“Hello VR World!” (V0.9)

Samsung Oculus Gear VR/ Windows 10/Unity 3D/Visual Studio 2015 Development Edition

By Dan Lesser

After a brief introduction to VR through Google Cardboard and 3-D video apps like *Vrse*, I had only one thing to say – “Sold!” This year, 2016, I am firmly convinced is not only the Year of the Monkey, but the Year of VR Goggle-face from poorly-adjusted VR headsets creating a goggle imprint on people’s faces. So two months ago, I plunked-down and bought a Gear VR headset / Galaxy Note 5 mobile phone combo so as to dive head first into learning how to create immersive 3-D applications, and how to lie convincingly that my own VR-face was due to hitting the powder skiing rather than playing *Smash Hit* too long...

My professional programming background is generally in .NET business applications, however, and so for me, there is an experience gap between the skills needed to create a first-generation consumer VR application and past technical talents honed to pay the bills for the privilege of being lucky-ducky to have a first generation, consumer VR device in the first place. I imagine that is the case for any number of fellow flatland developers taking their first, virtual baby steps (except for you fun-loving game devs and roboticist cyberpunks out there, *salud*! :-). Now given that I’m not a VR sensei by any means, closer to a VR white belt, my intent here is to create a tutorial that is the VR equivalent of “Hello World” – using the Unity3D Game Engine with Visual Studio 2015 on Windows 10.

1. Voxel-izing a Proud Teletype Terminal Coding Tradition

Once you’ve finished, “Hello VR World” for Gear VR will contain a 3-D, stereoscopic world (which for once is truly what you make of it) and an animated character who waves at the World and says “Hello VR World” when you gaze at her and tap on the Gear VR touchpad. In addition, you will learn how to create a multiverse of worlds like your first world, simply by gazing at the ground and then tapping on the Gear VR touchpad and *ex nihilo* generating, with apologies to Disney’s Aladdin, “A Whole New World.”

The venerable “Hello World” tradition cannot yet be a one-liner for VR. I don’t think we’ll have the one-liner “Hello VR World” until “reticle” or a better neologism is found for displaying a visual line of sight indicator within a HMD (head-mounted display) and these terms become as common place in the *google-franca* as a "mouse cursor" or "desktop." (Re-purpose "jeeper" anyone? Usage, circa 2023: "Point your jeeper at the dwindling bitcoins in your holodeck vault and blink your peepers twice to begin the retinal scan that submits your tax holo-data to the Internal Revenue Service..."). Mostly this won't be a one-liner, though, because a "Hello VR World" needs to first create a simulated VR world to say "Hello" to.

The following tutorial assumes:

- Windows Development Environment (Recommend Windows 10)
- C# or Similar Programming Background
- Samsung Gear VR 1.0 Hardware with a compatible mobile device such as Galaxy 6 or Galaxy Note 5
- You are new to VR and Unity both. If you already have some grounding in Unity but no Gear VR experience, or take your tutorial shaken not stirred with digressions, I recommend you instead try the following Oculus tutorial: "Tutorial: Build a Simple VR Unity Game". In addition, I was helped immensely in getting started with "Guide to building GearVR projects in Unity 5" by Matt Ratcliffe (props and thanks), who has additional tips for wireless debugging once you are up and running.
- Cartesian as an adjective to you does not mean "of/or pertaining to a shopping cart" but rather a x/y/z 3-D coordinate system (and for double-bonus Jeopardy points Descartes)
- Speaking of the theme of existential, "Matrix" philosophers, you remember Zhuangzi's wise words: "**Now I do not know whether I was then a man dreaming I was a butterfly, or whether I am now a butterfly, dreaming I am a man. *Or whether I was a butterfly wearing a proboscis mounted, ultra high-def graphical display with stereo surround sound who lived in my parent's basement and didn't get out and flutter-socially all that much."*** (First part attributed to Lin Yutang's translation- thanks Lin / Wikipedia.)

2. Tools of the VR Trade:

The following applications/SDKs/accounts are used in the course of this tutorial:

Applications/SDKs
Chocolatey v0.9.9.11
PowerShell 5
Unity 5.3.1
Visual Studio 2015 Community Edition
Visual Studio Tools for Unity 2.1
Android Studio 1.5.2
Android SDK v19+
GitHub
Oculus Utilities for Unity
Adobe Fuse CC / Mixamo *

Accounts Needed/Will Create:

- Unity Account
- Oculus Developer Account
- Github
- AdobeID *

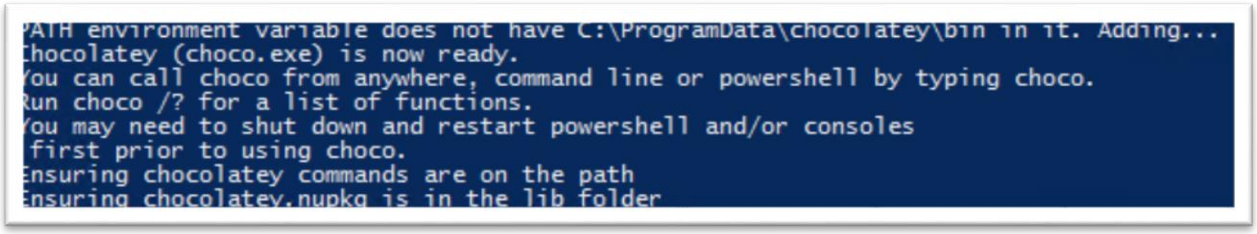
* optional, if you would like to generate your own animated character/animations, please see appendix…

If you do not already have the right components installed, or are missing some, follow the next two steps to get with the program already and install the pre-requisites using a package manager.

2.1 “Hello Chocolatey!”

If you want to avoid a time-wasting download/link following marathon, and provision new development machines lickety-split, I will assume for this tutorial that you already have (in which case skip to the next step) or will otherwise first install the Chocolatey download manager.

- Open the PowerShell command line as Administrator
- If you don’t know whether you can execute remotely signed PowerShell scripts, run the following command: **Get-ExecutionPolicy**
- If the Execution Policy returned is not **RemoteSigned** or **Unrestricted**, run the following command:
    - **Set-ExecutionPolicy RemoteSigned**
- Per https://chocolatey.org/, run the following command to download and execute the remote Chocolatey script:
- iex ((new-object net.webclient).DownloadString('https://chocolatey.org/install.ps1'))
- If you see the message **“Chocolatey (choco.exe) is now ready”** after the Invoke-Expression (**iex**) command completes, go ahead and close the PowerShell command line window. (If you see a red error message like: “chocolateyInstall.ps1 cannot be loaded because running scripts is disabled on this system.” Try to run as Administrator again, step #3 and then re-run step #4…)



```
PATH environment variable does not have C:\ProgramData\chocolatey\bin in it. Adding...
Chocolatey (choco.exe) is now ready.
You can call choco from anywhere, command line or powershell by typing choco.
Run choco /? for a list of functions.
You may need to shut down and restart powershell and/or consoles
 first prior to using choco.
Ensuring chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder
```

2.2 “choco install” Away

Once you have installed Chocolatey, running the PowerShell command line as Administrator, install the pre-requisites with the following commands (**omitting any that you have already installed**). I like to add the **-y** flag at the end to avoid getting prompted for confirmation on each package, but it is optional. Note that depending on your bandwidth and space availability, this could take several hours and use 15+ GB space since not including installation arguments/answer files to remove unnecessary components for the Hello World example:

PS C:\> choco install androidstudio android-sdk github visualstudio2015community unity -y

So, thanks Chocolatey, I’ll leave you to it, a coffee break or entire “Walking Dead” Season 6 it is… until I get Gig-internet.

2.3 Non-Chocolatey Pre-requisites

Unfortunately, not everything can be installed with Chocolatey (or PowerShell 5's Install-Package) yet so there are a few additional pre-requisite installation steps, assuming you successfully got the first items installed.

2.3.1 Android SDKs

From the Windows Start Menu > Android SDK Tools > open the Android SDK Manager, and add API 22 for Android 5.1 compatibility, Build Tools version 22. (If you have a different version of Android with Gear VR, be sure to select any additional compatible APIs you wish to use.) Install and accept the license.

