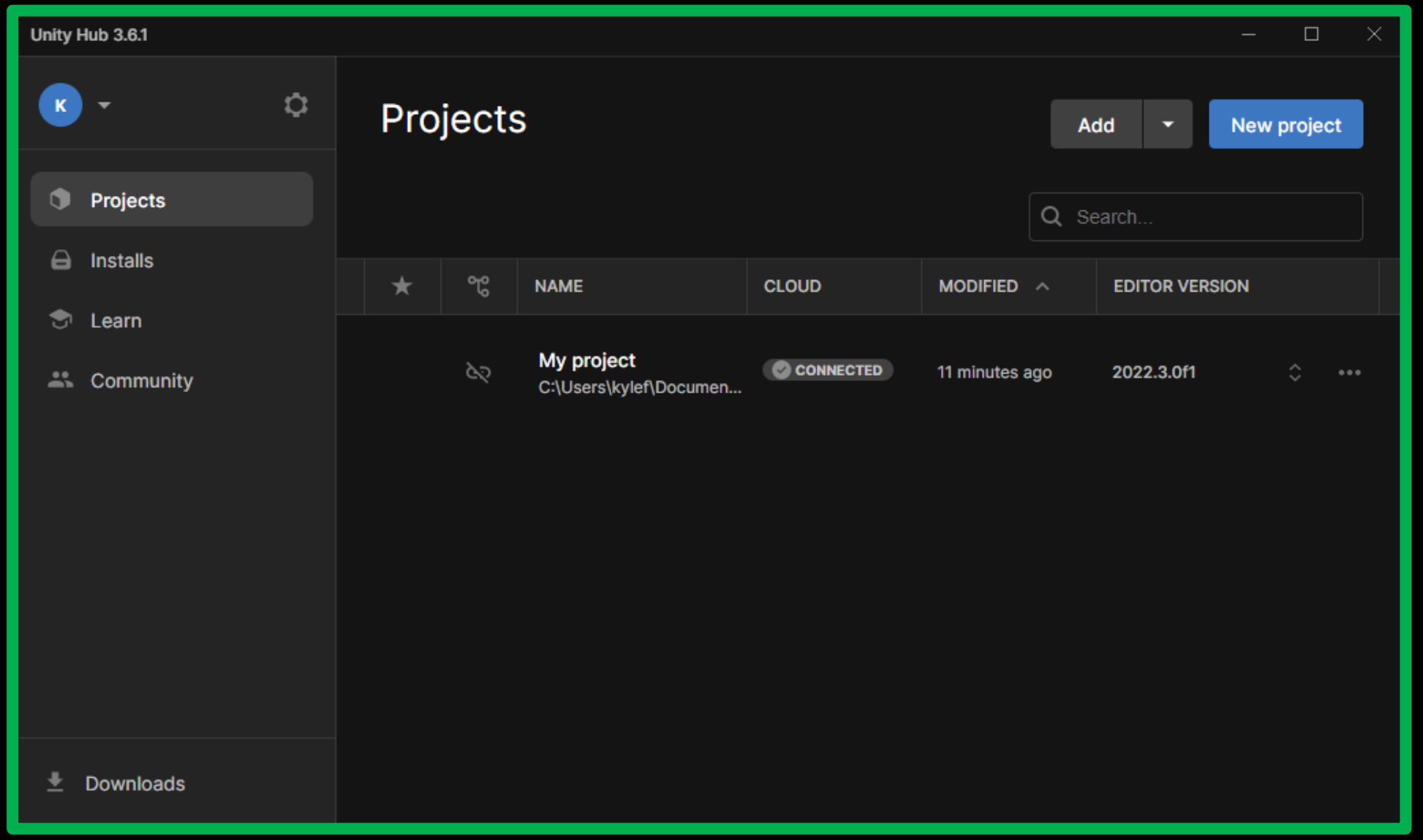
Tutorial Document by Kyle Furey

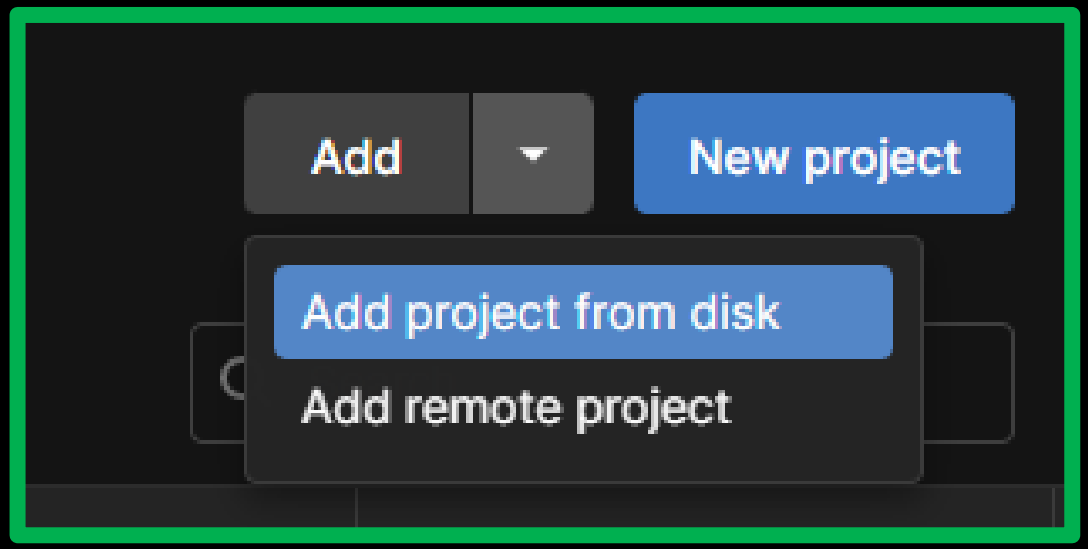
Made for educational purposes.

Table of Contents

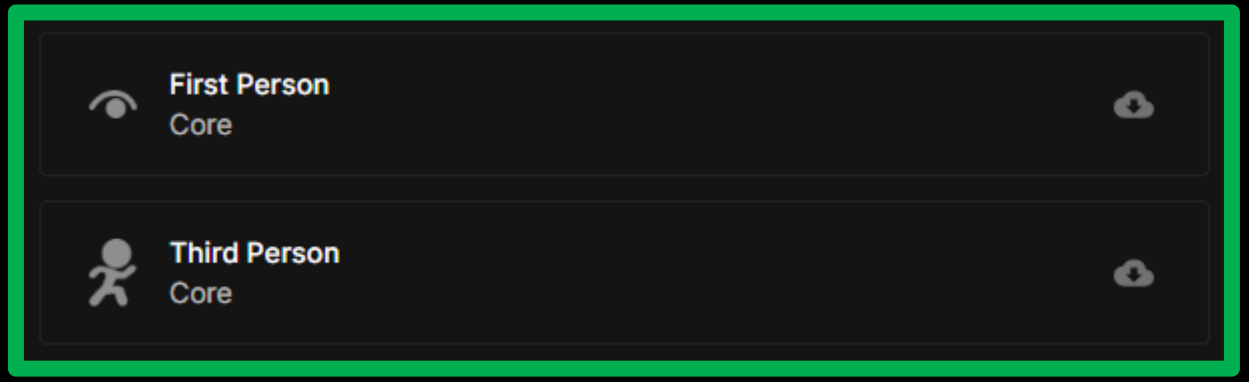
Slide 3 - 5 – Unity Hub, Creating and Opening Projects
Slide 6 - 10 – Navigating Unity Editor
Slide 11 - 14 – Unity Editor Tools
Slide 15 - 20 – Inspector and Component Types
Slide 21 - 24 – Assets and Prefabs
Slide 25 - 30 – Visual Scripting

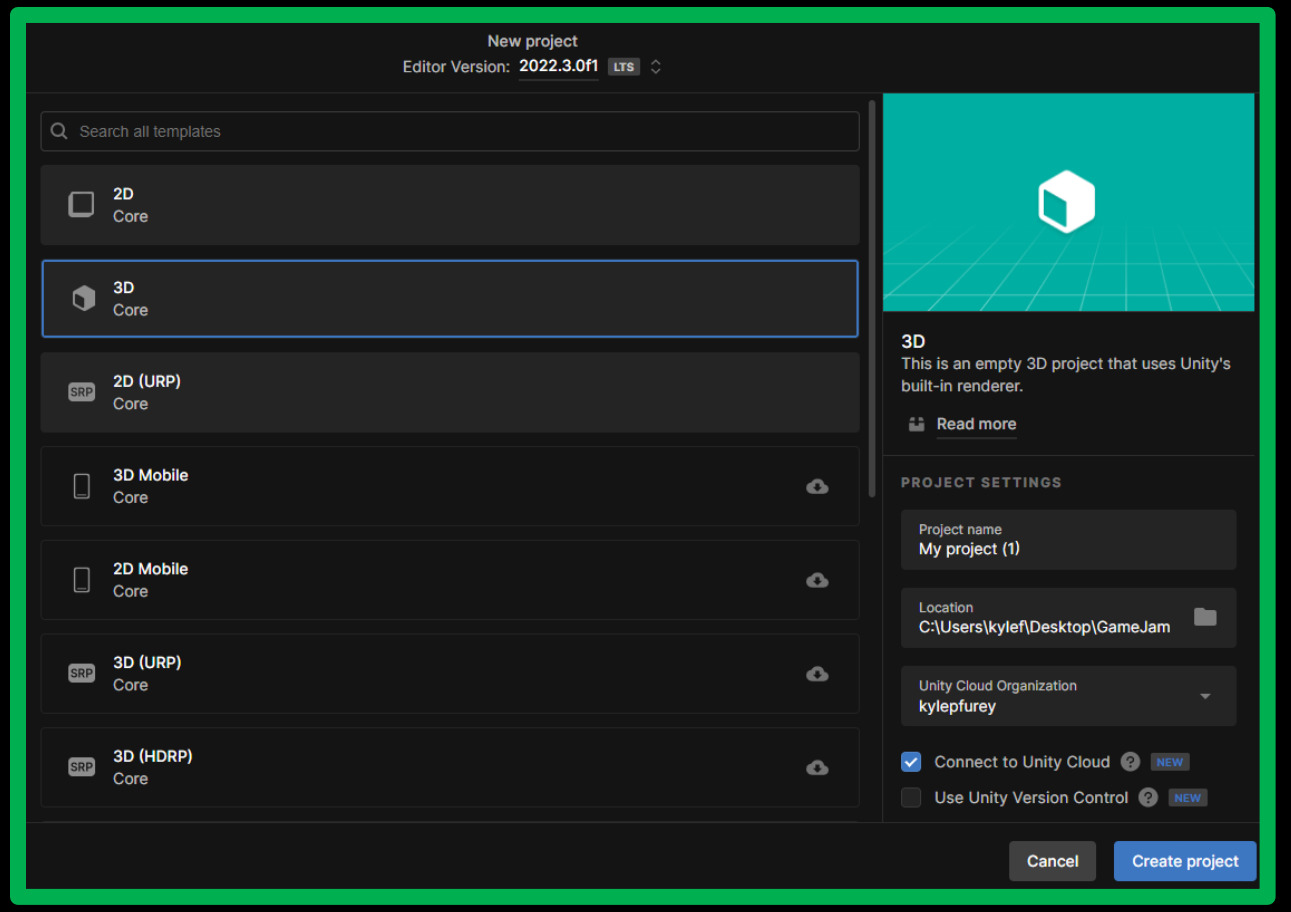


This is the Unity Hub. Your projects will show up here.

