Porting Placeables to NWN2

Bob J. Hall

Version 1 – 1/24/2017

You may find a nice 3D model available for download from a site and you decide you want to import it into Neverwinter Nights 2 (NWN2). What do you do? I've had some experience in this task, so I'm going to attempt to relate my method of importing a model as a static placeable. Hopefully, this may be of some use to you. There are various alternate techniques available, so I'd also encourage experimentation and looking for other methods.

Tools

For importing 3D models into NWN2, I use the following tools:

- **Blender** version 2.77 or subsequent – 3D modeling tool
  - **Blender MDB Import/Export Plugin** – import or export NWN2 .mdb files
  - **Neverblender for Blender 2.69, 2.7x** – import or export NWN .mdl files
- **GIMP** version 2.8 or subsequent – graphics editor
  - **GIMP DDS Plugin** – import or export .dds files for use with NWN2 (not NWN)
- **MDBConfig** – edit packet header information in .mdb files

I also use the graphical **Vim** editor for editing 2da files, but I'd only recommend that if you are handy with the vi editor.

Required Skills

Before proceding with this tutorial, you will need to be familiar with editing 3D models in Blender. There are various tutorials available across the internet that are very helpful for getting up to speed. Meanwhile, here are some handy commands that I use in the 3D View:

- Numbered key pad keys for rotating the object and zooming in or out
- 5 – toggle perspective/orthographic viewing mode
- Mouse wheel – zoom in and out
- shift-middle-mouse-button & drag – move the scene about
- right mouse button – select part under the arrow
- tab – put the selected object in (or out of) edit mode
- a – select/deselect all visible faces of a part in edit mode
- b – select parts of a model by dragging a rectangle
- c – select parts of a model by dragging a circle about, then type Enter
- g – move the selected part, followed by x, y, or z to limit movement along one axis
- s – scale the selected part, followed by x, y, or z to limit scaling along one axis, then a number to give the scaling factor

- r – rotate the selected part, followed by x, y, or z to define the axis, then a number to give the degrees of rotation
- h – hide the selected faces, then alt-h to unhide them
- p – separate the selected faces into a new part
- k – a free-form cutting tool to insert new edges
- D – duplicate the selected faces, followed by an esc to exit
- ctrl-tab – change the selection mode
- ctrl-e – various options such as subdivide, rotate faces, mark seam, &c.
- ctrl-v – merge adjacent vertices of selected faces
- alt-m – merge two or more vertices
- ctrl-z – undo the last step

These will at least get you started.

Selecting a Model to Port

The first step is to determine whether a model would make a good import project. Here are the things I look for when assessing a model:

- Does it have a suitable license that will allow you to distribute it in your model?
- Are the number of faces too high?
- Are good quality textures included?
- Is the file format one that I can import?

The best licenses for importing a model are Public Domain, CC0, Commercial Use, or Royalty Free. You'll need to examine the particulars, but these types of license usually mean you are free to use the model as you choose. The next best is CC-BY, CC-SA, or Non-Commercial, as you can typically use these with a freely distributed module as long as you include some type of acknowledgement. Personally I like to include credits for all of the models in any set I distribute. Some licenses such Editorial Use or Display Purposes Only indicates you can't distribute the model in your set, so I immediately discard these as candidates.

Ideally, a model will have a low number of triangular faces up to, say, 1-2,000. The larger the model, the more faces it can have without impacting performance. Small models such as a cup or book should be kept well under a hundred if possible. The absolute limit on an MDB part is 32,767 vertices, which limits the number of faces so something around that order of magnitude. You could get around this by dividing your model into multiple parts, but consider that you may end up putting a significant performance hit on the game engine by doing so.

Good quality textures are typically 1024 or more along each side. Smaller parts can get by with smaller textures. The best texturing efforts typically use a texture atlas and show some effort to add interesting details such as wear, stains, and baked shadows to enhance the look. It's not always possible to assess how good the texture is from the preview shots, so you may need to download the model and view it in Blender. The graphics editing tool GIMP can handle a variety of different texture formats, allowing you to export them into the format required by MDB files. But there are other graphics tool that can do a similar job, such as Paint.net.

If a model does not include textures, or they are of lower quality than you would like, you can always do your own texturing. But this is going to add to the time needed to port the model.

As for file formats, I've had the most success trying to import models into Blender that are in .blend or .obj format. (The later often results in over-scaled models, so I have to scale it down to around 2% of the original.) In theory, it also can import .3ds, .dae (Collada), and .fbx formats, although these don't always work reliably. It won't import ASCII .fbx files, for example. SketchUp (.skp) files can be exported via that tool for import into Blender, but those aren't always the best quality. I don't know of a way to import .max files, other than to purchase a license for 3DS Max.