2.3.2 Unity Download Assistant

Download the Unity Download Assistant:

http://unity3d.com/get-unity/download?ref=personal

The download assistant will allow you to install optional components for Unity (not currently available from choco). For this tutorial, you will minimally need to download:

- Standard Assets
- Microsoft Visual Studio Tools for Unity
- Windows Build Support
- Android Build Support

2.3.3 Reboot and Initialize Visual Studio

You will most likely need to reboot after the last step. I recommend initializing Visual Studio if you have just installed it in an earlier step.

2.4 Dev-readying your Android Phone

To debug and deploy your application to your phone, you will need to enable developer mode for Android and Gear VR Service developer mode:

- In Settings > About Device > Builder number, tap on the Build number until you get a message that Developer mode is enabled and that you are special and Google loves you, yada.
- Go back one screen and tap on "Developer Options" > toggling on "Developer Options", toggling on "USB debugging" and possibly toggling off "Verify apps via USB", (or be

prepared to have to restart your app once after each new build if the malware scan kills your app)

- In Settings > Applications > "Gear VR Service" > "More", tap on the "VR Service Version" until you see the Developer options show below that version. Toggle on "Developer mode" and notice the flicker that appears on your screen, as it renders as it does in VR mode aside from lacking stereoscopic elements in non-VR apps. You will need to turn this on before any deployments of new builds to your device. Toggle on "Show icon on the launcher" to make it more convenient to find Developer mode and toggle it on and off.

2.5 Git Clone HelloVRWorld

So now that you have the magic wands of VR sorcery within reach, it's time to get some raw materials to cast your holographic spells:

Open the PowerShell command line, and clone the **hellovrworld** repository with git source control (installed with Chocolatey):

PS> git clone https://github.com/RemedialRobotics/hellovrworld.git

Under default GitHub for Windows Settings, this would clone the repository to following file system path:

PS> cd (Join-Path $env:USERPROFILE -ChildPath "Documents\GitHub\hellovrworld\")

**e.g. cd C:\Users\Dan\Documents\GitHub\hellovrworld\*

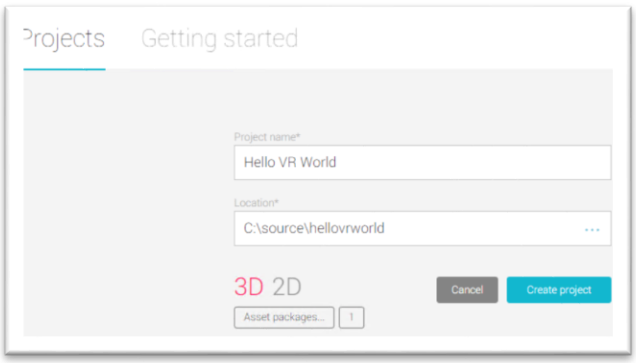
The **hellovrworld** repository will contain two folders:

- **Hello VR World** - the (mostly) complete Unity Project, (minus your own osig required to run)
- **StarterFiles** – containing some basic planet texture .tiffs courtesy of NASA, three .fbx 3D model/animation files and one .mp3 audio file that you will import into your unity project in the course of this tutorial

3. VR Genesis

Open Unity. The first time you open Unity, you will need to register a Unity account, sign in, and select Personal Edition (free if your organization has less than $100K in annual revenue like my money-burning, side hobby…uh bootstrapped, up-and-coming start-up).

Enter the Project name: "Hello VR World," and create a local source repository path with a folder for your project. Make sure 3D is selected and then select "Asset packages…" and add Visual Studio 2015 Tools and finally, click on the "Create project" button.



Project name*: Hello VR World
Location*: C:\source\hellovrworld

For a quick introduction to using the Unity Editor effectively, I recommend reviewing the commonly used Unity Screens in "Learning the Interface."

Once the Unity Editor loads, the first step for a new Unity project is to create a new subfolder called "_Scenes" in the "Project View" in the lower left pane of Unity in the default layout. Press CTRL+S and save your first Scene called "MainScene" in the _Scenes folder.

3.1 In the Beginning, there was the Sphere…

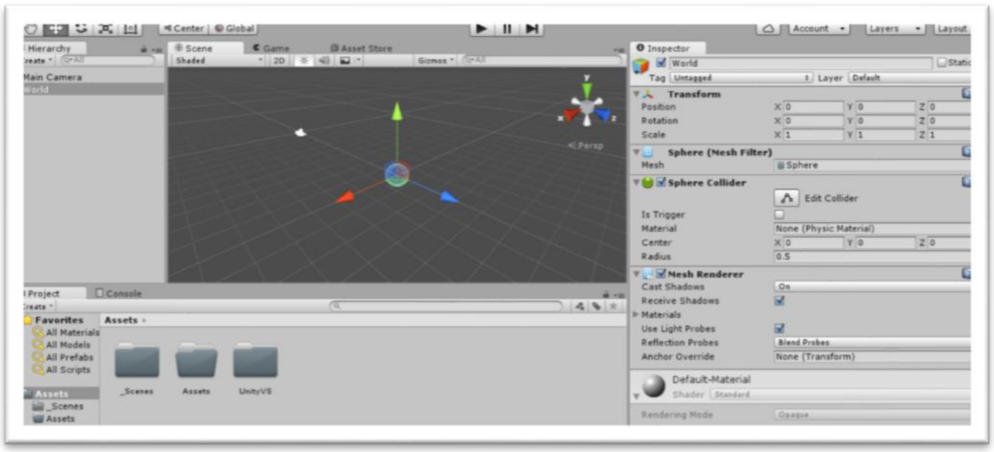
Within each scene in Unity, is a hierarchy of Game Objects appearing in the Scene, by default displayed at left in the "Hierarchy View."

In the "Hierarchy Menu" > Select "Create" > "3-D Object" > "Sphere", to commence planet-making sequence in 5…4…3.

Above the "Scene View", toggle the "2D" button to change the Scene View to display a 3D perspective.

In the "Inspector View", making sure that the Sphere remains selected, change "Sphere", the default name of our new Game Object and rename it to "World." Check that the position of the Sphere is at the origin "Position" coordinates of (0,5,0). Note that if you are more familiar with web development and thinking of a vertical layer as a "z-index," keep in mind that in Unity that the zed is in the horizontal plane and the y-axis is vertical. Change the "Scale" to (5, 5, 5). If a Game Object does not have any expected motion, you can check "Static" to help improve your app's performance – but not in this case.

Now, it's time to get terraforming in earnest. *Sheol*, even if you have no graphic artist skills, we can at least try to match Star Trek II: Wrath of Khaaaaan! in our FX efforts thanks to Unity3D. (Or more likely, find professional graphic designs and animations you're looking for in Unity3D's Asset Store…a key Unity feature for small Indy game developers and no small part of its popularity as a Game Engine.)



World
Tag Untagged Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Sphere (Mesh Filter)
Mesh Sphere

Sphere Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 0 Y 0 Z 0
Radius 0.5

Mesh Renderer
Cast Shadows On
Receive Shadows ✓
Materials
Use Light Probes ✓
Reflection Probes Blend Probes
Anchor Override None (Transform)

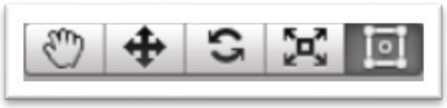
Default-Material
Shader Standard
Rendering Mode Opaque

The surface of our planet is a boring Default-Material. This will not do, mainly because we won't be able to see our planet's rotation clearly otherwise. Let's renovate this charming, fixer-upper geometry and slap on a fresh coat of geological eons. NASA provides public domain images we can use to create our planetary surface. You can copy **earthy.tif** (or a handful of planets) from the **hellovrworld** git repository you cloned earlier from the **StarterFiles** folder. Or if you can find an alternate planetary surface texture here: http://nasa3d.arc.nasa.gov/.

Once you have your chosen surface, create a new folder in "Project View" Assets called "Textures" and then drag and drop the **.tif** of your choice into the newly created folder from Windows Explorer. Now in the "Project" View, create a folder called "Materials", and then "Create" > "New Material" naming your material for your planet. Selecting the material in "Inspector View," select the circle icon next to Albedo and select your texture that you copied into the "Textures" folder.

Selecting the "World" in the Hierarchy View, the material now becomes selectable in the "Inspector View" > "Mesh Renderer" component, expand the arrow to the left of the "Materials" field and select the new material you created (selecting the target icon to the right of "Materials" > "Element 0"). (Or simply drag the material onto the Sphere within Scene View to apply it.)

