

MFG319459

Preparing your Inventor and Fusion 360 3D models for use in AR/VR/MR with 3ds Max

Brent Jackson
Polyscopic

Learning Objectives

- How to transcode 3D CAD models into 3D polygonal models
- Learn how to optimize 3D models for use on mobile, VR, or AR
- Learn about texture baking 3D models
- Learn design guidelines for different MR platforms

Description

Viewing your CAD 3D models in augmented reality (AR), virtual reality (VR), and mixed reality (MR) isn't something that can be done with a one-size-fits-all, single-click solution. Tessellating, optimizing, and transcoding must be done to convert your CAD (parametric) 3D model into an AR/VR/MR-ready (polygonal) 3D model. CAD models consist of infinitely accurate curves and multiple materials that must be transformed into triangulated representations of these curves with materials converted to textures and baked onto the models for use on devices such as the Microsoft HoloLens. This class will provide a step-by-step demo showing how to convert your 3D CAD models into AR/VR/MR-ready assets using 3ds Max software. We will also show you how to create your own deployable Mixed Reality application for viewing your 3D models using Unity's real-time rendering engine and MRTK, the Mixed Reality Toolkit. The speaker has saved companies hundreds of thousands of dollars by producing MR design reviews, and will enable you to do the same.

Speaker

After several years working at Bridgestone as a Mechanical Engineer, Autodesk Vault Administrator, and Innovation Specialist, Brent was approached with an opportunity to start his own company, [Polyscopic LLC](#) to consult to Microsoft's Mixed Reality department. He now spends his days working with Microsoft's designers, developers, program managers, and leadership team unblocking enterprise customers so they can succeed with integrating Mixed Reality products into their business workflows. Brent's experience in manufacturing, mechanical engineering, CAD, innovation, design, and Unity development has given him a unique ability to empathize with everyone in the industry 4.0 pipeline from first line workers to final decision makers. He is passionate about the future of work and striving to help make digital transformation easier for everyone.

Why use Autodesk 3ds Max to prepare 3D content for mixed reality applications?

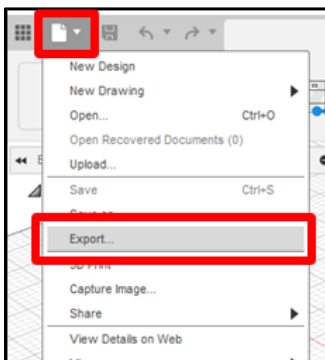
When preparing CAD files for Dynamics 365 mixed reality applications, there are numerous performance and quality roadblocks that prevent a smooth transition from CAD to real-time. 3ds Max is a digital content creation suite that has a unique ability to bridge parametric models and real-time polygon modeling. This tutorial demonstrates how to use the 3ds Max conversion and optimization capabilities to prepare 3D CAD models for use in mixed reality.

How to transcode 3D CAD models into 3D polygonal models

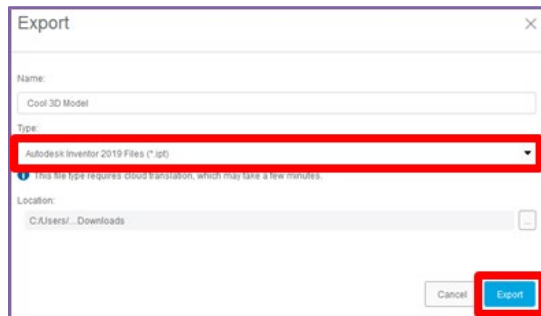
Import a file into 3ds Max

Open a new scene in 3ds Max, and then on the **File** menu, select **Import>Import** to import your 3D model.

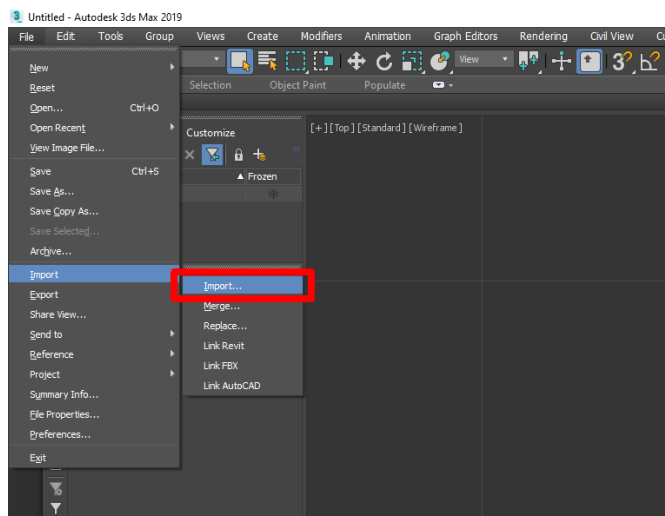
Fusion 360 users: If you are using a Fusion 360 file, first navigate to Fusion 360 and export the file as an Inventor .ipt file by going to File -> Export.



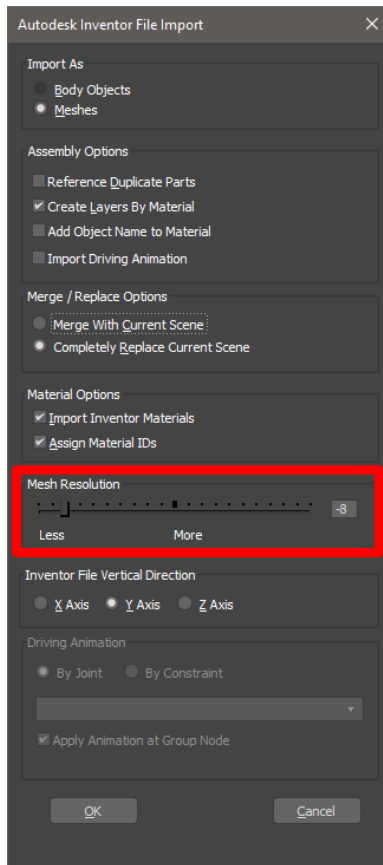
Then selecting "Inventor .ipt" as the file type



Once you have saved your file, go to File -> Import -> Import in 3ds Max.



When you are importing your model, use the following settings. You can change the mesh resolution to be higher if it's a simple model, but in general you can maintain a good visual fidelity and a major reduction in polygons if you use -8.