Import into Blender

The first step is to download the model files to a new folder and extract them from the archive. It can require some drilling down through the folders to find the models and textures. I like to move them all to the top level of my working directory so I know what I'm working with. A good set includes the diffuse texture, normal map, and specular texture for all models.

The current version of Blender comes with a set of import scripts that are accessible from the File menu. Most of them are straight forward: just select the file format you want to import from the Import menu, choose the file, then click on the Import button at the upper right.

If the model is not immediately visible, check the outliner panel to see if any new objects were added. The issue then may be one of scaling; the model may be much too large (or too small) to be visible. You can try selecting the object in the outliner, move the arrow to the 3D view, type 's' for scale, then enter a scaling factor. For .obj files, I usually try a scale of .02 (2%) as the object is most likely much too large.

For .blend files, you won't need to do an import, However, depending on the setup you may be placed in a camera viewing mode; type 0 to break out of it.

Extracting the Texture Files

The imported model parts may or may not have a texture attached. You can find out by selecting the part in the 3D View, then select the Material tab in the Properties panel. You should see a preview of the material. The actual texture map for the selected Material is visible under the Texture tab. If you see a 'New' button with no field underneath, then there is no texture assigned. For each object, you will need to assign a texture file here before you export the model to an .mdb file.

As a starting point, I like to assign the diffuse texture that comes with the model. If there is an existing texture, you cal go to the Texture tab than click on the first X in the panel. This will reset you to the New pick. To assign a texture, click on the New then go down to the Open button in the Image section. This will allow you to choose a texture file from your PC.

Once a texture is assigned, I select the object for edit in the 3D View, choose all faces, then go to the UV/Image Editor to view the UV mapping. Ideally it will be nicely mapped out to the texture grid, but this isn't always the case.

There's a whole art to effectively UV mapping a model to a texture file. It's something you'll need to practice to get right. Simply unwrapping a model with the 'u' key in the 3D View may leave you with an uneven UV map that doesn't texture at all well. To get a fairly even layout, you may need to apply Seams (via ctrl-e) to the model in various locations to create breaks in the unrolled UV map. For curved shapes like UV spheres, you will likely need to apply seams so as to unwrap the equator and two poles separately. And so forth.

The "Browse Image to be linked" pop-up menu in the UV editor can be used to select the assigned diffuse texture. Once this is done, you should be able to perform a render in the 3D View and look at the material assignment.

In some cases an object may have more than one material assigned to it. These I like to split up into separate parts–one for each material. In the 3D View, I put the model in edit mode and deselect all faces. In the Material tab of the Properties panel, I select one of the materials and click Select. This will select all faces that have that material assigned. In the 3D view, I then type 'p' to separate by Selection. I then rename that part to reflect the material type. I then do the same for all but one of the other materials.

In the worst case you may end up with an object that has no materials assigned. You will need to go through the steps needed to create your own texture, which will require looking around for a suitable image then editing it in a graphics editing tool. Here it can be useful to export the UV map of the objects. I select an object for editing, choose all faces, then go to the UV/Image Editor and click on Export UV Layout from the Image menu. In the graphics editor I then paste the UV layout as a separate layer and use it as a template for building the texture.

Cleaning Up the Model

For convenience it is best to position the middle of the model at the origin of the 3D View, because that will be the center of rotation in the NWN2 toolset. One way to do this is to put the model in edit mode, select all faces with 'a', then type 'n'. The resulting panel has a Transform section at the top showing the median coordinates of the selected vertices. I find it best to put this in Global model so it isn't effected by any of your rotations and translations. You can now enter 0.0 for the X and Y coordinates.

If this doesn't produce good results, then try selecting two vertices at extreme ends of the model – the resulting X or Y coordinates will then tell you how far you need to shift the entire model in the opposite direction. The vertical position of the model may also need to be adjusted, as the zero coordinate along the Z-axis serves as the ground level.

You will need to carefully check the direction of the surface normals on the model. To do this, put a part in edit mode, type 'n', scroll down to the Mesh Display section and click on the icon with the orange face on the cube. This will cause the normal directions to be displayed. If the normal vectors are uncomfortably long or short, you may need to modify the size of the Normals.

At this point you need to check all around the model to make sure the Normals are facing in the right direction – toward where the player will be viewing the placeable. In Edit mode with Solid viewport shading, the grayness of a face should clue you in to where adjacent, merged faces have opposite normals. If you have any faces with the normal in the opposite direction, they will appear as transparent gaps within the game.

To reverse their normals, select those faces, click on the Shading/UVs tab in the 3D View, the click Flip Direction. There is also the ctrl-n hotkey, which will make a best guess at orienting the normals of the selected face in an outward direction – although not always successfully.

Judging the right scale for the model is a matter of practice. You could compare the size to an interior door, which is 1.8 wide by 2.7 tall within Blender. In other cases you will need to try it out in the NWN2 toolset before deciding on a final scale. I often compare the parts to similar placeables, and resize accordingly.

