



# *Unity Interface*





# Unity GameObjects

- The GameObject is the most important thing in the Unity Editor. Every object in your game is a GameObject. This means that everything thinks of to be in your game has to be a GameObject. However, a GameObject can't do anything on its own; you have to give it properties before it can become a character, an environment, or a special effect.
- A GameObject is a container; we have to add pieces to the GameObject container to make it into a character, a tree, a light, a sound, or whatever else you would like it to be. Each piece is called a component.
- Depending on what kind of object you wish to create, you add different combinations of components to a GameObject. You can compare a GameObject with an empty pan and components with different ingredients that make up your recipe of gameplay. Unity has many different in-built component types, and you can also make your own components using the Unity Scripting API.

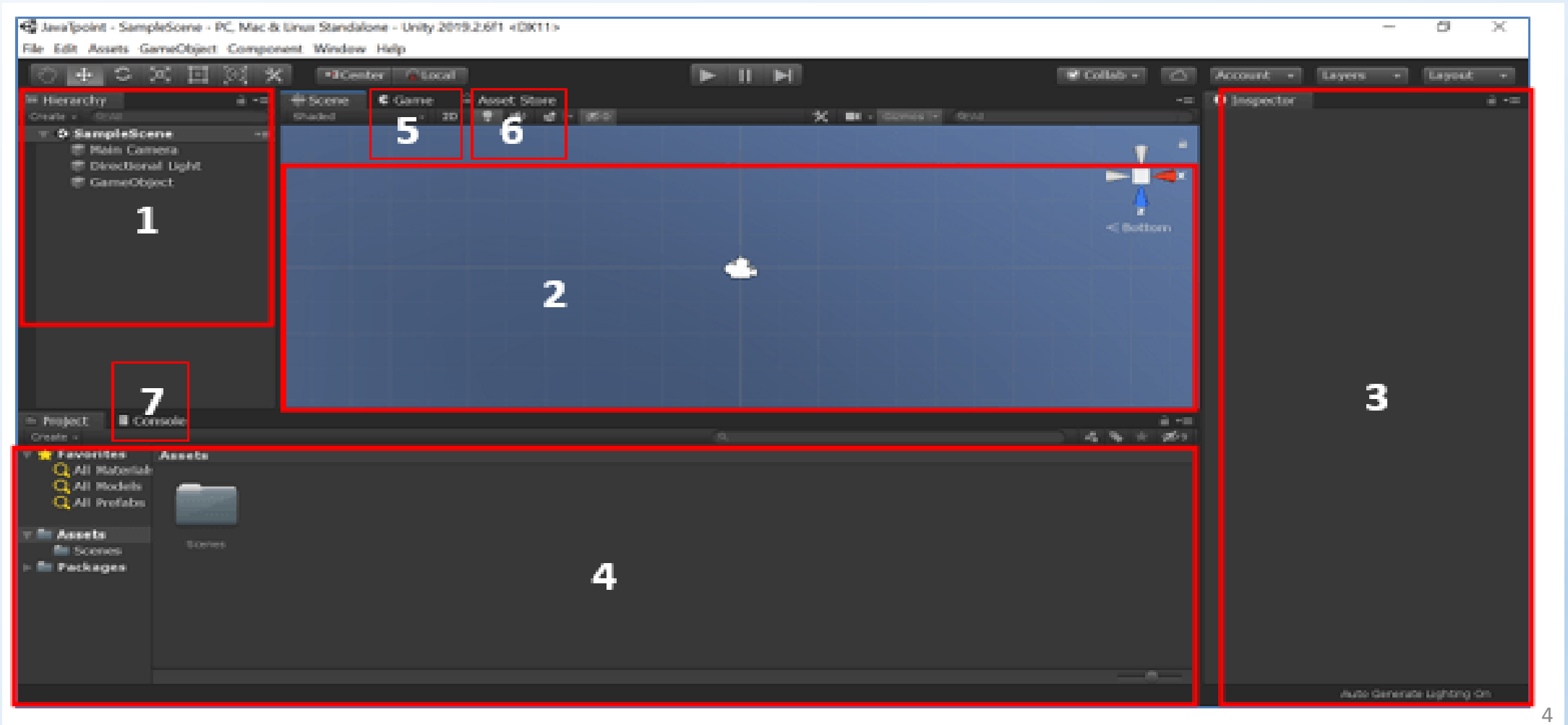


# Unity Components

- Unity is a component-based system. Unity components are functional pieces of every GameObject. If you don't understand the relationship between components and GameObjects, first read the GameObjects page before going any further.
- To give functionality to a GameObject, you attach different components to it. Even your scripts are components. So we can say, components are isolated functionality that can be attached to an object to give that functionality to that particular object. That means, when an object requires a specific type of functionality, you add the relevant component.
- A GameObject is like a container for many different components. By default, all GameObject automatically have a **Transform** component. This is because the Transform defines where the GameObject is located and how it is rotated and scaled. Without a Transform component, the GameObject would not have a location in the world.



