VR Development

Agenda

1. VR Development Overview
2. Unity Design Patterns
3. Reading Discussion

How do we develop VR applications?

VR Development Tools

Code: OpenGL, DirectX
Editors: Unity, Unreal Engine, Lumberyard
Web: WebGL, Three.js, A-Frame
Platforms: Oculus, SteamVR, Cardboard, Daydream

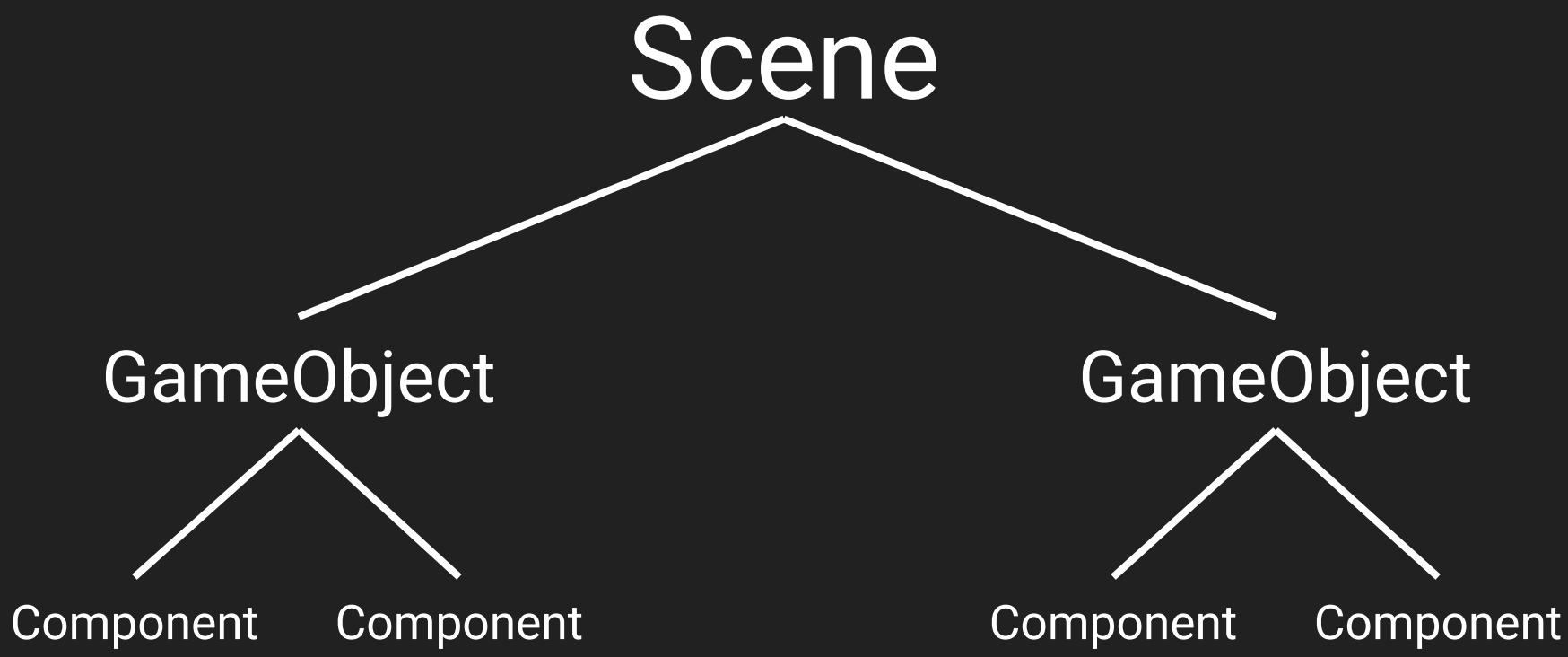
Game Engines

Frameworks providing structure for Game/Simulation development

Provide:

- 3D Rendering, Physics, Sound, Scripting, Animation, Asset Management, Plugins, Event Systems

Unity Basics


Scene
GameObject
GameObject
Component
Component
Component
Component


Hierarchy
Create
All
SampleScene
Main Camera
Directional Light
Parent
Child 1
Child 2
Child 3

