Lab 1.2 Unity basics

(Note: you don't have to submit anything for this lab, but I highly recommend you to try the Todos to get familiar with Unity basics)

Introduction

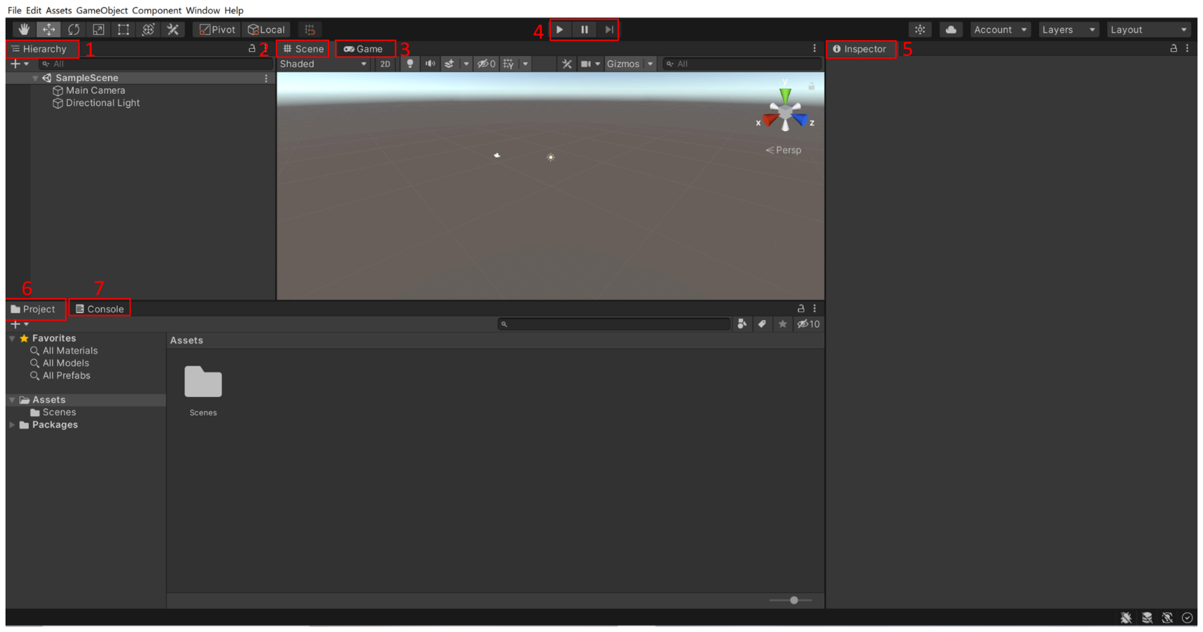
Last time we downloaded and set up Unity. You should be able to create new projects in the future for the coming projects or the final project. In this lab, we will work on some Unity basics.

Here are the things covered:

- Unity panels
- GameObject and Scripts (C#)
- Engine basics (lights, camera, rigidbody, etc.)

