NWN2 Toolset Tutorial

(atiaxi @ gmail . com)

October 8, 2007

Contents

Introduction

The NWN Vault (see Appendix A) has a number of tutorials for the Neverwinter Nights 2 Toolset, but none - as of this writing - that take you through the process of making an entire module from scratch. Neverwinter Nights 1 had a Bioware-created tutorial which did exactly this. What I've done in this document is to "port" the NWN1 tutorial to the NWN2 toolset.

What's in the tutorial?

(Almost) everything you need to know about the NWN2 toolset. How to create areas, link them together, populate them with friendly objects and not-so-friendly monsters, create quests, and even some scripting. Everything that was in the original tutorial has been included. For an exhaustive list, consult the table of contents.

What's not in the tutorial?

Some specific NWN2 topics are not covered - for instance the campaign editor, which allows for easier linking between modules. World maps. The plot summary present in the original tutorial [1]; I'm in murky enough legal waters just porting the guide over, I'd rather not copy-and-paste entire sections from the original.

[1] If you'd care to read it, it's at `http://nwn.bioware.com/builders/modtutorial.html`.

A word on plugins

The NWN2 toolset comes with the ability to write plugins for it - to extend the functionality of the toolset itself. People have used this functionality to do many useful things; for instance, NWN1 had a creature wizard - some enterprising people recreated it as a plugin for NWN2. Others address problems that will be mentioned in this tutorial (i.e. inability to preview music)

That said, if you're new to the toolset adding plugins may seem a bit out of your league. This tutorial will assume you don't have any aside from the ones that come with the game (which we won't be using, anyway).

If you'd like to learn about plugins at any time, there is a short tutorial on using them in appendix B.

A word on versions

The instructions on this tutorial were written for the 1.0.1115.0 version of the toolset, and 1.10.1115 (english) version of NWN2. Considering I started writing it on version 1.6 (no, it didn't take that long to write, I just hadn't updated in a while), it's probably safe to say that it'll work for other versions as well.

A word in general

In my experience, the toolset likes to crash randomly, often during autosave. You might be tempted to turn off the autosave off, and indeed this works for a number of people. For me, though, there were other crashes, so the autosave was greatly helpful in making sure I didn't lose my work. The point of this section is simply to say: Save Often!

1 Fern, the Frontier Town

Introduction

This tutorial introduces the New Module and New Area wizards and explains how to paint terrain.

1.1 Creating a New Module

Creating a module is pretty straightforward. By default, whenever you start up the toolkit, an empty module will have already been created. If you've been working on a module already and want to create a new one, you can use the 'File' menu to pick 'New' and then 'Module'.

The first thing you should do is to save this empty module. From the 'File' menu, select 'Save' and type in a name. (I'm going to use 'tutorial' here, but you don't have to.) A message will pop up saying that your module doesn't have a valid start location, but we'll fix that soon enough by creating an area.

1.2 Creating a New Area

To create a new area, follow these steps:

1. From the 'File' menu, select 'New' and then 'Area'
2. Next to 'Area Tag', type in a quick name for the area. This isn't the name that will be shown to the player, it's just for toolset use - names like this are referred to as tags, and you'll be seeing a lot of them. I'm using 'area_fern'.
3. The default choice next to "Area Type" is Exterior or Interior. As this is a town, we'll go with the default.
4. Click 'Next'
5. This next screen will allow you to determine how large you want the area to be. You can choose from the built-in sizes on the right or pick an exact size using the sliders on the left. The default of 'small' is fine for us, so just press 'Finish'

That's it! You've just made your first area, a big flat grassland. If you'd like to take a look at it from inside the game, just use the 'File' menu and pick 'Run Module'. A warning will pop up, telling you that, unless you've baked your area, you won't be able to walk around.

1.3 Baking

A major difference between the NWN2 toolkit and the original is that outdoor areas are no longer at a pre-set height. You can make rolling hills, deep canyons - whatever you'd like. The trade-off for this is that sometimes you have to specifically set where your player can and cannot go. For the most part, the toolset will do this for you.

What does this have to do with Baking? Baking is the process of taking the walk/can't walk information and putting it into the module. Every time you change an area, you'll probably need to re-bake it.

To bake an area, go to the 'File' menu (cancel out of the earlier 'you must bake your area first' warning if you haven't already), choose 'Bake Current Area' and say yes when it warns you it might take a while.

Once it's completed, choose 'File' again and 'Run Module'. You'll get the same warning - it doesn't seem to know when you have and haven't baked things - but this time just hit 'ok' and after a time the game will start up, already loading your module and with a default character to run around as.

When you're tired of running around on this plane, exit the game and come back to the toolkit.

1.4 Camera Control

When first created, areas appear pretty close-up. To change this, you can use the keyboard and mouse to view everything differently:

Action	Mouse
Pan	[Ctrl] + left mouse drag
Change height	[Ctrl + Shift] + left mouse drag
Rotate	[Ctrl] + right mouse drag OR Middle mouse drag
Zoom	Wheel mouse

A word on Tiles

For this demonstration, make sure your "Grid" option is turned on; it's in the upper left right below the "File" menu. With this on and the camera zoomed out enough, you'll see a grid of red and black lines, with a white box centered around the default start area. Anything inside the white box is somewhere the player can walk - anything outside is painted there just for decoration. The areas cordoned off by the black lines are "Megatiles" - for the most part you don't have to worry about the distinction but there are times where it matters (and these times will be pointed out in the tutorial)

If you were paying close attention, you may have noticed a discrepancy - when you created this area, you specified 8 tiles by 8 tiles, but the grid is only showing you 4x4! No, you weren't cheated out of extra tiles; turn on the "Surface Mesh" option and, if you look carefully, you'll see light green lines indicating the actual tile locations.

For the most part, you'll probably want "Grid" on, and "Surface Mesh" off.

1.5 Painting Terrain

The bottom-right part of the toolkit has tabs labeled "properties", "Blueprints", "tiles", and "Terrain". For this part of the tutorial, we'll be dealing with terrain, so select that tab. Keep in mind that the interface of the toolkit is insanely customizable, so if the terrain tab is too small for your liking (it is for me, and I've got a widescreen monitor!) you can easily resize it.

1. Under 'Environment Tools', press the 'Terrain' button.

2. Your cursor in the main area should change to a circle inside another circle. This indicates what will be affected by your painting - the center circle is affected more than the outer one. As Fern is a mining town, it'll need mountains. Go to the north end of town just past the white border (again, anything painted outside this border is for decoration only, as the player can't go there) and press the left mouse button. You'll see the terrain rise.

3. As you can see, the longer you hold down the button, the higher your mountain becomes. We want to create a large mountain range, so slide up the 'Size' slider to 60, and the 'Outer' slider to 20. This first slider

controls the size of the inner circle, and the second one controls the size of the outer. With a cursor that large, you can easily create a huge mountain range north of the area your characters can walk in.

4. If you've made a mistake and made some of the mountains too tall, you can click the 'lower' button under the 'terrain tools'. This works just like the 'raise' tool you have been using, except of course it lowers terrain. This isn't just for making mountains, you can create valleys and canyons with this.

5. The 'Noise' tool creates areas of bumpiness. You've already made the mountains but if you want to make, for instance, rougher but still passable terrain, the noise tool may be helpful to you. Just be careful on areas where tiles overlap - the bumps on one tile may not line up at the border!

6. If you're like me, the first time you did this your mountains ended up a bit... pointy. That's where the 'Smooth' tool comes in. Press the "Smooth" button in the terrain section and then press the 'G' button in the "Brush" section above it. You'll see it'll change the size of the brush for you. These pre-set sizes can be helpful if you'd rather not manually set the sliders yourself. Now, find the spiky portions of your mountain and press the button over them. If you smooth it too much, you can always raise/lower it back again.

7. One good way of making plateaus is to hold down the button in 'Smooth' mode. Another is using the 'Flatten' tool. By default, 'Flatten' will change whatever section you use it on back to ground level. Helpful if you've accidentally made a volcano in the middle of town! To change the height that Flatten will use, you can scroll down in the Terrain tab until you see the Eyedropper button. Press it, then click anywhere on the terrain. The point you've chosen is the new Flatten point! Try to create a plateau somewhere on the mountainside, then smooth things over so it doesn't look unnatural.

8. The final terrain tool to look at (for now!) is the 'color' tool. Click the 'color' button and then the square next to the word "Color" under the Eyedropper button, and you'll paint the scenery whatever color you want. For instance, if you wanted your mountains to look a more realistic brown, you could select the exact shade you'd like, and then paint them all. Keep it mind it's more of a tint than making it that exact color - if you mess up at any point, use White, as that's the same as no tint at all.

9. The mountains are nice, but even if you do paint them brown, they still look like grass. We're going to use the "Texturing" Environment tool (you may need to scroll up in the Terrain tab to see it) to fix that. Press the Texturing button. You'll see a brush much like the same one used to move the terrain around in the first place; it works exactly the same way. Below that, though, you'll see the currently selected texture and a

list of all of them. Pick a texture that looks appropriate for mountains; I used TT_GR_Cliff_02 for the vertical parts of the mountainside, and TT_GR_Rocky_03 everywhere else. Even when you paint the textures, you'll see the grass still there underneath; this is because the engine blends whatever texture you're currently using with the one underneath it. You can change the degree to which this blending happens by adjusting the "Pressure" setting - 0% is somewhat pointless because your new texture won't show up at all, and 100% will completely cover up whatever's underneath, which is more useful. Even if a texture is completely painted over, however, it's still considered to be there by the engine.

10. Underneath the textures is a portion called "Selection (Advanced)" which tells you what textures you're using on the current megatile. Even though NWN2 lets you paint terrain however you want, it still thinks of things in terms of tiles (or in this case, 4x4 areas of tiles, i.e. the Megatiles), and there's a limit to how many textures you can have in a tile (Six, including the default grass). Also, eep in mind that the more the variety of textures in your area, the harder it will be on your video card. How do you get rid of a texture if you make a mistake? It may initially seem like you can't, but there is a trick to it:

    (a) Spray some TT_GM_Mud_01 on an unimportant tile (This isn't part of the general process, I'm including it here so we'll have something to erase)

    (b) Make sure the tile you want to modify is selected - you can do this by pressing the "Select Terrain" button and then clicking on the tile with the offending texture. Unfortunately, this process selects the Megatile that your tile is in. The Megatile level is as fine-grained as you can go with texture changes.

    (c) Now go to the Terrain tab and press 'Texturing'. Go down to "Selection (Advanced)" and you'll see all the textures that make up your selected tile. Press the icon next to TT_GG_Grass_19 to select it.

    (d) Now change the mode back from "Select Terrain" to "Paint Terrain", and change the pressure to 100%

    (e) Paint over the offending mud, and you've restored everything to grassy goodness!

11. If you go back to selecting your tiles, you'll see that the texture you painted over is, as promised, still counted among the textures on the tile. As a result, the above technique is mainly good for touching up if you've painted your rock texture onto the ground instead of the mountain and need to push it back a bit by painting over it with the original grass. But suppose you're tired of the original grass. You want to replace it entirely with a different texture: the Swapper tool lets you do this! In fact, if you swap a texture you don't like with one that's already in the tile, that texture

will be deleted[2]. Here's how to use the swapper to make our grass a little greener:

(a) In the terrain tab, under "Texturing" and "Selection (Advanced)", select the TT_GG_Grass_19 again.

(b) Press the "Swapper" button in "Terrain Texturing" above. The swapper dialog will appear - this is like the texture equivalent of Find and Replace. On the left will be your currently selected texture, and on the right will be what you want to change the grass to. I went with TT_GG_Grass_12 for that more forest-like look.

(c) If you don't like the change, you can always undo it, either by using Edit -> Undo or using the swapper once again (the latter is helpful if you know you don't want the original texture)

12. Another way to give your area flavor is to use the "Grass" environment tool (below 'Terrain' and next to 'Water'). If you click it, you'll see a similar 'brush' control as before. Below that, you'll be choosing the blade size, and how they vary. The default size is fine, but unless you want something to look like it's just been mowed, change the variation to something like 0.33 - this will ensure that each blade is a little taller or shorter than its neighbors.