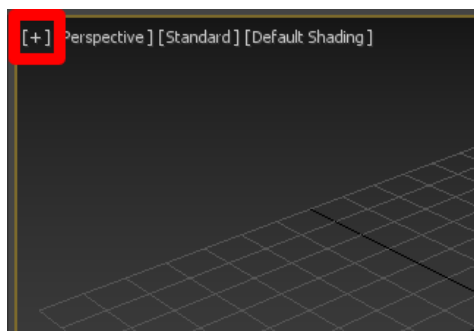


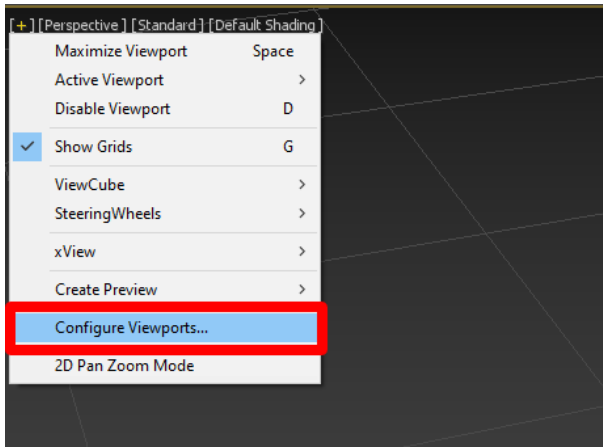
Optimize a 3D model for use on mobile, AR, and VR

If the polygon count is too high ([see Performance targets](#)), the model won't perform well in mixed reality applications. You can optimize the 3D model to reduce the polygon count so the model performs better. To see the polygon count, first set up the viewport to show polygon statistics.

Show polygon statistics

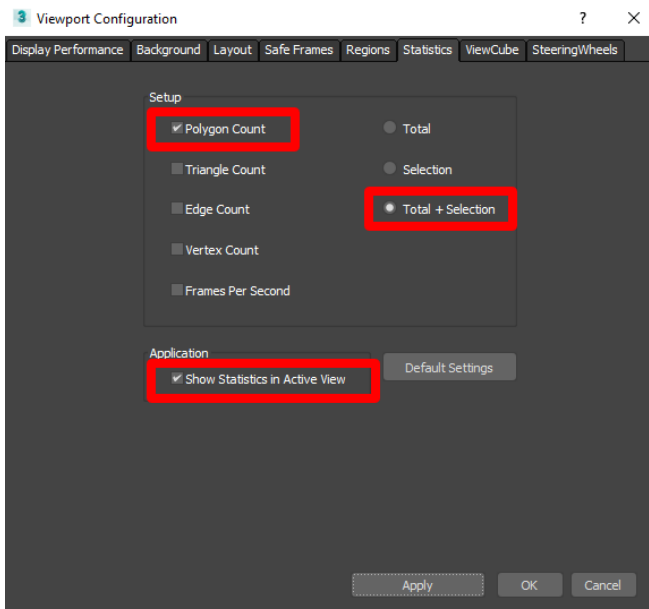
To view the number of polygons in your scene, select **+** in the upper-left corner of any viewport window to open the **Configure Viewports** options.



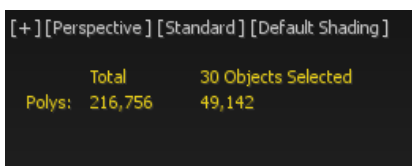


In the **Viewport Configuration** screen, select the **Statistics** tab.

Under **Setup**, select the **Polygon Count** check box, and then select the **Total + Selection** option. Under **Application**, select the **Show Statistics in Active View** check box. When you're done, select **OK**.



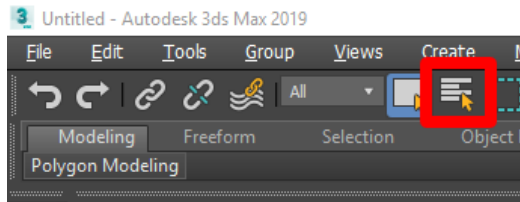
You'll see the total poly count of your model, and the total poly count of any objects that you have selected.



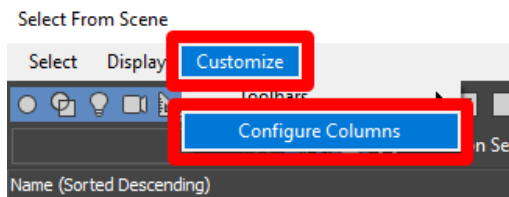
Select high poly objects

The best way to reduce the size of your model while maintaining visual fidelity is to find the objects with the highest poly count, and reduce them the most. Objects such as screws and grills can have thousands of polygons that are rarely seen.

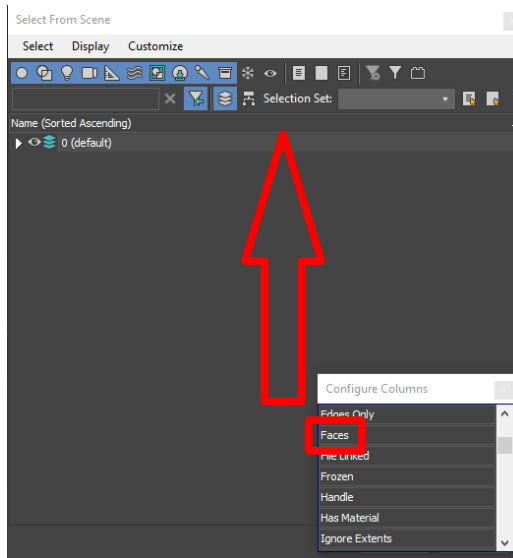
Select the **Name** button to open the **Select from Scene** window.



Select **Customize > Configure Columns**.



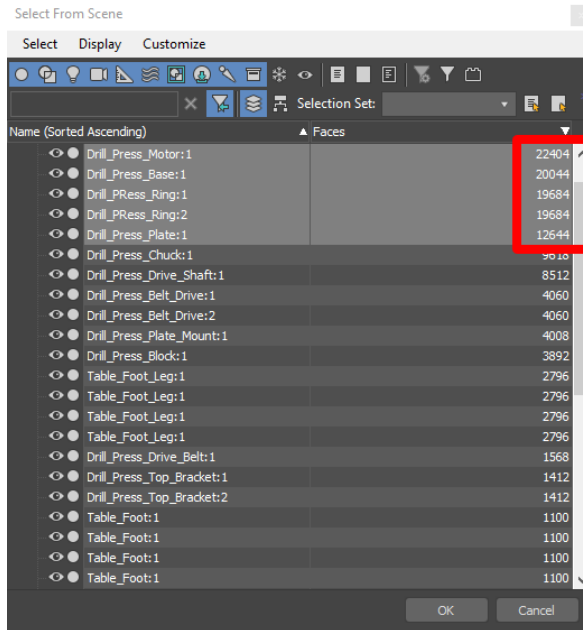
Drag **Faces** next to **Name** to activate the column.