# Interface





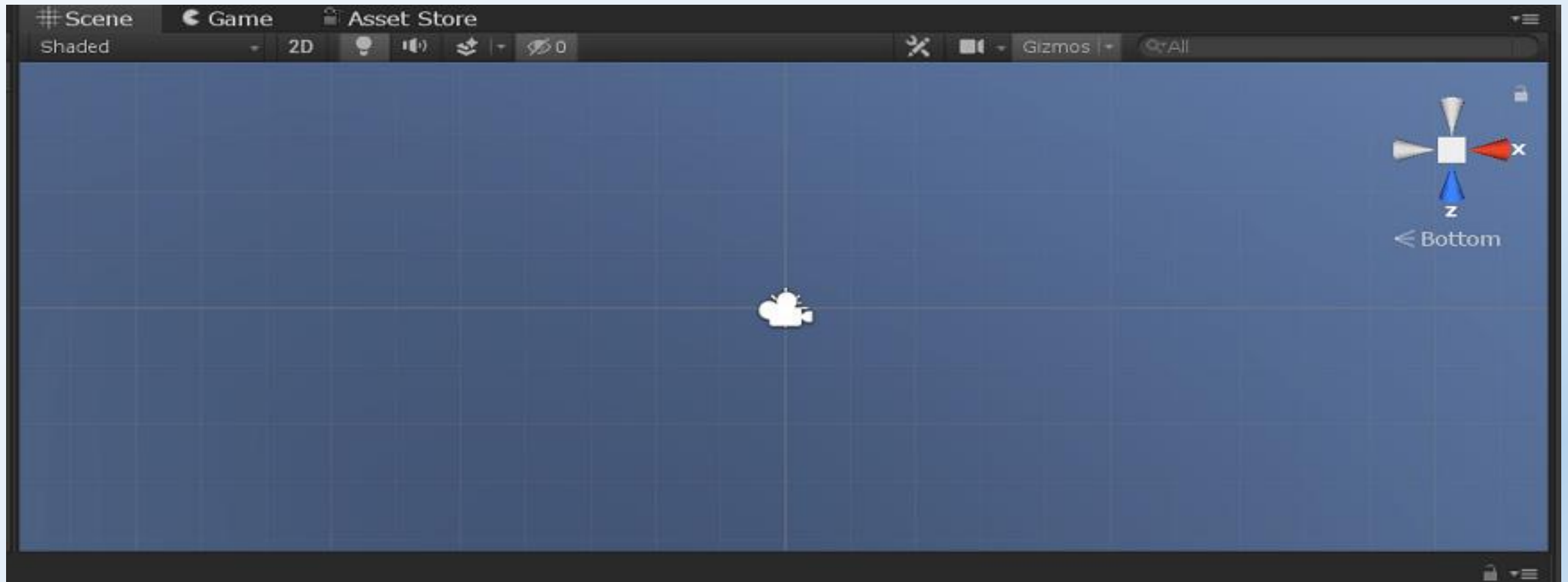
# Hierarchy



- This is the hierarchy window. This is the hierarchical text representation of every object in the scene. It is where all the objects in your recently open scene are listed, along with their parent-child hierarchy.
- Each item in the scene has an entry in the hierarchy, so the two windows are linked. The hierarchy defines the structure of how objects are attached to one another.
- By default, the Hierarchy window lists GameObjects by order of creation, with the most recently created GameObjects at the bottom. We can reorder the GameObjects by dragging them up or down, or by making the parent or child GameObjects.



# Scene





# Scene

- This window is where we will create our scenes. This view allows you to navigate and edit your scene visually.
- The scene view can show a 2D or 3D perspective, depending on the type of project you are working on.
- We are using the scene view to select and position scenery, cameras, characters, lights, and all other types of GameObject.
- Being able to select, manipulate, and modify objects in the scene view are some of the most important skills you must learn to begin working in Unity.