13. At the bottom, you'll see the types of Grass you can paint. Unlike textures, these are actual 3D objects that'll be popping up out of the ground, so be aware that if you saturate an area with them performance is going to suffer accordingly. I'm painting some wheat around the south and west sides of town(specifically, AG_Wheat_01).

14. Finally, let's make a lake to the east of Fern. Press the Terrain button at the top of the Terrain tab, then lower an area to the east.

15. Press the 'water' button up top, next to 'grass' and below 'texturing'.

16. Like all the other tools, water has a similar 'brush' section. By default, water will be painted at ground level, which usually causes problems. You'll likely want it slightly below ground level. You can change this with the 'height' slider, or use the Eyedropper tool (the same way you used it for Flatten, earlier) to set the desired height. I'm setting my height to -3. Then, paint in the water over the hole

Bake the area, then save your module. Feel free to start up the game and wander around the new, more picturesque area.

[2]As of version 1.10, this appears to work correctly - however, if you have an older version of the toolset, I highly recommend that you upgrade - older versions needed to restart the toolset for the change to even take effect, and could - in worse cases - end up corrupting the tile so you painted a black texture instead of what you wanted!

1.6 Painting Blueprints

Terrain's all nice and good, but you'll probably want to have a building at some point, too. Here's how to set it up:

1. Click the "Blueprints" tab. Everything in the game that's not terrain or the grass objects are here. Again, the default size may be a little small; you can resize at will.

2. Within the Blueprints tab, select 'Placeables'. Placeables range from small decorations to large buildings; we'll be putting up one of the latter. Drill down from "01 - BUILDING PROPS" and find "Inn {Tint}". Select this.

3. If you move your cursor around the map, you'll see the inn attached to it. Don't worry if it's not facing the direction you want it now, just pick a spot and press the mouse button to place it there.

4. Once you've done that, you'll notice there's another inn still attached to your mouse. To get rid of this, simply press Escape or click the 'Select Objects' button.

5. Select the inn; you can do this via the 'Select Objects' button, or by looking in the "Area Contents" tab in the lower left. This tab shows you everything you've placed, and it's laid out like the tabs on the right that placed them. Once you've selected the inn, you can move it somewhere else or rotate it. Rotating is done by clicking the various "Rotate 45 degrees" buttons on the top, or by shift-right-clicking on the object.

6. Now we're going to add the cave for our mine. On the 'Placeables' tab, drill down from "03- NATURE PROPS" and select "Cave Entrance {01}". Place it up at the north of town, near the mountains - but be sure to keep it within the white border, otherwise the players may not be able to get near it!

7. Go to the 'terrain' tab, select the 'Terrain' environment tool, and play around with the area surrounding the cave to make it a bit more realistic. To make sure the terrain underneath the cave is flat, press the 'select objects' button up top, select the cave, then press the 'Flatten Under' button at the bottom of the Terrain tab (near the eyedropper).

8. Now's the time to discuss the terrain tools left out of the last section: Walk and Non Walk. These both control where the player can and can't go. Press the Non Walk button.

9. The screen will be overlaid with a mesh of green, black, and yellow. The black areas are out-of-bounds (make sure your cave isn't on this), the yellow areas can't be walked on, and the green areas can. Using the Non-Walk brush, you can make other areas unwalkable. You probably don't want people scaling the cliffs you just made around the cave, after all, so

mark them unwalkable now. If you make a mistake, you can use the Walk button to make an area walkable.

Once again, bake, save, and try the area out. This time you can see what the effects of the walkable areas are.

1.7 Starting Location

Finally, there's the starting location. By default, it's in the middle of the first area you create. You can select it and move it by using the 'Select Object' button and the mouse, or you can place a new one by clicking the 'Set Start Location' button and clicking anywhere in the area. There can only be one start location for the module, so any time you place a new one, the old one is lost. You can rotate the location; the player will start facing whatever direction you've got them looking. I placed the start location right in front of the inn, looking toward the mountains.

Extra Credit

Add more terrain to the outskirts of Fern; expand the lake into an ocean, or add more plateaus to one of the sides. Use texturing to draw a road that leads from the outskirts to the mine or to the inn.

2 The Fernesk Mine

Introduction

This tutorial covers indoor terrain and other placeable objects.

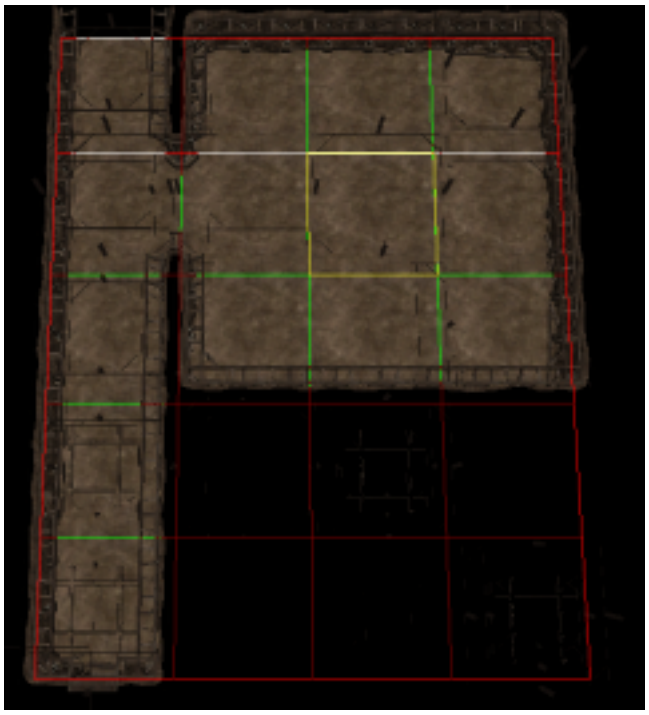
2.1 Interior Areas

The vile Gnashgab and her warriors have invaded the Fernesk mine! Of course, in order to do this, they're going to have to have a mine to invade.

1. Start up the toolset and load your tutorial module, if it isn't already. From this point on, I'm going to assume you either have the toolset up and running with the module you're working on already loaded, or that you know how to do so.

2. There's more than one way to create an area! On the upper-left part of the toolkit, you'll see a tabbed window with labels like "Areas", "Conversations", "Scripts", etc. Click the "Areas" tab (if it's not already selected), and right-click inside the window. Pick "Add".

3. I'm using the tag 'area_mine' for this. Given that this is a mine, it counts as being 'Interior', so set that as well before hitting 'next'.

Figure 1: 

4. We'll be using a custom size for the caves, so make it 5 tiles high by 4 tiles wide. Hit 'Finish' and behold the blackness that is your new area! If in fact blackness is all you can see, make sure your "Grid" and "Occlusion Grid" options are both selected. You should see a black area divided by a red grid.

5. Unlike the outdoors, Indoor areas are created the same way they were in NWN1: Tiles. Each square in the red grid is a tile (no megatiles to worry about in the indoors!) The 'Tiles' tab should be on the right of your window, next to the 'Terrain' tab. Select it.

6. Now drill down from "Standard_Mine" and pick "Door(Hall_End_1_Door_Variant)". Your cursor will now have a hall ending on it. We'll want to put it in the lower left (This is Row 0, Column 0 - you can tell by looking at the 'terrain' tab under "Selection (Advanced)" with the texture option selected. Granted, that's a bit unwieldy a way to look) but it's not facing the right way! Right-click to position it so the door is on the south.

7. Select "Hallway" now. Again, it needs to be rotated to get it facing the right direction. Once you've done that, fill in the entire column with hallways!

8. Now, we're going to create a 3x3 area in the middle of the map. Choose 'Open Floor' from the list and place it two squares over from the hallway and two blocks down from the top (Row 3, Column 2). Then:

   (a) Put a "1_Wall" tile directly north of the center tile.

   (b) To the northeast, place "2_Walls" (You will need to rotate it to get it to fit correctly)

   (c) The rest of the 3x3 area can be filled in using only "1_Wall" and "2_Walls", suitably rotated, with the exception of the area we're going to be using to connect it to the hallway (This will be the tile to the west of the center, and the hallway to the west of that). Try this now, putting "1_Wall" to the south of center, "1_Wall" east, and "2_Walls" southeast, northwest, and southwest.

9. Choose the first "Floor_2_Corners" (It's the one with two dots on top), rotate the tile s the 2 corners are facing the hallway, and place it west of center.

10. Finally, we connect the hallway to this area. We're going to replace the hallway west of the tile we just placed with "1_Wall_2_Corners" Again, rotate so the tiles match up with the 3x3 area, and the wall is lined up with the rest of the hallway. What you've created should look something like Figure 1.

2.2 Resizing the Area

There's no simple menu option for resizing areas, and at first it looks like it may be harder than you'd think. It turns out, the option is somewhat hidden but you end up with more control over how the area resizes.

1. In the 'Areas' box (by default, in the upper-left), make sure "area_mine" is selected.

2. Next, click the 'Properties' tab (It's next to the 'Blueprints' and 'Tiles' tabs)

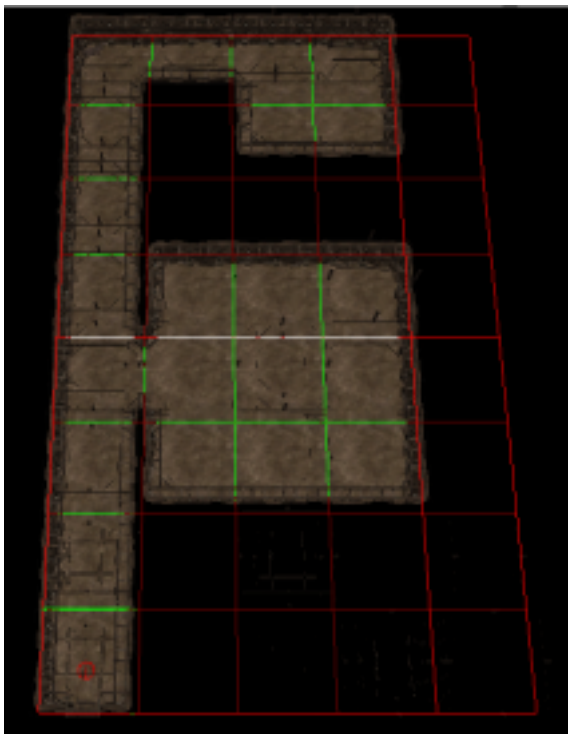
3. Scroll down through the properties to 'Size' and click the '...' button.

4. Not only can you resize your area, but you can decide how it's being resized. Hitting '+' on the top box, for instance, will add blank tiles to the top of your area. In our case, we need to add 3 more tiles to the top and 1 tile to the right, making our area 5x8.