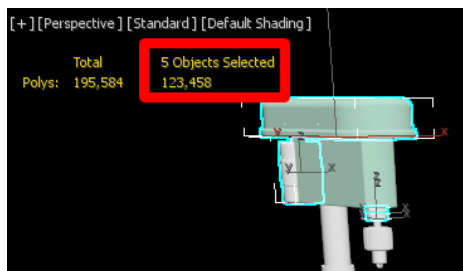
Select the **Faces** tab a few times so that your objects are now sorted from highest face count to lowest.

TIP: You can also search at the top of the **Select from Scene** menu. If your model has lots of fillets, try searching for those. Fillets use lots of polygons and can be reduced without affecting the overall visual fidelity of the model.

Select the objects with the highest face counts, and then select **OK**.

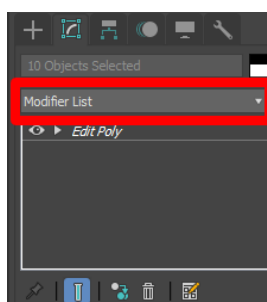


You now have the highest poly objects selected. For the example model, 123K of the 195K polygons exist in 5 objects. The next section in this topic shows how to reduce the poly count of these objects.



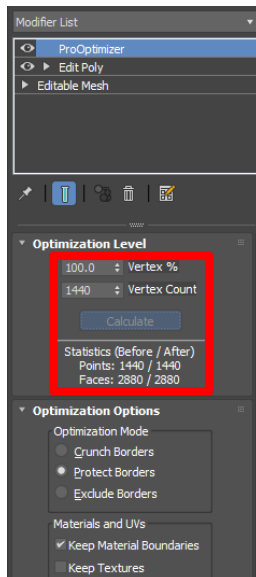
Reduce polygon count

Open the **Modifier List** located on the right side of the viewport.

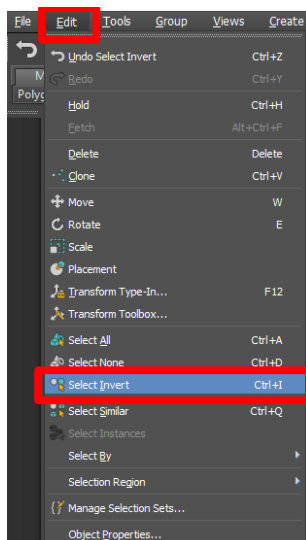


Select **ProOptimizer** from the list.

Select **Calculate** to unlock the **Optimization Level** value adjustments. Try different values for **Vertex %** that range from 10 - 30% until you find the highest level of reduction that still maintains a visual fidelity that meets your standards.



On the **Edit** menu, select **Select Invert** and then add a **ProOptimizer** to the rest of your model. Follow the same steps as you did before, but don't go as low as 10 - 30%. Reduce these other objects until you reach a polygon count that matches the recommended [Performance targets](#) for your specific use case and still provides good visual fidelity.

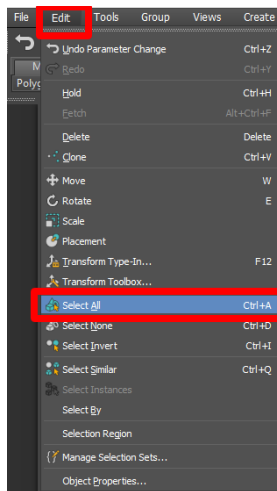


TIP: You can be as granular with what you reduce as you like. If there are specific parts that need higher fidelity, select them and raise the percentage value to meet your needs. Try different techniques until you find one that works best for you.

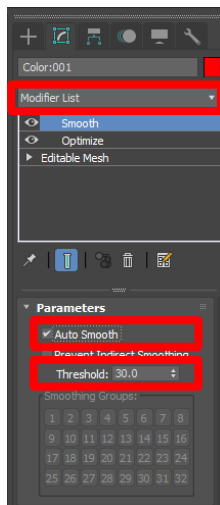
Work with curved surfaces

When curved surfaces are present on 3D models, they might appear faceted. You can soften the appearance of these surfaces by using **Smooth**.

On the **Edit** menu, select **Select All** to select all 3D models in the scene.



In the **Modifier List**, select **Smooth**.



Under **Parameters**, select the **Auto Smooth** check box, and then adjust the **Threshold** value until the faceted surfaces appear smooth. The default threshold is 30.0, which is usually pretty good.

[!NOTE] You can also apply the **Smooth** modifier to individual 3D models if they each require a different threshold.

At this point, your model may be optimized enough for use in mixed reality. If you think it will work fine in this form, you can skip to exporting your model as a GLB file. If the model is still too complex and has lots of materials, go to the next section.

Learn about texture baking 3D models

If there are more than ten materials on the 3D model, combining them into a single material can increase performance. You can do this by baking material colors into a single image map. This is optional, but is a good idea if you find that you experience performance issues when viewing your 3D model. The goal is to have one object with the colors of the original 3D model, and another that represents the combined 3D models to bake to.

Notes:

This process only works if the materials have not yet been converted to **Physical Material**.

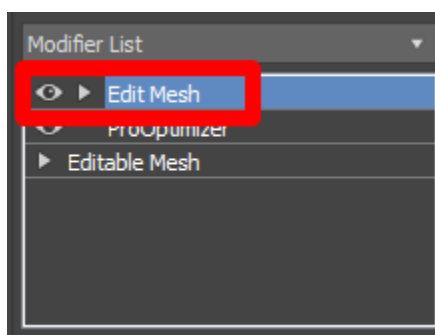
Baking, UVs, and texture maps are complex. The goal of this tutorial is not to make you an expert in texture baking, but to help you get through the process so that you can use your 3D models with Dynamics 365 mixed reality apps. For this reason, this tutorial doesn't go into great detail on texture baking.

Prepare the 3D model

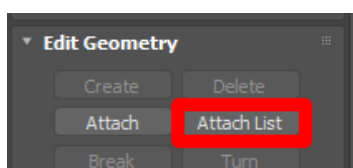
Choose a single object from your model hierarchy, rename it to **Original** and add an **Edit Mesh** modifier to it.

Go to the object selection.

