AS324903

Revit to Real Time: Importing BIM into XR with 3ds Max and Unity

Christopher Diggins
Head of Research, VIM

Learning Objectives

- import 3D models and BIM data from Revit into 3ds Max
- use 3ds Max to prepare and optimize models and scenes for import into Unity
- compile a Unity game for desktop or Magic Leap
- use the Forge Design Automation API to run automated scripts

Description

Architectural plans created with Revit software present a unique set of challenges when importing data into real-time interactive experiences using Unity or other game engines, whether targeting desktop, mobile, web, or XR—augmented reality (AR), virtual reality (VR), or mixed reality (MR). This course will discuss different methods used to export 3D models and BIM (Building Information Modeling) data out of Revit and prepare it with 3ds Max software for importation into a Unity game running on the Magic Leap. We'll survey various tools and techniques used to optimize and prepare the models for efficient experiences on low-power AR or VR devices, while maintaining access to the BIM data right through to the final Unity game experience. You'll also learn how you can use the Forge Design Automation API to automate this process with very little programming.

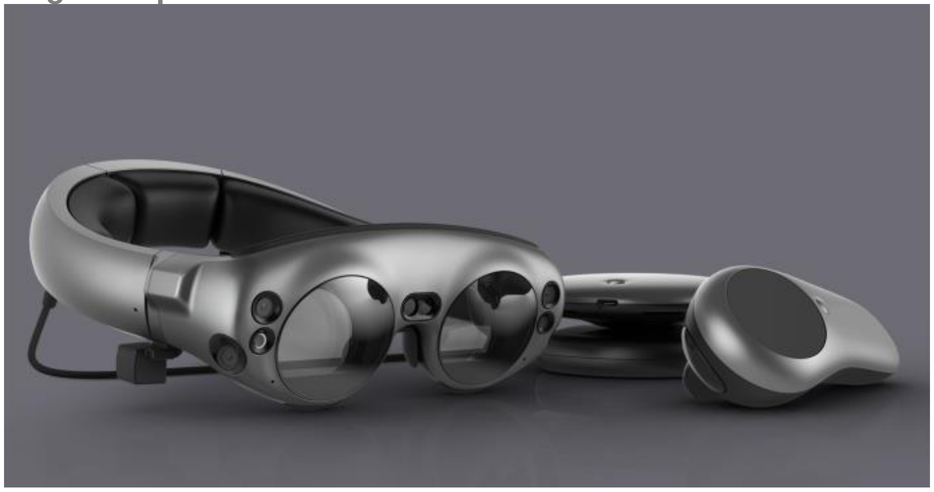
Speaker

Christopher Diggins is the head of research at VI. He leads a team that is currently focused on computational geometry, optimizing access to large real-world design data for use on low-power devices, and automated construction of game-ready assets with semantics from reality capture data.

Introduction

This presentation covers a workflow for bringing Revit geometry and BIM data into Unity experiences via 3ds Max to create interactive mixed reality experiences on the Magic Leap One. Several of the tools and techniques covered are generally applicable to other types of XR experiences as well as applications targeting mobile and desktop computing platforms.

About the Magic Leap One



THE MAGIC LEAP ONE

The Magic Leap One (https://www.magicleap.com/magic-leap-one) is a lightweight wearable computer that includes AR glasses, speakers, camera, sensors, and controller designed to provide an immersive mixed reality experience. The Magic Leap One superimposes 3D computer generated images over real-world objects by tracking and creating a continuously updated 3D representation of the environment.

About Game Engines

Game engines are quite different from 3D art and CAD tools, and more closely related to software development environments. Rather than producing a 3D scene which is then transformed into a file, animation, or rendering, a game is a software application, optimized to display 2D or 3D graphics in real-time on different platforms from desktop computers to mobile devices to specialized VR or AR devices.

While it is possible to write an XR experience from scratch for a specific platform using that platforms native API, game engines allow developers to develop applications more quickly which target multiple platforms. For example, at VIM by using Unity we share a lot of the same code base between our Desktop application and the Magic Leap device.

From Revit to 3ds Max

Common Workflows

The three most common workflows for bringing Revit data into 3ds Max are the following:

- **Revit to FBX –** poor tessellation, slow, BIM data not easily accessible, supports RVT linked files
- **Revit to DWF –** good tessellation, instances, no BIM data, does not support linked RVT files
- **3ds Max Revit Importer –** medium good tessellation, custom control, BIM data, doesn't support linked RVT files

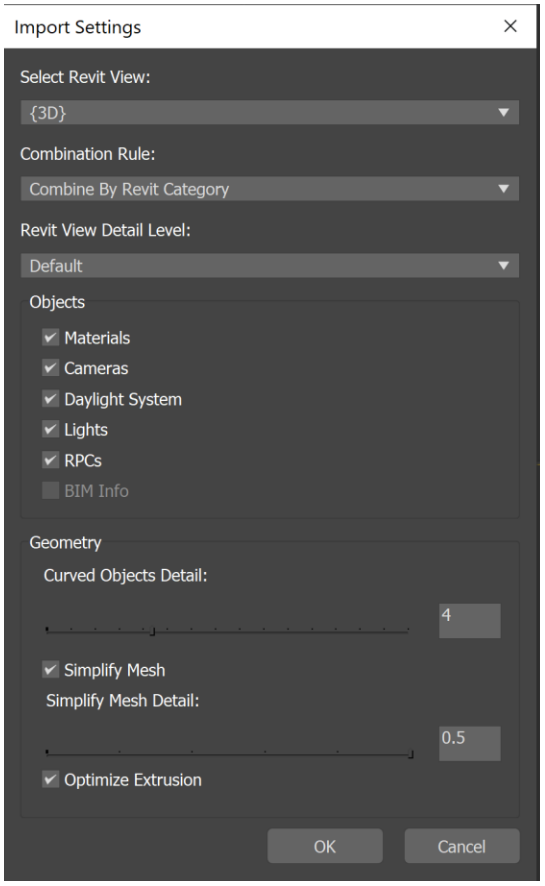
The third option works is the one chosen for the workflow discussed here, because it is easier to access the BIM data and to control the level of detail of curved objects.

A Note about File Link

One can use the "File Link Manager" https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2020/ENU/3DSMax-Data-Exchange/files/GUID-81F21D8B-006B-44A9-AF34-C6742D8C09D4-htm.html] to import RVT files into 3ds Max but because we are ultimately going to be using 3ds Max in an automation context on the cloud, it is much simpler to just "import".

Importing the Revit File into 3ds Max

For a video walkthrough of this process see https://knowledge.autodesk.com/support/3ds-max/getting-started/caas/screencast/Main/Details/7423e85b-143f-42f9-951e-49692c51d24c.html.


Import Settings

Select Revit View: {3D}

Combination Rule: Combine By Revit Category

Revit View Detail Level: Default

Objects
- ✓ Materials
- ✓ Cameras
- ✓ Daylight System
- ✓ Lights
- ✓ RPCs
- BIM Info

Geometry

Curved Objects Detail: 4

- ✓ Simplify Mesh

Simplify Mesh Detail: 0.5

- ✓ Optimize Extrusion

OK Cancel

REVIT IMPORT DIALOG

The Revit import dialog is accessible from “File menu > Import > Select File To Import dialog > Files Of Type drop-down list > Choose Revit importer (*.RVT)”. Documentation is here: https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/3DSMax-Data-Exchange/files/GUID-1B4591EE-A040-4543-9EBC-D02A7EBF2E79-htm.html. Automating the import of the Revit file can be done via the following MAXScript command:

importFile <filename_string> #noPrompt

and the options can be controlled by using the “RevitImporterSetOption” function (https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/3DSMax-MAXScript/files/GUID-C6560DDD-3117-4F8B-9280-F990D240F298-htm.html).