5. Fill the leftmost column with hallways again, except for the very top.

6. Put "2_Walls_1_Corner" in the upper-left corner of the map.

7. Use the "Hallway" tile to the east of that tile.

Figure 2: 

8. We'll create a 2x2 area here in the same way we make a 3x3 one earlier (we just won't have the center tile). East of the Hallway tile you just placed, put a "1_Wall_1_Corner_Variant" (The first one)

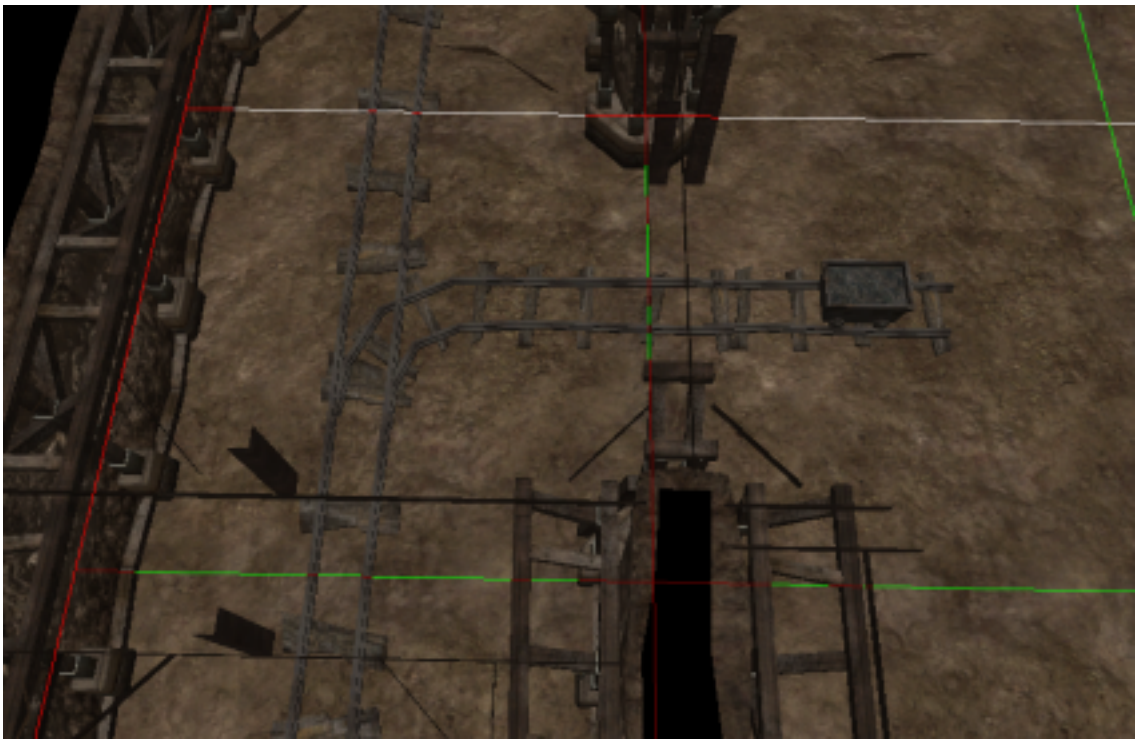
9. Put "2_Walls" East, South-east, and South of that tile to complete the 2x2 area. Your mine should look something like figure 2 at this point.

To walk through what you've just created, you'll need to place the module's start location (Using the "Set Start Location") somewhere in the mines; It's probably easiest to put it by the entrance. Once that's done, save, bake, and run!

2.3 Game Objects

You had a sneak peek of the Blueprint tab in the first tutorial, when we put an inn down in town. It doesn't just have buildings, though - it also has the smaller sort of items you'd need to make caves something other than featureless caves!

1. In the Blueprints tab, with "Placeables" selected, under "02 - MANMADE PROPS", choose "Mine track {2}". We're going to place tracks all along the hallway. Place one by the door now.

Figure 3: 

2. There are a number of problems. First, it's facing the wrong way. Select it (In 'Select Objects' mode), and click "Rotate Object 45 Left" twice. Secondly, it's not a very long track. While we could copy and paste it over and over again to get the amount of track we need, there's a somewhat easier way.

3. Make sure the object is selected, and go into the Properties tab. Under "Appearance", you'll see "Scale". Change it to "2,1,1" - the tracks will be larger (but more stretched out!). You can use this technique for any placeable object, so it's great for customization - just remember that scaled-up things don't always look as good as they did before, especially if (as we did), you only scale along one axis.

4. With the rail selected, choose 'copy' from the Edit menu (or press the keyboard shortcut, Ctrl+c) and then 'paste' (Or Ctrl+v). A copy of the rail will be attached to your cursor! Place it above the first one (connected, of course). Repeat to put rails on the whole hallway!

5. Take a "Mine Track {3}" and place it near the entrance between the hallway and 3x3 area. Rotate it 180 degrees and make it so that the bottom overlaps with the track you've already laid down.

6. Put a "Mine Track {2}" at the other end of your new spur, and a "Mine Track {1} after that. Now you have an actual mine cart in the area where mining would be done, and a way for it to connect back to the main hallway. If these instructions were less-than-clear, look at figure 3.

There's an interesting caveat at this point. Bake and save your area, then run it in the game. Try to go to these new tracks, and you'll find your way blocked!

Half Baked

Earlier in the tutorial I mentioned that, generally, the NWN toolkit does a good job of determining where your character can and can't go based on terrain and the items you've put down. This is one of the cases where it doesn't. The overlapping tracks confused the game into thinking that they're actually blocking off movement. To see the result of this in the toolset, turn on the 'Surface Mesh' and 'Baked' options on your toolbar. The walkable areas are in yellow, the unwalkable white. The tile with two rails on it is encased in white.

How to fix it? Well, this is indoors so unfortunately it's not as easy as painting a walk/don't walk brush over it (and even outdoors that doesn't always work). What we can do is tell the engine that the tracks don't matter for collision purposes.

1. Select all of the mine tracks (with the exception of Mine Track {1}, we don't want the player to be able to walk through a mining cart!). The easiest way to do this is by going to the "Area contents" tab on the left, clicking 'placeables', drilling down from "02 - MANMADE...", clicking the first mine track you see and then shift-clicking the next to last. You should see (almost) all the tracks light up.

2. Right-click in the editing window with the tracks. A menu should appear; choose the "Convert" option, then "Placeables to Environment Objects"

3. Once you re-bake the area, all will be well! You don't even have to start the game up to test this, you can turn the same options as before ("surface mesh" and "baked") on and look - you'll see all the tracks except for the mine cart are now walkable. While you probably don't want to be in this mode the whole time (in fact, feel free to turn it off now), these options can be invaluable - if you're ever having problems with areas being impassable and you didn't mean that to be the case, turn them on and track the offending Paintables down!

If you run the game at this point, you'll see you can effortlessly pass through the tracks. In fact, if you look closely at your character, that's exactly what's happening! The collision detection is entirely missing for Environment objects, so be careful you don't convert anything that's supposed to block the player into one.

The Mine's starting to shape up, but there's one major problem - there's nothing to mine! We'll change that soon enough:

1. Under "03 - NATURE PROPS", choose "Rock Face {01}". The first thing you'll notice about it is that it's huge! Place it along the southern wall of the 3x3 area.

2. Get rid of the other Rock Face attached to your cursor by hitting 'escape' or pressing the 'Select objects' button. Now select your rock face and rotate it 45 left, and move it so it's taking up the southeast half of the room.

3. If you were to fire up the game right now (after baking, of course), you'd find that everything works as you'd expect - the giant blocks of ore are non-walkable and you can get to them just fine. So now we're going to mix things up a bit. Select and right-click the rock face and choose "Convert", then "Placeables to Environment Objects"

4. "Why?" I hear you asking. If you were to bake and run the module now, the player could pass right through the rocks, which is clearly not what we want to happen, and everything was workign fine before. But the more placeables in an area, the more complicated it is and the more strain it puts on the CPU. Environmental Objects, on the other hand, require less resources. Still, we don't want the player to walk through the rocks, no matter how many more FPS it would give us! So, choose "Triggers" next to the Placeables button.

5. Drill down on "Empty" and select "Walkmesh Cutter".

6. It's not exactly intuitive, but Walkmesh Cutters are a kind of trigger. Normally you'd use triggers to make things happen when the player enters them, but this time you'll be using it to prevent exactly that. Triggers can be any polygon - you click once in the edit window to indicate a point, then keep clicking for any additional points. You can press 'escape' when you're done. Draw a Walkmesh cutter around the rocks. It's probably easiest to do this viewing from the top-down. You'll need to click once in the upper-right of the room, once in the lower-right, and once at the southwesternmost point of the rocks. If the trigger doesn't end up the way you want it, you can delete it and try again. Because the rocks are probably in the way, the easiest way to select the cutter is, again, from the "Area Contents" window.

7. Bake the area. Turn on the "Surface Mesh" and "Baked" options again to see the results of your handiwork. Again, if it isn't to your liking you can easily delete the trigger and repaint, or just right-click with the trigger selected and choose "Repaint Trigger".

This was a somewhat contrived example for how to use a Walkmesh cutter - there's little risk that a single rock face is going to overload the player's PC. Normal circumstances where you'd use a cutter are if you really do have a great many objects that are bogging the scene down and need to make them

Environmental but not allow the player to pass through them, or if you end up having to convert something to an Environmental object because its tile got overloaded (like we did before with the mining cart rails) but want some of the objects you converted to still not be passable.

Now we need to decorate the upper 2x2 area!

1. This place used to be the office and break area for the miners - now it's broken up. Go back to Placeables and pick "Table{Broken02}" from "02 - MANMADE PROPS" and put it in the northeast corner. Throw some "Chairs{Broken01}" over there too, while you're at it.

2. The goblins didn't come in peace, and some of the miners weren't giving up without a fight. Choose from the many corpses available to you - I personally went with Corpse 9, 12, and 14 - and litter the entrance of the 2x2 area with them. Be sure to rotate them a bit so it doesn't look like they're lined up: They didn't die orderly-like, and the Goblins sure aren't going to straighten them up! If you place too many or have them cross too much of the entrance they'll make it impassable - either make them environmental objects and use walkmesh cutters to ensure the player can't trample over them or just move them to make a path.

3. Finally, in the southeast corner of the room, place an "Altar {3 - TINT}". Objects with TINT in their name can be further customized!

   (a) With the altar selected, go to the 'Properties' tab. Under "Appearance" you'll see "Tint". Expand that entry and you'll see three colors.

   (b) Select any of the colors, and you'll see a down-arrow on the far-right of it. Pressing it will let you choose a new color.

   (c) The colors will change in real-time as you go, so once you've brought up the color chooser you don't need to dismiss it to see what it's done. (You may need to zoom in and/or position the camera well first, however) I made the border of the sash black and the main part red, but what changes you want to make (if any!) are up to you.

4. There's one more object to place: The quest objective himself, Jacen. The Goblin hasn't been kind, either. Choose one of the corpses you didn't use earlier (I went with Corpse 13, it looks more harshly treated) and place it near the altar.

5. Before we move Jacen to the altar itself, we'll need to raise him up so he'll be atop it. You can move a selected placeable up and down by either using the PageUp and PageDown buttons or holding down the Alt key and moving the mouse. Of course, if you'd rather not fiddle with getting it perfectly right s that it looks like he's on the altar and not floating above it or partially embedded in it, there is an easier way. Select Jacen, then the "Properties" tab. Under "Misc", there's a "Stackable" property - set it to true, then simply drag Jacen to the altar and he'll automatically

be placed on top! If you need to fine-tune his location, you can still use the other ways of moving him vertically. Whatever way you prefer, move Jacen on top of the altar and rotate him so he's lined up correctly.

2.4 Encounters

Even though it's not under the "Triggers" category, an Encounter is drawn like one, and behaves like one. The player walks into a designated area and monsters spawn where you indicate. These spawns can be dynamic - fewer monsters for one character but more or harder monsters for a large party. Encounters are the easiest way to get fights started.

1. In the "Blueprints" tab, select "Encounters". This is next to "Triggers".
2. Drill down from "Very Easy" and choose "Goblin Gang".
3. Your cursor changes to a + symbol - the next thing you do is draw the trigger (the same way you'd draw a walkmesh cutter). When the player enters this area, the monsters will spawn. I drew mine a tile up from the entrance of the mine. Make sure it covers the entire width of the hallway, as you don't want players sneaking past it!
4. Make sure your trigger is selected (You can select it by using the Area Contents window under 'Encounters) and then click the 'Properties' tab.
5. Under "Behavior", change "Minimum number of creatures" to 2 and Maximum number of creatures to 2. Change "Spawn once?" to True unless you like endless tides of goblins. Finally and most importantly, turn "Active" to 'True'.
6. Now the encounter needs someplace to spawn the enemies. You need to put one spawn point out for every possible creature to be spawned by the encounter. In our case that's two, but keep it in mind if other encounters you make stop working. Ensure the encounter is still selected, then press the "Paint Spawn Point" button (next to "Set Start Location") and put two spawn points where the hallway meets the mining area. (I had to rotate them so they'd be pointing toward the player)
7. Place another, similar, encounter just north of where the goblins spawn. Put the spawn points for this one in the upper-left part of the map, where the hallway turns toward the miner's former office.