3.2 Here Comes the Sun…

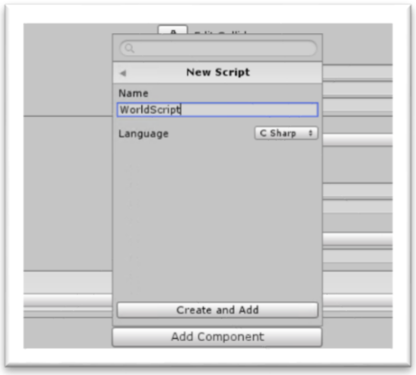


In the "Hierarchy view", select the "Directional Light". Unity includes a directional light by default in new scenes. In "Inspector View", Rename the "Directional Light" to "Sun." You can change the location of the light, by setting the (x,y,z) Position coordinates or selecting the Sun in the "Hierarchy View" and position icon in the Scene View toolbar and then manipulating the light icon in the scene view so that it shines on the planet as you like it (after positioning selecting Rotation to alter the angle of incidence). **Take away:** you can both manipulate a Game Object's initial starting position, rotation ("orientation") and scaling from the "Inspector View" > "Transform" parameters, or from using the graphical drag and drop toolbar within "Scene View."

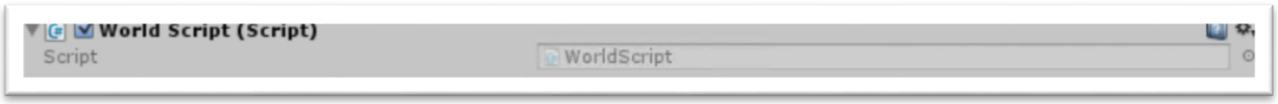
3.3 Give it a Whorl…

We have already have a default directional light. We have camera (by default). We now need action…which in Unity is provided through the use of scripts.

Select the World, either in "Scene View" or "Hierarchy View." In the "Inspector View," click on the "Add Component" button and select "New Script", changing the name to "World" and keeping "C Sharp" selected (the alternative being JavaScript). Press "Create and Add."



You can now click on the World filename in the Script field to open the World **MonoBehavior** subclass in Visual Studio 2015. Note the check box to the left of each component, you can uncheck it to disable the Component temporarily during development or while previewing your scene in "Game View." (Additionally, Game Objects have a checkbox that can temporarily remove them and their children Game Objects from the Scene when unchecked.)



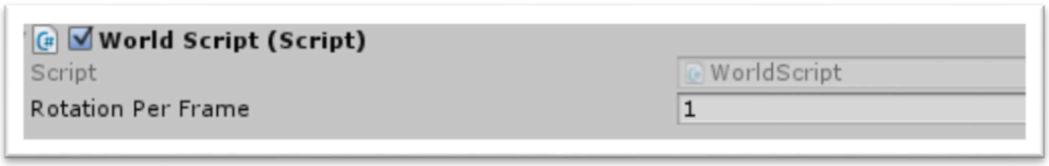
Each active Game Object in Unity3D that appears in the Hierarchy View has associated coordinates in the space within the scene or locally relative to a parent Game Object, and have common methods associated with motion. By adding components to the Game Object, you can additively alter the physics emulation and behavior of an object, the display and rendering of an object, sound associated with an object or its interactions with other objects or inputs. To make the World rotate, you will use the transform property of the Game Object and call the Rotate method. Contrary to common static C# APIs, many of the event-related methods that you add to your script classes are not overriding the base classes' implementations or abstract methods, or implementing particular interfaces.

They instead consist of well-known methods that are called by the Game Engine on a game event loop, that is at the beginning of a Scene (with Awake () and Start()) or frame by frame/different timings depending on the method variant (e.g. Update(), FixedUpdate(), LateUpdate()). The most common two methods are "Start", which you use much like a constructor in initializing the Game Object and "Update" which is called during each frame by the Game Loop.

Note that we can add public fields to our script classes that are then settable in Script components within the Unity Editor as parameters – most often **float** or **int** value types or references to other Game Objects in the scene that need to interact with the current Game Object. Once I add:

```
public float rotationPerFrame;
```

The component for my World Script is updated in the Unity Editor to allow me to set a parameter that can alter how much the Game Object will rotate per frame (note how it converts a camel-cased field name to a readable Editor name):



World Script (Script)	
Script	WorldScript
Rotation Per Frame	1

```
using UnityEngine;
using System.Collections;

public class World : MonoBehaviour {

       //public value fields show as configurable properties in the Unity3D Inspector
       //Window

       public float rotationPerFrame=1f;

       // Use this for initialization
       void Start () {

       }

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

                     transform.Rotate(0f, rotationPerFrame,0f);
       }
}
```

You can preview what this will do in VR on the “Game View” by pressing the Play button after selecting the “Game View” tab. Observe that the Planet now continually rotates while the game loop is running when rotationPerFrame is non-zero. (Note that you must stop the Game View before making other changes if you want the changes to be saved, although helpfully you can modify parameters in “Inspector View,” while playing in Game View in order to experiment/preview changes before making them more permanent). For kicks, experiment with altering the position of the planet over “Time.deltaTime” or with FixedUpdate() so that Kepler is proud of your “smooth, orbital moves.” (Just don’t ask me how to get ellipsoidal…per Clint Eastwood, “A Tutorial’s Got to Know its Limitations.”)

4. Build

We can preview our scene to our heart's content in Game View, but I promised VR on your Android, and I shall deliver after some wee, additional plumbing steps.

4.1. Get an OSIG File to Use the Oculus API

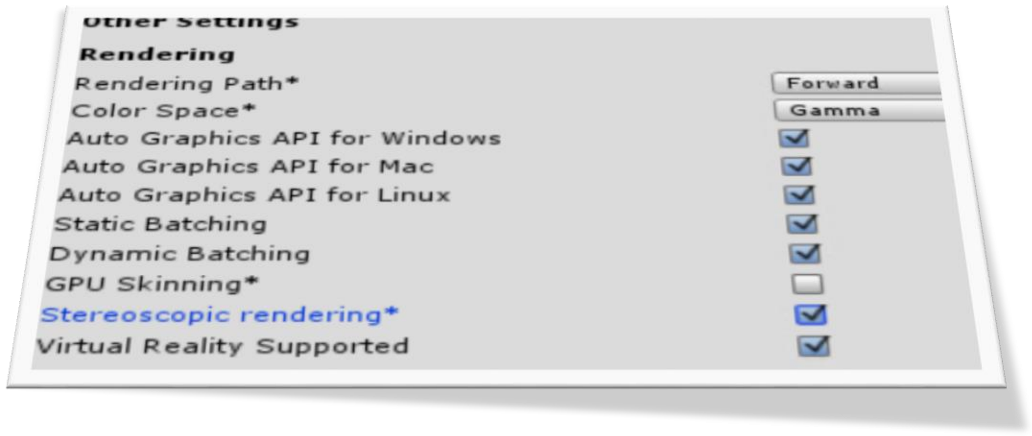
Unless you happen to like getting an app permission smack down when you try to first run "Hello VR World" on your Gear VR ("Thread priority security exception. Make sure the APK is signed"), you will need to add a signed "hall pass" to your app. Within "Project View" > "Assets", add a folder called "Plugins" > add a sub-folder below that called "Android" > and then finally add a sub-folder below that called "assets."