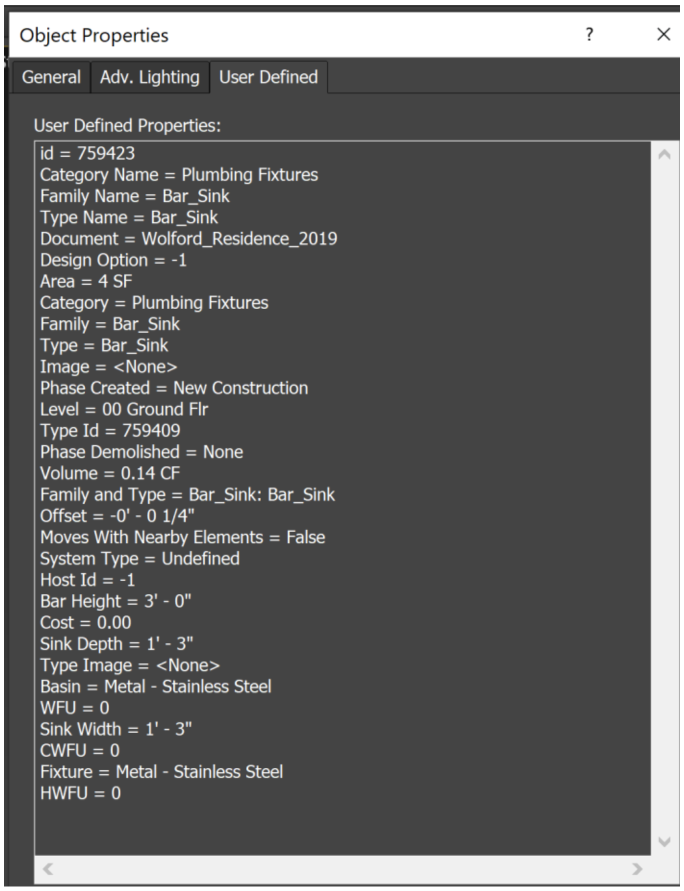
If Possible, Avoid a Revit File Upgrade to newer Version

If the version of the Revit file is older then the 3ds Max version a Revit import is going to be considerably slower, especially for larger models, because the Revit file will have to be upgraded to the newer Revit model.

Accessing the BIM Data in 3ds Max

The BIM data from Revit is accessible on objects as “User Defined Object Properties” when Revit files are imported into 3ds Max. This is effectively an array of strings, separated by newline characters, that have a key / value pair. You can access the Object Properties dialog (which is unfortunately modal) in 3ds Max via right clicking on any object in the scene, or in the scene explorer.

.



```
id = 759423
Category Name = Plumbing Fixtures
Family Name = Bar_Sink
Type Name = Bar_Sink
Document = Wolford_Residence_2019
Design Option = -1
Area = 4 SF
Category = Plumbing Fixtures
Family = Bar_Sink
Type = Bar_Sink
Image = <None>
Phase Created = New Construction
Level = 00 Ground Flr
Type Id = 759409
Phase Demolished = None
Volume = 0.14 CF
Family and Type = Bar_Sink: Bar_Sink
Offset = -0' - 0 1/4"
Moves With Nearby Elements = False
System Type = Undefined
Host Id = -1
Bar Height = 3' - 0"
Cost = 0.00
Sink Depth = 1' - 3"
Type Image = <None>
Basin = Metal - Stainless Steel
WFU = 0
Sink Width = 1' - 3"
CWFU = 0
Fixture = Metal - Stainless Steel
HWFU = 0
```

OBJECT PROPERTIES DIALOG IN 3DS MAX

Accessing BIM Data using MAXScript

You can access the BIM data from MAXScript using the “getUserProp <key>” name which will return the associated value if present, or the predefined MAXScript value of undefined. See: https://help.autodesk.com/view/3DSMAX/2017/ENU/?guid=__files_GUID_AF1F51D4_449B_4C4D_9F58_85DB145BC0B9_htm

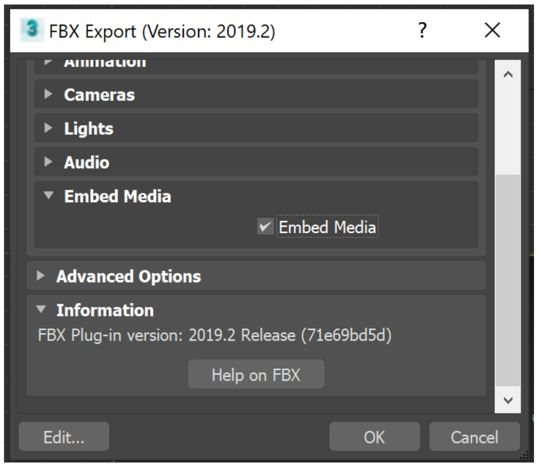
Convert to Standard Material using Scene Converter

When importing FBX files exported from 3ds Max into Unity the easiest way to deal with Materials is to convert them to standard materials, and to convert files to texture maps. This can be done via the Scene Converter (https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/3DSMax-Manage-Scenes/files/GUID-6AB71647-E71C-4E44-872E-603D3FFB7F8A-htm.html).

Unfortunately, because a dialog will always display when calling “ConvertScene”, the Scene Converter cannot be scripted using 3dsmaxbatch.exe or the Design Automation API. This is because the dialog UI will cause the application to hang waiting for user input. Instead we wrote our own MAXScript (available here) for converting the scene.

Exporting to FBX

When exporting to FBX, be sure to embed the textures in the media.


FBX Export (Version: 2019.2)

Animation
Cameras
Lights
Audio
Embed Media
✓ Embed Media
Advanced Options
Information
FBX Plug-in version: 2019.2 Release (71e69bd5d)

FBX EXPORT OPTIONS DIALOG: CHECK EMBED MEDIA

Using the 3ds Max Forge Design Automation API

The 3ds Max Design Automation API on Forge, is effectively a headless 3ds Max instance running on the cloud that can be used to execute scripts and plug-ins. It currently costs 6 credits an hour to run.

The prerequisites for using the sample here is:

- Forge developer account
- Node.JS – a standalone JavaScript engine

A great example to start from is: https://github.com/Autodesk-Forge/design.automation.3dsmax-nodejs-basic. This example uploads a sample 3ds Max file to the API, runs a simple MAXScript that exports it to an FBX, and returns the FBX file.

Only a small modification of this example was needed to enable it to open a Revit file, convert the Autodesk materials to standard materials, and the Autodesk bitmaps to standard bitamps, and then export the FBX file. We posted the modified sample on Github here: https://github.com/vimaec/au2019-forge-revit-to-fbx.

The primary steps for this workflow are this:

- Create a Forge account and a Forge app
- Download the project Github repository.
- Customize with your application id, secret key, and unique object storage key
- Run the three steps using Node.JS:
  - Create and upload app - `node .\createAndUploadApp.js`
  - Create work item – `node .\createActivity.js`
  - Execute task – `node .\executeTask`

If you want to further customize the MAXScript in the handlebars template here if desired.

Bringing 3D Data into Unity

The process of bringing FBX data into Unity is demonstrated in this video.

Quick Introduction to Unity

When using Unity, you will interact with several different pieces of software:

