

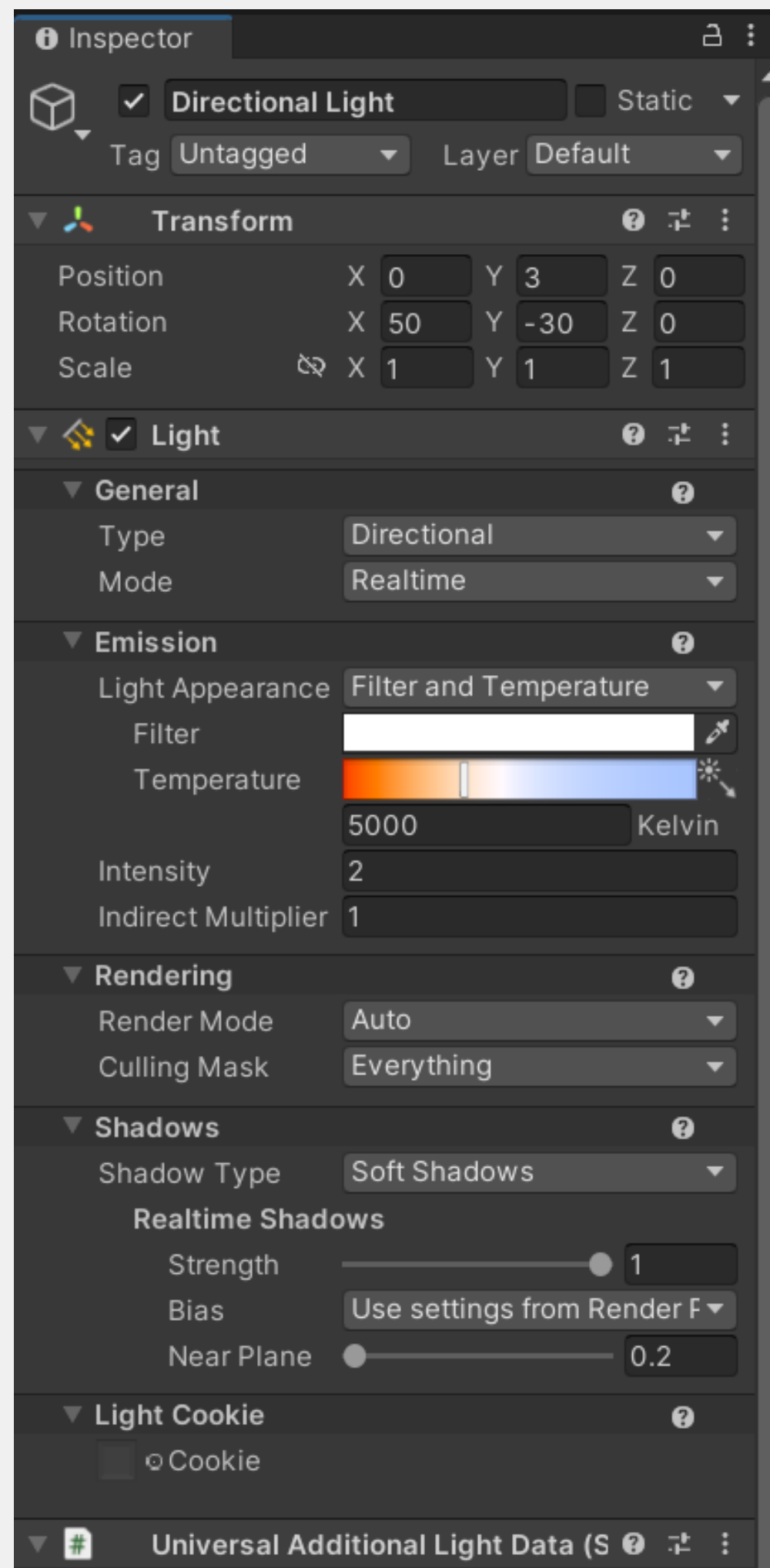
Scripting and UI

Jack Miller and Mieszko Muskala

Day 1 Review

- Game Engines
- Unity Interface
- Cameras, Lights, and Objects
- Scripting in C#

