



Importing assets into Unity is easy, but getting assets out of Unity can be tricky. The goal of **Export2Maya** is to make exporting assets out of Unity easier, while retaining as much data from Unity as possible. **Export2Maya** will write directly to the **Maya Ascii** file format, generating Maya scene files for you.

What's New – Export2Maya 2.2.0

Export2Maya Completely Re-Written from the ground up

- Much more organized than before

- Overall speed increases 2x – 5x faster

Standard Mesh Export

New Mesh Edge Generation - 3x faster!

- Using StringBuilder to build strings of edges

- Using List.BinarySearch for speed improvements

Skinned Mesh Export

- Same features as Standard Mesh Export

Blendshapes

- Single Frame Blendshape Support

- Multi-Frame Blendshape Support

Material Export

- Added support for the following standard material properties:

 - _MainTex

 - _SpecGlossMap

 - _BumpMap

 - _EmissionMap

Lights Export

- Directional Light Export

- Point Light Export

- Spot Light Export

- Area Light Export

Terrain Export

- Standard Terrain Mesh Export

- Standard Terrain UV Export

- Standard Terrain Secondary UV Export

- Standard Terrain Textures Export

- Standard Terrain Splatmaps Export

Camera Export

- This feature is currently in beta, as I feel it still isn't 100% there yet

Current Limitations

- **Texture Export** - This version of the texture export only reads the following standard shader properties (*Next version will have a better system to specify which properties of your custom Unity shaders map to which properties of your Maya shaders*):
 - _MainTex.
 - _SpecGlossMap
 - _BumpMap
 - _EmissionMap
- **Bind Poses** – Bind Pose export is not currently possible due to the fact that Unity doesn't store all relevant information needed to recreate the bind pose accurately. This is a Unity limitation, and hopefully the Unity Developers can fix it soon.
- **Terrains** – Detail meshes and grass export is not currently supported. This feature is planned for next release.

Installation - Unity Package

- 1) From inside the Unity editor, select **Assets > Import Package > Custom Package**.
- 2) Browse to the **Export2Maya.unitypackage** and import it. A new folder will be created in your Project called **Export2Maya**.
- 3) A new menu item will now be available from the main Unity menu bar (**Window > Export2Maya**).

Installation - Maya MEL Script

- 1) Once the **Export2Maya** unity package has been imported into your Unity project, you will find an **Export2Maya.mel** script under the **Export2Maya** folder.
- 2) Browse to your Unity Project and copy the **Export2Maya.mel** script into your Scripts folder in your Maya Documents location:

Windows XP

`%Documents and Settings%\<username>\My Documents\maya\<maya version>\scripts`

Windows Vista and Windows 7

`%Users%\<username>\Documents\maya\<maya version>\scripts`

Mac OS X

`~/<username>/Library/Preferences/Autodesk/maya/<maya version>/scripts`

Example: `C:\Users\user\Documents\maya\2014-x64\scripts`

- 3) To Create the **Export2Maya Menu**, type this into either the Command Line at the bottom of the Maya UI or into the Script Editor:
`Export2Maya () ;`
- 4) Press enter. You should now have an **Export2Maya** menu option in the top menu bar of the Maya UI.
- 5) If you would like to make the **Export2Maya** menu automatically appear when you start Maya, you will have to modify your **userSetup.mel** file. If you have not yet created a **userSetup.mel** file, then you will need to make one. Browse to your Scripts folder in your Maya Documents location again.
 - 5a) If there is a **userSetup.mel** file already, open it and add `Export2Maya () ;` at the very end and save it.
 - 5b) If there is no **userSetup.mel** file, create a new text file and name it to **userSetup.mel**. Open it, and write `Export2Maya () ;` and save it.

Note - Capitalization and spelling are important! Make sure `Export2Maya () ;` and `userSetup.mel` are spelled correctly!

How to Use - Unity Editor

- 1) Select GameObjects either in the Hierarchy view, or the Scene view.
- 2) Open the **Export2Maya** exporter window from the Unity menu bar (**Window > Export2Maya**).
- 3) Select which version of Maya you would like it to write to (2016, 2015, 2014, 2013, etc).
- 4) When ready, press the **Export Selection** button. Choose a name for the Maya file and where you want the file to be saved.
- 5) Click **Save** and **Export2Maya** will begin processing your GameObjects and save out your Maya file.

Note - Depending on scene / mesh complexity, the export process may take a while to complete.

Usage Info

Keep in mind that the data **Export2Maya** exports is a straight port of how the data is stored inside Unity. Because of this, do not expect the mesh to come back into Maya the way it was originally exported from Maya. The way Unity stores mesh data is different than the way Maya stores it. Unity is a game engine, and as such it converts your mesh data and stores it in a way that is fast to process and is fit for real time use.

