Super-short Scripting Basics in Unity

About scripting

- Unity uses C# 2.0 or Javascript
- We'll be going over C# on these slides
  - It has more features
- The rest of these slides assumes you know either C#, Java, or C++

About C#

- C# is a lot like Java and C++
  a. Garbage collected language (no `delete`!)
  b. Almost everything is a pointer
  c. Need to define variable type

C# `struct` and `class`

- C# `class` is the same as Java and C++ classes
  a. Calling `new` will create a new pointer

```
class DemoOne {
    public int x;
    public DemoOne(int p) { x = p; }
}
DemoOne test1 = new DemoOne(1);
DemoOne test2 = test1;
test1.x += 1;
// test1.x is 2, test2.x is 2
```

C# struct and class

- C# struct is treated like a value than a pointer

```
struct DemoTwo {
    public int x;
    public DemoTwo(int p) { x = p; }
}
DemoTwo test1 = new DemoTwo(1);
DemoTwo test2 = test1;
test1.x += 1;
// test1.x is 2, test2.x is 1
```

C# properties

- C# properties are basically getters and setters for a variable

```
struct DemoThree {
    private int x;
    public DemoThree(int p) { x = p; }
    public int TheX {
        get { return x; }
        set { x = value; }
    } // use #1: int y = test1.TheX;
}     // use #2: test1.TheX = 3;
```

C# attributes

- C# attributes adds special properties to a variable, depending on compiler/editor

```
[System.Serializable]
class DemoFour {

    [SerializedField]
    public int x;

    public DemoFour(int p) { x = p; }
}
```

C# Homework

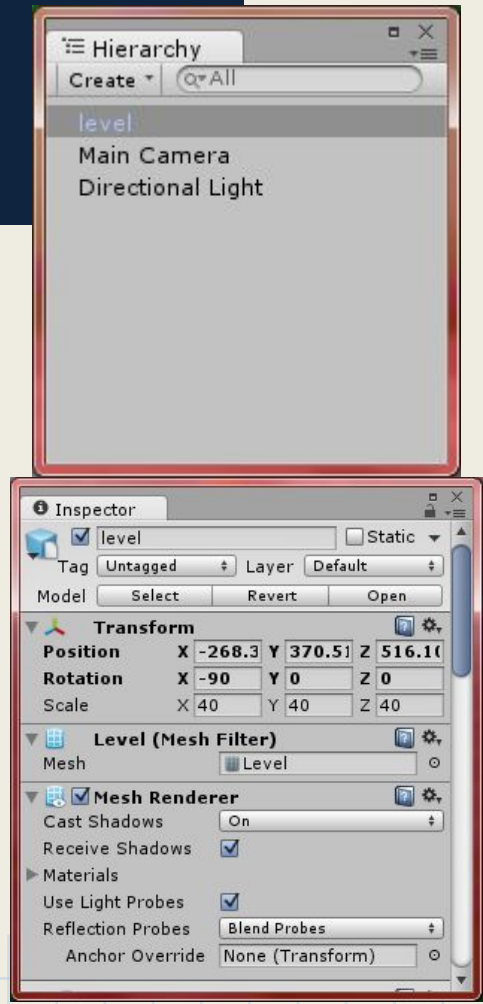
- Some things to look into:
  a. Function parameter modifiers (`ref`, `out`, `params`)
  b. Difference between `const` and `readonly`
  c. Anything imported from "`using System.Collections.Generic`" (`List<T>`, `Dictionary<T>`, `HashSet<T>`)
  d. What is `var`?

