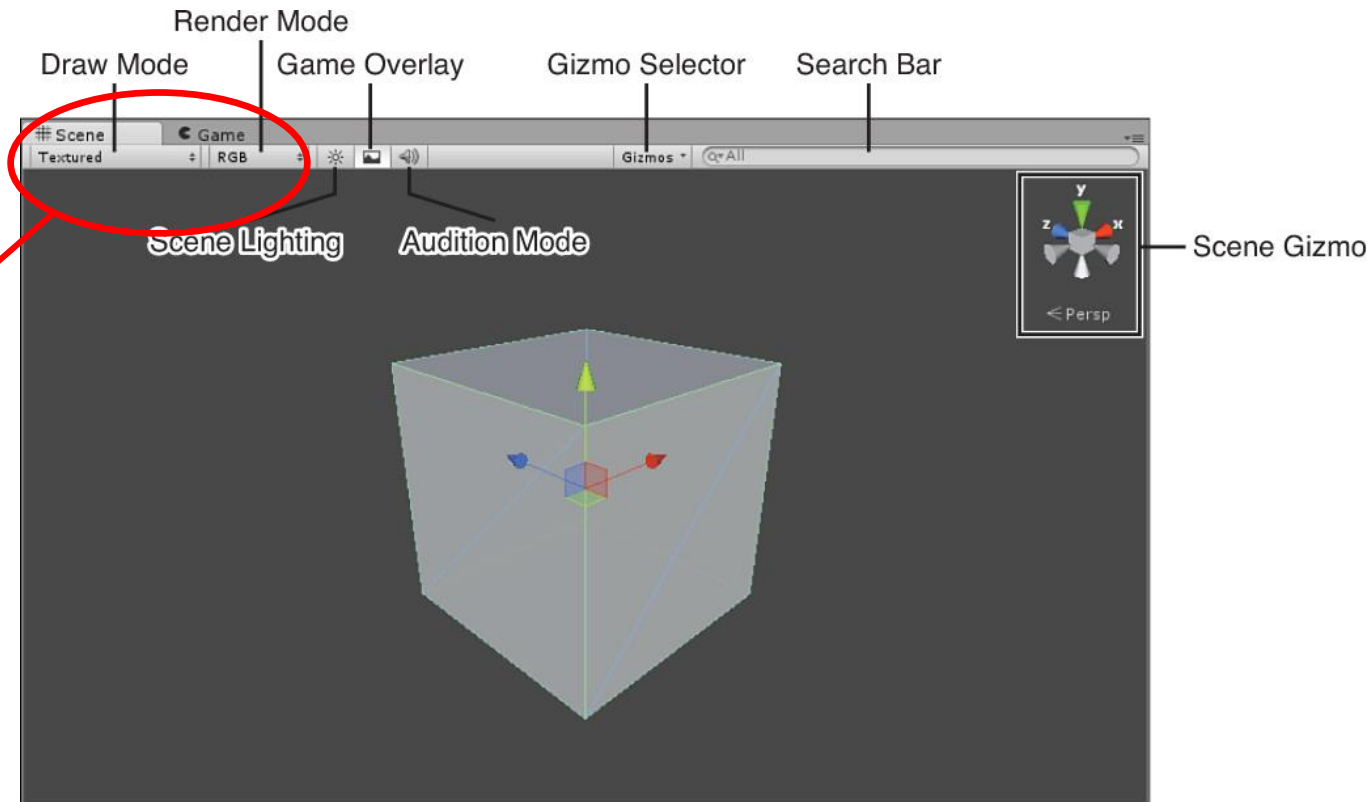
- If you click the check box next to the object's name, it will become disabled and not appear in the project.
- Drop-down lists (such as the Layer or Tag lists) are used to select from a set of predefined options.
- Text boxes, drop-downs, and sliders can have their values changed, and the changes will be automatically and immediately reflected in the scene
- Each game object acts like a container for different components (such as Transform, Camera, and GUILayer). You can disable these components by unchecking them.
- Components can be added by clicking the Add Component button.

Unity Editor

The Scene View

It enables you to see your game visually as it is being built. Using the mouse controls and a few hotkeys, you can move around inside your scene and place objects where you want them.

Shaded



Unity Editor

The Scene View

Shaded/Draw mode: This controls how the scene is depicted. By default it is set to Textured, which means objects will be drawn with their textures.

Shaded/Render mode: This controls how the objects in the scene are drawn. By default, the Render mode is RGB.

Scene lighting: This control determines whether objects in the Scene view will be lit by default ambient lighting or by lights that actually exist within the scene.

Game overlay: This determines whether items like skyboxes and graphical user interface (GUI) components appear in the Scene view. This also controls whether the placement grid is visible.

Unity Editor

The Scene View

Audition mode: This control sets whether an audio source in the Scene view functions or not.

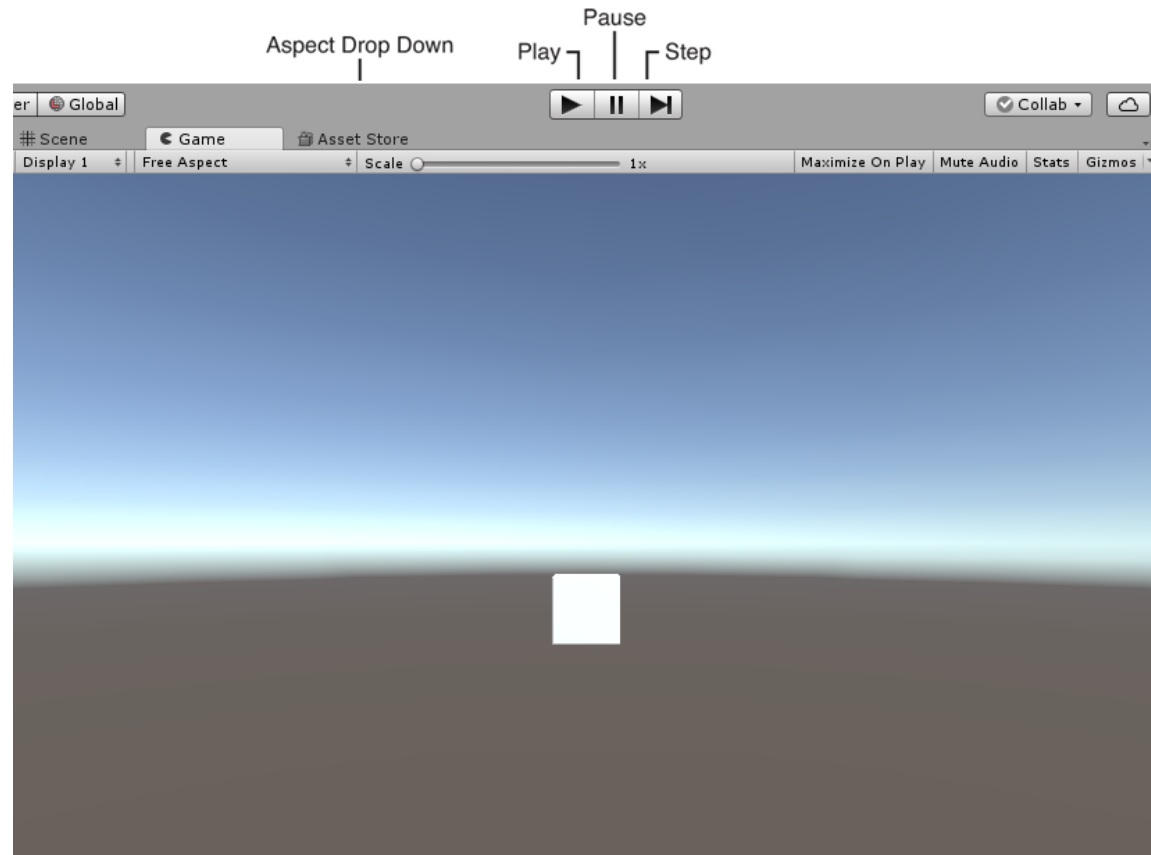
Gizmo selector: This control enables you to choose which “gizmos” appear in the Scene view. A gizmo is an indicator that gives visual debugging or aids in setup.