Enabling and Disabling Components



```
1 using UnityEngine;
2 using System.Collections;
3
4 public class LightScript : MonoBehaviour {
5
6     private Light myLight;
7     // Use this for initialization
8     void Start () {
9         myLight = GetComponent<Light> ();
10    }
11
12    // Update is called once per frame
13    void Update () {
14        if(Input.GetKeyUp(KeyCode.Space))
15        {
16            myLight.enabled = !myLight.enabled;
17        }
18    }
19 }
```

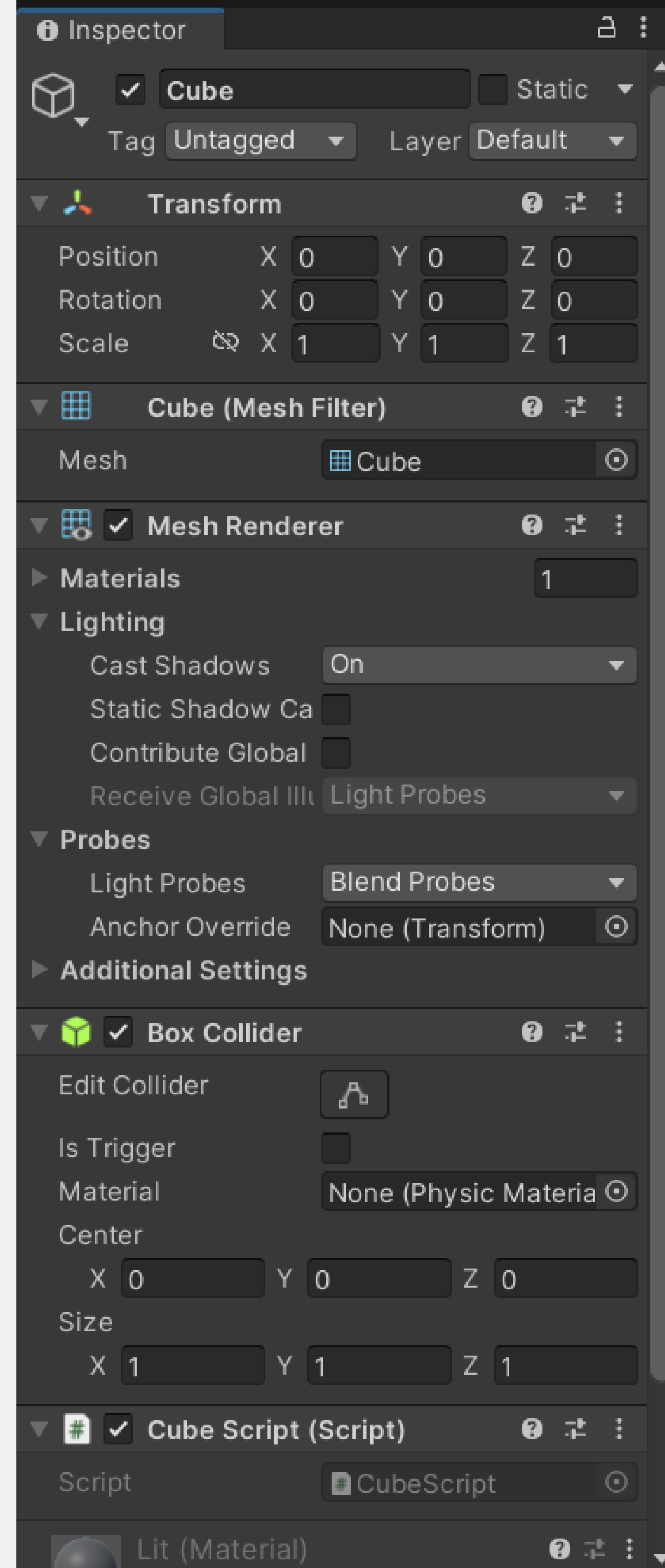
Activating Game Objects

- Making a GameObject inactive will disable every component and turn off any attached renderers, colliders, rigid bodies, scripts, etc...
- Any scripts that you have attached to the GameObject will no longer have Update() called