You can tell which waypoints are associated with which encounter by selecting the encounter - the spawn points of that encounter will have red flags on them, whereas all the others remain yellow.

If you wish, you can test the module now - but be careful! The default character has no weapons so you're likely to take a pounding even against a few goblins.

2.4.1 Creatures

Encounters aren't the only way to get bad guys into a level. You can also place them from the list of all creatures!

1. Once again on 'Blueprints', select 'Creatures' (The second button of the bunch).
2. Expand the "Humanoids" node and select "Goblin". This will be our Gnashgab, after some modification. Place her near the Altar.
3. Making sure she's selected, go to the Properties tab and scroll down to "Basics". Change her name to Gnashgab. If you like, you can also change the Localized description to be what you'd like Gnashgab to be described as.
4. Scroll down to "Character Sheet" and select "Classes". Click the '...' at the end.
5. In the dialog that pops up, select 'Humanoid'. The Class and Level options should appear to the right. Change the 'Humanoid' class to 'Cleric' and the level to '2', then press 'ok'.
6. Make a woman out of Gnashgab by changing the gender to Female. Not that the appearance will reflect this, but the important part is that we know.
7. Change 'Base Hit Points', 'Current Hit Points', and 'Character Sheet hit Points' to 10.
8. Our goblin's not going to be much of a shaman if she can't cast spells. Change her Wisdom to something sane like, say, 14.
9. Go to the 'Spells' tab. It's far right of the 'Preview' and 'Inventory' tabs along the top; if you can't see it, press the arrow buttons on the tabs until you can.
10. Pick "Cure minor wounds" from the list and hit the '>' button. You may need to resize the table somewhat to be able to see this, but so far our cleric's memorized 0 of this spell. Change that to 1.
11. Repeat this process for Virtue and Inflict Minor Wounds, but give her two charges of that.
12. Change 'Level' to 1, and then add 1 of "Bless", 2 of "Cure Light Wounds", and 1 of "Inflict Light Wounds"
13. Finally, some customization. Since any changes you make will be reflected almost immediately, you may want to zoom in on our goblin leader. There's not much we can do to make our goblin look female, but we can make her bigger - go back to the 'Properties' tab and change the scale to 1.5,1.5,1.5. Play with the tints if you'd like to further differentiate her.

Extra Credit

If you'd like to test how this works as an actual battle, drop some weapons in the level, near the entrance. You can drop items the same way you place other placeables (except they're in their own section, "Items"). It's amazing the difference a longsword and some chainmail make!

3 Doors and Area Transitions

Introduction

Now, I'll show you the wonderfully enthralling world of doors. Even if that doesn't sound exciting, your player's not going to go very far without having the areas linked up in some way!

3.1 Doors

Doors are like other placeables, only they're very restricted in where they can and can't go. Essentially, any time you put a placeable or tile down that has a doorway, you can put a door down there. Luckily, pretty much anything that looks like an entrance can have a door. In fact, unlike NWN1, most things which look like they ought to have a door already do! The inn you placed in town, for instance, has two. The cave entrance outdoors even has one! Not all tilesets that can have doors do, however. We can change that:

1. Open the "area_mine" area.
2. At the entrance of your mine there's a doorway but it leads to nothing but darkness! Go to the "Blueprints" tab.
3. With "Doors" selected, drill down from "01 - INTERIOR DOORS" and select one that you like. I went with "{Caves Door}" because it looked more sinister than the others.
4. If you put your cursor somewhere in the area (not near a doorway) you'll see the door attached to it. If you move your cursor next to a place where the door can be, it'll snap into place automatically. All you need to do then is click the mouse to place the door. Do this now with the door you've chosen and the empty mine entrance. Be sure to check the Area Contents window to make sure you only placed one! The player's not going to be happy if they open a door to see another door.

Doors behave pretty much like you'd expect. They can be opened, closed, locked, bashed in, and - probably most importantly - can function as area transitions.

3.2 Area Transitions

As beautiful as they can be, the game would be somewhat boring if we could only stay in one area. Area Transitions allow the player and the party to move from one area to the other. They can be from doors to doors, Area Transition Trigger to Area Transition Trigger, or any mixture of the two. This tutorial will focus on doors:

1. Open the "area_fern" area.
2. In the "Area Contents" window, select "Doors".
3. Drill down from "02 - EXTERIOR DOORS" and select "{Cave Entrance 1}"
4. In the "Properties" tab (under "Basics"), change the tag to "door_fern_to_mine"
5. Go back into "area_mine" and select the door you placed in the previous section.
6. In the "Properties" tab, change its tag to "door_mine_to_fern"
7. The above just gives the doors easy ways for us to identify them when setting up the transitions. They're not absolutely necessary, but in general you'll probably want to re-name the tags of anything you'll be referring to later. That done, scroll the properties down to "Transition".
8. Next to "Linked To", type "door_fern_to_mine"
9. Change "Link Object Type" to "Transition to a door"
10. Set "Party Transition" to "True" - this ensures the entire party comes with the player when they exit the mines.

If you were to play the game right now, you'd find that the player could leave the mines, end up in town, and then have no way to get back to the mines! Clearly, we need to set up the other half of the transition.

1. Open the "area_fern" area and, as before, select the {Cave Entrance 1}.
2. In the "Properties" tab, scroll down to "Transition".
3. Enter "door_mine_to_fern" for "Linked to"
4. Make "Link Object Type" into "Transition to a door" and "Party Transition" to "True".

Congratulations! You can now move between the mines and the town.

3.3 Loading Screens

Now that you're traveling from section to section, you've probably noticed that the loading screens you see have nothing at all to do with what you're traveling to. That's because they're random by default. It's very easy to change them:

1. Select "area_fern", and then go to the "Properties" tab.
2. Under "General", you'll see an entry for "Load Screen". Clicking the arrow at the end of the entry gives you a list of every load screen. Clicking one will give you a (sort of) preview. Double-clicking will choose the given one. I went with "Rural 01"
3. Select "area_mine" and do the same thing for it. I liked "Mines 01"

Now when you enter an area, you'll see something appropriate to it!

Extra Credit

Create another indoor area; this time based on the Inn. Then hook it up to the outside world. Lock one of the Inn doors by setting its "Locked" property to True. You can make the key be anything in the game by specifying that object's tag in "Key Tag". If the player has that item in their inventory, they can then open the door!

4 Create The Cast

Introduction

Everything's laid out - the town of Fern is in place, the mines are full of monsters, and there might even be an inn for the players to stay at. You can play the entire adventure right now. Of course, the only thing missing is a reason for you to do so! Most modules are going to require some kind of non-fighting interaction, and that's where NPCs come in.

4.1 Townsfolk

There are two ways to create a custom creature. One is the way we created Gnashgab - simply place an ordinary creature in the world and modify its properties until it's the way we like it. This works well for one-off creatures, but if you end up wanting to place more of that kind of creature or have it spawned in by an encounter trigger, then you'll need to create a blueprint. There are two approaches to that, as well:

Falstadd

Falstadd is the mayor of the fine town, and father to the (sadly, doomed) Jacen.

1. Go to the "area_fern" area, and pull up the "Blueprints" tab.

2. With "Creatures" selected, drill down from "Humanoids" and right-click on "Human". Choose "Copy Blueprint" and "Module"

3. You'll see another 'Human' blueprint appear, this one in bold. The bold part indicates that it's part of this module, rather than being built-in. Right-click on this new human and choose "Properties (new window")

4. I suggest expanding the new properties window, otherwise it'll be hard to see what you're doing. We're going to create a very specific human here, but we'll need to see how we're affecting things in order to go any farther. Drag the 'preview' tab of this new window to the right , andyou'll see the new window split into two; properties on the left and a preview on the right. This lets us see what we're changing as we do it.

5. Modify his appearance as you see fit. I went with head #6 (because as mayor, he's a somewhat aged fellow), Hair 0, Appearance (facial hair) True, and tinted his hair white.

6. Falstadd's no longer a fighter. Go to the "Inventory" tab, and then click "Edit" at the bottom. You'll see that he's got some fancy stuff on him! Click each of the slots that has an item in it, and press the 'delete' key to get rid of it. Falstadd looks much more the part of a mayor once you hit 'ok' here.

7. Back to the Properties tab. Under "Basics", you'll see a field called "Comment". This has no effect on the game, it simply is a place for you to put anything you'd like to remind yourself. Two spots below that, you'll see "Resource Name". When spawning from a script, you use this name. Since we won't be doing that here, the default is fine. The important properties here for our purposes are "First Name", which we should change to "Falstadd" and "Tag", which we will change to the similar "c_falstadd". You can also change the Localized Description field to add some more personality.

8. You can control where a blueprint goes by changing the "Classification" entry. I changed the existing text to "Custom", which puts Falstadd in his own category and makes him easier for me to find (you can also find NPCs in your module by changing the active selection to "Module" instead of the default "All"). You can create subcategories by using pipe symbols - for instance, if I wanted Falstadd to be in a category called "Custom" and a subcategory called "Good Guys", I'd make his Classification "Custom|Good Guys".

9. We're done with Falstadd for now - close the new properties window.

10. Our creation should be selected already - if not, find the blueprint and select it. Place him in front of the inn.

Veran

Veran's the bad guy, though the players won't know that at first. He's also a convenient to demonstrate the other way of creating a blueprint:

1. In the "Blueprints" tab, with "Creatures" selected, right-click on "Humanoids". Select "Create Blueprint" and "Module".

2. A new blueprint called "creature1" will appear at the bottom of all the blueprints. As before, right-click it and select "Properties (new window)". You'll probably want to resize this window, too. As before, drag the 'preview' tab to the right to get it alongside the properties.

3. The first thing you'll notice is that Veran's a dwarf by default. Naturally, it'd be more appropriate if he were a sneaky, conniving Human. While we're at it, let's change the rest of his appearance. I decided on Head #1, Hair #5, and tinted most of him an evil red.

4. Go to the "Basics" tab. Each tab past "Store" essentially displays things which are already on the main "Properties" tab, but usually these tabs have more options or are easier to edit. When playing with creatures the PCs are meant to fight, it's best to use these tabs. Change the creature's name to "Veran" and its tag to "c_veran".

5. He's still a dwarf by default - change his subrace to "Human", though, and he'll shape up quickly.

6. While he's certainly uncouth, our Veran is no Barbarian. change his class to "Rogue" and give him three levels in it. Then click on "Reset from Package". Initially all the other statistics were assuming a level 1 barbarian - this will change them to be the package you've chosen for him (I picked "Rogue, Assassin")

7. Go to the "Statistics" tab and make Veran less pitiful. I went with (from the top down) 12,15,10,12,10,13. Press the "Reset for MAX" button by his hit points to calculate them.

8. The "Feats" tab is important as well. If there's anything you'd like Veran to have, this is the place to give it to him. The "Skills" tab next over is also useful, but given that there's nothing we'll really have him do that uses them, I left them all except for "Appraise" at 0.

9. Let's suit him up a bit. Under "Inventory", press "Edit". In the "Items" tab, drill down from "Weapons/Bladed/Light" and drag a "Dagger +1"

to the Right Hand. Make sure to make this droppable, so the player gets something for their trouble. Give him some armor, too (I went with "Leather Armor (rogue)").

10. Back in the "Properties" tab, click on "Faction ID" (under "Basics") and select "Hostile". For now, Veran's going to have no subtlety whatsoever about his intentions. Change his Sound Set to something more human (I chose "Human, Male, Guard, Gruff")

11. Because this creature was created from scratch and not copied from an existing creature, it doesn't have any scripts. This means, among other things, it won't fight correctly and it won't drop that shiny shiny dagger when it dies. You could go down to the "Scripts" section and manually enter the right scripts, but there's a faster way:

    (a) Click the "Import Properties" button at the top of the new window and select "Script Set".