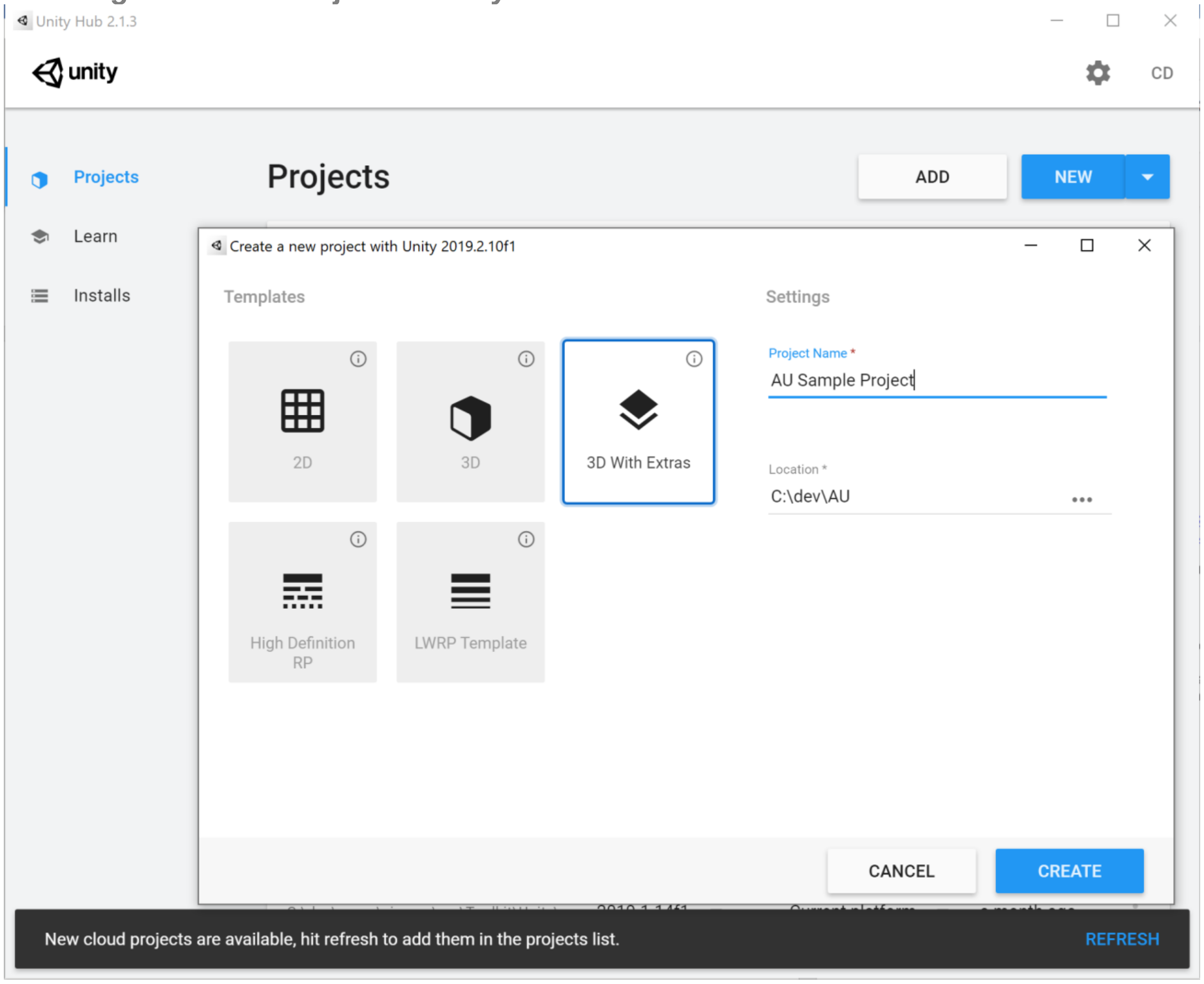
- **Unity Hub** – used for installing and managing different versions of the editor, and managing your Unity
- **Unity Editor** – this is the environment in which you author and can debug your Unity application.
- **Visual Studio** – used for editing and debugging scripts
- **Art tools (e.g. 3ds Max or Maya)** – In many scenarios you will need one or more art tools to author custom 2D or 3D assets, such as sprites, textures, models, animations, etc.
- **Your Unity application** – you can build and run your application to test an optimized version of the application outside of the Unity editor environment.

Unity Concepts

The following concepts are key for understanding the Unity documentation:

- **Projects** – A collection of all files needed to build a game
- **Assets** – A model, script, or other component that is used to build a game
- **Game Object** – a node in the scene graph with one or more components
- **Component** – an entity attached to a game object (e.g. script, transform, mesh, renderer)
- **Script** – a C# file containing a class derived from MonoBehavior
- **Plugin** – a DLL containing C# code (Standard or .NET 4.6)
- **Editor script/plugin** – a C# file or DLL that is inside a folder named “Editor”

Creating a New 3D Project in Unity



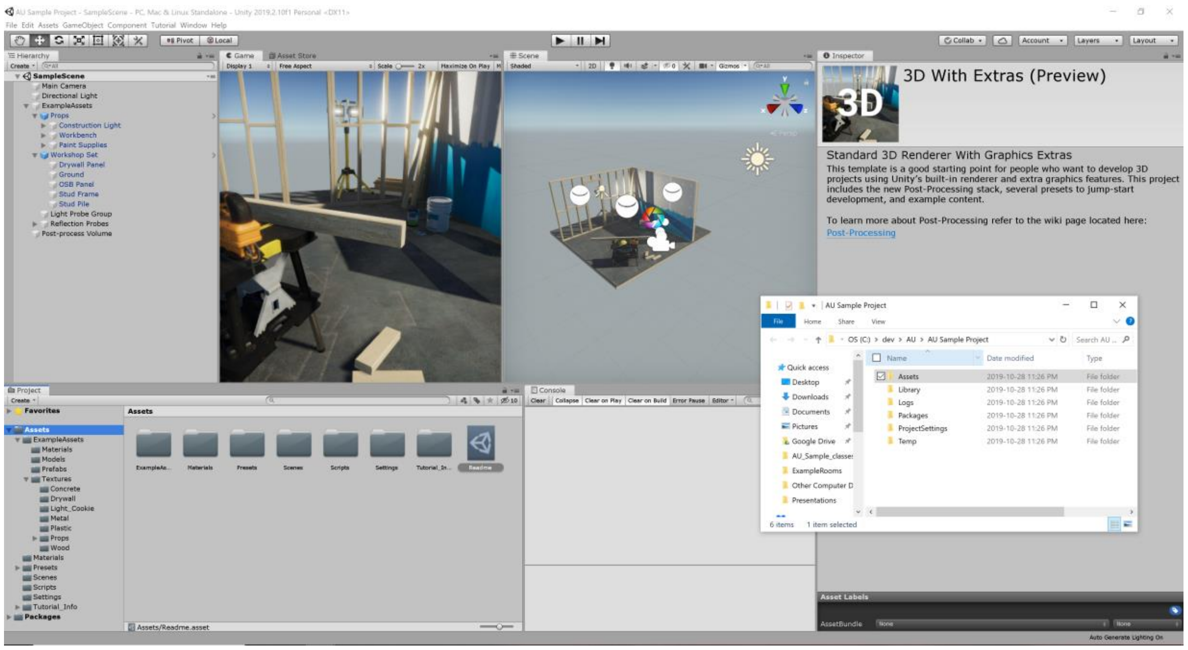
Project Name *
AU Sample Project

Location *
C:\dev\AU

New cloud projects are available, hit refresh to add them in the projects list.

Projects are added and managed via the Unity hub application installed on your computer. For your first introduction to 3D application development with Unity I suggest opening the "3D with Extras" template.

The 3D with Extras Template


AU Sample Project - SampleScene - PC, Mac & Linux Standalone - Unity 2019.2.10f1 Personal <DX11>
File Edit Assets GameObject Component Tutorial Window Help
Hierarchy
Game
Scene
Inspector
Project
Console
AU Sample Project

3D With Extras (Preview)

Standard 3D Renderer With Graphics Extras

This template is a good starting point for people who want to develop 3D projects using Unity's built-in renderer and extra graphics features. This project includes the new Post-Processing stack, several presets to jump-start development, and example content.

To learn more about Post-Processing refer to the wiki page located here: Post-Processing

Name	Date modified	Type
Assets	2019-10-28 11:26 PM	File folder
Library	2019-10-28 11:26 PM	File folder
Logs	2019-10-28 11:26 PM	File folder
Packages	2019-10-28 11:26 PM	File folder
ProjectSettings	2019-10-28 11:26 PM	File folder
Temp	2019-10-28 11:26 PM	File folder

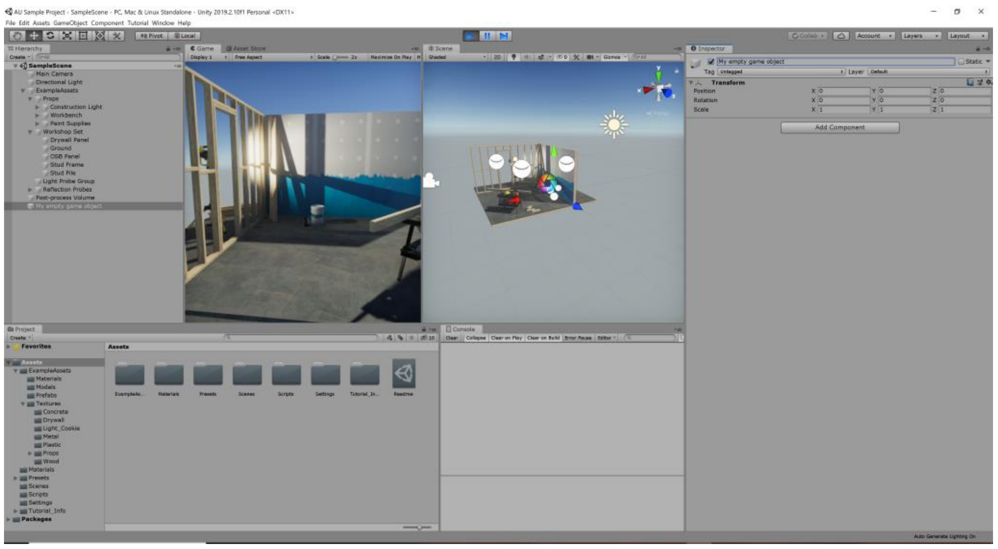
THE 3D WITH EXTRAS TEMPLATE IN UNITY

When starting from the 3D With Extras template you are given a new project with a default scene containing several assets ready to go.