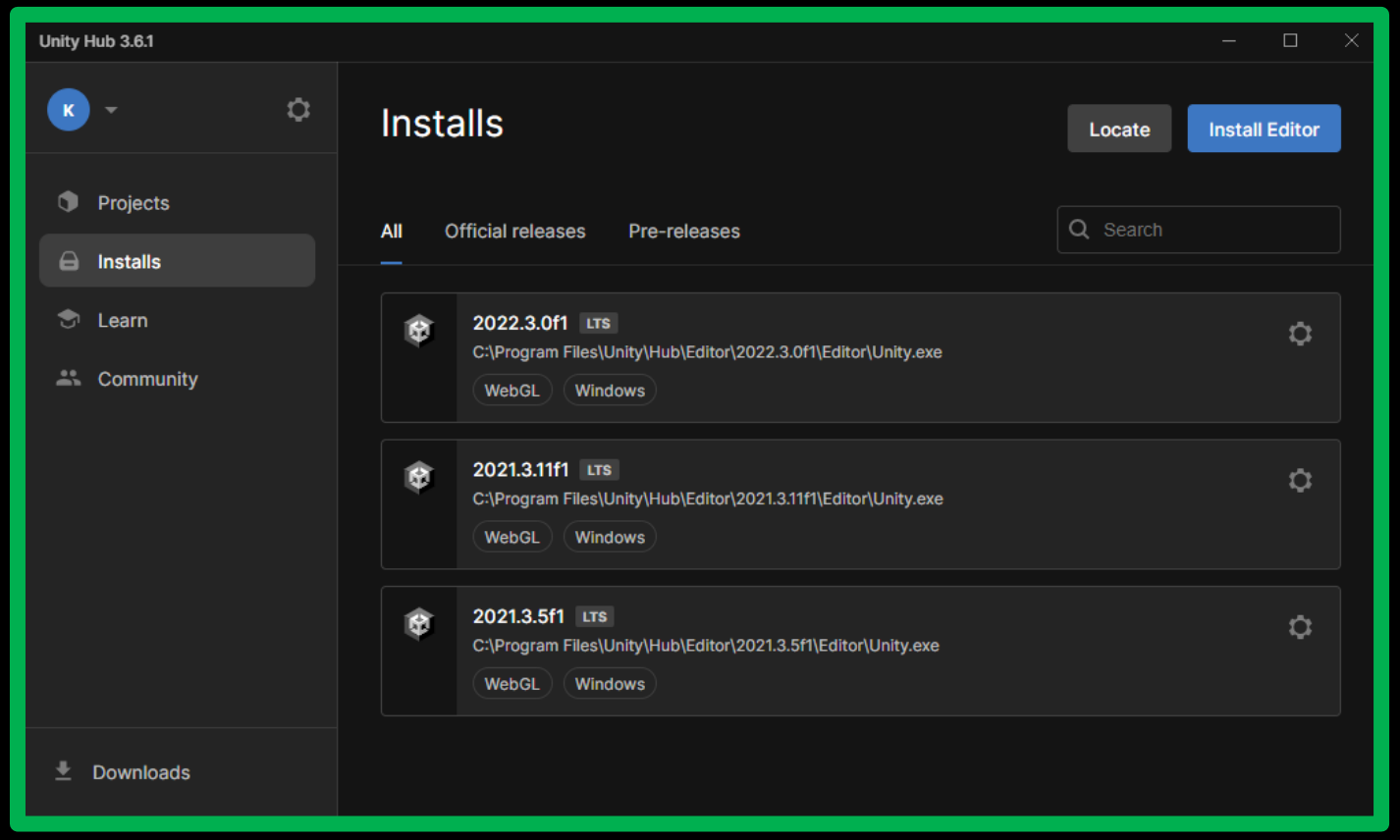


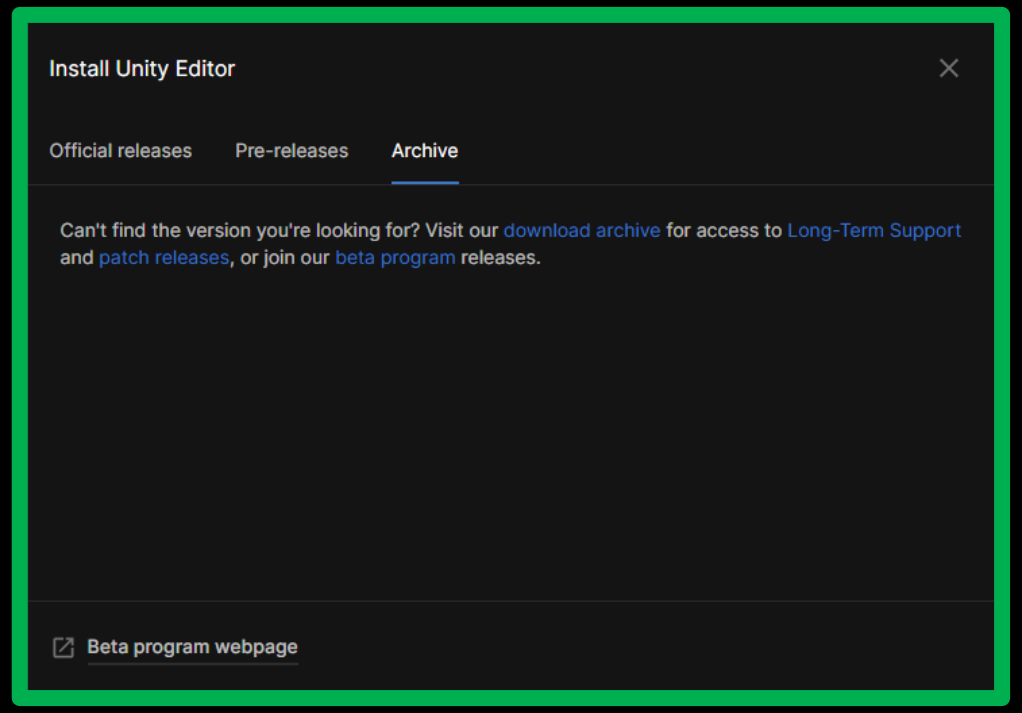
Use **Add project from disk** to find a project if you have one already on your computer.





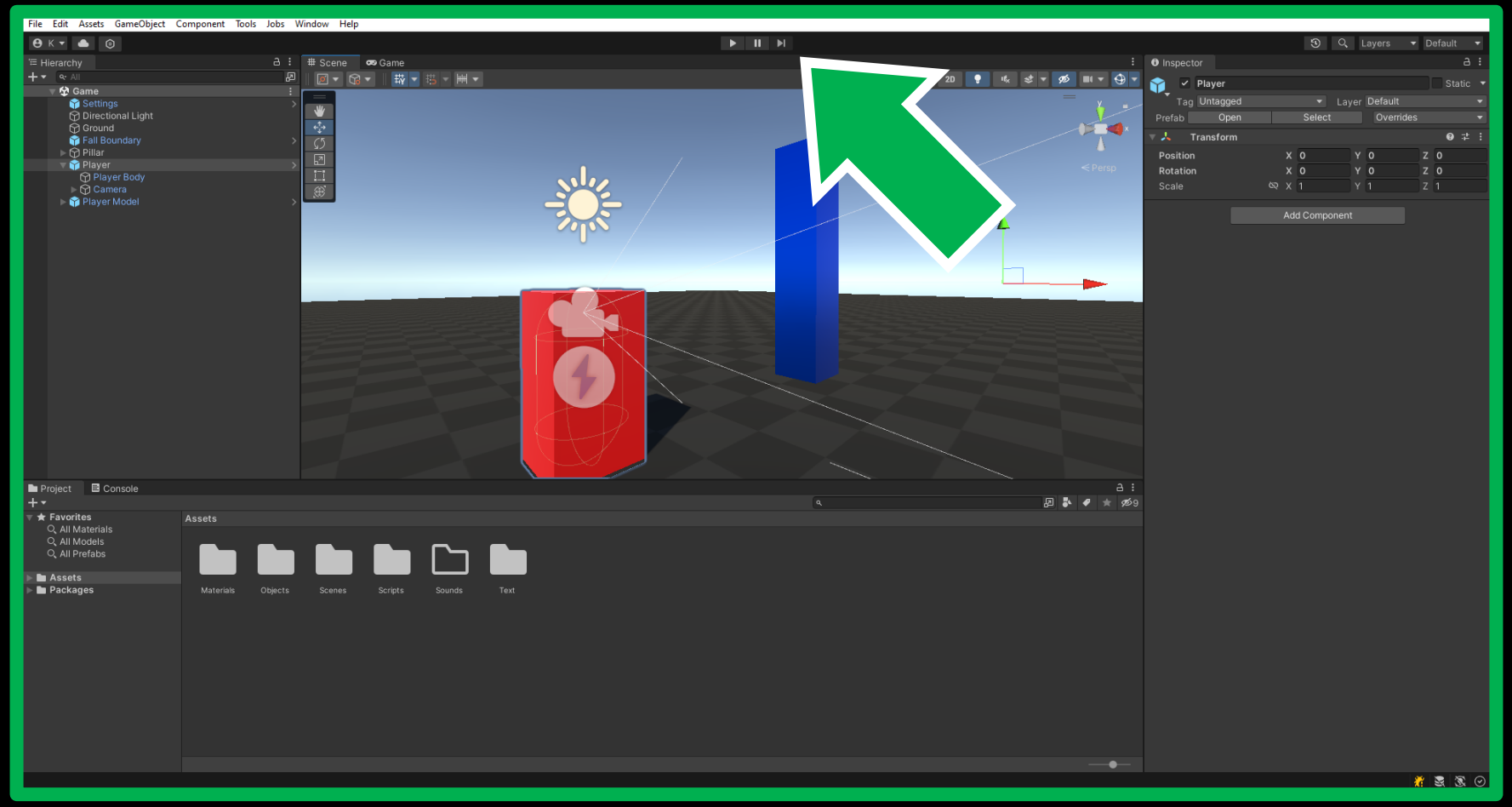
If not, click **New project** to make a new project. You can pick from different Cores (which is bare bones 3D or 2D), or from select templates. Some templates are only on specific Editor Versions.



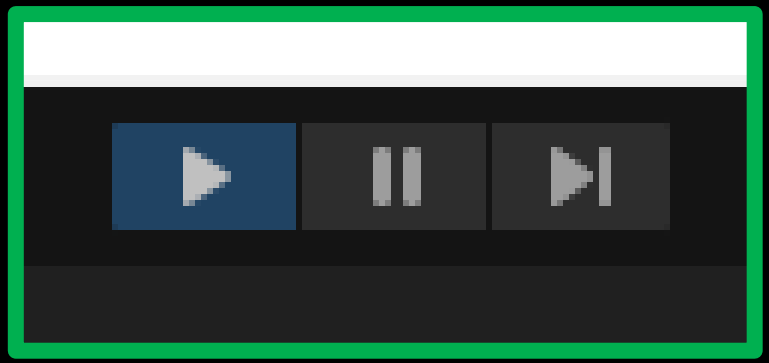


If you do not have the right version or are looking for a specific version of Unity, click **Installs** and then **Install Editor**.

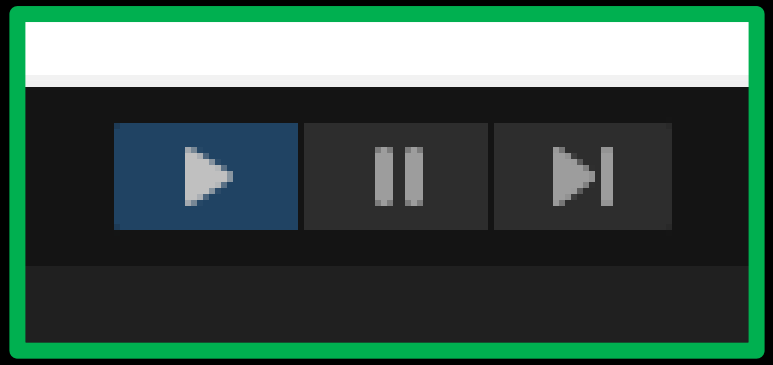
Click **Archive** and then **download archive** to bring up the website if you do not see your desired version. LTS versions of Unity generally have better performance.

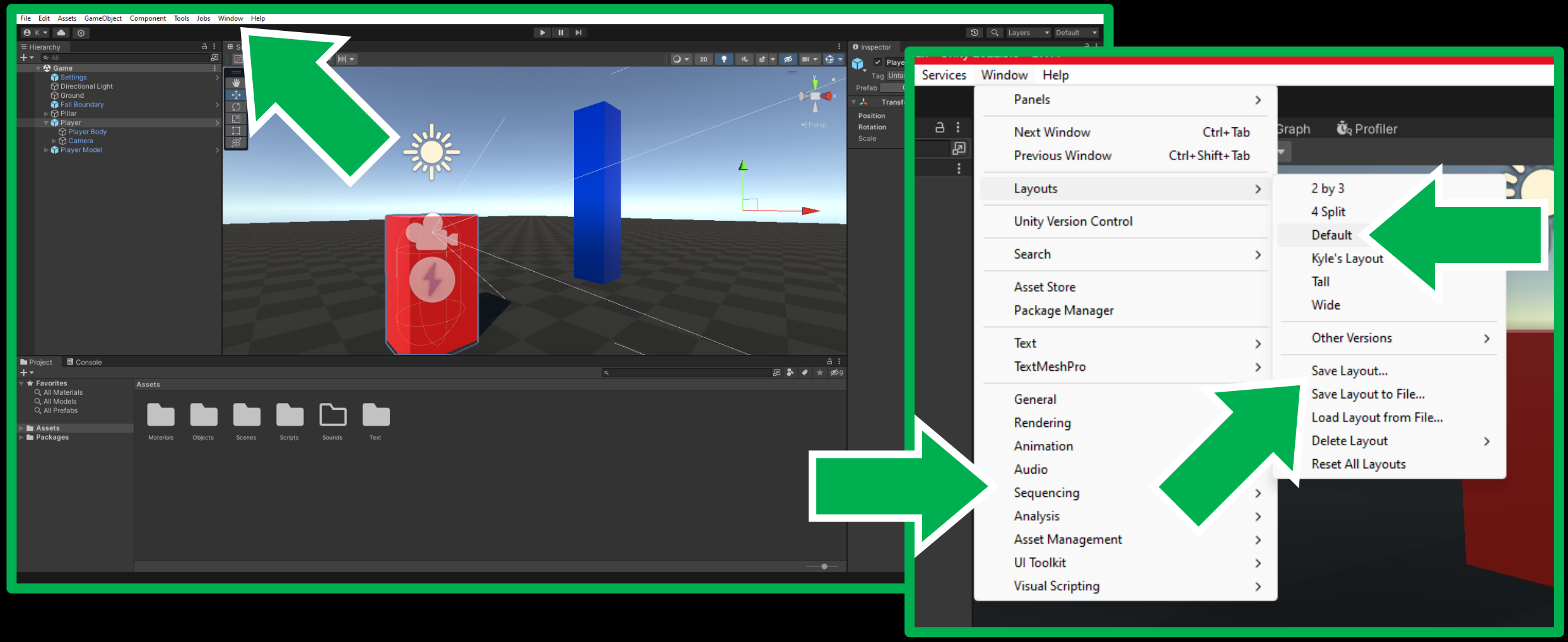


This is the Unity Editor. This is where you will be editing your game. You can click the arrow at the top of the screen to play your game.