# Inspector

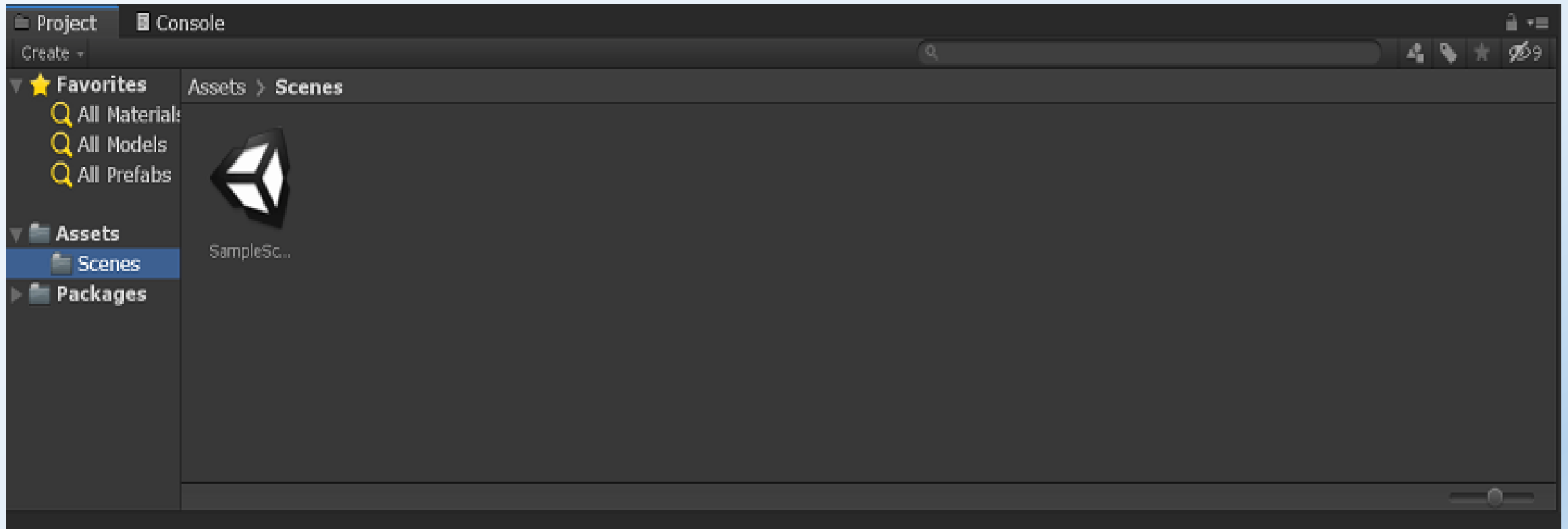


- The Inspector window allows you to view and edit all the properties of the currently selected object.
- Since different types of objects have different sets of properties, the layout and contents of the inspector window will vary.
- In this window, you can customize aspects of each element that is in the scene.
- You can select an object in the Hierarchy window or double click on an object in the scene window to show its attributes in the inspector panel.
- The inspector window displays detailed information about the currently selected GameObject, including all attached components and their properties, and allows you to modify the functionality of GameObjects in your scene.