Scene gizmo: This control serves to show you which direction you are currently facing and to align the Scene view with an axis

Unity Editor

The Game View

It allows you to “play” the game inside the editor by giving you a full simulation of the current scene.



Unity Editor

The Game View

Play: The Play button enables you to play your current scene. To stop the game from running, click the Play button again.

Pause: The Pause button pauses the execution of the currently running Game view. Clicking the Pause button again will continue running the game.

Step: The Step button works while the Game view is paused and causes the game to execute a single frame of the game. Pressing the Step button while the game is running will cause the game to pause.

Aspect drop-down: From this drop-down menu, you can choose the aspect ratio you want the Game view window to display in while running.

Unity Editor

The Game View

Maximize on Play: This button determines whether the Game view takes up the entirety of the editor when run.

Stats: This button determines whether rendering statistics are displayed on the screen while the game is running. These statistics can be useful for measuring the efficiency of your scene.

Gizmos: This is both a button and a drop-down menu. The button determines whether gizmos are displayed while the game is running. The drop-down menu (the small arrow) on this button determines which gizmos appear if gizmos are turned on.

Unity Editor

The Toolbar

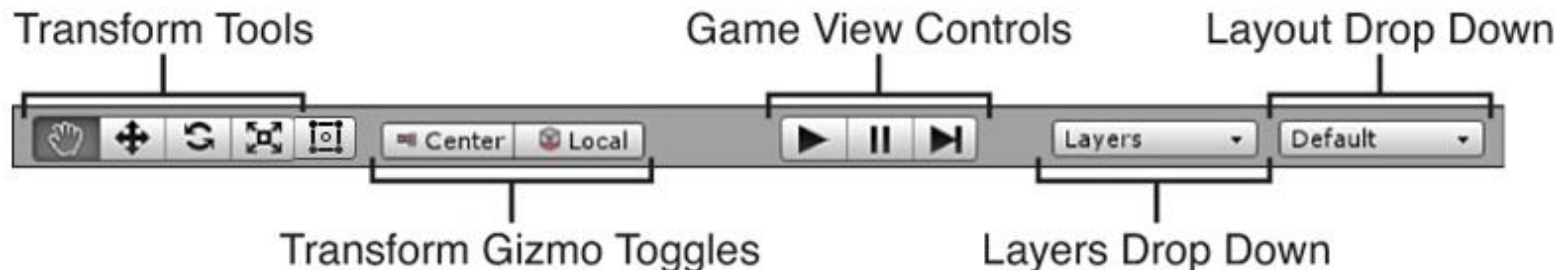
Transform tools: These buttons enable you manipulate game objects and are covered in greater detail later.

Transform gizmo toggles: These toggles manipulate how gizmos appear in the Scene view.

Game view controls: These buttons control the Game view.

Layers drop-down: This menu determines which object layers appear in the Scene view.

Layout drop-down: This menu allows you to quickly change the layout of the editor.



Unity Editor

The Hand Tool



The Hand tool (hotkey: **Q**) provides you a simple mechanic to move about the Scene view with the mouse

Action	Effect
Click-drag	Drags the camera around the scene
Hold Alt and click-drag	Orbits the camera around the current pivot point
Hold Ctrl (Command on Mac) and right-click-drag	Zooms the camera

- If you hold **Alt** while scrolling, however, you zoom in and out of wherever the mouse is currently pointing
- If you want to quickly navigate to, and zoom in on, a game object in your scene, you can do so by highlighting the object in the Hierarchy view and pressing **F**.

Unity Editor

Flythrough Mode

Flythrough mode enables you to move about the scene using a tradition first-person control scheme.

Action	Effect
Move the mouse	Causes the camera to pivot, which gives the feeling of “looking around” within the scene.
Press the WASD keys	The WASD keys move you about the scene. Each key corresponds with a direction: forward, left, back, and right, respectively.
Press the QE keys	The QE keys move you up and down, respectively, within the scene.
Hold Shift while pressing WASD or QE keys	Has the same effect as before, but it is much faster. Consider Shift to be your “sprint” button.

Game Objects

Game Objects

Every shape, model, light, camera, particle system, and so on in a Unity game all have one thing in common: **They are all game objects.**

The game object is the fundament unit of any scene.

Game objects are little more than a transform and a container.

This container exists to hold the various components that make objects more dynamic and meaningful

Creating Some Game Objects

1. Create a new project or create a new scene in a project you already have.
2. Add an empty game object by clicking the **GameObject** menu item and selecting **Create Empty** (or pressing **Ctrl+Shift+N**)
3. Look in the Inspector view and notice how the game object you just created has no components other than a transform. Clicking the **Add Component** button in the Inspector will show you all the components you could add to the object.
4. Add a cube to your project by clicking the **GameObject** menu item, and selecting **Cube** from the list.

Creating Some Game Objects

5. Notice the various components the cube has that the empty game object doesn't. The mesh components make the cube visible, and the collider makes it able to interact with other objects.

6. Add a point light to your project by clicking the **Create** drop-down in the Hierarchy view and selecting **Point Light** from the list.

7. You can see that the point light doesn't share any components with the cube and is instead focused entirely upon emitting light.

If no other light is available, you might also notice that your other objects went dark when the light was added to the scene. Because a light exists in the scene, Unity turns off its ambient lighting.