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class CubeScript : MonoBehaviour {
5
6     // Use this for initialization
7     void Start () {
8
9     }
10
11    // Update is called once per frame
12    void Update () {
13        if(Input.GetKeyUp(KeyCode.Space))
14        {
15            gameObject.SetActive (!gameObject.activeSelf);
16        }
17    }
18 }
```

Getting a Component

- GetComponent<Type>()
- Allows you access to any Component in the object
- You can access Parent and Children too



Calling Other Scripts

- Scripts are GameComponents, so you can use `GetComponent<Type>()` or `FindObjectOfType<Type>()` to obtain a reference to other scripts

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class KeyboardInput : MonoBehaviour {
5
6     private AnimationScript animationScript;
7
8     // Use this for initialization
9     void Start () {
10         animationScript = GetComponent<AnimationScript> ();
11     }
12
13     // Update is called once per frame
14     void Update () {
15         if(Input.GetKeyUp(KeyCode.Space))
16         {
17             animationScript.animate ();
18         }
19     }
20 }
```

```
10
11     // Use this for initialization
12     void Start () {
13         initialPosition = transform.position;
14     }
15
16     // Update is called once per frame
17     void Update () {
18         // Updated the position of the cube
19         updatePosition ();
20     }
21
22     public void animate (){
23         animating = !animating;
24     }
25 }
```

