

# Computer Game Programming (SE3173)

Ibrar Arshad  
Ibrar.arshad@cust.edu.pk

# Creating Gameplay

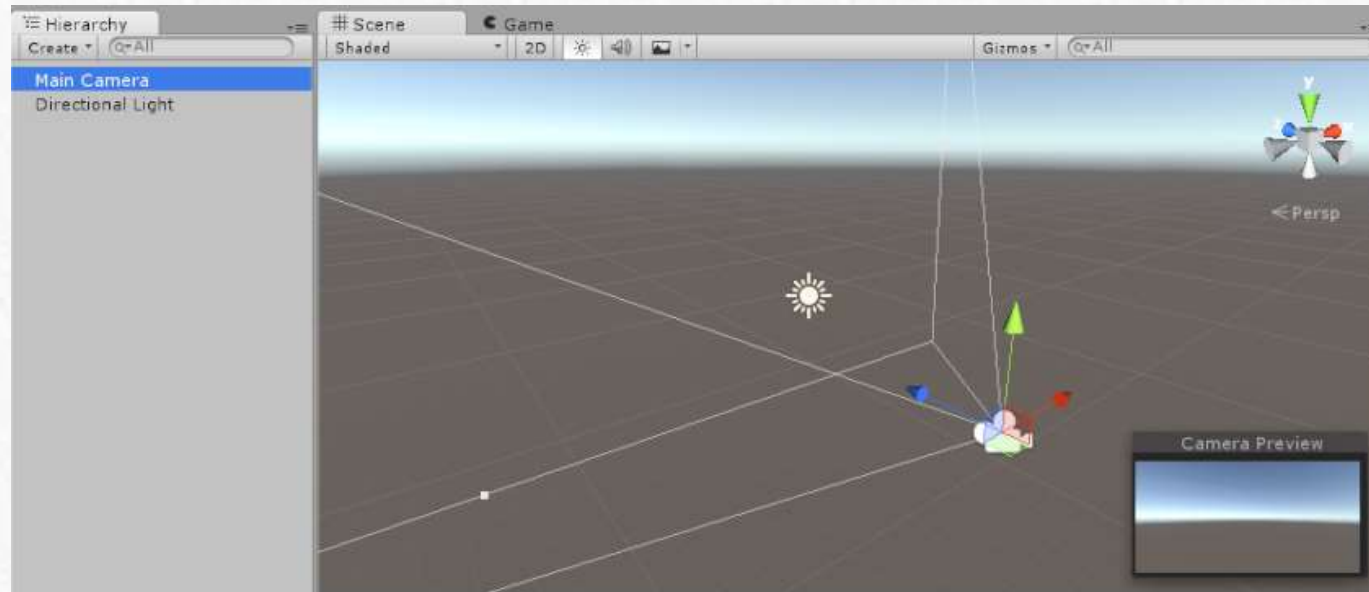
---

- Unity empowers game designers to make games
- What's really special about Unity is that **you don't need years of experience with code or a degree in art to make fun games**
- There are a handful of **basic workflow concepts needed** to learn Unity
  - Once understood, you will find yourself making games in no time
- With the time you will save getting your games up and running, you will have that much more time to refine, balance, and tweak your game to perfection

