My First Room

or

So you wanna' be a builder, huh?

The purpose of this tutorial is to get a complete newbie up and running with the toolset. By the end of the tutorial, you'll have a small home built, and we'll even go over how you can drop it into the game as a place of your own, where you can stash all of your hard-won loot and cumbersome supplies.

The tutorial is meant for the absolute beginner. If you're good at figuring things out on your own, or you've already used the Construction Set a bit, the tutorial may move a little slow for you. Advanced topics like creating your own unique items, scripting, and module cleanup are covered in later tutorials. A list of all my current tutorials is available here.

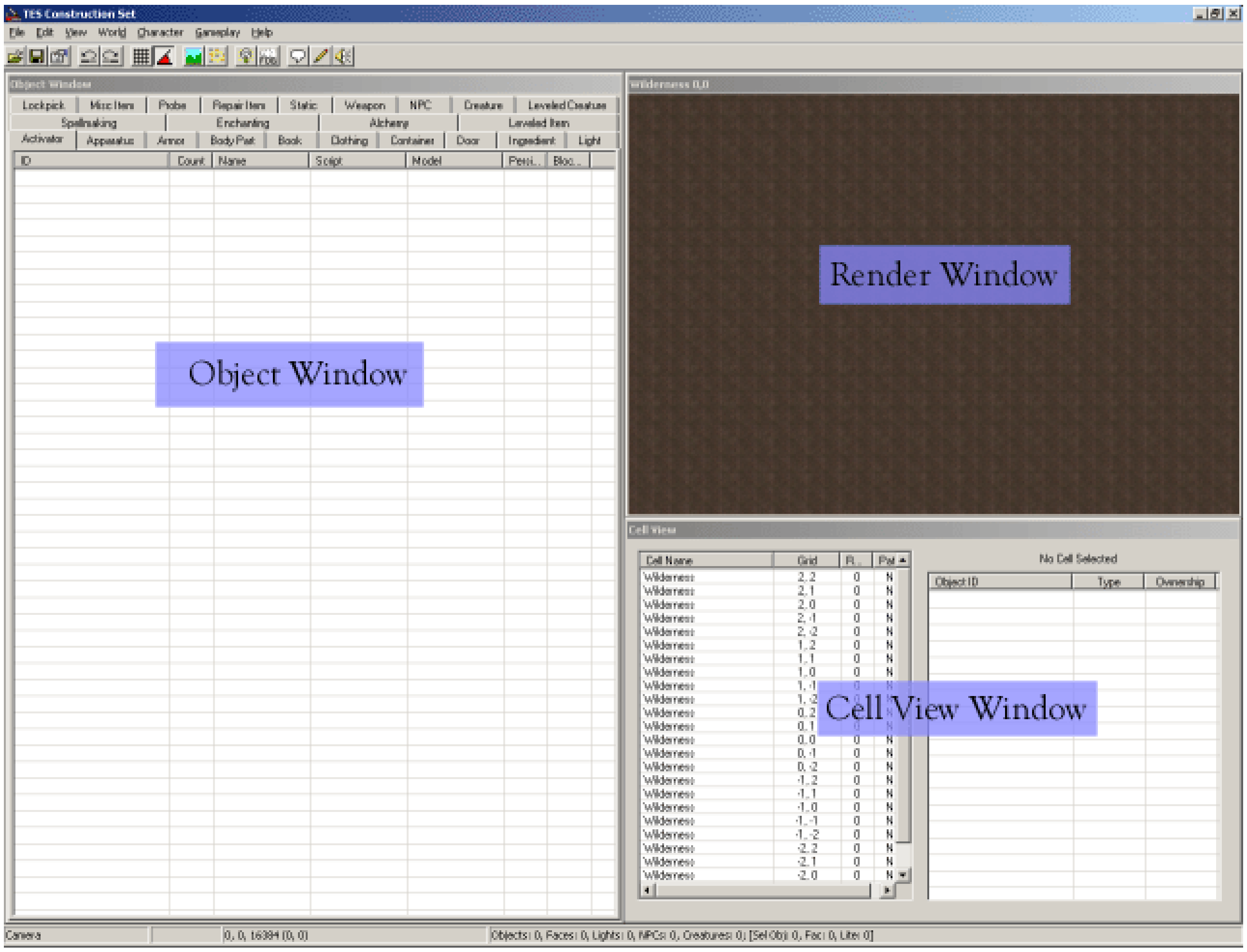
Ground Zero

So you've played around in the game a little and now you're just itchin' to see what that toolset that everyone's talking about can do for *you*, eh? The first thing you're probably wondering is, "So how do I install this thing?" Your game did come with two CDs, after all, and one of them's labeled, "Construction Set".

Actually, if you've installed Morrowind, chances are you've already installed TESCS (The Elder Scrolls Construction Set), too. It was one of your options during installation. If you're not sure if you did, or you're sure you didn't, then go ahead and run the Setup.exe on the CS CD. What's important is that you *do not* try to install this before installing MW. Apparently, the devs tell us, *that's a Bad Idea(tm)*.

Right, all ready to get started? One more thing, first. *Bump your resolution as high as you can possibly stand.* TESCS has lots of windows with lots of information and your Render Window is just one of them. Trying to run TESCS in a low resolution (anything under 1024x768) makes Baby Jesus cry. Well, it makes *me* cry, anyway. I run it at 1280x1024 myself (on a 19" monitor), and I still have a hard time managing my real estate with all those windows.

Once you're inside TESCS, you'll see you have three windows in front of you:

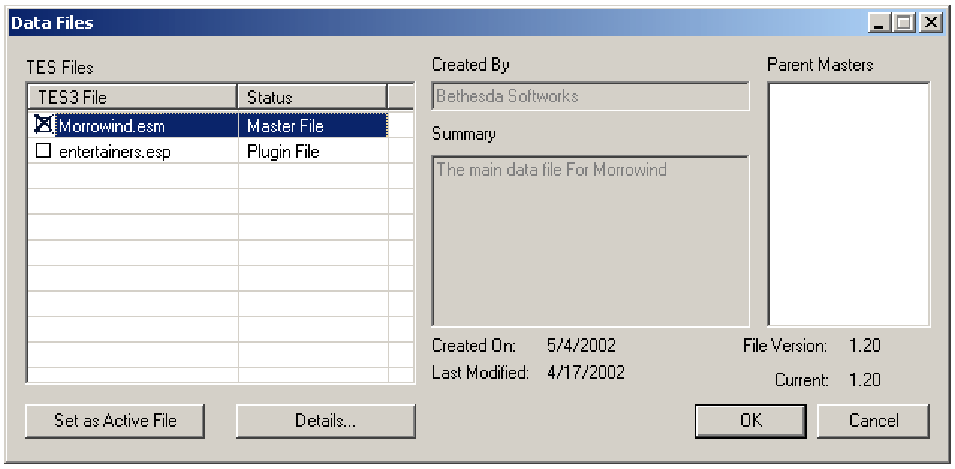

TES Construction Set
File Edit View World Character Gameplay Help
Object Window
Lockpick | Misc Item | Probe | Repair Item | Static | Weapon | NPC | Creature | Leveled Creature
Spellmaking | Enchanting | Alchemy | Leveled Item
Activator | Apparatus | Armor | Body Part | Book | Clothing | Container | Door | Ingredient | Light
ID | Count | Name | Script | Model | Persi... | Bloc...
Wilderness 0,0
Render Window
Object Window
Cell View
No Cell Selected
Object ID | Type | Ownership
Cell View Window

Cell Name	Grid	R...	Pat
Wilderness	2, 2	0	N
Wilderness	2, 1	0	N
Wilderness	2, 0	0	N
Wilderness	2, -1	0	N
Wilderness	2, -2	0	N
Wilderness	1, 2	0	N
Wilderness	1, 1	0	N
Wilderness	1, 0	0	N
Wilderness	1, -1	0	N
Wilderness	1, -2	0	N
Wilderness	0, 2	0	N
Wilderness	0, 1	0	N
Wilderness	0, 0	0	N
Wilderness	0, -1	0	N
Wilderness	0, -2	0	N
Wilderness	-1, 2	0	N
Wilderness	-1, 1	0	N
Wilderness	-1, 0	0	N
Wilderness	-1, -1	0	N
Wilderness	-1, -2	0	N
Wilderness	-2, 2	0	N
Wilderness	-2, 1	0	N
Wilderness	-2, 0	0	N

Camera | 0, 0, 16384 (0, 0) | Objects: 0, Faces: 0, Lights: 0, NPCs: 0, Creatures: 0; [Sel Obj: 0, Fac: 0, Lite: 0]

You'll also find that, browsing through all the tabs in Object View, you don't have a whole lot to work with. What you need to really get started is all the stuff in the Morrowind.ESM file, the file you've been using so far (unwitting or no) to play the pre-packaged game.

Hit the File menu, up at top-left, and select Data Files. Double-click on Morrowind.esm so that it's checked, and click OK.



TES3 File	Status
☒ Morrowind.esm	Master File
☐ entertainers.esp	Plugin File

Created By: Bethesda Softworks

Summary: The main data file For Morrowind

Parent Masters:

Created On: 5/4/2002

Last Modified: 4/17/2002

File Version: 1.20

Current: 1.20

(Note: you may see some "plugin" (ESP) files already listed, any that you've downloaded. You don't need to check them, even if you're using them in your current game. In fact, do *not* check any of them, unless you really know what you're doing. We'll get into more detail on what these Data Files mean, later.)

Woah! Now you have *scads* of stuff to work with. Take a few moments to browse through all of the tabs in the Object Window. It can be overwhelming at first, to be sure, and you may start to think to yourself: "Wait - am I going to spoil any of the game by looking at this stuff? Maybe I should wait until I'm done with the game?" Well ... that's up to you. You *may* spoil a few parts of the game by working with the CS, but myself, I've actually managed to find out very little about the pre-packaged game working in TESCS. I'm not digging through every single object and looking at all the scripts, though, and if you know you're the curious type, but you don't want the plot spoiled, and you can suppress the creative bug for a little longer, then you might want to put off design for a while.

To make all of that information in Objects a little less daunting, let's walk through each of the tabs you have available, to get a brief overview of what each is for:

Activator

There's more to it than this, but think of Activators as switches and levers that you would use to "activate" something, like a door or a gate.

Apparatus

For Alchemy. These are a special kind of object because dragging one onto yourself from your Inventory window will open the Alchemy dialogue.

Armor

You wear this stuff because you don't like dying. Armor, unlike Clothing, has an AR (Armor Rating) and a Health (how much wear and tear it can take).

Body Part

Armor and Clothing are both actually composed of these - you'd use a Body Part as the "Biped Object" for creating your own new piece of Armor or Clothing. Think of the difference between like so: the Body Part is the generic 3D model with no specific information, except which part of the body it attaches to (and upon which layer: skin, clothing, or armor). Armor and Clothing, on the other hand, have specific information, like AR or Enchantments. Two pieces of armor could both use the same Body Part, but have entirely different stats.

Book

Books, notes, writs, and magic scrolls.

Clothing

Unlike Armor, Clothing has no AR nor Health.

Container

These are the chests, barrels, sacks, and crates of the world. Note how each already has its own predefined contents (even if that's "empty"). If you want to make a crate full of crab meat, you *create a new container* and stock it full of that yummy dead crustacean flesh -- but more on that later. (The creating, not the dead crustacean flesh.)

Door

Doors are, well, doors. Actually, if you're clever, you'll quickly realize that "Doors" are actually *Transporters*, they transport you somewhere different when you "activate" them (which means using the Spacebar, by default, in game). So a door doesn't actually have to be a solid mass of wood under a stone archway - notice "Eat Me" and "Drink Me", for example.

Ingredient

You use these things in Alchemy. Only an Ingredient may be used in Alchemy (in the four windows where you may select said ingredients), and only Ingredients provide those "effects" that you mix-and-match to create potions. Say, don't know how Alchemy works? Here's a *real quick* run-down: you combine two Ingredients with the *same effect*, and you get a potion with that effect (if you succeed your Alchemy check). See how Daedra's Heart and Void Salts both have "Restore Magicka" as an effect? Mix those two, and you'll get a potion of Restore Magicka.

Light

Anything that gives off light is a light. A dim glow coming from lichen, a candle on the desk, a torch in a sconce (including the flames), these are all lights.

Lockpick

You use one of these to pick open doors. The game only comes with a pretty limited selection of them, but feel free to be creative! You could use a model for a teddy bear as a new kind of uber-lockpick and stick it on the secret Master Assassin's bed by the cute pink pillow!

Misc Item

These are things you can pick up and carry around that don't otherwise fit into one of the other categories. Gold, keys, kitchenware, and all the little "details" that make the world seem so much more real.

Probe

Like Lockpicks, you use these in Thievery, to disarm traps.

Repair Item

Used with the Armorer skill to repair worn armor and weapons.

Static

You may at first be confused by the difference between Misc Items and Statics. Misc Items are things you can pick up and carry with you. Statics, you cannot. That means the most of the world is actually comprised of Statics - and it's with Statics that you'll be spending the majority of your time building. Everything from landscape to architecture to potted plants to furniture fits into the Static category.

Weapon

Unlike Armor, Weapons do not use a template (Body Part), but rather all of the information for any given weapon is defined by its own unique object.

NPC

All of the actors of the world. *Including yourself!* The star of the show is listed but humbly as, "player". *All the world's a stage, and all the men and women merely NPCs.*

Creature

The difference between a creature and an NPC is that creatures only have a "type" (Creature, Daedra, Humanoid, Undead), whereas NPCs have a Race, Class, and Faction/Rank. Otherwise, creatures are precisely the same (in terms of interface) as NPCs.