Real work

Unity Panels

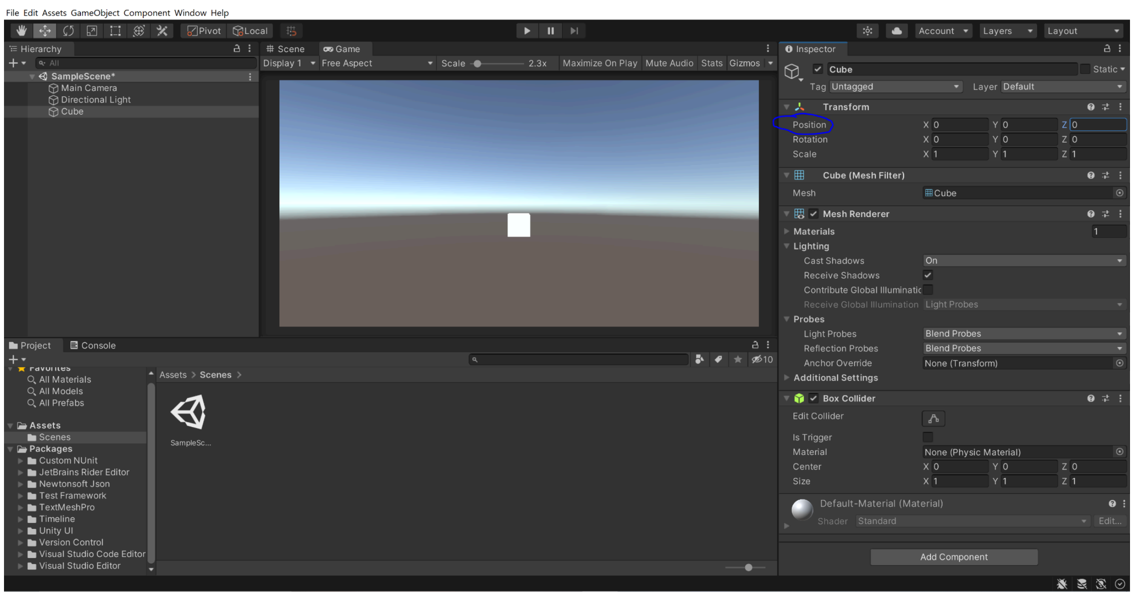


1. **Hierarchy:** This panel manages all the components in the scene, such as the camera, light, and objects. It's very flexible to add new objects to the hierarchy. You can either create them directly or drag them from the folders.
2. **Scene:** You can visualize the objects here, including their position, orientation, scale, etc. You can use your mouse to zoom in/out, rotate the view, or drag objects around.
3. **Game:** This panel is for visualizing the scene at runtime. You will be able to see the objects from the chosen camera. The default one is the main camera, but you can choose from different cameras if you have several.

4. **Play/Pause/Step:** This is not a panel, but it will allow you to run your game and debug your code. Every time you press the start button, Unity will start a new game. You don't need to worry much about how Unity does it (really nice, isn't it).
5. **Inspector:** This panel is used for changing the parameters of the GameObjects. One advantage of Unity (and other game engines) is that you can change parameters without making any changes to the code. You can also change these parameters temporarily at runtime (by clicking the play button). The changes made at runtime won't be applied after you stop playing.
6. **Project:** You can manage the source files (scripts, downloaded assets, Prefabs, etc.). Anything you imported (such as the Oculus integration package we will use later) will be found here.
7. **Console:** This panel displays messages at runtime. It shows the execution information, such as building success or error. You can also print messages through the built-in Debug.Log function.

GameObject ([Scripting API: GameObject](#))