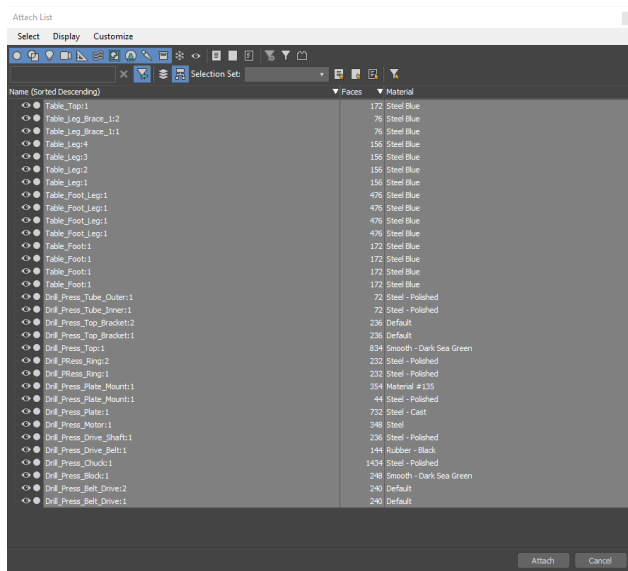
Attach all the 3D models by selecting one object and adding another **Edit Mesh** modifier to it. It doesn't matter which 3D model you select.



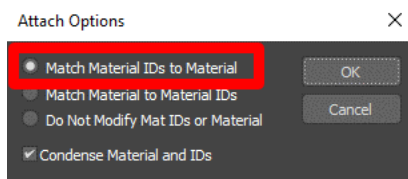
Under the **Edit Geometry** section, select the **Attach List** button. This button shows the available models in the scene that you can combine together. Select all of them and click



Select all of the 3D models in the attach list, and then select **Attach**.



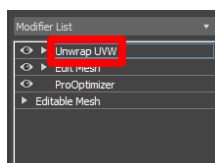
When prompted, select **Match Material IDs to Material**, and then select **OK**.



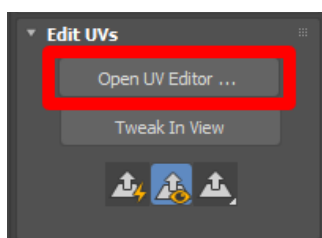
All of the individual meshes are combined into one.

Unwrap UVs

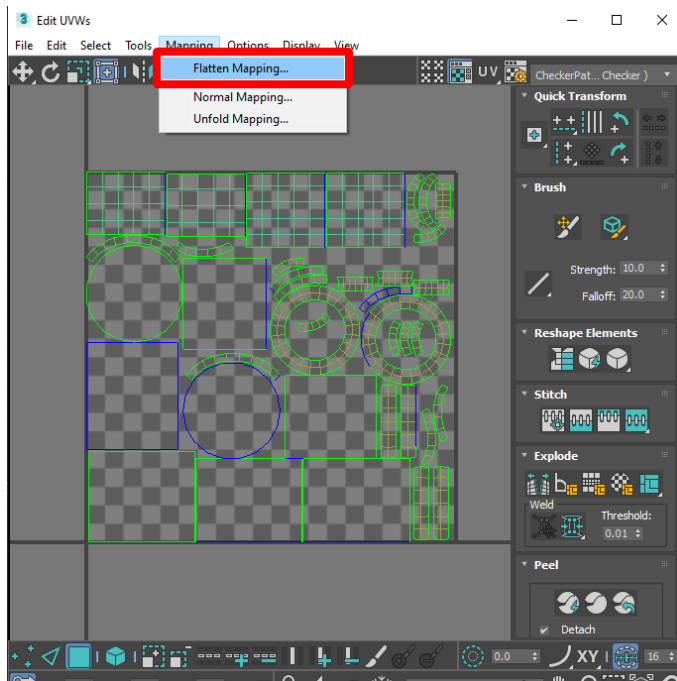
Unwrap the duplicate 3D model by selecting and applying the **Unwrap UVW** modifier from the **Modifier List** for your original mesh. Select **Polygon** in the drop-down to edit the UV faces.



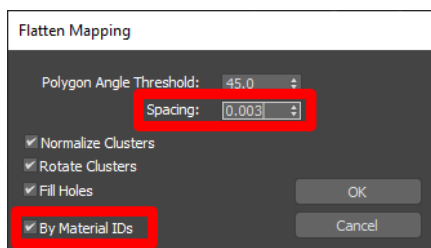
Under **Edit UVs**, select **Open UV Editor**.



In the **Edit UVWs** window, select **Mapping > Flatten Mapping**.



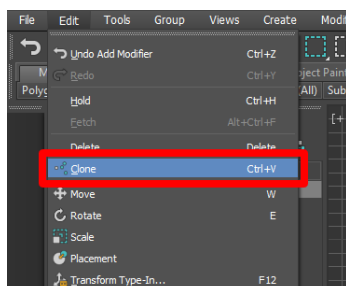
In the option box that appears, give the UVs some padding by setting **Spacing** to **0.003**, select the check box for **By Material IDs**, and then select **OK**.



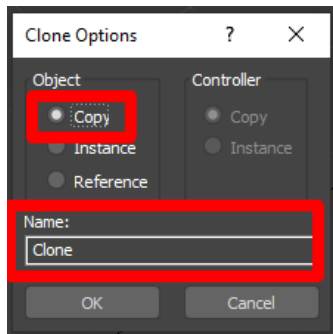
Create a copy of the original mesh

Now that the original mesh has been prepared, you need to create a copy of it to bake the texture to.

To clone the 3D model, select it, and then select **Edit > Clone**.



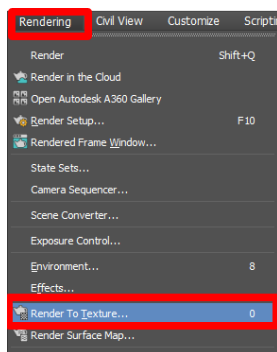
Select **Copy**, and then rename the object so that you know it's the cloned object.



Bake texture

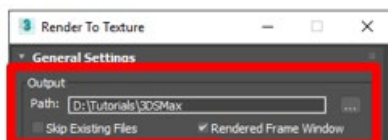
Select the cloned mesh (the one you want to bake the texture onto).

On the **Rendering** menu, select **Render To Texture**.

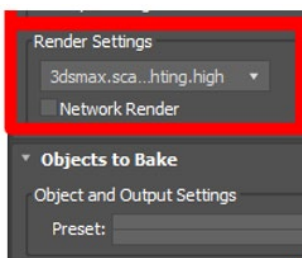


Set up the **Render to Texture** menu in the following ways:

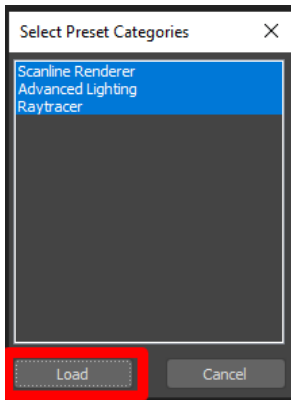
- Under **Output**, set **Path** to the location where the map will bake. You can leave the default setting if you don't have a specific destination in mind.