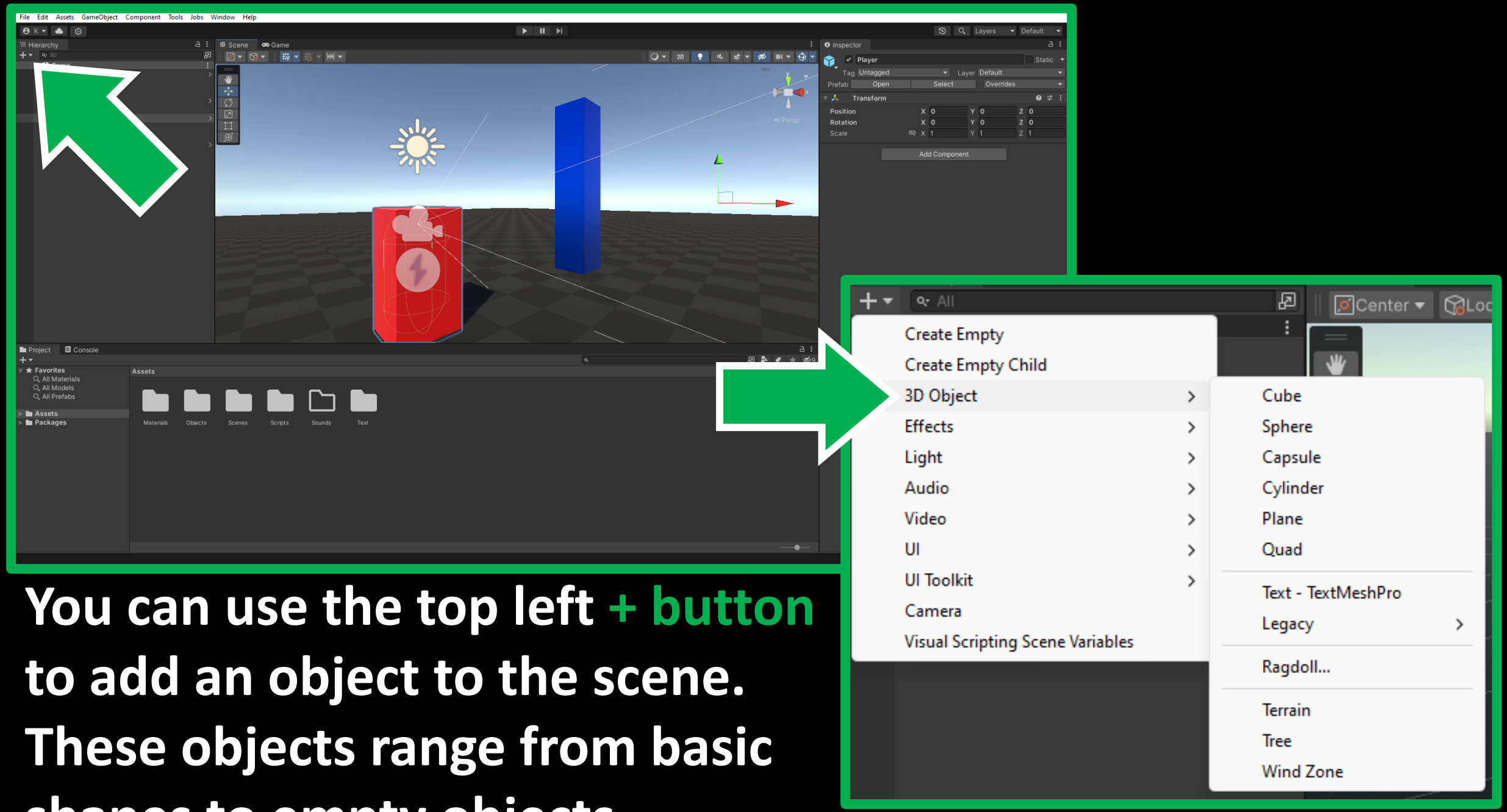


Warning! While any of these symbols at the top of your screen are blue, all changes will not be saved as the game is in Play Mode.

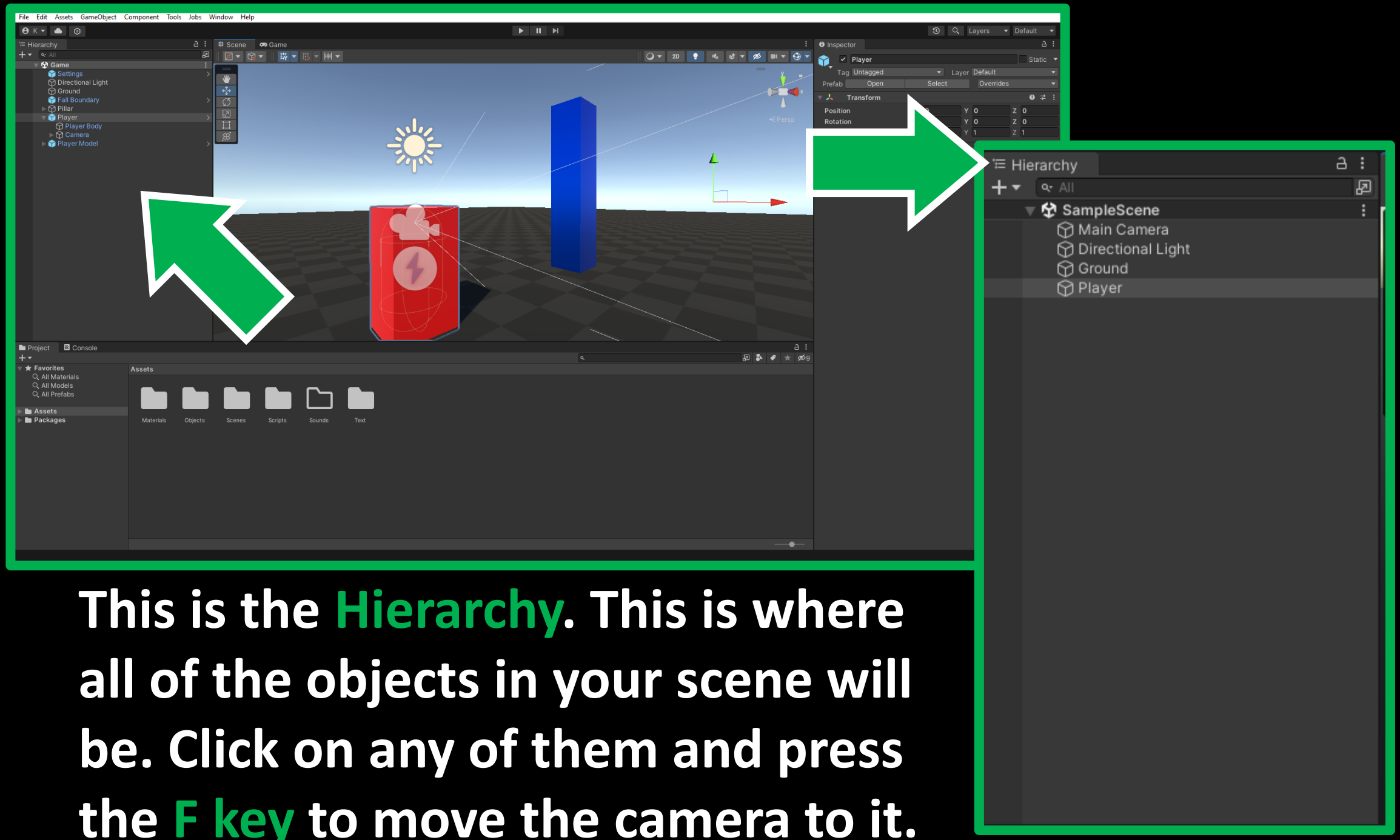




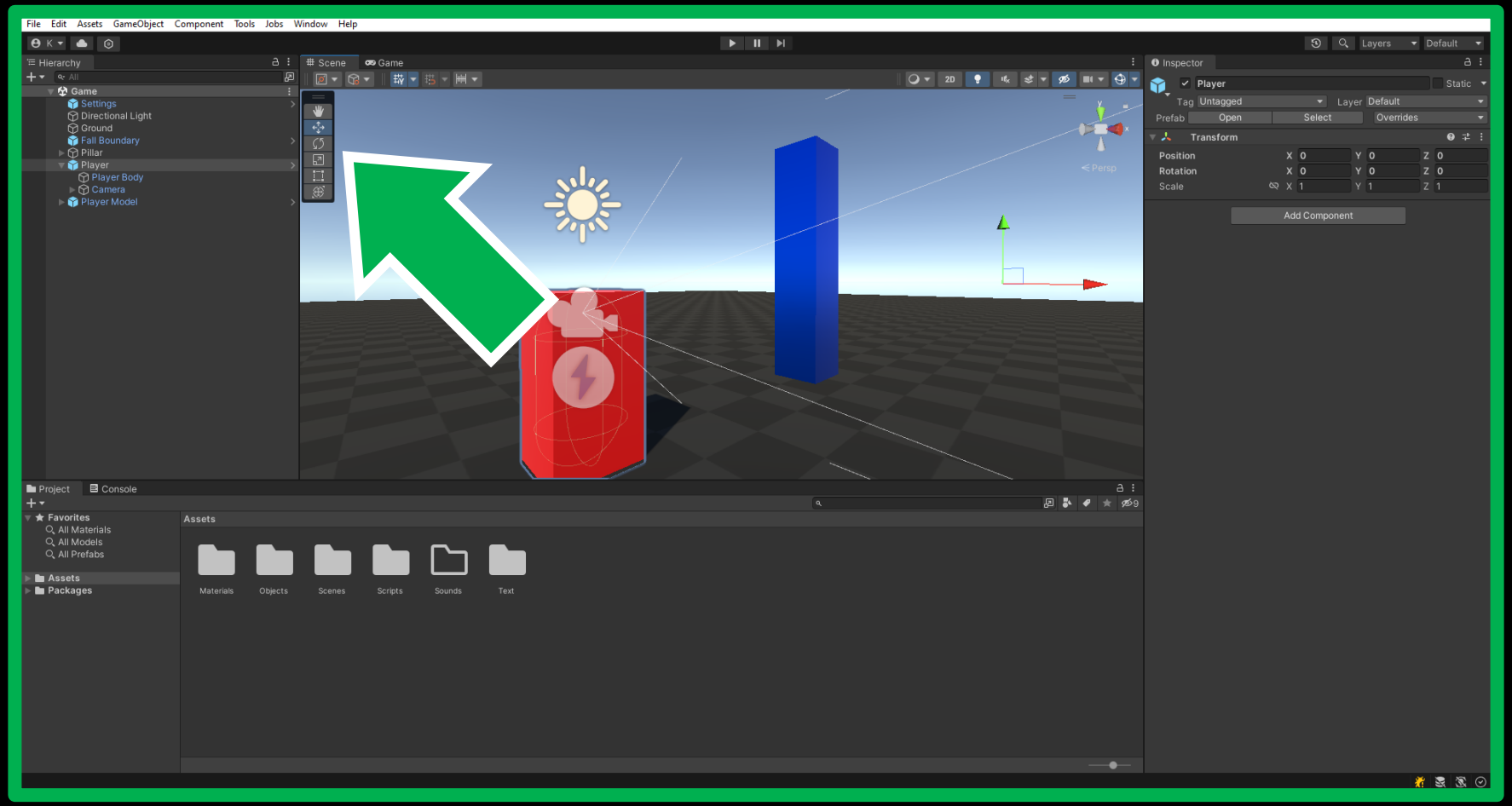
If your layout is reset, or if you want to save your own layout, click **Window**, **Layouts**, **Default** or **Save Layout**. You can also find other panels in the list under General.

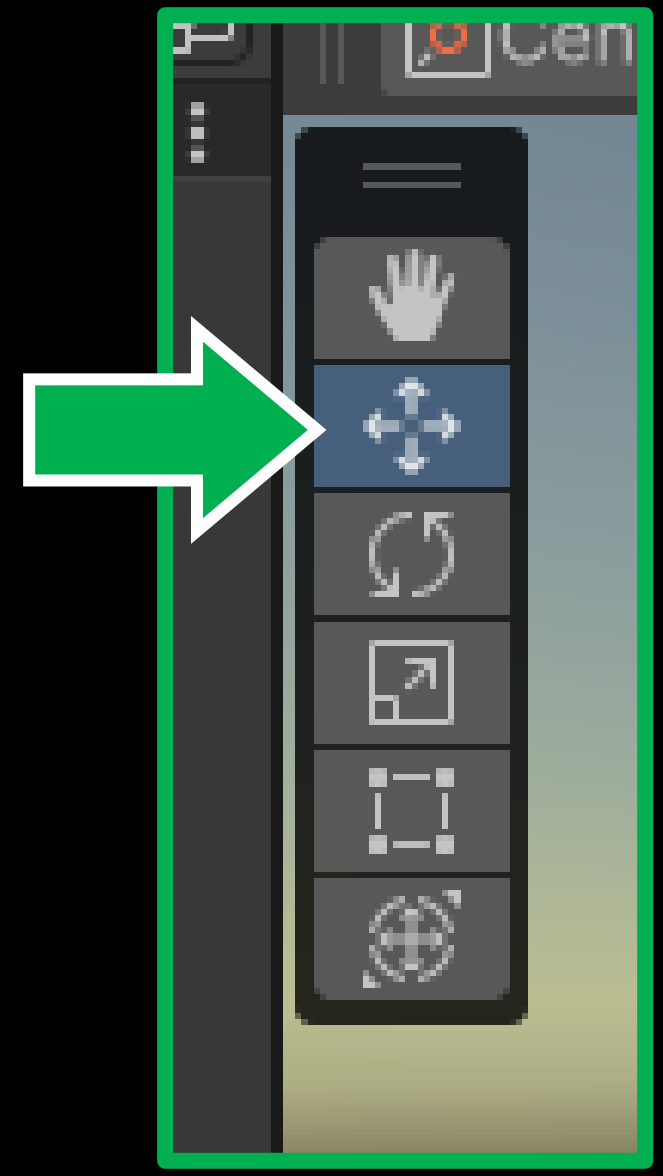


You can use the top left + button to add an object to the scene. These objects range from basic shapes to empty objects.

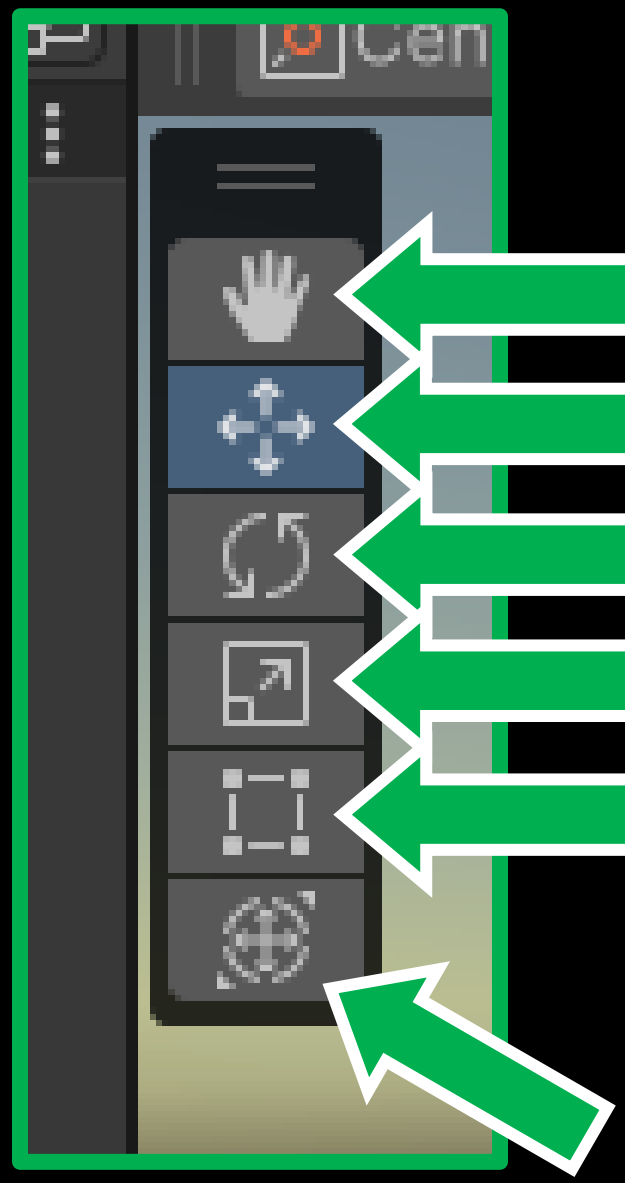


This is the Hierarchy. This is where all of the objects in your scene will be. Click on any of them and press the F key to move the camera to it.





