Day 1 Review

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

• Control game object’s motion using Unity’s physics engine

• Ability to apply gravity to the game objects

• FixedUpdate() is recommended when applying force or controlling Rigidbody settings in a script
Colliders

- Allows physical interaction between objects
  - Rigidbody must attached to at least one game object
- Colliders react with other colliders
- Can also be used for selecting objects
Enabling and Disabling Components

```csharp
using UnityEngine;
using System.Collections;

public class LightScript : MonoBehaviour {

    private Light myLight;
    // Use this for initialization
    void Start () {
        myLight = GetComponent<Light> ();
    }

    // Update is called once per frame
    void Update () {
        if(Input.GetKeyDown(KeyCode.Space)) {
            myLight.enabled = !myLight.enabled;
        }
    }
}
```
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

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

```csharp
using UnityEngine;
using System.Collections;

public class KeyboardInput : MonoBehaviour {

    private AnimationScript animationScript;

    // Use this for initialization
    void Start () {
        animationScript = GetComponent<AnimationScript> ();
    }

    // Update is called once per frame
    void Update () {
        if(Input.GetKeyUp(KeyCode.Space))
        {
            animationScript.animate ();
        }
    }

    public void animate()
    { 
        animating = !animating;
    }
}
```
Activity

• Implement a feature allowing the player to "eat" cubes
Particle Systems

• Uses a large number of small objects to mimic “fuzzy” phenomena

• Fire, Smoke, Rain, Snow, Clouds, etc.
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

• Finish your game with following features:
  • Control the player's movement using the ASWD keys
  • Implement a feature allowing the player to "eat" cubes within the farm
  • Display the number of cubes eaten by the player
  • Include a "restart" button to allow the player to restart the game

• Bonus Features:
  • Enable smooth rotation of the player using mouse input
  • Replace the cube model with a customized model of your choice
  • Show the elapsed time since the player started the game
  • Implement a time limit for the player's gameplay session
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!