Leveled Creature

Also known as Ninja Monkies - because that's what you'll see in the editor for a Leveled Creature. These are creatures that spawn differently based on the level of the player. Take a look at the very first one in the list: ex_ascadianisles_lev+0. The list on the right shows you what kind of monsties you might run into should you drop one of these bad boys down: from a lowly kwama forager at level 1, to the mighty betty netch at level 10 or beyond. Multiple creatures can be listed for a single level, meaning there's a chance that any one of those creatures might spawn.

Spellmaking

If you're familiar with the Spellmaking interface in the game, this will make plenty of sense to you. This is where you create new "Spells". Not *Spell Effects* mind you, but new Spells - like a spell that grants you Levitation at level 40 for 60 seconds, grants you Invisibility for 30 seconds, and bestows Shield level 20 upon you for 120 seconds.

Enchanting

Much like Spellmaking, this is related directly to the Enchanting interface in the game, except that you're designing generic (or specific) enchantments in this case, which you could then apply to Armor, Clothing, Weapons, etc. Again, you can't define new *effects*, but you can mix and match any combination of existing effects however you see fit.

Alchemy

Potions (and such) that apply an enchantment. Spells do the same thing, but Alchemy objects don't require any sort of skill check, and generally come in the form of a potion that the player quaffs.

Leveled Item

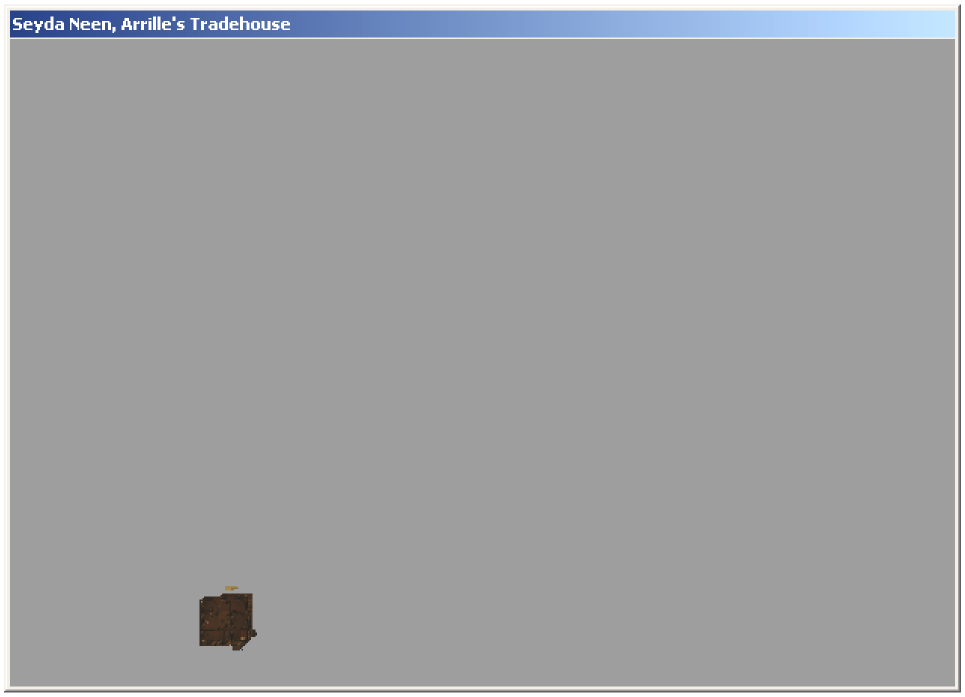
Read the description for Leveled Creature, and now think "Leveled Item". Yup. These are loot-objects that change based on the player's level, so a player at level 1 might open a chest only to find 5 gold inside, while a player level 10 could find a sword of Betty Netch Ass Kicking.

The Cell

By now you're probably *just dying* to get into the editor and actually *do something*, but you'll have to bear with me for just a little longer. Besides knowing what all of those tabs in the Object Window do, it's absolutely *imperative* that you know your way around the Render Window. That's the window at the top-right, which at start just displays what looks like a flat, mottled brown surface.

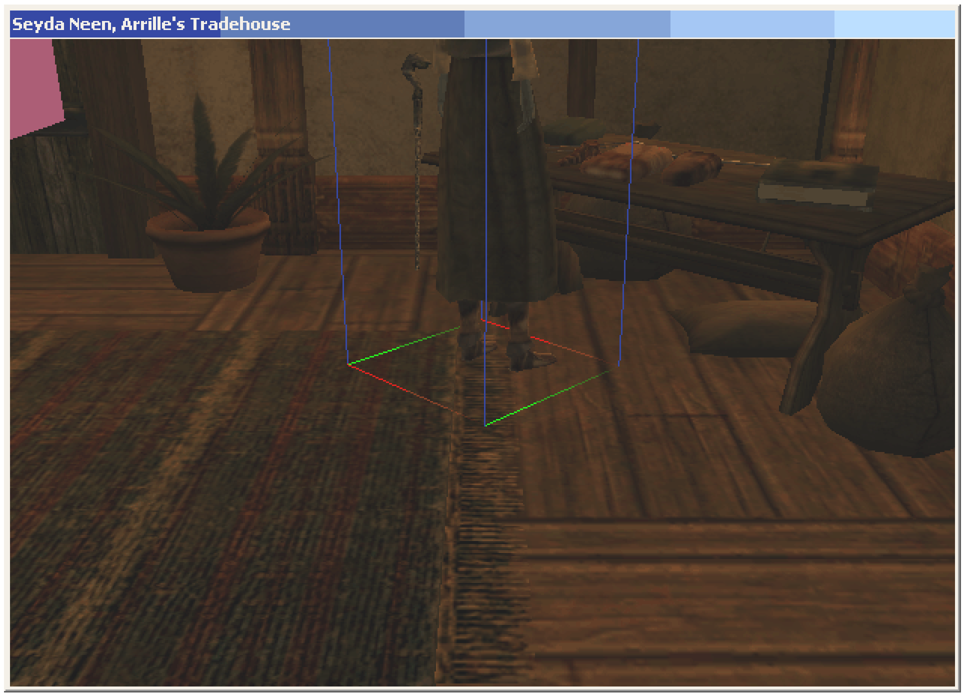
You should have your Morrowind.ESM file loaded, so you'll have a list of cells (Interior and Exterior) in the Cell View Window, below the Render Window. Highlight an entry in the list, and on your keyboard, hit the letter S. You'll see that you scroll down to the beginning of the S entries. You can type multiple letters, too, like 'SE', in short sequence. (Note that you may need to click away from your "result", or use one of the arrow keys, between "jumps".) Find the entry for "Seyda Neen, Arrille's Tradehouse", and double-click on it.

After a brief loading sequence, you'll probably see something like:



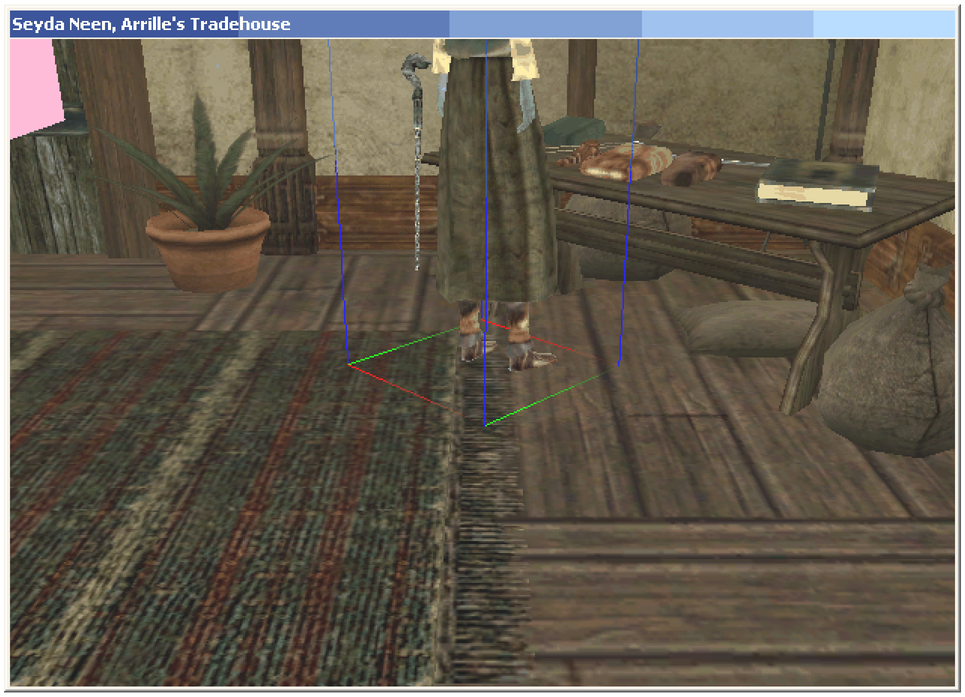
"Um, what the heck is *that?*" you're wondering. Well, it's good ol' Arrille's Tradehouse, indeed - zoomed *way* out. The grey "ether" surrounding the little blotch of color is just that: empty space, the nothingness that you would see were you able to look through a hole in the wall. This isn't actually a part of the rest of the world, it is indeed its own "cell" of space, that is linked with the outside by means of a door that teleports you to a different cell - the cell for Seyda Neen.

The quickest way to get yourself zoomed in and ready to work with a cell is to double-click on one of the objects listed on the right half of the Cell View Window. Try double-clicking Tolvise Othralen.



Ahh, much better. Well, kinda.

It's a little dark in here, and when you're editing things, you'll usually want a very clear view of everything. You can turn on all the lights with the A key:



"Now how the heck do I move around?" you're wondering next, grasshopper? It is good you should ask. Getting a hang for navigating in the Render Window is one of the first, and most important skills you will learn, in TESCS.

First, you'll always find you have to click on the Render Window itself before you can navigate in it. Now hold down the Shift key on the keyboard, and move the mouse around. Holding down the Shift key is the equivalent of putting you into "mouselook" mode.

Get the hang of mouselooking before you go much farther. Hit D on the keyboard (de-select) and see how that affects your mouselook.

You can zoom your view in and out with the mousewheel, or by holding down V and moving the mouse, much like you use Shift. In general, V provides a more granular level of zoom, and is a bit less agonizing than flinging through the wheel five times just to get zoomed out to the right level you want. Now pan the camera around a bit by holding down the spacebar (or middle mouse button), and dragging. Practice using combinations of shift, V, and spacebar to move yourself around Arrille's Tradehouse. Try going downstairs and back up, or turning a circle around an object.

If this is your first 3D editing experience, it may be a bit disconcerting at first to have to move around in so unnatural a fashion: you'll find you want to walk, strafe, jump, and look around an object with it centered in your field of view. It just takes some getting used to. Remember that you're operating in true 3-dimensional space, here, and the controls which allow you to move and look don't necessarily assume such nuisances as gravity and the limited mobility of the human body.

If you get yourself lost, remember that you can always double-click on an object in the Cell View window to re-orient yourself. There are two other ways to get yourself back on track: the T and C keys. Select an object somewhere just off-camera, and hit T. You'll see you zoom in to the object, looking down at it from above. Try the same thing with C. Now you get a "perspective" view. This helps if you see an object that you want to zoom in on or center your view upon, but don't want to go hunting it down in the object list.

Moving around using just the C and T keys can become very natural after a while, and you'll find yourself zipping around your cells with the greatest of ease.

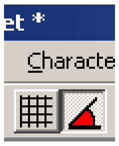
This is handy, too: not sure exactly what you have selected? The name of it appears at the lower-left, as well as its type, coordinates, and cell (respectively, moving over to the right).

Now what about moving objects inside of the Render View? Well, this is done with the most intuitive of actions: simply select the object, and drag it around with the (left) mouse button. (I'll say "left", but if you have your mouse buttons switched, of course I mean your "primary" mouse button.) Give Hrisskar a double-click, remember to re-select your Render Window, hit T to give yourself a better view of him, and drag him around like the big ol' pussycat he is.

Note how you can only move him when your mouse gives you the four-way arrow icon. This makes moving very thin or small objects difficult, so using zoom and mouselook to get a good angle is critical.

Now use Shift (mouselook) to give yourself a different angle on him, and try dragging him around a little more. You should see that you continue to move him only along the X and Y planes (left/right, forward/back), but not along the Z plane (up/down). To move an object along the Z axis, you have to hold down Z, and drag with the mouse. In fact, you can also hold down either X or Y to move an object only along one of those planes, too. This is very useful for aligning objects.

Speaking of aligning objects, you may have noticed that as you drag around ol' Hrisskar, you don't have much precision - you're free to place him pretty much wherever you like. This is usually fine for things like NPCs or objects, but when it comes time to put architecture together, you're probably going to want to make sure everything aligns perfectly, with no visible seams or gaps. This is easily accomplished with the snap-to-grid button:

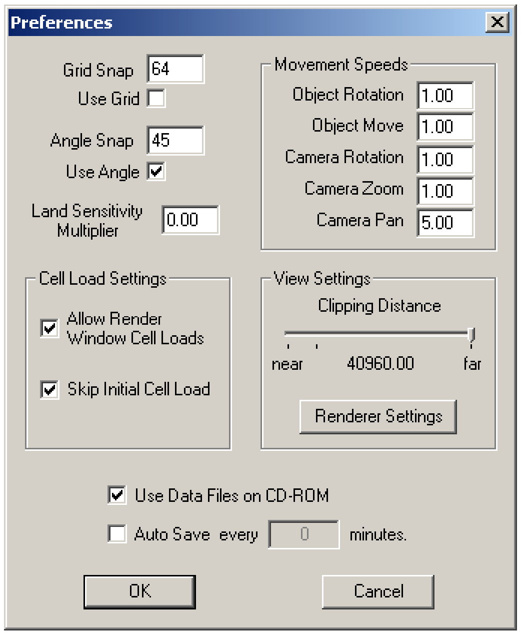


And as you can see, by default it's off. The button next to it, snap-to-angle, is used for rotating objects, and by default it's on. Now, with snapping turned on, try moving Hrisskar again. You'll see he now snaps around at an increment, and you can line him up easily with the walls or the floor.