Notice that the project is reflected within a folder structure on your file system that is continuously monitored by the Unity project. When files are added, removed, or removed in the asset browser the changes are reflected in the file system, and vice versa. You can access the file system by right clicking on an asset of on the project explorer and choose “Show in Explorer”.

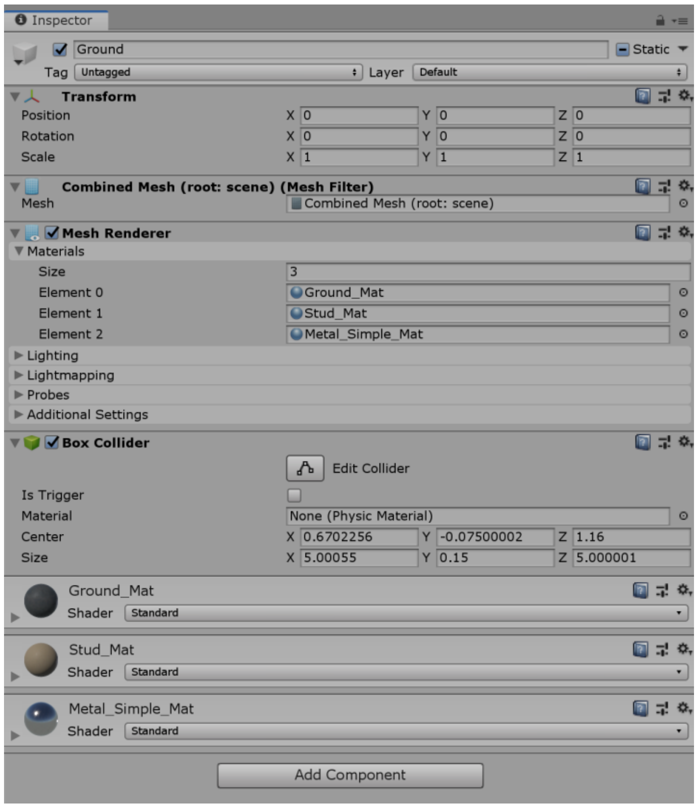
Game Objects and Components

Every object in a scene in a Unity project is associated with a “game object” (https://docs.unity3d.com/Manual/GameObjects.html). Game objects provide a name, and a couple of optional properties (such as tag, layer, and a static/dynamic flag) for an object in the scene.



ADDING AN EMPTY GAME OBJECT TO THE SCENE

Components



Inspector

Ground — Static
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

Combined Mesh (root: scene) (Mesh Filter)

Mesh: Combined Mesh (root: scene)

Mesh Renderer

Materials

Size	3
Element 0	Ground_Mat
Element 1	Stud_Mat
Element 2	Metal_Simple_Mat

Lighting
Lightmapping
Probes
Additional Settings

Box Collider

Edit Collider

Is Trigger			
Material	None (Physic Material)		
Center	X 0.6702256	Y -0.07500002	Z 1.16
Size	X 5.00055	Y 0.15	Z 5.000001

Ground_Mat
Shader Standard

Stud_Mat
Shader Standard

Metal_Simple_Mat
Shader Standard

Add Component

THE GROUND PLANE GAME OBJECT WITH COMPONENTS

A game object is effectively a holder of components, and always contains at least one component: the “transform” component. The transform component manages the child-parent relationships. Every component has a backlink to the game object with which it is associated.

Common examples of components include:

- **Transform** – manages the position, rotation, and scale and the parent and child relationships of the object (https://docs.unity3d.com/Manual/class-Transform.html)
- **Mesh filter** – manages access to a 3D mesh (which is a separate non-game object) (https://docs.unity3d.com/Manual/class-MeshFilter.html)
- **Mesh renderer** – used to associate a material and render properties for drawing a mesh (https://docs.unity3d.com/Manual/class-MeshRenderer.html)
- **Collider** – used for hit testing and physics simulations (see https://docs.unity3d.com/Manual/CollidersOverview.html)
- **Light** – illuminates the scene https://docs.unity3d.com/Manual/Lighting.html
- **Camera** – providing a point of view for the user (https://docs.unity3d.com/Manual/CamerasOverview.html)
- **LOD Group** – used for managing lower resolution representation of objects when far from the camera (https://docs.unity3d.com/Manual/class-LODGroup.html)
- **Script** – a class derived from MonoBehavior that has hooks for responding to different events like object creation, or frame update (see https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html)

Unity Scripting

There is a lot that can be said about Unity scripting. But the following are a few important things for a C# developer to realize:

Plug-ins are not different from Scripts

There are standalone C# files or code can be bundled in a DLL (called a plug-in). Both approaches are equivalent and have access to the same APIs. The difference is that DLLs are decompiled first.

Unity Compiler

Unity compiles C# scripts or recompiles Managed assemblies using either Mono or a custom backend called IL2CPP (https://docs.unity3d.com/Manual/IL2CPP.html). This means that you should assure that your code is compliant with the same version of Windows .NET Framework or Windows Standard that Unity is supporting (see https://docs.unity3d.com/Manual/CSharpCompiler.html and https://docs.unity3d.com/2019.1/Documentation/Manual/dotnetProfileSupport.html).

Unity Debugger

Unity provide a debugger that can be added to Visual Studio. See: https://docs.unity3d.com/Manual/ScriptingToolsIDEs.html.

Unity Editor Scripts

You can write new scripts/plug-ins for execution in the Unity Editor by putting the C# files or DLLs in a folder within your project with the magic name “Editor”

Importing FBX Assets into the Project

Before you can use a 3D model in a Unity scene have to first import the 3D asset into the Unity project (https://docs.unity3d.com/Manual/ImportingModelFiles.html). This process can be slow for extremely large FBX files with lots of objects. An FBX file can contain multiple 3D models, materials, textures, and animations.

Set Colliders On in Asset Import Settings

Make sure that “Colliders” is on in the asset import settings tab. You may have to reimport the model afterwards. This facilitates raycast testing so that you can create a point and click script.

Instantiating the FBX Assets as a Prefab

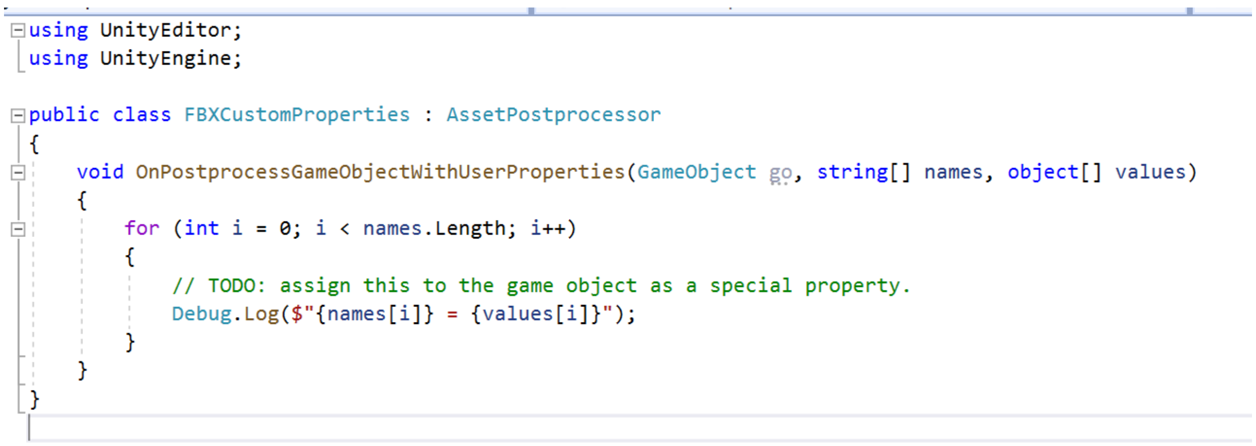
Once you have imported an FBX asset into the project, assuming it contains at least one mesh, it can be dragged into the scene to create a “prefab” instance associated with a game object. A prefab is a game object associated with multiple configured components, property values, and child game objects. For more information see: https://docs.unity3d.com/Manual/Prefabs.html.

