Create Your Own Virtual Reality

London Lowmanstone

What will you learn?

- How to create a virtual reality app in Unity
- How to upload this app to an iPhone

Setup - bit.ly/vrseminar

- Install Unity (Personal) https://store.unity.com/
- Install MonoDevelop
  (If not installed already - Microsoft Visual Studio will suffice as well)
  http://www.monodevelop.com/download/
- Download Version 0.6 of the Google Cardboard SDK (Source code zip)
  https://github.com/googlevr/gvr-unity-sdk/releases?after=v0.8.0
- If you have an iPhone - Download XCode and get a developer account
- Order any Google Cardboard v2 on Amazon (should be less than $10)
  (This is the only step that should cost money)

What is Virtual Reality?

- Immersive experience
- Many applications
  - Entertainment
  - Job training
  - Remote control
- Starting to take off
  - New devices
  - New audiences

Getting Started - Unity

- Engine for creating worlds
- Easy to build VR apps
- Free

Getting Started - VR

Let’s set up our world for virtual reality!

- Create a new project
- Import the SDK
  - Import Package > Custom Package > CardboardSDKForUnity
- Choose layout
- Add in CardboardMain Prefab
- Delete the Main Camera

Building Our World

Let’s create our virtual reality world!

- Terrain
- Player

Terrain

Let’s make some mountains! (We’ll add color later.)

- GameObject > 3D Object > Terrain
- Make it bigger
  - Edit Width and Length under Terrain Settings (the gear icon)
  - Don’t make it too big
- Draw mountains
  - Use “Raise Terrain” button (mountain with arrow up icon)

Player

Let’s get the player to fly around!

- GameObject > 3D Object > Sphere
  - We’ll attach the camera to this later
- Make it fly
  - Add Component > Rigidbody
    - Makes the ball into a physical object
    - Turn off gravity - we want to fly!
  - Add Component > New Script > Fly

Programming - Fly.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Fly : MonoBehaviour {
	// Use this for initialization
	void Start () {

	}

	// Update is called once per frame
	void Update () {

	}
}
```

Programming - Fly.cs

```
public class Fly : MonoBehaviour {
    // Speed that we want this object to travel at
    public float speed;
    // The object that points in the direction we want to move
    public GameObject pointer;

    // Use this for initialization
    void Start () { }
    ...
}
```

Programming - Fly.cs

```
// Update is called once per frame
void Update () { }

// Make the object fly
void FixedUpdate () {
    Rigidbody rb = GetComponent<Rigidbody> ();
    rb.velocity = speed * pointer.transform.forward;
}
```

Using the Script

- Scripts can be dragged onto any object
- Fill in values for the public variables
  - Speed
    - 10 (just to see what's going on)
  - Pointer
    - "Head" under CardboardMain object

CardboardMain versus Head

- *CardboardMain* is the body that moves
  - We attach motion scripts to CardboardMain
- *Head* is the part that tilts and rotates when the user looks around
  - We use head as the pointer for other objects
- They both work together

Attach Camera to Sphere

- Create a new “Follow” script for CardboardMain (our camera)

```
public class Follow : MonoBehaviour {
    // The object to follow
    GameObject target;

    . . .

    // Use LateUpdate for smoother movements
    void LateUpdate() {
        gameObject.transform.position = target.transform.position;
    }
}
```

First Test

Let’s try it!

- Bump up the speed on the Player object (the ball) to 50
- Hit play!
- Hold down the option key and move your mouse to fly
- Hold down control to tilt (this won’t change the direction you’re travelling)

Organize Your Workspace

- Maintain a good folder structure in your project
  - I like to sort by the types of items
  - It will help you to find things faster when you need them
  - Makes understanding another person’s project much easier

Improvements

- Looks
  - Textures
- Landscape
  - Adding new objects
- Programming
  - Fix collisions
  - Move more smoothly

Textures

- Allow objects to look different
  - Can use any PNG image
- Static and flat
  - Can look 3D with a special “normal” PNG
- Allows you to paint the mountains
- Can download from online

Unity Store

- Find assets for free
- Choose what you need from each package
  - Limits import time and installation size
- Find some textures you like

Paint the Mountains

- Terrain > Paint Texture (the paintbrush icon) > Edit Textures
  - Drag the texture PNG into the RGB box and the Normal into Normal box
- Add more textures
- Make the mountains look however you want

Landscape - Adding Objects

- Unity can display FBX objects (.fbx, “Filmbox”)
- You can find these online
  - Drag them into your project’s “assets” folder in your computer’s filesystem to use them
- I’m going to add a castle
  - https://free3d.com/3d-model/fantasy-castle-40715.html
- “Generate Colliders” on FBX object

Programming - Movement

- Quick Collision Fix
  - Increase the radius of the ball
    - 50 works well
- Long-term Solution
  - Unity doesn’t like directly setting velocity
  - Use force instead
  - Better physics

Programming - Jedi (Using the Force)

```
public float maxSpeed; // should be set around 50
public float acceleration; // should be set around 10000
public GameObject pointer; // should be CardboardMain > Head object

void FixedUpdate () {
        Rigidbody rb = GetComponent<Rigidbody> ();
        rb.AddForce (acceleration * pointer.transform.forward * Time.deltaTime);
        rb.velocity = Vector3.ClampMagnitude (rb.velocity, maxSpeed);
}
```

Colliders

- Define where objects touch
- Define what happens when objects touch
- Can be triggers
  - No physical action
  - Act as sensors

Physic Materials

Let’s make our mountains bouncy!

- How objects act when they touch
  - Dynamic Friction
  - Static Friction
  - Bounciness
- Right click in Project tab > Create > Physic Material
- Drag onto the Collider component of the mountain terrain

Congrats!

You just built your first virtual reality world!

Now we just need to upload it to your phone...

Xcode

- Used to make apps for iOS
- Unity exports to XCode
  - This process is called “Building”

Building the Project

- File > Build Settings
- Switch Platform to iOS
- Player Settings (beneath “platform”)
  - Company and Product Name
    - Doesn’t really matter
  - Bundle Identifier
    - Fill in with company and product name from above
  - Orientation - Landscape Left
  - Can add an icon for the app
- DO NOT CHECK “Virtual Reality Supported”
  - This is only for newer versions of the SDK
- Click “Build” - name the folder and click “Save”

Uploading to Your Phone

- Open the entire folder in XCode
  - Not just the XCode project inside
  - Newer versions of Google Cardboard SDK have issues if you don't
- Unity-iPhone > General > Signing > Choose your team
- Add "Security.framework" to Linked Frameworks and Libraries
  - Beneath "Signing"
- Build-Settings > Enable Bitcode > No
  - This will not allow you to publish your app to the app store
  - I have yet to find a workaround (If you find one, let me know)
- Press the "Play" button in the top left to build and run the app

Extra Things

- Google Cardboard has a button
  - Boolean “Cardboard.SDK.Triggered”
- CardboardMain has options to remove display elements
  - Back button, settings button, alignment marker
- Displaying text in VR
  - Change “Screen Size” under CardboardMain > Emulation Settings
- Sounds
- Multiple scenes
  - Wind and Trees

Enjoy!

- Build some amazing worlds
- Share them with friends
- Encourage them to create their own!