Transforms

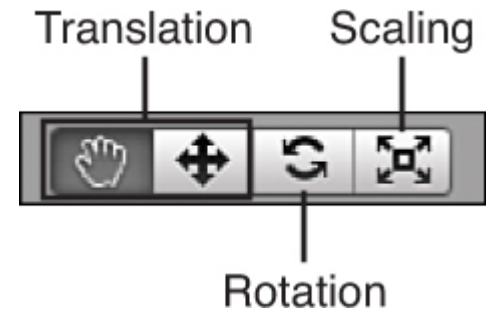
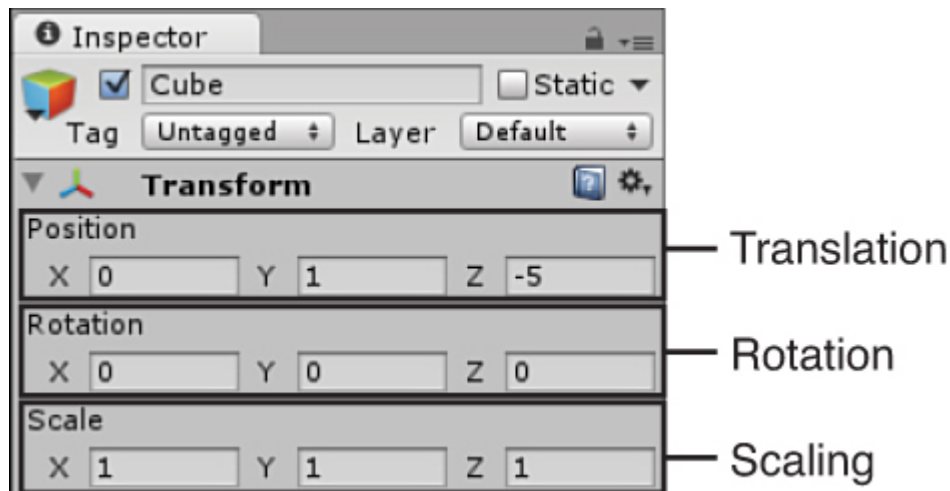
When dealing with 3D objects, you will often hear the term **transform**. Depending on the context, transform is either a noun or a verb.

All objects in 3D space have a **position**, a **rotation**, and a **scale**. If you combine them all together, you get an object's transform (noun).

Transform can be a verb if it refers to changing an object's **position**, **rotation**, or **scale**. Unity combines the two meanings of the word with the transform component.

Transforms

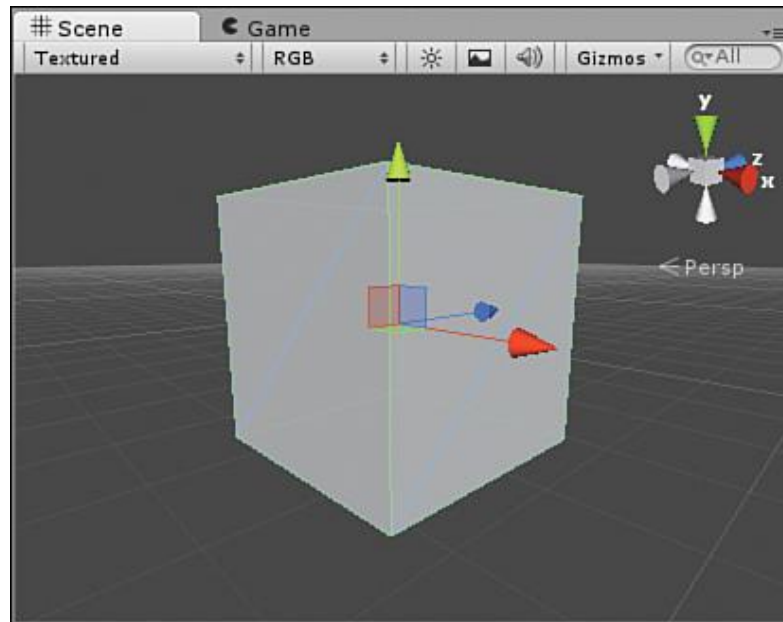
Because the transform is made up of the position, rotation, and scale, it stands to reason that there are three separate methods (called transformations) of changing the transform: translation, rotation, and scaling (respectively). These transformations can be achieved using either the Inspector or the transform tools



Translation

Three arrows pointing away from the center of the object along the three axes are the translation gizmos, and they help you move your objects around in the scene.

Clicking and holding on any of these axis arrows causes them to turn yellow. Then, if you move your mouse, the object will move along that axis



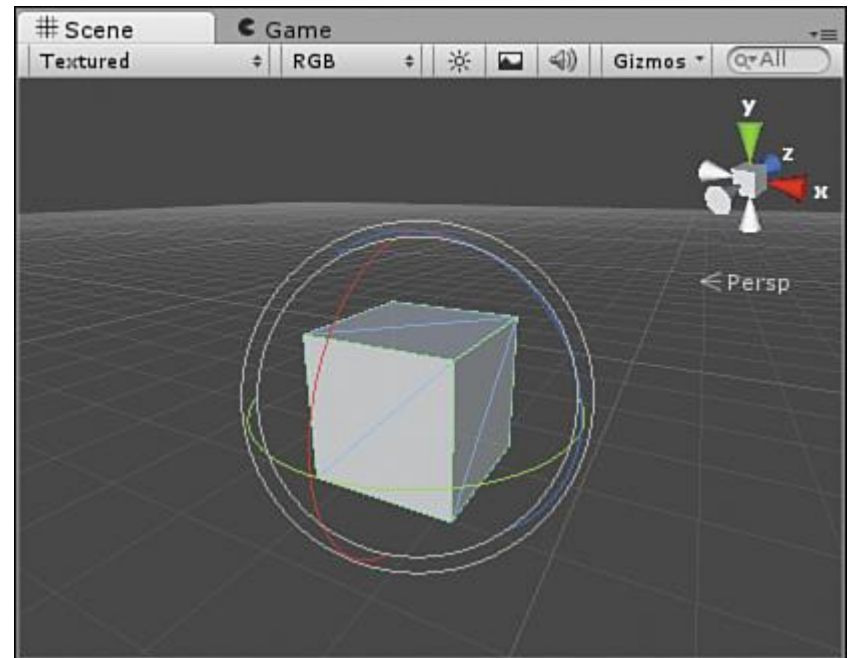
(Hotkey: **W**)

Rotation

Rotation enables you to redefine which direction the x, y, and z axes for a particular object point.

Rotation gizmos are circles representing the object's rotation path about the axes. **Clicking and dragging** on any of these circles turns them yellow and rotates the object about that axis.

(Hotkey: **E**)