# Unity – Scenes

---

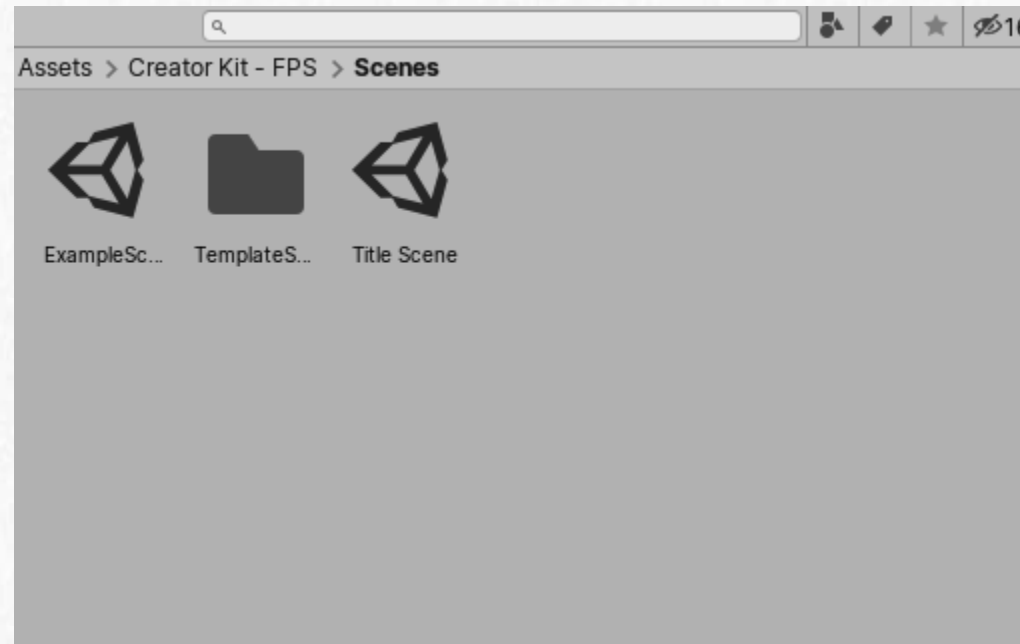
- **Scenes** are where you work with content in Unity
- They are the assets that contain all or part of a game or application
  - For example, you might build a simple game in **a single scene**, while for a more complex game, you might use **one scene per level**, each with its **own environments, characters, obstacles, decorations, and UI**
- You can create any number of scenes in a project.



# Creating a New Scene

---

- Use the [New Scene Dialog](#) to create a new scene from a specific scene template
- By default, the New Scene dialog opens when you create a new scene from the menu ([File > New Scene](#))
- To save the scene you're currently working on, choose [File > Save Scene](#) from the menu



# GameObjects

- The **GameObject** is the most important concept in the Unity Editor.
- Every object in your game is a **GameObject**, from characters and collectible items to lights, **Cameras** and special effects
- However, a **GameObject** can't do anything on its own; you need to give it properties before it can become a character, an environment, or a special effect.





# GameObjects

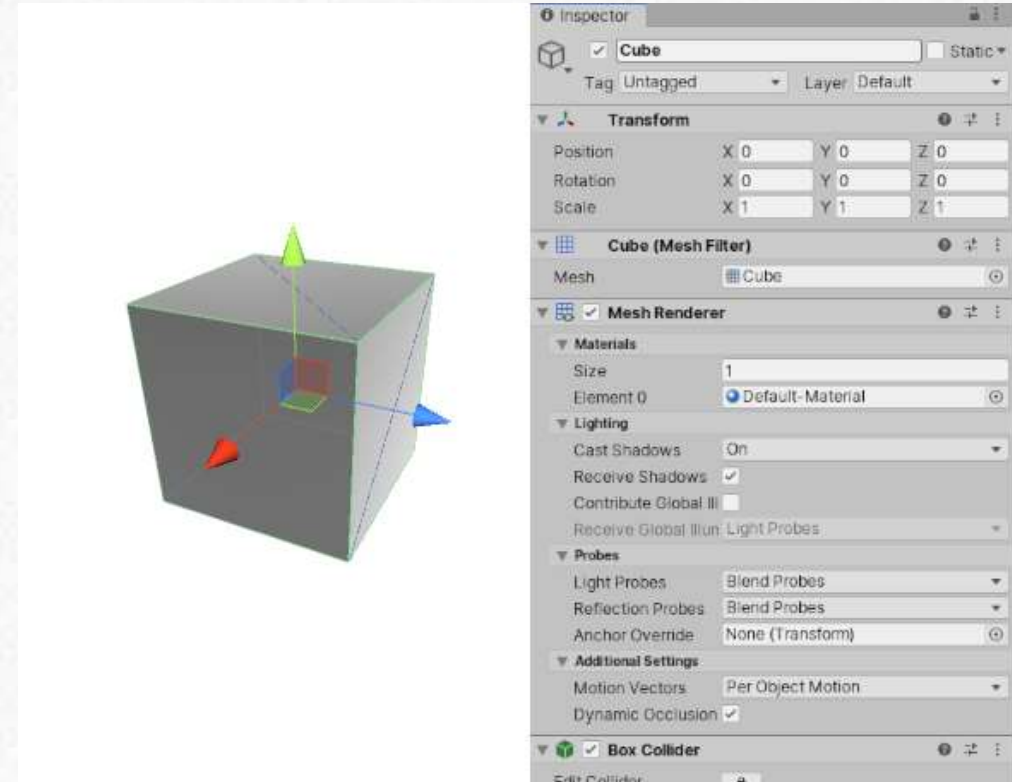
---

- **GameObjects** are the fundamental objects in Unity that represent **characters, props and scenery**
  - They do not accomplish much in themselves, but they act as **containers for Components**, which implement the functionality
- To give a GameObject the properties it needs to **become a light, or a tree, or a camera**, you need to add components to it
  - Depending on what kind of object you want to create, you add different combinations of components to a GameObject.
  - Unity has lots of different built-in component types