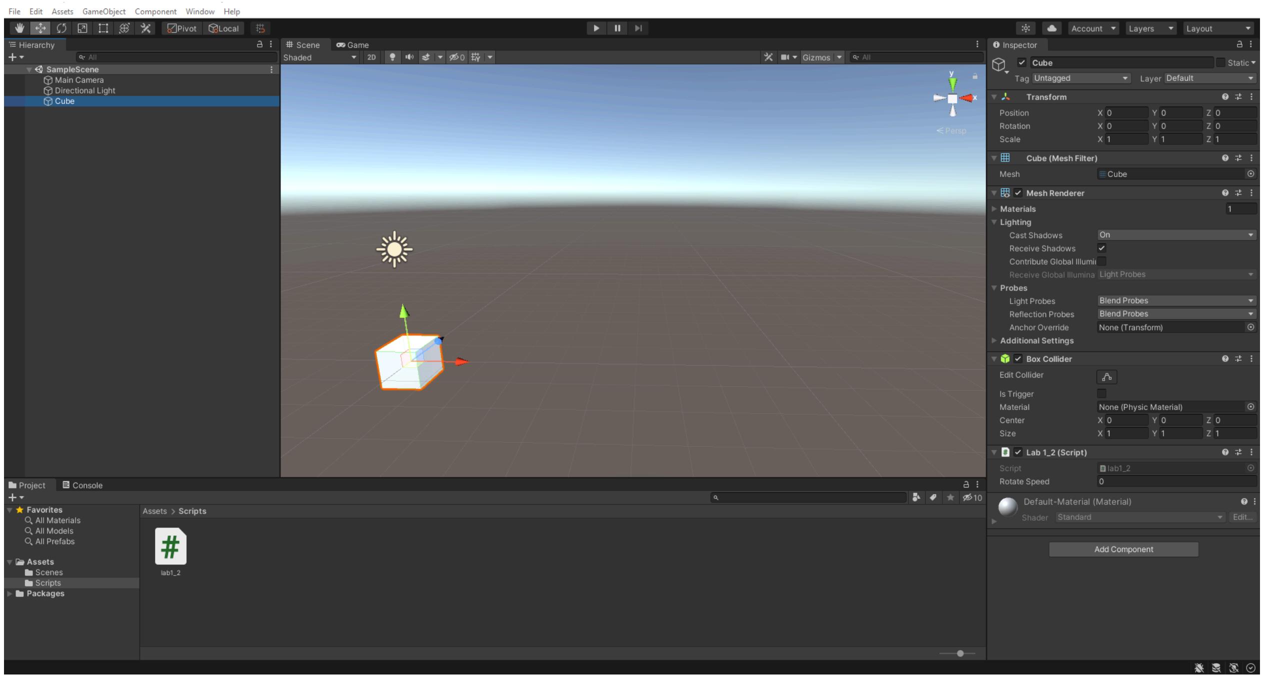
1. GameObject is the "Base class for all entities in Unity Scenes." Everything in the Hierarchy panel is a GameObject. You can read the manual above and see its properties and functions. Understanding GameObject is very important for manipulating the objects in the scene at runtime.
2. Todo: Create a cube by right-clicking in the Hierarchy panel and choosing "3D Object -> Cube". Then change its position to (0, 0, 0). You would view it through the main camera in the Game panel if you didn't move the main camera. You can also play with other properties such as its render or material.



Cube | Static
Tag Untagged | Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Cube (Mesh Filter)
Mesh Cube
Mesh Renderer
Materials 1
Lighting
Cast Shadows On
Receive Shadows ✓
Contribute Global Illuminatic
Receive Global Illumination Light Probes
Probes
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Additional Settings
Box Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1
Default-Material (Material)
Shader Standard

3. To manipulate any GameObject at runtime, you must attach scripts to it. You can attach several scripts to a single GameObject, but be careful since it may become too complicated to debug.

4. Todo: First, create a folder called “Scripts” in Assets in the Project panel (of course, you can create a folder anywhere with any name you like). Then in the folder, create a new C# script by right click and choosing to Create -> C# script. After creating the script, drag and attach the script to the cube. I will provide you with some sample code, and feel free to add your code.



Cube — Static
Tag Untagged — Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

