Hackathon Open Challenge

Mixed Reality with Hololens

Intelequia VR Team

Versión 1.0

August 14, 2017

Table of contents

1. Introduction

This document is a start-up guide to help developers of BBVA's Open Challenge to prepare their Mixed Reality (MR) projects using Microsoft holographic technology.

The aim of the developers is to design a MR solution to improve the work space of the trader of the future.

But developing high quality MR projects requires prior knowledge such as:

- Prepare holographic tooling and SDK
- Animation, lighting and modeling of 3D objects
- Develop scenes with Unity3D

And also some experience with Hololens, the hardware device Head Mounted Display (HMD) designed by the Microsoft team.

For this reason the organization of the event has decided to democratize the contest allowing the developers to do their projects using only the emulator and later providing them a Hololens for their presentation in Chicago. This prevents the developer from having the Hololens hardware, which is currently in a development phase, with a high price and geographical limitation for its distribution.

This guide, therefore, collects in an organized way the basic procedures and techniques necessary to successfully execute a MR project in the Hololens simulator provided in the latest versions of Microsoft Visual Studio.

Finally, you can note that this guide has been developed by Intelequia VR Team. Any doubts or clarifications that contestants have about the procedures described can direct it to the mailbox of technical support hololens@intelequia.com, indicating in the body of the same the data of registry of the contestant.

2. Prerequisites

Currently, there is no separate SDK for the development of Windows applications of Mixed Reality for that reason we must use Visual Studio with the Windows SDK 10 (version 1511 or higher). Therefore, Mixed Reality Apps are developed with the same tools as the Windows Store Apps.

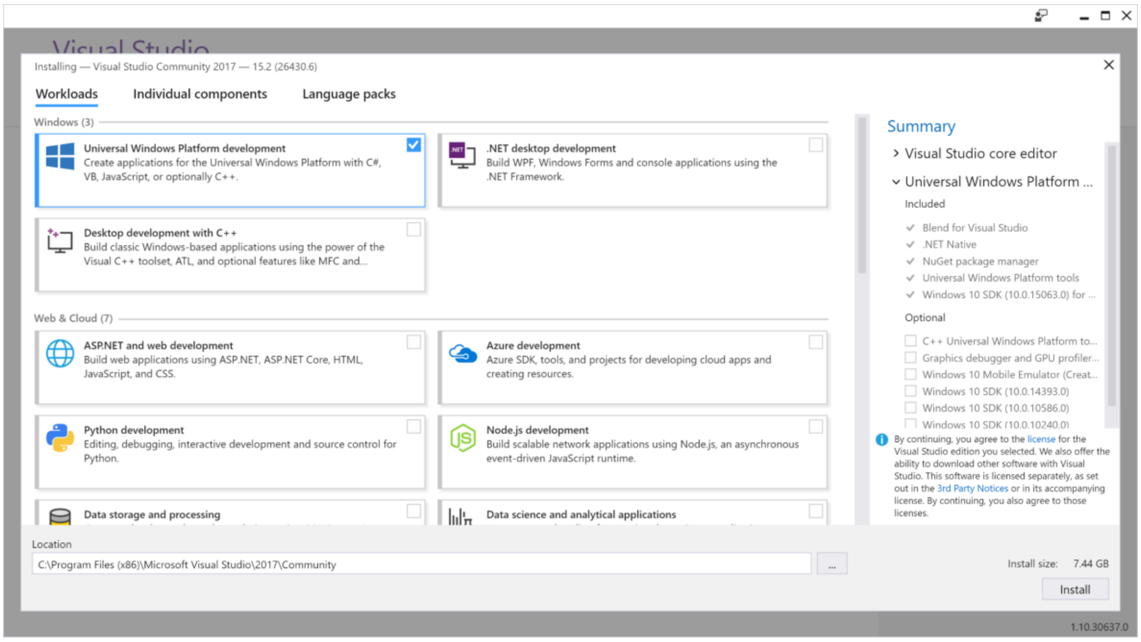
Thanks to the existence of the Hololens emulator you do not need to have a physical device to develop or test your Mixed Reality Apps.

Although it is not strictly necessary, we recommend using the Unity3D game engine so that the development of your Apps is much easier. In this Hackaton only support will be given to the projects made with this engine.

2.1. Visual Studio

Although the version of Visual Studio 2015 Update 3 is also supported, we recommend using Visual Studio 2017 for a better experience. All editions of Visual Studio are supported including the Community.

During the installation of Visual Studio we must select **Universal Windows Platform development**:


Installing — Visual Studio Community 2017 — 15.2 (26430.6)

Workloads | Individual components | Language packs

Windows (3)

- [✓] **Universal Windows Platform development** — Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.
- **.NET desktop development** — Build WPF, Windows Forms and console applications using the .NET Framework.
- **Desktop development with C++** — Build classic Windows-based applications using the power of the Visual C++ toolset, ATL, and optional features like MFC and...

Web & Cloud (7)

- **ASP.NET and web development** — Build web applications using ASP.NET, ASP.NET Core, HTML, JavaScript, and CSS.
- **Azure development** — Azure SDK, tools, and projects for developing cloud apps and creating resources.
- **Python development** — Editing, debugging, interactive development and source control for Python.
- **Node.js development** — Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.
- **Data storage and processing**
- **Data science and analytical applications**

Summary

- Visual Studio core editor
- Universal Windows Platform ...
  - Included
    - ✓ Blend for Visual Studio
    - ✓ .NET Native
    - ✓ NuGet package manager
    - ✓ Universal Windows Platform tools
    - ✓ Windows 10 SDK (10.0.15063.0) for ...
  - Optional
    - C++ Universal Windows Platform to...
    - Graphics debugger and GPU profiler...
    - Windows 10 Mobile Emulator (Creat...
    - Windows 10 SDK (10.0.14393.0)
    - Windows 10 SDK (10.0.10586.0)
    - Windows 10 SDK (10.0.10240.0)

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

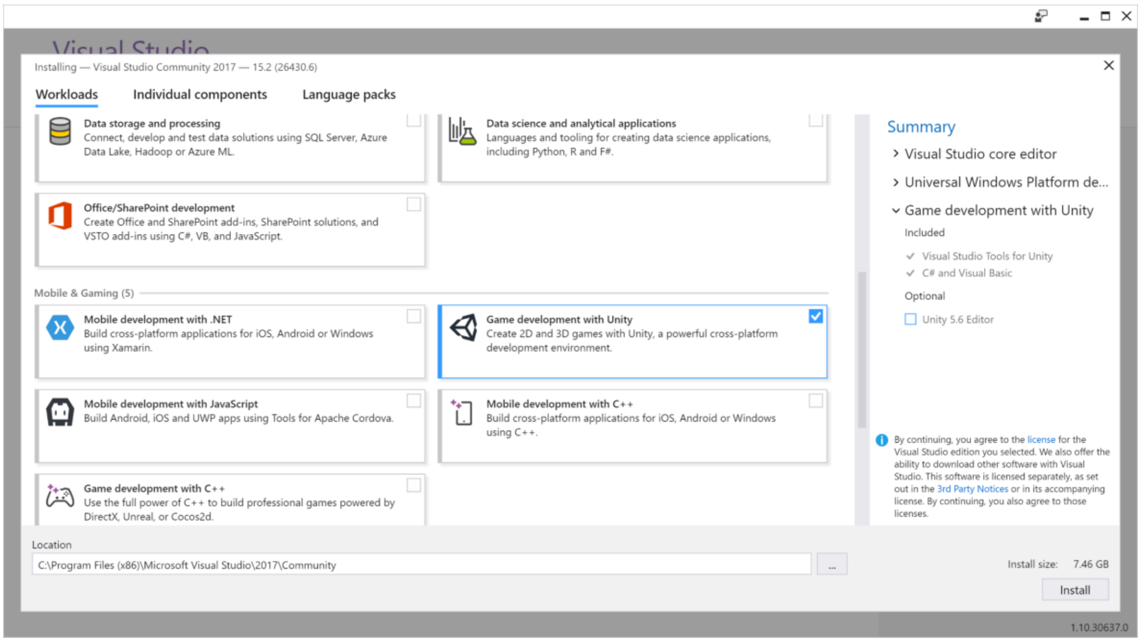
Location: C:\Program Files (x86)\Microsoft Visual Studio\2017\Community

Install size: 7.44 GB

Install

1.10.30637.0

If we are not going to install any version 5.6.0f3 or higher of Unity3D we will also select the components **Game Development with Unity**



Installing — Visual Studio Community 2017 — 15.2 (26430.6)

Workloads | Individual components | Language packs

Data storage and processing
Connect, develop and test data solutions using SQL Server, Azure Data Lake, Hadoop or Azure ML.

Data science and analytical applications
Languages and tooling for creating data science applications, including Python, R and F#.

Office/SharePoint development
Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.

Mobile & Gaming (5)

Mobile development with .NET
Build cross-platform applications for iOS, Android or Windows using Xamarin.

[✓] Game development with Unity
Create 2D and 3D games with Unity, a powerful cross-platform development environment.

Mobile development with JavaScript
Build Android, iOS and UWP apps using Tools for Apache Cordova.

Mobile development with C++
Build cross-platform applications for iOS, Android or Windows using C++.

Game development with C++
Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Location
C:\Program Files (x86)\Microsoft Visual Studio\2017\Community

Summary
- Visual Studio core editor
- Universal Windows Platform de...
- Game development with Unity
  - Included
    - ✓ Visual Studio Tools for Unity
    - ✓ C# and Visual Basic
  - Optional
    - ☐ Unity 5.6 Editor