# GameObjects (Solid Cube)

---

- A solid cube object has a **Mesh Filter** and **Mesh Renderer** component, to draw the surface of the cube, and a **Box Collider** component to represent the **object's solid volume** in terms of physics.



# GameObjects Components

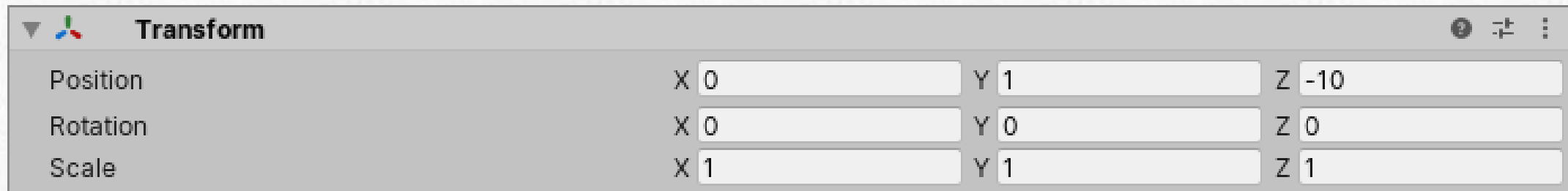
---

- A GameObject always has a **Transform component** attached (to represent position and orientation) and it is not possible to remove this.
- The other components that give the object its functionality can be added from the editor's **Component menu**



# Transforms

- The **Transform** is used to store a GameObject's **position, rotation, scale** and parenting state and is thus very important
- A GameObject will always have a **Transform component** attached - it is not possible to remove a Transform or to create a GameObject without one.
- The **Transform** component determines the **Position, Rotation, and Scale** of each object in the scene
- Every GameObject has a Transform.



The image shows a screenshot of the Unity Transform component inspector. The title bar says 'Transform' with a dropdown arrow on the left and icons for help, expand, and menu on the right. The inspector is divided into three sections: Position, Rotation, and Scale. Each section has three input fields for X, Y, and Z values.

Property	X	Y	Z
Position	0	1	-10
Rotation	0	0	0
Scale	1	1	1

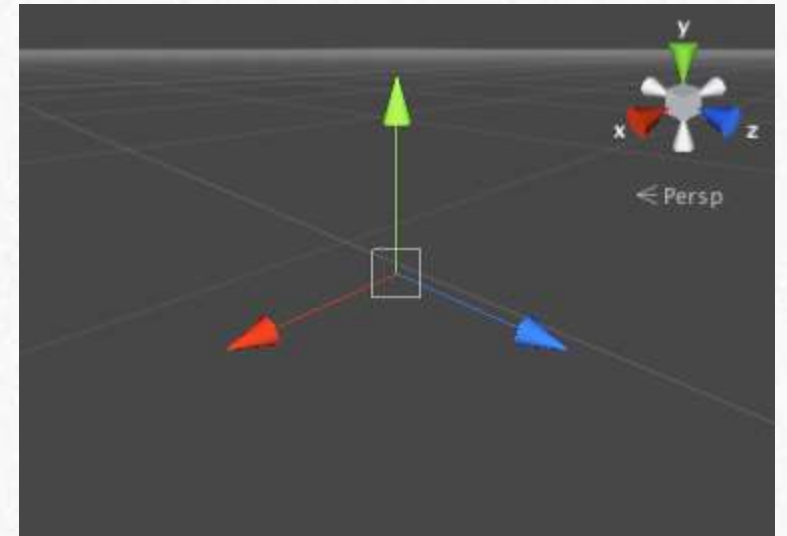
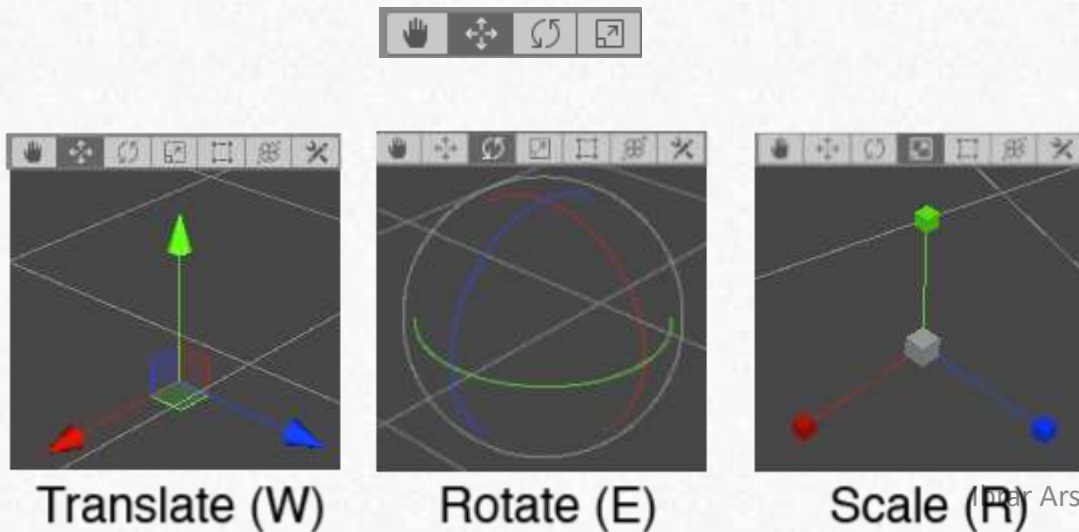
# Transforms Properties