# Project



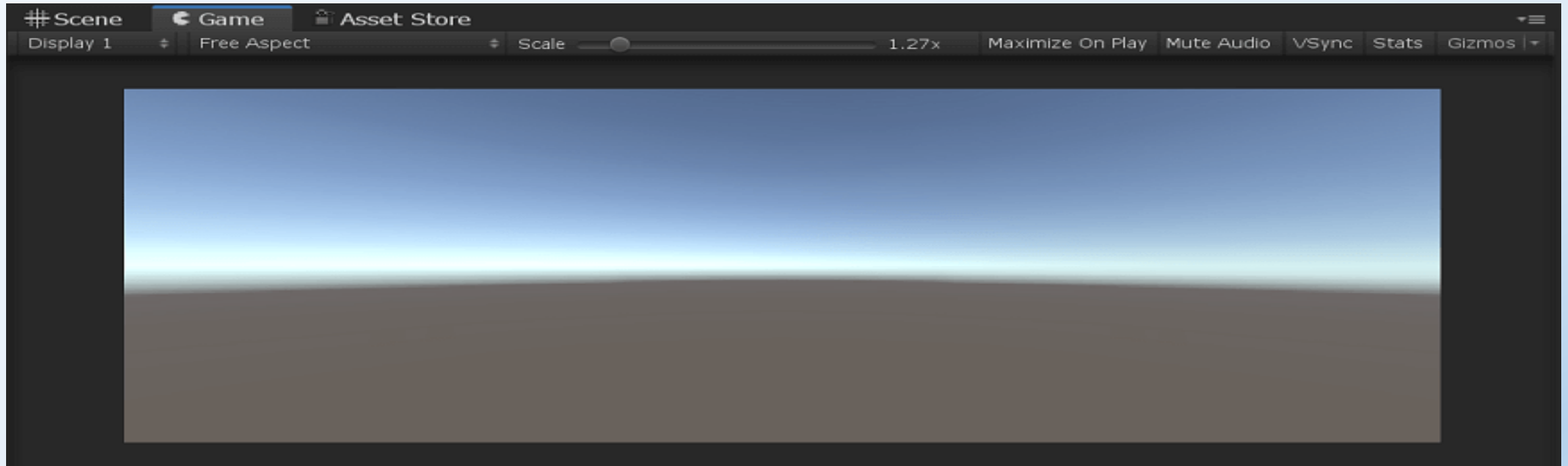


# Project

- This window displays the files being used for the game. You can create scripts, folders, etc. by clicking create under the project window.
- In this view, you can access and manage the assets that belong to your project.
- All assets in your project are stored and kept here. All external assets, such as textures, fonts, and sound files, are also kept here before they are used in a scene.
- The favorites section is available above the project structure list. Where you can maintain frequently used items for easy access. You can drag items from the list of project structure to the Favorites and also save search queries there.



# Game Window



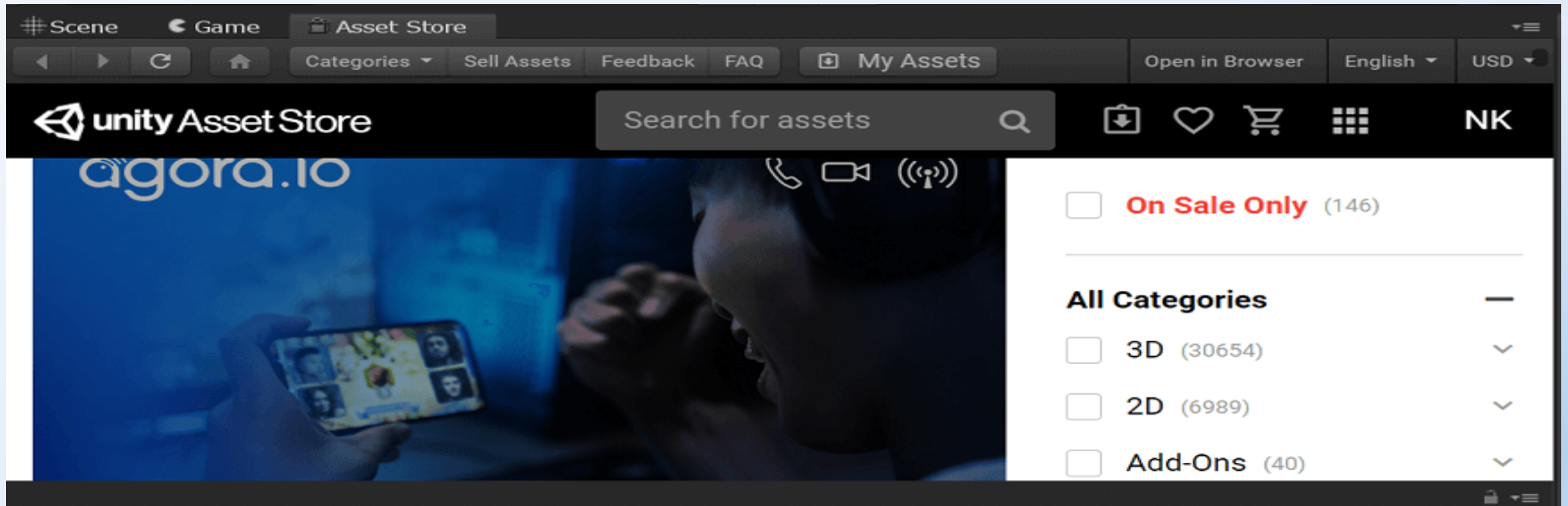


# Game Window

- This window shows the view that the main camera sees when the game is playing. Means here, you can see a preview window of how the game looks like to the player.
- It is representative of your final game. You will have to use one or more cameras to control what the player actually sees when they are playing your game.