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Install size: 7.46 GB

Install

1.10.30637.0

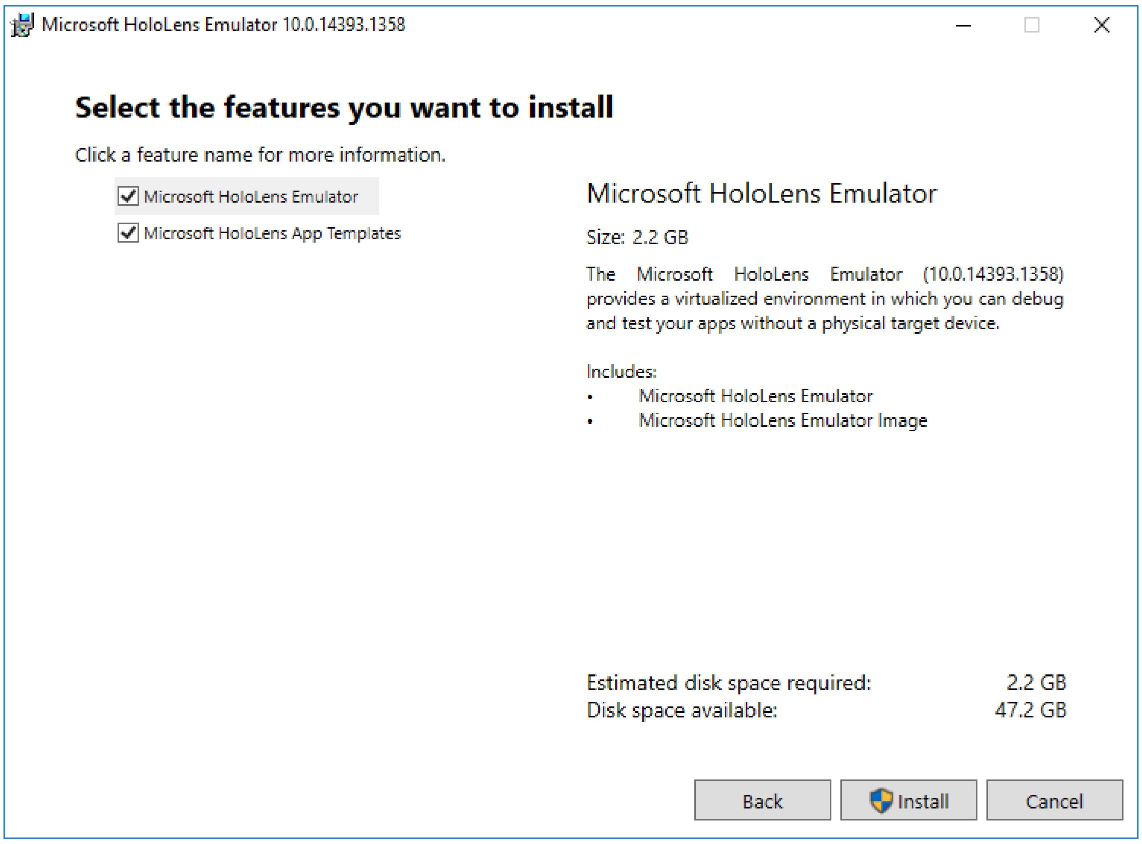
You can download a copy of the Visual Studio Community for free by accepting the terms of use. To see all the available downloads, consult this address: https://developer.microsoft.com/en-us/windows/downloads

2.2. Hololens Emulator

The emulator will allow you to run Windows Holographic applications on a virtual machine without needing a physical Hololens. This includes a virtual image of Hololens running the latest version of Windows Holographic OS. If you have already installed a previous version of the emulator, this build will install side-by-side.

Your system must support Hyper-V for the installation to run successfully. See system requirements for more details.

You can download the emulator from the following address: https://go.microsoft.com/fwlink/?linkid=852626



Select the features you want to install

Click a feature name for more information.

- [✓] Microsoft HoloLens Emulator
- [✓] Microsoft HoloLens App Templates

Microsoft HoloLens Emulator

Size: 2.2 GB

The Microsoft HoloLens Emulator (10.0.14393.1358) provides a virtualized environment in which you can debug and test your apps without a physical target device.

Includes:

- Microsoft HoloLens Emulator
- Microsoft HoloLens Emulator Image

Estimated disk space required:	2.2 GB
Disk space available:	47.2 GB

2.3. Unity3D

The Unity3D gaming engine accelerates the development of holographic applications and seamlessly integrates with the Windows 10 SDK to produce solutions compatible with Visual Studio 2015 or 2017.

You can get version 5.6 or higher (including the recent 2017.1) from its official site: https://store.unity.com/download

2.4. System Requirements

The first requirement is that you have the Windows 10 SDK (included in the Windows Universal Application Development package that you can select during the Visual Studio installation). This SDK where it works best is in the operating system Windows 10 but is also supported in Windows 8.1, Windows 8, Windows 7, Windows Server 2012 and Windows Server 2008 R2.

On the other hand, the requirements for the installation of Visual Studio 2017 are:

- Sistema Operativo:

- Windows 10 version 1507 or higher: Home, Professional, Education, and Enterprise (LTSB and S are not supported)
- Windows Server 2016: Standard and Datacenter
- Windows 8.1 (with Update 2919355): Core, Professional, and Enterprise
- Windows Server 2012 R2 (with Update 2919355): Essentials, Standard, Datacenter
- Windows 7 SP1 (with latest Windows Updates): Home Premium, Professional, Enterprise, Ultimate

- Hardware:
    - 1.8 GHz or faster processor. Dual-core or better recommended
    - 2 GB of RAM; 4 GB of RAM recommended (2.5 GB minimum if running on a virtual machine)
    - Hard disk space: 1GB to 40GB, depending on features installed
    - Video card that supports a minimum display resolution of 720p (1280 by 720); Visual Studio will work best at a resolution of WXGA (1366 by 768) or higher

The HoloLens emulator is based on Hyper-V and uses RemoteFx for hardware accelerated graphics. To use the emulator, make sure your PC meets these hardware requirements:

- Software:
    - 64-bit Windows 10 Pro, Enterprise, or Education (The Home edition does not support Hyper-V or the HoloLens emulator)
    - DirectX 11.0 or later
    - WDDM 1.2 driver or later
- Hardware:
    - 64-bit CPU with 4 cores (or multiple CPU's with a total of 4 cores)
    - 8 GB of RAM or more
    - In the BIOS, the following features must be supported and enabled:
        - Hardware-assisted virtualization
        - Second Level Address Translation (SLAT)
        - Hardware-based Data Execution Prevention (DEP)

If your system meets the above requirements, **please ensure that the "Hyper-V" feature has been enabled on your system** through Control Panel -> Programs -> Programs and Features -> Turn Windows Features on or off -> ensure that "Hyper-V" is selected for the Emulator installation to be successful.

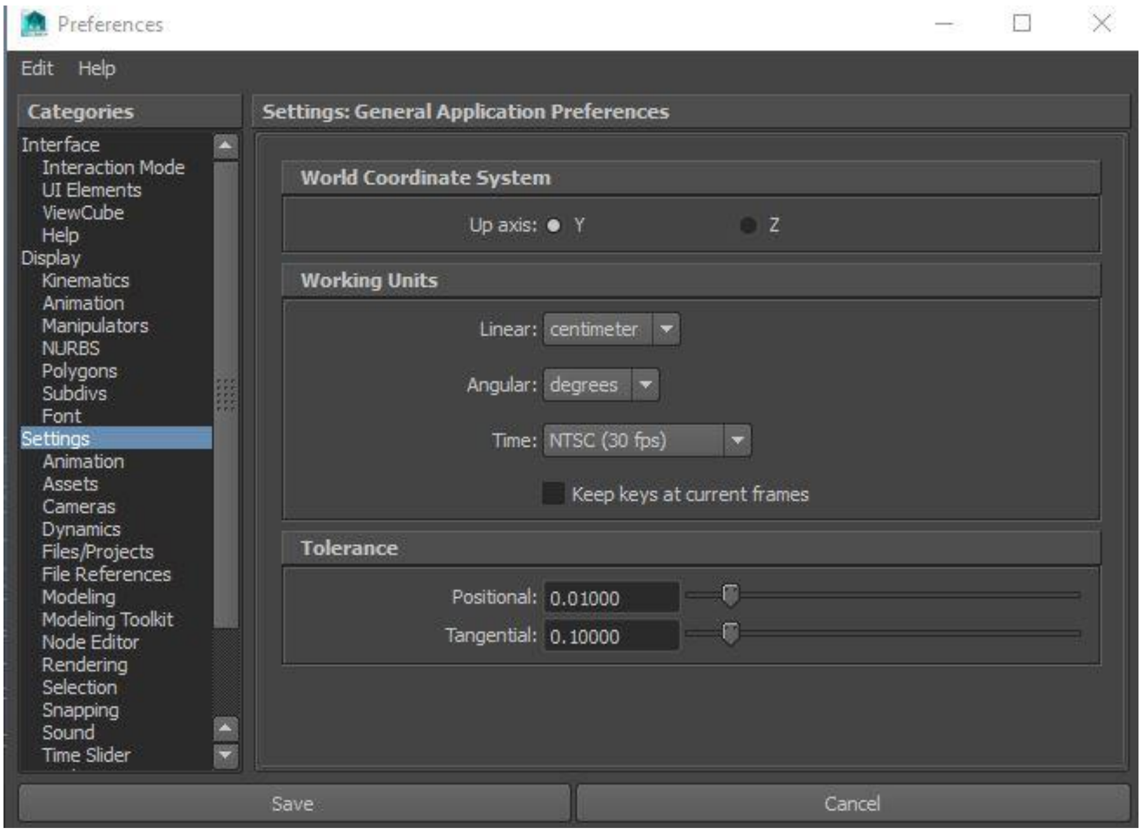
3. How to make Holograms

In this section we will explain how you can create holograms. That is, 3D elements suitable for loading into your Unity3D scenes.

We advise using professional 3D modeling tools like Maya3D or 3DS MAX and understand that you have the basic knowledge of artistic design.

3.1. Configure your 3D software

Set your system and project units to Metric for your software to work consistently with Unity. The system unit default for Maya is centimeters.


Preferences

Settings: General Application Preferences

World Coordinate System

Up axis: ● Y ○ Z

Working Units

Linear: centimeter

Angular: degrees

Time: NTSC (30 fps)

Keep keys at current frames

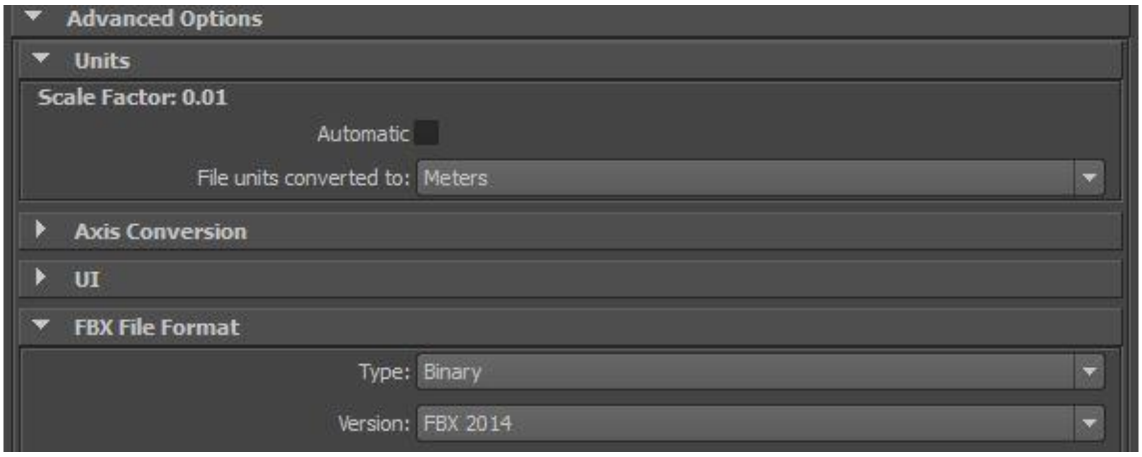
Tolerance

Positional: 0.01000

Tangential: 0.10000

Get the scale right from the beginning. Make art so that they can all be imported at a scale factor of 1, and that their transforms can be scaled 1, 1, 1. Use a reference object (a Unity cube) to make scale comparisons easy. Choose a world to Unity units ratio suitable for your game, and stick to it.