---

Property:	Function:
<b>Position</b>	Position of the Transform in X, Y, and Z coordinates.
<b>Rotation</b>	Rotation of the Transform around the X, Y, and Z axes, measured in degrees.
<b>Scale</b>	Scale of the Transform along X, Y, and Z axes. Value “1” is the original size (size at which the object was imported).

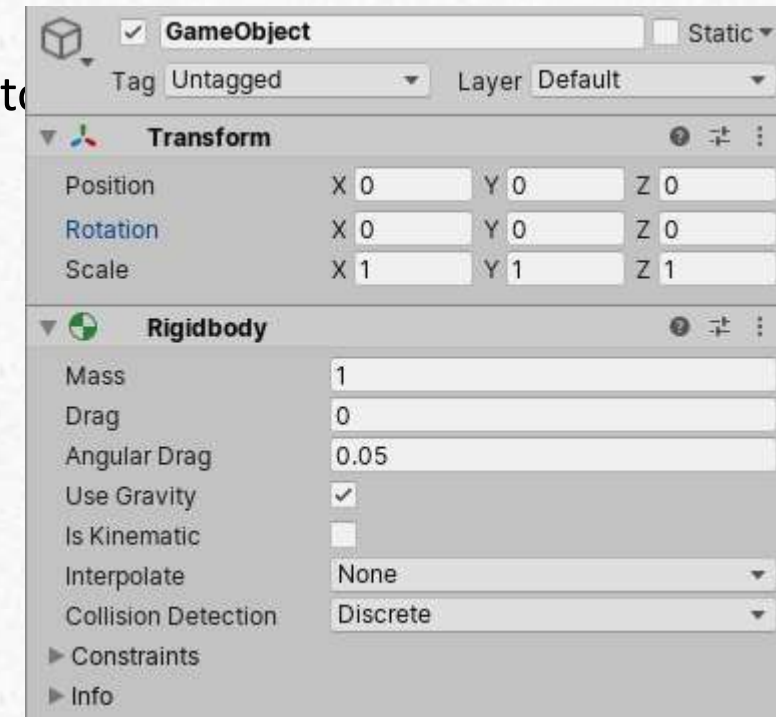
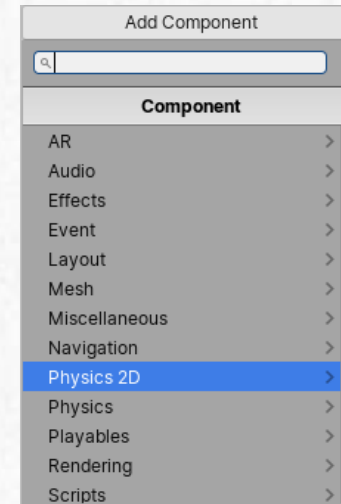
# Editing Transform

- Transforms are manipulated in 3D space in the X, Y, and Z axes or in 2D space in just X and Y. In Unity, these axes are represented by the colors red, green, and blue respectively.
- A Transform can be edited in the [Scene View](#) or by changing its properties in the [Inspector](#)
- In the scene, you can modify Transforms using the Move, Rotate and Scale tools. These tools are located in the upper left-hand corner of the Unity Editor.