# Asset Store



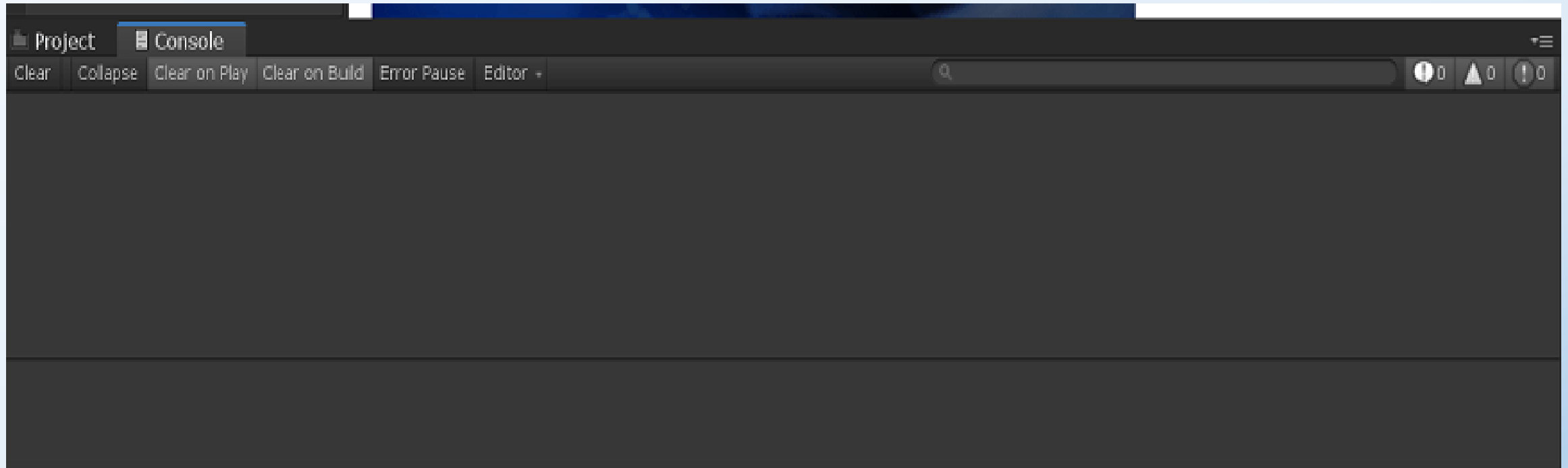


# Asset Store

- The Unity asset store is a growing library of free and commercial Assets created both by Unity Technologies and also members of the community.
- A wide variety of Assets is available, covering everything from Models, Textures, and animations to whole Project examples, tutorials, and Editor Extensions.
- The assets are accessed from a simple interface created into the Unity Editor and are downloaded and imported directly into your project.



# Console







# Console

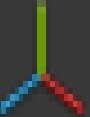



- If you are familiar with programming, you will already know that all the output messages, errors, warnings, and debug messages are shown here. It is similar for Unity, except output messages are done a bit differently than you think.
- The console window of Unity shows the errors, warnings, and other messages generated by Unity.
- You can also show your own messages in the console using the `Debug.Log`, `Debug.LogError`, `Debug.LogWarning` function



# Transforms and Object Parenting

- In Unity, the Transform component has three visible properties - the position, rotation, and scale. Each of these properties has three values for the three axes. Means, Transform is used to determine the Position, Rotation, and Scale of each object in the scene. Every GameObject has a Transform.



▼  **Transform**   

Position	X	0.131	Y	0.06	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1.104	Y	1	Z	1



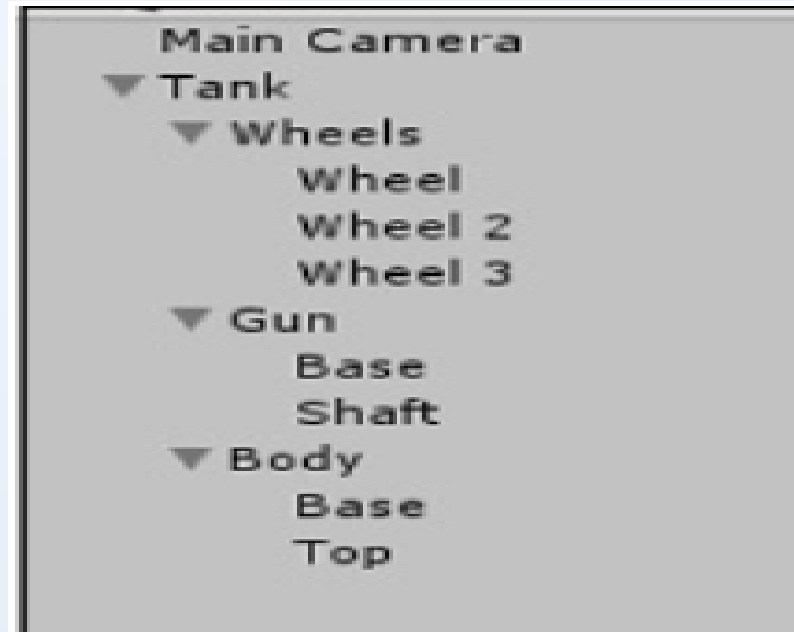
# Properties

- **Position:** This is the position of the transform in X, Y, and Z coordinates. 2D games generally do not focus on the Z-axis when it comes to positioning. The most frequent use of the Z-axis in 2D games is in the creation of parallax.
- **Rotation:** This property defines the amount of rotation (measured in degree) an object is rotated about that axis with respect to the game world or the parent object.
- **Scale:** The scale of the object defines how large it is when compared to its original or native size. For example, let us take a square of 2x2 dimensions. If the square is scaled against the X-axis by 3 and the Y-axis by 2 we will get a square of size 6x4.
- These properties are measured relative to the transform's parent. If the transform has no parent, the properties are calculated in world space.