Unpacking Textures

After importing an FBX file exported from 3ds Max that contains embedded textures, those texture need to be unpacked before the materials will be recognized and rendered correctly.

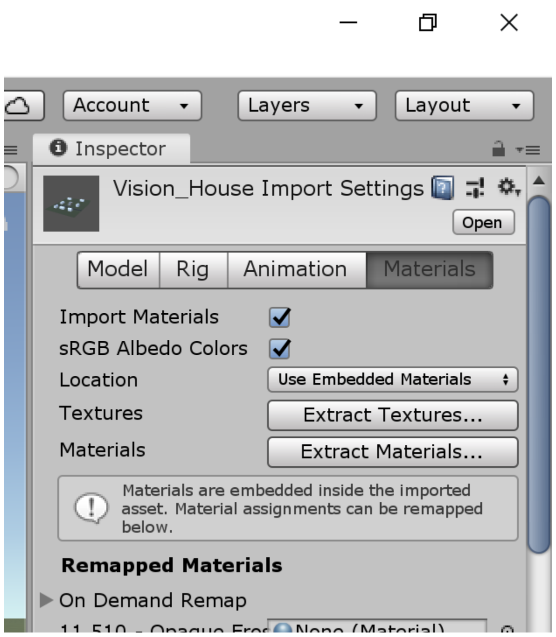
Accessing the BIM Data

When the FBX file is exported from 3ds Max (after import via Revit) the BIM data is accessible to Unity, and can be extracted by writing an `AssetPostProcessor` script (https://docs.unity3d.com/ScriptReference/AssetPostprocessor.html) which must be put in a folder named “Editor” in the project.



```
using UnityEditor;
using UnityEngine;

public class FBXCustomProperties : AssetPostprocessor
{
    void OnPostprocessGameObjectWithUserProperties(GameObject go, string[] names, object[] values)
    {
        for (int i = 0; i < names.Length; i++)
        {
            // TODO: assign this to the game object as a special property.
            Debug.Log($"{names[i]} = {values[i]}");
        }
    }
}
```


Vision_House Import Settings

Import Materials ✓

sRGB Albedo Colors ✓

Location: Use Embedded Materials

Textures: Extract Textures...

Materials: Extract Materials...

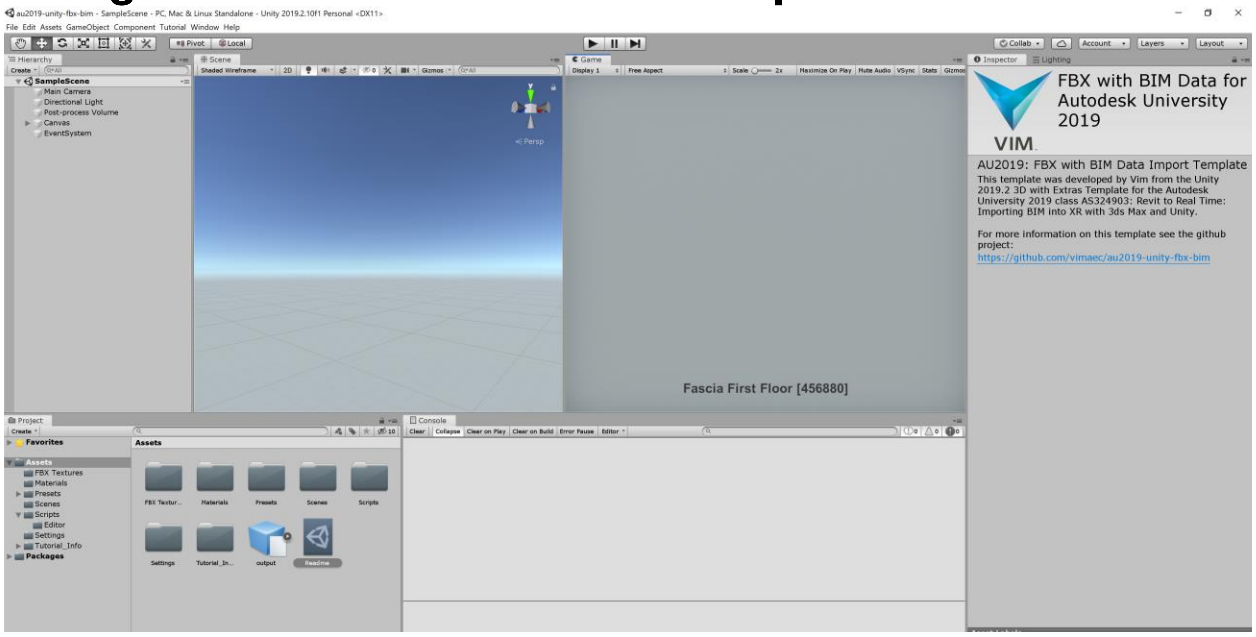
Materials are embedded inside the imported asset. Material assignments can be remapped below.

Remapped Materials

On Demand Remap

THE EXTRACT TEXTURES BUTTON IN THE IMPORT SETTINGS TAB

Using the FBX with BIM Data Template


FBX with BIM Data for Autodesk University 2019

AU2019: FBX with BIM Data Import Template

This template was developed by Vim from the Unity 2019.2 3D with Extras Template for the Autodesk University 2019 class AS324903: Revit to Real Time: Importing BIM into XR with 3ds Max and Unity.

For more information on this template see the github project:
https://github.com/vimaec/au2019-unity-fbx-bim

Fascia First Floor [456880]

[HTTPS://GITHUB.COM/VIMAEC/AU2019-UNITY-FBX-BIM](https://github.com/vimaec/au2019-unity-fbx-bim)

We've [posted on Github a template project for this course](https://github.com/vimaec/au2019-unity-fbx-bim) that contains an enhanced FBX Import script for retrieving, storing, and viewing BIM data and object names. It is based on 3D with Extras Template shipped with Unity 2019.2

- Adds three scripts:
    - **BimData** – Holds BIM data as list of strings in the form of “key = value”
    - **BimDataClicker** – Shows BIM data and object name depending where mouse is
    - **PostProcessFBX** – Editor script customizes FBX file load
        - Add BimData components to objects populated from 3ds Max user defined properties
        - Makes object static
        - Assure colliders are created
- Adds a GUI Text label – to display object name
- Customizes the light set-up – avoid light baking

Creating a Magic Leap Experience

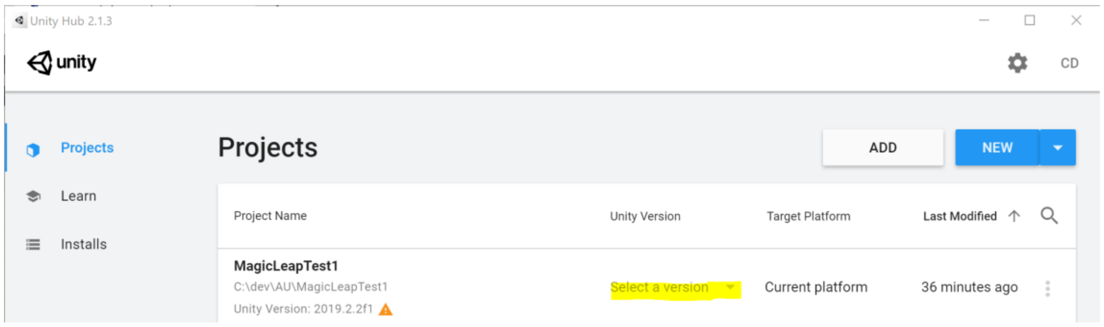
What are the Magic Leap Capabilities

The Magic Leap is truly an advanced wearable computer, and the documentation contains a number of Unity samples that demonstrate the different capabilities of the device:

- Track eye position
- Tracking hand position and pose
- Record and playback audio
- Detect planar surfaces
- Take pictures
- Record video
- Light tracking
- Image tracking and recognition
- Meshing

Using the Unity Hub to Add and Upgrade Project

The Unity Hub is used for managing different projects and editor versions. Unity versions are upgraded very frequently, so when you add a project created for a Unity version that you don’t have locally you can’t open it until you choose a version to upgrade to.