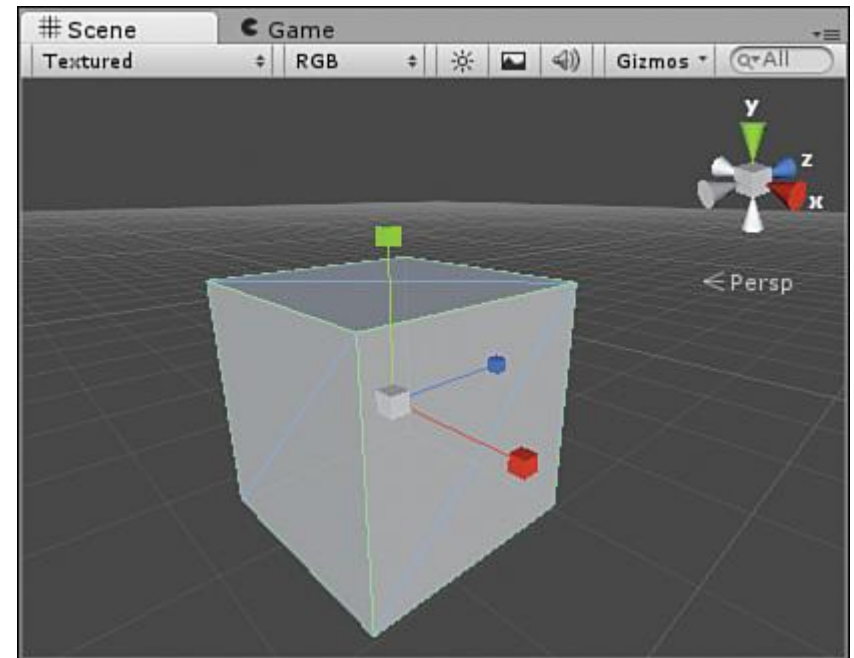


Scaling

Scaling causes an object to grow or shrink within a 3D space

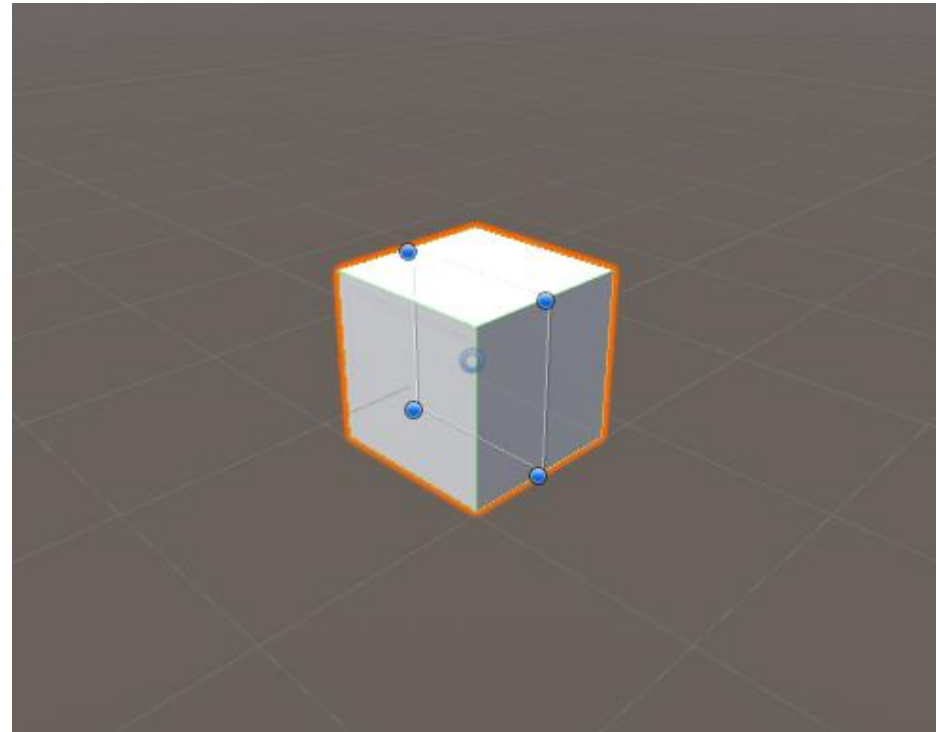
Scaling an object using the the scaling gizmos on any axis causes its size to change on that axis.

(Hotkey: **R**)



Parallel Scaling

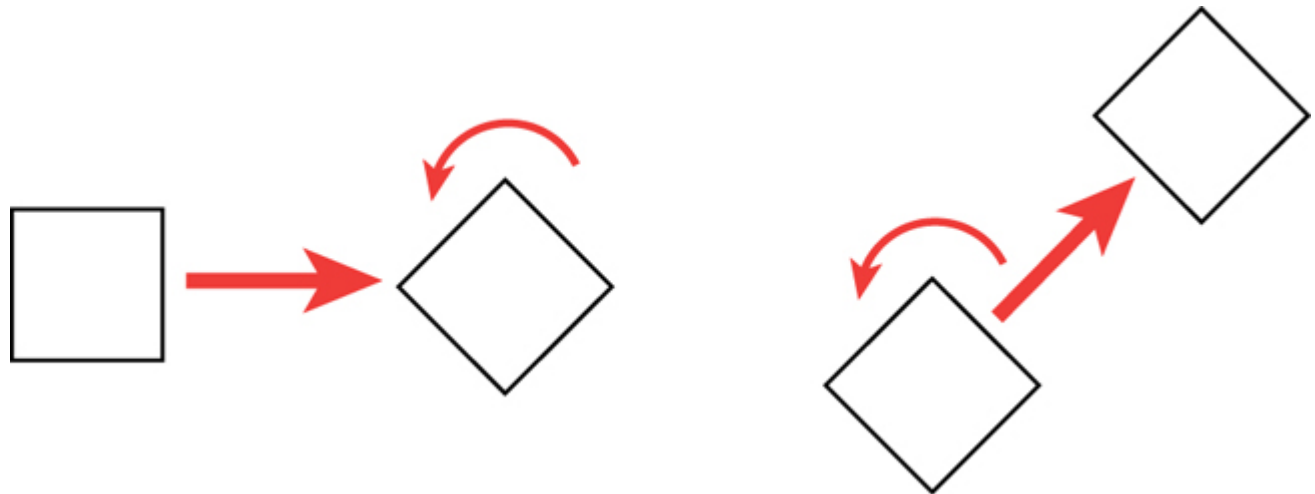
Scaling causes an object to grow or shrink within a 3D space



Hazards of Transformations

As mentioned before, transformations use the **local** coordinate system. Therefore, the changes that are made can potentially impact future transformations.

Not paying attention to transformation order can have unexpected consequences



Transforms and Nested Objects

To **nest** game objects in the Hierarchy view drag one object onto another one and that changes the way transformations work slightly.

Transformations applied to the parent object work as normal.

A child object's transform is relative to that of the parent object, not the world.

Therefore, a child object position is not based on its distance from the origin, but the distance from the parent object. If the parent object is rotated, the child object would move with it.

Transforms and Nested Objects

Remember, when a transformation is applied, it is not applied to the object, but to the object's coordinate system. An object isn't rotated, its coordinate system is. The effect is that the object turns.

When a child object's coordinate system is based on the local coordinate system of the parent, any changes to the parent system will directly change the child (without the child knowing about it).