You will need to register a developer account with Oculus and create an osig file (named something like *oculussig_<h3xad3cimal m0nstr0s1ty here>)* that you will copy and paste into Assets\Plugins\Android\assets\, following the instructions for Unity on the osig page:

https://developer.oculus.com/osig/

4.2 APK

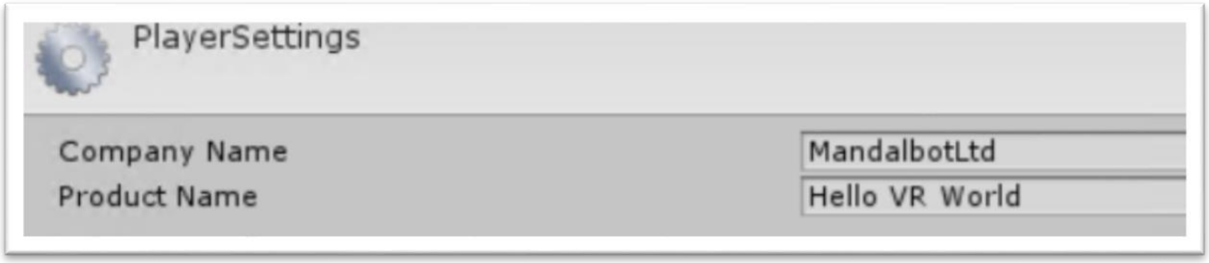
To prepare our app for deployment go to "Edit" > "Project Settings" > "Player Settings", and select the Android icon. For the purposes of building for the Oculus Mobile SDK we need to target the Android SDK, version 19 or greater. For Gear VR, I currently target Version 22 (Android 5.1). To enable VR, go to "Edit" > "Project Settings" > "Player Settings" > "Other Settings" and check the "Stereoscopic Rendering" and "Virtual Reality Supported" options. Since the introduction of Unity 5, there is now inherit basic support for VR rendering without necessarily having to install a headset-specific SDK (although that remains a practical necessity if you plan to distribute your application).



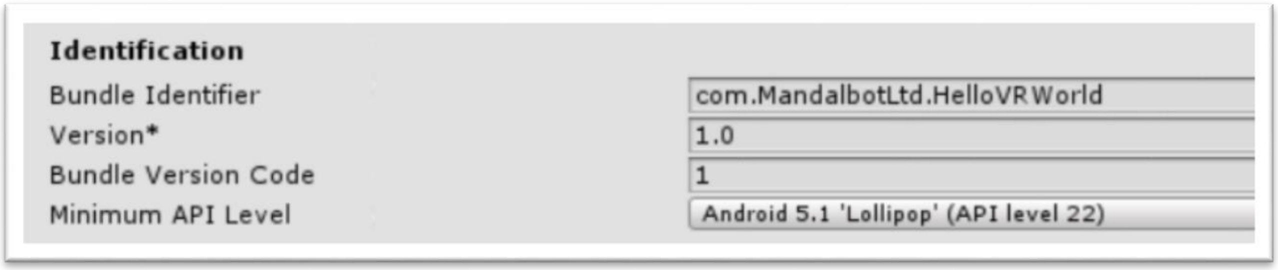
Other Settings	
Rendering	
Rendering Path*	Forward
Color Space*	Gamma
Auto Graphics API for Windows	✓
Auto Graphics API for Mac	✓
Auto Graphics API for Linux	✓
Static Batching	✓
Dynamic Batching	✓
GPU Skinning*	☐
Stereoscopic rendering*	✓
Virtual Reality Supported	✓

To avoid a build error when building to Android, make sure that the Company Name and Product Name in the "Player Settings" corresponds to the Android bundle identifier of com.<Company Name(Minus

Spaces)> >.<Product Name (Minus Spaces)>, e.g. com.AcmeAnvil.HelloVRWorld, and "Acme Anvil" / "Hello VR World."

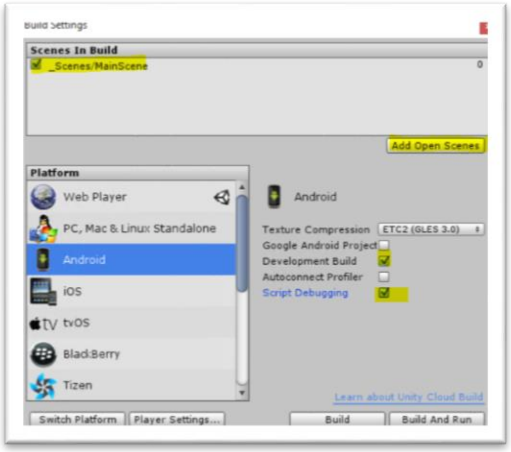

PlayerSettings

Company Name	MandalbotLtd
Product Name	Hello VR World


Identification

Bundle Identifier	com.MandalbotLtd.HelloVRWorld
Version*	1.0
Bundle Version Code	1
Minimum API Level	Android 5.1 'Lollipop' (API level 22)

After you save those changes, open "Build Settings" and select "Add Open Scenes" to add the current scene to the build.   Select Android as the current build target. "Textures are ideally stored with 4 bits per texel in ETC2 format for improved rendering performance and an 8x storage space reduction over 32-bit RGBA textures."[i] Whatever that means translated out of game-dev-speak, I suggest selecting that same setting for "Texture Compression." Also in Build Settings, check "Development Build" and "Script Debugging" (to allow the use of adb, the Android debugger).   If your code compiles successfully when you press "Build" or "Build and Run", Unity will generate an APK (Android Package) that will be deployed to your phone. (The first time you build an Android project on Unity, you will be prompted to select the Android SDK build folder, although usually you just have to confirm with "OK.")


Build Settings

Scenes In Build
- [✓] _Scenes/MainScene 0

Android

Texture Compression	ETC2 (GLES 3.0)
Google Android Project	☐
Development Build	✓
Autoconnect Profiler	☐
Script Debugging	✓

Now is the moment of truth. Don't attach your Gear VR headset to your phone just yet, nor try to use the Gear VR headset USB port, which is power-only. Making sure you have Developer mode enabled on the Gear VR Service app on your phone and that your phone remains unlocked, plug-in your phone to the USB port and select "Build and Run." Once your application displays on screen, disconnect your phone from the USB port and connect to the Gear VR Headset and peer into your Hello World. Weep tears of joy at the birth of your first VR app, or weep tears because I forgot something in this tutorial that was essential for you to get successfully to this point. My bad…

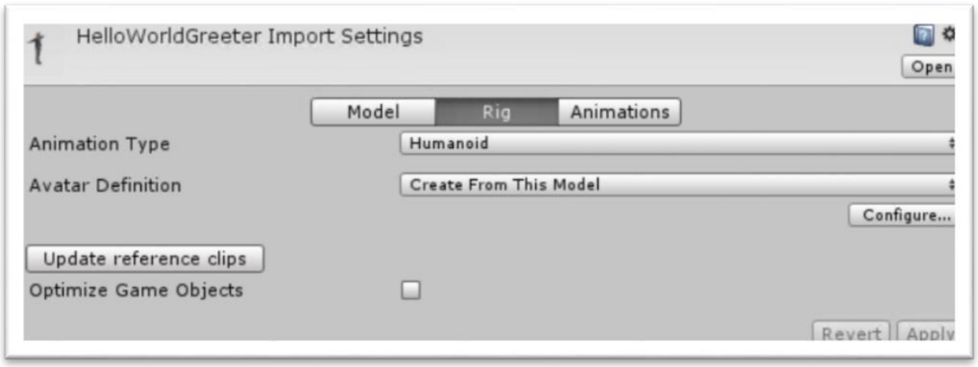
5. Atlas Shrugged and said "So what?"

Okay, it's a start, but it's a relatively lame world. I admit it. We need a means of interacting with the world, of sprinkling signs of life in the void and ways of introducing invasive, targeted marketing advertisements for embarrassing, quasi-medical conditions to pay our app's rent.   "You can't hide in your VR cocoon forever…If those around you suffer when hearing your annoying nasal laughter five times or more a month you may suffer from a serious condition we invented to sell Lafatrox ER…" Luckily, Unity Personal gives us options for the first part, the second part you need the Pro version and no shame.

5.1 Importing an Avatar – definition 1: (origins - Sanskrit) Blue Extraterrestrial Blockbuster

Our Avatar will play the role of livening things up and saying "Hello VR World." But in our current world the Avatar lacks the ground beneath her feet. First, in "Hierarchy View" > "Create" > "Plane." Rename the Plane as "Ground" and expand the Transform > Scale to (100,100,100) or as wide as eyes can see thar' yonder.

Create a folder called "Characters" in "Project View" and import all three .fbx files. Select Hello World Greater.fbx in Project View.   In "Inspector View", select Rig > Animation Type and change it to "Humanoid". Change the "Avatar Definition" to "Create From This Model" and Configure. In the Configuration window, accept and hit okay to all of the mappings. Next press the "Apply" button. Drag the "HelloWorldGreeter.fbx" into the "Hierarchy View."   Next configure the two animation .fbx files the same way, until when it says "Avatar Definition" select "Copy From…" but ***don't*** drag those .fbx files into "Hierarchy View."