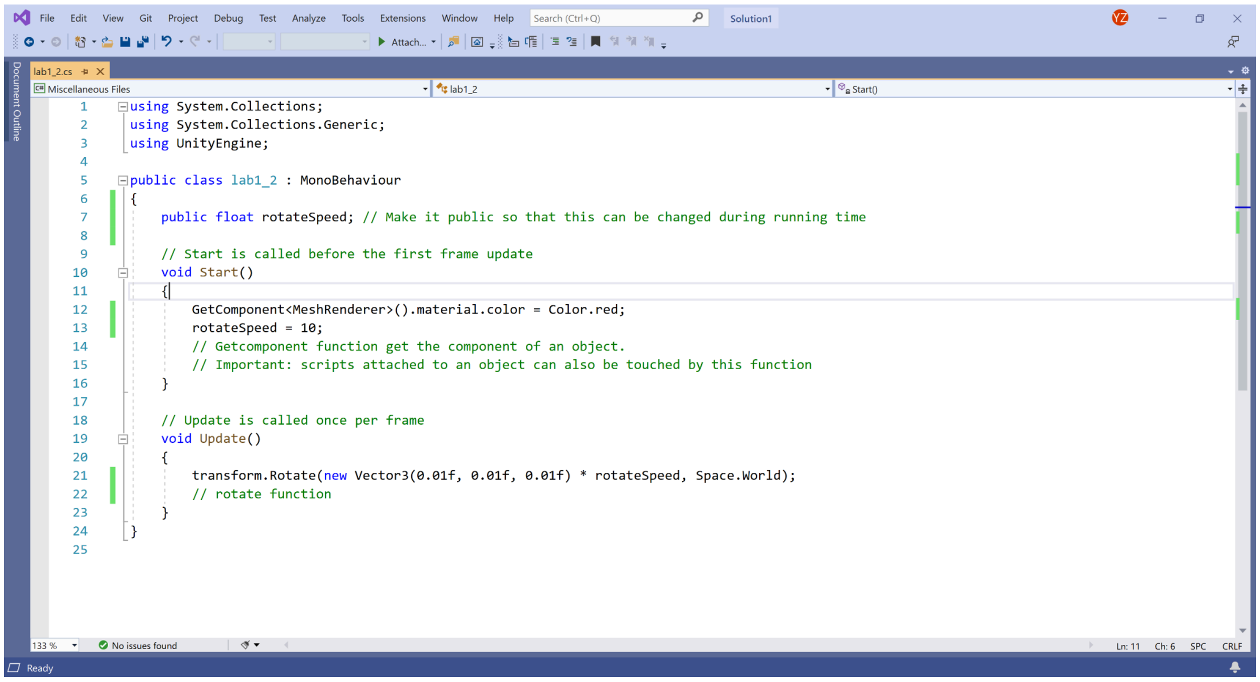
Cube (Mesh Filter)
Mesh Cube

Mesh Renderer
Materials 1
Lighting
Cast Shadows On
Receive Shadows ✓
Contribute Global Illumi
Receive Global Illumina Light Probes
Probes
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Additional Settings

Box Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1

Lab 1_2 (Script)
Script lab1_2
Rotate Speed 0

Default-Material (Material)
Shader Standard



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class lab1_2 : MonoBehaviour
{
    public float rotateSpeed; // Make it public so that this can be changed during running time

    // Start is called before the first frame update
    void Start()
    {
        GetComponent<MeshRenderer>().material.color = Color.red;
        rotateSpeed = 10;
        // Getcomponent function get the component of an object.
        // Important: scripts attached to an object can also be touched by this function
    }