- Change the **Render Settings** to **3dsmax.scanline.no.advanced.lighting.high**.

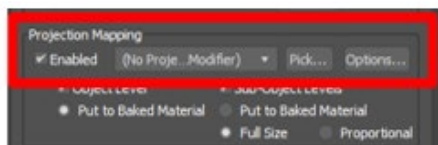


c. When you're prompted to **Select Preset Categories**, leave the entries all highlighted, and then select **Load**.

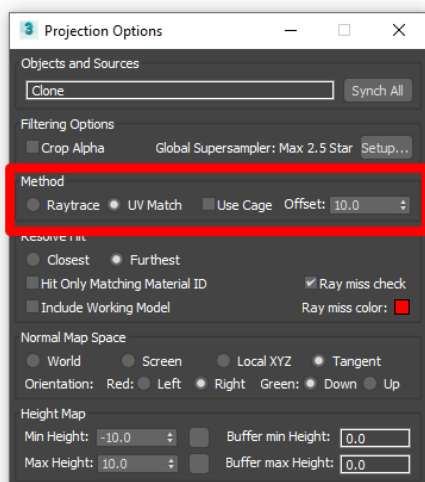


d. Under **Projection Mapping**, do the following:

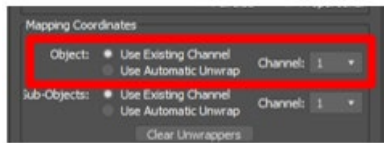
i. Select the **Enabled** check box, select **Pick**, select the original source 3D models you want to bake the color from, and then select **Add**.



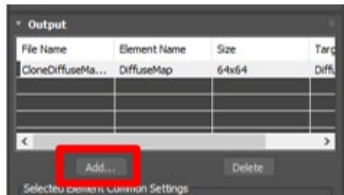
ii. Select the **Options** button next to the **Pick** button, and then in the **Method** section, select the **UV Match** option, clear the **Use Cage** check box, and close the window.



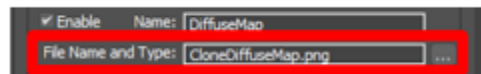
d. Under **Mapping Coordinates**, select the **Use Existing Channel** option, and then set the channel to **1**.



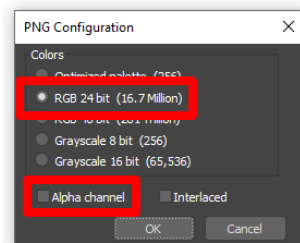
e. Under **Output**, select **Add**, select **DiffuseMap**, and then select **Add Elements**.



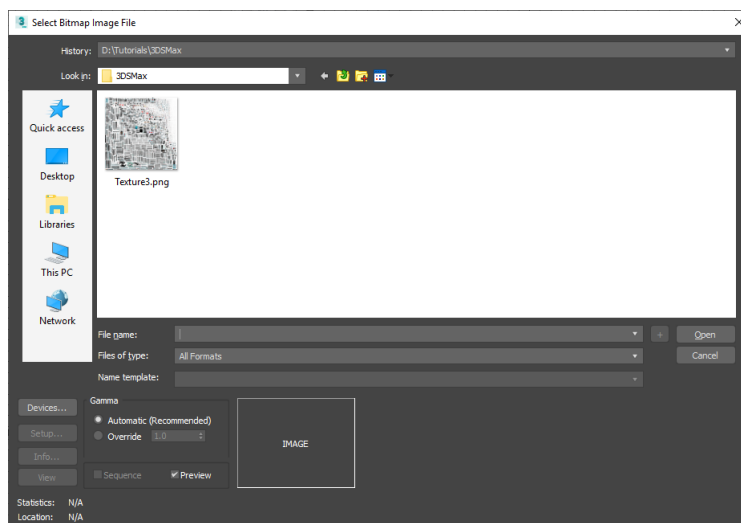
f. Select the three dots next to **File Name and Type**, and then select **.png**.



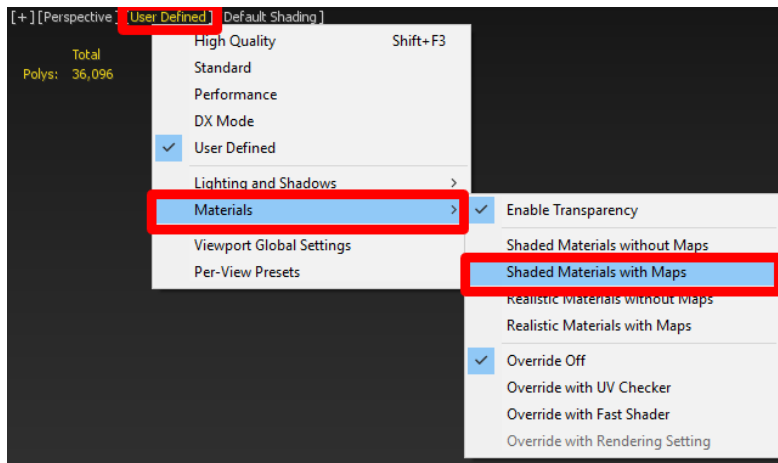
g. In the pop-up menu, select the **RGB 24 bit** option, clear the **Alpha channel** check box, and then select **OK**.



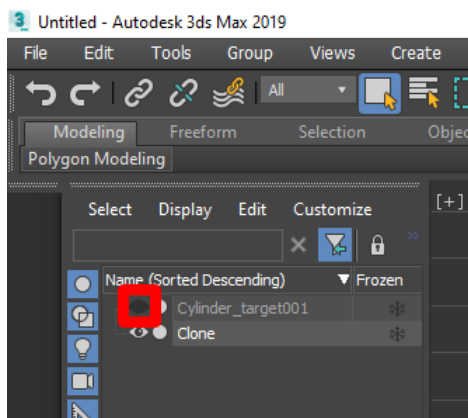
File, select the texture created earlier.



To view the new texture on the model, in the viewport, go to: **User Defined > Materials > Shaded Materials with Maps**.



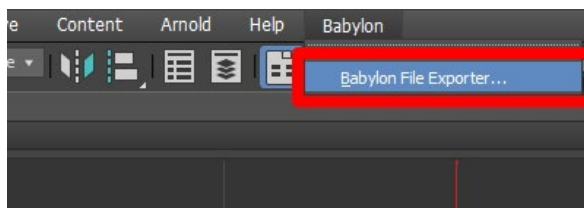
Hide the original 3D model so you can see the optimized 3D model with its texture.