HelloWorldGreeter Import Settings

Animation Type: Humanoid

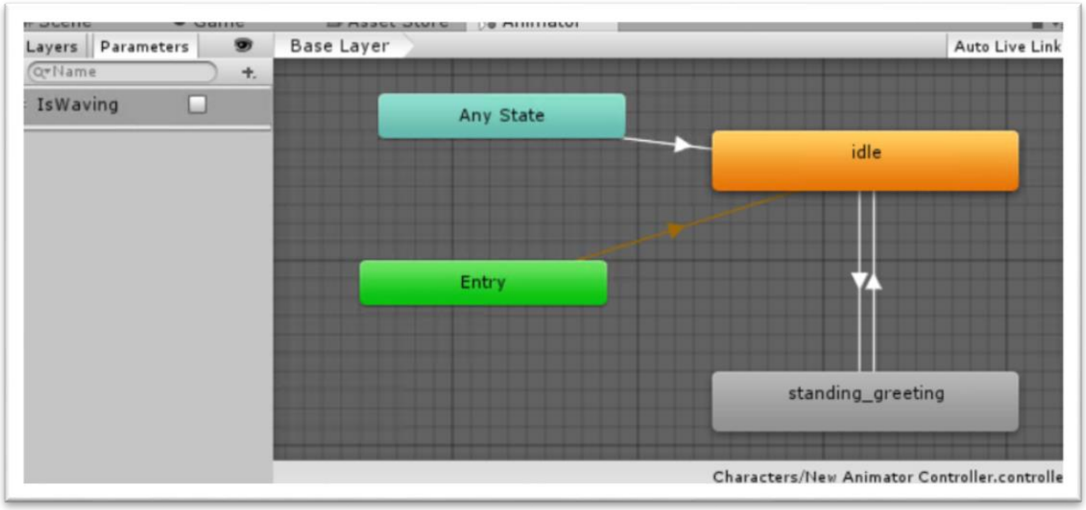
Avatar Definition: Create From This Model

Optimize Game Objects: ☐

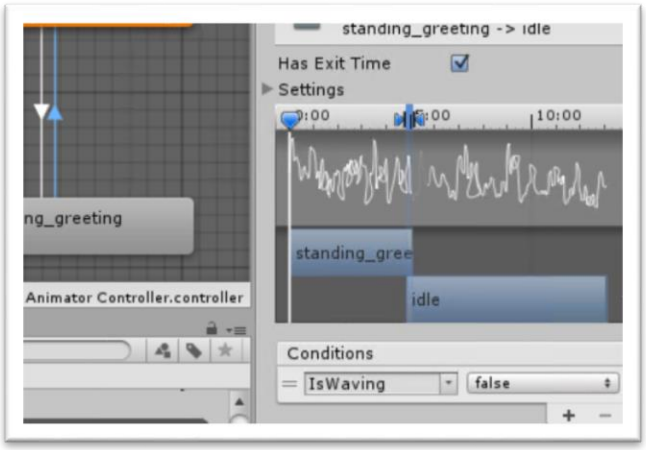
5.2 Animation Controller – Master of State Machine Puppets

In the “Character” folder in “Project View”, select “Create” > “Animation Controller.” Select the “HelloWorldGreeter” in “Hierarchy View”, and in “Inspector View” > “Animator”, select the “New Animation Controller” as the Controller.

Next, open up the “New Animator Controller” from the “Project View.” Select the “Parameters” tab, press the ‘+’ button and add “IsWaving” as a Boolean. Drag each of the two .fbx animation files into the base layer screen. Add transitions between the “Entry” / “Any State” and “idle” and between “idle” and “standing_greeting.”



Select the transition arrow from “idle” to “standing_greeting” and under the Conditions tab, press the ‘+’ button and set IsWaving to true. For the transition arrow you added in the opposite direction, set “IsWaving” to false.



5.3 Triggering Hello

Associate a Script Component with the HelloWorldGreeter named naturally enough, HelloWorldGreeter. This script will act as a bridge between the state machine and other Game Object application events.

```
public class HelloWorldGreeter : MonoBehaviour {

    public bool isWaving;

    private Animator _animator;
       // Use this for initialization
       void Start () {

        _animator = GetComponent<Animator>();

    }

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

        _animator.SetBool("isWaving", isWaving);

    }
}
```

5.4 Adding the Other VR Dimension: Sound

Create an Audio folder in "Project View" and drop the hellovrworld.mp3 into it from Windows Explorer. Select the "HelloWorldGreeter" and in "Inspector View," add an "Audio Source" Component. Select the hellovrworld.mp3 file for the Audio Clip. Uncheck the "Play on Awake" setting to avoid the sound playing on the Scene opening.

5.4.1 Audio Playa' Gotta Play

In the "HelloWorldGreeter" class, you'll add a public method below that will activate the playing of the HelloWorld.mp3. ***For audio purists:*** note that uncompressed sounds like .wav-format files are a more common choice than .mp3s and that I haven't yet heard of a robotic, steampunk IoT phonograph that can synchronize playing vinyl tracks alongside your VR experience, but fret not- that doesn't mean someone won't do it and be on Kickstarter tomorrow.

```
public void SayHelloVRWorld()
    {
        var audio = GetComponent<AudioSource>();
        audio.Play(5000);
    }
```

5.4.2 D.J. State Machine

Open the State Machine Animator Controller again and in the base layer click on the "standing_greeting" box. In the "Inspector View", select "Add a Behavior" and name the new Behavior "GreeterBehavior."

We can send a message from the state machine by attaching a State Machine Behavior to the "standing_greeting" state, and overriding the **OnStateEnter** method. We want the character to say "Hello VR World" in approximate sync with the character's hand wave, and so when the standing greeting state triggers we will "broadcast" a message to the "HelloWorldGreeter" that will invoke the public **SayHelloVRWorld** method added prior:

```
public class GreeterBehavior : StateMachineBehaviour {


    override public void OnStateEnter(Animator animator, AnimatorStateInfo stateInfo,
int layerIndex)
    {
              animator.gameObject.BroadcastMessage("SayHelloVRWorld",
              SendMessageOptions.DontRequireReceiver);
    }

}
```

Test the set-up in "Game View" by checking IsWalking on the "HelloWorldGreeter," while playing. In the next section, you will link the gamer's input to the greeter's reactions and cosmic cycles of creation and destruction.

6 Adding a "Jeeper" and User Interactions with the Oculus Unity SDK

Although it is possible to use Unity 5's generic VR and Input classes for controllers, there can be added complexity in some cases with doing so, and some requirements for submitting your VR App to the Oculus store, such as accessing the Oculus' universal menu are not possible without installing the SDK in your Unity Project. So you will now download and will import the *OculusUtilities.unitypackage* and use some of the Gear VR /Oculus specific SDK classes instead.

From the Oculus "Downloads > "Engine Integration", select "Oculus Utilities for Unity 5" Details button to get the latest available version, relinquish your virtual soul by checking the EULA agreement and "Download." Unzip the OculusUtilities<V>.zip file, and double-click on the *OculusUtilities.unitypackage*. When prompted to import the package files within Unity, do so, and note the new OVR folder in your "Project View."

6.1 OVR Player Controller

In "Project View" > "OVR" > "Prefabs", drag the "OVRPlayerController.prefab" into the "Hierarchy View." Prefabs are templated Game Objects and collections of Game Objects that offer a measure of reusability and maintainability. By dragging the prefab into the “Hierarchy View,” you have created an instance of the OVRPlayerController and its child Game Objects, including an OVR Camera Rig. To use the Gear VR/Oculus specific Camera Rig, you should first select “Hierarchy View” > “Main Camera” in “Inspector View” and uncheck it to the left of the Game Object Name to remove it from the scene, or go ahead and delete the “Main Camera.”

You may need to reposition the OVRPlayerController so that the Hello World Greeter and the World are visible, keeping in mind you can preview the initial perspective in “Game View.”

6.1.1 Import hellovrworldcursor.unitypackage

Double-click on **hellovrworldcursor.unitypackage**, and import the Cursor prefab and Jeeper.cs script, adapted from Oculus SDK Examples Script CrossHair3D.cs (that I highly recommend downloading and reviewing as you go further). Drag the Cursor prefab into the “Hierarchy View.”