There are several ways to adjust an object in the scene. Click these icons to select a different tool.



Tip! Hold the Control key to snap to grid!

Hand Tool – Moves the editor camera.

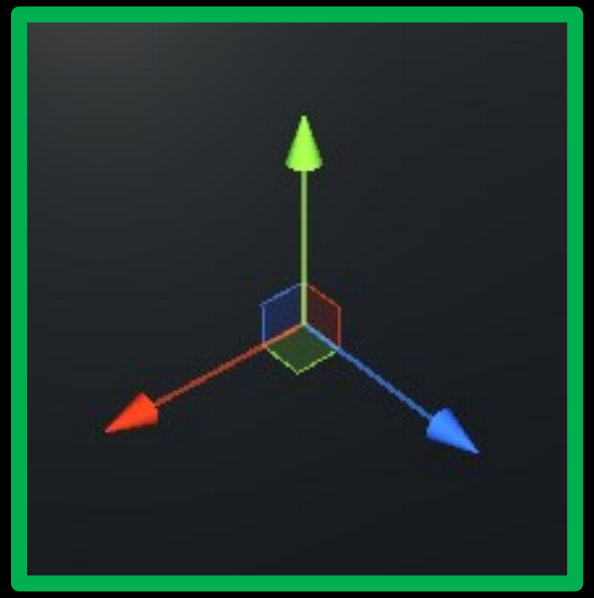
Move Tool – Moves objects in 3D.

Rotate Tool – Rotates objects in 3D.

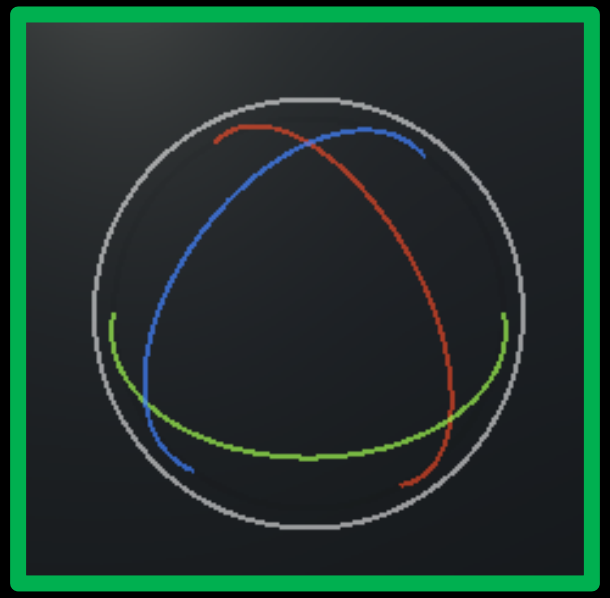
Scale Tool – Scales the size of objects in 3D.

Rectangle Tool – Scales the size of objects seamlessly from an edge.

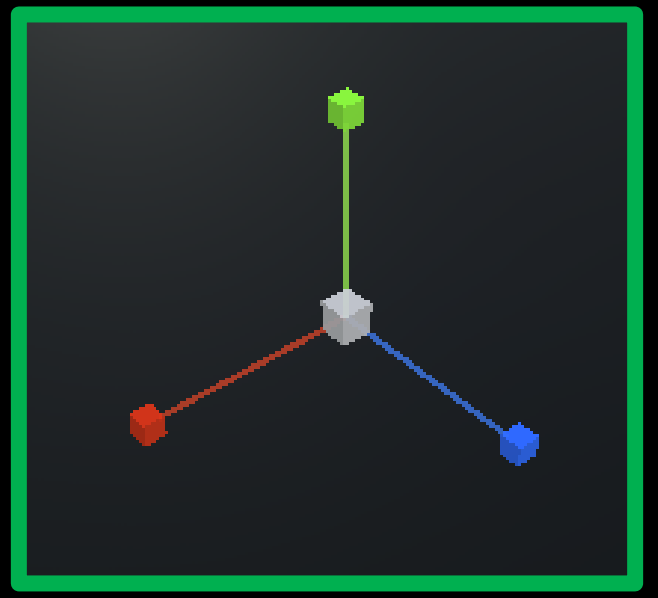
Transform Tool – Move, rotate, and scale objects in one tool.



Move Tool



Rotate Tool



Scale Tool

Rectangle Tool

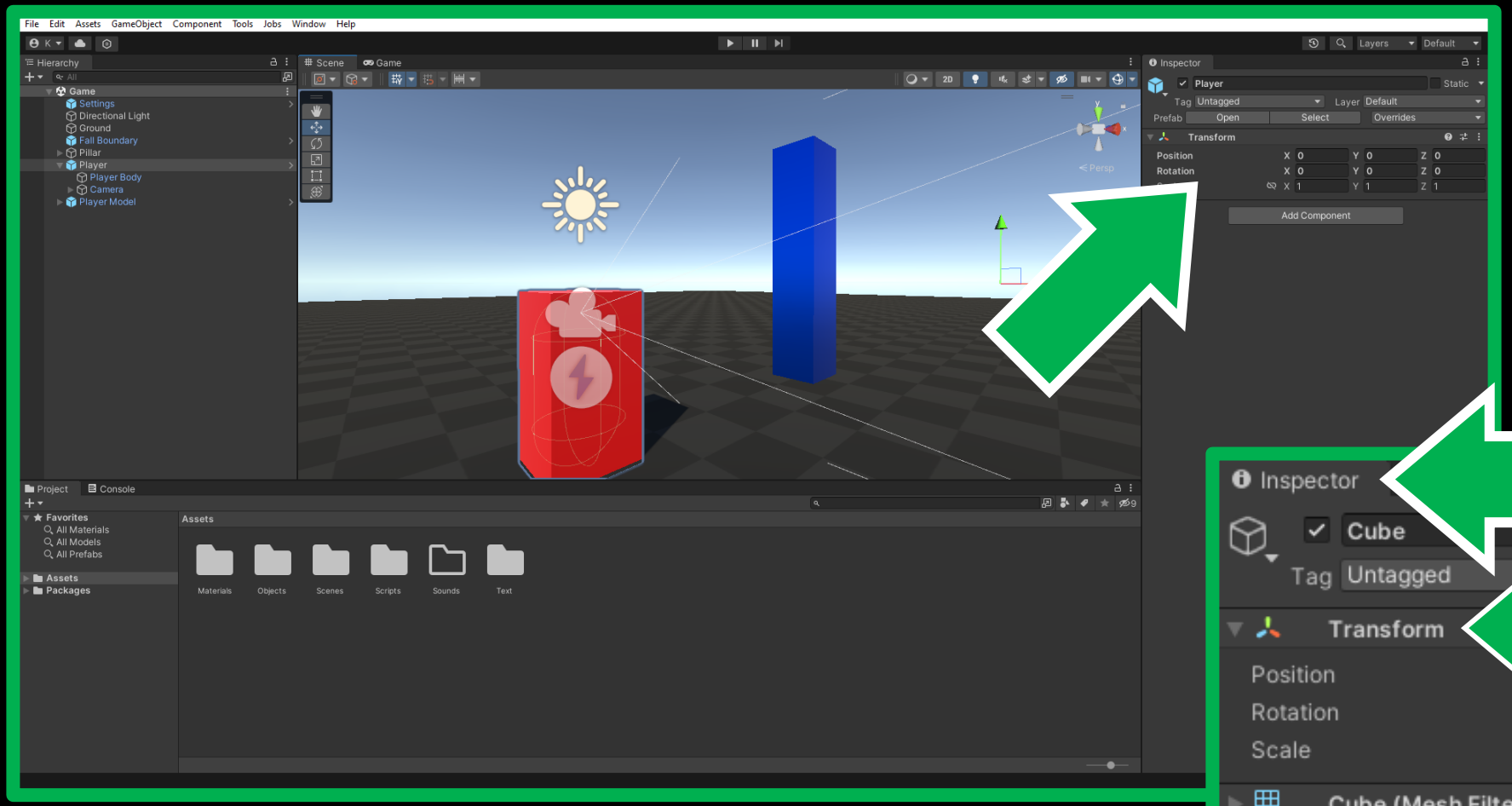
Green (Y Axis)
Up and Down

Red (X Axis)
Left and Right

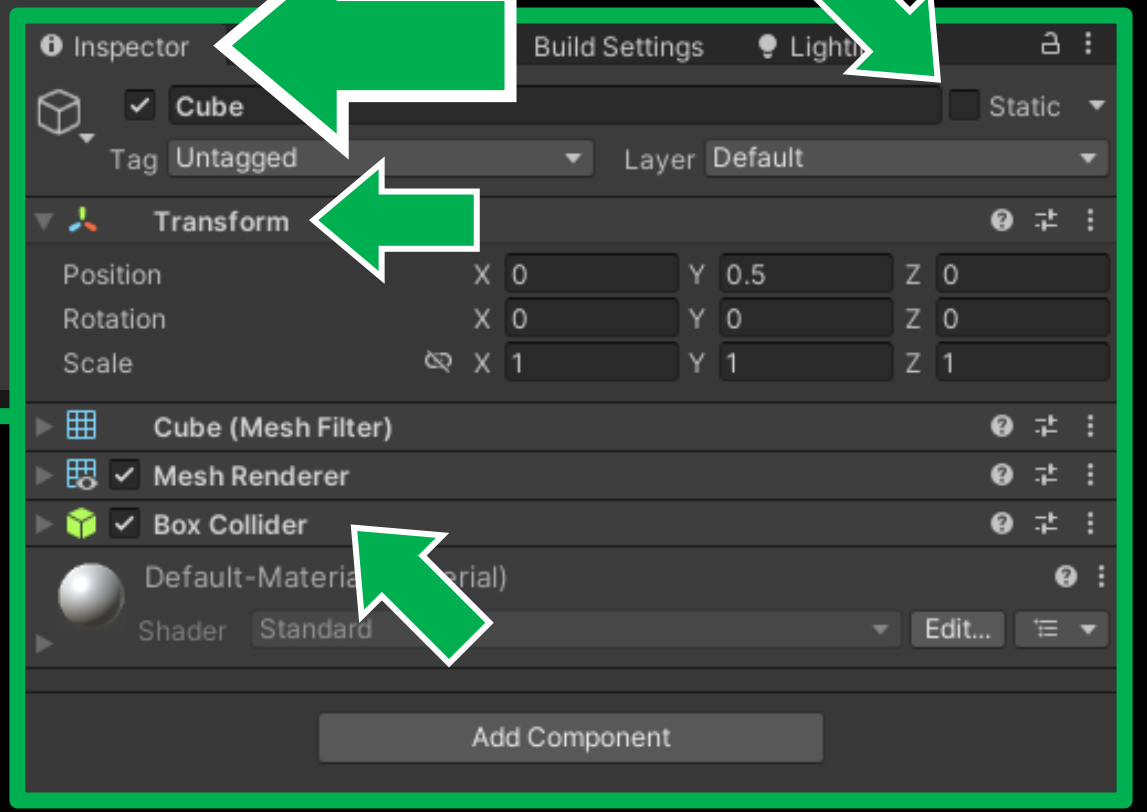
Blue (Z Axis)
Forward and Backward

You can drag an object on to another object to make it a Child of that Parent object. Parenting an object makes it the “origin” of its children. This means when that object is moved, rotated, scaled, or even deleted, its children will do the same relative to its parent. Relative values follow their parent values as their origin (0, 0, 0). For example, a parent object rotated 90 degrees will automatically cause its children to be rotated 90 degrees, even if the child displays 0 degrees rotated.