    // Update is called once per frame
    void Update()
    {
        transform.Rotate(new Vector3(0.01f, 0.01f, 0.01f) * rotateSpeed, Space.World);
        // rotate function
    }
}
```

This is Visual Studio 2019, and the current default version should be 2022, which doesn’t make a difference.

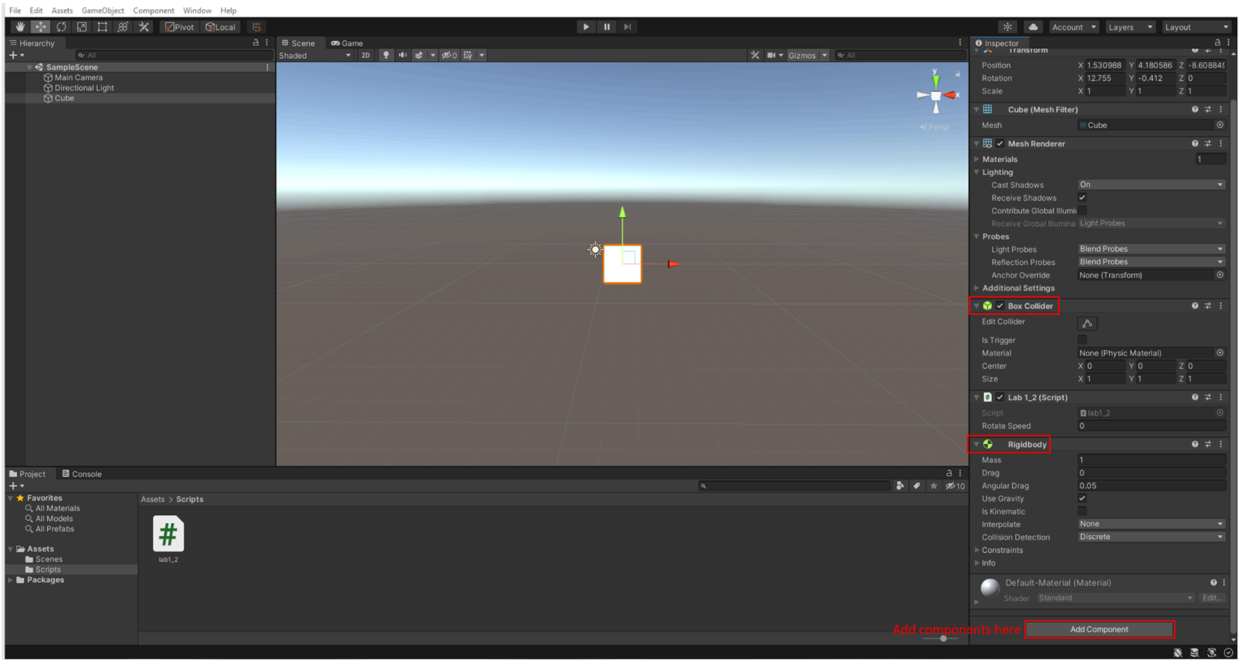
Light and camera

Light: Unity supports many basic light types, such as spot and directional. For the projects in this course, you don't need to worry about the light. However, you can do some fancy things by changing the light settings or adding more light sources.

Camera: Unity provides two camera models, perspective (pinhole) and orthographic. The perspective model works similar to our own eyes, with depth information. The orthographic model works like a 2D scene. These two kinds of camera models are enough for all the projects in this course.

Rigidbody (Todo (optional): Unity3D Physics - Rigidbodies, Colliders, Triggers)

Unity has a robust physics system, and rigidbody is the basic component. A rigidbody can be attached to objects and "insert" them into the physics system. To attach a rigidbody, you can click the "Add Component" button in the Inspector panel and search for rigidbody.



Transform
Position X 1.530988 Y 4.180586 Z -8.60884
Rotation X 12.755 Y -0.412 Z 0
Scale X 1 Y 1 Z 1

Cube (Mesh Filter)
Mesh: Cube

Mesh Renderer
Materials: 1
Lighting
Cast Shadows: On
Receive Shadows: ✓
Contribute Global Illumi
Receive Global Illumina: Light Probes
Probes
Light Probes: Blend Probes
Reflection Probes: Blend Probes
Anchor Override: None (Transform)
Additional Settings

Box Collider
Edit Collider
Is Trigger
Material: None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1

Lab 1_2 (Script)
Script: lab1_2
Rotate Speed: 0

Rigidbody
Mass: 1
Drag: 0
Angular Drag: 0.05
Use Gravity: ✓
Is Kinematic
Interpolate: None
Collision Detection: Discrete