MagicLeapTest1
C:\dev\AU\MagicLeapTest1
Unity Version: 2019.2.2f1

Select a version

Current platform

36 minutes ago

WHEN LOADING A PROJECT FROM AN OLDER VERSION, CHOOSE THE UNITY VERSION

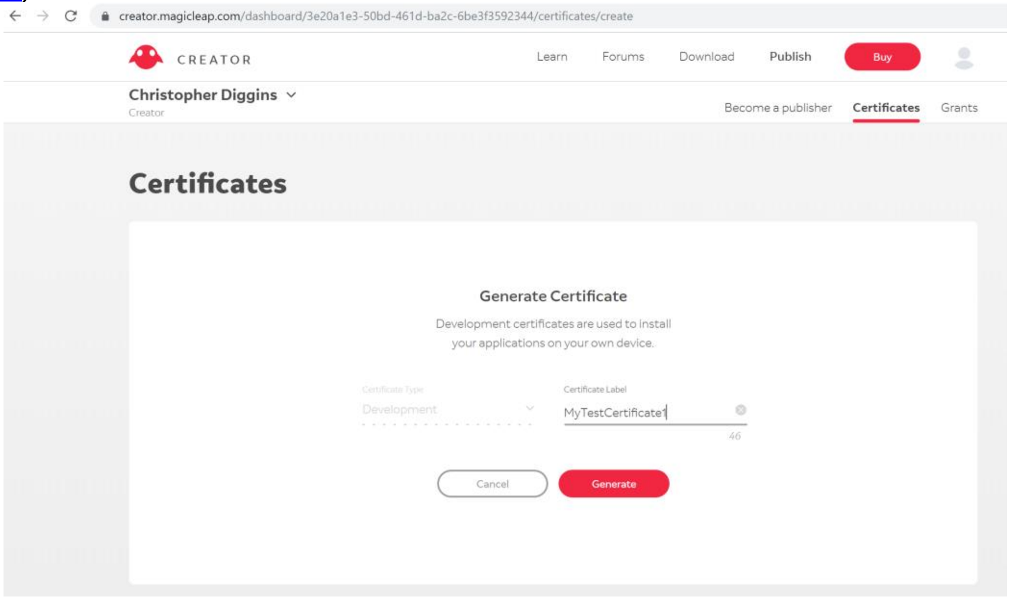
Building a Magic Leap Project for Unity

Download the Magic Leap SDK (aka Lumin SDK)

Download the Magic Leap SDK (https://creator.magicleap.com/home)

Creating a Certificate

Certificates are used to sign applications and are required to run Magic Leap applications and install them ono a Magic Leap device (https://creator.magicleap.com/learn/guides/developer-certificates),



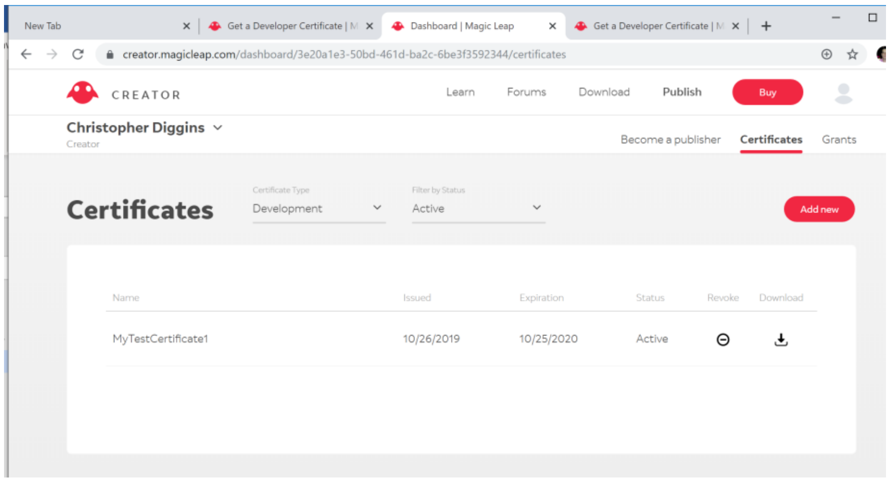
Development certificates are used to install your applications on your own device.

Certificate Type: Development

Certificate Label: MyTestCertificate1

46

GENERATING A DEVELOPER CERTIFICATE FOR SIGNING YOUR APP


New Tab
Get a Developer Certificate | M
Dashboard | Magic Leap
Get a Developer Certificate | M
creator.magicleap.com/dashboard/3e20a1e3-50bd-461d-ba2c-6be3f3592344/certificates
CREATOR
Learn
Forums
Download
Publish
Buy
Christopher Diggins
Creator
Become a publisher
Certificates
Grants

Certificates

Certificate Type: Development

Filter by Status: Active

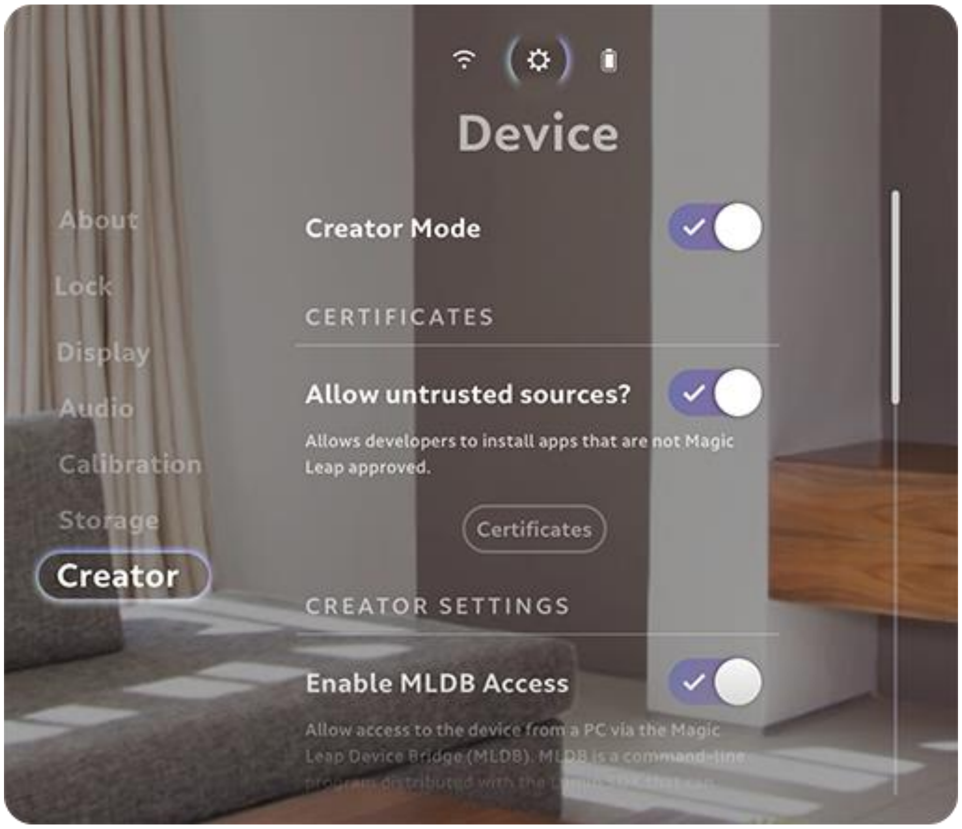
Add new

Name	Issued	Expiration	Status	Revoke	Download
MyTestCertificate1	10/26/2019	10/25/2020	Active	⊖	⤓

Downloading the Certificate Once it is Ready on the Server

Enabling Developer Mode

When installing local apps on Magic Leap One during your development cycle, you must enable Developer mode and allow untrusted sources (https://creator.magicleap.com/learn/guides/develop-device-setup):



Creator Mode ✓

Allow untrusted sources? ✓

Allows developers to install apps that are not Magic Leap approved.