# Object Parenting

- In Unity, GameObjects follow a Hierarchy system. Using this hierarchy system, GameObjects can become parents of other GameObjects. When a GameObject has a parent, it will perform all its transform changes with respect to another GameObject instead of the game world.
- A parent object causes all children objects to move and rotate the same way the parent object does, although moving children objects does not have any effect on the parent. Children themselves can be parents; e.g., your hand is the child of your arm, and fingers are children of your hand.





# Internal Assets

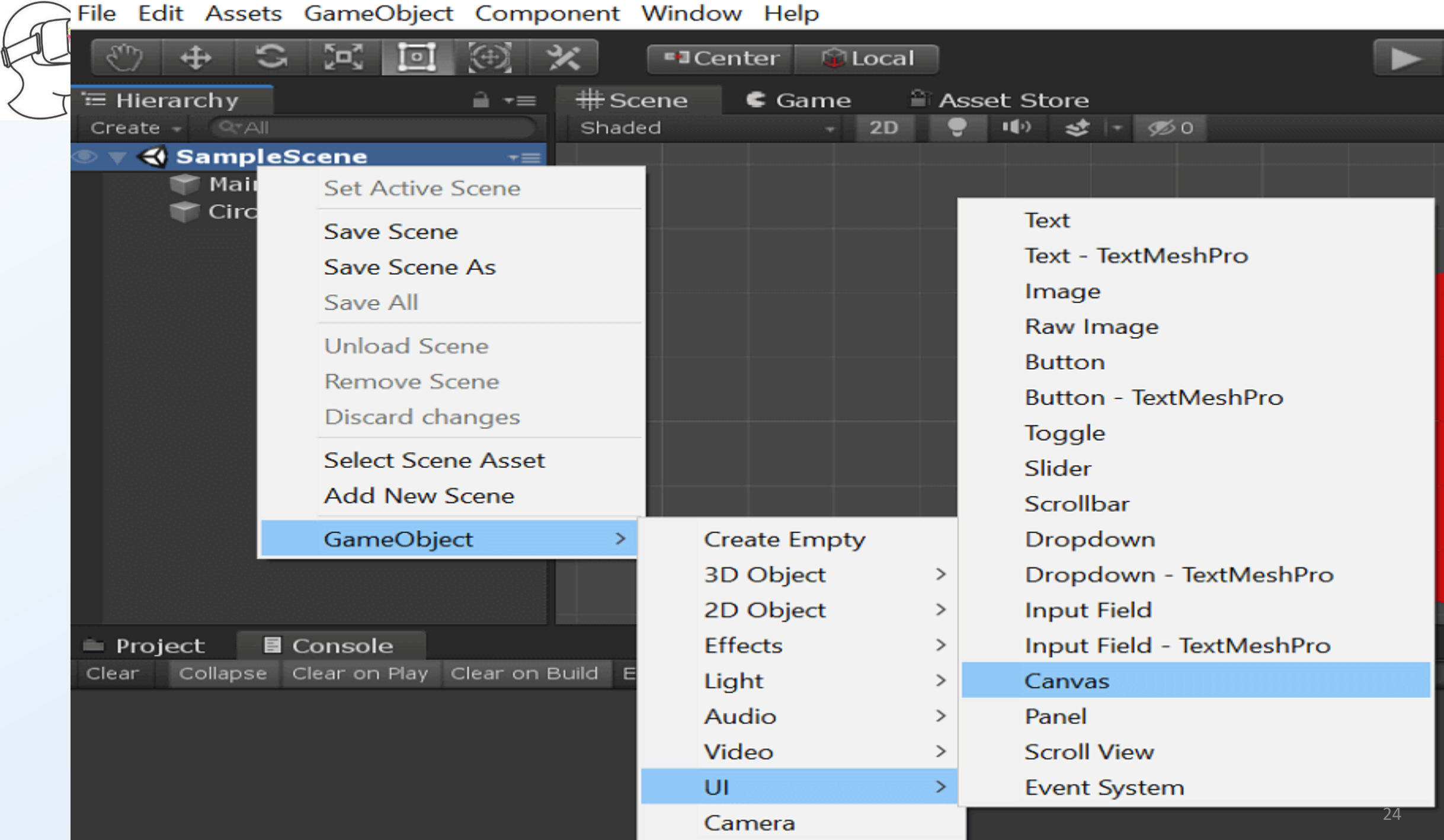
- Along with the external assets that you import from other programs such as image, audio files, 3D models, etc., Unity also offers the creation of internal assets. These assets are formed within Unity itself, and as such, do not need any external program to create or modify.
- Let's see some examples of internal assets:
- **Scenes:** These act as levels.
- **Animations:** These contain data for the animation of GameObject.
- **Materials:** These are used to define how lighting affects the appearance of an object.
- **Scripts:** The code which will be written for the GameObjects.
- **Prefabs:** These act as a blueprint for GameObjects so they can be generated at runtime.