Make sure you check the FBX import and export scale settings in Inspector.


Advanced Options

Units

Scale Factor: 0.01

Automatic

File units converted to: Meters

Axis Conversion

UI

FBX File Format

Type: Binary

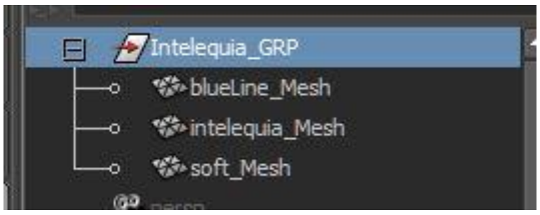
Version: FBX 2014

Animation frame rate defaults can be different in different packages, it is a good idea to set this consistently across your pipeline (for example, at 30fps).

Name objects in your Scene sensibly and uniquely. Avoid special characters such as *()?"#$.

Use simple but descriptive names for both objects and files in order to allow for duplication later

Keep your hierarchies as simple as you can.


Intelequia_GRP
- blueLine_Mesh
- intelequia_Mesh
- soft_Mesh

3.2. Meshes

Build with an efficient topology. Use polygons only where you need them.

Optimise your geometry if it has too many polygons.

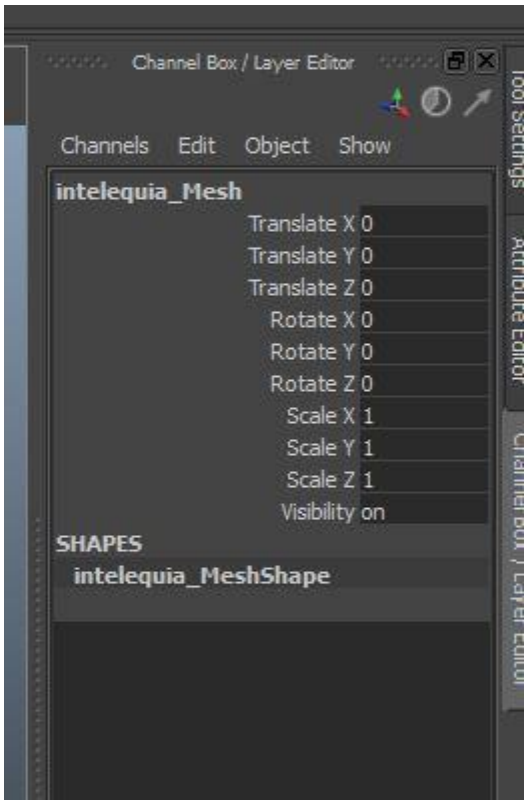
Where you can afford them, evenly spaced polygons in buildings, landscape and architecture will help spread lighting and avoid awkward kinks.

intelequia
SOFTWARE SOLUTIONS

The method you use to construct objects can have a massive effect on the number of polygons, especially when not optimized. A good rule of thumb is often to start simple and add detail where needed. It's always easier to add polygons than take them away.

Put character and standing object pivots at the base, not in the centre. This makes it easy to put characters and objects on the floor precisely. It also makes it easier to work with 3D as if it is 2D for game logic, AI, and even physics when appropriate.

Remember to Freeze the transformations and **delete the history mesh**.



intelequia_Mesh

Translate X	0
Translate Y	0
Translate Z	0
Rotate X	0
Rotate Y	0
Rotate Z	0
Scale X	1
Scale Y	1
Scale Z	1
Visibility	on

SHAPES
intelequia_MeshShape

Make all meshes face in the same direction (positive or negative z axis). This applies to meshes such as characters and other objects that have a concept of facing direction.

3.3. Textures

If you author your textures to a power of two (for example, 512x512 or 256x1024), the textures will be more efficient and won't need rescaling at build time. You can use up to 4096x4096 pixels, but 2048x2048 is the highest available on many graphics cards and platforms.

Work with a high-resolution source file outside your Unity project. You can always downsize from the source, but not the other way round.

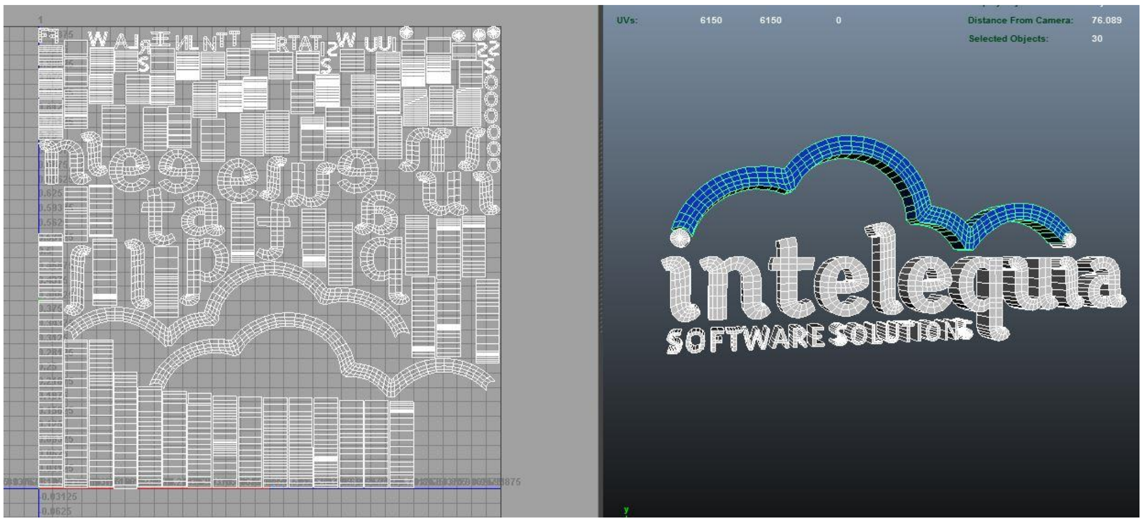
Make sure your 3D working file refers to the same Textures, for consistency when you save or export.

For alpha and elements that may require different Shaders, separate the Textures.

Make use of tiling Textures which seamlessly repeat. This allows you to use better resolution repeating over space.

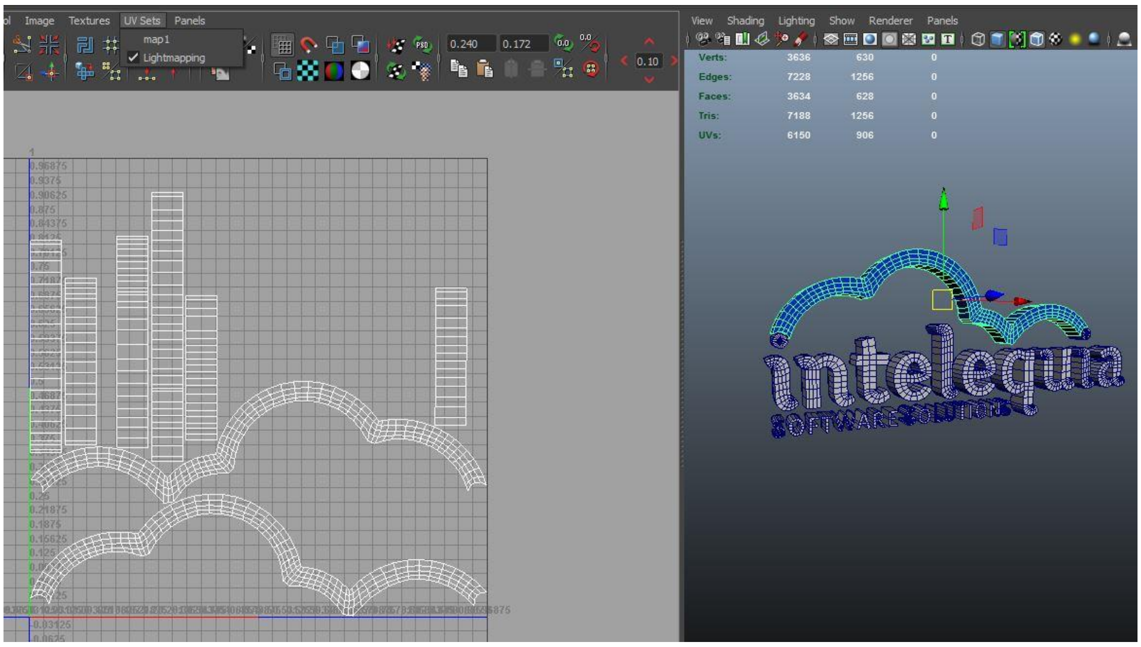
3.4. UV Mapping

UV mapping are used to project a 2D image to a 3D model's surface for texture mapping.



Unity also uses a second UV Set for the Lightmap process.

The Lightmap is technically a second UV Map. The Unity engine needs it to bake lightning information onto the model. You can have Unity create the Lightmap on its own while importing the mesh but we recommend to build your own on a second UV Map channel **avoiding overlapping faces on your LIghtmapping UV set.**

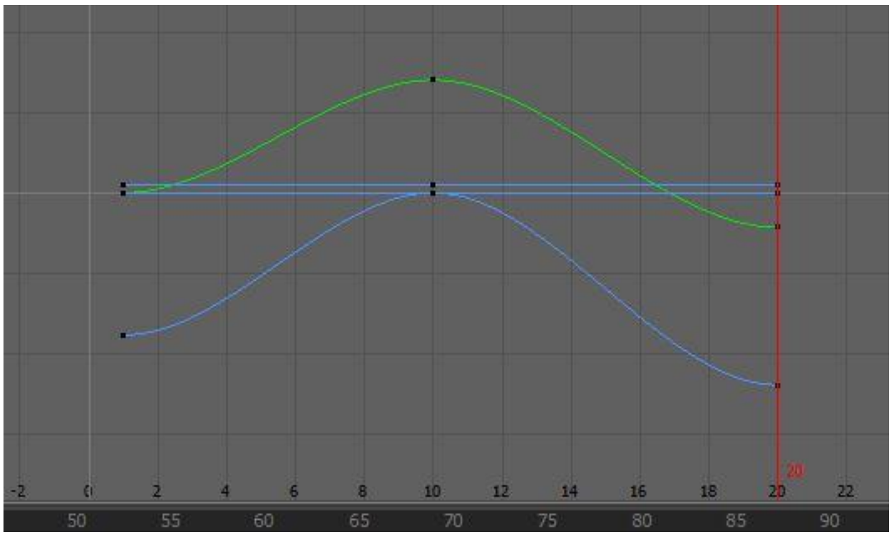


Verts:	3636	630	0
Edges:	7228	1256	0
Faces:	3634	628	0
Tris:	7188	1256	0
UVs:	6150	906	0