    (b) Choose "b_StandardScripts.xml". This will fill in Veran with the default scripts that every creature gets, enabling him to fight and die and drop that dang dagger!

12. Finally, close the new window and place Veran in front of the inn, somewhere near Falstadd.

For a fun time, try this out in-game and watch the sparks fly! Remember to move the start location back to Fern if you don't want to have to walk through the mines.

5 Factions

Introduction

When we last left our hero, he was beating Veran senselessly (or, if your game played out like mine, getting stabbed repeatedly). However, having the ultimate bad guy just start killing everyone makes things a little too obvious! For now, he's going to have to stay in the background.

5.1 Editing Factions

Each creature in the game has a faction. The creature's faction determines how it reacts to all the other factions. In the most basic case, this means that a faction which is hostile to the player's faction (like, for instance, the Hostile faction is) will attack on sight. Ones that like the player enough to defend them (like, for instance, the Defender faction) will jump in on the player's side. Two enemy factions that hate each other will even fight amongst themselves!

Each module has five factions: Player, Hostile, Commoner, Merchant, and Defender. This can, of course, change! Each faction's feelings toward another is ranked from 0 (attack on sight) to 100 (help them out).

Collaborator Faction

We're going to make a new faction: The Collaborator. They're like merchants, but won't fight Hostile creatures.

1. From the "View" menu, choose "Factions". You'll see a new tab appear next to "area_fern", labeled "Faction". This displays every faction in your module, and its default relations with every other faction (including the player).

2. Press "Add Faction". A new faction, named "Faction", will appear. The horizontal axis here represents how they feel toward everyone, and the vertical represents how that faction feels about them. These default values are perfect, as it happens.

3. Click the newly created "Faction" faction. Properties should appear to the right. The first (and only) thing we'll want to do is change its name to "Collaborator". Close the Faction window.

4. Select Veran (make sure you're selecting the Veran in area_fern, and not his blueprint! Once you've placed a creature or object, changes to its blueprint no longer effect the placed one), scroll down to "Faction ID", and choose "Collaborator".

Miner Faction

Not all the miners in the game are dead - some have been forced into slavery by the goblins, and continue their work. If we just made them commoners, however, they'd be shot by the Goblin encounters we've already put in the mines. No, we need another faction:

1. From the "View" menu, choose "Factions". Press the "Add Faction" button once more.

2. In the same way as before, name the new faction "Miner". Also as before, leave the values as they are. The miners won't join in to help the players, but they're certainly not on the goblins' side.

5.2 Miners

Previously we created blueprints for one-off characters, which may seem a bit of overkill. In this case, we're making a blueprint for all the miners left in the mine, so we'll be re-using it.

1. Open the "area_mine" area.

2. On the "Blueprints" tab, with "Creatures" selected, drill down from "NPCs" and right-click "Commoner". Choose "Copy Blueprint" and "Module". As before, right-click on the newly created blueprint and pick "Properties (new window)".

3. Change our commoner's appearance. I picked head #2, Hair #16, and facial hair True for that scraggly look. I tinted the clothes dingier and used 'scale' to make him scrawnier, too.

4. Under "Basics", change the first name to "Miner" and the tag to "n_miner". Its faction should likewise be "Miner". I changed the Localized Description, too.

5. Because we're actually going to be referring to this later, change the "Resource Name" to "n_miner"

Miner Group Encounter

Here we're going to use the new blueprints. While we could simply place them ourselves, I'm going to take the opportunity to discuss making your own encounters:

1. Under "Blueprints" with "Encounters" selected, right click and select "Create Blueprint" and "Module".

2. Right click the newly created "encounter1" and select "Properties (new window)"

3. Change the localized name to "Miners" and the tag to "en_miners". Make sure the faction is "Hostile", though! Encounters will only spawn if their faction is one that would attack the player. Don't worry, the NPCs spawned by the encounter retain their player-friendly attitude.

4. Under "Behavior", select "Creatures". Click the "..." to the right of the entry.

5. Press "Add" in the dialog that appears. Select the new line and, under the "Creature" property, scroll down to "Miner" and select it.

6. Change the maximum and minimum number to "3" and "SingleSpawn" to True. Press "ok" to close the dialog.

7. Back in the new properties window, we're also going to change the maximum and minimum number of creatures to 3. The difference between this change and the previous one was the previous was on a per-creature basis. If you were spawning in, say, a mine overseer with the miners, you'd set it to max and min of 1, and keep the others at 3. Whereas this setting is encounter-wide.

8. Difficulty doesn't matter here, so leave it alone. Change "Spawn Once" to "True" because we don't need an endless stream of miners!

9. Set the "Player-Only" to true - we only want the miners to appear when the player triggers them, not monsters.

10. Close the new window. Now, paint the encounter trigger area; I suggest doing so right before the entrance to the 3x3 area, just south of where the goblin spawns are.

11. With the trigger area selected, paint three spawn points inside the 3x3 area.

This should be a good time to save and see what's changed! If you want to check out the miners, put the start location back in the mines or walk up to them. You might want to temporarily turn the goblin encounters off (by setting their 'active' property to 'False') so you don't have to fight your way in just to see if it's working.

Extra Credit

Under "Blueprints", select "Waypoints". Drill down from "Empty" and you'll see a number of waypoints. The "Map Note" waypoint can be used to put a note on the player's map. Put a few around Fern and the mines, editing their "Map Note Text" properties to reflect what the player will see.

If you place an ordinary Waypoint and look at its comment, you'll see it's got quite a lot of text in it! The text is describing how to make your NPCs wander around without your interference. Follow the instructions there to make your miners wander once spawned in. In case you can't read the tooltip, the steps are:

1. Create the creature you intend to have walk over these waypoints. We've already done that part, above.

2. Make sure "WalkWayPoints()" is in the body of the OnSpawn script for the creature. We made our creatures from Commoners, and Commoners have this, so this also requires no work on our part.

3. Place a series of waypoints along the route you wish the character to walk.

4. Select all of the newly created waypoints and right click. Choose the "Create Waypoint Set" option.

5. You'll be prompted to enter the name of the waypoint set. For this to work, the name has to be WP_tag, where tag is the tag of the creatures to wander the waypoints. In our case, enter "WP_n_miner".

6 Journals and Conversations

Introduction

Way back in tutorial 4, I promised we'd have non-fighting interaction. So far that's been limited to watching NPCs either stand around or wander aimlessly. Also missing is our reason for going on this quest! This chapter should start fleshing some of that out.

6.1 Quests and Journals

The Journal is one of the most important parts of any module. While you could manage small quests without it, the player would be lost the moment they reloaded a save game from a week ago and forgot what they were supposed to be doing (not that that's ever happened to me!). So we're going to create some journal entries for our module:

1. From the "View" module, select "Journal" and then "Module". The Journal editor will appear next to the open areas.
2. Each quest in the journal is called a "Category" by the tool. Initially it'll be entirely blank - click the "Add Category" button to add a line.
3. Select the line, and properties will appear below. Change the name to something more suitable (Like "Falstadd's Quest"), its tag to "jt_falstadd", the priority to "Highest", and XP to 2000.
4. There are going to be three stages to the quest, and we're going to add them all at once: Select Falstadd's Quest and press the "Add Entry" button, then repeat this twice more.
5. Select the first entry and type a description of the first stage of the quest. You can either try to type in the small space the entry allows up top, or the larger blank area in the lower-right (to the right of the properties). I entered: "Fern's mayor, Falstadd, has asked you to find his son Jacen in the mines".
6. Change the second entry to something like "You have recovered Jacen's ring from his body. Falstadd must be informed".
7. Make the final entry something appropriately sad, such as "Falstadd now knows the unfortunate fate of his son." Also change the "Endpoint?" property of this entry to "True", to reflect that it's the last journal entry for this quest.

6.2 Conversation

Conversations are (for the most part) the NPC saying something to you, your reply, and their reaction. It's also a great way to drive journal entries:

1. The "Areas" window on the left is actually a tabbed interface. By default it's on "Areas" - change it to "Conversations", which is the tab directly to the right of the selected one.
2. Right-click in this new empty window and choose "Add". (Be warned - for some reason this takes an inexplicably long time on my machine. Your mileage may vary).

3. The Conversation editor will appear alongside the open areas. You can also see the Properties of the conversation if the "Properties" tab is open and the new conversation selected. Rename the conversation by clicking on it in the Conversations window and selecting "Rename". I picked something memorable, like "Falstadd".

4. The conversation starts at the Root. Select it and press the "Add" button.

5. A dialog will appear. This is what the NPC says when you begin chat. This node will be the one for after the PC has been formally greeted, so change the text to something like "Ah, <FirstName>, it is you - what have you learned of the mine? The "<FirstName>" part is a token - the game sees this and replaces it with the player character's first name. It's great if you want to refer to, for example, the character's gender but don't know it ahead of time. A full list of tokens is available in the drop-down box of every major text editing area (next to the 'insert' button, which you press to put the selected token into the text).

6. With the newly-created node selected, press "add" again. This will be the player's reply. First we'll do the case where the player's discovered everything, so enter something like: "The mine was overrun with goblins who forced the miners to work for them. I did everything I could, but Jacen was among the lost. I have only his ring for you."

7. Now we'll make a reply to that. Hit "Add" with that selected, and type: "I thank you for freeing the miners, but... my son! It is too high a price to pay."

8. That bit over, select the first node ("Ah, <FirstName>, it is you") and press "Add" for another, less depressing option. Enter "Nothing yet." as text.

9. Falstadd's a man of ideas. Select that node and add to it. for text, have Falstadd give some advice like "Talk to Veran - he's been looking for people to go into the mine. He's our local shopkeep as well, so he should be able to outfit you."

10. All that conversation was assuming we'd already been given the briefing, which we haven't typed yet. This is the step where we do that. Select the root node and add to it, with something along the lines of "Please, kind <class>, you must help me!"

11. Now to railroad the PC into giving a response. Add something like "You can call me <FirstName>. What's going on?"

12. The reply to this should be suitably dramatic: "Our miners have disappeared - among them, my son, Jacen! Nobody dares go into the mine now."

13. Add a node to this last one, but delete the text, leaving an empty entry behind. If there are no replies to this line, the player will see it as "End Dialog", and if there is more afterward, it will be "Continue".

14. Finally, we want to tell the player about Veran and tell them to get outfitted, but we'd rather not type all of that again! Thankfully, there's a way to link to other nodes. Click the "Talk to Veran" node and press the "Set Link Destination" button. Then select our last line (which should currently say "End Dialog") and press "Insert Link". A grey line of dialog should appear - this indicates it's a link to the earlier node. Conversation will pick up there, and any changes you make to that branch of conversation will be reflected in this one as well.

15. Go to "area_fern" and select Falstadd - make sure you're selecting the one in area_fern, not the blueprint. Go to the "Properties" tab and, under "Basics", change his conversation to "falstadd".

Save the game, move the start location back if you need to, and then go chat up the mayor!

6.3 Using Scripts

Well there you go. Now you can talk to Falstadd, learn what's going on, and complete the quest without ever leaving the conversation! What? You should have to actually do the work to make the conversation happen? Picky picky....

1. Open up the Falstadd conversation again. (It's in the "Conversations" tab next to "Areas")

2. Click the first node ("Ah, <FirstName>, it is you"). Change the tab at the bottom from "Animations" to "Conditions". These are the conditions that have to be satisfied in order for the line of conversation to appear. Press "Add"

3. A line will appear in the conditions - a greyed out "And", a "Not" checkmark, and a drop-down menu under "Script". It's here that we indicate what script must be satisfied in order for this condition to be true. We're going to use "gc_local_int".

4. Below the line you just entered, a script will appear. This is the script you've just chosen. Even if you don't understand the language, built-in scripts have comments that are very helpful (they begin with //, or are enclosed by /* ... */, and are colored green). You may want to re-size the script window so you can read it, because we're not entirely done. We picked a script, but the script still needs to know what we want from it. Information going into a script is called its "Parameters", and they're documented by the comments. In order to fill out these parameters, we need to press the "Refresh" button. A number of new entries will appear to its right: one for each parameter.

