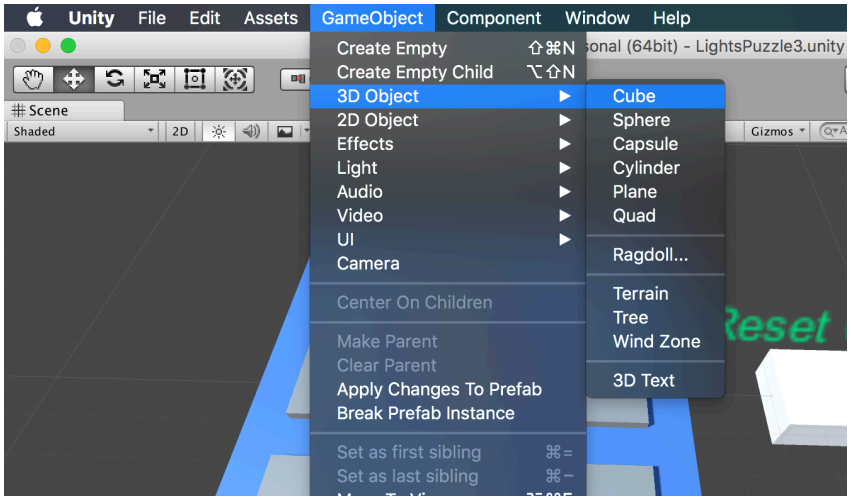
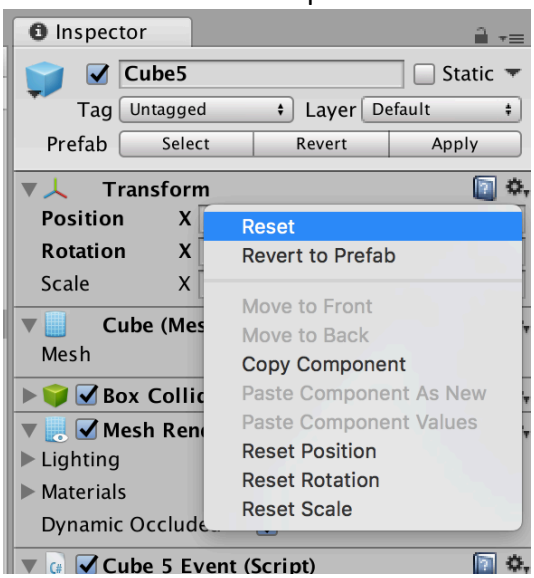
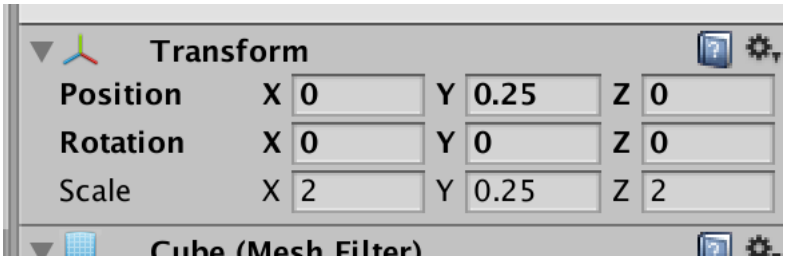
1. Download Unity3D and get a personal license as a student developer.
2. After opening the Unity3D, I found two objects in the 3D scene, one is the light source and another is the camera, I can easily understand what are they used for based on their names.
3. In order to build a 3*3 lights game in a 3D scene, I should create 9 objects in the 3D scene, so first step is to find how to add an object (cube) in the scene. After exploration, I found how to add an object into game scene. Press GameObject in the top menu -> 3D object -> 3D cube.