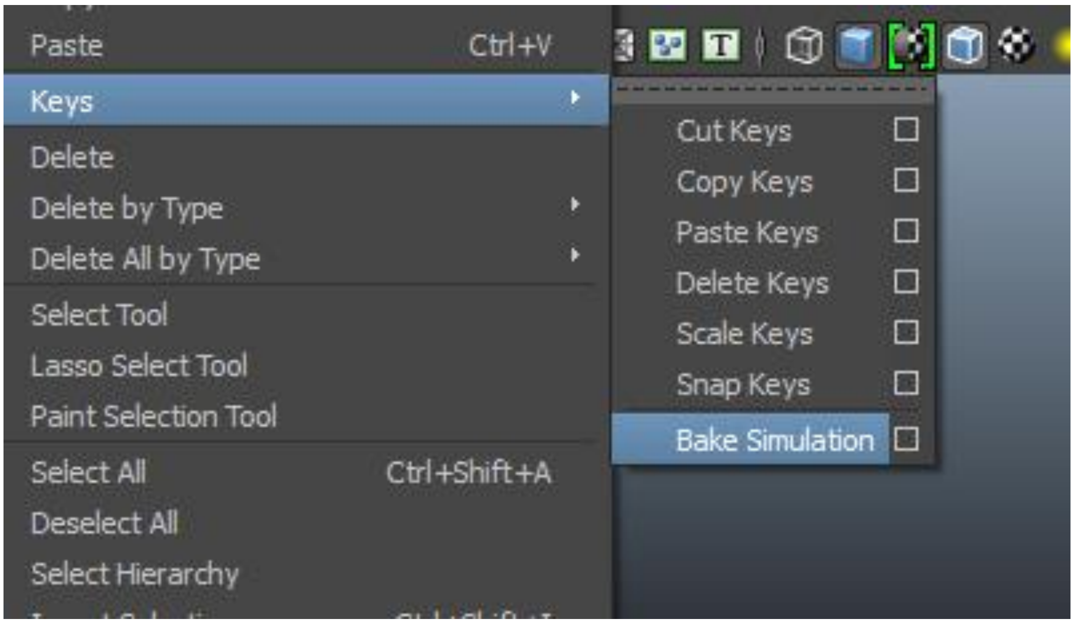
3.5. Animation

You will use most of the times two types of animation, objects moving on the 3d space or joint driven systems such as humanoids characters.

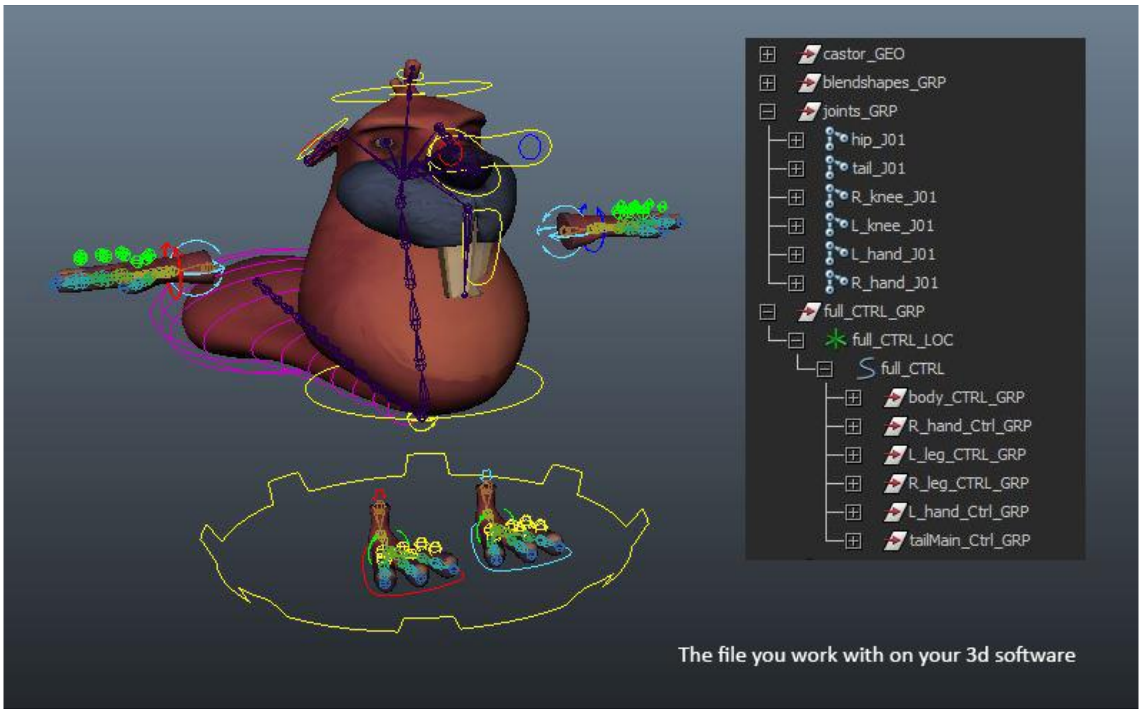
When you have only 3d objects moving (without a joint system and a skinned mesh), its recommended not to bake your animation on your 3d software. Just make your animation with the keyframes you need and export it.



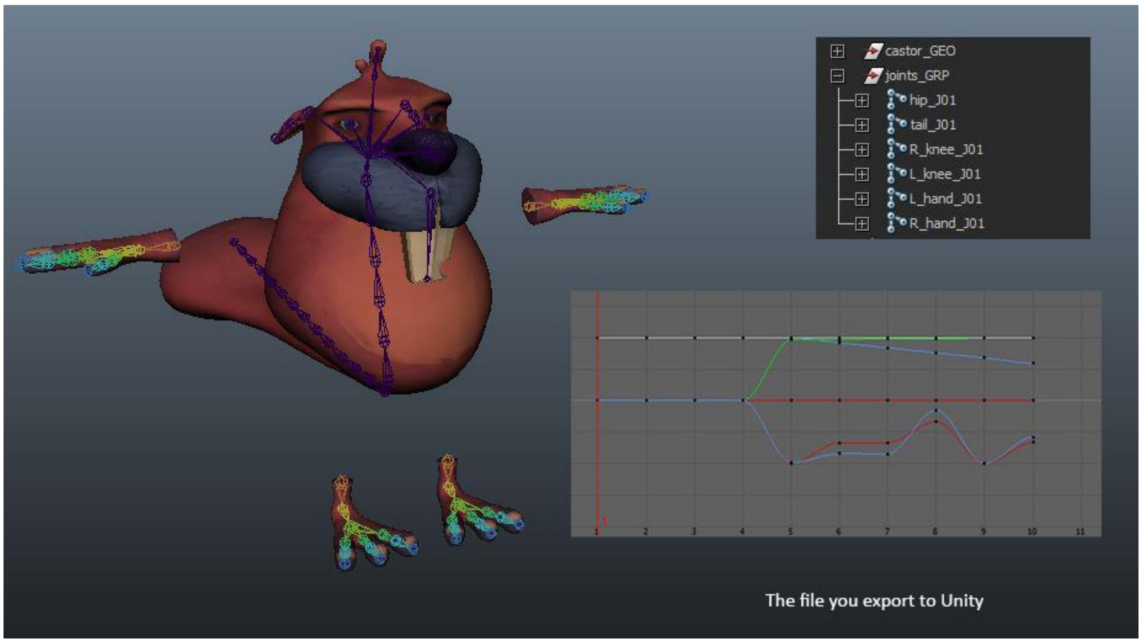
If you need to export humanoid animations make sure you just export your joint hierarchy system and the skinned mesh. The joint system must be baked, not the mesh. Baking a simulation allows you to generate a single animation curve for an object whose actions are being provided by simulation rather than by keys and animation curves (keysets).



Use your first 5 to 10 frames of your timeline to store a T-Pose of your character to configure your avatar on Unity.



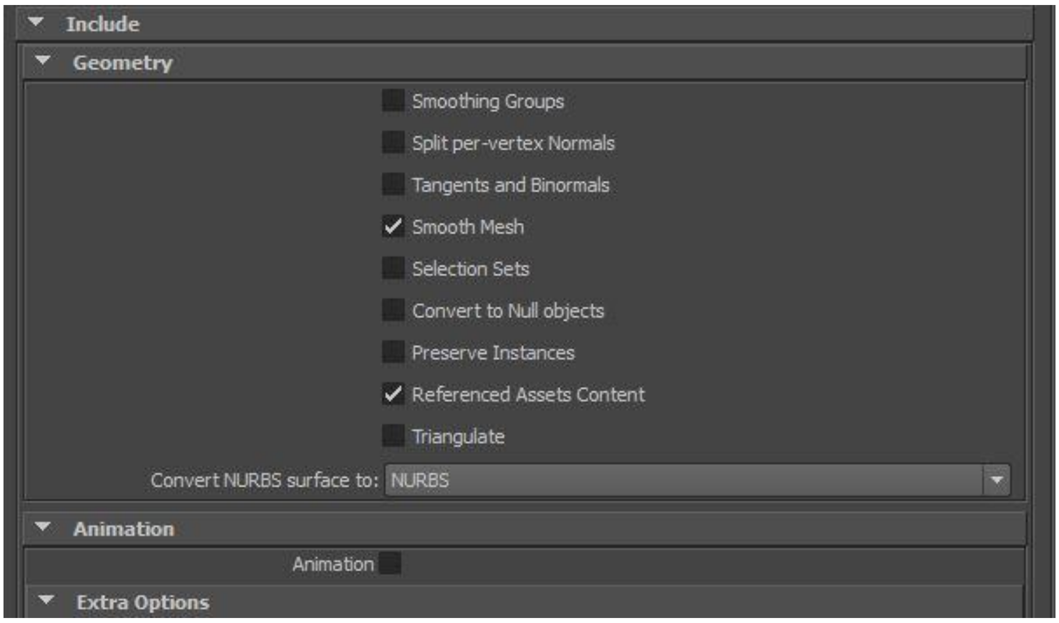
After your animation is done, select all your binded joints and bake that simulation to store a key on every frame. Then delete your control rig and export only the joint system and the mesh. You can delete your Blendshapes, they are stored on the main Mesh.



3.6. Export your Hologram

The most common file type is FBX. It can store mesh and animation data.

For static objects uncheck the animation settings.