Jeeper uses ray casting from the current centerline for the Gear VR’s orientation to show and move the cursor:

```
Ray ray = new Ray(cameraPosition, cameraForward);
            RaycastHit hit;
            //Send out "eye beams" from the camera...and display the jeeper close by.
            if (Physics.Raycast(ray, out hit))
            {
                //Change the Jeeper location relative to the object hit by the ray cast
                transform.position = hit.point + (-cameraForward * offsetFromObjects);
                transform.forward = -cameraForward;



                hit.transform.gameObject.BroadcastMessage("OnRayCastHit", hit.point,
SendMessageOptions.DontRequireReceiver);

            }
```

6.1.2 Message Received

In the HelloWorldGreeter script, add the method “OnRayCastHit.” When this is triggered the isWaving state is toggled, and the state machine can transition between waving and idle.

```
    public void OnRayCastHit(Vector3 vector)
    {
          isWaving = !isWaving;
    }
```

6.2 Oculus User Interface Guidelines

The Gear VR back button is accessible through the mappings that Unity has for a keyboard's escape button (if it doesn't work when you build it, investigate in the Input Manager, accessible via "Edit" > "Project Settings" > "Input"). In Jeeper.cs, add the following snippet to the LateUpdate() method.

```
if (Input.GetKeyDown(KeyCode.Escape))
 {
        OVRManager.PlatformUIGlobalMenu();

 }
```

Although, you should consult with the latest guidelines and best practices from Oculus if you are submitting your application to the Oculus store with respect to expected behavior and timing for how long a user presses the back button on the Gear VR, (since the back button often also used within VR applications to navigate between scenes and menus) this gives

6.3 Using the Power of Prefab to Implement the Many Worlds Hypothesis

6.3.1 Prefab-ulous

Create a folder in Assets called Resources, and open it. Right click, and select Create > Prefab, name it "Earth." Drag the "World" from the Hierarchy View and drop on the "Earth." Create additional prefabs, such as for "Pluto", "Neptune" and "Venus" in "Resources." Add additional textures and materials for the other planets. Prefabs will show in the Hierarchy View menu as blue. To disassociate the World from its prefab, select the World and then "Game Object" > "Break prefab instance." Modify the material for the World, create a new prefab and drag and drop the World from the "Hierarchy View" on its new prefab.

6.3.2 Planets are a Dime a Dozen

Create a new script/class: **AWholeNewWorld** and associate it with the **Ground** as a Script component. Add a public method `void OnRayCastHit(Vector3 vector)` that will be active when the user's "Jeeper" is pointed at a point on the **Ground,** and the user taps on the Gear VR Touchpad.

```
public float minHeight = 6f;
    public float maxHeight = 15f;

public void OnRayCastHit(Vector3 vector)
    {

       //Match precisely the names of your planet prefabs.  Add any additional materials
      // as you did prior with Earth
        string[] planetPrefabs = new string []{ "Earth", "Pluto", "Neptune", "Venus" };

       //Match index range of planetPrefabs
        int planetIndex = Random.Range(0, 3);
       //The Prefabs must be in the “Resources” folder….
       var newPlanetPrefab=Resources.Load(planetPrefabs[planetIndex]);

       //Pseudo-randomly change the position height for the newly created planet relative
       //to the point on the Ground where the Jeeper was pointed.
        float adjustment = Random.Range(minHeight, maxHeight);
        Vector3 adjustedVector = new Vector3(vector.x,vector.y + adjustment, vector.z );

        //Genesis Project!

        Game Object newPlanet = (Game Object)Instantiate(newPlanetPrefab, adjustedVector,
Random.rotation);

        //do something else....


    }
```

6.3.3 Hitchhikers Guide to Nerd Metaphor

In the “World.cs” script, add a public method that will go all “death-star” on your personal Alderaan when a message is broadcast from the “Jeeper” that the gamer is looking directly at a particular world and the gamer has signed the world’s doom by tapping on the Gear VR touch pad or using the dark side of the Force (when someone forks this tutorial and integrates Emotiv brain-wave biofeedback...):

```
public void OnRayCastHit(Vector3 vector)
    {
        Destroy(this.gameObject);

    }
```

7. The VR World is now your Virtual Oyster

Congratulations! Marvel at the magnificent starter VR universe you have created and shout to everyone within earshot: “I have become VR Programmer, Maker of Hello Worlds!” Laugh maniacally at realizing your newfangled, demigod powers. But remember kids, VR safety first! (An epic season of *Tosh.0* / VR Goggle Slapstick is coming soon...don’t star in it.) Now get the shuck out of here and get creating. I highly recommend going through Unity’s video tutorials to understand the core of Unity’s Game Engine capabilities next, even if your next VR app is not a game but will simply leverage Unity’s 3D UI capabilities. Enjoy!

Appendix A - Create Your Own “Hello World” Greeter or Something

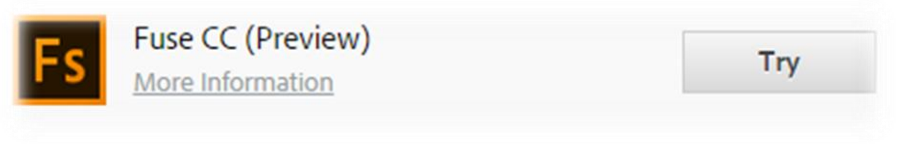
Mixamo, recently acquired by Adobe, makes it easy for a non-artist to create unique, human animations that can then be imported into Unity3D. Fuse CC is the Adobe Creative-Cloud branded character generator that can be combined with realistic motion capture animations on the Mixamo website.

A1.1 Install Adobe Creative Cloud

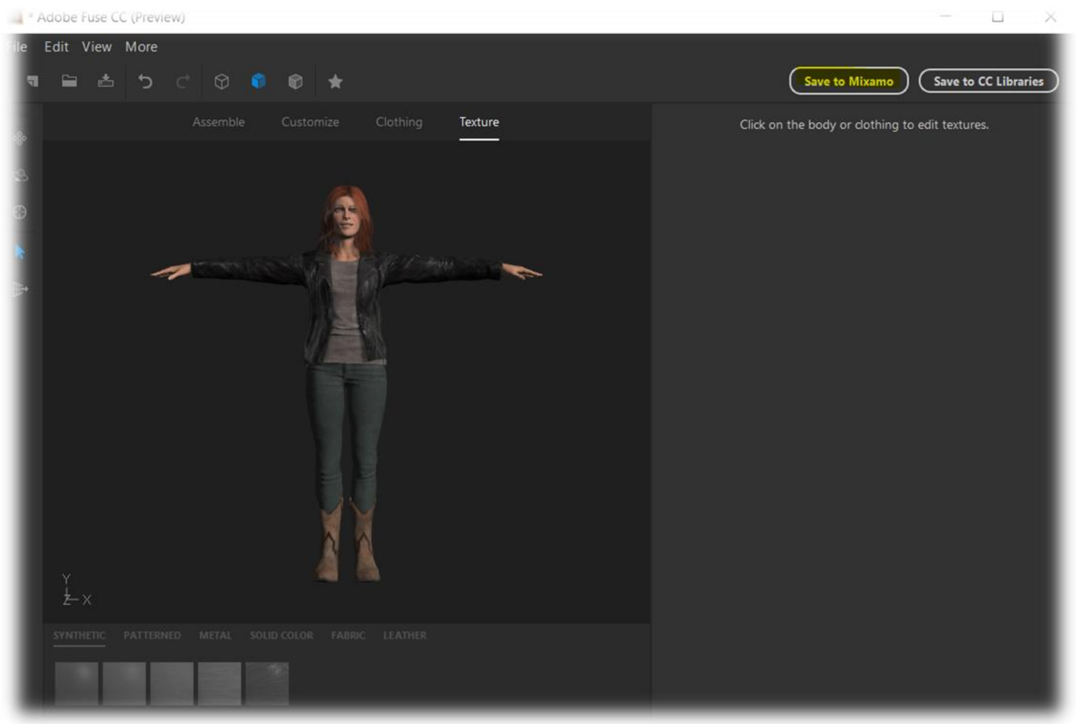
```
C\:>  choco install adobe-creative-cloud -y
```