5. Enter "quest_begun" (without the quotes) for the first parameter. The purpose of this script is to see if the player's talked to Falstadd before. When they do, we're going to set this variable to '1' - all we have to decide is is the name of the variable we'll be checking. You can put as many variables on as many objects as you like - they're very useful for keeping track of where you are in a quest or what the player's already said.

6. For the second parameter, enter "=1" (also without quotes). That'll make sure that the "quest_begun" variable is compared with "1", and that this check will only be true if the quest has been set up. Leave the last entry blank - it will default to the current speaker, who should be Falstadd.

7. Now we'll actually set that variable - otherwise the chat will never appear! Click on the sixth node ("Please, kind <class>") and change the bottom tab to "Actions". Press "Add" there, too.

8. For our script, we're going to choose "ga_local_int". The "g" part of its prefix indicates that it's a global script, and the "a" part means it's an action - the rest of it is named just like the conditional script we used before. You'll see that pattern quite a bit. Press "Refresh" to make the parameters appear.

9. Without quotes, enter "quest_begun" in the first entry.

10. Enter "1" in the second - again without quotes. See the comments for details about everything you can do with this parameter; the simplest thing is just to set it, which we're doing here. Again leave the last one blank, as it will default to Falstadd.

11. We're almost to a usable conversation! Let's make it so that the players can't just go around telling people that Jacen's dead without some sort of proof. Select the second node ("Yes, the mine was overrun"), switch back over to Conditions, and press "Add" one more time.

12. This time the function we're looking for is "gc_check_item". Select it and press "Refresh".

13. The "gc_check_item" function checks to see if the player (or, optionally, anyone in the party) has an item with the given tag. For the first parameter, enter "item_ringJacen". Of course, this item doesn't exist yet, but it will.

14. Put "1" for the next parameter. This will check the entire party to see if they have it, in case you offloaded it to another character.

15. The ring's a quest item, so now that the quest is complete, we're probably going to want to give the ring back. Click the third node ("I thank you..."), go to 'Actions', and press "Add".

16. The function we want here is "ga_take_item"[3] - select it and press "Refresh".

17. The first parameter is the item to take - again, we want "item_ringJacen". The second is how many to take. We're going to set this to -1, which is shorthand for "All of this item that the player has". There should only be one, but it won't hurt. Leave the last parameter alone.

Save the game and give it a run-through. You should be able to have a normal conversation with Falstadd, and he'll even remember who you are! Of course, you can't check the part that requires you to have the ring yet. Also, there's something else missing.

6.4 Conversations and the Journal

We did go through an awful lot of trouble to write up journal entries earlier. We probably shouldn't let it go to waste.

1. Make sure Falstadd's conversation is open. Select the fifth node ("Talk to Veran...").

2. Select the "Node" tab (between "Actions" and "Animations"). Something that looks suspiciously like properties appears below the conversation.

3. Select the "Quest" line, then click on the drop-down arrow. You'll see tags for all the journal categories, and when you select a category you'll see the entries in it. Select "jt_falstadd" and the first entry. (double click on "1 - Fern's mayor, Falstadd...."; the entry next to Quest should look like "jt_falstadd (1)" if you've done it correctly).

4. Likewise, we want to end the quest once we inform Falstadd of what's going on. Click the third node ("I thank you...") and change its Quest entry to "jt_falstadd (21)" (That one's the "Falstadd now knows..." entry).

Now you should get journal updates with the conversation.

Veran's Conversation

After directing the player, repeatedly, to talk to Veran, we should probably give him something to say. At this point I'm going to assume you know how to add nodes to a conversation, so I'll mostly be telling you where to put what:

[3] You may be wondering how to find the function that you need. Simply put: Guessing. While I have experience with NWN1 scripting, I haven't done any in NWN2 - I find the scripts I need by looking over all of them, choosing likely candidates, and reading the comments. In this case, when I saw a "ga_give_item", it seemed a sure shot, but I still had to look at the comments. Upon doing so, I discovered that ga_give_item is the function that gives items to the player! That established, however, it seemed pretty obvious that there would be a "ga_take_item", and after some quick searching I found it.

1. Add another conversation (right-click the conversations window and choose "Add"). Rename this one "Veran".

2. Make a suitable greeting for the first node. Something like "Hello, traveler! I am Veran, owner of Veran's Splendid Sundries. How may I help you?"

3. We're going to make three replies here:

   (a) One, wherein we ask for details: "Hello Veran, I am <FirstName>. Falstadd said you needed people for the mines?"

   (b) Another, wherein we ask to see the goods: "I could use some sundries, show me what you have."

   (c) And finally, one for people who clicked on Veran by accident: "Nothing right now, thank you."

4. Now let's develop the plot a bit. Add a node to the second one ("Hello Veran..."), and explain a bit more about the mines: "Alas, our poor mayor isn't what he used to be. His own son, Jacen, is trapped in the mines with the rest of those poor men. If you intend to do something about it, may I caution, you'll need to be outfitted."

5. We're going to give the player two options at this point:

   (a) Click the fourth node ("I could use some sundries...") and then press "Set Link Destination". Then select the third node ("Alas, our poor mayor...") and press "Insert Link".

   (b) Using this same technique, add a link to the last node ("Nothing right now, thank you") so the player can turn down Veran's offer.

6. Select Veran (again, make sure you're selecting the one in area_fern, not the blueprint) and change his conversation to... wait for it... "veran".

You should be able to have full conversations with both Veran and Falstadd now!

Extra Credit

Falstadd's conversation ends rather abruptly - this is because when the game gets to a node with no player response, it ends the conversation right there. If you happen to be looking at the text area in the lower left when this happens, you'll see what Falstadd said, but it's very easy to miss. Go back to his conversation, and add blank entries after the "I thank you" and "Talk to Veran" nodes. These will be rendered both in-game and in the toolkit as "[END DIALOG]".

Veran has a much fancier conversation than he needs. Open it in the conversation editor, then click the "Properties" tab. You'll see a number of properties. The one that controls whether you get fancy cutscene style dialog or NWN1

style dialog is the "Neverwinter Nights 1-style Dialogue" property. Change it to "True".

If you made an inn in the last Extra Credit, now's the time to make an innkeeper. Put him somewhere in the inn. Have a script action in his conversation award the player 1000XP (which should level up a 1st level character) - just make sure to use local ints to guard against the player doing it again!.

7 Items and Inventory

Introduction

It seems like every time I keep getting us closer to the end of this thing, I think of some other wacky thing we need to learn. For instance, the beginning and end of the quest are done, but they depend on an item - which we haven't made yet - to proceed! Also, Veran - the cheat - keeps teasing us with the promise of a store but not doing anything about it. We'll take these one at a time:

7.1 Creating an Item

Specifically, we're creating this "Jacen's ring" that you've heard so much about.

1. We begin by creating a blueprint. Select the "Blueprints" tab, and make sure "Items" is highlighted within it. Then drill down from "Miscellaneous" to "Jewelry" to "Rings".

2. Because this ring's not going to have any special abilities, select the "Silver Ring". Right-click it, and select "Copy Blueprint", and "Module". This should all seem pretty familiar by now.

3. Right-click the new "Silver Ring" and select "Properties (new window)". This time you don't need to separate the Preview window out, as the ring looks like a loot bag when dropped.

4. Click on "Icon" and, using the arrow-button, select an appropriate icon. This is how the item will look in the player's inventory. I went with "it_ring_copclassring"

5. Change the "resource name" to something more sane. I liked "it_copperRing".

6. Make the "Localized Name" into "Jacen's Ring", and its tag "item_ringJacen".

7. Come up with something suitable for "Localized Description". Since the ring's already been identified, you don't need to provide a separate one for that instance.

8. Under "Behavior", change "Plot" to true - this makes it so the ring can't be destroyed by the player.

9. Because we're going to need this in about six steps, change the Classification to something you'll remember which will make the ring easier to find. I just used "Plot".

10. You're done with the item - the only thing to do now is place it on Jacen! Close the properties window, open the "area_mine" area, and move over to where Jacen rests in peace.

11. Select him, then click the "Properties" tab. Since we started this as just a corpse, we're going to require a bit of modification to make Jacen semi-presentable.

12. First of all, change his "Localized Name" to "Jacen". Then add a Localized Description that seems appropriate. ("You can tell instantly by the family resemblance that this is none other than Jacen, Falstadd's Son" - I love making descriptions that contradict what the player sees, don't you?)

13. Change "Static" to "False" - this makes it possible for the player, scripts, etc, to access this object. The Static flag is what separates interesting items from decoration.

14. Because we want the player to interact with the corpse (i.e., robbing it), make "Usable", "Plot", and "Has Inventory" also True.

15. Jacen is set up to be as interactive as a dead body is likely to be. Go to his "Inventory" tab and press the "Edit" button. Find the ring (If you put it where I did, it's under "Plot" -> "Jacen's Ring") and drag it into his inventory.

There we go, an item! As nice as it is, though, it's not very beneficial for the player. Let's create an item that is!

7.1.1 Creating a Magical Item

We're going to make an amulet specially created to fight against the goblins that are prevalent in the mines.

1. On the "Blueprints" tab, with "Items" selected, drill down from "Miscellaneous" to "Jewelry" to "Amulets" and select "Silver Necklace".

2. Right-click on the necklace and select "Copy Blueprint" and "Module". Then right click on the new silver necklace and choose "Properties (new window)".

3. Change the icon to your liking. I thought "it_nk_shaman01" looked appropriate for dealing with goblins.

4. Change the Resource Name to something we'll remember later. I went with "it_amulet_goblin"

5. Make the Localized name something cool-sounding, like "Goblinsbane Amulet"
6. The tag should likewise be memorable. I liked "item_goblinsbane"
7. Make both localized descriptions something descriptive. A fun unidentified description was: "This crude-looking amulet must serve some purpose.", and I thought a suitable backstory for an identified description would be: "Ever since the goblins invaded Fern, amulets of this kind have been popping up all over the place. Some are even genuine."
8. Now for the interesting part. Select "Item Properties", then press the "..." button.
9. Now we're talking! The kind of bonus we want here is "AC Bonus vs. Racial Group". Expand that, and select "goblinoid".
10. Click "Add Property".
11. The default is a +1 bonus. That's fine for our purposes, but you could easily change it by selecting the line in the "Property" window and changing the value in the lower-right. Press "OK" then close the new window.

7.2 Managing Merchants

I don't know about you, but I've had it up to here with Veran and his holding out on us! It's time to create a store!

1. Open "area_fern", and center the camera near Veran.
2. In the "Blueprints" tab, with "Stores" selected, drill down from "Empty" and choose "Sundry Store - Low".
3. A waypoint will be attached to your cursor. This waypoint represents the store itself - put it somewhere near Veran.
4. Select the newly placed store, then click on the "Properties" tab.
5. Change the Localized name to something a bit better, like "Veran's Splendid Sundries".
6. Likewise, change the tag to reflect this: "store_veran"
7. The store starts out with "0" for most of its settings, which isn't going to work well at all. Change them as follows:
   (a) Identify Price to something else - 100 is the standard.
   (b) Price Percentage for selling items to the player - This is the percentage of actual worth the store will charge you. Veran's a jerk, but set it to 100 anyway, otherwise we won't be able to fund this expedition!

(c) Price Percentage for buying items from the player - the percentage of actual worth the store will give you. Set it to 30. That's more like our jerk!

(d) Set the maximum buy price to -1. This means he'll buy anything. Give him 10000 gold to buy things with. It's unlikely he'll have to use all of it, but better safe than sorry!

8. You can look at the contents of the store by selecting the "Store" tab (between "inventory" and "basics"). Select this tab and press "Edit".

9. Veran's shop doesn't have a whole lot - specifically, it's missing weapons and armor. Select the "Armor" tab, then drill down from "Armor" and drag some suitable armor to the left. I added "Light Shield", "Chain Shirt", "Leather Armor", "Padded Armor", and "Chainmail" (Be sure to be setting the "Infinite" checkbox as you add these, the player may want more than one!)