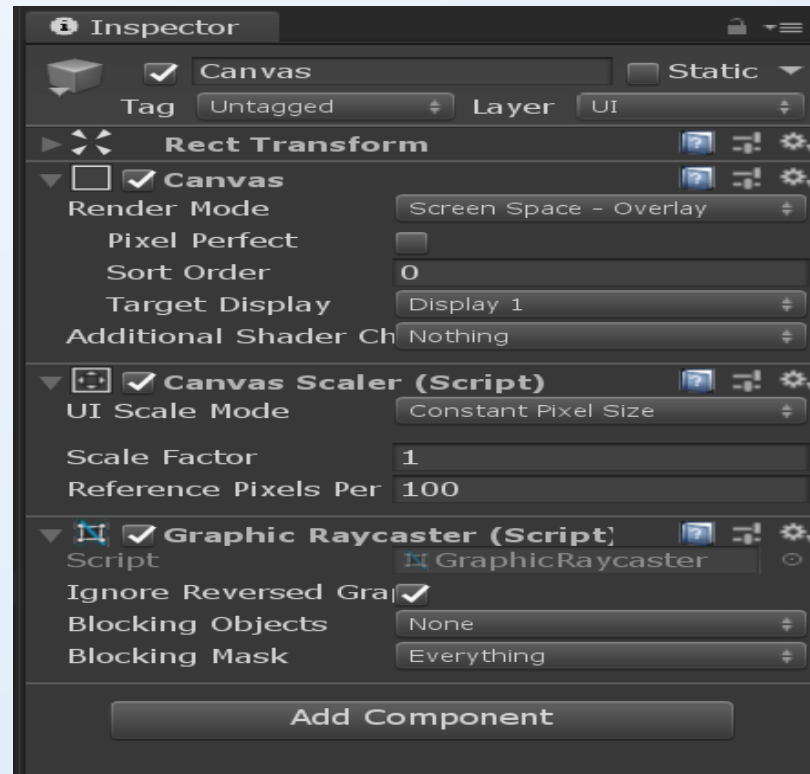
# Unity UI

- Unity UI (User Interface) is used to create a user interface in your game or application.
- UI Canvas
- The UI Canvas is like a drawing sheet for UI elements on the scene, where all UI elements will render. When you create a UI element without an existing canvas will automatically generate one.
- UI Canvas acts as the master of all UI elements on the screen. Because of that, all UI elements are required to be the child game object of the canvas game object.
- To add the canvas in your scene right, click on the Scene name or Main Camera from the Hierarchy tab and select GameObject -> UI -> Canvas.





# Canvas Components





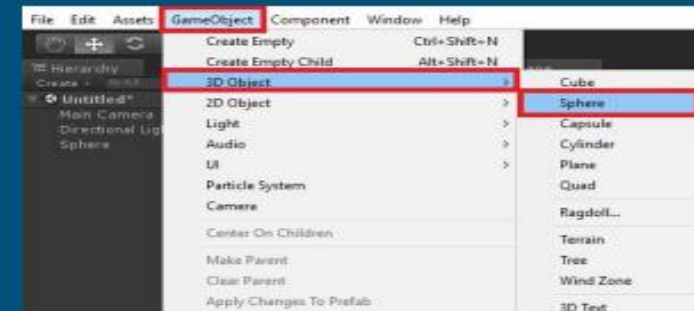
# Scripting



# Scripting

## Unity Scripting: What is a Script?

- ... but what is a script in Unity?
- Scripts are really just **custom components**!
- When you create a Script, you're creating your very own component.
  - You can give that component behaviour, properties, fields, and values.
- You add scripts to GameObjects just like any other component!
- First, let's make a GameObject to add the script to.