Include

Geometry

- [ ] Smoothing Groups
- [ ] Split per-vertex Normals
- [ ] Tangents and Binormals
- [x] Smooth Mesh
- [ ] Selection Sets
- [ ] Convert to Null objects
- [ ] Preserve Instances
- [x] Referenced Assets Content
- [ ] Triangulate

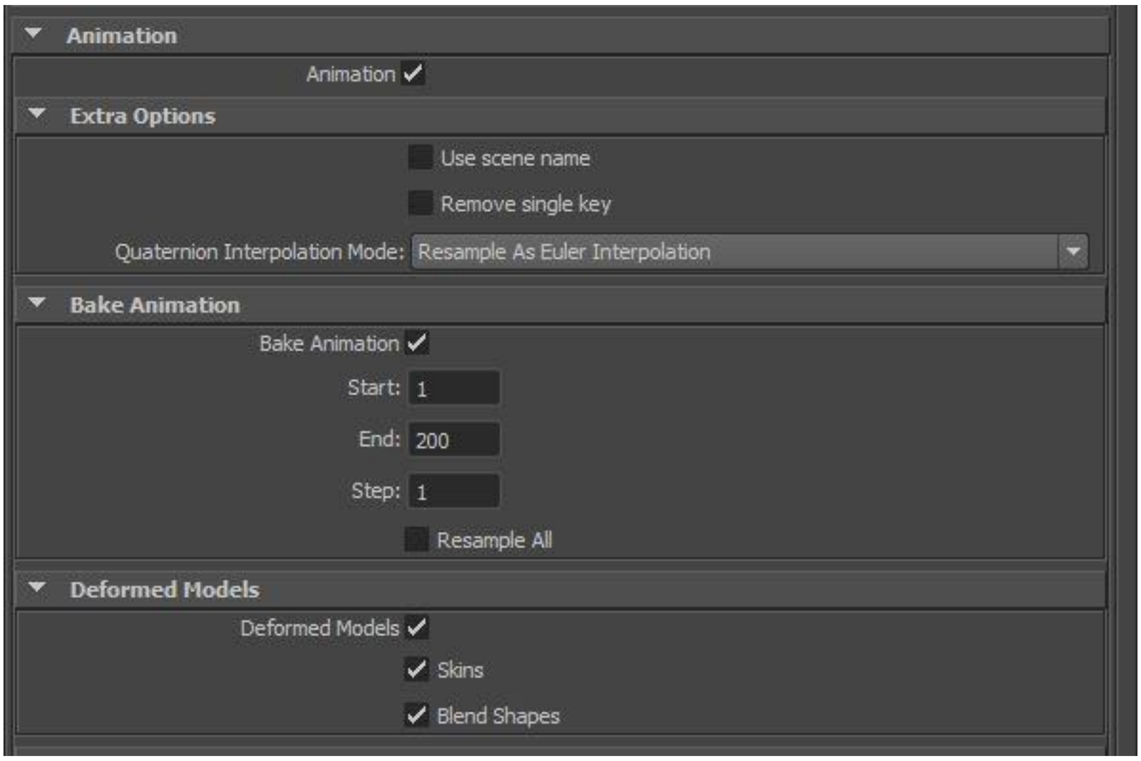
Convert NURBS surface to: NURBS

Animation

Animation [ ]

Extra Options

For animated Holograms you have to check the animation setting. If you have skinned meshes remember to bake your animation before exporting it, you can also do it here in the export settings but it's not recommended. Also check the Blend Shape option if your mesh has them.


Animation

Animation ✓

Extra Options

Use scene name

Remove single key

Quaternion Interpolation Mode: Resample As Euler Interpolation

Bake Animation

Bake Animation ✓

Start: 1

End: 200

Step: 1

Resample All

Deformed Models

Deformed Models ✓

✓ Skins

✓ Blend Shapes

4. How to create Unity scene

The fastest path to building a mixed reality app is with Unity. We recommend you take the time to explore the Unity tutorials. If you need assets, Unity has a comprehensive Asset Store. Once you have built up a basic understanding of Unity, you can visit the Mixed Reality Academy to learn the specifics of mixed reality development with Unity. Be sure to visit the Unity Mixed Reality forums to engage with the rest of the community building mixed reality apps in Unity and find solutions to problems you might run into.

Now that you have seen how you can create and export your professional holograms in FBX format, it is time for you to create your first Unity3D project where you can import these elements into scenes that allow you to interact with Hololens capabilities.

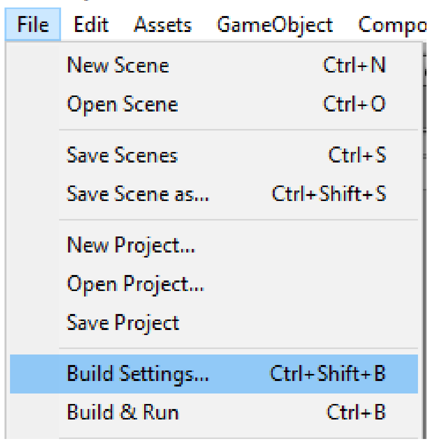
4.1. Configuring a Unity project

If you've just created a new Unity project, there are a small set of Unity settings you'll need to change for Windows Mixed Reality, broken down into two categories: per-project and per-scene.

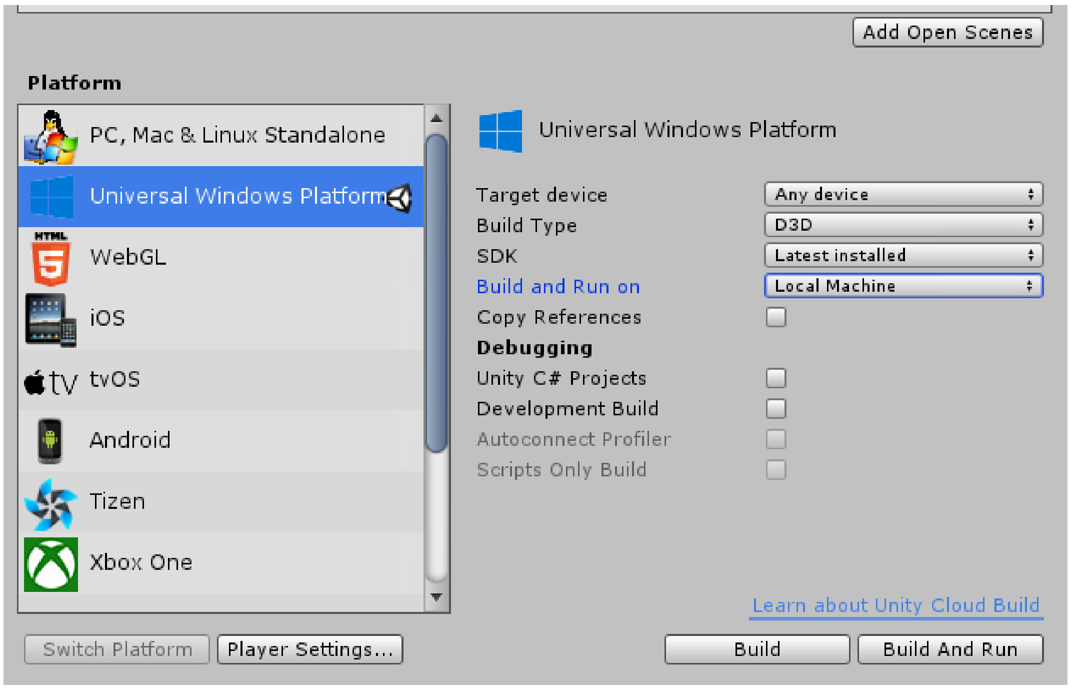
Per-project settings

To target Windows Mixed Reality, you first need to set your Unity project to export as a Universal Windows Platform app:

1. Select **File > Build Settings...**



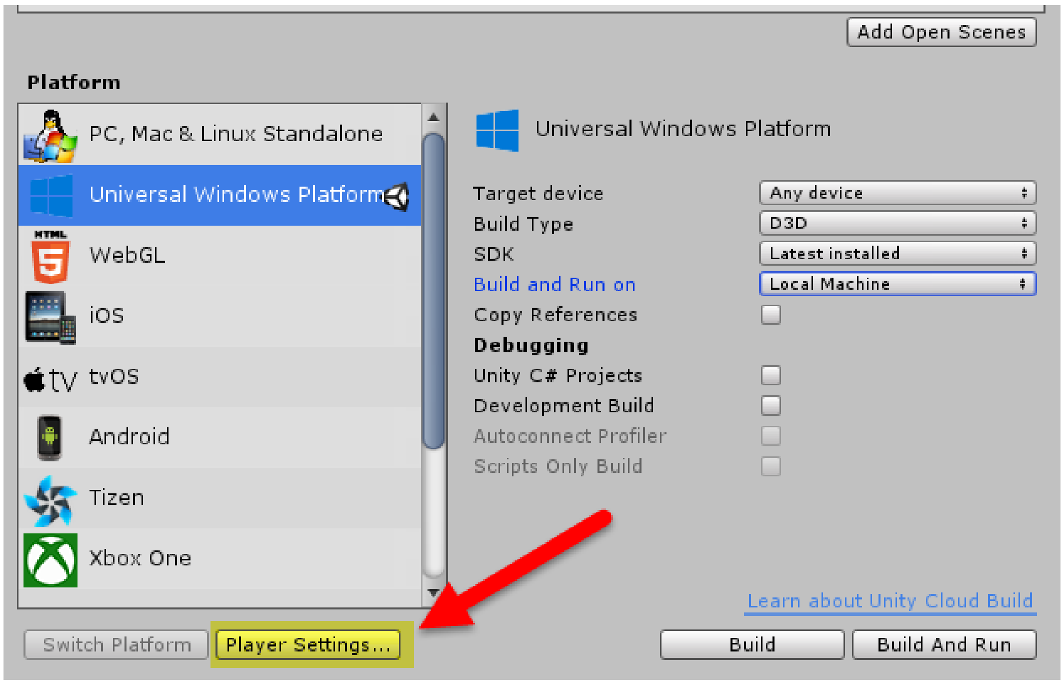
2. Select **Windows Store** in the Platform list and click **Switch Platform**
3. Set **Target device** to **Any Device** to support immersive headsets or switch to **HoloLens**
4. Set **Build Type** to **D3D**
5. Set **UWP SDK** to **Latest installed**


Add Open Scenes
Platform
PC, Mac & Linux Standalone
Universal Windows Platform
WebGL
iOS
tvOS
Android
Tizen
Xbox One
Universal Windows Platform
Learn about Unity Cloud Build
Switch Platform
Player Settings...
Build
Build And Run

Target device	Any device
Build Type	D3D
SDK	Latest installed
Build and Run on	Local Machine
Copy References	☐
Debugging	
Unity C# Projects	☐
Development Build	☐
Autoconnect Profiler	☐
Scripts Only Build	☐

We then need to let Unity know that the app we are trying to export should create an immersive view instead of a 2D view. We do that by enabling "Virtual Reality Supported":