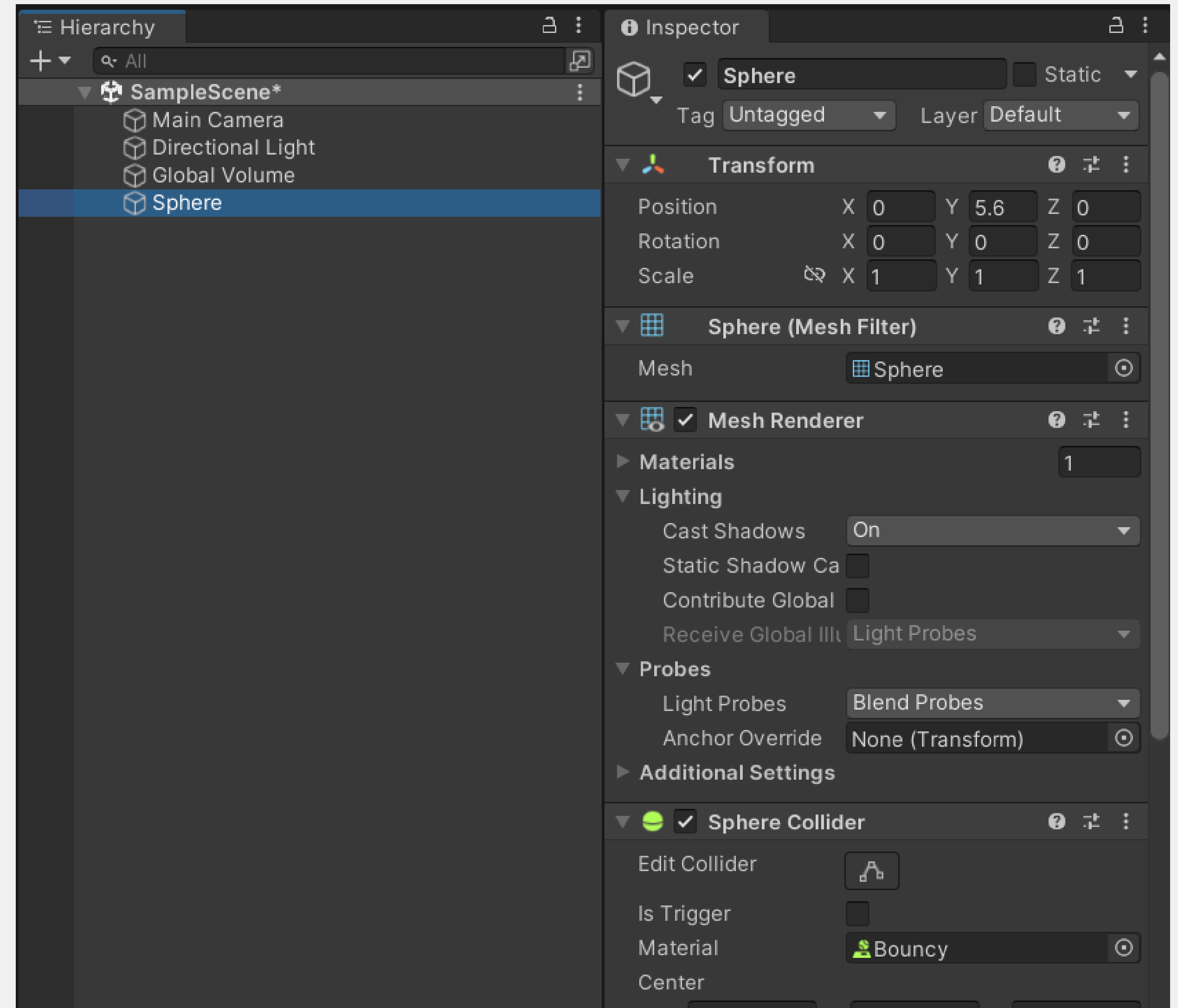
Particle Systems

- Uses a large number of small objects to mimic “fuzzy” phenomena
- Fire, Smoke, Rain, Snow, Clouds, etc.

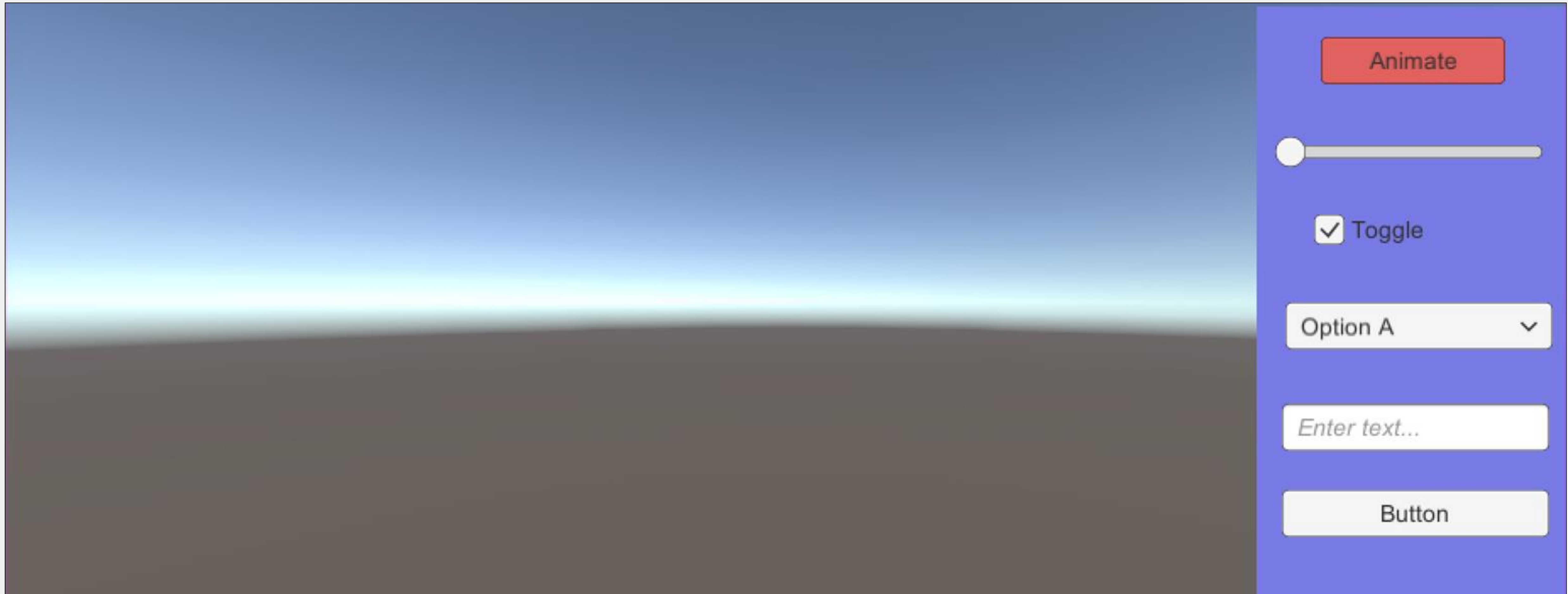


Colliders

- Allows physical interaction between objects
- Colliders react with other colliders
- Can also be used for selecting objects