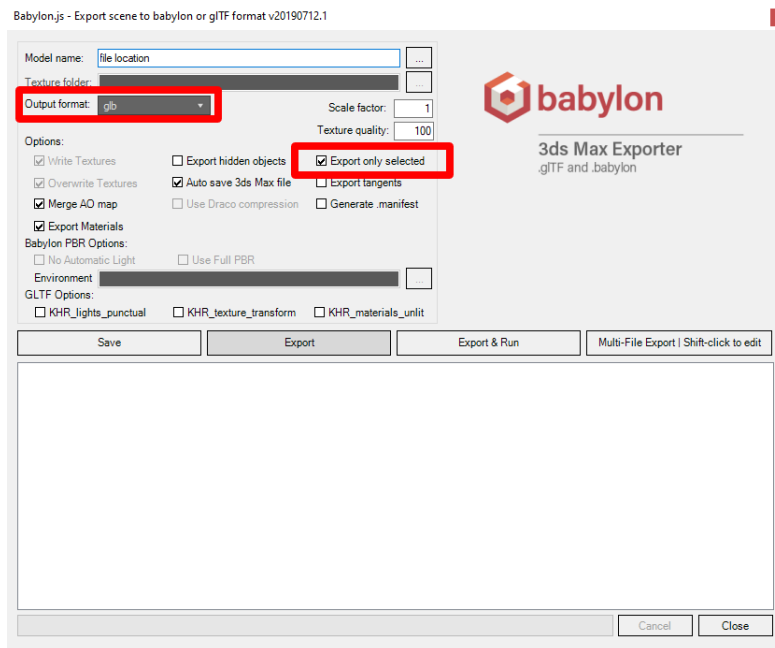
Export the 3D model

Select the cloned model.

On the **Babylon** menu, select [Babylon File Exporter](#).



Make sure **glb** is selected under **Output format**, and then select the **Export only selected** check box so all necessary or selected 3D models are exported.



Select **Export**.

Congratulations! You have now modified your CAD 3D model into a 3D model that is ready for use in Mixed Reality applications.







Learn design guidelines for different MR platforms

In the upcoming section you will learn design guidelines for different MR platforms, and how to deploy this to your own Mixed Reality application using Unity and the mixed reality tool kit.

Performance targets

The 3D models you have produced can be used on mixed reality headsets, immersive headsets, and mobile devices. The goal when optimizing these models is to provide the highest possible visual fidelity without adversely affecting performance.

The following table lists some general conservative targets to aim for when preparing 3D models for a range of hardware. When in doubt, target the midrange profile for a balance of fidelity and performance.

	Low-scene complexity 	Medium-scene complexity 	High-scene complexity 
 Mixed Reality	Objects: 1-3 per scene Triangles: <100,000 Materials: 1-2 per object	Objects: 4-10 per scene Triangles: <30,000 Materials: 1-2 per object	Objects: 10+ per scene Triangles: <10,000 Materials: 1-2 per object
 Immersive headsets	Objects: 1-3 per scene Triangles: <15,000,000 Materials: 1-2 per object	Objects: 4-10 per scene Triangles: <500,000 Materials: 1-2 per object	Objects: 10+ per scene Triangles: <150,000 Materials: 1-2 per object
 Mobile	Objects: 1-3 per scene Triangles: <500,000	N/a	N/a

For more information on best practices, you can visit the documentation that I recently published for Microsoft at

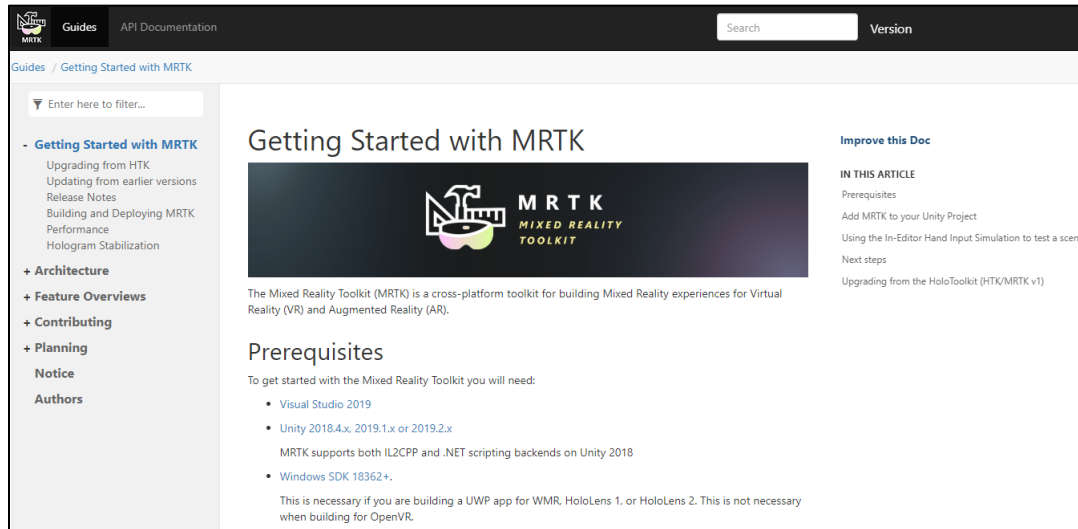
Creating a 3D model viewer for HoloLens with Unity and MRTK:

Now that we have our 3D model, it's time to check it out on a HoloLens!

Setting up your developer environment

To do this, we need to make sure our dev environment is set up correctly. Navigate to the Mixed Reality Toolkit's getting started page to find directions on how to correctly set up Unity, MRTK, and Visual Studio.

<https://microsoft.github.io/MixedRealityToolkit-Unity/Documentation/GettingStartedWithTheMRTK.html>