1. From the **Build Settings...** window, open **Player Settings...**



Add Open Scenes

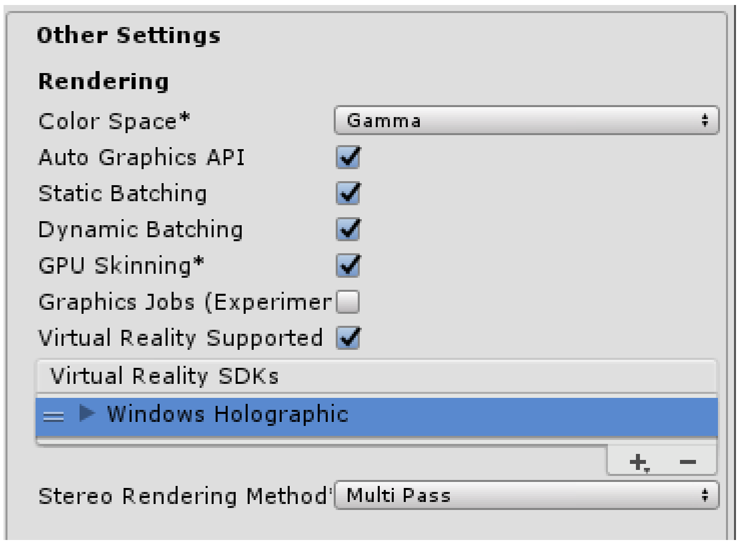
Platform

- PC, Mac & Linux Standalone
- Universal Windows Platform
- WebGL
- iOS
- tvOS
- Android
- Tizen
- Xbox One

Universal Windows Platform

Target device	Any device
Build Type	D3D
SDK	Latest installed
Build and Run on	Local Machine
Copy References	☐
Debugging	
Unity C# Projects	☐
Development Build	☐
Autoconnect Profiler	☐
Scripts Only Build	☐

2. Select the **Settings for Windows Store** tab
3. Expand the **Other Settings** group
4. In the **Rendering** section, check the **Virtual Reality Supported** checkbox to add a new **Virtual Reality SDKs** list and confirm **"Windows Holographic"** is listed as a supported device.



Other Settings

Rendering

Color Space*	Gamma
Auto Graphics API	✓
Static Batching	✓
Dynamic Batching	✓
GPU Skinning*	✓
Graphics Jobs (Experimer	☐
Virtual Reality Supported	✓

Virtual Reality SDKs

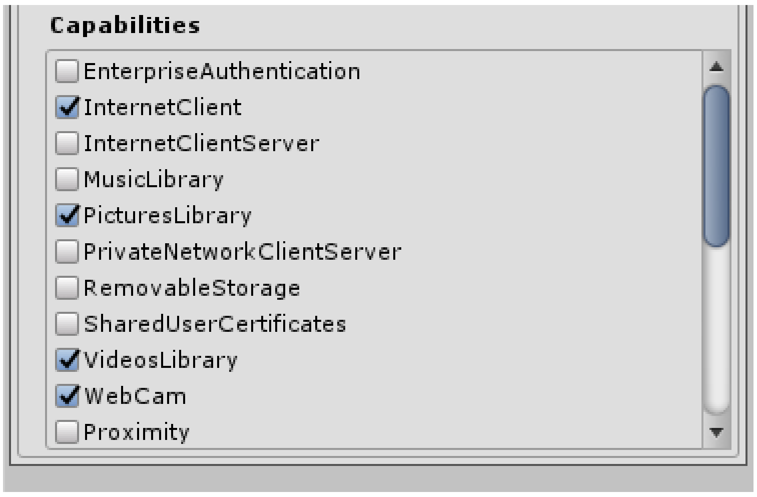
- Windows Holographic

Stereo Rendering Method* Multi Pass

For an app to take advantage of certain functionality, it must declare the appropriate capabilities in its manifest. The manifest declarations can be made in Unity so they are included in every subsequent project export. The setting are found in **Player Settings > Windows Store > Publishing Settings > Capabilities**.

The applicable capabilities for enabling the commonly used APIs for Holographic apps are:

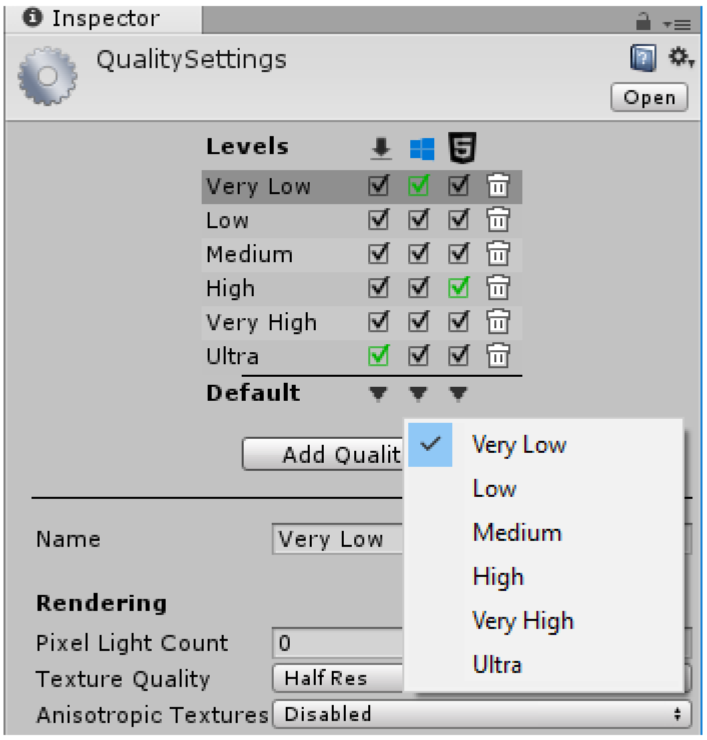
Capability	APIs requiring capability
SpatialPerception	SurfaceObserver
WebCam	PhotoCapture and VideoCapture
PicturesLibrary / VideosLibrary	PhotoCapture or VideoCapture, respectively (when storing the captured content)
Microphone	VideoCapture (when capturing audio), DictationRecognizer, GrammarRecognizer, and KeywordRecognizer
InternetClient	DictationRecognizer (and to use the Unity Profiler)


Capabilities

- [ ] EnterpriseAuthentication
- [x] InternetClient
- [ ] InternetClientServer
- [ ] MusicLibrary
- [x] PicturesLibrary
- [ ] PrivateNetworkClientServer
- [ ] RemovableStorage
- [ ] SharedUserCertificates
- [x] VideosLibrary
- [x] WebCam
- [ ] Proximity

HoloLens has a mobile-class GPU. If your app is targeting HoloLens, you'll want the quality settings tuned for fastest performance to ensure we maintain full framerate:

1. Select **Edit** > **Project Settings** > **Quality**
2. Select the **dropdown** under the **Windows Store** logo and select **Fastest**. You'll know the setting is applied correctly when the box in the Windows Store column and **Fastest** row is green.



Inspector

QualitySettings

Levels				
Very Low	✓	✓	✓	
Low	✓	✓	✓	
Medium	✓	✓	✓	
High	✓	✓	✓	
Very High	✓	✓	✓	
Ultra	✓	✓	✓	
Default				

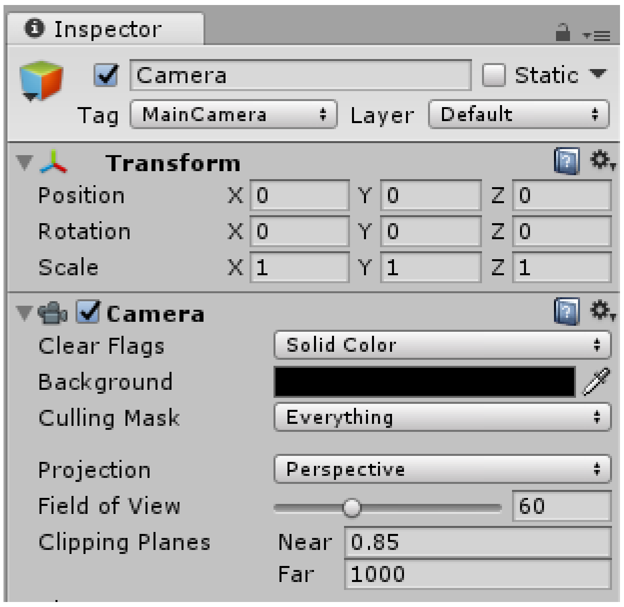
Name	Very Low
Rendering	
Pixel Light Count	0
Texture Quality	Half Res
Anisotropic Textures	Disabled

Per-scene settings

Once you enable the "Virtual Reality Supported" checkbox, the Unity Camera component handles head tracking and stereoscopic rendering. There is no need to replace it with a custom camera to do this.

If your app is targeting HoloLens specifically, there are a few settings that need to be changed to optimize for the device's transparent displays, so your app will show through to the physical world:

1. In the **Hierarchy**, select the **Main Camera**
2. In the **Inspector** panel, set the transform **position** to **0, 0, 0** so the location of the users head starts at the Unity world origin.
3. Change **Clear Flags** to **Solid Color**.
4. Change the **Background** color to **RGBA 0,0,0,0**. Black renders as transparent in HoloLens.
5. Change **Clipping Planes - Near** to the HoloLens recommended 0.85 (meters).


Inspector
Camera
Static
Tag MainCamera
Layer Default

Transform

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

Camera

Clear Flags	Solid Color
Background	■
Culling Mask	Everything
Projection	Perspective
Field of View	60
Clipping Planes	Near 0.85
	Far 1000

If you delete and create a new camera, make sure your camera is **Tagged** as **MainCamera.**

4.2. Review the experience scale

Once you've configured your project as described above, standard Unity game objects (such as the camera) will light up immediately for a seated-scale experience, with the camera's position updated automatically as the user moves their head through the world.

Adding support for Windows Mixed Reality features such as spatial stages, gestures, motion controllers or voice input is achieved using APIs built directly into Unity.

For true **world-scale** experiences on HoloLens that let users wander beyond 5 meters, you'll need new techniques beyond those used for room-scale experiences. One key technique you'll use is to create a spatial anchor to lock a cluster of holograms precisely in place in the physical world, regardless of how far the user has roamed, and then find those holograms again in later sessions.

In Unity, you create a spatial anchor by adding the **WorldAnchor** Unity component to a **GameObject**.

Adding a World Anchor

To add a world anchor, call AddComponent<WorldAnchor>() on the game object with the transform you want to anchor in the real world.

```
WorldAnchor anchor = gameObject.AddComponent<WorldAnchor>();
```

That's it! This game object will now be anchored to its current location in the physical world - you may see its Unity world coordinates adjust slightly over time to ensure that physical alignment. Use persistence to find this anchored location again in a future app session.