Assets > DemoScript1.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class DemoScript1 : MonoBehaviour
6  {
7      // Start is called before the first frame update
8      void Start()
9      {
10
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17     }
18 }
19
```

unity update loop

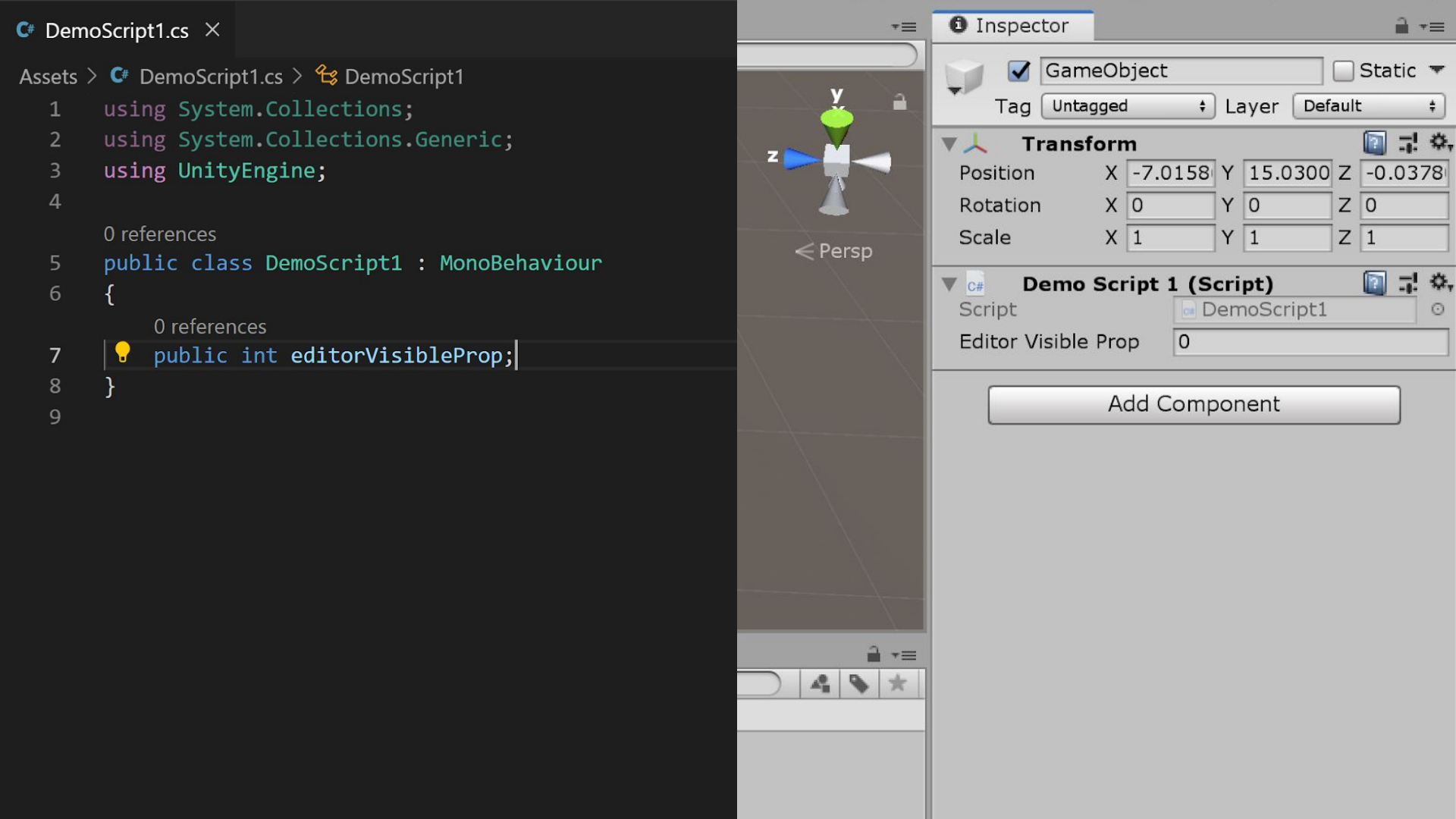
Important Functions

Awake(): Executed once when instantiated
Start(): Executed once just before first update
Update(): Invoked once every frame
FixedUpdate(): Invoked reliably at a consistent rate, maybe more than framerate

And many more

Misc Utility

Debug.Log(“msg”**):** Prints a given message or object to console
Time.?: Global reference for time related values
Time.time, Time.deltaTime, Time.timeSinceLevelLoad


DemoScript1.cs
Assets > DemoScript1.cs > DemoScript1
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
0 references
5 public class DemoScript1 : MonoBehaviour
6 {
0 references
7 public int editorVisibleProp;
8 }
9
y
z
Persp
Inspector
GameObject
Static
Tag
Untagged
Layer
Default
Transform
Position
X -7.0158
Y 15.0300
Z -0.0378
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Demo Script 1 (Script)
Script
DemoScript1
Editor Visible Prop
0
Add Component

Transform

Contains *position, rotation, and scale* values of a GameObject

- **Position:** Vector3 type with *x, y, z* values
- **Rotation:** Quaternion type with *x, y, z, w* values
- **Scale:** Vector3 type with *x, y, z* values

Quaternion

Used to represent all rotations in Unity, quaternions rely on complex numbers. You do not edit the components (x, y, z, w) directly, but apply rotations to it. They help avoid issues such as *Gimble Lock* which can occur when using Euler Angles