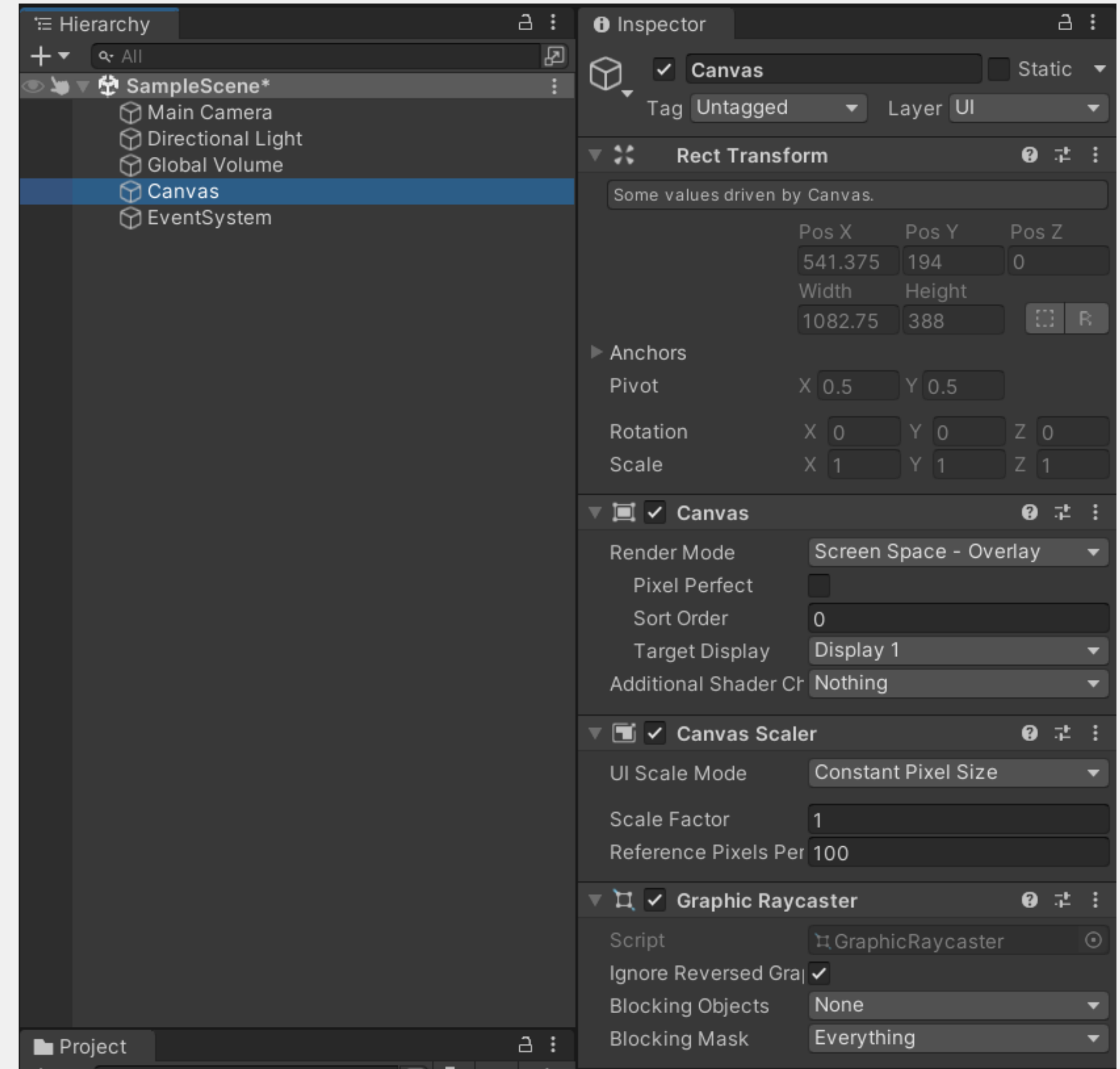


Unity User Interfaces



UI Canvas

- Everything UI starts with the Canvas
- Canvas is a GameObject
- All UI elements must be children of a canvas