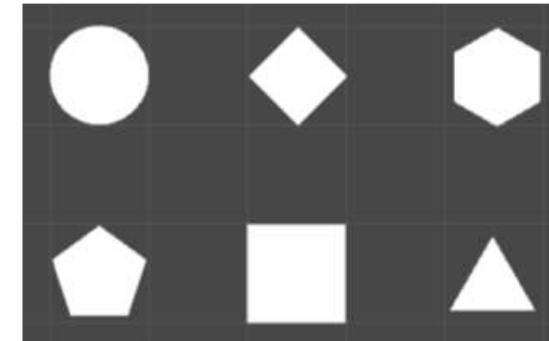
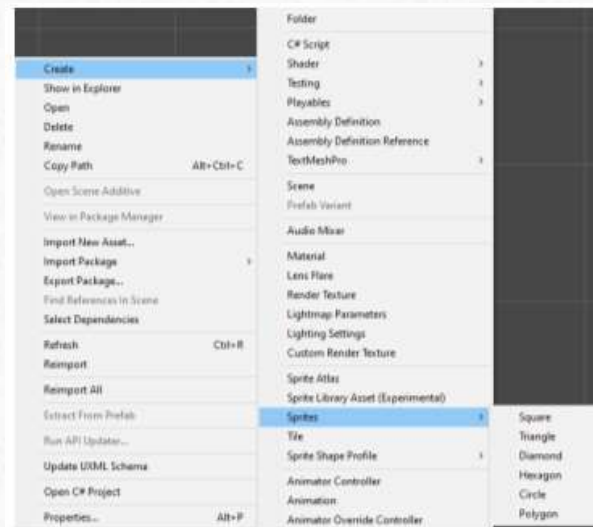
# Using Components

- **Components** are the nuts & bolts of objects and behaviors in a game
- They are the **functional pieces of every GameObject**
- You can add Components to the selected GameObject through the Components menu
  - Try this now by adding a **Rigidbody** to the empty GameObject we just created
  - Select it and choose **Component->Physics->Rigidbody** from the menu.
  - When you do, you will see the Rigidbody's properties appear in the Inspector



# 2D Primitive GameObjects

- Unity provides 2D Primitive **GameObject**s to help you prototype your Project quickly, without needing to import your own Assets
- To create a 2D primitive, go to **GameObject > 2D Object > Sprites** and select one of the following option
  - Square
  - Circle
  - Capsule
  - Isometric Diamond
  - Hexagon Flat-Top
  - Hexagon Point-Top
  - 9-Sliced

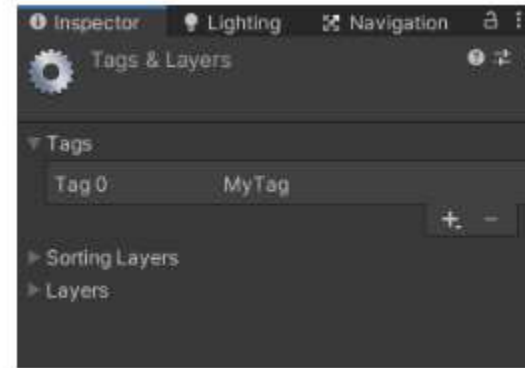
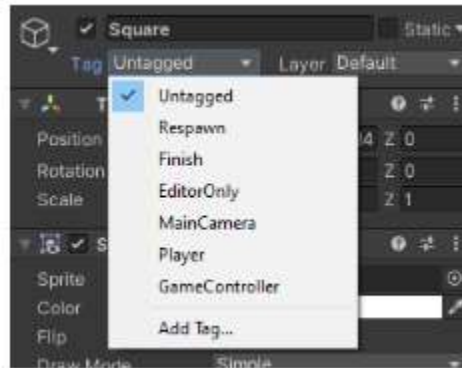




# Tags

---

- A Tag is a reference word which you can assign to one or more GameObjects.
  - Tags are useful to identify GameObjects for scripting purposes: finding an object by its tag, identify collision objects, etc.
  - Once you name a Tag, it cannot be renamed later.
  - A GameObject can only have one Tag assigned to it