10. On the "Weapons" tab, I added "Throwing Axe", "Shortbow", "Light Crossbow", "Morningstar", "Mace", "Short Sword", "Longsword", "Dagger", "Handaxe", "Bolt", and "Arrow"

11. Here we can add our created item! Drill down from "Miscellaneous", "Jewelry", and "Amulets" to locate the Goblinsbane amulet we created - it should be easy to find, it being bold. Select it and choose "Add Item". I added it under the "Rings" tab, even though it's not technically a ring.

12. Finally, on the "Potions" tab, I added "Potion of Cure Light Wounds" (Under "Miscellaneous" and "Potions"). Press "OK" when you're done.

7.2.1 Opening a Store

1. We've set up the store, but if you were to look at this area in-game, you wouldn't see anything. That's because our store waypoint is only there for our editing convenience - it doesn't appear in-game. We have to make it appear by using a script call in Veran's conversation. Open his conversation now.

2. Select the sixth node ("I could use some sundries...") and select the "Actions" tab. Press "Add".

3. The script we're looking for is the aptly-named "ga_open_store". Select it and press "Refresh"

4. Set the first parameter to the tag of our store ("store_veran")

Now you should be able to shop at friendly Veran's!

Extra Credit

Veran's being awful brazen, just standing there with his knife drawn. Move it from his hand to his inventory.

Speaking of Veran, we know because I've been repeating it that he's the evil mastermind behind the mine invasion. But we haven't given the player a way to know this yet! Make an item patterned after a book (Miscellaneous -> Books) called "The Plan" which details Veran's scheme. Place it in Gnashgab's inventory (be sure to set its "Droppable" flag!)

8 Atmosphere

Introduction

Noticed anything in particular about our module that marks it as different than others (besides the fact that you still can't complete it, and I'm putting up yet another obstacle)? It's awful... quiet! And the mines, though functional, look fairly drab. Luckily, there's quite a few ways to change how things look and sound!

8.1 Area Lighting

The easiest way to change how an area looks is to alter the entire area at once!

1. Open "area_fern". Make sure that the area is selected in the areas list on the left, then click the "Properties" tab.

2. Under "Appearance", expand "Day/Night Cycle Stages"

3. 7 entries will appear, each representing a different time of day. The engine cycles through these depending on when it is, in-game. We're going to gloom-up the daytime a bit. Expand the "[1]" (Daytime) node, and change SkyHorizon and SkyZenith colors to darker versions. To see how your changes effect things, go to the "Day/Night" menu on top, and choose "Daytime".

4. Expand "GroundLight" and change its Diffuse and Specular to darker tones.

5. We can change our Fog here, as well. Expand the "Fog" option, and set "FogStart" to 20 to have it begin much closer to the player. Make it a whiter color, to more obscure the distance. In order to see this in the toolset, you'll need to have the "Fog" option selected (Between "Use Area Far Plane", and "Sky")

6. Expand "SunMoon" and change its diffuse and specular colors to darker colors. Here's where you'll see the most change in how the town looks.

While you can make it look stormy, you can't actually make it rain like in NWN1, at least not without using a toolset plugin. For the most part, though, a foreboding atmosphere will do wonders without having to see a drop.

8.2 Tile Lighting

There's more fine-grained control to be had over lighting as well!

Faking it

1. Open "area_mine".

2. With "area_mine" selected in the areas dialogue, go to the Properties tab. Unlike outdoor areas, indoor areas don't have a day/night cycle. In editing the stages, then, the only one that matters is the last ("Default"). I changed the GroundLight to be completely black, and made the SunMoon diffuse and specular much more grey. For that finishing touch, make the fog a greyish haze that starts at '5' or so. The result is a very depressing-looking mine![4]

3. As thrilling as this is, it's nothing we haven't seen before. I did promise fine-grained control, didn't I? Make sure that "Tiles" (between "Paint Terrain" and "Set Start Location") is selected and go over to Gnashgab. Select the tile he's on.

4. In the "Properties" tab, you'll see a number of properties - they belong to the tile itself! The first two are of greater importance here - TintFloor and TintWall. That's right! Just like placeables with "TINT" in their name, any tile can also be tinted! If at first this doesn't seem to do anything, however...

5. That's because it doesn't. While every tile has the "Tint" properties, not every tile responds to them. If you place, say, a "Standard Castle", "Floor_4_Corners" tile off in an unused space, you can immediately see the effects of the tints as you select them. This not only allows you to simulate tile lighting, but also lets you change the entire theme of a building. Sadly, mines only have the one theme. (If you do this, remember to remove the tile when you're done experimenting; you don't need the player wondering why there's a castle adjacent to our cave!)

The Real Way

1. Okay, okay, we'll do real lighting! Unlike NWN1, it's not tile-based, but rather point-based. First we need to set it up. From the "Blueprints" tile,

[4] An easy way to see how all this looks in-game without moving the start location around every time is by right-clicking on any of the waypoints and choosing "Run Module at Waypoint". You'll be presented with some options for launching the module that you don't ordinarily have as well!

with “Placeables” selected, drill down from “02 - MANMADE PROPS”. Select “Brazier” and put two of them near the endpoints of the altar.

2. Now change the Blueprints category to “Placed Effects”. These are special particle effects - like fire - which can really make a scene more dynamic. Place two of them (I chose “Bonfire”), one near each end, then put them atop the brazier.

3. I’m sure you’ve noticed that, while pretty, these fires don’t actually have any lighting effects. You may have also noticed a Blueprints category called “Lights”. Select that now. Unfortunately, there’s only one light in there at the moment, a plain white light. Clearly, we want something else. Right click on the white light and choose “Copy Blueprint” and “Module”.

4. Yep, even lights have blueprints. Select your newly created light and right-click it. Choose “Properties (new window)”

5. The first thing we want to change about the light is its color. Expand the “Color” node and make each entry in it a shade of orange. The next item, “Lerp Target Color”, you can ignore. It’s possible to blend between two different colors in a light source, and that setting would determine the other color, but we don’t need that ability right now.

6. Change the “Resource Name” to something fitting, like “lt_fire” and the localized name to something descriptive like “Firelight”. Make the tag reflect the new status as well: “lt_fire” is good here to.

7. If you were to click on our new light now and drag it around the area, you’d see it’s the right color and all, but it’s a bit... static. Thankfully, we can change that. Change “Flicker” to “True”.

8. It still doesn’t look like firelight. We want to change the “Flicker Type” to “Jumpy” to have it rapidly change. Likewise, it’s not changing rapidly enough. Make “Flicker Rate” something like 8.

9. If you look at the light now, it’s gone insane! Clearly, we need a more subtle touch. Change “Flicker Variance” to 0.1

10. It’s still making the shadows jump around crazily. In fact, the most expensive thing, computationally, that the light requires is shadow calculation. Using two in one room like we’re doing here may overtax some computers. Let’s save the module-playing populace (and ourselves!) a headache, scroll back up to “Appearance”, and turn “Casts shadow?” to false.

11. We’ve perfected our new lights. Place two of them in the scene, next to the Braziers, and then move them atop it where the fire is.

12. While turning off the shadows is a good solution, and it works, but what if we want shadows? Suppose we just don’t want the ugly shadows cast by the Braziers? There’s another solution:

(a) Turn the "Casts Shadow" property of the lights back on. (You may have to select them via the "Area Contents" tab, I'd placed mine close to the bonfire).

(b) Select the braziers (either by shift-clicking, or doing this to both of them) and, under their "Properties" tab, press the drop-down menu next to "Casts shadow?"

(c) Change this to "Don't cast shadows"

(d) This is neat, but Jacen and the altar are somewhat hard to see now, aren't they? Select them both and change their "Receives Shadows" property to "Don't receive shadows".

8.3 Ambient Sounds

We still haven't addressed the fact that our module is virtually silent. We'll start with making Fern a little more vibrant:

1. Open up "area_fern".
2. With "area_fern" selected, go to the "Properties" tab.
3. Under "Ambient Sound (daytime)", choose something appropriate. I went with "Forest Day 1" for the daytime, and turned its volume up to 25.[5]
4. "Forest Night 1" seemed like a good title for the night.
5. The "Environmental Audio Effects" change how sound effects are processed. A sword clash echoes differently outdoors than it does in a tight cavern. I changed this to "Outdoors, rolling plains".
6. For "Battle Music", I picked "Forest Combat 1", Daytime Music, "Forest Day 1", and Nighttime music "Forest Night".

If you start the game right now, you should be able to hear the music. Your next step is to do something similar for the Fernesk mine. Keep in mind it's no longer a happy, there's-only-a-slight-chance-of-permanent-injury-here sort of place, it's an oh-god-goblins-they-have-my-arm kind of mine. I went with "Crypt 1" for the day/night music, "Dungeon Combat 2" for that classical battle music, and of course "Mine Small" for ambient sounds.

[5] The toolset, sadly, cannot preview the music. One option is to make the change in music and start up the module (changing the Module's start hour to preview the night music) repeatedly. However, if you're feeling adventurous, there are plugins to fix the problem: see appendix B. That will enable you to preview any non-ambient sound you'd like. You can find the ambient sounds, in .WAV format, in the "Ambient" directory of your NWN2 installation.

8.4 Sound Objects

Areas had global lighting settings, and to make things more individual they also had the ability to tint tiles and add point lights. There's sound settings for areas, so you might surmise there'd be more fine-grained control over the audio of an area... and you'd be right!

1. If you're not already there, open "area_mine".
2. Go to the "Blueprints" tab, and select "Sounds". You'll see a number of categories. We want a suitably interesting sound for when the player enters the cave.
3. From "Background", choose "Eerie Cave 02". This is going to get very annoying, very quickly, so place it on the ground and select it, then go to the "Properties" tab.
4. Switch "Looping?" to "False" and "Continuous?" to false. That'll stop it from repeating over and over in the toolset! (And in the game as well). This sound is a non-positional sound, meaning that it won't sound like it's coming from any place in particular
5. Zoom out in the toolkit - you'll see a large purplish sphere - this is the range in which the PC will hear the sound. If you want a sound to be more localized, you can modify its "Maximum Distance" property.
6. Next, from the "Environmental" and "Water" sections, choose "Water Drips". Place it in the center of the hallway. If you look at its properties, you'll see that this is a Random sound - that is, it'll always sound like it's coming from somewhere random - but on average it'll sound like where you placed it. It's the "Random Position?" property that controls this behavior - the "Random?" property merely picks at random from one of the sounds in the set each time. You can change the variation by changing the Random Range (x) and Random Range (y) properties; I made (y) much larger than (x), so that it would always sound like it was coming from somewhere in the hallway area.
7. If you zoom in on the sound, you'll notice that it's just constantly playing over and over again. We can fix that by changing its "Looping" to false, but that stops it altogether! No worries: Change looping to false, and Continuous to true! The difference is that looping constantly plays a sound. Continuous plays it once, then waits an amount of time (the Interval time, which is right below the Continuous setting and is in $1/1000^{\text{th}}$seconds) before doing it again. Moreover, this time can be modified by the Interval Variation. I set both to 5000.
8. Finally, from "Dungeon" and "Mines", choose "Miner Pickings". Place it in the middle of the 3x3 area. This is the simplest kind of sound - it will always sound like it's coming from where you place it. I put it near one of the waypoints that the miners will be walking near.

9. This has the same problem - it keeps going and is monotonous. Again, change Looping to False and Continuous to true. Modify the intervals to be about 1000. The pick noises are still near-constant, but there's now quite a bit of variety to them. Feel free to mess with the intervals if you'd like to make it different.

Extra Credit

More sounds! Put bat sounds in the mine! Give the inn some jovial music! Notice how all the fires we've placed so far are silent? Not anymore!