There's actually a much easier way to get something to align with the floor (or any surface), though: the F key. Use Z to drag Hrisskar a ways up into the air, then hit F. He should land perfectly flat on the floor again. Drag him over to and above the bar and drop him on top of it. Drop him on top of Elone. Trying hitting F multiple times: see how Hrisskar sinks to new depths (he continues to snap to each progressive surface level down). Get used to using F. F stands for "friend".

I mentioned rotating. Yes, you can rotate things, too. What good would your world be if everything only faced north? Well, might be kind of eery in a cool way, but a little boring, too. You use the right (alternate) mouse button to rotate. And as you can see, your rotate is snapped, as well (by default to 45-degree increments). You can turn that off with the other button shown above. And, as with moving objects, you can also change the axis upon which you're rotating by holding down either X, Y, or Z while you drag with the right button. **But,** Z doesn't rotate on the Z axis, as you'd first expect. In fact, **Z rotates on the Y axis**, and you don't need to hold down any key to rotate on the Z axis (it's the default). Yes it's a little weird. But I guess it's not so bad once you get the hang of it. Also, note how NPCs will only rotate on the Z axis: you can only change an NPC's facing, not its tilt or pitch.

You'll probably find you want to change the increments by which you're moving or rotating things, at some point. You can change those increments by going to File, Preferences, and setting them in the Grid Snap and Angle Snap fields:


Preferences

Grid Snap: 64
Use Grid: ☐
Angle Snap: 45
Use Angle: ☑
Land Sensitivity Multiplier: 0.00

Movement Speeds
Object Rotation: 1.00
Object Move: 1.00
Camera Rotation: 1.00
Camera Zoom: 1.00
Camera Pan: 5.00

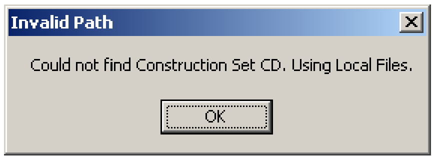
Cell Load Settings
☑ Allow Render Window Cell Loads
☑ Skip Initial Cell Load

View Settings
Clipping Distance: near 40960.00 far

☑ Use Data Files on CD-ROM
☐ Auto Save every 0 minutes.

Some of the other settings in this window bear mentioning, too. The Movement Speeds change how quickly you can move things around with the mouse (including the camera, when in mouselook mode). Not the *increment* of how *far* things move, but how quickly they get there. In general, leaving these at 1.00 is a good idea, unless you find you need to move things really long distances really fast. I also often touch the Camera settings when I need greater or finer control over the camera.

The "Use Data Files on CD-ROM" option is, by default, checked, and you'll probably want to leave it checked, unless you've copied all of the files from your CS CD onto your hard drive (which can be useful, but I won't cover that in this tutorial). You will find, however, that with that option checked, if you hit OK to this dialogue,


Invalid Path

Could not find Construction Set CD. Using Local Files.

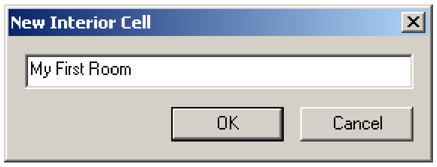
this little nag may come up. It's telling you that you don't have the CS CD in your CD-ROM, so it won't be able to "use" those files. Everything's fine, and you don't have to have the CS CD in your CD-ROM to run the editor (except in more advanced scenarios, where you need to grab some of the meshes and art from the CD), so just click OK if you get that one. You'll still need the game CD in your CD-ROM to test your mod anyhow, so swapping CDs every time you want to look at your work so far could get a little annoying.

Ex Nihilo

Okay, now that you've trudged through all that, you're finally ready to actually do some building, grasshopper.

You're going to want a space of your own to work in. What you need to do is create a new cell - we'll use an Interior Cell, since they're a bit easier to work with, when you're first starting. Go to the World menu up top, and select Interior Cell. Now click the New button.

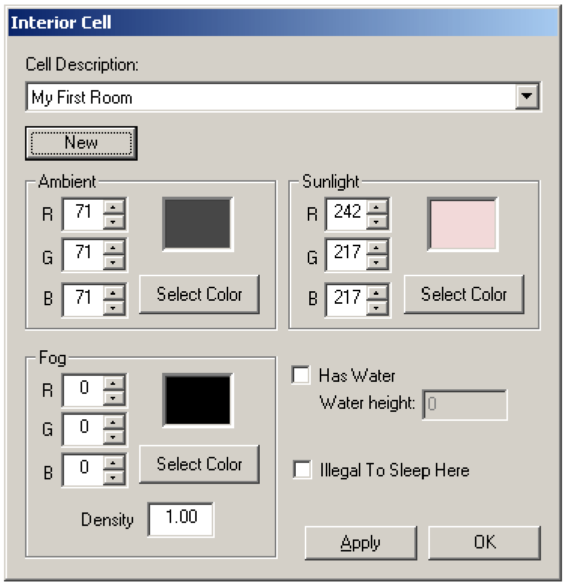
Give it a name:


New Interior Cell

My First Room

OK | Cancel

And just accept the defaults:


Interior Cell

Cell Description: My First Room

New

Ambient — R 71, G 71, B 71 — Select Color

Sunlight — R 242, G 217, B 217 — Select Color

Fog — R 0, G 0, B 0 — Select Color — Density 1.00

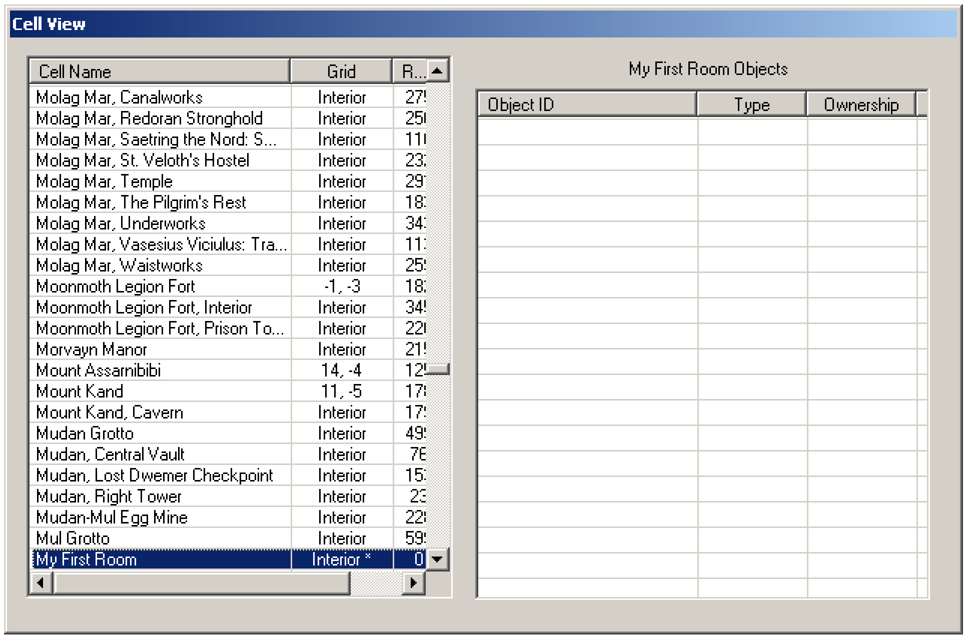
☐ Has Water — Water height: 0

☐ Illegal To Sleep Here

Apply | OK

Click OK.

"Whuh? Nothing happened!" Not to fear, grasshopper. Your cell is there, you just have to open it, same as you did with Arrille's Tradehouse. Find it in the Cell View, and double-click it:


Cell View

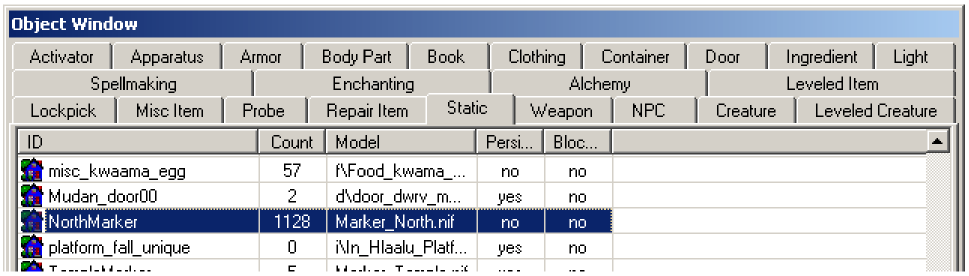
Cell Name	Grid	R...
Molag Mar, Canalworks	Interior	27
Molag Mar, Redoran Stronghold	Interior	25
Molag Mar, Saetring the Nord: S...	Interior	11
Molag Mar, St. Veloth's Hostel	Interior	23
Molag Mar, Temple	Interior	29
Molag Mar, The Pilgrim's Rest	Interior	18
Molag Mar, Underworks	Interior	34
Molag Mar, Vasesius Viciulus: Tra...	Interior	11
Molag Mar, Waistworks	Interior	25
Moonmoth Legion Fort	-1, -3	18
Moonmoth Legion Fort, Interior	Interior	34
Moonmoth Legion Fort, Prison To...	Interior	22
Morvayn Manor	Interior	21
Mount Assarnibibi	14, -4	12
Mount Kand	11, -5	17
Mount Kand, Cavern	Interior	17
Mudan Grotto	Interior	49
Mudan, Central Vault	Interior	76
Mudan, Lost Dwemer Checkpoint	Interior	15
Mudan, Right Tower	Interior	23
Mudan-Mul Egg Mine	Interior	22
Mul Grotto	Interior	59
My First Room	Interior *	0

My First Room Objects

Object ID	Type	Ownership

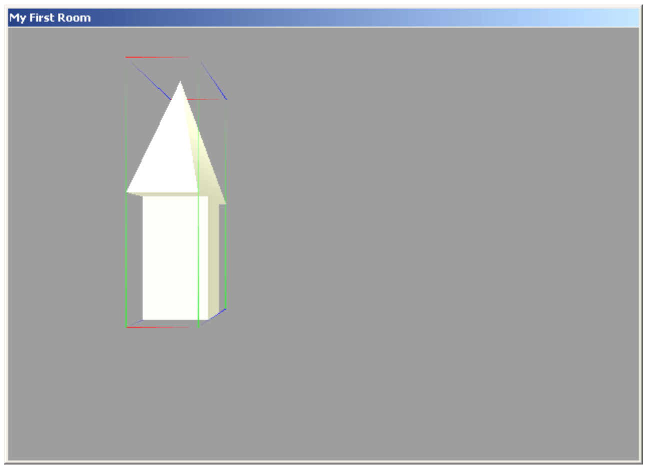
And your new room contains... *absolutely nothing!* Well, well, where to begin?

You always begin by placing a "NorthMarker". This is what helps the player maintain his mystical preternatural understanding of where north always is. A NorthMarker is a Static type of object, so in your Object Window, open the Static tab, and find NorthMarker.



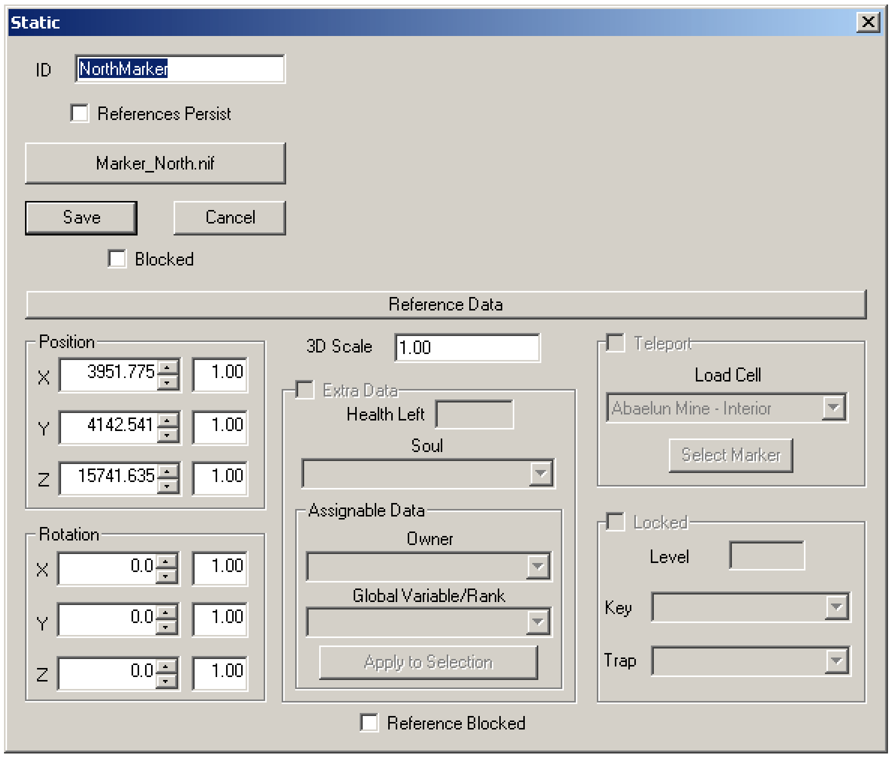
ID	Count	Model	Persi...	Bloc...
misc_kwaama_egg	57	f\Food_kwama_...	no	no
Mudan_door00	2	d\door_dwrv_m...	yes	no
NorthMarker	1128	Marker_North.nif	no	no
platform_fall_unique	0	i\In_Hlaalu_Platf...	yes	no

The trick to adding objects to your cell is drag-and-drop, which may not be immediately intuitive, but does make mod creation pretty quick once you get the hang of it. Drag the NorthMarker anywhere into the endless grey of your Render Window.