# Prefabs

---

- Unity's **Prefab** system allows you to create, configure, and store a **GameObject** complete with all its components, property values, and child GameObjects as a reusable Asset
- The Prefab Asset acts as a template from which you can create new Prefab instances in the **Scene**
- When you want to reuse a GameObject configured in a particular way – like a non-player character (NPC), prop or piece of scenery – in multiple places in your Scene, or across multiple Scenes in your Project, you should convert it to a Prefab
- This is better than simply copying and pasting the GameObject, because the Prefab system allows you to automatically keep all the copies in sync.

(<https://learn.unity.com/tutorial/prefabs-e#5f49e5bcedbc2a280d7cf66f>)

# Prefabs Examples

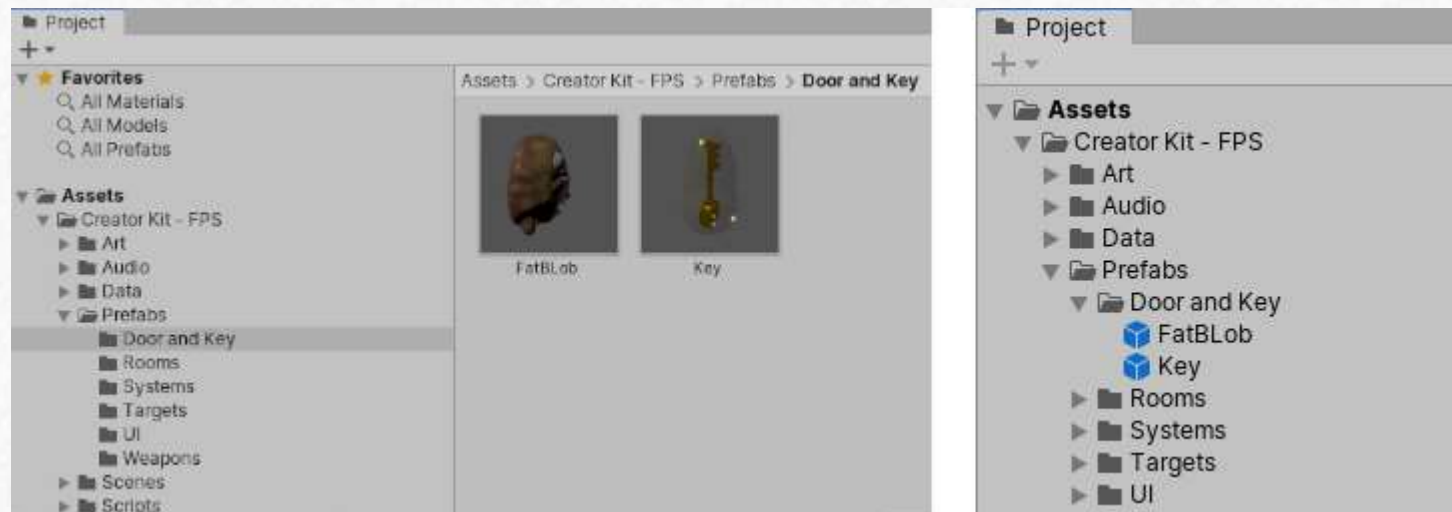
---

- Some common examples of Prefab use include:
  - Environmental Assets - for example a certain type of tree used multiple times around a level (as seen in the screenshot above).
  - Non-player characters (NPCs) - for example a certain type of robot may appear in your game multiple times, across multiple levels. They may differ (using *overrides*) in the speed they move, or the sound they make.
  - Projectiles - for example a pirate's cannon might instantiate a cannonball Prefab each time it is fired.
  - The player's main character - the player prefab might be placed at the starting point on each level (separate Scenes) of your game.