A1.2 Install Fuse CC

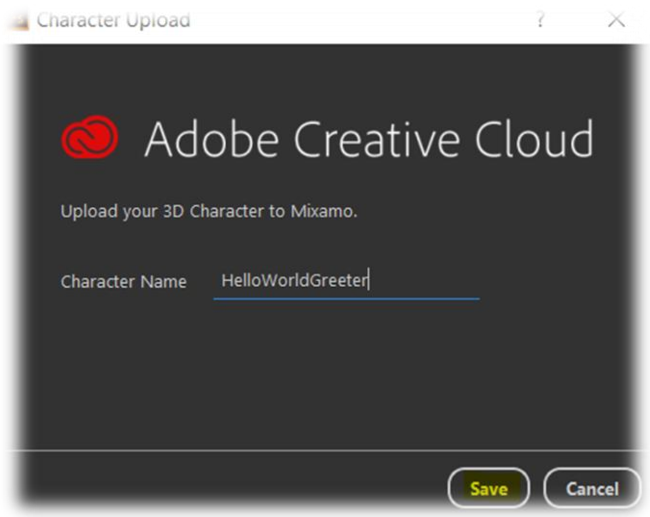
Once installed, open Adobe Creative Cloud and sign-in with or first create your Adobe ID to login. Select the "Try" button to the right of to Fuse CC (Preview), which is free as of February 2016 while Adobe works the kinks out of their acquisition of Mixamo. (**Note**: in my first attempt, I had to uninstall and reinstall Fuse CC after my computer crashed and I could no longer open the application. Another quirk is that the download percentage in the Adobe Creative Cloud downloader remained at 0% until the entire application was downloaded- this doesn't necessarily mean it is not downloading. Additionally, it may depend on your graphics card, VM, or RDP set-up, but when I tried to run Fuse on an Azure-hosted Windows 10 VM it crashed with memory access violation errors…)



Once the download has completed, open Adobe Fuse CC and create a new character. Once you've gone through the steps of Assembling, Customizing, Clothing and Texture, "Save to Mixamo."


* Adobe Fuse CC (Preview)
File Edit View More
Save to Mixamo
Save to CC Libraries
Assemble Customize Clothing Texture
Click on the body or clothing to edit textures.
SYNTHETIC PATTERNED METAL SOLID COLOR FABRIC LEATHER

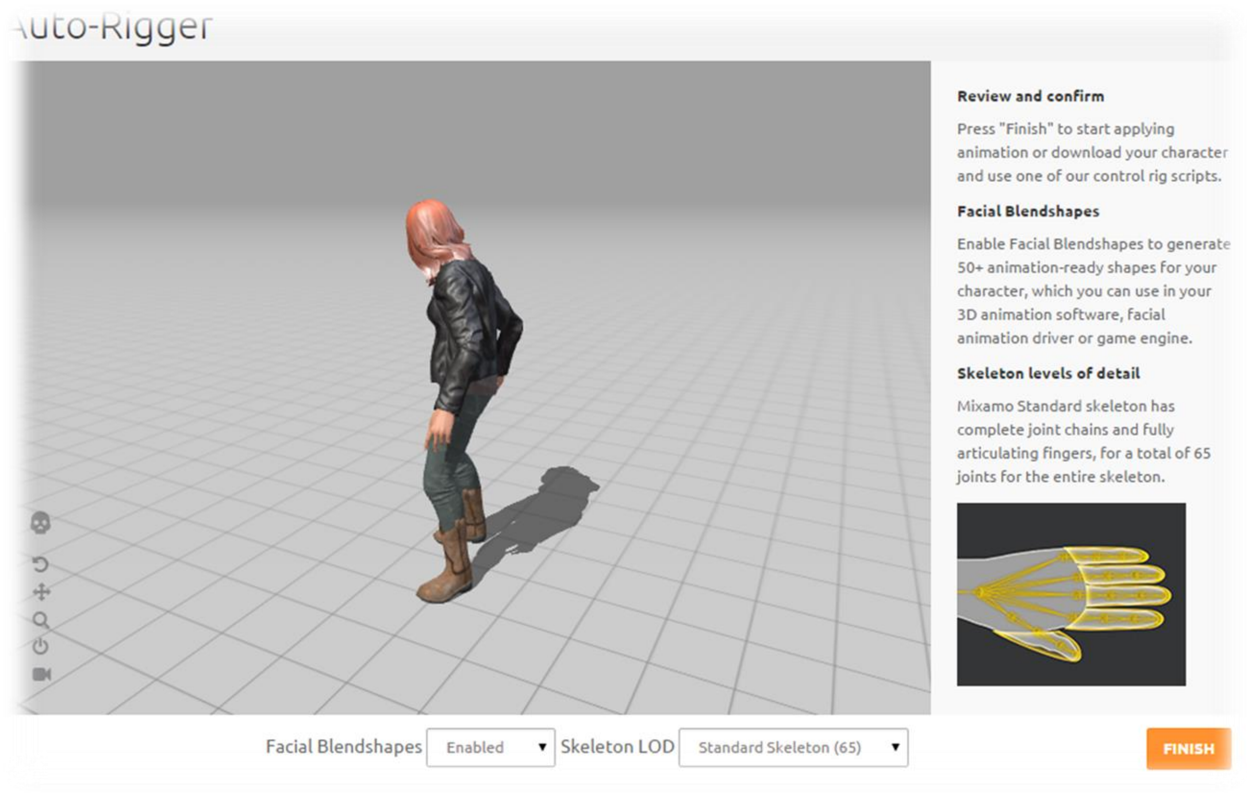
Name your character and press "Save" to upload the character model to Mixamo. Mixamo will create a custom skeleton rig that can then be used in combination with pre-existing motion captures to animate the character.



Upload your 3D Character to Mixamo.

Character Name HelloWorldGreeter

When your character has been successfully uploaded and processed and the Mixamo website opens up with the Auto-Rigger page, select the "Finish" button.



Review and confirm

Press "Finish" to start applying animation or download your character and use one of our control rig scripts.

Facial Blendshapes

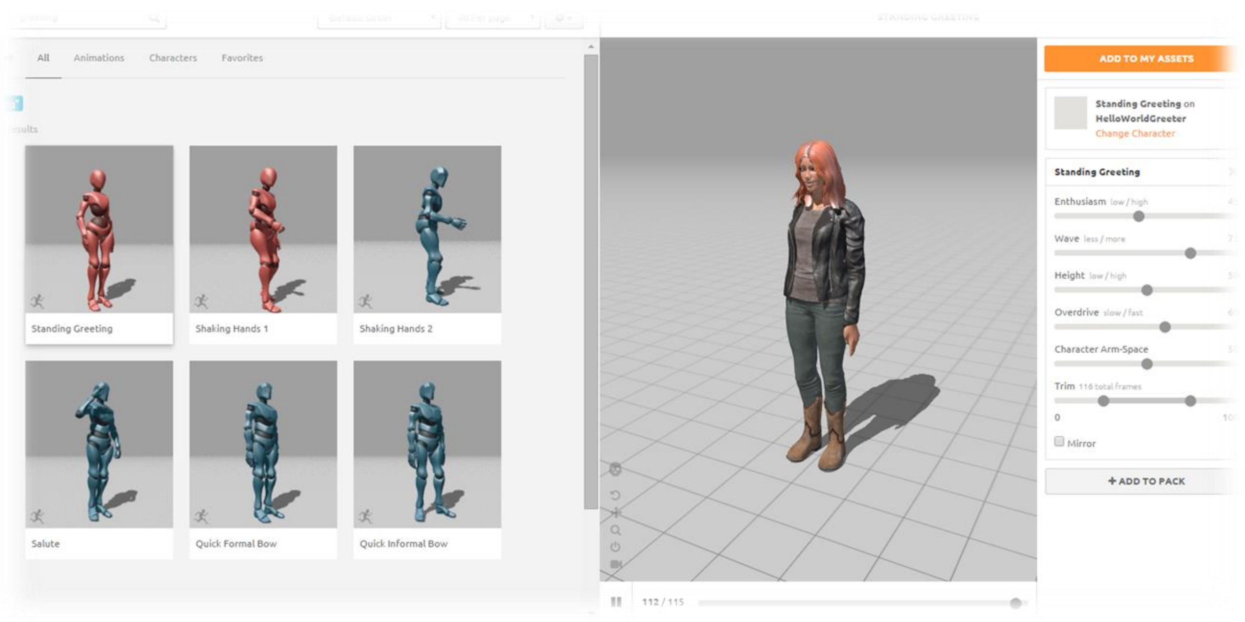
Enable Facial Blendshapes to generate 50+ animation-ready shapes for your character, which you can use in your 3D animation software, facial animation driver or game engine.