A few things to know while working with Maya scenes exported from Unity:

- **Name Clashing** - In Unity, it is perfectly acceptable to have multiple objects named the same, but Maya does not support this. The scene will error out on load if there are multiple objects with the same name.

In order to fix this, **Export2Maya** implements an object renaming mechanism to check for name clashes. If you have multiple objects with the same name, **Export2Maya** will append a number to the end of the object name as it writes it to the Maya scene file in order to make it unique. Of course, if all your objects are uniquely named to begin with then you won't have to worry about anything.

Also, Maya is very strict with what characters are allowed in object names. Maya only allows underscores, letters and numbers. If you have any spaces in your object names or other special characters, they will be converted to underscores to fit correctly into Maya naming conventions.

So to sum it up - Your objects might be named slightly different in Maya than they are in Unity. If your Unity objects are already uniquely named, and do not contain any special characters in the name, then your Maya object names will be identical to what is in Unity.

- **Normals** - In order to preserve the normal information as it is inside Unity, the normals are written directly into the Maya scene. Because of this, Maya will **automatically lock the normals**. Keep this in mind if you try to soften or harden edges after loading the scene file. You will have to **unlock the normals** before you can edit them. To unlock the normals, inside Maya select your object and choose:

Polygons > Normals > Unlock Normals

- **Split Meshes** - Unlike Maya, in Unity you cannot have a merged vertex if it is the boundary of a UV edge. Because of this the mesh triangles in Maya may not be merged together, creating split seams in your mesh. If you want to merge your mesh seams, select the vertices that need merging and select:

Polygons > Edit Mesh > Merge □

Under the **Merge Options**, set the Threshold to a low value, such as **0.001** then click **Merge**.

Lightmap Workflow

So what exactly does this **Export2Maya MEL Script** do? The best way to explain it would be to go over the lightmap baking process, and then explain how the script simplifies this process.

The way Unity handles lightmaps is:

- Each object to be lightmapped has a secondary UV set with non-overlapping UVs.
- Each object's secondary UV's are within the 0 to 1 UV space range.
- Instead of lightmapping each object separately (which would result in hundreds of lightmaps) it packs multiple object's uvs together and scales the secondary UV's down to fit into the same 0 to 1 UV space.
- The position, offset, and index values are recorded for each object, so each one knows what part of which lightmap texture to sample to get the lightmap data.
- Once all the UV space has been filled for a given lightmap texture, a new lightmap texture (and lightmap index) is created so more objects can be lightmapped at the resolution specified.
- A scene could have 1 or more lightmap textures. **Export2Maya** will write out the secondary UV set for objects with the correct tiling and offset already in place. However, depending on how the user would like to use the objects inside Maya, all the objects are kept as separate, individual meshes. It is much easier to combine meshes if need be than to extract them and maintain the correct transform values.
-

All that is needed to bake lightmaps inside Maya is, you have to know which objects belong to which lightmap texture. If you know this and select all the correct objects, you could combine them into a single object and bake a lightmap for them.

The problem is, if a scene has hundreds, or thousands of objects, how do you know which objects belong to which lightmap?

Luckily, **Export2Maya** will do the dirty work for you. When you export the scene from Unity, it will find which lightmap index the object is using, and based on that index, it will create a display layer in Maya which has the name of the lightmap the object is indexing so you can easily select the objects that belong to that lightmap.

The only drawback is when objects are grouped under each other.

Lets say for example we have a **sphere** mesh and a **cube** mesh. The cube is parented under the sphere. The **sphere** (parent) uses **lightmap index 0** and the **cube** (child) uses **lightmap index 1**. If you were to select the objects that were associated with **Lightmap_Layer_0** (the sphere), both objects would be selected. Since Maya is hierarchy based, selecting the parent object selects the children objects as well. If you selected both and combined them, the wrong data would be baked to **lightmap 0** since the wrong objects are included in the bake.

So if you want to bake lightmaps in Maya, you have to un-parent all the objects you want to bake, so they are no longer parented under any objects and they don't have objects parented under them.

You could go through and do this by hand, then right click on the display layer and select the objects that belong to that layer, but this is very tedious.

The accompanying **Export2Maya.mel** script will look at your scene, it will find all the display layers and figure out which objects are assigned with each layer. It will then un-parent all the objects and combine them based on which layer they belong to. This will leave you with a scene with a few meshes that are easily selectable and easy to bake, and also avoids any conflicts with accidentally selecting objects with the wrong lightmap index.

Lightmap Tutorial

So you have exported your Unity Scene file to Maya and would like to setup lightmaps for it. This quick tutorial will give you the steps needed to get your scene ready for lightmap baking, and how to apply those lightmaps inside Unity. This tutorial will be aimed at rendering lightmaps in **Vray**, but the same techniques can be applied to **Mental Ray**. This tutorial will be using the **Angry Bots** scene that comes with Unity.