Enable MLDB Access ✓

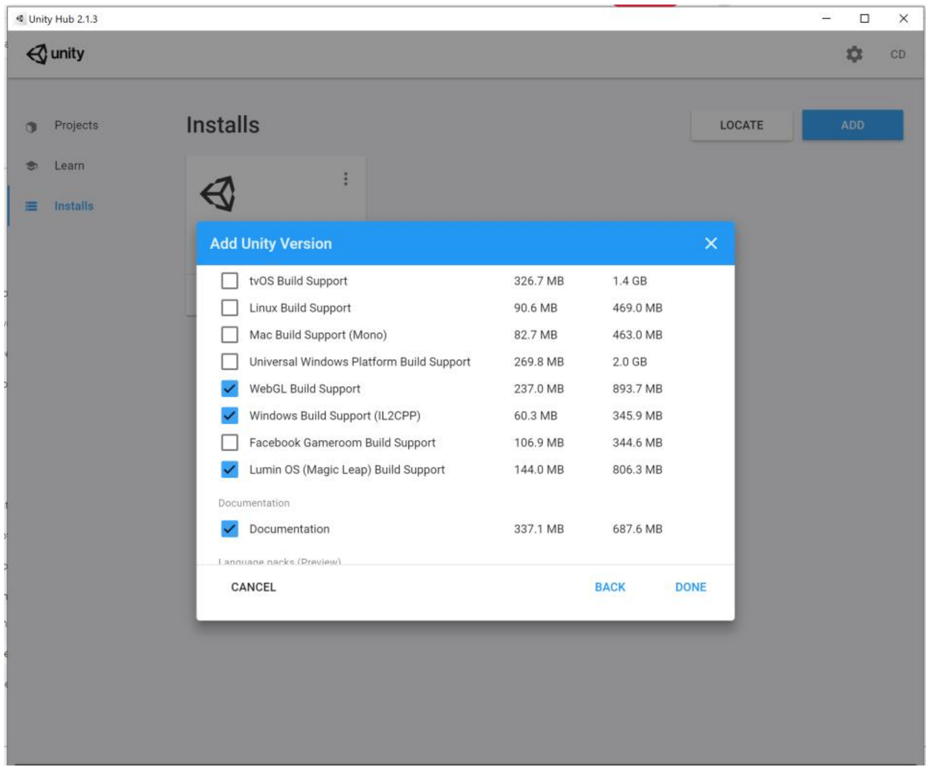
Allow access to the device from a PC via the Magic Leap Device Bridge (MLDB). MLDB is a command-line program distributed with the Lumin SDK that can

Enabling Developer Mode on the Magic Leap One

Installing and Running the Unity Magic Leap Template

The best place for getting started with Magic Leap development on Unity is via the Unity Magic Leap Template.

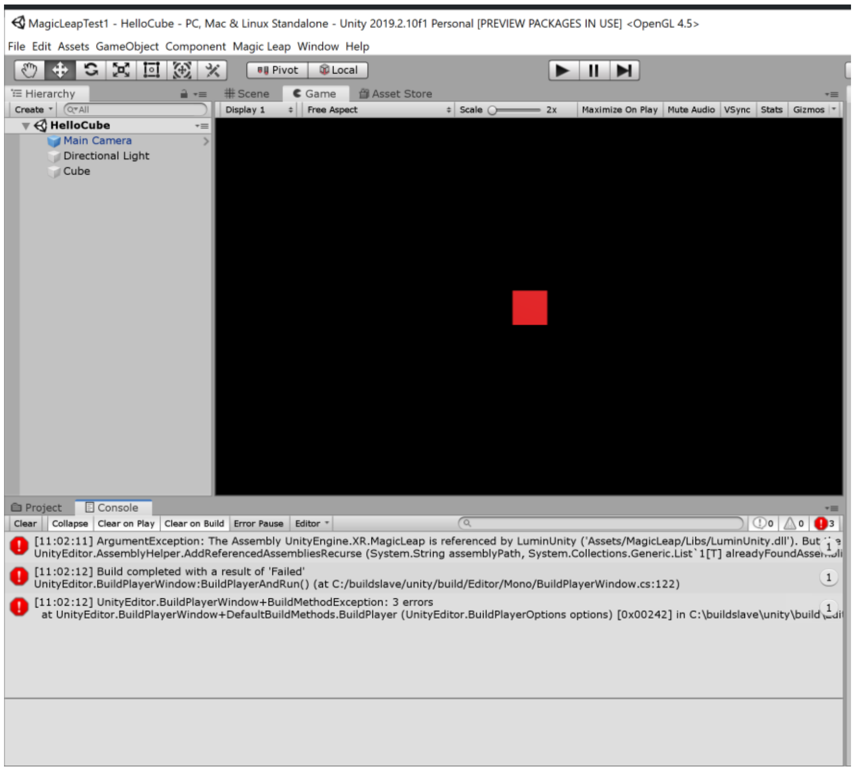
1. Download the Unity magic leap template (https://github.magicleap.com/DevRelSamples/Unity-Project-Template/releases).
2. Extract the zip file in a directory of your choice.
3. Using Unity Hub, download Unity 2019.2.x and make sure Lumin support is checked during installation



☐	tvOS Build Support	326.7 MB	1.4 GB
☐	Linux Build Support	90.6 MB	469.0 MB
☐	Mac Build Support (Mono)	82.7 MB	463.0 MB
☐	Universal Windows Platform Build Support	269.8 MB	2.0 GB
✓	WebGL Build Support	237.0 MB	893.7 MB
✓	Windows Build Support (IL2CPP)	60.3 MB	345.9 MB
☐	Facebook Gameroom Build Support	106.9 MB	344.6 MB
✓	Lumin OS (Magic Leap) Build Support	144.0 MB	806.3 MB
	Documentation		
✓	Documentation	337.1 MB	687.6 MB

Assure that you have an up to date Unity Editor with Lumin OS Support Checked

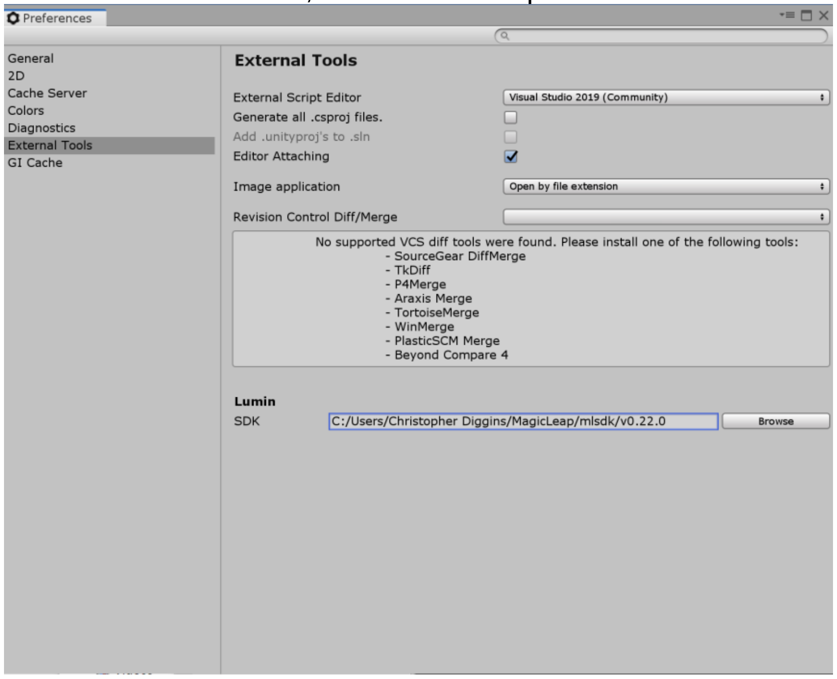
4. `ADD` the project using Unity Hub
5. Open the project using Unity Hub



```
[11:02:11] ArgumentException: The Assembly UnityEngine.XR.MagicLeap is referenced by LuminUnity ('Assets/MagicLeap/Libs/LuminUnity.dll'). But
UnityEditor.AssemblyHelper.AddReferencedAssembliesRecurse (System.String assemblyPath, System.Collections.Generic.List`1[T] alreadyFoundAssem
[11:02:12] Build completed with a result of 'Failed'
UnityEditor.BuildPlayerWindow:BuildPlayerAndRun() (at C:/buildslave/unity/build/Editor/Mono/BuildPlayerWindow.cs:122)
[11:02:12] UnityEditor.BuildPlayerWindow+BuildMethodException: 3 errors
  at UnityEditor.BuildPlayerWindow+DefaultBuildMethods.BuildPlayer (UnityEditor.BuildPlayerOptions options) [0x00242] in C:\buildslave\unity\build
```