Now for a little bit of good-housekeeping advice. It's generally a Good Idea(tm) to center your cell around the coordinates of 0,0. If you've worked with 3D before, this should make sense to you. If you're just scratching your head, then please bear with me. The justifications for a lot of my "good-housekeeping" tips make much more sense later on, but it's better to get into the habit of them now.

Double-click on the NorthMarker. Double-clicking on an object is how you bring up its properties. In this case, we'll see the standard properties window for a Static type of object:



Static

ID: NorthMarker

☐ References Persist

Marker_North.nif

Save | Cancel

☐ Blocked

Reference Data

	Position	
X	3951.775	1.00
Y	4142.541	1.00
Z	15741.635	1.00

	Rotation	
X	0.0	1.00
Y	0.0	1.00
Z	0.0	1.00

3D Scale: 1.00

☐ Extra Data — Health Left, Soul

Assignable Data — Owner, Global Variable/Rank, Apply to Selection

☐ Teleport — Load Cell: Abaelun Mine - Interior, Select Marker

☐ Locked — Level, Key, Trap

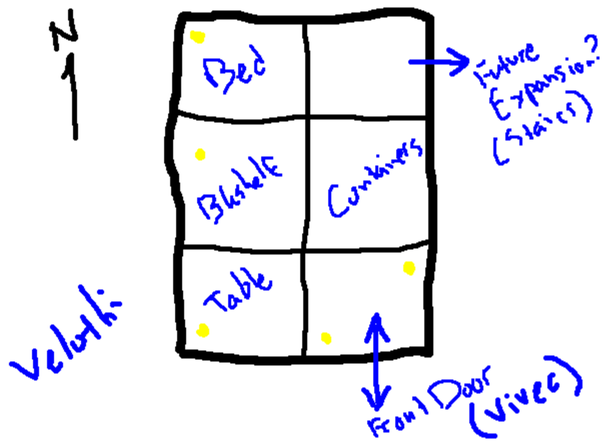
☐ Reference Blocked

Notice how my position coordinates are way funky. The first object you drag into a blank Render Window tends to be way out there. You can manually edit these values, so just plug in 0, 0, 0 for each of them. Then **click the X at the top-right**. Do not click Save. In general, it is a **Bad Idea(tm)** to click the Save button for any object. Unfortunately, the underlying reasons for this are far more complicated than I can get into now (though I do have another tutorial that answers this question in very thorough detail), so for the time being you'll just have to take it from me: it's another of those good-housekeeping tips. **Don't click Save!**

Hmm, now where'd that NorthMarker go? Well, your camera is still looking at the same point in space, but your NorthMarker is somewhere completely different. But you remeber how to instantly zoom onto any object in the cell, right? Sure you do: double-click it in the Cell View Window. (And notice how it's currently the only object listed for this cell.)

Now that we have a NorthMarker, the real fun can begin! Right? Well, no, not quite. There's an important step in building that, sadly, most overlook. It's called *design*, and I can already hear you groaning, but fear not, just this once I've done the work for you, so you won't have to waste that whole extra hour drawing out your design on paper so as to save yourself five hours fiddling around and making mistakes in the editor.

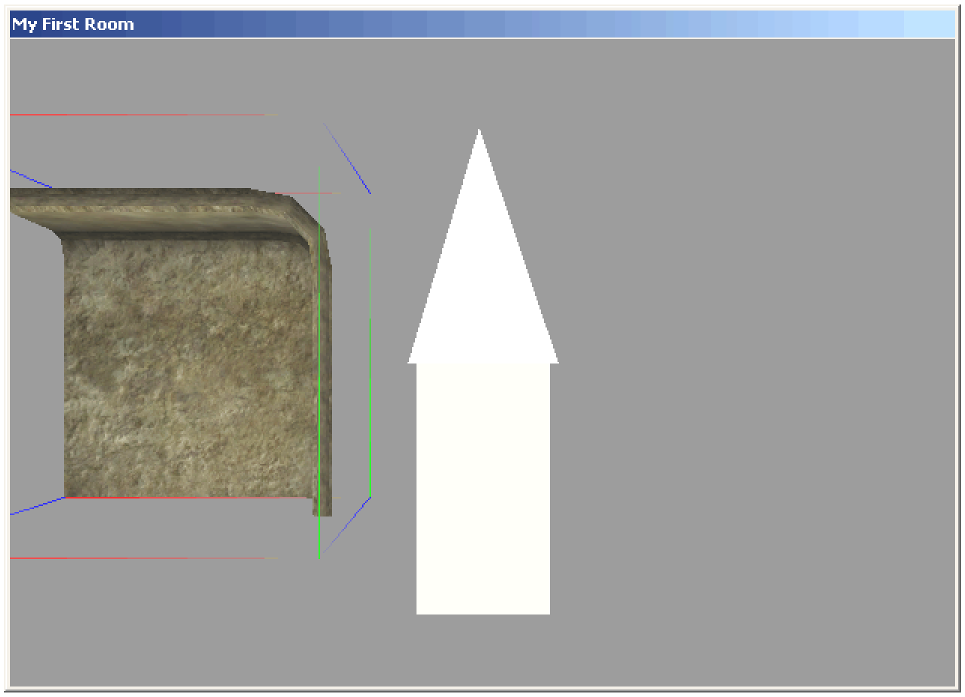
Here's the basic layout your "home" is going to have, to start:



Be it ever so humble! Not a very grand design, to be sure, but we're starting small, after all. At the very end of this tutorial, we'll be setting your new "home" down in Vivec, so I've chosen the "Velothi" architectural style, which is in-line with the rest of Vivec. As you get more accustomed to all of the objects and Statics available to you, your ability to design more complex and architecturally diverse cells will grow.

Referring to our strikingly accurate blueprint, we'll see we have six "blocks" of interior space, based on a grid; four corners, and two walls. Building rooms in TESCS is based on assembling "lego blocks" of architecture, which is certainly far easier than building everything in a strictly 3D fashion, as you would do with the editors for most FPS games. In fact, all you'll need to build the design on our blueprint are two specific "lego blocks": a corner block, and a wall block.

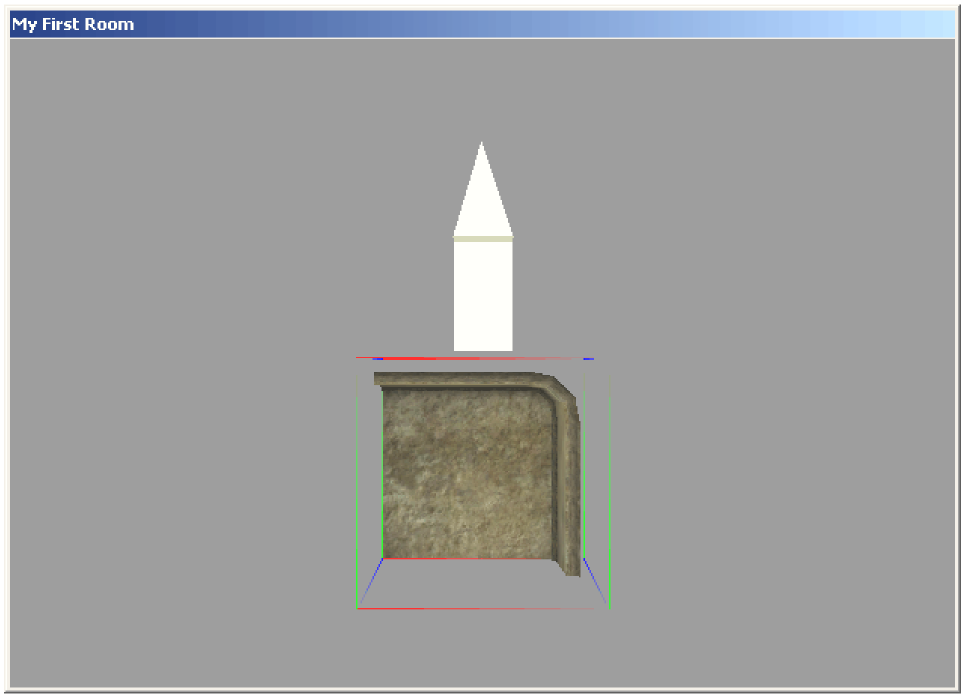
Again, I've done the work for you ahead of time, and found the two specifics Statics you need. The first one is named in_v_s_int_corner_01 (interior - velothi - small - interior corner). Find it in the Static list, and drag it out into the Render Window.



In keeping with our good-housekeeping tips, we're going to set the floor to be at Z 0. Double-click on your corner, and edit the Z position to 0. Leave the other two alone for the time being, we'll get there. Remember, click the X when you're done. Don't worry, the editor doesn't need you to click any OK button, and certainly not the Save button.

Now get yourself a top-down view of the corner with the T button, and make sure you have both grid-snapping and angle-snapping turned on (the two buttons up top, remember?). You may also want to get into your preferences, and set your grid-snapping to 64, since we're moving around fairly large objects, right now.

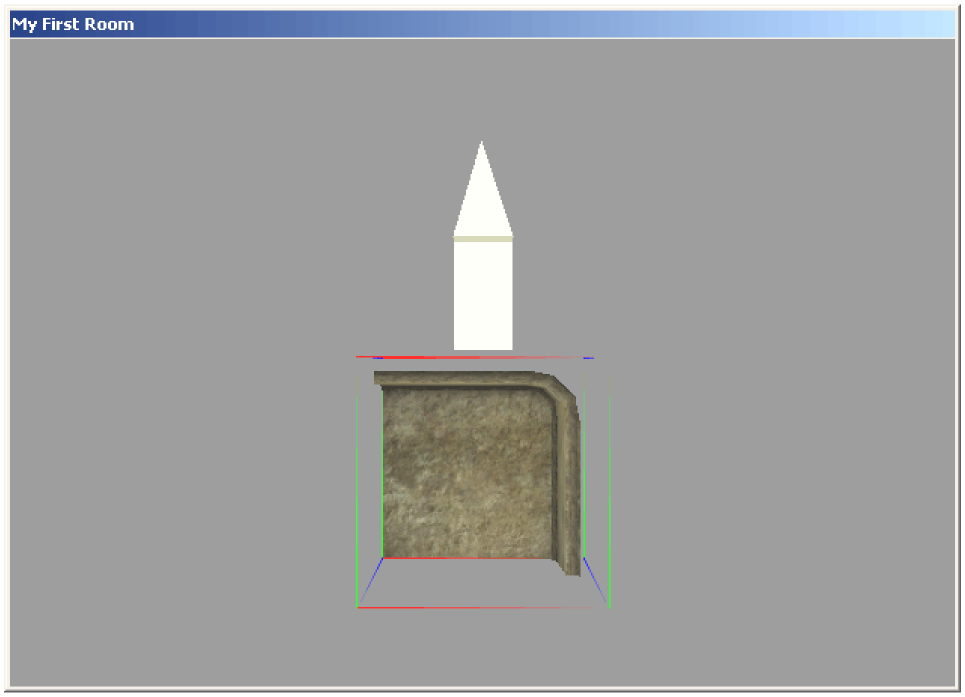
Got all that done? Good. Drag your corner until it's positioned like about so:



If you need to rotate it to get it into the proper facing (the corner pointing northeast), go right ahead. You use the right mouse-button to do so, but you already knew that. Also, it may "snap" down to a different Z level, and you'll have to get back into its properties to reset it to 0. (This tends to happen more often than I like in the editor, mild annoyance.)

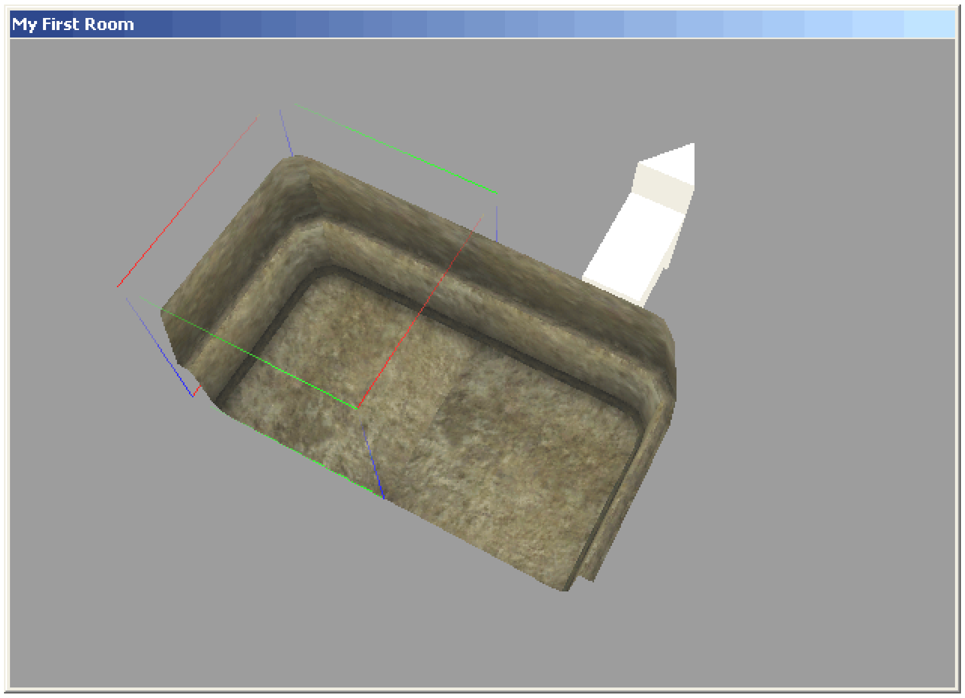
Now we need another corner, that will be just to the west of this first one. There are a few ways you could go about creating it. You could drag in another in_v_s_int_corner_01, and drag it around a while to get it to line up. You could use the Ctrl+C and Ctrl+V keys to copy the corner you have selected, and paste in a duplicate (and drag it around a little, to line it up). You could use Ctrl+C and Shift+Ctrl+V to paste a copy in-place, which makes lining up the new duplicate quite a bit easier. Or just use Ctrl+D, which is exactly the same as using Ctrl+C / Shift+Ctrl+V.