Models, Materials, and Textures

Models, Materials, and Textures

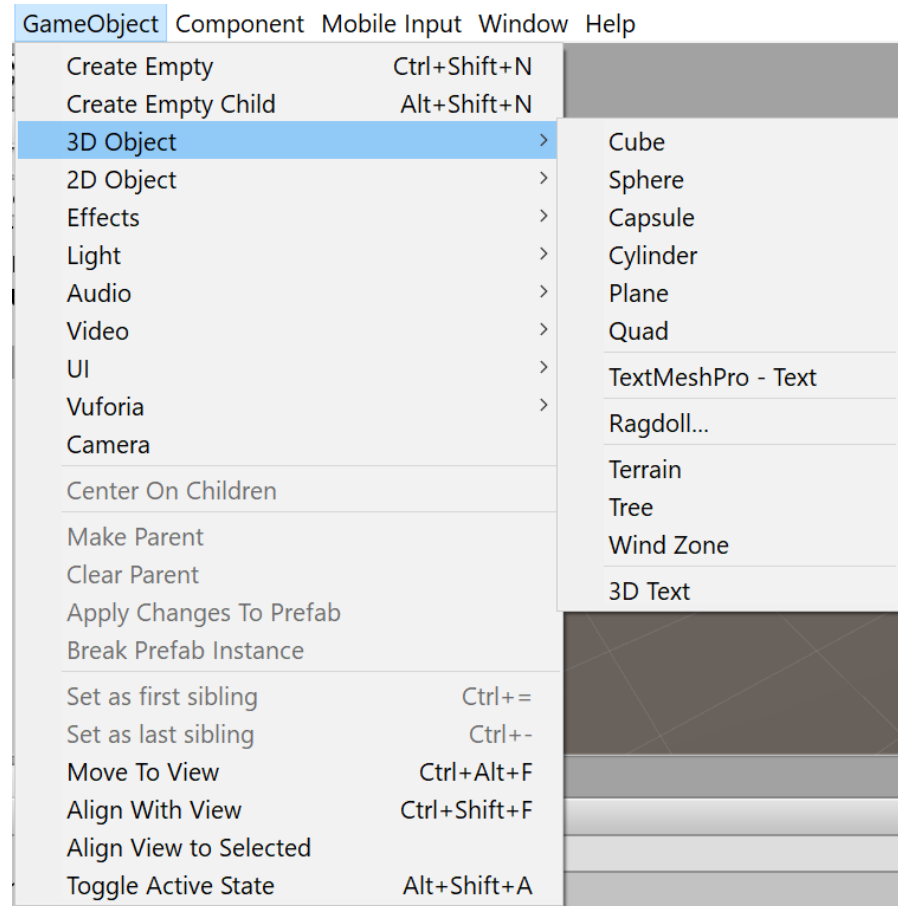
A **mesh**, at its most simple, is a series of interconnected triangles. These triangles build off of each other in strips to form basic to very complex objects. These strips provide the 3D definitions of a model and can be processed very quickly.

The terms **model** and **mesh** are similar, and you can often use them interchangeably. There is a difference, however. A mesh contains all the vertex information that defines the 3D shape of an object.

A model is an object that contains a mesh. A model has a mesh to define its dimensions, but it can also contain animations, textures, materials, shaders, and other meshes.

Built-In 3D Objects

Unity comes with a few basic built-in meshes (or primitives) for you work with. These tend to be simple shapes that serve simple utilities or can be combined to make more-complex objects.



Built-In 3D Objects

Nesting objects in Unity enables you to easily make simple models using the built-in meshes.

Just place the meshes near each other so that they form the rough look you want.

Then nest all the objects under one central object. This way, when you move the parent, all the children move, too.

Importing Models

Unity makes it rather easy to bring your own 3D models into your projects.

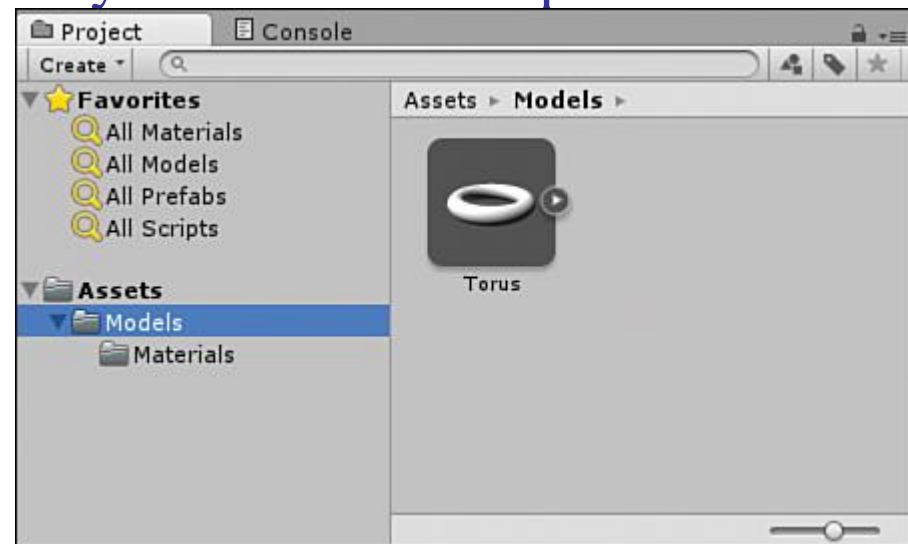
Just placing the file containing the 3D model in your Assets folder is enough to bring it into the project.

From there, dragging it into the scene or hierarchy builds a game object around it.

Natively, Unity supports .fbx, .dae, .3ds, .dxf, and .obj files. This enables you to work with just about any 3D modeling tool.

Importing Your Own 3D Model

1. Create a new Unity project or scene.
2. In the Project view, create a new folder named **Models** under the Assets folder. (Right-click the Assets folder and select **Create > Folder**.)
3. Locate the **Torus.fbx** file provided on Canvas.
4. Click and drag the **Torus.fbx** file from the file browser into the Models folder that you created in step 2. In Unity, click the **Models** folder to see the new Torus file.



Importing Your Own 3D Model

5. Click the **Torus** asset in the Models folder and look at the Inspector view. Change the value of the scale factor from 0.01 to 1 (or 100) and click **Apply**.

6. Drag the **Torus** asset from the Models folder onto the Scene view. Notice how a Torus game object was added to the scene containing a mesh filter and mesh rendered. These allow the Torus to be drawn to the screen.

If you click the Torus object, you see how it is made up of many connected triangles.

Models and the Asset Store

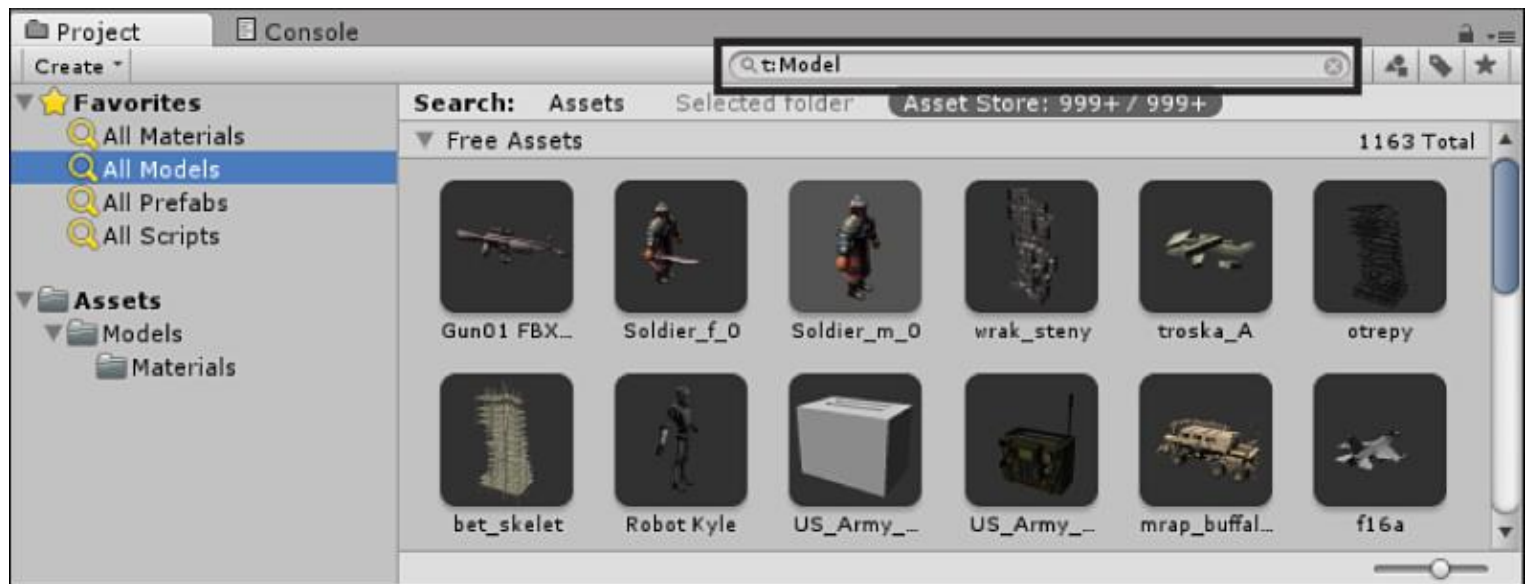
The Asset Store provides a simple and effective way to find premade models and import them into your project.