UI Text

- Use textmeshpro whenever you need text
- Right-click hierarchy -> UI -> Text-Textmeshpro
- Text properties can be set in the Inspector
- Can be changed during runtime through scripting



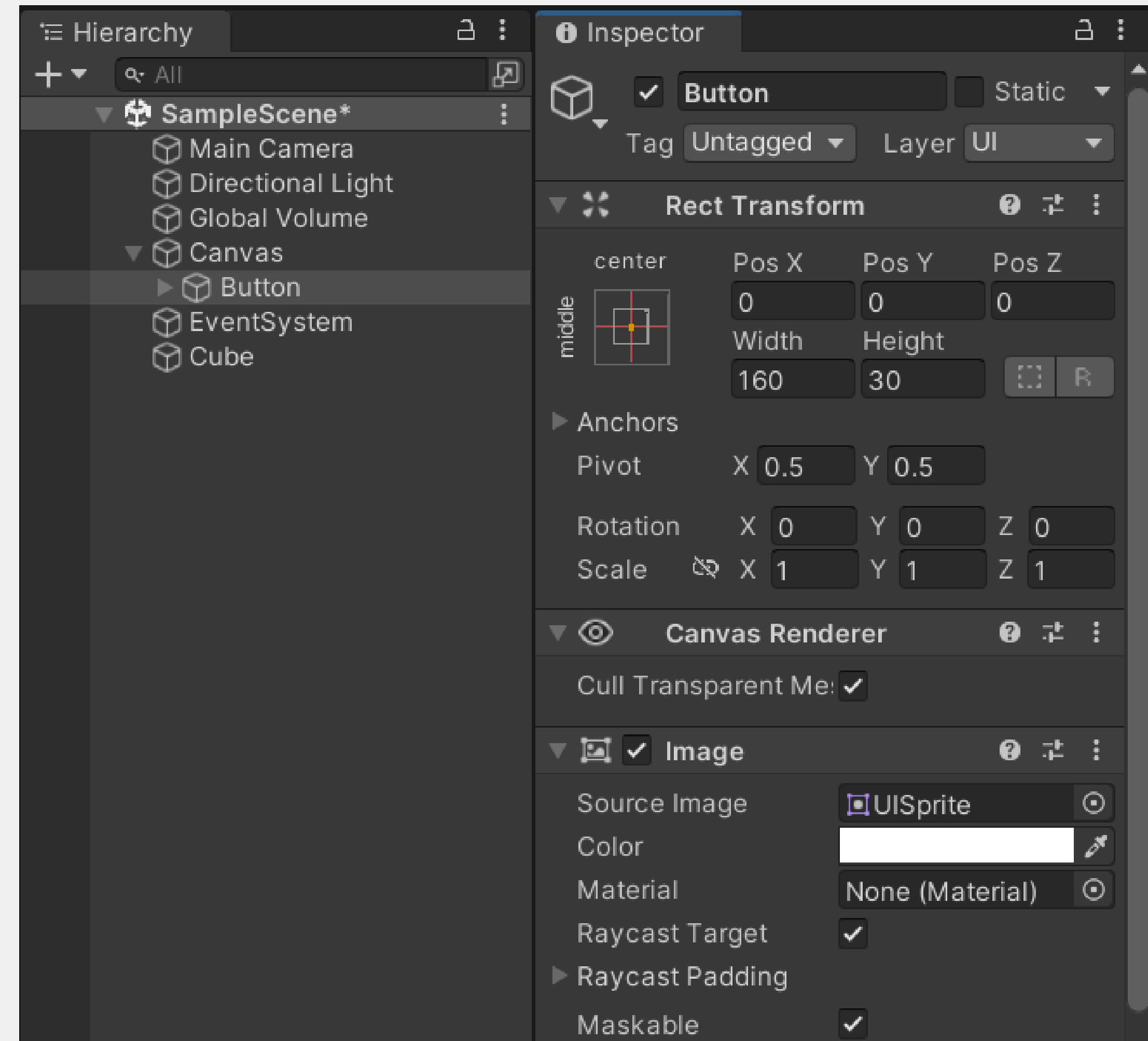
UI Image

- Can be used for almost anything, button, slider, etc.
- When importing an image, you must define what type of texture it is (Normal Map, Light Map, Sprite)
- For UI, we want a Sprite