Constraints
Info

Default-Material (Material)
Shader: Standard

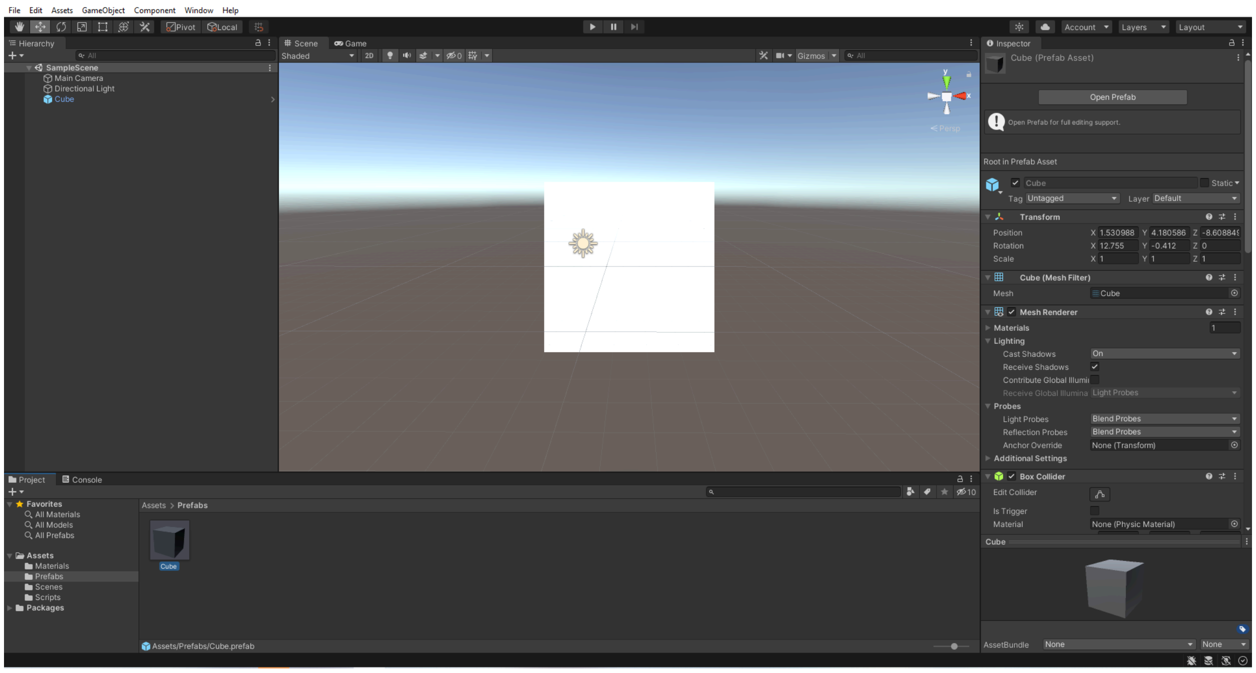
To detect collisions between objects, you need to attach colliders (you can have more than one collider) to them. Usually, you attach basic colliders, such as box colliders or sphere colliders, to avoid massive calculations. However, you can also have a mesh collider to make the collision more accurate. If you want to download assets from the asset store for your projects (even free ones), you can find some assets with mesh colliders.

Here are some basic functions of detecting collision:
OnCollisionEnter(Scripting API: Collider.OnCollisionEnter(Collision)) and OnTriggerEnter(Unity - Scripting API: Collider.OnTriggerEnter(Collider))

Collision is quite complicated, and the best way to learn about it is to practice it. The video attached is helpful. It gives you a clear explanation of collisions and triggers and contains some practice. If you want some specific colliding effect, please post on Piazza or contact TA for solutions.

Prefabs (Prefabs)

Unity allows you to store a GameObject you have set up and use it in the same or other scenes. Then, you can simply drag the object to any folder you want and create a prefab. Unlike copying and pasting, you can use the same object with the same configurations, even in other projects.



Cube (Prefab Asset)

Open Prefab for full editing support.

Root in Prefab Asset

Cube — Static
Tag Untagged — Layer Default

Transform
Position X 1.530988 Y 4.180586 Z -8.60884
Rotation X 12.755 Y -0.412 Z 0
Scale X 1 Y 1 Z 1

Cube (Mesh Filter)
Mesh Cube

Mesh Renderer
Materials 1
Lighting
Cast Shadows On
Receive Shadows ✓
Contribute Global Illumi
Receive Global Illumina Light Probes
Probes
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Additional Settings

Box Collider
Edit Collider
Is Trigger
Material None (Physic Material)

Cube

AssetBundle None None