Models on the Asset Store are either free or paid and come alone or in a collection of similar models.

Some of the models come with their own textures, and some of them are simply the mesh data.

Models and the Asset Store

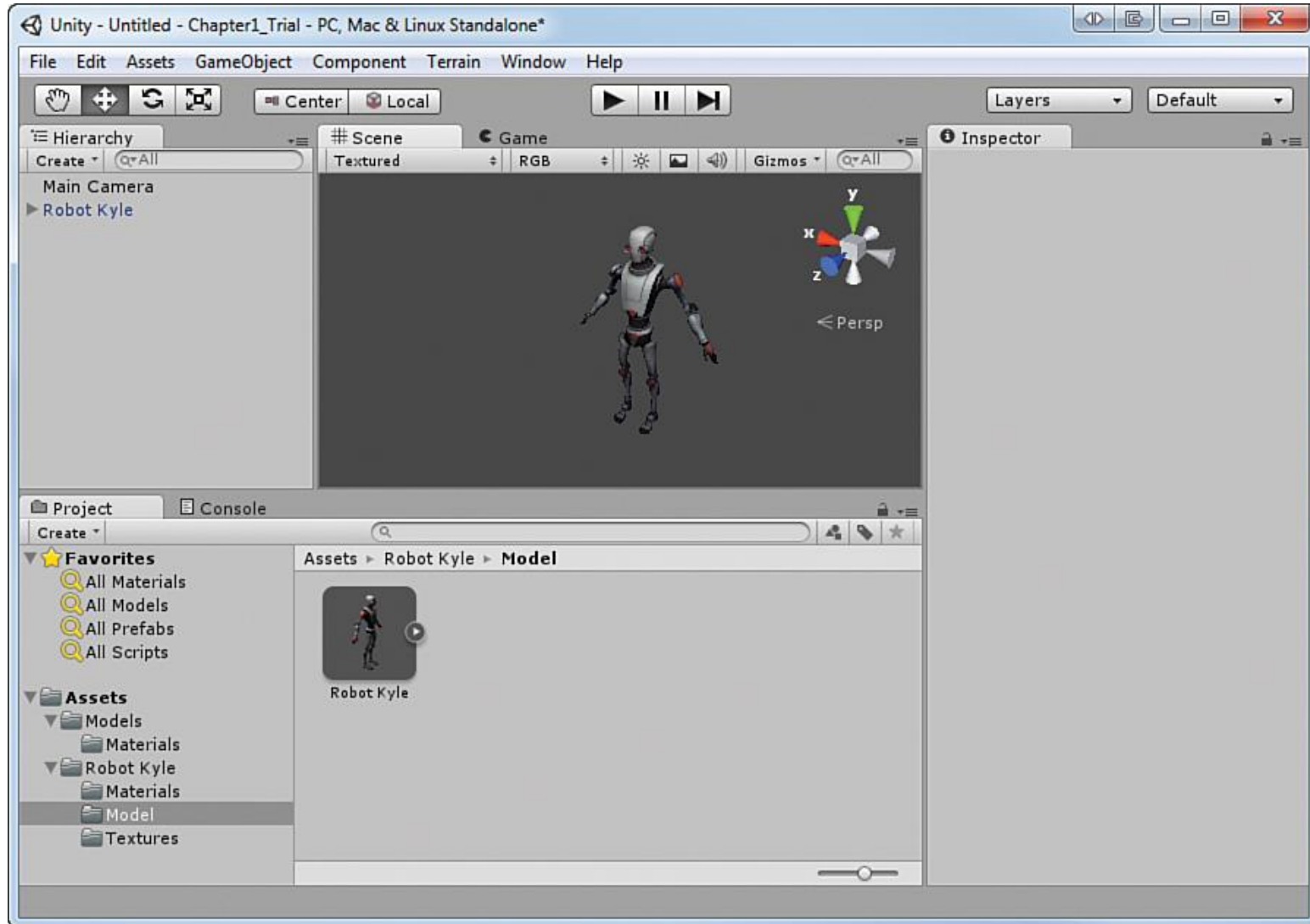
1. Create a new scene (click **File > New Scene**). In the Project view, type **t:Model** into the search bar
2. In the search filter section, click the Asset Store: 999+/999+ button.



Models and the Asset Store

3. Locate **Robot Kyle** and select it.
4. In the Inspector view, click **Import Package**. At this point, you may be prompted to provide your Unity account credentials.
5. When the Importing Package dialog opens, leave everything checked and select **Import**.
6. There will now be a new asset folder called Robot Kyle. Locate the robot model under **Assets > Robot Kyle > Model** and drag it into the Scene view.