1. **Export your Scene to Maya** - Open up the **Angry Bots** example scene in Unity, select all the objects and use **Export2Maya** to write out a Maya scene file of the entire scene.
2. **Open the exported Angry Bots scene in Maya** - Notice that there are display layers called **Lightmap_Layer_0**, **Lightmap_Layer_1**, and **Lightmap_Layer_2**. This means in the **Angry Bots** scene, it found 3 lightmap textures that the exported objects are referencing.
3. **Configure Vray for Lightmap Baking** - Make sure **Vray** is selected as your desired renderer, and open the **Render Settings** window.
 - 3a. **Under the Vray Common tab** - Set the Image Format to EXR. (not multichannel, the standard EXR). The default values are fine. **Vray** will default to 16 bit (half float) which works fine in Unity, and is smaller than full 32 bit EXR's. If you run into any issues, set the bit depth to 32 bit.
 - 3b. **Under the Render Elements tab** - Find the Raw Total Light render element. Select it and click **Add** to add this render pass to your render. This is going to be your lightmap pass.
 - 3c. **Run the Export2Maya > Setup Lightmap Baking command** - After a minute or so you will notice 3 new objects have been created, **Lightmap_Layer_0_mesh**, **Lightmap_Layer_1_mesh**, and **Lightmap_Layer_2_mesh**. These are the combined meshes of each of the objects associated with the lightmap display layers.
 - 3d. **Create a Vray Bake Set** - Select the newly combined meshes, and under the Rendering menu, choose: **Lighting / Shading > Assign Single Vray Bake Options to Selection**. A new **Vray Bake Options** node will be created.
 - 3e. **Set Bake Options** - Select the **vrayBakeOptions** node that was just created and open the **Attribute Editor**. Set the options as follows:
 - UV set** - Here you will need to input "lightmap" instead of "map1". This will tell **Vray** to bake to the lightmap uvs instead of the default uv set.
 - Output Texture Path** - This will be a temporary path where the baked textures will go.
 - Baked Image Width** - This is the size of the baked texture. You will have to find the lightmap textures already in your Unity scene and see what resolution the lightmaps are baked at. In the **Angry Bots** demo, they are set to **1024x1024**, so set **Baked Image Width** to **1024**.
4. Scene Setup Complete!

Congratulations! Your scene is now setup and ready to bake lightmaps. Granted, if we were to bake them at this point they would be completely black because we haven't added any lighting yet. At this point, you would add lights and light the scene however you wanted the final lighting to appear in game. You might want to turn on GI depending on how you want the scene to look as well.

Lightmap Bake

When you are ready to bake your lightmaps, do the following:

1. Select the **Lightmap_Layer_0_mesh**, **Lightmap_Layer_1_mesh**, and **Lightmap_Layer_2_mesh** objects.
2. Open the **Vray Bake Options** window:
Lighting / Shading > Bake with Vray... □
3. Turn **Skip Objects with no VrayBakeOptions Assigned** to **ON**, and **Assign Baked Textures to** **OFF**.
4. Now you are ready to bake the lightmaps. Choose either **Bake and Close** or **Bake**. **Bake and Close** will close the bake window once the bake has completed, and **Bake** will keep the window open.
5. Once the bake has completed, you will notice that there are 2 images for every baked object, one with the object name and one with **rawTotalLight** in the name. The first image of each object is the baked texture. Since we don't need the baked textures and only the lightmaps, you can delete all the images that do not have **rawTotalLight** in the name. You should be left with the following images, or something similar:
 - baked-Lightmap_Layer_0_meshShape.rawTotalLight.exr
 - baked-Lightmap_Layer_1_meshShape.rawTotalLight.exr
 - baked-Lightmap_Layer_2_meshShape.rawTotalLight.exr
6. Your lightmaps are almost ready to be imported into Unity. Browse to the **Angry Bots** project location and make a copy of the existing lightmaps and put them somewhere else on your computer. If you want to change the lightmaps back, it is good to have a backup copy. Also take note of the names of the lightmaps, and you will have to rename your **exr** images to match them. For example, the **Angry Bots** lightmaps are called:
 - LightmapFar-0.exr
 - LightmapFar-1.exr
 - LightmapFar-2.exrYour images must be named the same. **The Layer_0, Layer_1, and Layer_2** in your images names correspond to the **Far-0, Far-1, and Far-2** of the original lightmaps.
7. Once the lightmaps are renamed you can copy and paste the new lightmaps into the **Angry Bots** lightmap folder and Unity will auto load them for you. You should now be using your custom generated lightmaps inside Unity.
8. That's it! Your game will now be using your custom baked lightmaps from Vray.

As always if you happen to find a bug, or if you have a suggestion to improve **Export2Maya**, please do not hesitate to contact me (either by email: atrusdni@aol.com or send me a PM on the forums, user name: **Fishypants**). I would like to make this tool as robust as possible, so your feedback and suggestions will help to improve the quality and feature set of the tool.