Back to Unity scripting

- Scripts in Unity manipulates data held in components for the engine to handle
- Unity’s own graphics, physics, and audio engine reads that data and apply changes to the screen/speakers
- Data is stored using an Object-Component model

Object-Component model

- Scenes contain a list of `GameObject`s
- Each `GameObject` contains a list of `Component`s
- The Hierarchy pane lists all the `GameObject`s
- The Inspector lists a `GameObject`'s `Component`s



Sample code

```
public class MoveTransform : MonoBehaviour {
    [SerializedField]
    private Vector3 moveDirection
    private Transform transformCache;
    private void Start () {
        transformCache = GetComponent<Transform>();
    }
    private void Update() {
        Vector3 position = transformCache.position;
        position += moveDirection * Time.deltaTime;
        transformCache.position = position;
    }
}
```

`MonoBehaviour` base class

- A script extending `MonoBehaviour` (colon is used in C# for extending) turns a script into a special `Component`
- Basically turns a script that can be attached to a `GameObject` in the Hierarchy pane just like any other `Component`

`SerializedField` attribute

- Adding `[SerializedField]` above or next to a member variable declaration exposes that variable to the inspector pane
- In the example code, we turned a `Vector3` (a `struct` containing 3 `float`s: `x`, `y`, and `z`), named `moveDirection` available to the inspector

GetComponent<T>()

- GetComponent<T>() grabs a component attached on the GameObject the script also happens to be attached to.
- MonoBehaviour's read-only public property gameObject and transform corresponds to the GameObject the script is attached to, and its Transform (a component containing position, rotation, and scale) respectively

`Start()` and `Update()` events

- In Unity, events are defined by simply declaring the function with the correct spelling and parameters
- `void Start()` gets called on the frame the game starts
- `void Update()` is called every frame except the frame `Start()` is called
- `void FixedUpdate()` is called every 0.02 seconds
  a. Useful for physics calculation, as it adds consistency
  b. Called much more often than `Update()`

`Time` Properties

- The `Time.deltaTime` read-only property defines how much seconds has passed between frames
  a. Works both in `Update()` and `FixedUpdate()`
- `Time.timeScale` property allows you to slow down or quicken time
- `Time.time` retrieves the number of seconds that passed since `Start()`
  a. This changes proportional to `Time.timeScale`

To review

```
public class MoveTransform : MonoBehaviour {
    [SerializedField]
    private Vector3 moveDirection
    private Transform transformCache;
    private void Start() {
        transformCache = GetComponent<Transform>();
    }
    private void Update() {
        Vector3 position = transformCache.position;
        position += moveDirection * Time.deltaTime;
        transformCache.position = position;
    }
}
```

Other quick notes

- You can print stuff on Unity's console by using `Debug.Log()`

```
Debug.Log("Hello World");
```

Other notes

- To get a reference to a `GameObject` or `Component` different from the one the script is attached to, simply create a serialized variable

```
[SerializedField]
private GameObject someOtherObject;
```

- When the variable is exposed to the Inspector, move the `GameObject` in-interest to the variable

Other notes

- Transform's `rotation` property is a `Quaternion` (`struct` that represents a 1D rotation matrix)
- Using Eular Angles functions are advised

```
Transform.rotation =
Quaternion.Eular(90, 0, 0);

Vector3 angles =
Transform.rotation.eularAngles;
```

Other notes

- Scripts attached to Colliders or Rigidbodies has the following events available to them:
- `void OnCollisionEnter(Collision info)`
- `void OnCollisionExit(Collision info)`
- `void OnCollisionStay(Collision info)`
- `void OnTriggerEnter(Collider other)`
- `void OnTriggerExit(Collider other)`
- `void OnTriggerStay(Collider other)`

Recommended

- Where possible, let Unity do the work! Look into these components and static classes:
  a. AudioSource (sound maker)
  b. All Colliders
  c. Rigidbody
  d. Joints (physics connections)
  e. Animator (component for animations)
  f. Anything from `using UnityEngine.UI;` (or UGUI)
  g. Mathf and its set of float-related functions
  h. Input and its set of input-related functions
  i. Physics.Raycast() function

Google Cardboard

- For Google Cardboard, check documentation on StereoController and its properties
- Also check out GazeInputModule, which works with with UnityEngine.UI, or UGUI

Recommended

- Check the Unify Community for any free scripts
- The Unity Assets Store is another good place to look for scripts and assets, too!