9 Scripts

Introduction

The scripting language that NWN2 uses is essentially the "C" programming language (any purists reading that sentence and about to compose me an angry e-mail about the many differences, I'd like you to take note of my use of the "Essentially" qualifier). If you don't know the language, it's not hard to pick up enough to get around the toolset. While C is usually used for making operating system kernels and writing games in the first place, its use in NWN2 is mostly just to make special things happen. Earlier in section 6.3, you used these scripts to determine if a node was supposed to be visible and to set variables to indicate whether you'd talked to someone before. That was using the built-in scripts, and you can learn quite a bit just from looking at them as most are fairly well commented. This section, however, is for rolling your own!

9.1 OnAcquireItem

Remember how I've been stringing you along on how to finish building this whole Jacen quest for several chapters now? Guess what? We're actually going to do it! The one bit we're missing is having the player able to pick up Jacen's ring. There are two ways to go about scripting this, and I'm going to cover one, then the other. I'd recommend the latter of the two, but it won't hurt to do them both to see how it's done and gain some insight into scripting:

Method 1: The Brute Force Approach

1. In the "Scripts" tab - this is another tab which shares its location with Areas and Conversations - right click and select "Add".

2. From the "View" menu, choose "Module Properties". Your properties tab now displays the properties of the module itself. This is where you'd set the name of the module and the description the player sees when selecting it. More immediately important, it also houses the "On Acquire Item" script. This script will fire whenever the player gets an item - since we

want to intercept that event in order to update the journal, the obvious place to do so is here.

3. Under "Scripts", there should be an entry next to "On Aquire Item Script". Copy that name to the clipboard, write it down, or otherwise remember it - we'll need it in a moment

4. Back to your new script. Type in the following (Though you don't have to type in the comments - they're there to tell you what each line is doing):

```
void main() {
  object oPC; // The PC that's getting the item
  object oItem; // The item the PC is getting
  // Asks the module what the last item was
  oItem = GetModuleItemAcquired();
  // Make sure the item is a valid one
  if(GetIsObjectValid(oItem)) {
    // See if the item we're getting is
    // the one we're interested in.
    if(GetTag(oItem) == "item_ringJacen") {
      // Figure out who has it
      oPC = GetItemPossessor(oItem);
      // Update the journal
      AddJournalQuestEntry("jt_falstadd",11,oPC);
    }
  }
```

5. This isn't quite done yet! This script works for the very specific case we have here, but there's other things the old On Acquire Item script did that we're not doing. So we need to add in a call to pass control over to it:

```
  ExecuteScript("x2_mod_def_aqu", OBJECT_SELF);
}
```

6. As the name implies, that command executes the script you give it. The first parameter is the script to execute, which is the one we took note of in step 3. Finally, the last parameter is the object that's running the script - we can use ExecuteScript to run scripts on specific objects (for instance, a door-closing script might be written to run from the door itself). OBJECT_SELF is a way to refer to the current object, no matter what that is.

7. Right-click on the script name in the Scripts window and rename it. I went with "tutorial_onacquire"

8. Go back to the module properties ("View" -> "Module Properties"), select the "On Acquire Item Script" property, and change it to the script you just made.

As an aside, the code I've given in this section is almost verbatim from the NWN1 tutorial. A great many things have changed since NWN1, but a lot of their scripts work, and most of the techniques are still in place. Even for those that aren't, the old ways are still supported.

Method 2: The Modern Approach

The first method works. Why wouldn't you do it that way? Well first of all, you're usurping the module's built-in item handling. While we call the handler after we do our checking, there still may be unforeseen consequences of doing this. Secondly, it's a bit of a maintenance nightmare, isn't it? Every time you have an item that you want to do something when the player gets it, you're going to have to go back and edit this file. It's going to get very large quickly, and be very difficult to debug if something goes wrong.

If by any chance you looked at the x2_mod_def_aqu script, you may have noticed something in the comments:

```
// * Generic Item Script Execution Code
// * If MODULE_SWITCH_EXECUTE_TAGBASED_SCRIPTS is set to TRUE on the module,
// * it will execute a script that has the same name as the item's tag
// * inside this script you can manage scripts for all events by checking against
// * GetUserDefinedItemEventNumber(). See x2_it_example.nss
```

The important line in those comments is the third: By default, whenever the player gets an item, the game will execute a script with that item's tag! This solves the two problems the old way presents: We're using the system rather than replacing it, and adding a script for a new item is as simple as, well, adding a script for a new item!

If you did the first method and now want to try this next one, it's relatively simple to revert:

1. Go to "View" and "Module Properties"
2. Under "Scripts", find "On Acquire Item Script" and change it back to "x2_mod_def_aqu". Reverted!

If you're ready, we're going to make a new script:

1. In the "Scripts" tab (to the right of "Conversations"), right-click and choose "Add".
2. This script is much simpler!

```
#include "ginc_item_script"
void main() {
  object oPC      = GetModuleItemAcquiredBy();
  object oItem    = GetModuleItemAcquired();
  int iStackSize  = GetModuleItemAcquiredStackSize();
```

```
  object oFrom    = GetModuleItemAcquiredFrom();
  // Once marked complete, never run this script again.
  if (IsItemMarkedAsDone(oItem, SCRIPT_MODULE_ON_ACQUIRE_ITEM))
    return;
  //      Don't run this script unless it's acquired by a player or party member
  // (This event runs at the start of the game)
  if (!IsItemAcquiredByPartyMember())
    return;

  // Update the journal
  AddJournalQuestEntry("jt_falstadd",11,oPC);

  // Permanently mark item as complete so script will never run again (even if the
  MarkItemAsDone(oItem, SCRIPT_MODULE_ON_ACQUIRE_ITEM);
}
```

3. Of course, the difference is that this script will only be called when the player gets the item that shares its name. Speaking of which, we should re-name our script appropriately. Right-click on the newly created script's name in the Scripts window, and choose "Rename". The name you'll want to use is "i_item_ringJacen_aq".

Even this way of making the script is somewhat laborious. If only there was a template for this sort of thing! Wait, there is! You can right-click in the Scripts area and choose "Add From Template". You'll be given a number of templates to choose from, among which is "Item Acquire". We've already made our script so we don't need it now, but keep it in mind should you need to create another. The code it makes may look strangely familiar!

9.2 Custom Conversation Scripts

Many things have script properties that you can override as we did above. But that isn't all scripts can do! As you saw before, they can be used in conversation as actions or conditions. Here we'll make a custom action: Falstadd gives us nothing for our trouble other than his thanks. The man's understandably upset, but we should at least get some XP:

1. Right click in the "Scripts" window to create a new script. Rename it to "ga_falstadd_xp"

2. Enter this rather short script:

```
void main() {
  GiveXPToCreature(GetPCSpeaker(), 2000);
}
```

3. Open Falstadd's conversation.

4. Select the third node ("I thank you for..."), and change the tab below to "Actions". You'll see the ga_take_item script that we were using before. Click "Add"

5. Choose "ga_falstadd_xp".

There we go, a reward for our trouble! Of course, the problem with this script is that it's not very general. Every time we change the quest's XP amount, we'll have to change it in this script too. I mean, what did we go through the trouble of setting the XP property in the journal if we're not going to use it?

Here's what you'd have to do to make the script more general:

```
#include "ginc_journal"
void main() {
  int iXP;
  // How much XP is this worth?
  iXP = GetJournalQuestExperience("jt_Falstadd");
  // Give it to the party
  RewardPartyQuestXP(GetPCSpeaker(), iXP);
}
```

This gets the XP amount directly from the journal, and gives it to the whole party rather than just the PC doing the talking. An even more general way of doing this would be to have your function take, as a parameter, the tag of the journal entry. In fact, the developers have already done this for us: ga_give_quest_xp is a script that takes a journal entry and gives the party the XP for it.

That's it. We're there! You can play the entire thing, end-to-end. Congratulations, you've created a module!

Extra Credit

Those miners are awful boring, aren't they? They mine regardless of whether you've killed the goblins or not. Give them a conversation wherein they thank the player, award some XP, and then get the heck out of there. Useful scripts for this include "ga_move_exit".

Falstadd's got an awfully short-term memory. He doesn't remember, for instance, that you've told him what's going on in the mine! Add a conversation node to the beginning that'll only appear if the player's completed the quest. Rather than using local ints, you can use the "gc_journal_entry" script to track the player's journal.

The player starts peniless, which makes it rather pointless of us to have started up a shop. Write a script for your module's OnClientEnter to give the player some money. Useful functions here are GetFirstPC and GiveGoldToCreature.

9.3 What Now?

If you wanted to expand this module, there are a few plot holes to wrap up. For instance, the player is aware, once they loot Gnashgab's body, of the fact that Veran is behind the invasion. But even if you take "The Plan" and wave the book in Veran's face, he won't react at all. And of course there's the question of his motivation for doing so. Not to mention there's a whole kingdom beyond Fern!

And if Fern doesn't excite you, you can always create your own module. After all, now you know how!

A Resources

NWN2 Scripting (`http://www.nwn2scripting.com`) If you really want to learn how to script - and I mean really learn, from the bottom up - this is your site. This will teach you how to program in C (the language that NWScript uses) and how to apply that knowledge to NWN2.

NWN2Toolset (`http://www.nwn2toolset.com/`) The graphical buttons on the page can be somewhat hard to read, but there's very solid info here with a great deal of screenshots to guide you along.

NWN Vault (`http://nwvault.ign.com/`) There are a number of tutorials here, as well as modules, hakpaks, and pretty much any NWN resource you could think of. Additionally, you can host your completed modules on-site.

Official NWN2 Forums (`http://nwn2forums.bioware.com/forums/index.html`) FAQs, links to tutorials - even if you never intend to post (you'll need to register your CD key in order to do so in certain sections), there's a treasure trove of information on how to use the toolset available here.

B Plugins

Back in section 8.3, I said that there was no way in-toolset to preview the music available. And that's true. But if you're feeling brave, you can dive into the world of plugins. Keep in mind that this is by no means necessary - you can use the toolset just fine without any additional plugins. But if you want to fill in some of the gaps in the set, here's what you can do:

The plugin I'm going to use is (PowerBar `http://nwvault.ign.com/View.php?view=NWN2PlugIns.Detail&id=34`):

1. Download the .zip file from the Vault. You may have more than one choice, depending on your game version. You can check the version of your game by launching the NWN updater and looking at the first thing it reports. Mine said "Your game version is: 1.06.980 English" (and then it proceeded

to update to 1.10). You can check your toolset version by launching it: The version number appears below the splash screen and also when you select "About..." from the toolset's "Help" menu.

2. This particular .zip file comes with a text file named README. As its name implies, it contains detailed information about the plugin. More importantly, toward the end, it has a section called "INSTALL INSTRUCTIONS". Before following them, be sure to shut your toolkit off!

3. Follow the install instructions. Because they can apply to pretty much any toolkit plugin, I'm going to reproduce them in a more generic form here:

   (a) Copy all .dll files into your Neverwinter Nights 2 Toolset\NWN2Tollset\Plugins folder. By default, this is "C:\Program Files\Atari\Neverwinter Nights 2\NWN2Toolset\Plugins".

   (b) Open the Toolset, and open the "View" -> "options" menu. For security reasons, the toolset won't open any plugin that doesn't come directly from Obsidian. The reason they do this is simple: A plugin is executable code, which means it could, if constructed maliciously, gain control of your entire system and do Bad Things. For this reason, make sure you check out the user comments on the vault! While I don't think there are any NWN2 viruses out there, it could theoretically be done. That said, to allow your new plugin to load, change "AllowPlugins" under "Security" to "Load all plugins"

   (c) This doesn't take effect immediately. You must exit the toolset, then start it back up again.

If everything's worked right (and as of this writing, October 2, 2007, with v1.10 of the game, the plugin loads and runs fine) you should be able to run the plugin from the Plugins menu. You'll be given another menubar (below and to the left of your existing menus) full of options. To preview music, go to the "Browse", "Music" option.

And that, in short, is how to use plugins. Keep in mind, this one was designed to add another menubar with functionality. Others might work differently - giving you an extra wizard you didn't have before, for instance. As always, your mileage may vary. Still, it's nice to not have to start up the toolkit to hear the music, isn't it?