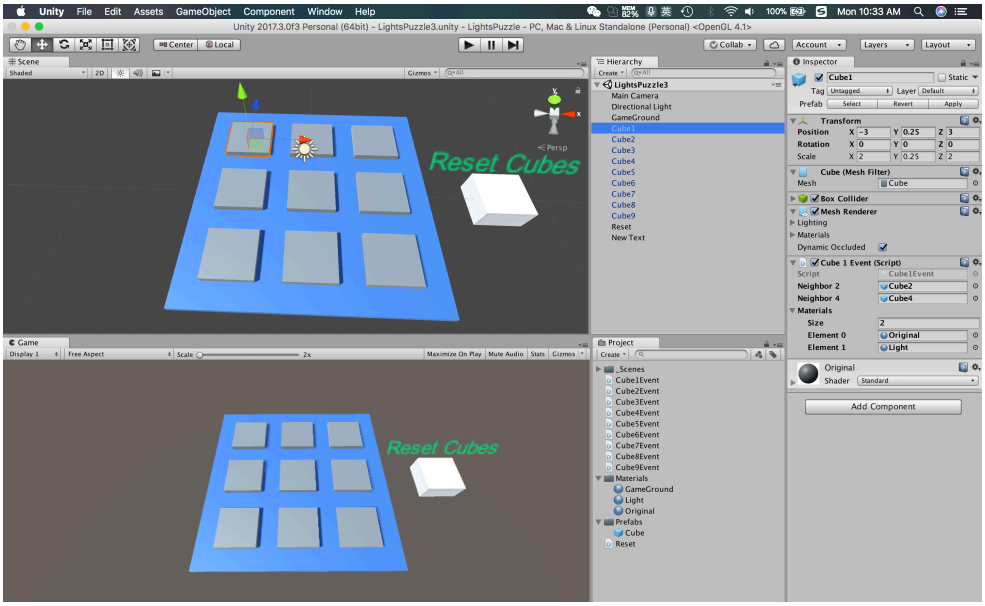
4. After building a 3D cube in the game scene, I continued to explore how to handle developer's view when creating games: Developer can use right button of mouse to change the view angle, use scroll wheel of mouse to zoom in and out, use cursor key in the keyboard to change view position.
5. The position of Game Objects are very important, after investigation, I found I can use Transform in Inspector to set the position of Game Objects. Every time I build a new Object, I will reset the position in Transform Inspector.



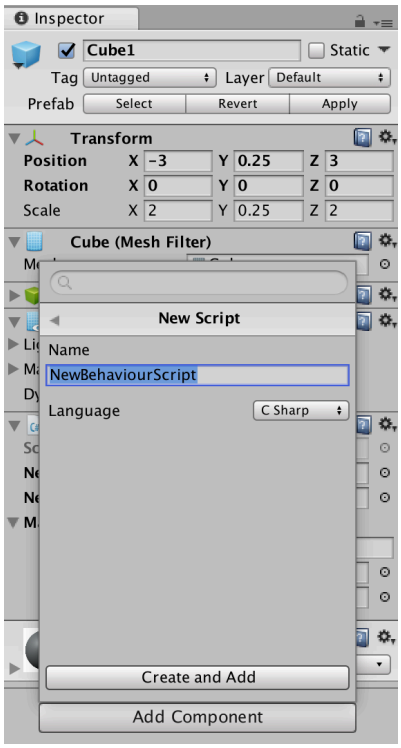
6. Position in Transform controls the position of current Object, Rotation controls the direction of three axis, and Scale controls the size of current Object in three axis.



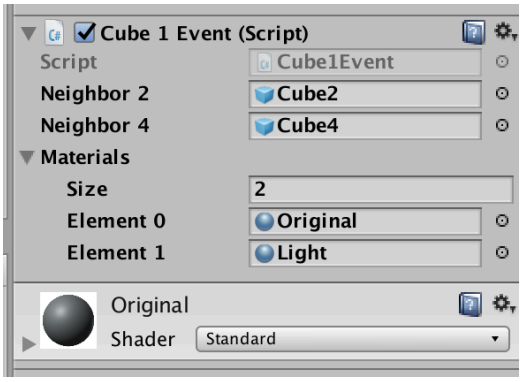
7. Then, I explored the functions of different windows in Unity. As shown in the figure below, the “Scene” window is the game scene editing window, the “Game” window is real game scene you will see when you start game, the “Hierarchy” window displays all the objects in the scene, the “Project” window lists all the supporting files for building this game (user can customize his/her own project files, like Materials, Prefabs, C# files, etc.), and the “Inspector” window gives the details about the current Object.