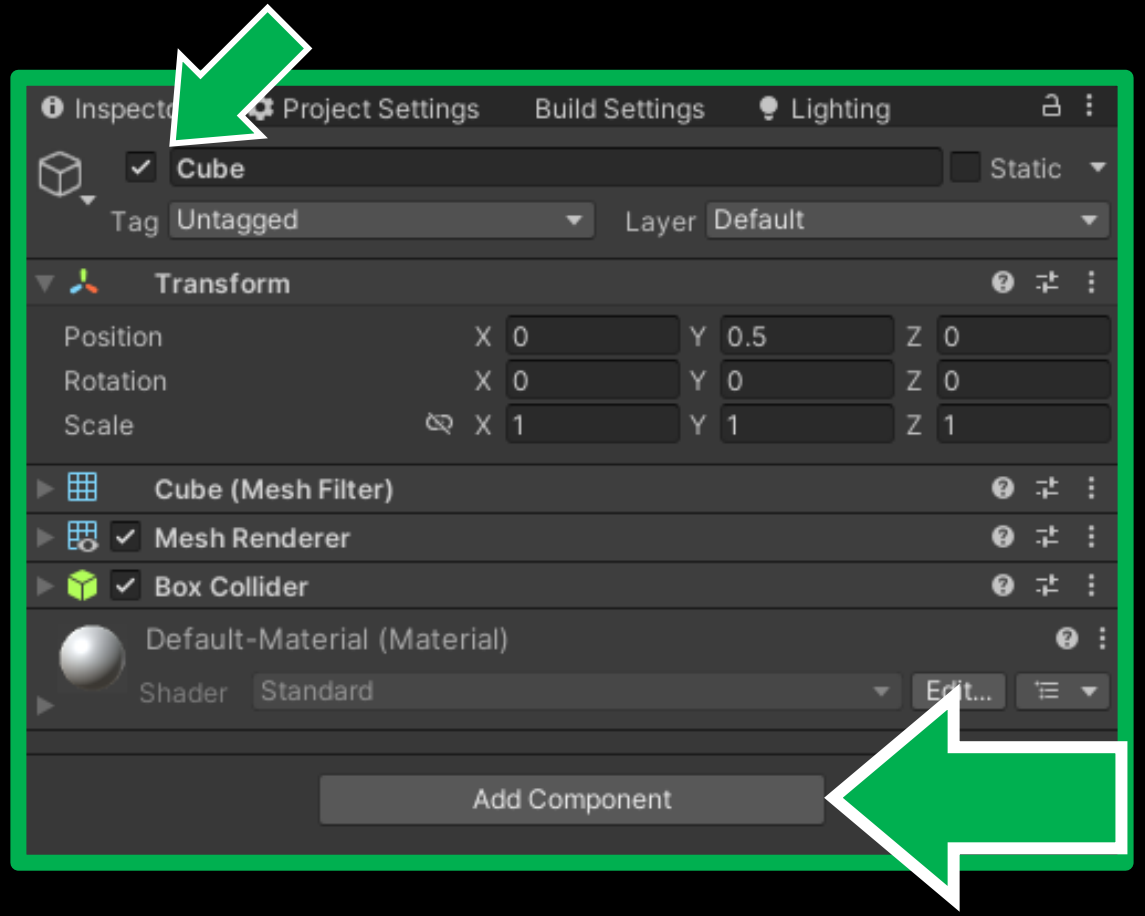
Parented objects are always above their child objects. They have an arrow next to their name indicating they have child objects.



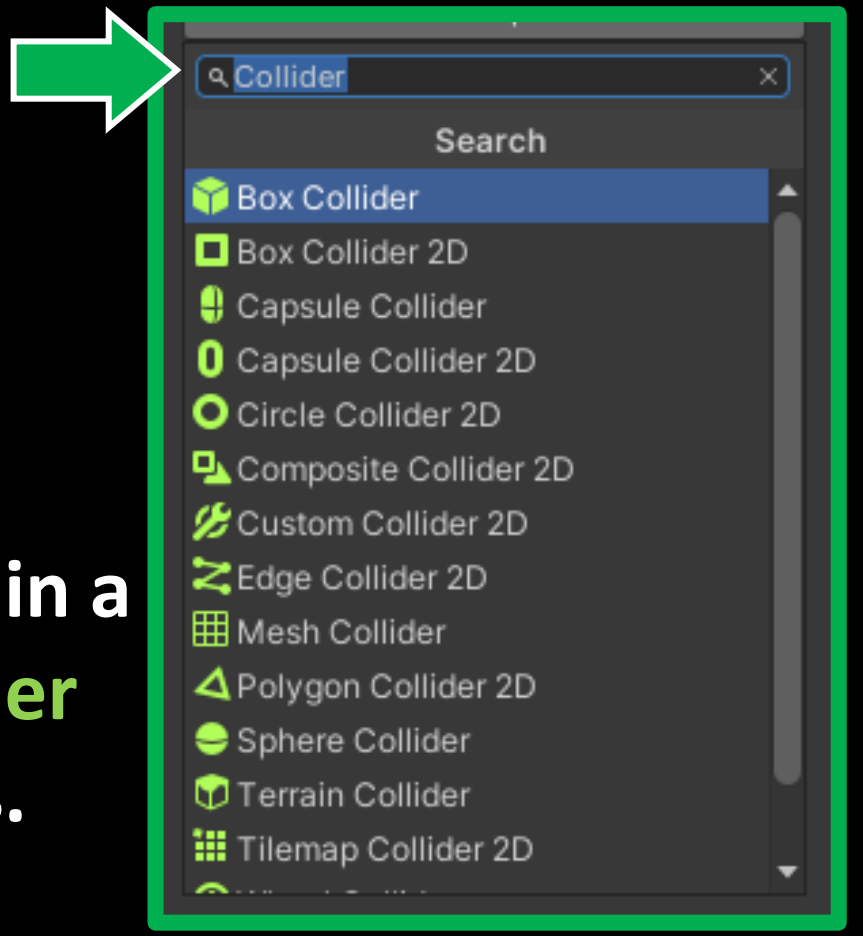
Tip! If an object does not move in a scene, click the Static check box.



This is the Inspector, where all an object's Components are located and where an object's Transform values are stored.

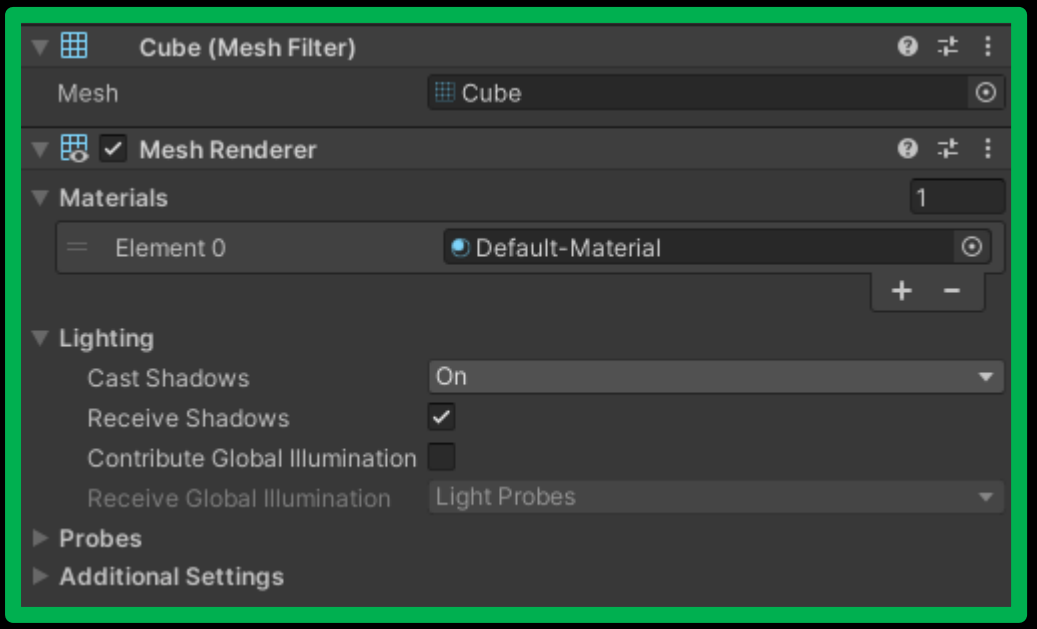


Tip! Click the check box to enable or disable a component. You can also right click the name of a component to copy, paste, or remove it!



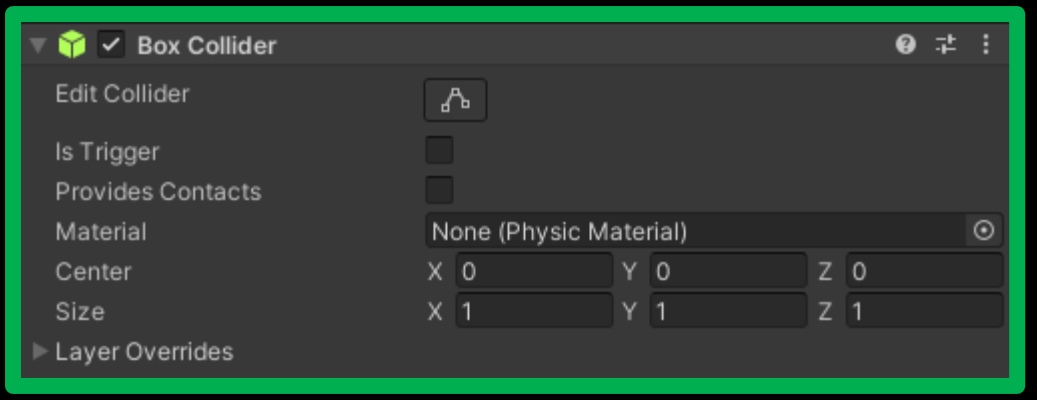
You can add Components to any object. Components are what make each object in a scene what it is. For example, type Collider in the search bar to find all collider types.

Transform – Tracks position, rotation, and scale as Vector 3 values.

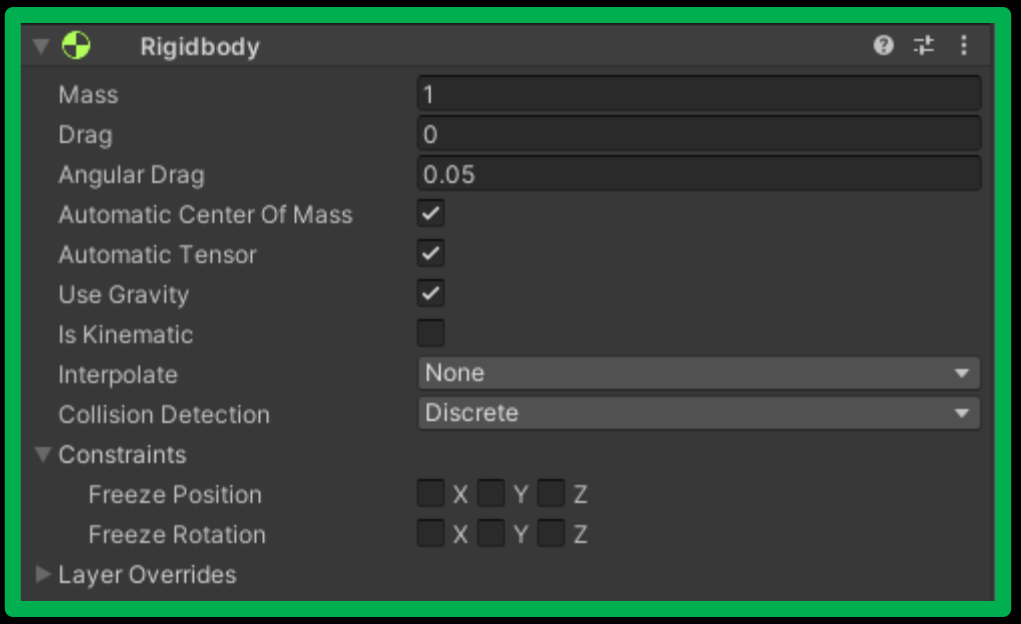


Mesh Filter – An object's model.

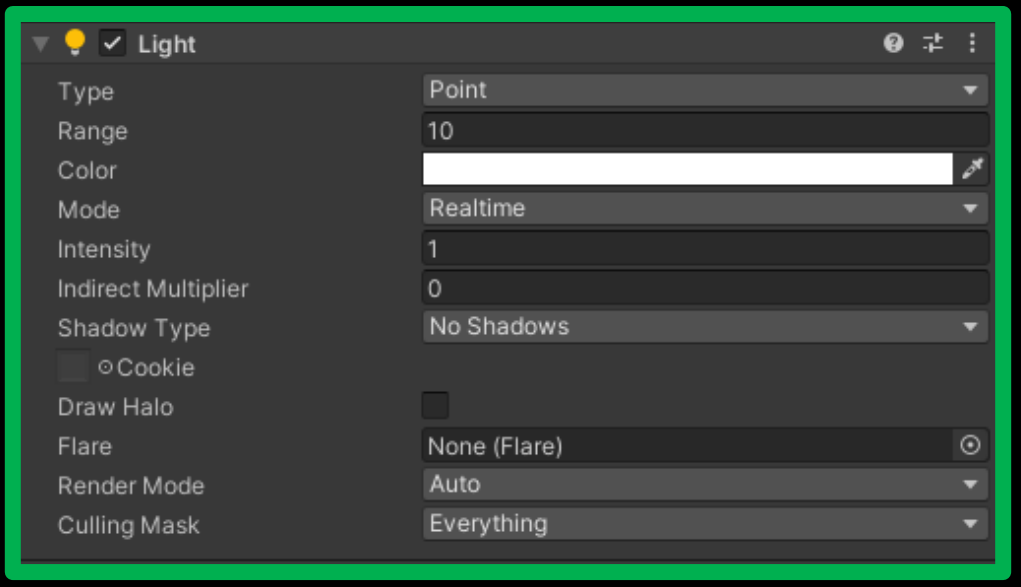
Mesh Renderer – An object's appearance, lighting settings, and materials.