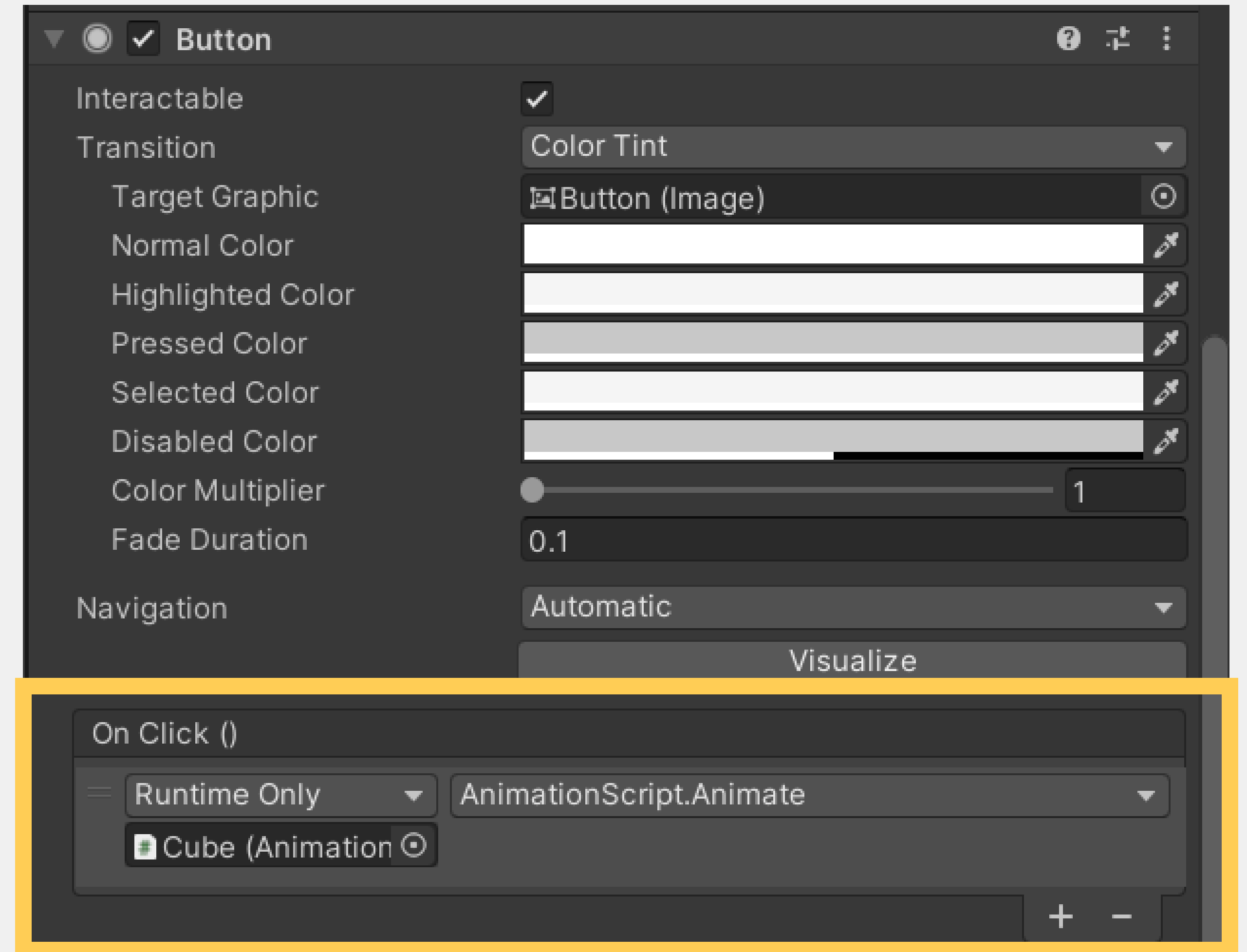
UI Button

- Button is a GameObject that must be a child of a canvas
- Many different options for styling



On Click()

- You can hook up a button to an action through the Inspector
- Chose your GameObject
- Choose your Component
- Choose your Method