Skeleton levels of detail

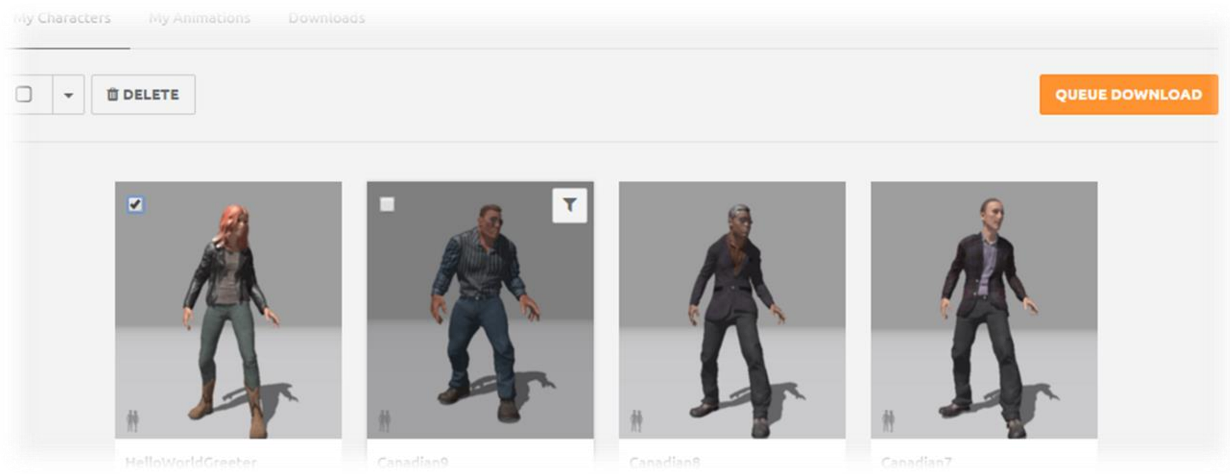
Mixamo Standard skeleton has complete joint chains and fully articulating fingers, for a total of 65 joints for the entire skeleton.

Facial Blendshapes: Enabled
Skeleton LOD: Standard Skeleton (65)

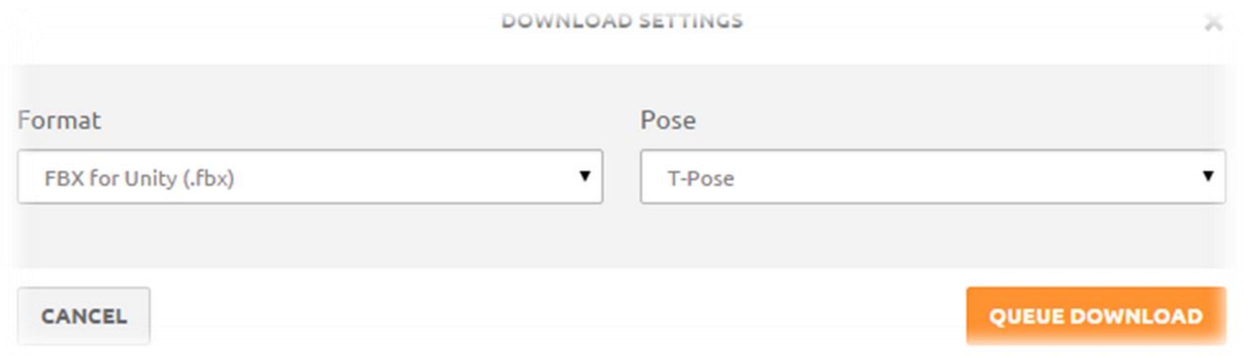
On the next screen, select the "Animate" button. In the web page that appears next, search for "Greeting" and select "Standard Greeting" by clicking on the motion captures at the left. Observe that the motions skeletons appear differentiated from whether the person who motion was captured was male (blue) or female (red). After the animation plays at right, you may modify the parameters to customize the wave motion and press the "Add to my assets" button. Next search, for a default state for the character, I suggest "Idle" and then again press the "Add to my assets" button.



A1.3 Export to Unity 3D



After selecting “My Assets” in the top-navigation menu, you will be taken to the “My Characters” page. Select the character you created with a checkbox and then click on the “Queue Download” button. When you are prompted with the pop-up, select “FBX for Unity(.fbx)” as the Format, and “T-Pose” as the Pose and press the “Queue Download” button.



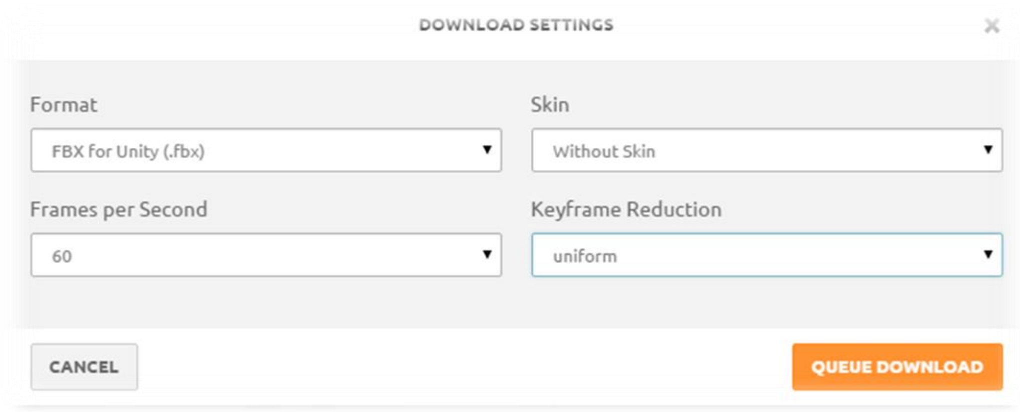
Format: FBX for Unity (.fbx)

Pose: T-Pose

A1.4 Export from My Animations

Now you can export the “Greeting” and “Idle” animations by clicking on “My Animations.”

For the download settings, I recommend FBX for Unity, “Without skin” (to avoid extra MBs in the final executable Unity program), and 60 frames per second since that is the recommend minimum framerate for Gear VR. After making your drop down selections, click on “Queue Download.”



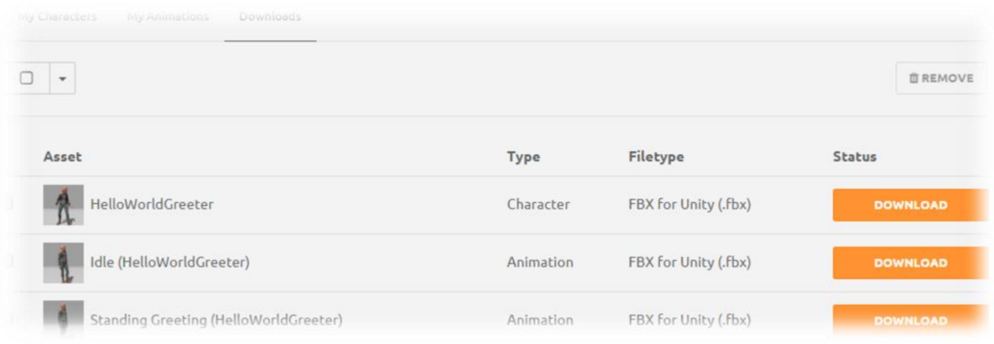
Format: FBX for Unity (.fbx)

Skin: Without Skin

Frames per Second: 60

Keyframe Reduction: uniform

After your character and animations have queued successfully in a few minutes and are ready for download, go to the Downloads page to download them, and adapt the instructions in section 5 to import them into Unity.



Asset	Type	Filetype	Status
HelloWorldGreeter	Character	FBX for Unity (.fbx)	DOWNLOAD
Idle (HelloWorldGreeter)	Animation	FBX for Unity (.fbx)	DOWNLOAD
Standing Greeting (HelloWorldGreeter)	Animation	FBX for Unity (.fbx)	DOWNLOAD

[i] https://developer.oculus.com/documentation/mobilesdk/latest/concepts/mobile-design-intro/

License