If the model consists of a large number of parts in the Outliner, then you have some work in store cleaning up and consolidating the components. Many of these parts may be mapped to the same textures, in which case it is a matter of merging the parts. This can be done, for example, by selecting the objects to merge in the 3D View or Outliner, then, in the 3D View, click ctrl-j (join). The goal is to keep the number of separate parts that need to be drawn to a minimum; each separate part requires a draw call by the game renderer.

Some imports may have more than one model, allowing you to create multiple placeables. I like to export these out into separate .mdb files. Unfortunately, the exporter will only allow you to export to everything into one file, so you'll need to delete the parts you don't want to export, do the export, then undo or restore the other parts. You could, for eample, save it all to a .blend file, then keep opening that as you select out each part for export. This can get tedious for a large number of models, so you may need to divide in halves and save – stir, rinse, repeat.

Creating Level-of-Detail Parts

For models with a lot of faces, it's a good idea to create level-of-detail (LoD) parts for your model. These are versions of your model that have the number of faces significantly reduced. NWN2 supports two levels of detail, identified by the part suffixes of _L01 and _L02. You can have one of each for every part in your model, plus skip LoD those parts that can not be slimmed down. For the L01 part, an ideal is to get the number of faces down to 2/3 the total in the main part. For the L02 part, you need to go even further, preferably getting it down to 1/3 the face count. Less is even better – the fewer faces, the less work the gsame engine needs to do to render the part.

The Decimate modifier in Blender now does a decent job of culling faces without losing too much detail. What I normally do is make a copy of the part, select the part, then apply the Decimate modifier from the Modifiers tab of the Properties panel. Next I put the 3D View into Texture mode (with suitable lighting added) so that I can see what happens when the decimate is applied. In the Decimate panel, the Ratio slider can be shifted back and forth to see what happens to the model. I tweak this, rotating the model around until I get an appearance that doesn't change the look too significantly. Finally I click on Apply and add the appropriate suffix to the copy.

Usually I decimate the L02 part first so I get an idea of what Ratio will work. Once I've got that, I do the L01 part and set it half way toward a 1.0 ratio compared to what I used for the L02 part. Hence, if the L02 part was reduced to 0.4, I try an L01 part with a value around 0.7.

Unfortunately, the Decimate modifier doesn't always produce good results. In this case I need to do some judicious manual culling, which can be a lot of work. One approach I favor is to eliminate the bevel edges from parts, then join the two open edges together. Bevel faces are typically much smaller than the other faces, so they won't be missed when viewing the model from a distance. Another approach is to eliminate small model attachments that have a lot of faces, such as handles on drawers. Finally, if the underside can't be viewed in the game, you may as well eliminate it from the level-of-detail parts. Likewise, faces in difficult to see locations (such as recesses or interior faces) can often be eliminated.

Adding Collision Meshes

For MDB model files, there are two collision meshes available, with the type determined by the suffix on the model part. You can include both, one or none. Without a bounding box, the game engine will use the bounding box of the entire model as the collision box.

To add the collision meshes, I typically import another model that has existing collision meshes, then alter those to my needs.

The outer mesh is a level 2 collision mesh with the suffix "_C2", shown as red in blender imported .mdb models. It is used to make a quick calculation to see whether a creature is going to intersect with the general shape of the placeable. This mesh usually consists of one or more simple rectangular boxes that cover the general form, keeping the required number of faces to the minimum necessary. However, there's no requirement that the C2 mesh be made of rectangular boxes. For example, you could use a wedge-shape for a stairway. But in general, simple rectangular boxes will tend to keep the number of faces low.

For me, the goal in creating the C2 mesh is to snuggly cover the form with a simple shape while minimizing any chance than a character's body will intersect with the collision mesh. It's a trade off between minimizing the collision chance and reducing the number of collision mesh faces, so you'll have to use some judgment. For example, if there is an open gap in the shape where the character can enter, then I put boxes around the sides and top of that opening. For a bench, I might have one box for the seat and legs plus a second box for the bench back. With an operhanging roof on a building, one box can be used for the building and a second, wider box for the roof.

The C3 mesh is a more granular collision mesh that is normally triggered after a collision is detected with the C2 mesh. It can have a higher number of faces so as to more snuggly cover the form, but there's no need to go overboard. The fewer faces in the C3 collsion mesh, then the less calculations the PC will need to make to detect a collision. I try to fit the form more snuggly than with the C2 collision mesh, so the C3 mesh may often have double or triple the number of faces. For example, if the model is a cynlindrical column, then the C2 mesh could be a four-sided box with 6 faces, while the C3 mesh is an eight-faced cylinder with 18 faces.