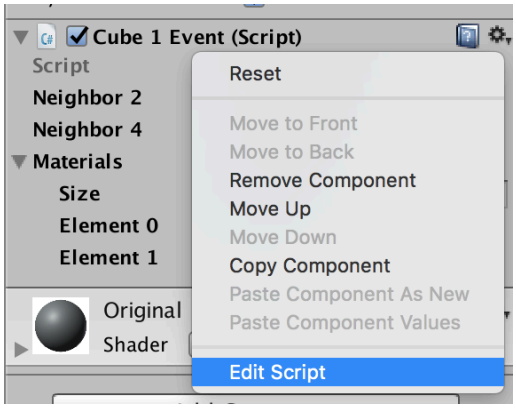
8. After the arrangement of Objects in the Scene, I tried to add code to different Cube Objects, because I guessed that there should be code to detect mouse click and handle the color changing event of different Cube Objects. Unity uses C# script to control Objects and C# script should be added to Object as a component. The steps to add a C# script to current Object is: Add Component in Inspector -> New Script -> Rename the C# script -> Create and Add, as shown in the following figure.



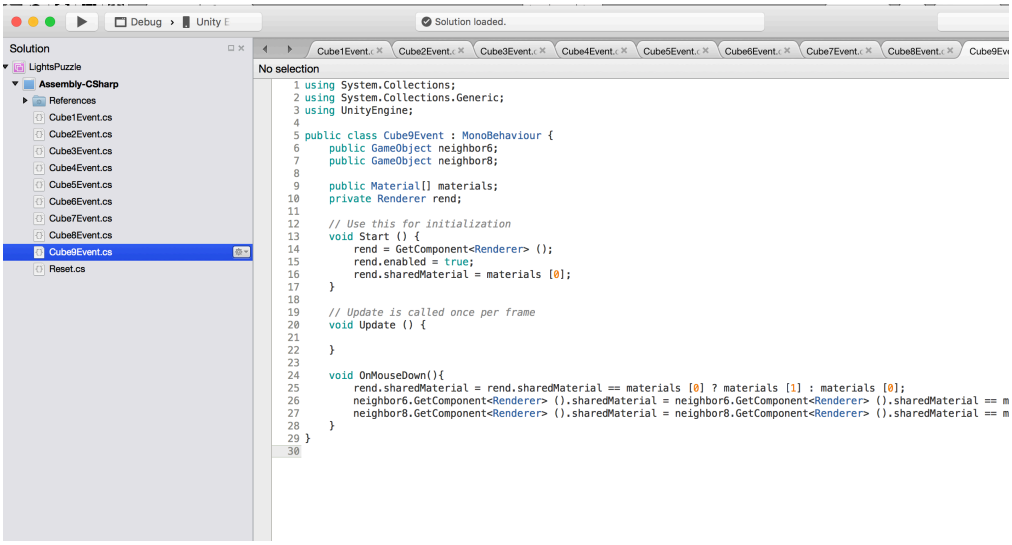
9. After adding a C# script component, we will see this Event Script in Inspector window, as shown in figure below.



Developer can press the right-upper setting gear and choose Edit Script to edit C# script added to Cube Object, as shown in the following figure.



10. After choose Edit Script, we will have a Script Editor showing as below figure. After a serials of trial, I knew that function Start() handles the initialization of current Object, public variable can be shown and set in Inspector window, just like figure above shows, and Update will be called once per frame.



11. In normal lights game, when player presses any object, it and its neighbors of up, down, left and right directions will change color, so I know have to add a function to detect a mouse event for the Cube Object, and handle color changing event. After search online and communicate with Chuanqi Shi, I knew I can use OnMouseDown() function. It will execute after player click on the Cube Object.
12. Then, the problems are how this Object knows its neighbors and how to change color. As I mentioned before that Public variables will be shown in Inspector window, so I set “public GameObject neighbor” in C# script to store one neighbor of the Object, and set the public variable in Inspector window, there are two ways to set the public variable: 1. Directly drag relevant Cube Object into the public variable in Inspector window; 2. Use the setting circle in the rightmost of the public variable.
13. After setting the public variable (neighbors and materials), I can change neighbors’ material by citing them in C# script like the following figure shows.

```
4  void OnMouseDown(){
5      rend.sharedMaterial = rend.sharedMaterial == materials [0] ? materials [1] : materials [0];
6      neighbor6.GetComponent<Renderer> ().sharedMaterial = neighbor6.GetComponent<Renderer> ().sharedMaterial == materials [0]?materials [1] : materials [0];
7      neighbor8.GetComponent<Renderer> ().sharedMaterial = neighbor8.GetComponent<Renderer> ().sharedMaterial == materials [0]?materials [1] : materials [0];
8  }
9
```