THE HELLOCUBE PROJECT WHEN YOU FIRST LOAD IT

6. Under Unity “Edit > Preferences …”, set the MLSDK path



External Tools

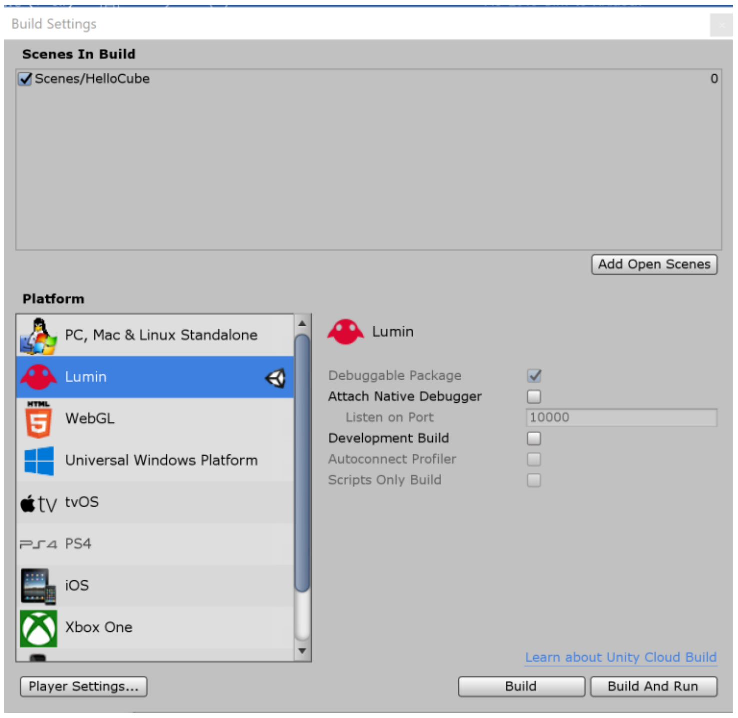
External Script Editor: Visual Studio 2019 (Community)
Generate all .csproj files. ☐
Add .unityproj's to .sln ☐
Editor Attaching ✓
Image application: Open by file extension
Revision Control Diff/Merge

No supported VCS diff tools were found. Please install one of the following tools:
- SourceGear DiffMerge
- TkDiff
- P4Merge
- Araxis Merge
- TortoiseMerge
- WinMerge
- PlasticSCM Merge
- Beyond Compare 4

Lumin
SDK: C:/Users/Christopher Diggins/MagicLeap/mlsdk/v0.22.0

SETTING THE MAGIC LEAP SDK PATH

7. Under `File > Build Settings`, make sure the build target platform is “Lumin”


Build Settings

Scenes In Build

☑ Scenes/HelloCube 0

Add Open Scenes

Platform

PC, Mac & Linux Standalone
Lumin
WebGL
Universal Windows Platform
tvOS
PS4
iOS
Xbox One

Lumin

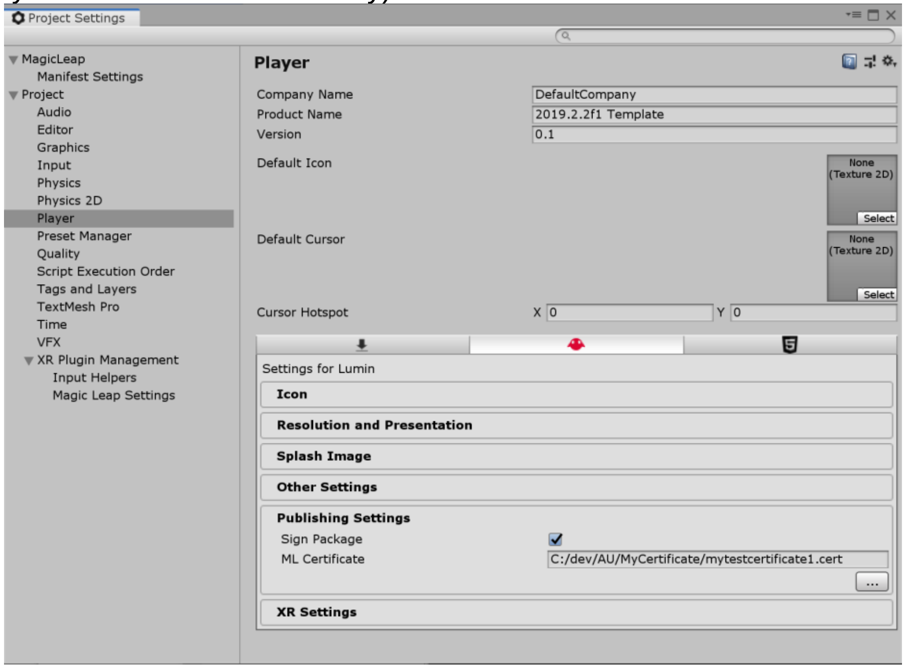
Debuggable Package	✓
Attach Native Debugger	
Listen on Port	10000
Development Build	
Autoconnect Profiler	
Scripts Only Build	

Learn about Unity Cloud Build

Player Settings... Build Build And Run

SETTING THE BUILD TARGET PLATFORM

8. Under project settings > publishing settings, set your certificate path (and make sure the .privkey file is in the same directory)


Project Settings

MagicLeap
  Manifest Settings
Project
  Audio
  Editor
  Graphics
  Input
  Physics
  Physics 2D
  Player
  Preset Manager
  Quality
  Script Execution Order
  Tags and Layers
  TextMesh Pro
  Time
  VFX
  XR Plugin Management
    Input Helpers
    Magic Leap Settings

Player

Company Name	DefaultCompany
Product Name	2019.2.2f1 Template
Version	0.1
Default Icon	None (Texture 2D)
Default Cursor	None (Texture 2D)
Cursor Hotspot	X 0 Y 0

Settings for Lumin

Icon

Resolution and Presentation

Splash Image

Other Settings

Publishing Settings

Sign Package	✓
ML Certificate	C:/dev/AU/MyCertificate/mytestcertificate1.cert

XR Settings

SETTING THE CERTIFICATE PATH

Build Game and Run

9. Make sure USB debugging is enabled between your device and computer (which requires MLDB access) and you're allowing untrusted sources
10. Open the `HelloCube` Scene from `Assets`>`Scenes`>`HelloCube`
11. Build and Run
12. Accept the cert if prompted after build is complete
13. There will be a cube 1 meter in front of where your current headpose session started

Appendices

Scaling up to Larger Projects

Large projects don't scale well when bringing them into or out of 3ds Max. This is because the 3ds Max scene graph was designed to handle the numbers of objects that an artist might create, hundreds or a couple of thousand. However, a medium sized building may have tens or hundreds of thousands of parts.

Some strategies to employ to help work with larger projects:

- Merge by Family Type (or Category) on Import – But you will lose BIM data
- Don't include all categories
- Match the Revit file version to the 3ds Max version
- Import at Medium or Low
- Custom attach

Game Engine Optimization

Game engines have to render frames multiple times a second (e.g. 30, 60 or more frames per second) to maintain a fluid user experience. The main factors that affect game engine render rates are:

1. How much work is done on the CPU per frame to prepare data
2. How long it takes to transfer data from CPU to GPU
3. How much work is done on the GPU to draw the data
4. How often this needs to happen per frame

Draw Calls and Batches

"A draw call is a call to the graphics API to draw objects, while a batch is a group of draw calls to be drawn together. Batching objects to be drawn together, minimizes the state changes needed to draw each object inside the batch. This is turn leads to improved performance by reducing the CPU cost of rendering the objects." - https://support.unity3d.com/hc/en-us/articles/207061413-Why-are-my-batches-draw-calls-so-high-What-does-that-mean-

In short:

- Fewer meshes means fewer draw calls
- Fewer textures means more batches
- Fewer triangles