I'll go with the Ctrl+D approach:



Yeah, it's there, it's just that it's *right in the same place* as your first corner, so it appears as though the two are just one. You've also kinda' lost your selection, even though the bounding box still appears (a semi-annoying glitch, I suppose), so you'll need to re-select your corner. (Just double-click either one in the Cell View objects list, below.)

Now get a top view, and rotate the corner in-place 90 degrees counter-clockwise, then drag it out to the left so that it "snaps" together with its neighbor:

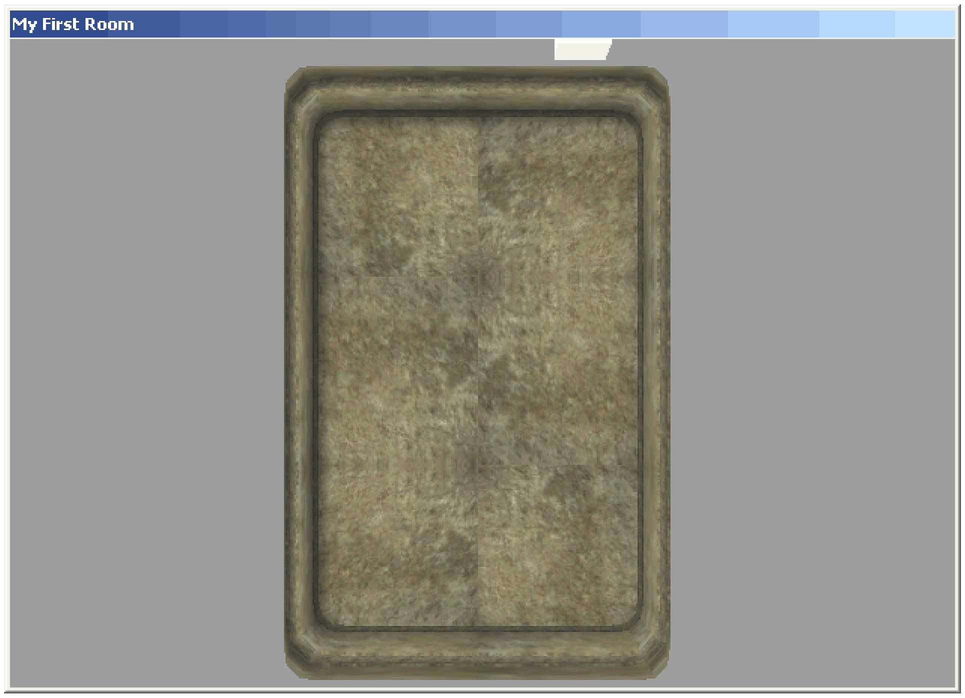


Woohoo! Okay, I know I've been a little harsh on the modem people out there with all these screenshots, so let's head on to a new page so as to be merciful.

RAD

In the programming world, the term "RAD" means "Rapid Application Development", and it usually means a special tool someone uses to make complete programs real fast. Well, here in TESCS, the trick to "RAD" (Rapid Architectural Development) is getting the hang of Ctrl+D, camera panning and zooming, and moving objects on just one axis with the X, Y, and Z keys.

You're going to need one more Static, the v_s_int_wall_01, in order to accomplish the completed bare-bones room shown in the original blueprint. Here's a top-down view of what it should look like when you're finished:



But I'm not going to walk you through it, this time. Get the hang of all the different keyboard shortcuts you need to use, mess up a few of the pieces and get them back into place. Once you've got the above assembled properly, and you feel pretty comfortable with your RAD skills, we'll move on to interior decoration.

Want a handy list of shortcut keys available for when you forget what the one is for zooming in without using the middle mouse wheel? From the editor, click Help, go to Contents, open the Building and Editing topic, Main Editing Windows, and The Render Window. (There's also a link to the same item in Getting Started: What is the Construction Set?, the very first topic.)

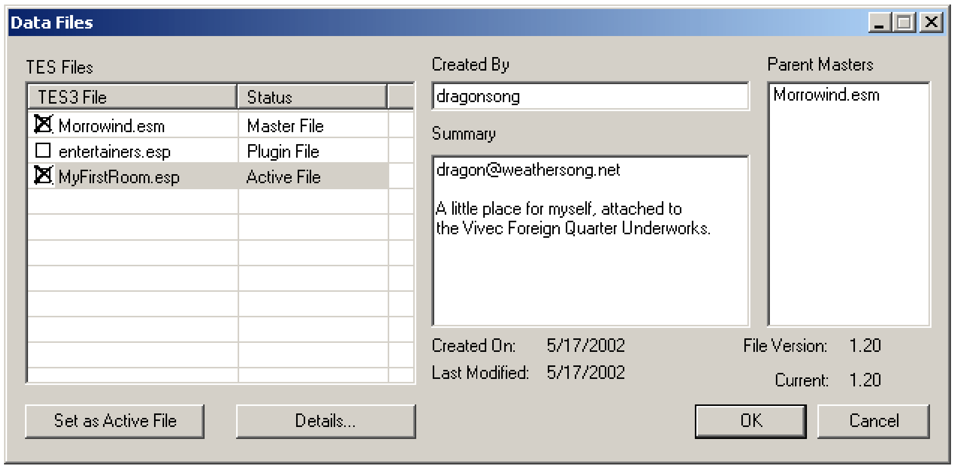
As you're working, you may start to feel like it's about time for that first Save. You certainly don't want to wait until you've got fifteen more Statics down, even if having to start over due to a sudden lightning strike *would* mean honing your RAD skills to an even sharper edge.

Go to the File menu, and simply hit Save. Since this is the first time you've chosen to save, you'll get a dialogue box asking you for the name of your mod (your ESP), and its location. Stay with the default location, since this is where the game looks for ESPs, and name it whatever you like.

And now for a bit of a detour:

What exactly is an ESP file? What kind of information gets stored in it? These may sound like boring details to you, grasshopper, but preparing yourself now for the gritty reality of designing user-friendly, clean mods is much better than realizing halfway through your uber money-dungeon-hack mod that you've been going about it all wrong.

Let's go back to the File menu, and choose Data Files this time.



TES3 File	Status
☒ Morrowind.esm	Master File
☐ entertainers.esp	Plugin File
☒ MyFirstRoom.esp	Active File

Created By: dragonsong

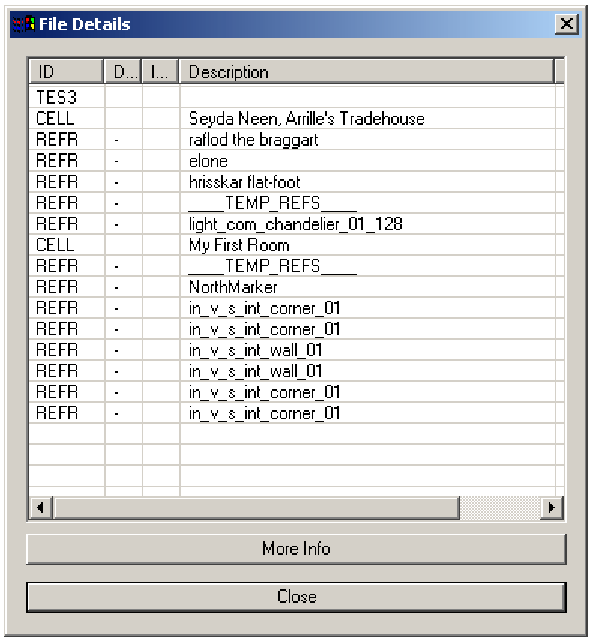
Summary: dragon@weathersong.net

A little place for myself, attached to the Vivec Foreign Quarter Underworks.

Parent Masters: Morrowind.esm

Created On: 5/17/2002
Last Modified: 5/17/2002
File Version: 1.20
Current: 1.20

You'll see your own mod listed, marked "Active File" (since it's the one you're working on, making changes to), with Morrowind.esm checked as well (remember that this is where you get all of your objects from), and there's a space where you can add notes. Now click the Details button:

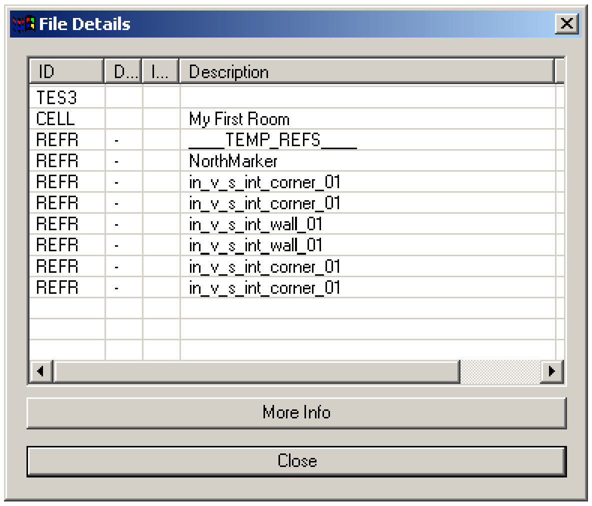


ID	D...	I...	Description
TES3			
CELL			Seyda Neen, Arrille's Tradehouse
REFR	-		raflod the braggart
REFR	-		elone
REFR	-		hrisskar flat-foot
REFR	-		____TEMP_REFS____
REFR	-		light_com_chandelier_01_128
CELL			My First Room
REFR	-		____TEMP_REFS____
REFR	-		NorthMarker
REFR	-		in_v_s_int_corner_01
REFR	-		in_v_s_int_corner_01
REFR	-		in_v_s_int_wall_01
REFR	-		in_v_s_int_wall_01
REFR	-		in_v_s_int_corner_01
REFR	-		in_v_s_int_corner_01

What we have here is a list of everything you've either added to or changed in the world. That is, when you play the game with this plugin activated, all of the things in this list are what get changed.

"Well, funny, why all of those extra things in Seyda Neen?" you should be wondering. (And your list may indeed appear quite different than the one pictured.) Well, those objects are marked as "different from original" because you played around with them when you were first experimenting in Seyda Neen. "But hey! I don't want my mod to change anything like that! How do I undo all of that?" you should be wondering. (You had better be wondering!) Indeed, making sure your mod is "clean" should be a priority concern of yours, especially when you're developing a mod you intend on providing to the general public!

I have a more in-depth tutorial that covers everything about designing and maintaining a clean ESP, but for now, to keep things simple, we'll just leave this list as it is. If you're really very concerned about changes you've made in Seyda Neen or elsewhere, then feel free to scrap your mod and start again from scratch: in the Data Files screen, double-click to de-select your plugin, and with just the Morrowind.esm selected, click OK. You'll re-load the master with no changes, and you can start again from "Ex Nihilo". When it comes time to save, just overwrite your old plugin with the new one. A "clean" details list on your Data File would look like so:



File Details

ID	D...	I...	Description
TES3			
CELL			My First Room
REFR	-		____TEMP_REFS____
REFR	-		NorthMarker
REFR	-		in_v_s_int_corner_01
REFR	-		in_v_s_int_corner_01
REFR	-		in_v_s_int_wall_01
REFR	-		in_v_s_int_wall_01
REFR	-		in_v_s_int_corner_01
REFR	-		in_v_s_int_corner_01

When you're done browsing in the Data Files window, hit Cancel. Clicking OK will re-load whatever you have selected, but if you were just there to look around, Cancel will return you right to where you were working.