Models and the Asset Store

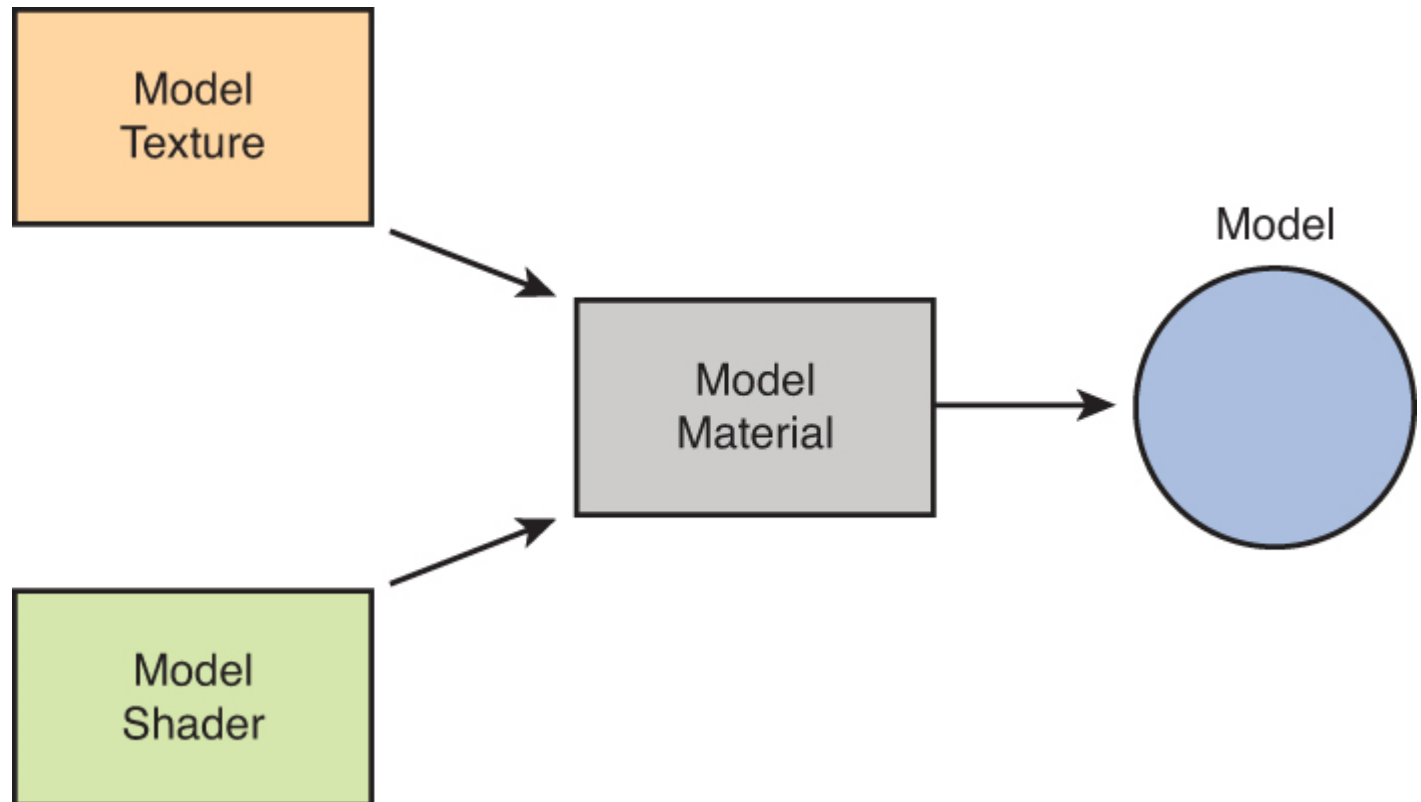




Textures, Shaders, and Materials

Textures, Shaders, and Materials

Graphical assets are broken down into textures, shaders, and materials



Textures

Textures are flat images that get applied to 3D objects.

Start by creating a folder for your textures; a good name would be **Textures**. Then drag any textures you want in your project into the **Textures** folder you just created.

When creating an intricate model, it is common to generate something called an unwrap. The unwrap is somewhat akin to a map that shows you exactly how a flat texture will wrap back around a model.

If you look in the **Robot Kyle > Textures** folder, you notice the Robot_Color texture. It looks strange, but that is the unwrapped texture for the model

Shaders

If the texture of a model determines what is drawn on its surface, the shader is what determines how it is drawn.

A material contains properties and textures, and shaders dictate what properties and textures a material can have.

Materials

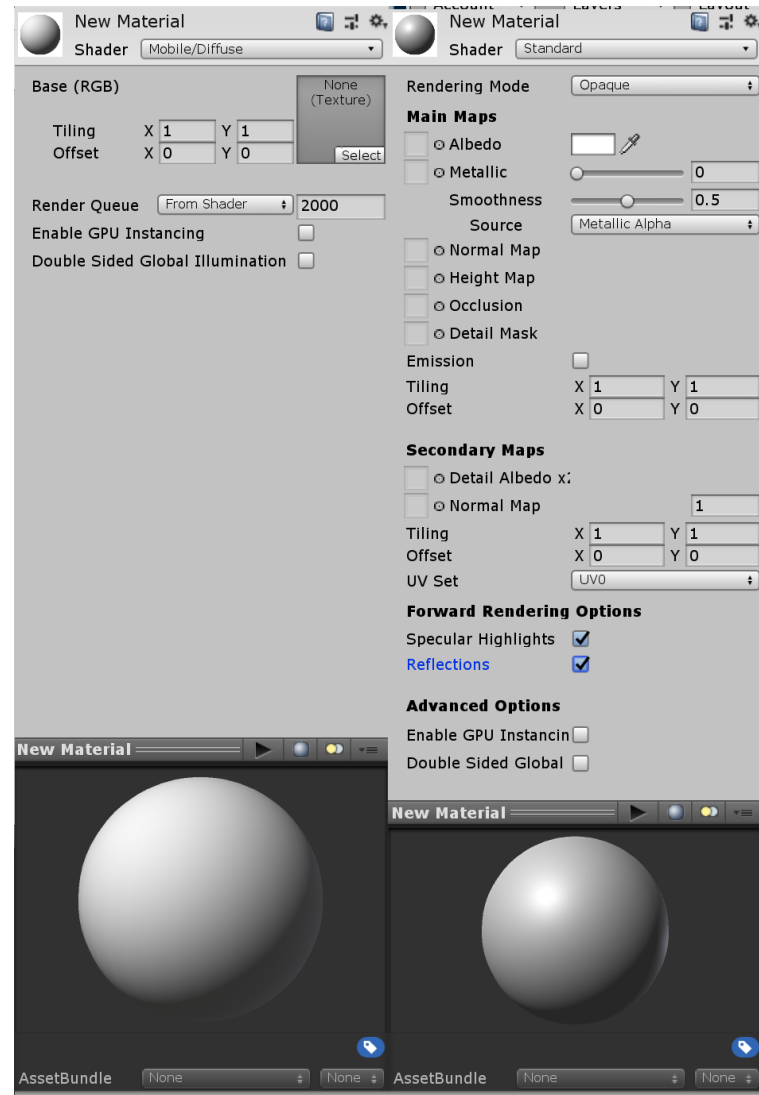
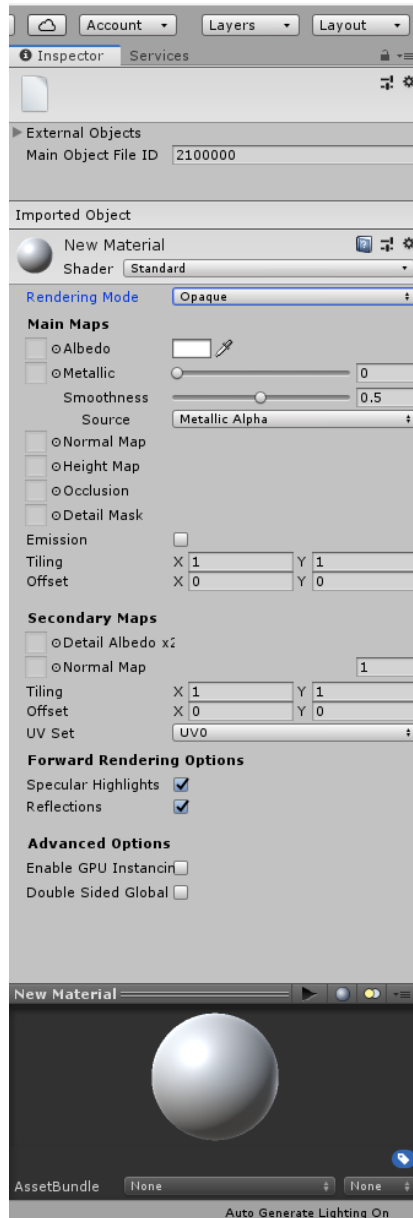
Materials are not much more than containers for shaders and textures that can be applied to models.