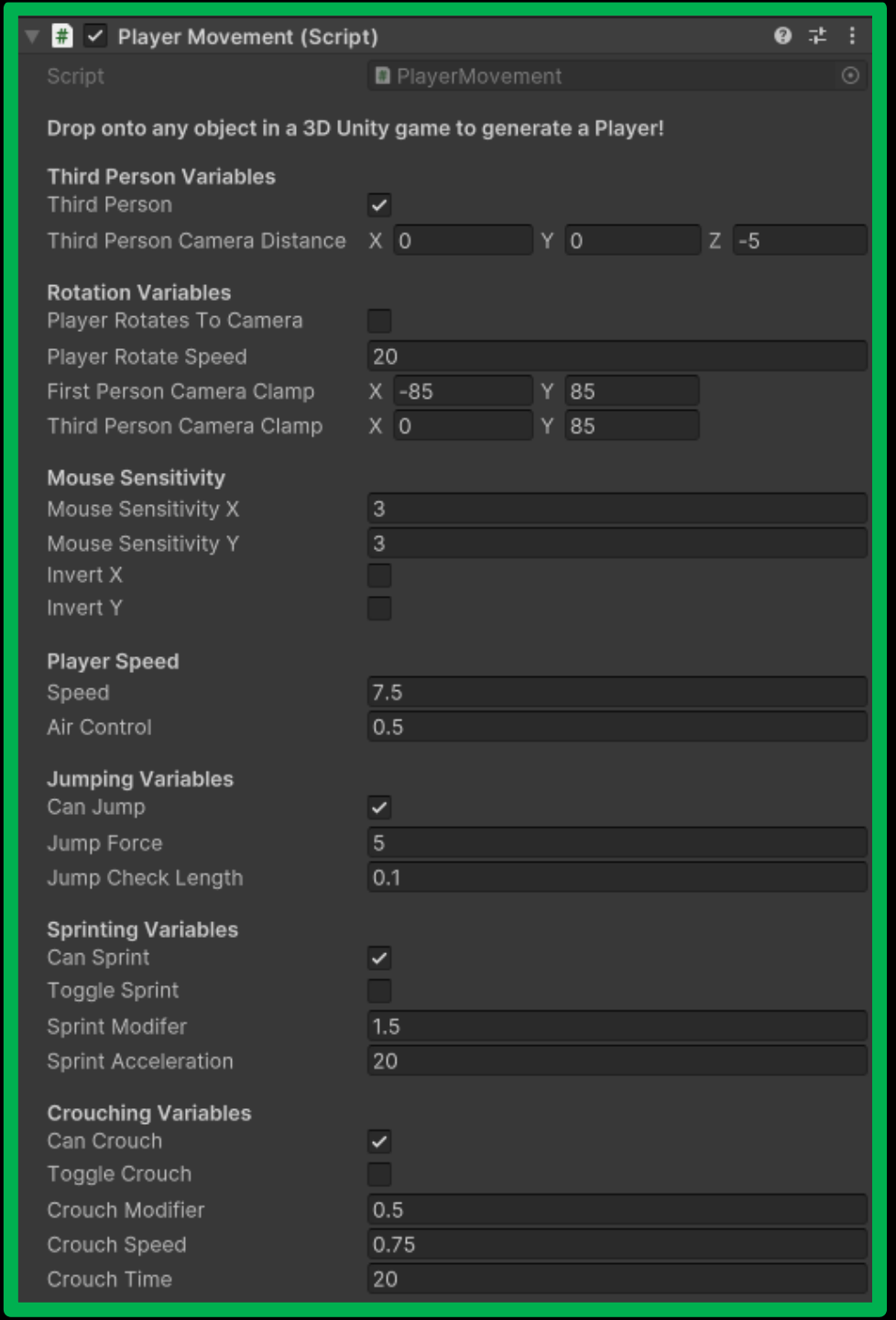
Collider – An object's collision.
Is Trigger is used to make it "passable" while still detecting collision. Use **Joints** for pivot points and **Physics Materials** for specific physics behavior.



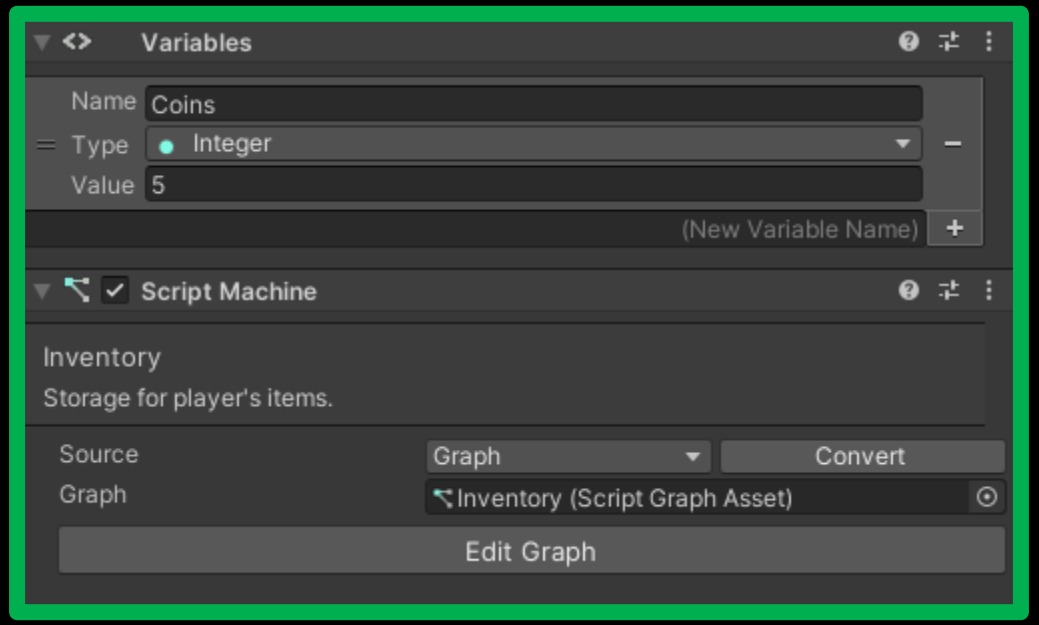
Rigidbody – Adds basic physics to an object. Features mass and gravity settings. Use constraints to lock position and rotation. Requires collision to not fall through the floor!



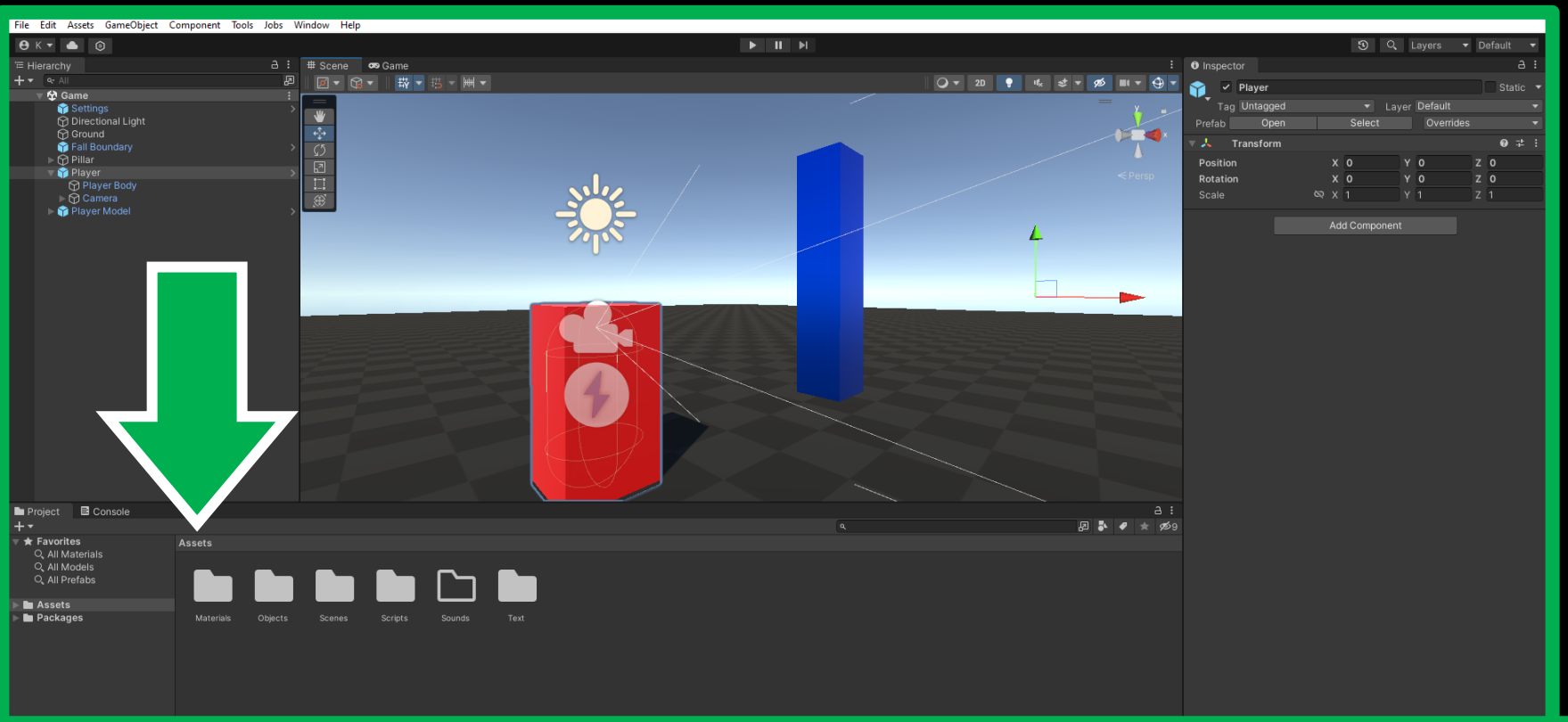
Light – Adds a light to the object. Point light is a sphere, spot light is a cone, and directional light is the "sun." Realtime lighting updates in the scene while baked lighting is stored as a texture.

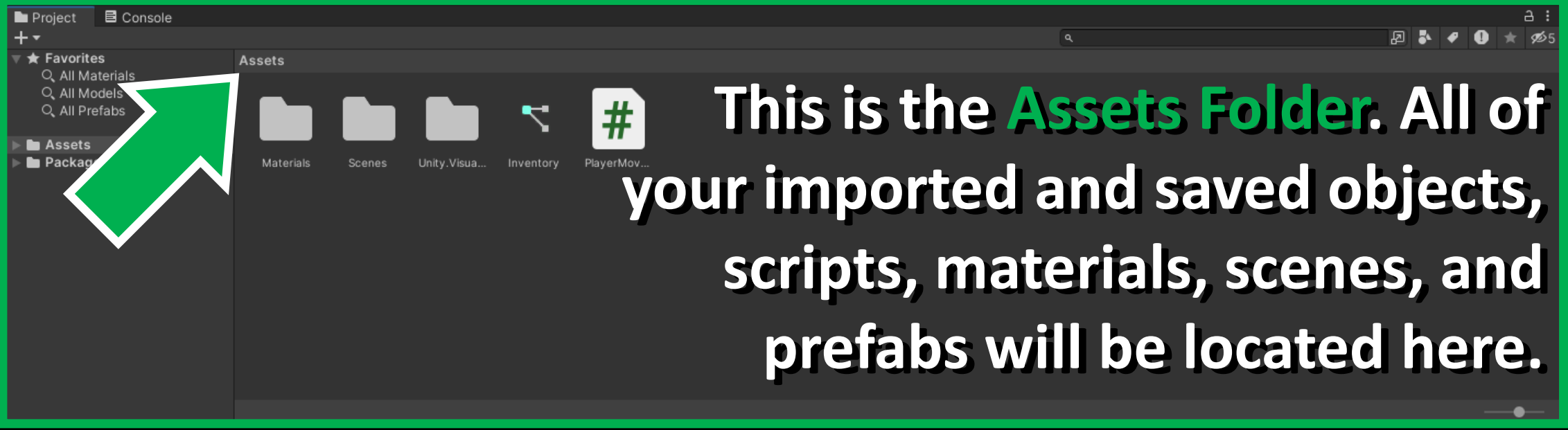


Script – You can add a customizable script from **C# code**. Scripts are how objects have behavior in a scene, but sometimes scripts have serialized fields that allow customization of values. These values can display as check boxes, numbers, words, drop downs, or even specific slots for **object** or **other component** references. Most scripts allow you to customize their values without further coding.

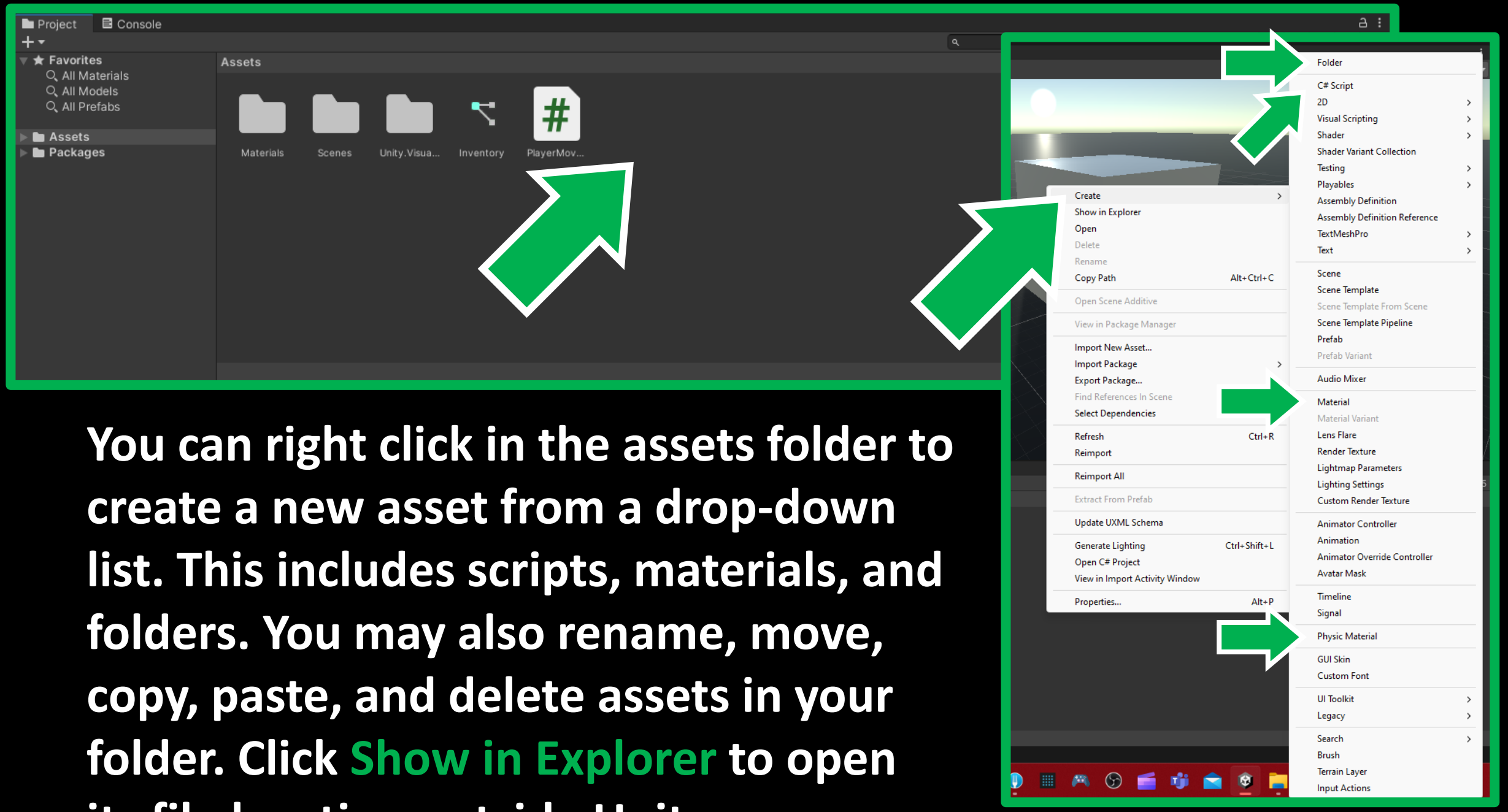


Script Machine – Using visual scripting, you can make your own behavior scripts. You can add them to an object by adding a Script Machine component. This includes a section to select which script graph you want to add, and another section for the variables associated with that object.