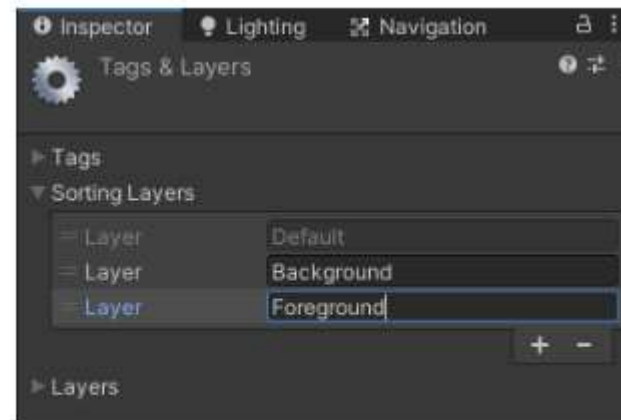
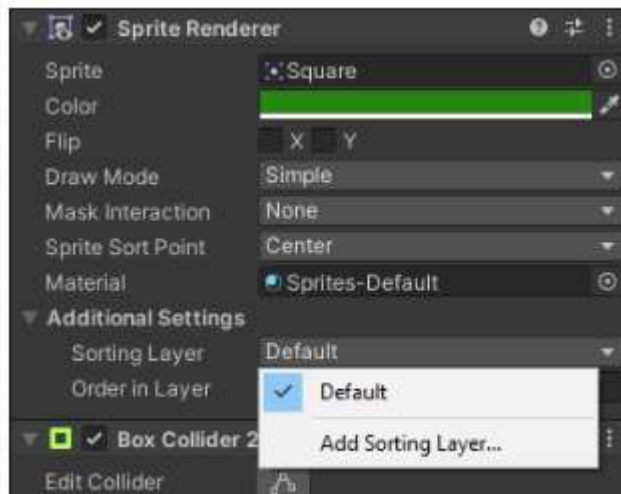
# Creating Prefab Assets

- To create a Prefab Asset, drag a **GameObject** from the Hierarchy window into the Project window
- The GameObject, and all its components and child GameObjects, becomes a new Asset in your Project window
- Prefabs Assets in the Project window are shown with a thumbnail view of the GameObject



# Unity – Sorting Layers

- Sorting Layers are used in conjunction with Sprites to define the overlay order of different Sprites.
- To change their order, drag the handle at the left-hand side of each Layer item.
- Change the value of the Order in Layer to set the Renderer's priority among other Renderers within the same Sorting Layer.





# Inputs

---

- Input allows the user to control your application using a device, touch, or gestures. You can program in-app elements, such as the graphic user interface (GUI) or a user **avatar**, to respond to user input in different ways.
- Unity supports input from many types of input devices, including:
  - Keyboards and mice
  - Joysticks
  - Controllers
  - Touch screens
  - Movement-sensing capabilities of mobile devices, such as accelerometers or gyroscopes
  - VR and **AR** controllers

# Input Manager

---

- The **Input Manager** window allows you to define input axes and their associated actions for your Project.
- To access it, from Unity's main menu, go to **Edit > Project Settings**, then select **Input Manager** from the navigation on the right.
- The **Input Manager** uses the following types of controls:
  - **Key** refers to any key on a physical keyboard, such as W, Shift, or the space bar.
  - **Button** refers to any button on a physical controller (for example, gamepads), such as the X button on an Xbox One controller.
  - A **virtual axis** (plural: **axes**) is mapped to a control, such as a button or a key

# Mapping virtual axes to controls

---

Key family	Naming convention
Letter keys	a, b, c...
Number keys	1, 2, 3...
Arrow keys	up, down, left, right
Numpad keys	[1], [2], [3], [+], [equals]...
Modifier keys	right shift, left shift, right ctrl, left ctrl, right alt, left alt, right cmd, left cmd
Special keys	backspace, tab, return, escape, space, delete, enter, insert, home, end, page up, page down
Function keys	f1, f2, f3...
Mouse buttons	mouse 0, mouse 1, mouse 2, and so on.

# Accessing the Controls

---

- You can also query input for a specific key or button with Input.
- GetKey and the naming conventions specified above. For example:

```
Input.GetKey("a");
```

```
float horizontalInput = Input.GetAxis ("Horizontal");
```

# Asset Workflow

---

- An asset is any item that you use in your Unity project to create your game or app. Assets can represent visual or audio elements in your project, such as 3D models, textures, **sprites**, sound effects, or music.
- Assets can also represent more abstract items such as color gradients, animation masks, or arbitrary text or numeric data for any use.
- An asset might come from a file created outside of Unity, such as a 3D Model, an audio file, or an image



# Importing Assets

- You can bring assets created outside of Unity into your Unity project. To do this, you can either export the file directly into the Assets folder for your project, or copy it into that folder
- For many common formats, you can save your source file directly into your project's Assets folder and Unity can read it
- Unity also detects when you save new changes to the file and re-imports files as necessary.