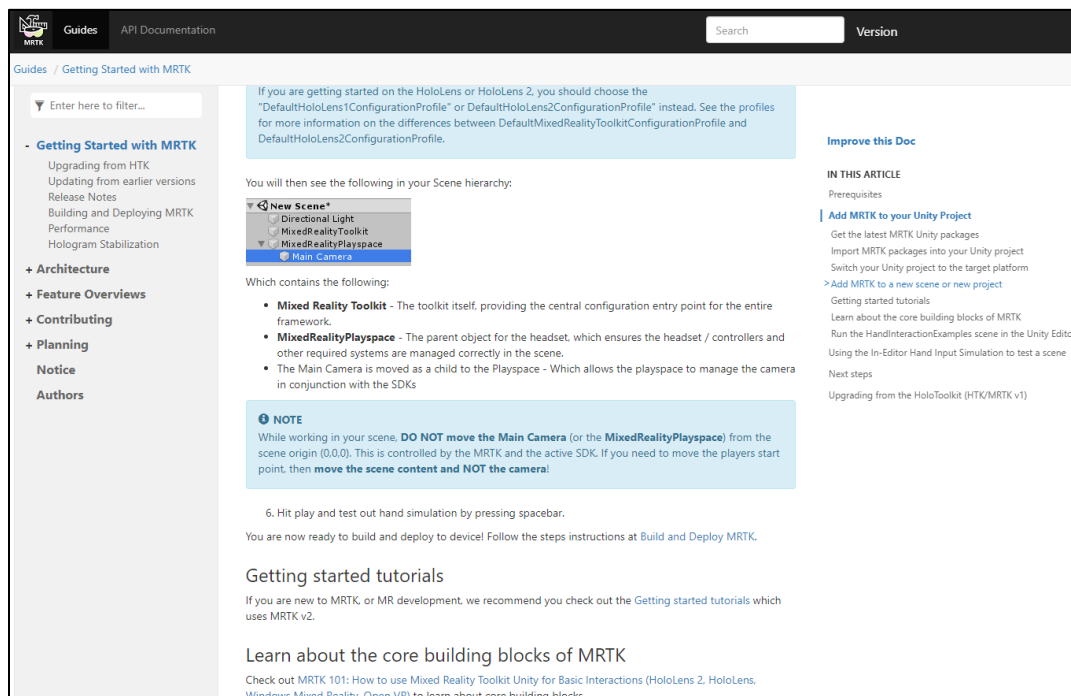


The screenshot shows the 'Getting Started with MRTK' page. The left sidebar contains a navigation menu with sections: Getting Started with MRTK (including Upgrading from HTK, Updating from earlier versions, Release Notes, Building and Deploying MRTK, Performance, and Hologram Stabilization), Architecture, Feature Overviews, Contributing, Planning, Notice, and Authors. The main content area has a header 'Getting Started with MRTK' with a logo. Below it, a paragraph states: 'The Mixed Reality Toolkit (MRTK) is a cross-platform toolkit for building Mixed Reality experiences for Virtual Reality (VR) and Augmented Reality (AR)'. A 'Prerequisites' section follows, listing: Visual Studio 2019, Unity 2018.4.x, 2019.1.x or 2019.2.x, and Windows SDK 10.0.17134.0. It also notes that MRTK supports both IL2CPP and .NET scripting backends on Unity 2018 and that a UWP app is necessary for WMR, HoloLens 1, or HoloLens 2.

Configure MRTK for HoloLens 1

Now that our developer environment is set up correctly, let's configure MRTK for use on HoloLens

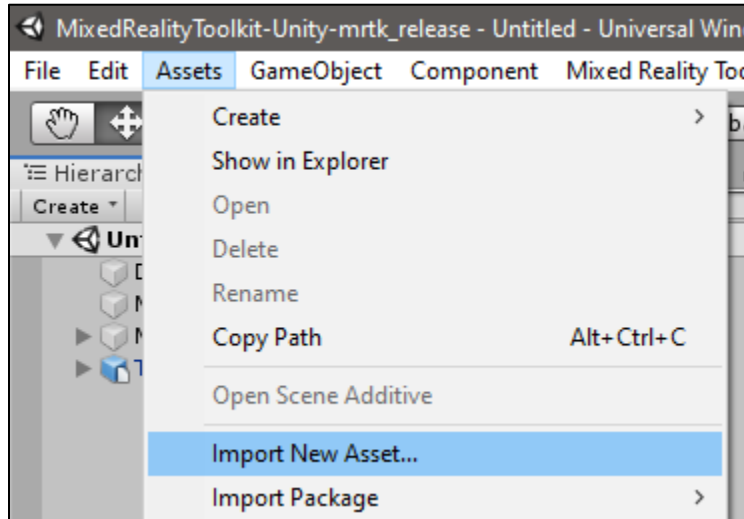
Luckily this information is also available on the same web page. Scroll down and follow the instructions to set up a basic scene. You can return here when you get to this point in the tutorial:



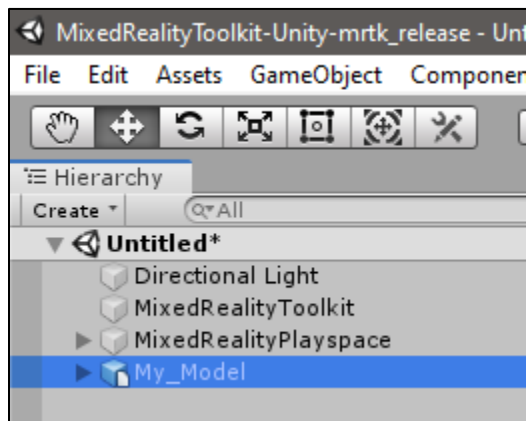
This screenshot shows the 'Getting started tutorials' section of the MRTK documentation. It includes a 'New Scene' screenshot showing a hierarchy with 'Directional Light', 'MixedRealityToolkit', 'MixedRealityPlayspace', and 'Main Camera'. A 'NOTE' box states: 'While working in your scene, DO NOT move the Main Camera (or the MixedRealityPlayspace) from the scene origin (0,0,0). This is controlled by the MRTK and the active SDK. If you need to move the players start point, then move the scene content and NOT the camera!'. The text continues: '6. Hit play and test out hand simulation by pressing spacebar. You are now ready to build and deploy to device! Follow the steps instructions at Build and Deploy MRTK.' It then links to 'Getting started tutorials' and 'Learn about the core building blocks of MRTK'.

Load your model into the scene:

Import your 3D model into the scene by navigating to **Assets -> Import New Asset**.



Next you will drag your 3D model into the scene Hierarchy pane on the left of the Unity Editor Window



Once your model is in the scene you will want to offset it from the camera, otherwise when you start the application, you will be standing inside of it!

To do this, select your model in the Hierarchy window (it will be highlighted blue as shown above) and in the Inspector pane on the right side of the window change the "Position" "Z" value to 2. This will move your model 2 meters in front of you. You don't have to use 2 meters explicitly, but this is a good starting point. If your model is much bigger, you may want to start it further away. A good rule of thumb is to divide length or width of your model (whichever is facing you when you want it to start) by 2 and then add 2 to that number. This will ensure that the edge of your model starts 2 meters away.

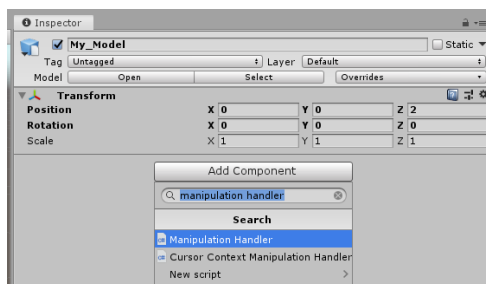
Make your 3D model interactable

To make your 3D model interactable, you will need to apply these two scripts to it:

- ManipulationHandler.cs
- BoxCollider.cs

This will allow you to grab your object and place it wherever you like.