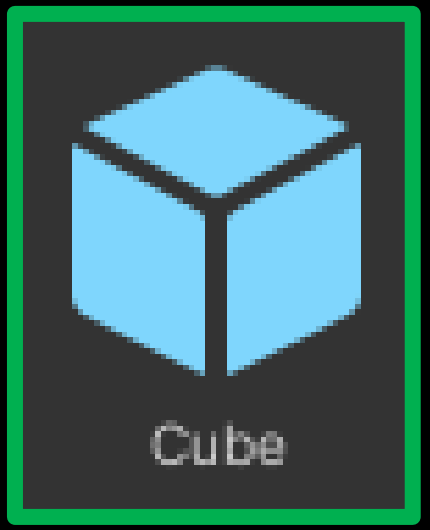


This is the **Assets Folder**. All of your imported and saved objects, scripts, materials, scenes, and prefabs will be located here.

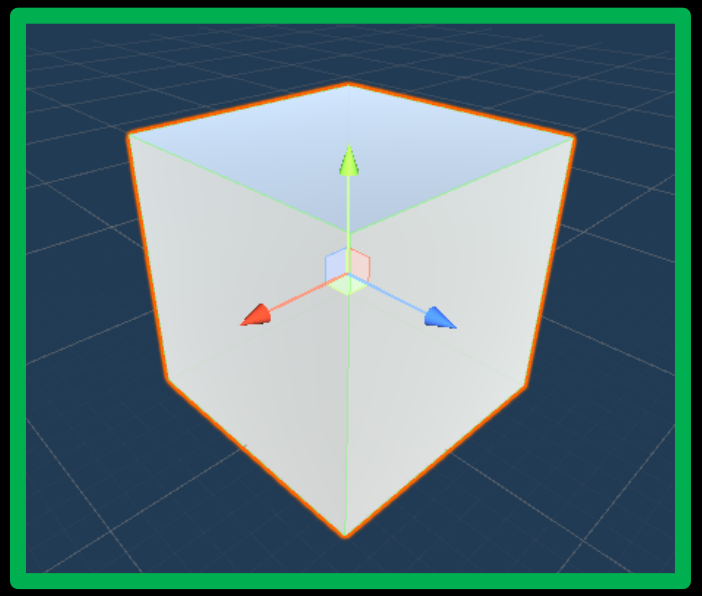


You can right click in the assets folder to create a new asset from a drop-down list. This includes scripts, materials, and folders. You may also rename, move, copy, paste, and delete assets in your folder. Click Show in Explorer to open its file location outside Unity.

Dragging an object from the scene into the Assets folder saves it as a **Prefab**. This is a great way to store objects you need to use multiple times. Editing a prefab (by double clicking it) updates all prefabs in your scenes when saved.

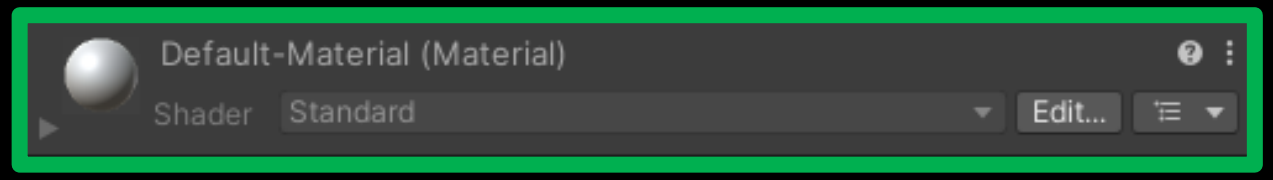


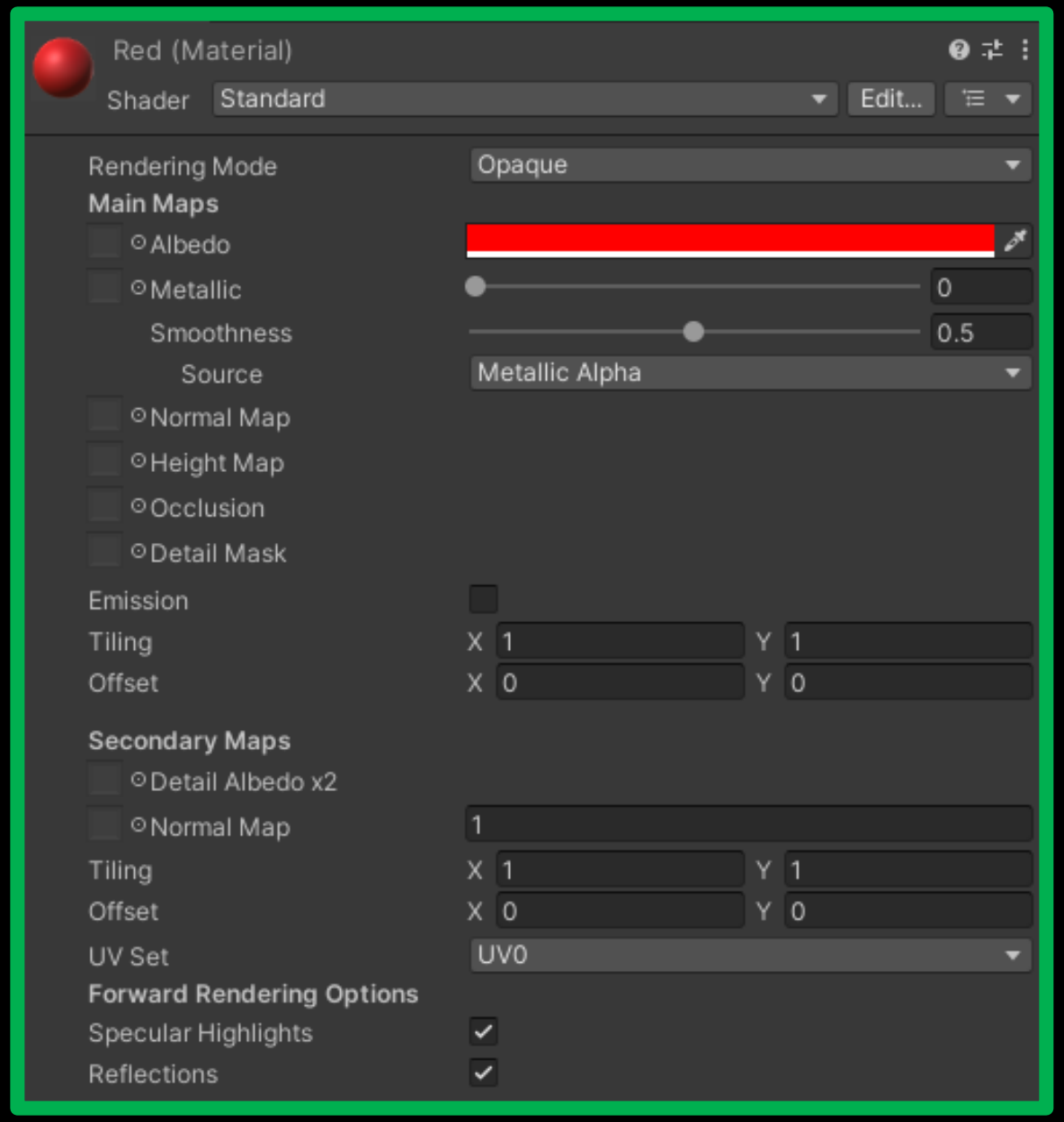
Prefab Icon



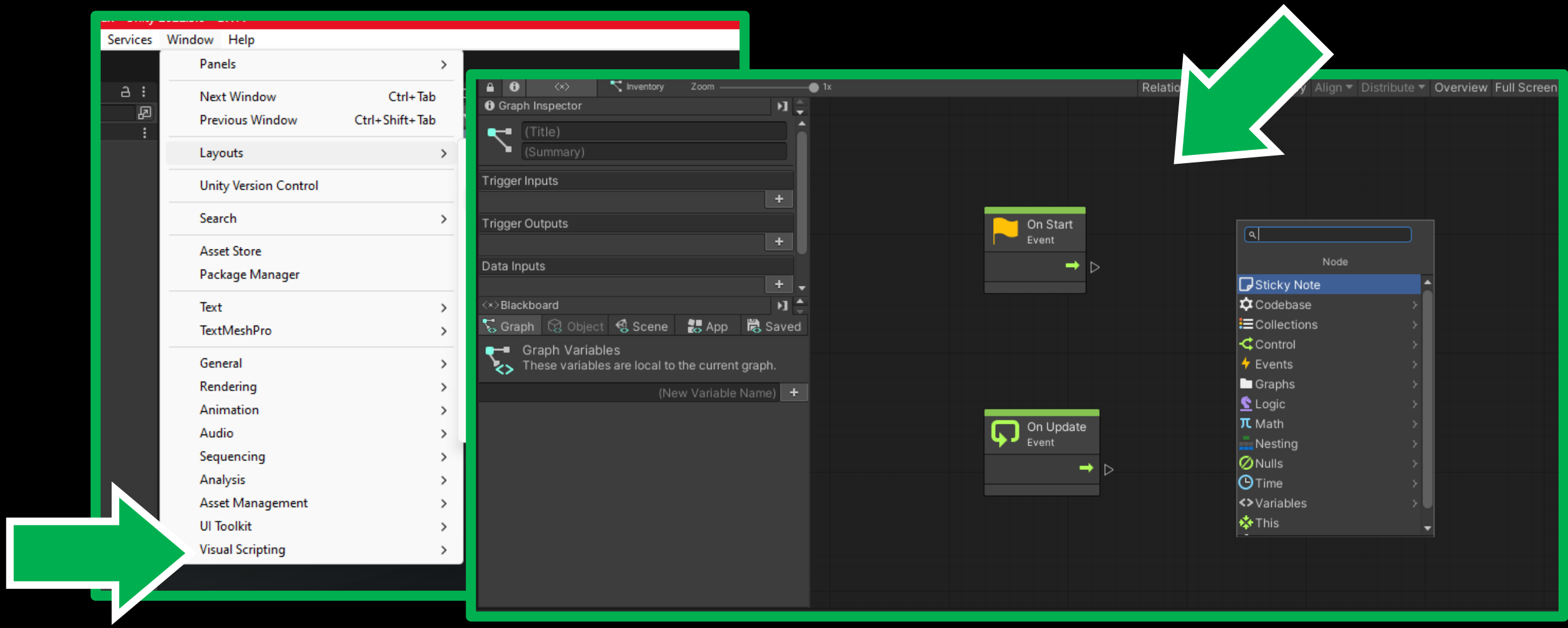
Prefabs always appear with **blue text** in the editor. Changes made to individual prefabs don't update every other prefab unless it is specifically the prefab template being edited. While editing a prefab template, the scene has a blueprint-like (or white) background.

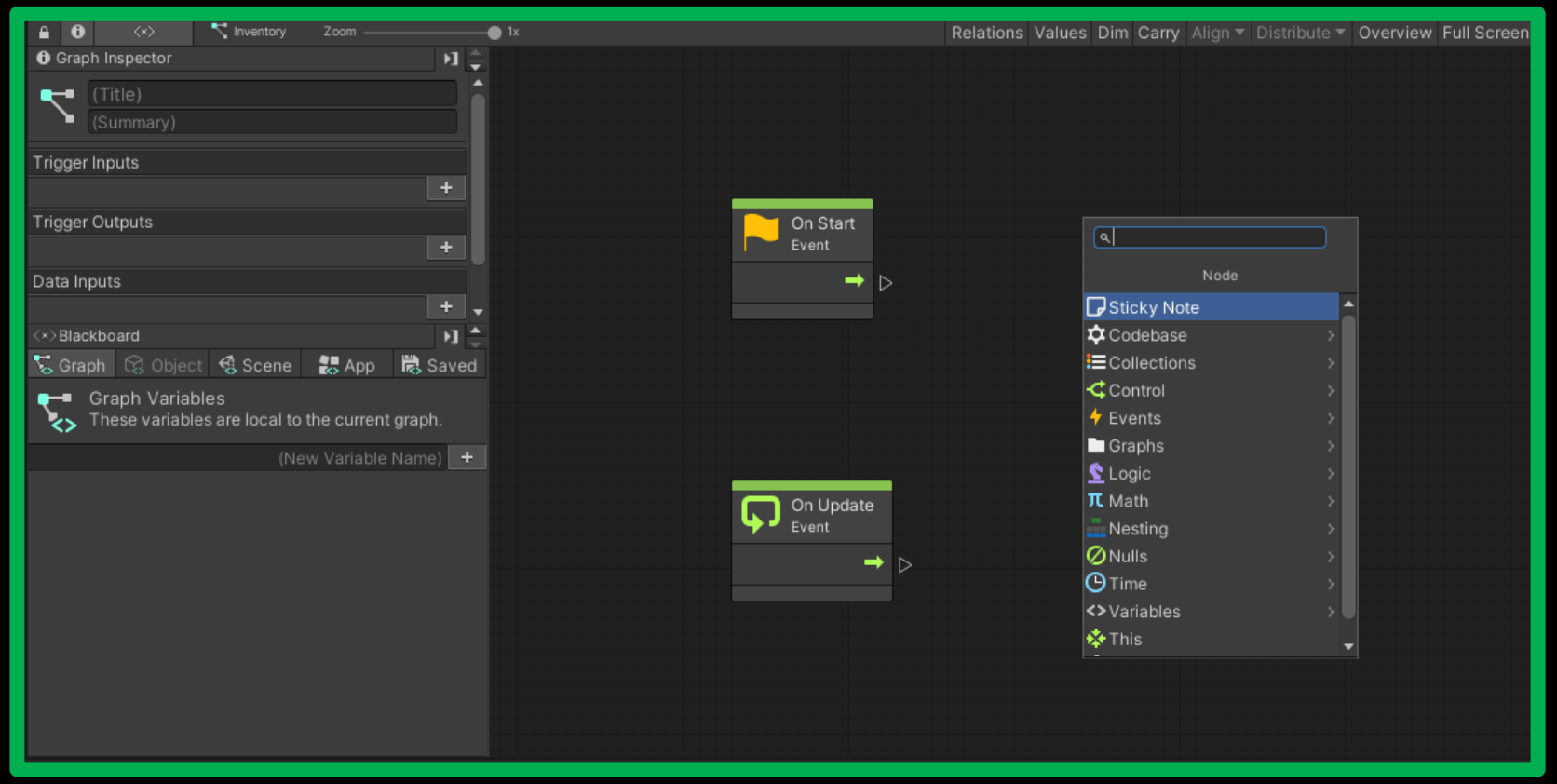
When creating a Material in the assets folder, there are some important settings to keep in mind. You can select whether to render Opaque or Transparent. Albedo is color, and Metallic and Smoothness define how shiny the material is. Emission is whether the material glows, how much it glows, and what color it glows. You can define a material's Texture for each of these settings by clicking the small circle in front of its name. Tiling controls the size of the texture across objects.