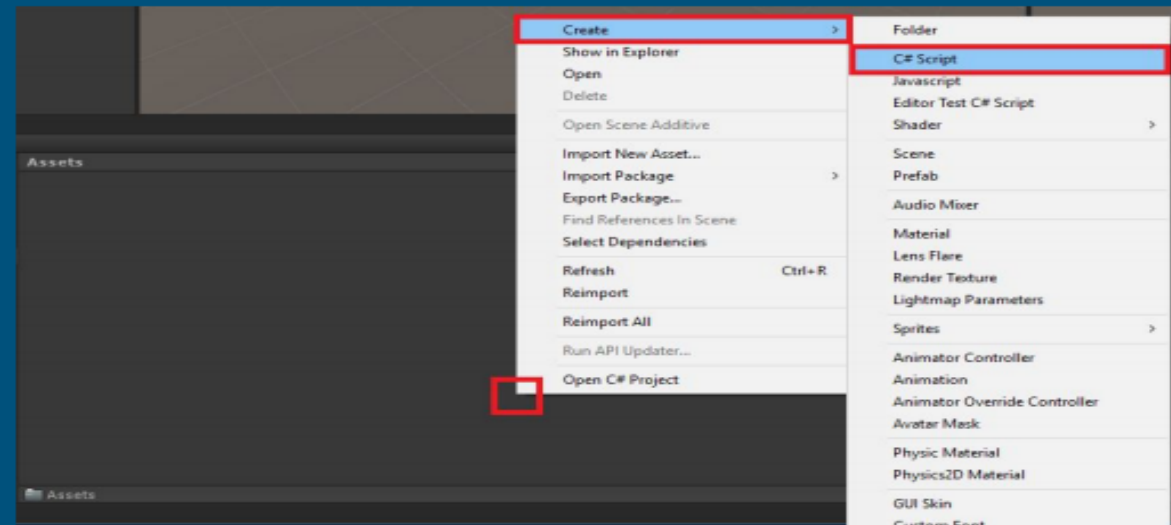




# Unity Scripting: Our First Script

- Now let's create a new C# script in Unity

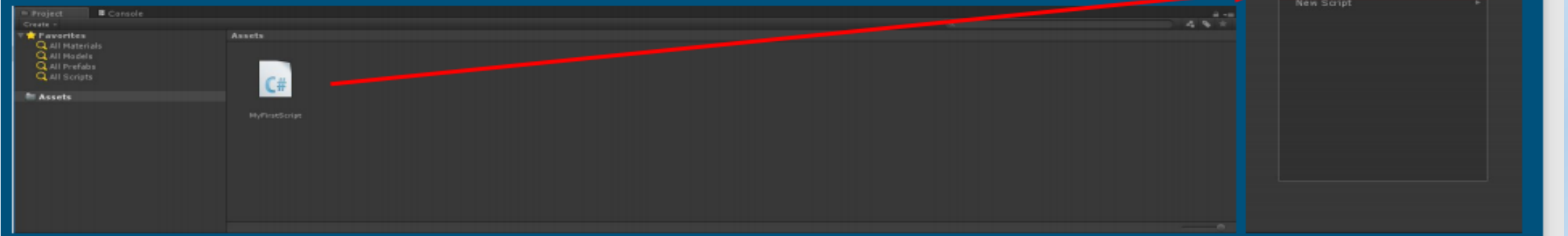
1. Right Click in "Assets" folder
  - a. You can also use "Assets" menu
2. Hover over "Create"
3. Click "C# Script"
4. Give it a name!





# Unity Scripting: Adding a Script

- Select the object you want to add the script to.
  - In this case, it's our sphere.
- Click “Add Component”
- Add your very own script!
- You can also just drag the script onto the object.







# Unity Scripting: Opening The Script

---

- We're now ready to dive into our new script!
- Go ahead and open your C# script.
  - If you're on Windows, this should open in Visual Studio.
  - If you're on Mac, it will open in MonoDevelop
  - Both of these are fine, they're just IDE's (Integrated Development Environments) for coding.
- You'll first notice a few things...
  - "MonoBehaviour"
  - "Start()"
  - "Update()"
- A MonoBehaviour is a Unity-specific class that every script **derives**.
- MonoBehaviour scripts are especially useful for game development.



# Unity Scripting: MonoBehaviour

- Start() and Update() are just **methods**.
- **Start()**: Runs once when the game begins.
  - Use to initialize script
- **Update()**: Runs **every frame**.
  - A game is divided into "frames".
    - Think of old-school flipbooks, each page is a "frame"!
  - This method will be called at least 90 times every second.
- Some others:
  - Awake(): Runs before start.
  - OnEnable(): Runs when the script is enabled.
  - FixedUpdate(): Framerate-independent update, for physics.

```
public class MyFirstScript : MonoBehaviour {  
  
    // Runs before start  
    void Awake() {  
  
    }  
  
    // Use this for initialization  
    void Start () {  
  
    }  
  
    // Runs when the script is enabled  
    void OnEnable() {  
  
    }  
  
    // Update is called once per frame  
    void Update () {  
  
    }  
  
    // Used for physics calculations  
    void FixedUpdate() {  
  
    }  
  
}
```