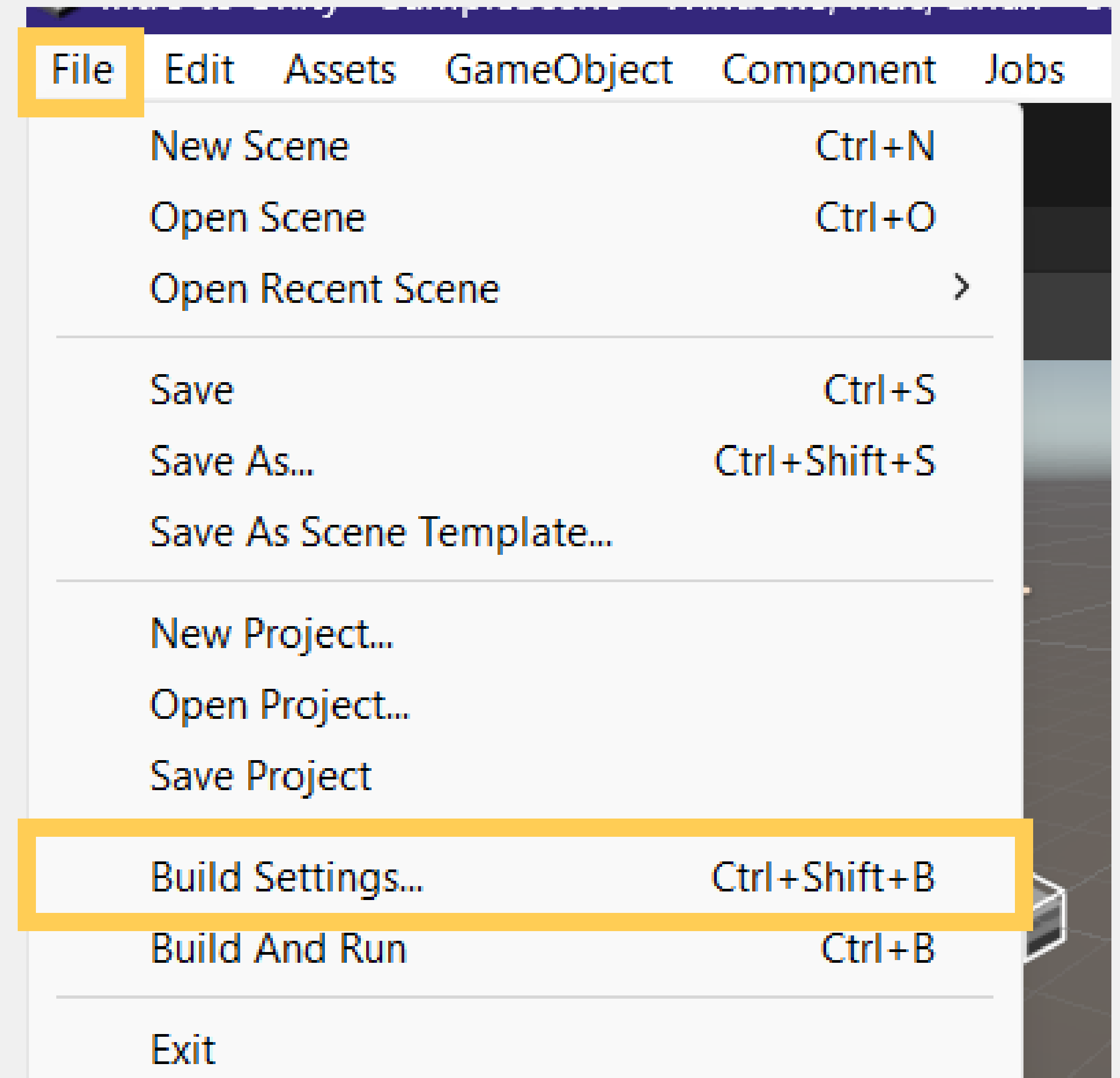


Activity

- Using the same scene
- Play around with the existing UI
- Add new UI elements and functionality

Creating an Executable

- What if I want to create a standalone app?
- Let's make an executable



Creating an Executable

- Add the desired scene
- Select your platform
- Build and Run!