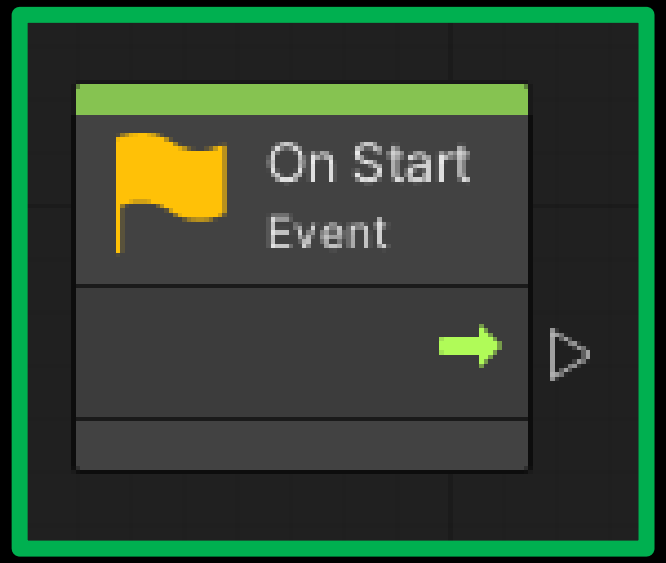


You can add Visual Scripting by clicking Window, Visual Scripting, Visual Scripting Graph. Click Create and wait until you see the following menu.

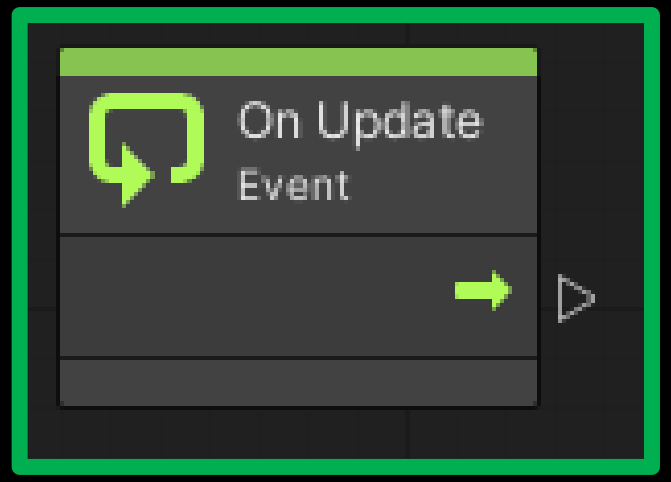




Visual Scripting is a simplified version of coding that allows you to give objects behaviors via a Script Machine. It is largely up to you to find out what you want to make and how to make it, but here are some important tips.

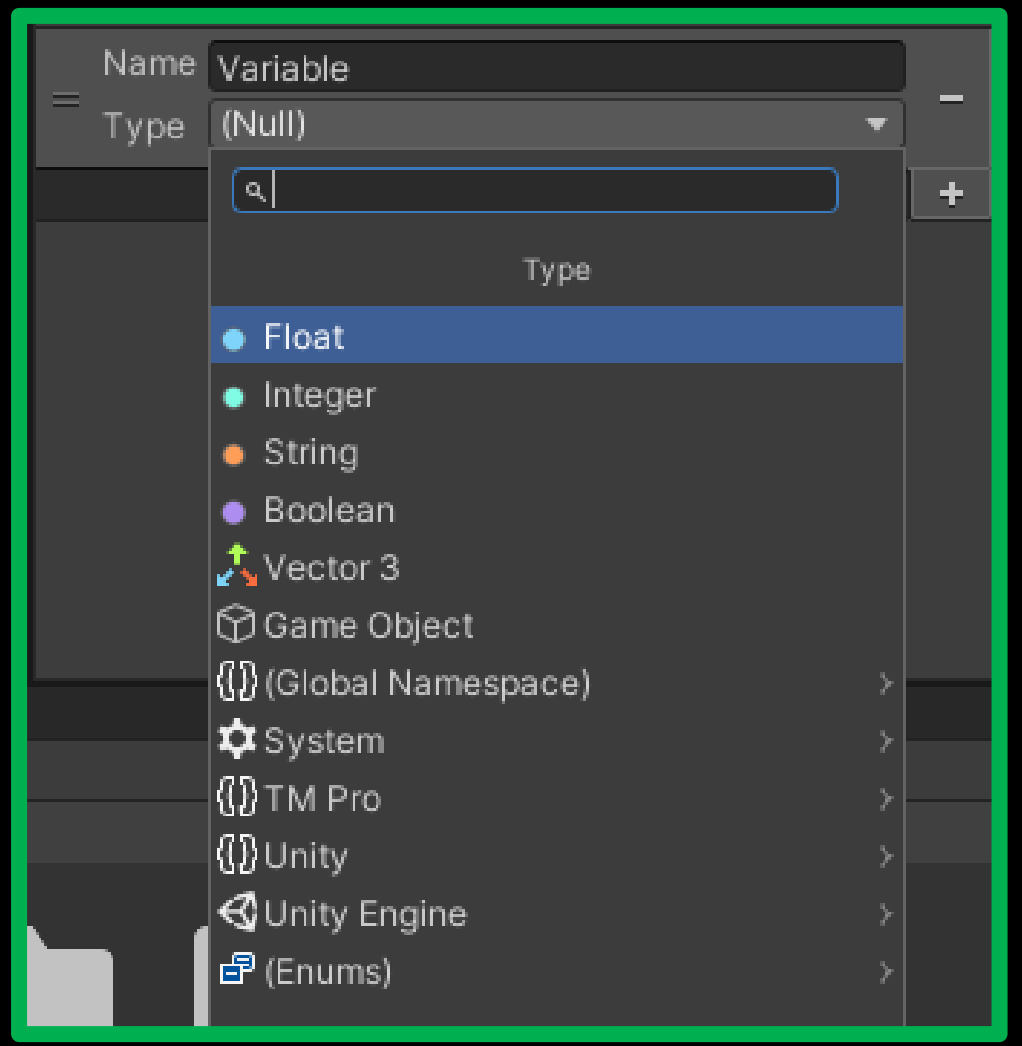


On Start is ran when an object is activated / created in a scene.

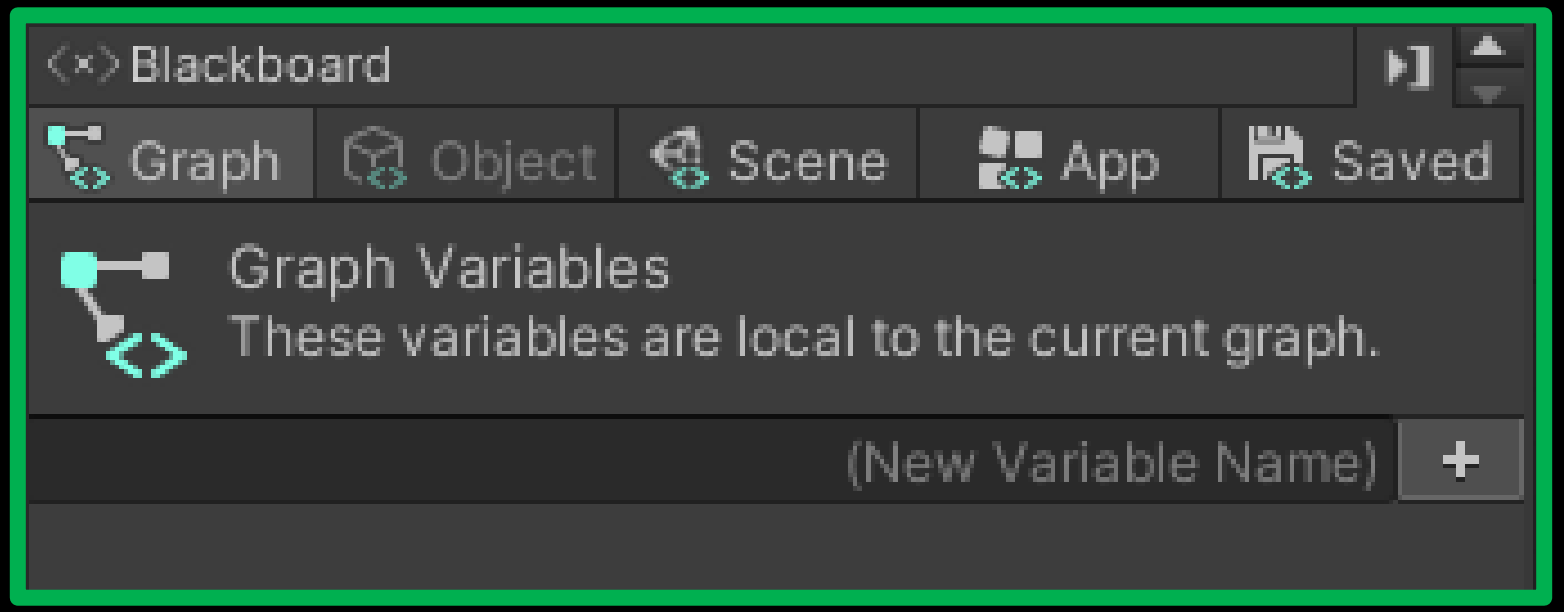


On Update is ran as fast as the game's framerate. **On Fixed Update** is similar but instead runs at fixed intervals, making it good for physics and consistent movement.

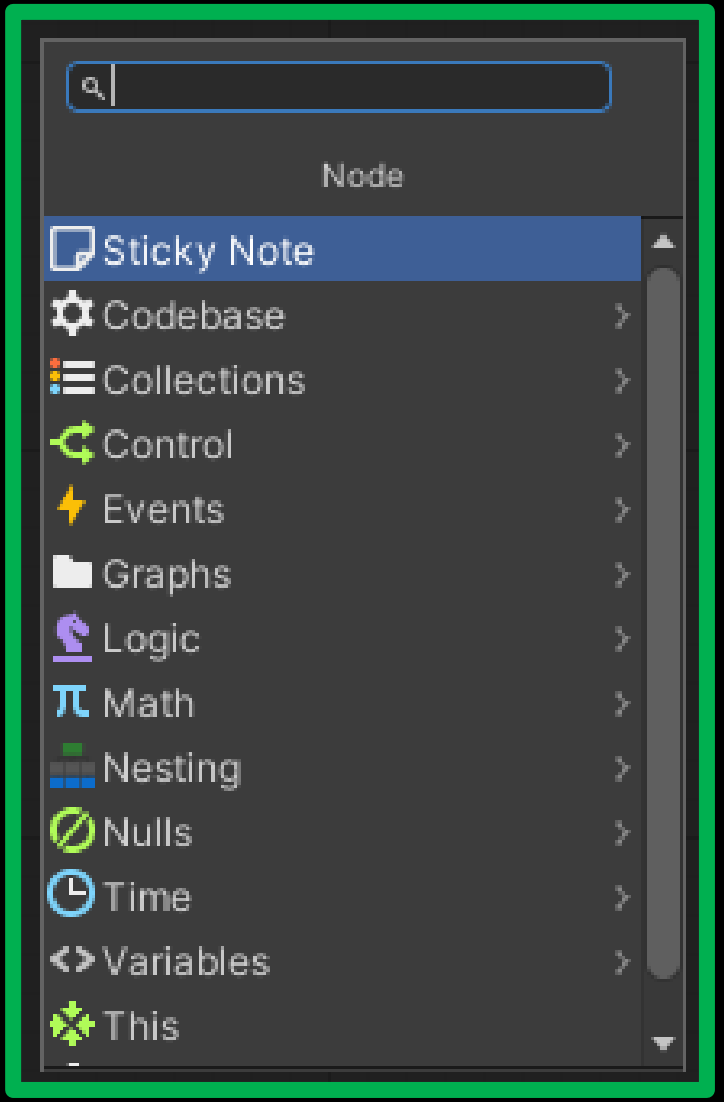
Variables are ways to store data on objects or in the scene. These can include **Floats** (decimal numbers), **Integers** (whole numbers), **Strings** (text), **Booleans** (true and false), **Vector 3** (three floats that store transform values), and **Game Objects** (objects in your scene). You can make a variable by first giving it a name, clicking the **+ button**, and then setting its type.



These variables are stored either in the Graph (or script), the Scene (your current world), or on an Object (whatever has the script as a component). Graph and Object variables are usually considered Private and are only accessible by that exact script, while Scene variables are usually considered Public and can be accessed by any script.



There are lots of different ways to create whatever you desire with scripting. A lot of coding is just trial and error and research. Search online for documentation about different parts of Unity's code to learn more, or simply try different nodes and see what happens!



Unity Documentation

Kyle Furey's Visual Scripting Tutorial

Tutorial Document by Kyle Furey

Made for educational purposes.