Removing a World Anchor

If you no longer want the GameObject locked to a physical world location and don't intend on moving it this frame, then you can just call Destroy on the World Anchor component.

```
Destroy(gameObject.GetComponent<WorldAnchor>());
```

If you want to move the GameObject this frame, you need to call DestroyImmediate instead.

```
DestroyImmediate(gameObject.GetComponent<WorldAnchor>());
```

Moving a World Anchored GameObject

GameObject's cannot be moved while a World Anchor is on it. If you need to move the GameObject this frame, you need to:

1. DestroyImmedaite the World Anchor component
2. Move the GameObject
3. Add a new World Anchor component to the GameObject.

```
DestroyImmediate(gameObject.GetComponent<WorldAnchor>());

gameObject.transform.position = new Vector3(0, 0, 2);

WorldAnchor anchor = gameObject.AddComponent<WorldAnchor>();
```

Handling Locatability Changes

A WorldAnchor may not be locatable in the physical world at a point in time. If that occurs, Unity will not be updating the transform of the anchored object. This also can change while an app is running. Failure to handle the change in locatability will cause the object to not appear in the correct physical location in the world.

To be notified about locatability changes:

Subscribe to the OnTrackingChanged event

Handle the event

The **OnTrackingChanged** event will be called whenever the underlying spatial anchor changes between a state of being locatable vs. not being locatable.

```
anchor.OnTrackingChanged += Anchor_OnTrackingChanged;
```

Then handle the event:

```
private void Anchor_OnTrackingChanged(WorldAnchor self, bool located)

{

       // This simply activates/deactivates this object and all
children when tracking changes

    self.gameObject.SetActiveRecursively(located);

}
```

Sometimes anchors are located immediately. In this case, this isLocated property of the anchor will be set to true when AddComponent<WorldAnchor>() returns. As a result, the OnTrackingChanged event will not be triggered. A clean pattern would be to call your OnTrackingChanged handler with the initial IsLocated state after attaching an anchor.

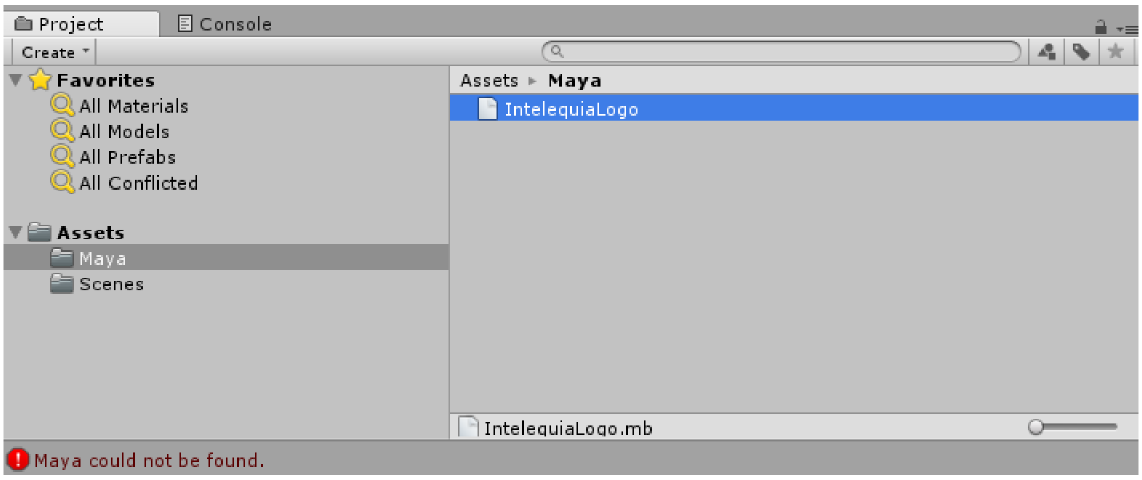
```
Anchor_OnTrackingChanged(anchor, anchor.isLocated);
```

4.3. Import Hologram

Unity natively imports Maya files. To get started, simply place your .mb or .ma file in your project's Assets folder. When you switch back into Unity, the scene is imported automatically and will show up in the Project view.

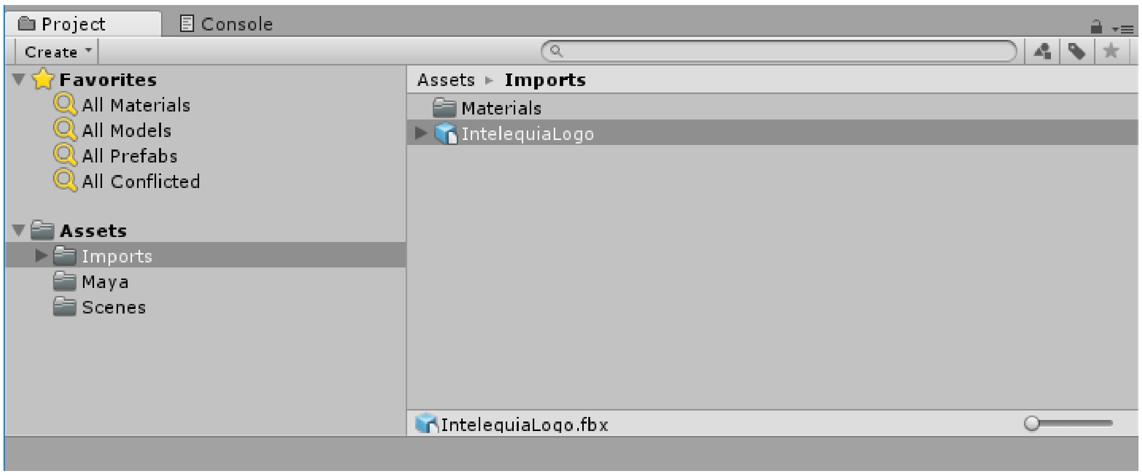
To see your model in Unity, simply drag it from the Project View into the Scene View or Hierarchy View.

In order to import Maya .mb and .ma files, **you need to have Maya installed on the machine** you are using Unity to import the .mb/.ma file. Maya 8.0 and up is supported.



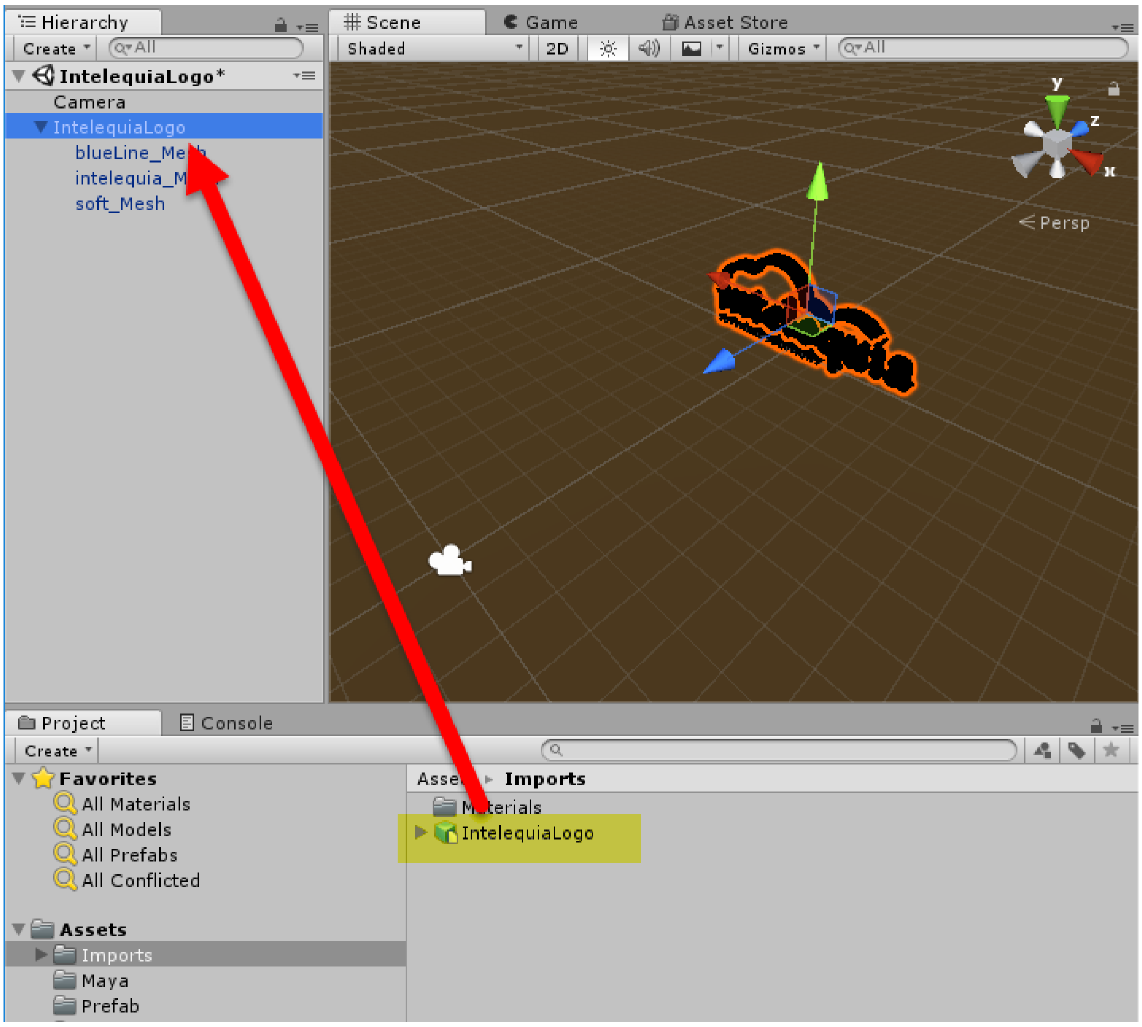
Maya could not be found.

If you don't have Maya installed on your machine but want to import a Maya file from another machine, you can export to fbx format, which Unity imports natively.

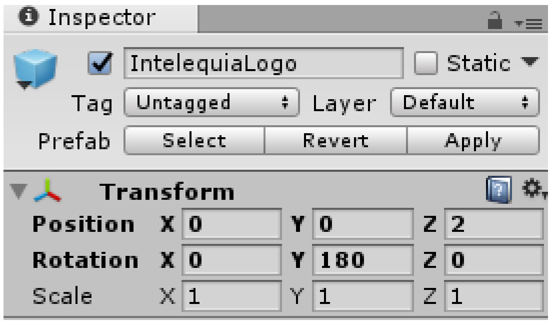


Once exported Place the fbx file in the Unity project folder. Unity will now automatically import the fbx file.

Now you can drag and drop your fbx mesh into scene hierarchy.