Referencing GameObjects

Main Camera: Globally stored as *Camera.main*

Searching the Scene

GameObject.Find(“name”);

GameObject.FindObjectOfType(Type t);

GameObject.FindObjectWithTag(“tag”);

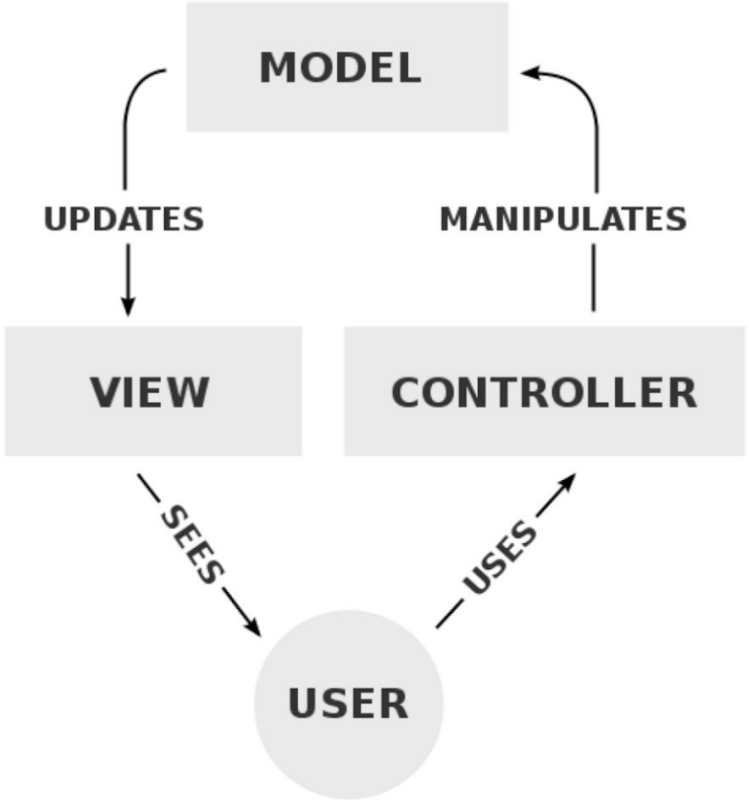
Attaching Through Editor

Create a public variable on a component with the type *GameObject* (or type of component) then drag desired object from Hierarchy onto public field in the inspector

Get Component

gameObject.GetComponent<ComponentName>();

MVC Design Pattern


MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER

Model View Controller (MVC)

Model:

Contains data that represents our application state

View:

What the user sees. Each render is based on the data and also enables user to interactions.

Controller:

The bridge between the model and view. When the user interacts with the view, the controller makes changes to the state and then causes the view to re-render.

Reading 2: VR’s Grand Challenge

What is your biggest barrier to VR immersion?

What was a challenge you found surprising?

In groups, share and discuss the technologies you found

What challenge does your technology address?
Is it compatible with the other technologies in your group, if not how could they be made to work together?

Reading 3: Haptic Feedback in VR

Pick 2 from a selection of novel VR Interface research papers and reflect on them, then envision your own. Include a sketch of a short storyboard for how it will be used *(you will not be graded on artistic talent)*

Due next Tuesday