To do this, simply click “Add Component” in the inspector, type “Manipulation Handler” and click on the Manipulation Handler script to add it to your object. As shown below. Do the same for to add the Box Collider script.




If you would like to add more functionality to your 3D model and scene, I suggest following “the MRTK 101: How to use Mixed Reality Toolkit Unity for Basic Interactions (HoloLens 2, HoloLens, Windows Mixed Reality, Open VR)” tutorial located here:

<https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-101>

That’s it! You can test your scene by clicking the play button at the top of the editor.



This will launch a HoloLens simulator that will allow you to confirm that everything is set up correctly. Once in the simulator launches, click in the game window with your mouse and then hold down shift and Left mouse button to grab the object. Move your mouse around to confirm that the model will move.

Once you have confirmed that your model is manipulatable, you can exit the play menu by clicking the play button again which is now blue  and move to the next section that will cover deploying your application to the HoloLens.

If you would like more information on how to use the test editor, check out this documentation on the MRTK website.

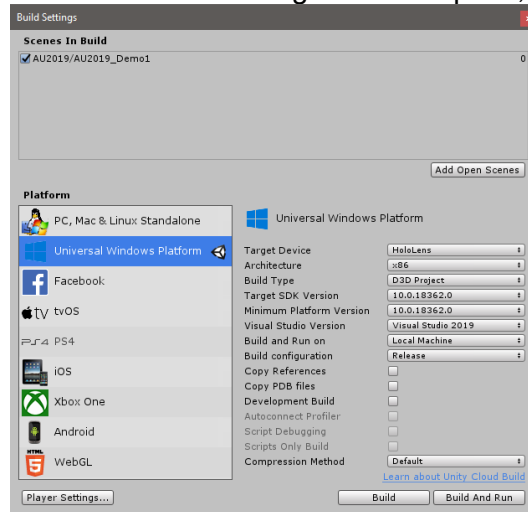
<https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-101#how-to-simulate-input-interactions-in-unityeditor>

Deploying your Mixed Reality Application

Now that are Scene is ready for deployment. Let’s get it on our HoloLens!

Confirm your build settings

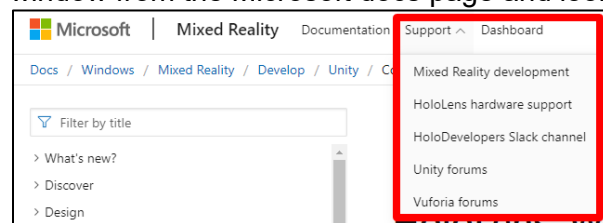
Open the “Build Settings” window by navigating to **File -> Build Settings**. When the Build Settings window opens, confirm that it looks like this:



If you do not have your scene in the build, make sure that you have saved it, and then click the “Add Open Scenes” button to add it to your build. The Mixed Reality Toolkit should take care of setting everything else up for you.

Now Click “Build”. A file browser will open where you should create a folder named “App” and then click “Select Folder”.

If there are no errors, then your build should complete within a few minutes, and the “App” Folder will open. If you receive build errors, then you can navigate to the support window from the Microsoft docs page and look for help here.

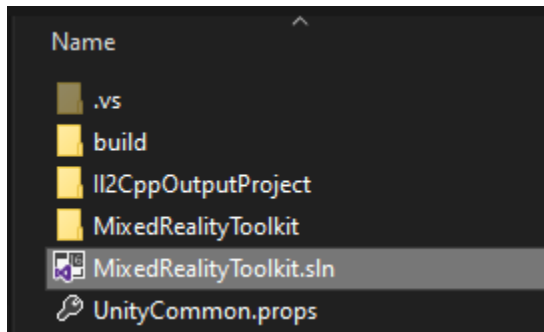


If you are still unable to solve your problems after researching here, then you can reach out to us at [Polyscopic](#) for professional assistance.

Deploy your application with Visual Studio

Now that your application has been built, let’s open it in Visual Studio and deploy it to our HoloLens.

Double click on the MixedRealityToolkit.sln Visual Studio Project to open your build in Visual Studio.



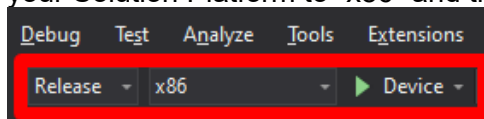
Connect your HoloLens to your PC

Plug your HoloLens into your PC with a USB cable and power it on. If you are using an enterprise edition, you will need to log in to the HoloLens.

If this is the first time you have built something on your HoloLens, go to System, Developer, put the device in developer mode. You will probably need to reset the device, then go back to this menu and select “Pair device.” A number will appear, when you run select “Deploy Solution” in Visual Studio, you will be prompted for a pairing request number, enter the number displayed inside the HoloLens. You will only have to do this pairing step one time, afterwards, any additional deployments will happen automatically.

Deploy your project to HoloLens

At the top of the Visual Studio window, set your Solutions Configuration to “Release”, your Solution Platform to “x86” and the destination to “Device” as shown below.

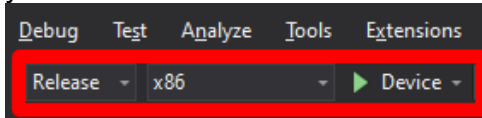


Now go to the “Build” menu and select “Deploy Solution”. This will deploy your application to your HoloLens. Once it has finished deploying, put on your HoloLens, look for the application called “Mixed Reality Toolkit” and launch it. You should now see your 3D model in front of you. Select it and move it around.

Congratulations! You have just developed a HoloLens application to visualize your optimized 3D Cad models at full scale! Now give yourself a pat on the back and go show it off to your coworkers! You are now a mixed reality developer!

Deploy your project to HoloLens

At the top of the Visual Studio window, set your Solutions Configuration to “Release”, your Solution Platform to “x86” and the destination to “Device” as shown below.



Now go to the “Build” menu and select “Deploy Solution”. This will deploy your application to your HoloLens. Once it has finished deploying, put on your HoloLens, look for the application called “Mixed Reality Toolkit” and launch it. You should now see your 3D model in front of you. Select it and move it around.

Congratulations! You have just developed a HoloLens application to visualize your optimized 3D Cad models at full scale! Now give yourself a pat on the back and go show it off to your coworkers! You are now a mixed reality developer!

Connect with us

Want to take your projects further? Connect with us at Polyscopic and we can help.



www.polyscopic.co

Brent@polyscopic.co