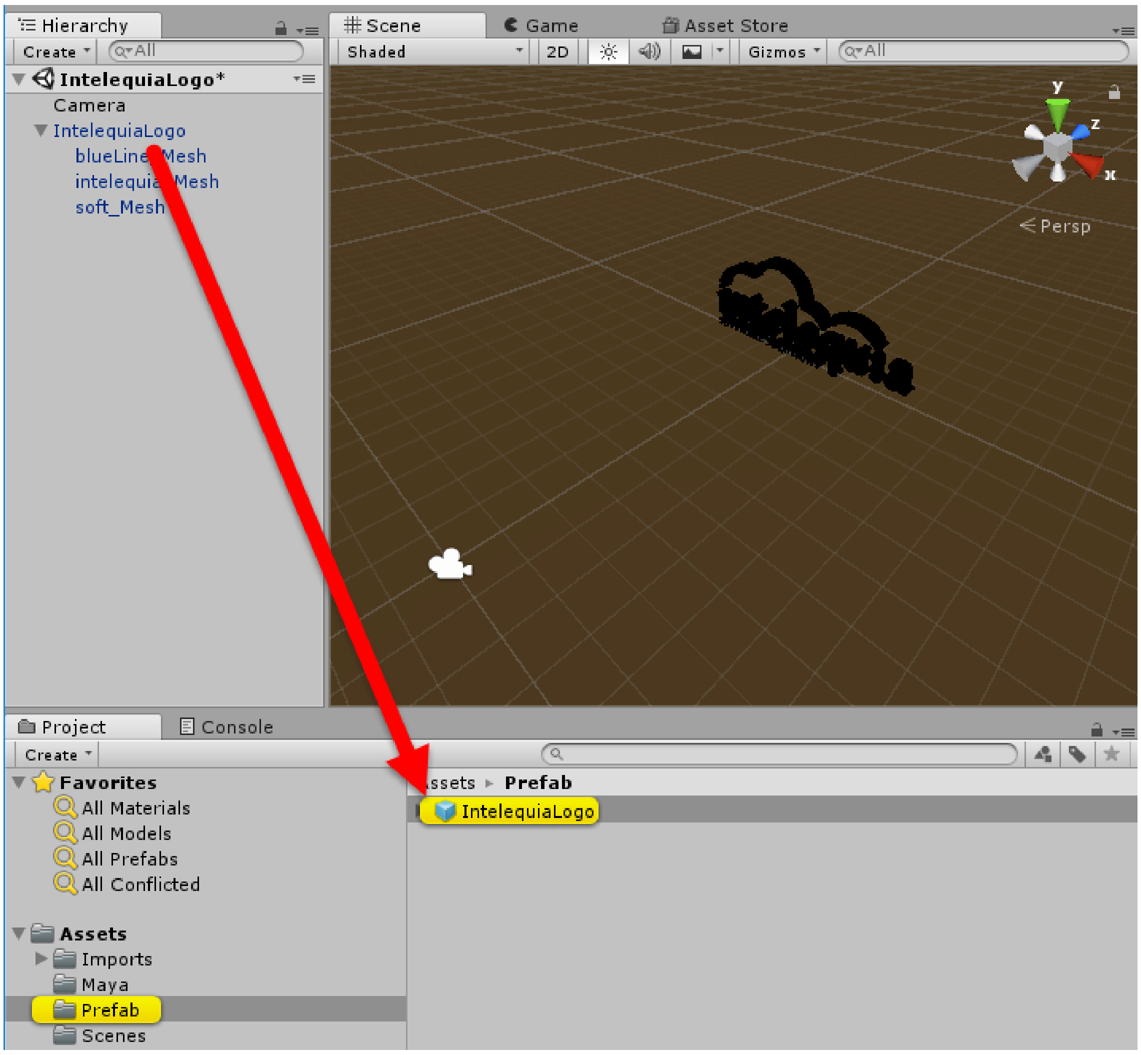


1. In the **Inspector** find the **Transform** component and change **Position** to (**X**: 0, **Y**: 0, **Z**: 2). *This positions the logo 2 meters in front of the user's starting position.*
2. In the **Transform** component, change **Rotation** to (**X**: 0, **Y**: 180, **Z**: 0) and change **Scale** to (**X**: 1, **Y**: 1, **Z**: 1). *This maintain the original scale of the logo.*
3. To save the scene changes, select **File > Save Scene**.



Position X 0 Y 0 Z 2
Rotation X 0 Y 180 Z 0
Scale X 1 Y 1 Z 1

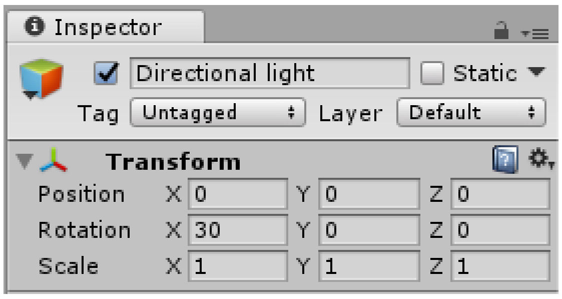
And later you can create a new prefab object if you drag and drop the scene object to any project folder.



Finally to complete the scene and that we can see the logo with its colors we must introduce some basic lighting.

1. In the top left corner of the **Hierarchy** panel, select the Create dropdown and choose **Light** > **Directional Light**.

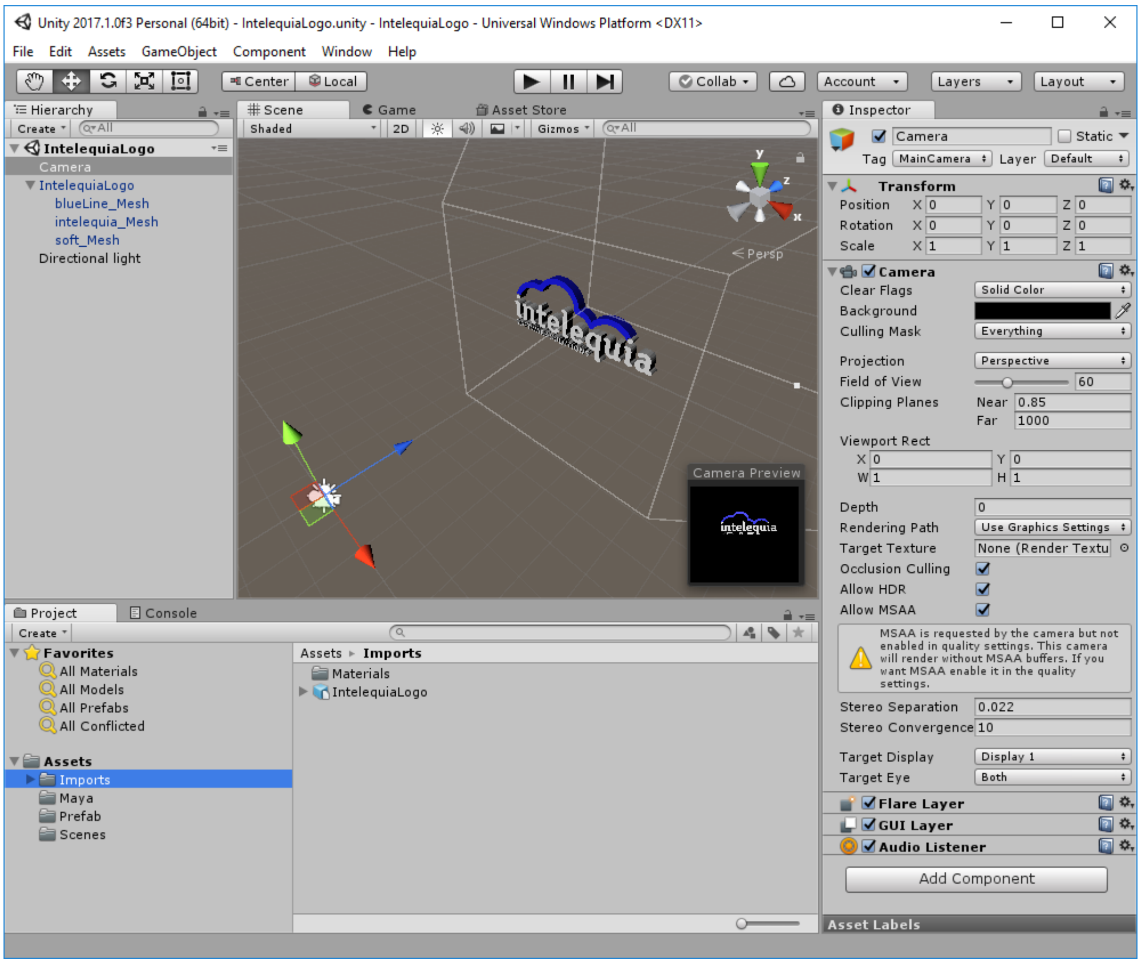
2. Select the newly created **Directional Light** in the **Hierarchy** panel
3. In the **Inspector** find the **Transform** component and change **Position** to (X: 0, Y: 0, Z: 0).
4. In the **Transform** component change **Rotation** to (X: 30, Y: 0, Z: 0).
5. To save the scene changes, select File > Save Scene.



Directional light
Tag Untagged Layer Default

Transform			
Position	X 0	Y 0	Z 0
Rotation	X 30	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Now your scene is ready to be exported to Hololens.



Camera
Tag MainCamera Layer Default

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

Camera	
Clear Flags	Solid Color
Background	■
Culling Mask	Everything
Projection	Perspective
Field of View	60
Clipping Planes	Near 0.85
	Far 1000
Viewport Rect	X 0 Y 0
	W 1 H 1
Depth	0
Rendering Path	Use Graphics Settings
Target Texture	None (Render Textu
Occlusion Culling	✓
Allow HDR	✓
Allow MSAA	✓
Stereo Separation	0.022
Stereo Convergence	10
Target Display	Display 1
Target Eye	Both

MSAA is requested by the camera but not enabled in quality settings. This camera will render without MSAA buffers. If you want MSAA enable it in the quality settings.

✓ Flare Layer
✓ GUI Layer
✓ Audio Listener

4.4. Build and deploy to emulator from Visual Studio

We are now ready to compile our project to Visual Studio and deploy to our target device.

Export to the Visual Studio solution

1. Open **File > Build Settings** window.
2. Click **Add Open Scenes** to add the scene.
3. Change **Platform** to **Windows Store** and click **Switch Platform.**
4. In **Windows Store** settings ensure, **SDK** is **Universal 10.**
5. For Target device, leave to **Any Device** for occluded displays or switch to **HoloLens**.
6. **UWP Build Type** should be **D3D**.
7. **UWP SDK** could be left at Latest installed.
8. Check **Unity C# Projects** under Debugging.
9. Click **Build**.
10. In the file explorer, click **New Folder** and name the folder "**App**".
11. With the **App** folder selected, click the **Select Folder** button.
12. When Unity is done building, a Windows File Explorer window will appear.
13. Open the **App** folder in file explorer.
14. Open the generated Visual Studio solution (MixedRealityIntroduction.sln in this example)

Compile the Visual Studio solution

Finally, we will compile the exported Visual Studio solution, deploy it, and try it out on the device.

1. Using the top toolbar in Visual Studio, change the target from **Debug** to **Release** and from **ARM** to **X86**.

The instructions differ for deploying to a device versus the emulator. Follow the instructions that match your setup.

Deploy to Emulator

2. Click on the arrow next to the Device button, and from drop down select **HoloLens Emulator**.
3. Click **Debug > Start without debugging**.