Walk Mesh

The walk mesh determines where creatures can freely move with respect to the placeable. You will have at most one walk mesh per model. Typically, I like to have the walk mesh perimeter enclose the C3 mesh, where possible, while keeping the number of mesh faces to a minimum. You are limited to 128 faces per area grid square, so a large density of mesh faces can make an area walk mesh more difficult, if not impossible to bake. Making the walk mesh slightly larger than the collision mesh will reduce the amount of collision calculations required.

With an mdb file imported into Blender, the type of walk mesh is determined by a *Material* setting. To change the walk properties for a selection of walk mesh faces:

1. In the *3D View*, put your walk mesh in *Edit* mode
2. Chose the *Face Select* option
3. Select only the faces to change
4. Go to the *Properties* panel
5. Select the *Material* tab
6. Click on the *W_...* walk type you want to use, then *Assign*

The red *W_unwalkable* type means that a creature will not be able to enter that face of the walk mesh. The remainder of the types determine the footstep sound used when the creature moves through the face. Typically the type I use is *W_wood* or *W_stone*. The types correspond to rows in the 'surfacemat.2da' file, and probably use the 'fs_*.WAV' sound files.

Exporting Your MDB File

For the final model, you will need to give each of the model parts unique names. These need to be completely unique for all possible models within the game, otherwise the game engine may end up swapping parts and you'll have a complete mess. It's good practice to use a consistent naming convention for this. For example, I might use something like plc_dr_house01_roof for a Drow house roof part, then name the resulting model file plc_dr_house01. The part names can be up to 32 characters long, but 28 characters should be your limit to allow for the standard suffixes.

For export, you'll need the latest Blender MDB Import/Export plugin. There are various export options available, but I now find it is easier to set those using MDBConfig. I use that tool to set my preferences for the materials properties, as well as assigning the diffuse, normal, tint, and glow maps. Here are some of the fields for RIGD (rigid) model parts in the .mdb file:

- Diffuse map – the actual texture that will be seen by the player
- Normal map – a special texture for creating bumpiness on the surface in game. The alpha channel of this file will hold the specularity level
- Tint map – a three color texture for applying the three tint channels to the diffuse texture
- Glow map – any colors here will glow on the model
- Diffuse color – a hue to apply evenly to the diffuse texture; I usually leave this white
- Specular color – the hue that will be applied to any reflected lighting
- Specular power – the maximum reflectivity of the model, which is modified by the specularity in the normal map. I usually give it a value such as 0.2 for dull objects
- Specular value – the smaller the value, the more point-like the reflected light; I use 20.0 for dull objects
- Alpha Test – two-bit alpha channel transparency
- Environment Mapping – applies reflected lighting from the environment, at a performance hit
- No Cast Shadows – the object has no shadows in game
- Projected Textures – this activates the positional glow for the character movement marker

Include in NWN2

To test your placeable model in NWN2, you'll need to add a new row to a copy of the placeables.2da file. The base file is located under your install folder, in Data/2DA_X2.zip. The Data/2DA.zip archive has a text file called placeables.txt, which has some information about the various fields.

As often as not though, I usually just copy one of the existing rows for a similar placeable, then just change the row number, Label, and NWN2_ModelName fields, setting the StrRef to four asterisks. The NWN2_ModelName needs to match the name of your .mdb model file, although the case doesn't matter. The Label field is what you will see in the Appearance menu within the toolset. The row number fields must be unique and contiguous, so that row 3452 follows row 3451, for example.

Now I copy the placeables.2da, .mdb file, and all required textures into my Documents/Neverwinter Nights 2/Override folder and launch the toolset. From the Placeables Blueprints list, I make a copy of an existing placeable then click on the Appearance menu and find the matching Label I assigned.

If the model doesn't appear in the Area view, there is probably an error in the placeables.2da file. Usually this is because the NWN2_Modelname doesn't match your .mdb file name. If the model appears, but it shows an odd-looking texture with light and dark patterns, then there's something wrong with the texture assignments. The texture files may be misnamed in the .mdb file, or in the wrong slots.

Once you get it looking right, it's time to make the final adjustments. You may need to change the model scale to something more suitable. If you spot any open gaps in the model surface, the faces in the model may have the Normals reversed.

When you are ready to publish, please visit the Reserved 2da ranges page on the NWN2Wiki and check out the rows you need from the Placeables.2da file. You may need to pad the file to the row numbers. It's often convenient to put your work in a hak file for use by module builders. For this I use Tanita's NWN2Packer v1.9. Just do the following:

- put your hak file in your Documents.Neverwinter Nights 2/Hak folder
- open your module
- select the Module Properties from the View menu
- in the Module Properties panel, click on the Hak Packs field
- add your module from the Hak folder
- save the module

You should now be able to access the placeable appearances you created. At this point I start creating blueprints for the various placeables. When I am done, I export the blueprints via the Export... pick under the File menu. The resulting .erf file is included in the hak pack distribution.