Most of the customization of materials depends on which shader is chosen for it, although all shaders have some common functionality

To create a new material, start by making a Materials folder. Then right-click the folder and select **Create > Material**.

Materials have a base texture, main color, tiling and offsets, and a preview of the material

Materials



Shaders

Some of the basic shaders

Shader	Description
Diffuse	Diffuse is the default shader for materials and is also the most basic. Light is evenly distributed across the diffuse object's surface.
Specular	Specular textures make an object look shiny. If you want to make an object seem to reflect a lot of light, this is the shader to use.
Bumped	Bumped shaders are generally used in conjunction with other shaders (as in bumped-diffuse or bumped-specular). These shaders use a normal map to give the flat texture a 3D, or bumpy, look. These are a great way to give your models a lot of physical detail without requiring complex modeling.

Shaders

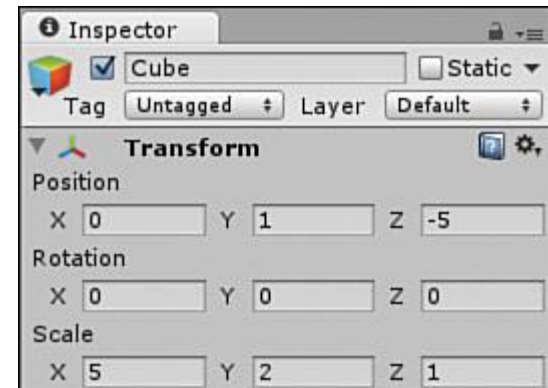
Common shader properties

Property	Description
Main Color	The Main Color property defines what color of ambient light shines on the object. This does not change the color of the object itself; it just makes the object appear different. For example, an object with a blue texture and a yellow main color will not turn yellow but green (because blue with yellow light looks green). If you want your model's color to remain unchanged, select white.
Specular Color	The Specular Color property determines what color the “shiny” parts of a specular model are. Generally speaking, this will be white unless you intend for it to appear as if another color of light is shining on the object.
Shininess	The Shininess property determines how shiny a specular object is.
Texture	The Texture block contains the texture you want to apply to your model.
Normal Map	The Normal Map block contains the normal map that will be applied to your model. A normal map can be used to apply bumpiness to a model. This is useful when calculating lighting to give the model more detail than it would otherwise have.
Tiling	The Tiling property defines how often a texture can repeat on a model. It can repeat in both the x and y axes.
Offset	The Offset property defines whether a gap will exist between edges of the object and the texture.

Textures, Shaders, and Materials

Applying Textures, Shaders, and Materials to Models

1. Start a new project or scene. Note that creating a new project will close and reopen the editor.
2. Create a Textures and a Materials folder.
3. Locate the Brick_Texture.png file on Canvas and drag it into the Textures folder created in step 2.
4. Add a cube to the scene. Position it at (0, 1, -5). Give it a scale of (5, 2, 1).

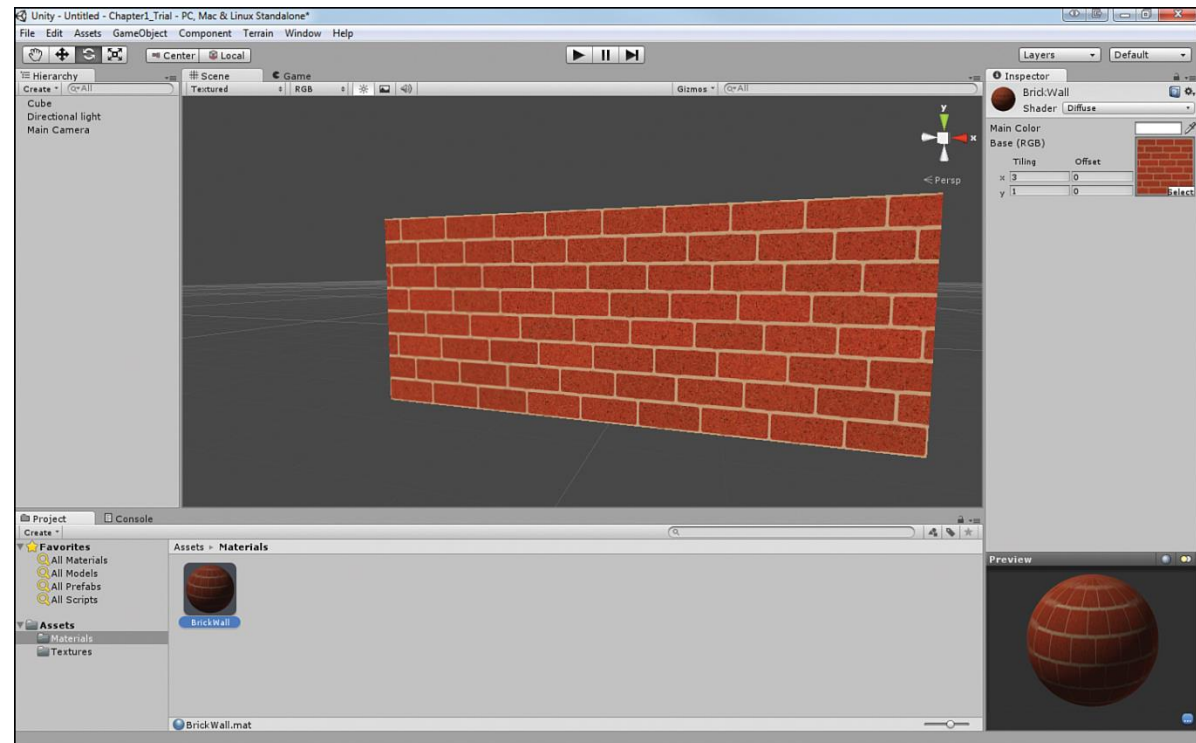


Textures, Shaders, and Materials

5. Create a new material (right-click the Materials folder and select **Create > Material**) and name it **BrickWall**.
6. Leave the shader as Diffuse, and in the texture block click **Select**. Select **Brick_Texture** from the pop-up window.
7. Click and drag the brick wall material from the Project view onto the cube in the Scene view.
8. Notice how the texture is stretched across the wall a little too much. With the material selected, change the value of the x tiling to be **3**.

Textures, Shaders, and Materials

9. Add a directional light to your scene if it is not already there (click **GameObject** > **Light** > **Directional Light**). Position it at (0, 10, -10) and give it a rotation of (30, 0, 0). We will discuss lighting more in a later hour.



Summary

- ☐ Introduction to Unity3D
- ☐ Game Objects
- ☐ Models, Materials, and Textures
- ☐ Textures, Shaders, and Materials