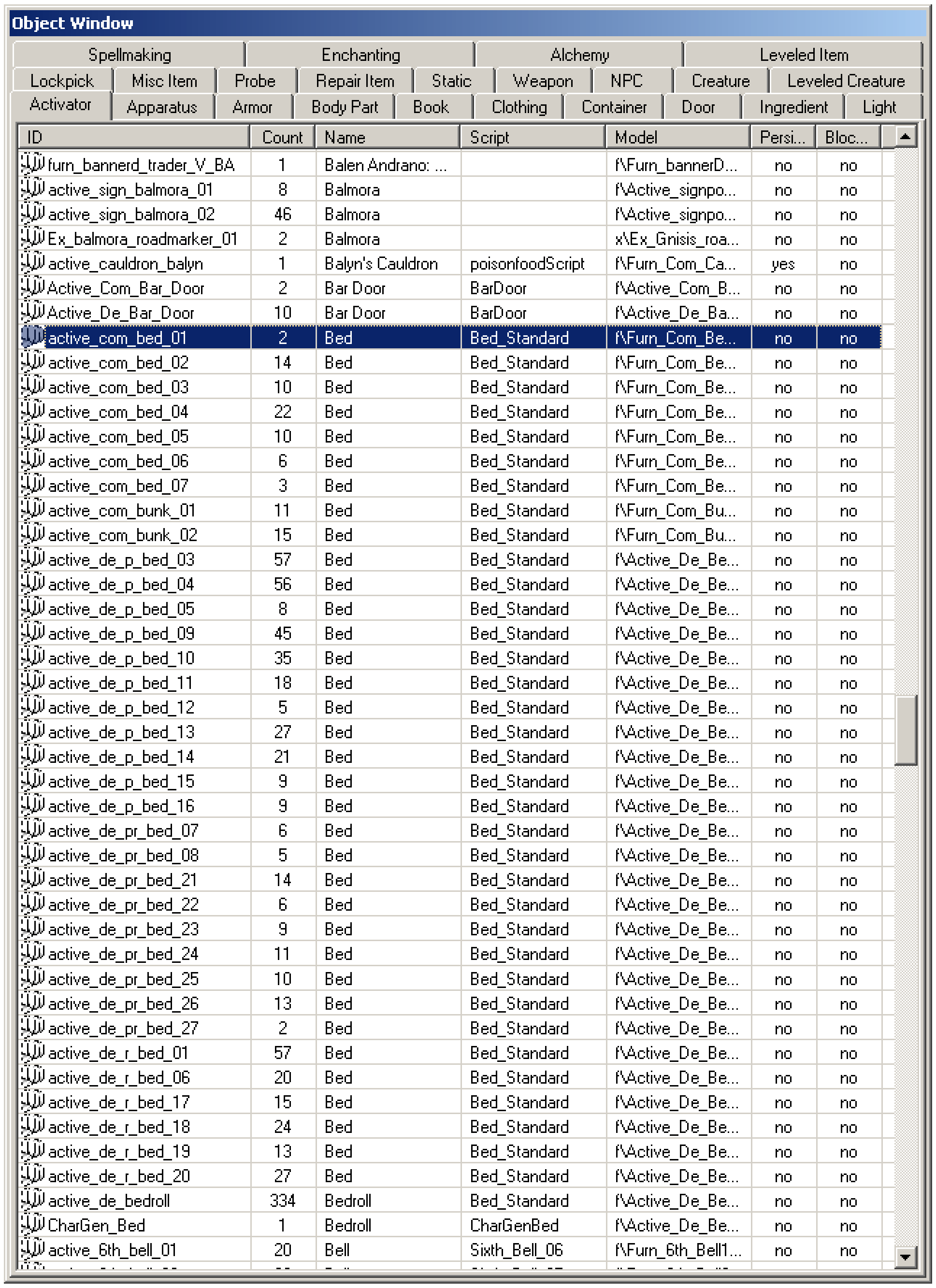
Note that when you exit the editor and come back, you'll use the Data Files dialogue to select both Morrowind.esm and your plugin file, to resume work on it. *You'll also need to choose "Set as Active File"* on your plugin file in order to actually make new changes to it. You'll be reminded of this should you not select an Active File.

Stuff

Of course an empty room with no rugs or wall hangings or furniture is pretty dull, so we need to add some **stuff** to it to make it a worthy home for our intrepid hero!

Going back to our blueprint, we've decided to put a bed in the northwest corner, a bookshelf just south, a table in the southwest corner, and a few containers (to stash loot in) on the east wall. At least, to start.

You'll find most of the things you need under the Statics tab, starting with "furn_", but for the bed you're actually going to need an Activator. That's because beds allow you to interact with them: specifically, when you "use" a bed, it gives you the opportunity to rest. Hit the Activator tab, and hunt down the list of beds under active_com_bed_. You can also click on the column header "Name" to sort all of the objects in the Activator list by Name, making it easier to track down an object you already know goes by the name of "Bed":

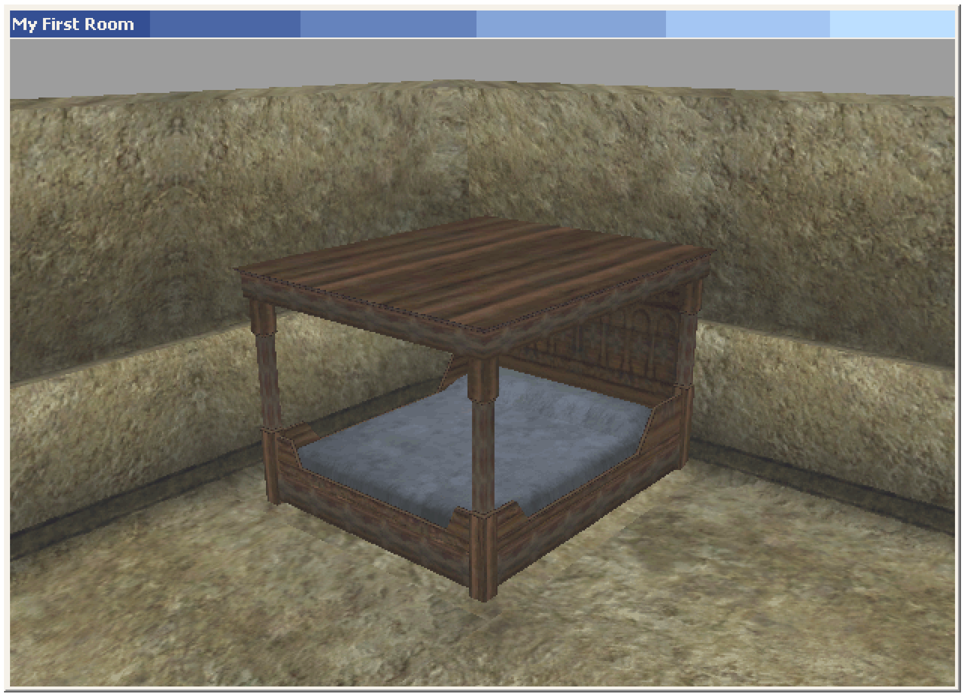


ID	Count	Name	Script	Model	Persi...	Bloc...
furn_bannerd_trader_V_BA	1	Balen Andrano: ...		f\Furn_bannerD...	no	no
active_sign_balmora_01	8	Balmora		f\Active_signpo...	no	no
active_sign_balmora_02	46	Balmora		f\Active_signpo...	no	no
Ex_balmora_roadmarker_01	2	Balmora		x\Ex_Gnisis_roa...	no	no
active_cauldron_balyn	1	Balyn's Cauldron	poisonfoodScript	f\Furn_Com_Ca...	yes	no
Active_Com_Bar_Door	2	Bar Door	BarDoor	f\Active_Com_B...	no	no
Active_De_Bar_Door	10	Bar Door	BarDoor	f\Active_De_Ba...	no	no
active_com_bed_01	2	Bed	Bed_Standard	f\Furn_Com_Be...	no	no
active_com_bed_02	14	Bed	Bed_Standard	f\Furn_Com_Be...	no	no
active_com_bed_03	10	Bed	Bed_Standard	f\Furn_Com_Be...	no	no
active_com_bed_04	22	Bed	Bed_Standard	f\Furn_Com_Be...	no	no
active_com_bed_05	10	Bed	Bed_Standard	f\Furn_Com_Be...	no	no
active_com_bed_06	6	Bed	Bed_Standard	f\Furn_Com_Be...	no	no
active_com_bed_07	3	Bed	Bed_Standard	f\Furn_Com_Be...	no	no
active_com_bunk_01	11	Bed	Bed_Standard	f\Furn_Com_Bu...	no	no
active_com_bunk_02	15	Bed	Bed_Standard	f\Furn_Com_Bu...	no	no
active_de_p_bed_03	57	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_p_bed_04	56	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_p_bed_05	8	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_p_bed_09	45	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_p_bed_10	35	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_p_bed_11	18	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_p_bed_12	5	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_p_bed_13	27	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_p_bed_14	21	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_p_bed_15	9	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_p_bed_16	9	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_pr_bed_07	6	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_pr_bed_08	5	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_pr_bed_21	14	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_pr_bed_22	6	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_pr_bed_23	9	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_pr_bed_24	11	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_pr_bed_25	10	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_pr_bed_26	13	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_pr_bed_27	2	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_r_bed_01	57	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_r_bed_06	20	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_r_bed_17	15	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_r_bed_18	24	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_r_bed_19	13	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_r_bed_20	27	Bed	Bed_Standard	f\Active_De_Be...	no	no
active_de_bedroll	334	Bedroll	Bed_Standard	f\Active_De_Be...	no	no
CharGen_Bed	1	Bedroll	CharGenBed	f\Active_De_Be...	no	no
active_6th_bell_01	20	Bell	Sixth_Bell_06	f\Furn_6th_Bell1...	no	no

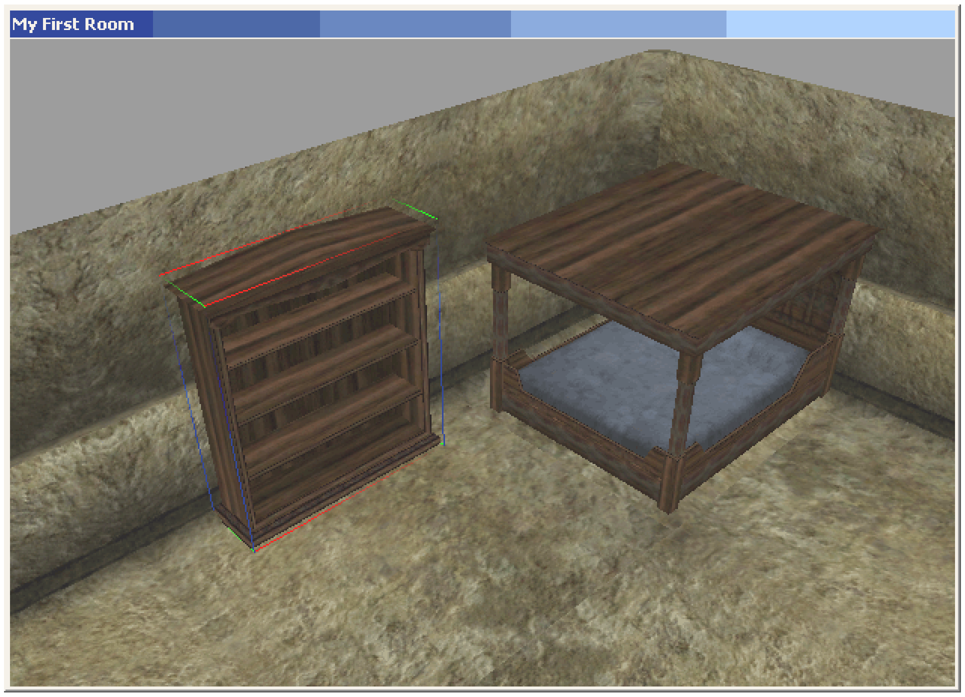
Wow, that's a lot of beds to choose from!

Go ahead and drag a few of them into your world and check them out. If you don't like your choice, just hit the Delete key (with the bed selected) to remove it from your cell. (Note that it's still "clean" to do this.) You can use the Count as a general guide for which are most popular amongst the citizens of Morrowind: in this case it's a tie between active_de_r_bed_01 and active_de_p_bed, although technically the Bedroll beats them both, with a whopping 334 total entries!

Once you've got your bed picked out, get it located in the correct cell, using F to drop it neatly onto floor level. You may find you're ready to jump down to a smaller snap setting, like 32 or 16, or if you like, just place the bed arbitrarily without snap turned on at all. Myself, I wanted something a bit lavish for my illustrious Thief, so I put down this beauty:



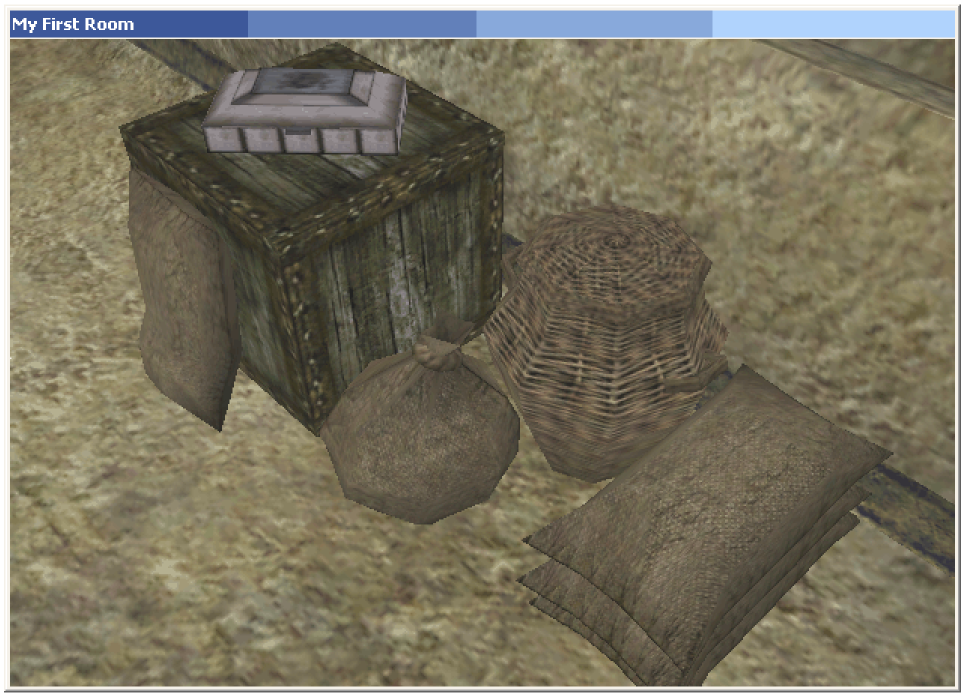
Next on the blueprint is that bookshelf. This time, we're just going to use a plain old Static. Statics don't have "names" so it can be a bit trickier to hunt down the one you want, but familiarizing yourself with everything starting with furn_ will eventually be a good idea anyhow. Don't get too ambitious just yet, though. Find furn_com_r_bookshelf_01, and set it up against the wall:



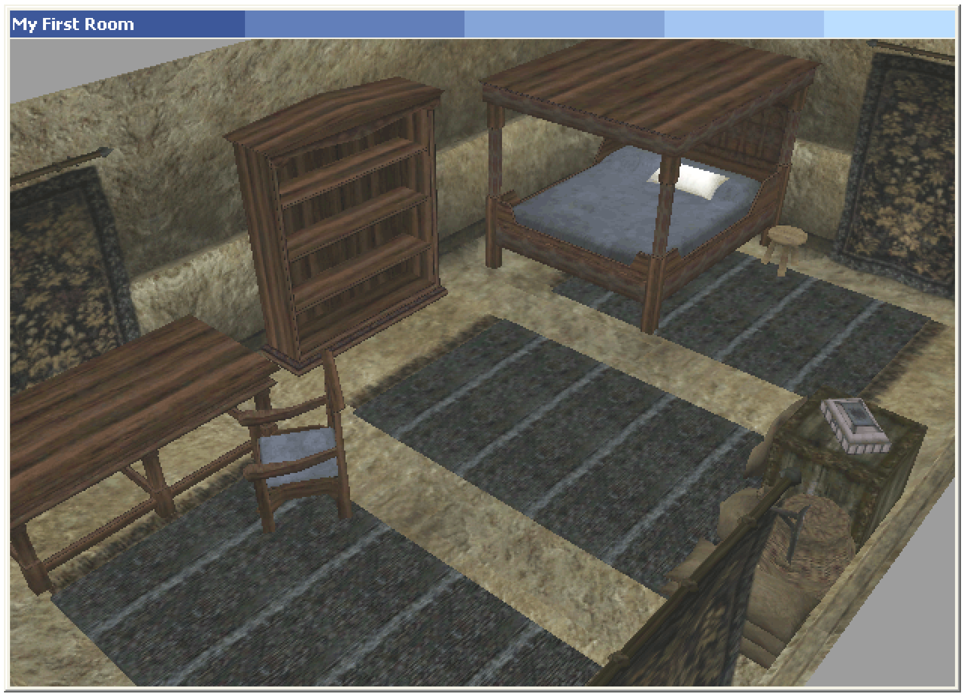
By now you should be getting the hang of getting things positioned in the world, and you should be thinking to yourself, "Gee, this is actually pretty easy!" Indeed, building an entire dungeon or even a whole town should no longer appear so daunting - it's really just a matter of time and attention to detail.

But back to our humble beginnings. The table and containers are next. For the table, use furn_com_r_table_01, and for the containers (found on the Container tab, of course), try an assortment of your own choosing of barrels, crates, sacks, and chests. Make sure you check the contents of the containers you use, though: use empty containers, unless you decide you'd like to give yourself a few presents - but whatever you do, do *not* try and get ahead by adding stuff into the containers you choose. For the same reason that you shouldn't hit Save on any of your objects, do not (just yet) try adding stuff to your containers! (What will happen is you'll change the contents for that same container type *everywhere in the world*.) The next tutorial in this series should cover things such as this, as well as making your own unique items in general.

Here's what I've got, when I finished:



You may be looking around your room and thinking, "Man, this place still looks pretty... bare." Well, feel free - spruce it up. Add more *stuff*. If you feel your character can't really afford much in the way of interior decorating, then keep it simple for now. If you've justified a rather posh lifestyle (after robbing every single noble plantation along the Lake Amaya waterfront) then go nuts. Myself, I decided to just keep things simple for now, and added three rugs (furn_de_rug_big_09), some wall hangings (furn_com_tapestry_03), a chair for my table (furn_com_r_chair_01), and a stool (furn_com_pm_stool_02) next to the bed where I can keep a night-light and something to read for those restive occasions between looting quiet manors. Oh, and a pillow for the bed - misc_uni_pillow_01 (a Misc Item type object). I take one last look around the room, fix up the Z on a few things (the table and the bed), and, voila!



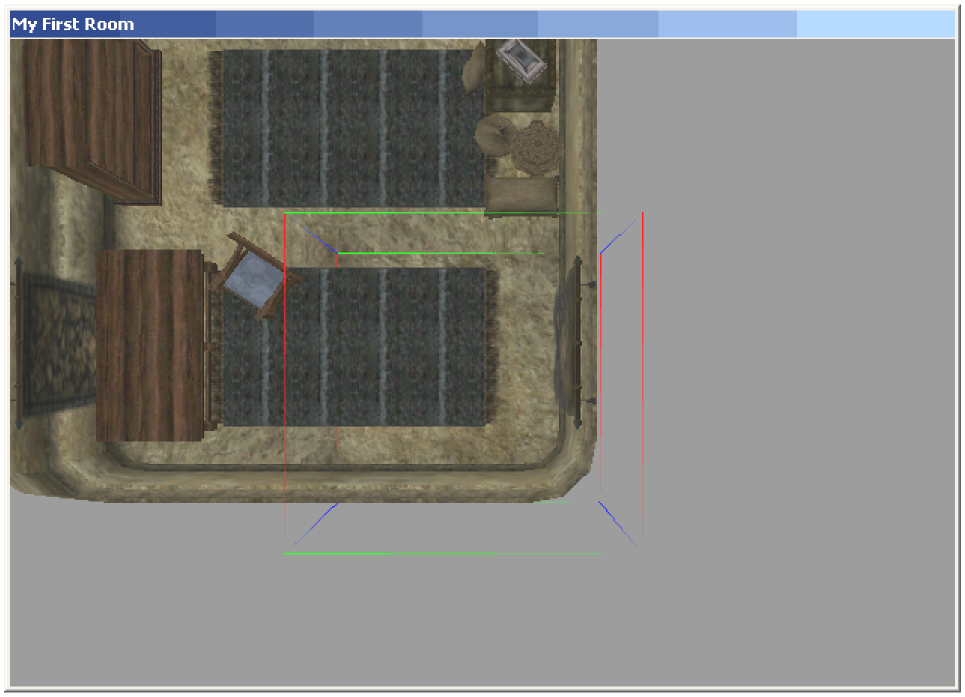
Coming and Going

So now you're probably just about desperate to see how your mod actually *looks*, in game, from the character's point of view.

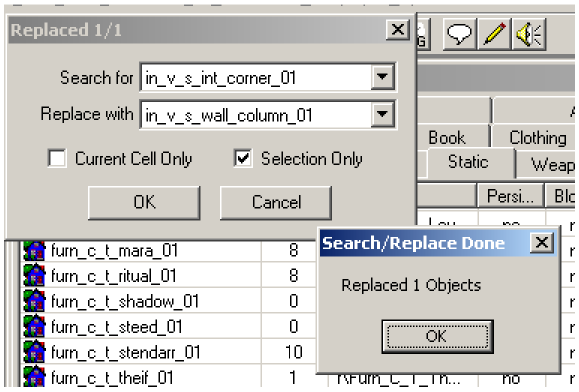
Before you can jump in and have a gander, though, you're going to need a way for your character to actually get into the room, since currently it's not actually linked with anywhere in the outside world.

Let's start by adding a door to the room. On our blueprint, it's in the southeast corner. Right now we have one of those in_v_s_int_corner_01 lego blocks there, but we're going to need to change that, to something with an archway for a door. Here's another good-housekeeping tip for you: if you've already got architecture in place that you just need to replace, it's easiest to use Search&Replace function to put the new architecture in exactly the same spot as the old architecture.

First of all, select the southeast corner:



Now go to the Edit menu, and select Search&Replace. You pick what you want to replace (Search for) and what to replace it with (Replace with). You can replace objects *everywhere* (**not** a good idea) by unchecking both boxes, or in the current cell only, or (what we want) just within the current selection (which can, remember, be more than one object). Note how you'll handily already have the corner object picked for you in Search for. Tick off Selection Only, and in Replace with, choose in_v_s_wall_column_01. Then hit OK:



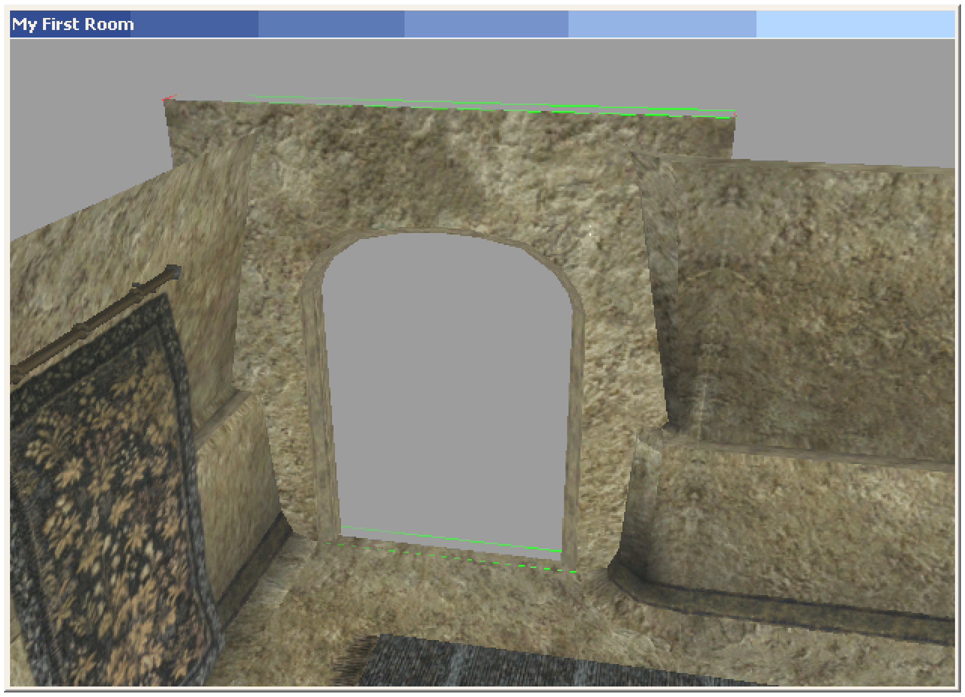
Replaced 1/1

Search for in_v_s_int_corner_01
Replace with in_v_s_wall_column_01
Current Cell Only [✓ Selection Only]

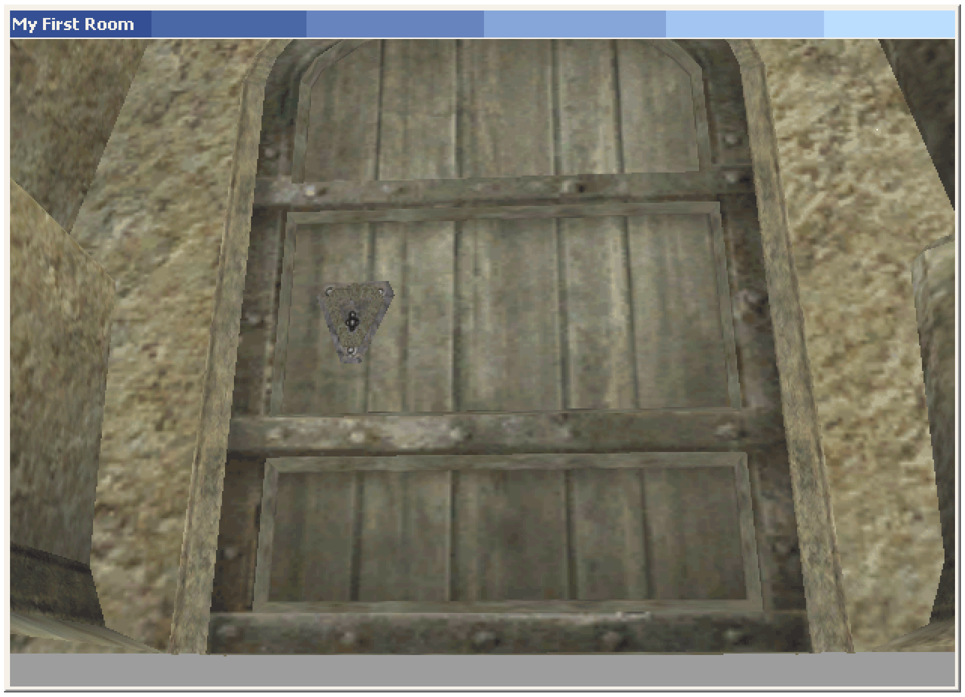
Search/Replace Done

Replaced 1 Objects

The new architecture isn't rotated quite right, though. Shift it 90 degrees clockwise. Now we need to add a "door jam" to the empty space, which will include the archway for the door. This is the Static named in_velothismall_dj_01, and you'll want it set up right on the south edge of your open corner:

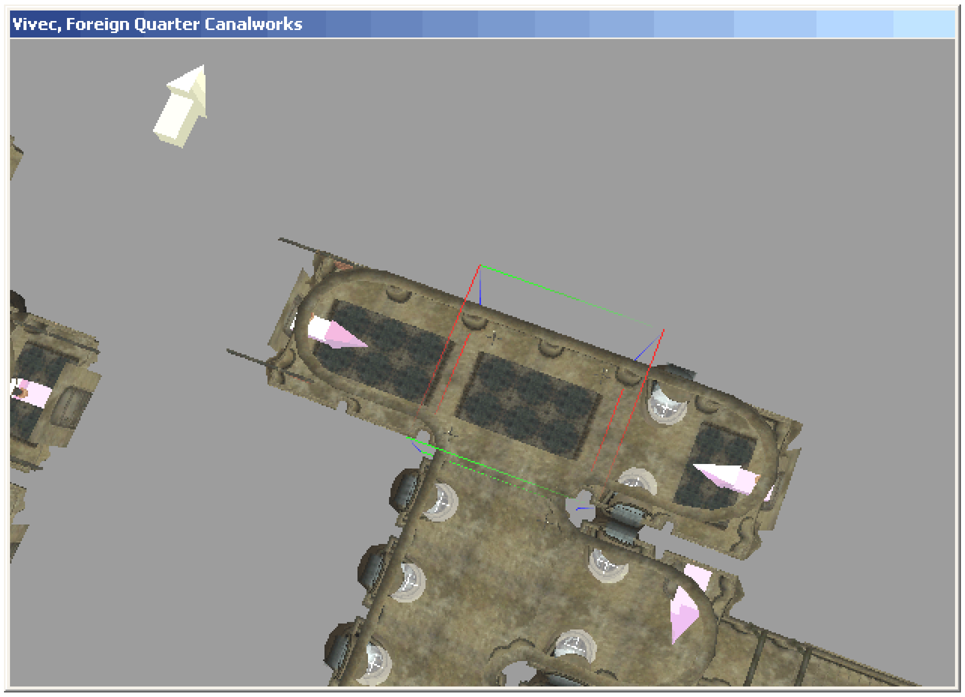


And finally the door - in_velothismall_ndoor_01:



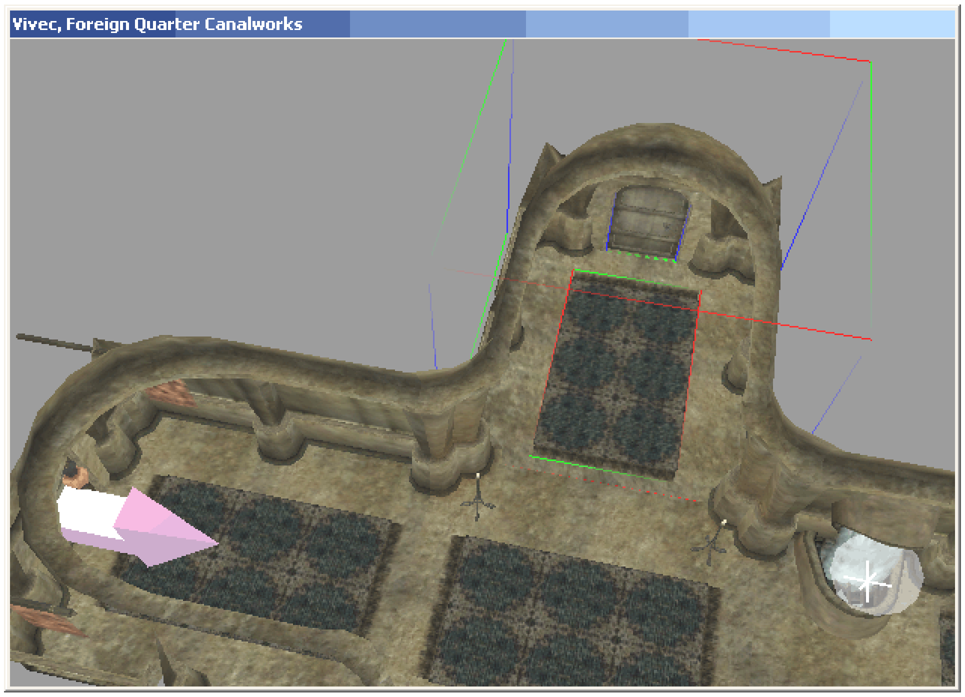
Great, now - where does this door lead? Well, save up your ESP, and let's head outside, to the cell named "Vivec, Foreign Quarter Canalworks".

I browse around and decide on a spot for the entrance to my lair. Rather odd to have a bedroom down in the canalworks, but being the cunning Master Thief that I am, I find it rather fitting that it should be hidden away down in so innocuous and uninviting a place.



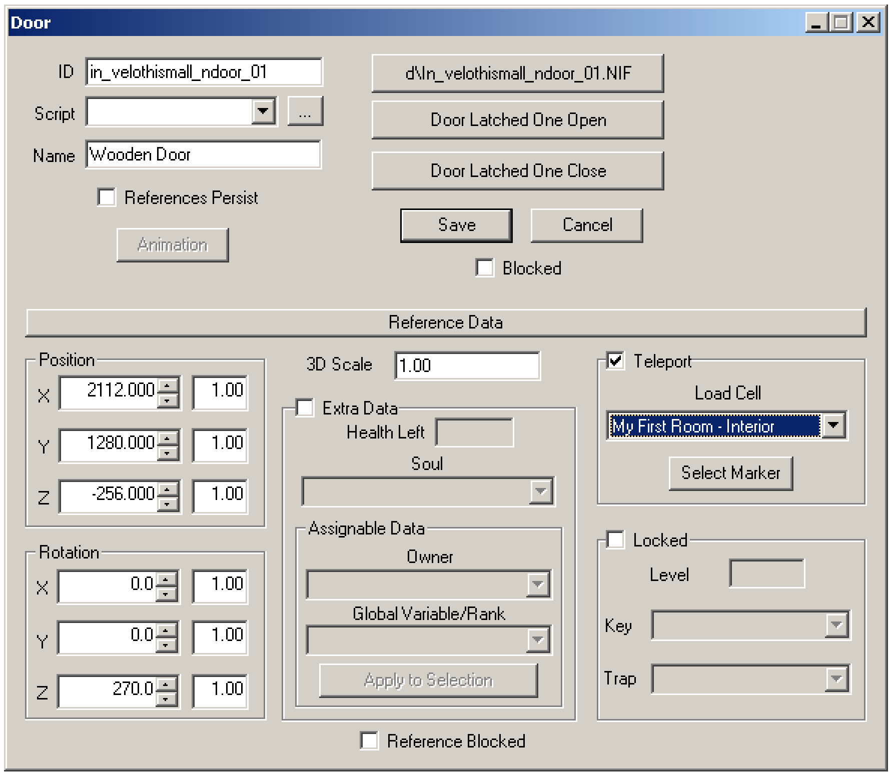
I need to do much the same thing I did in my interior cell: find a fitting replacement for the lego block I want to add my door to, perform a Search&Replace, add the door jam, and then add the door.

In this case, I decide I'm going to replace the 3way I have selected with a 4way, and add a connector to the new open node much like there already are on the west and east sides. I use a multiple selection of the connector, door jam and rug, for a Ctrl+D, that quickly lets me rotate and drag in all of the same stuff from the west hall into the new north addition. Then I drag in a new door for myself (in_velothismall_ndoor_01 again) as the final touch:



Later on, I might decide to put some personalizations outside my door, much like my neighbor down the west hall, but for now I'm just going to leave my entryway looking pretty unassuming.

The last step is to get these doors to actually teleport to one another. Starting with the door out here in the canalworks, I set my door to teleport, and set the teleport destination to My First Room,



Door

ID: in_velothismall_ndoor_01
Script:
Name: Wooden Door
☐ References Persist

d\In_velothismall_ndoor_01.NIF
Door Latched One Open
Door Latched One Close
☐ Blocked

Reference Data

	Position	
X	2112.000	1.00
Y	1280.000	1.00
Z	-256.000	1.00

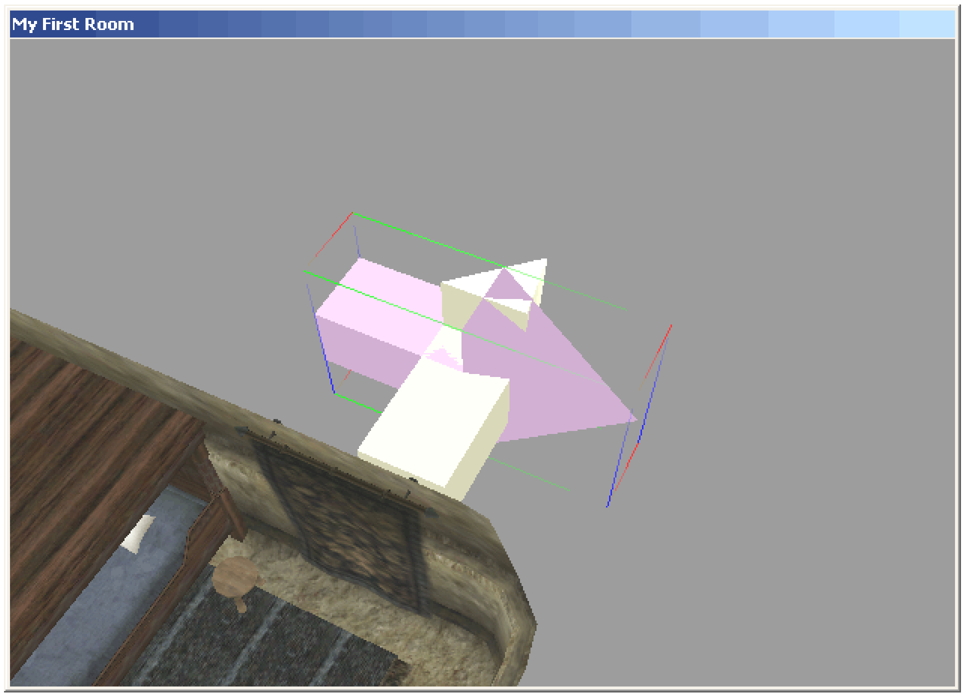
	Rotation	
X	0.0	1.00
Y	0.0	1.00
Z	270.0	1.00

3D Scale: 1.00
☐ Extra Data
☐ Reference Blocked

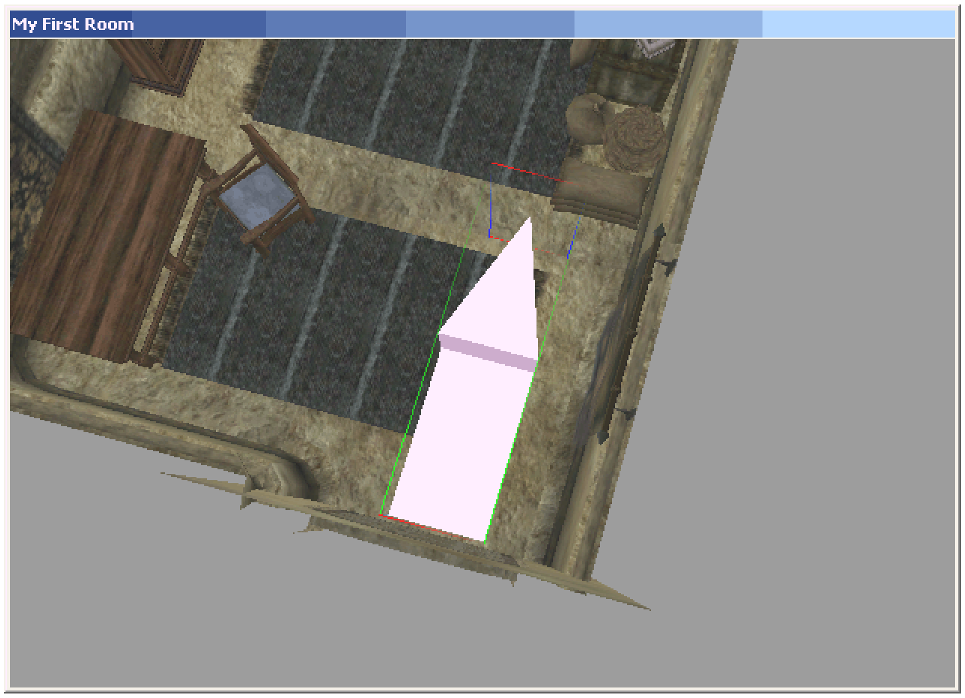
☑ Teleport
Load Cell: My First Room - Interior

☐ Locked

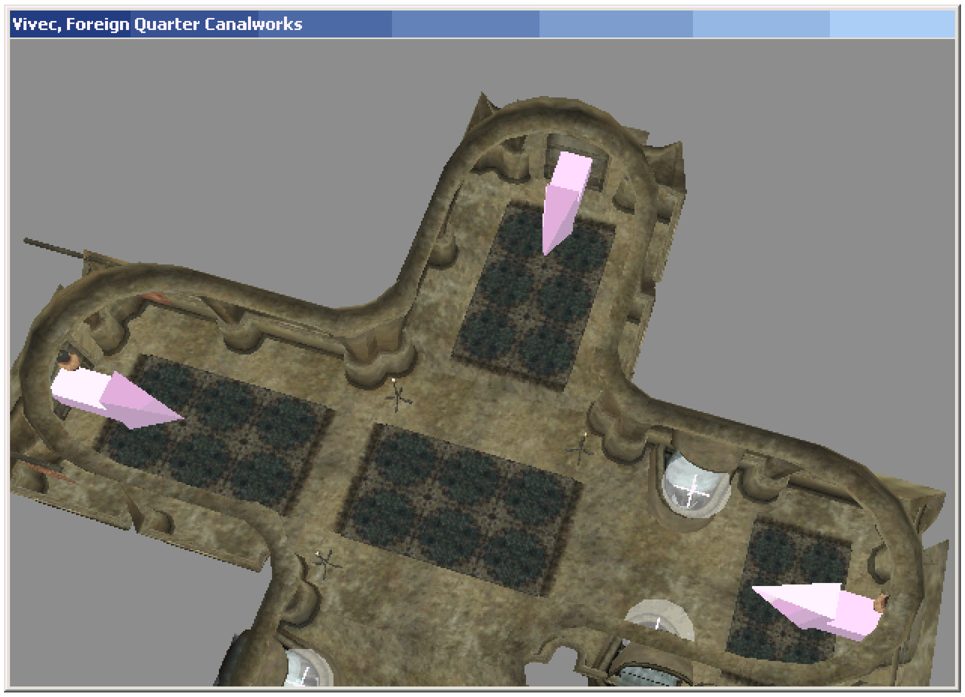
And then click Select Marker.



I'm immediately teleported back to the My First Room cell, my Render Window shows my new DoorMarker (I've changed the view to make it more obvious - note how it defaulted to a location of 0,0), and I still have the properties sheet up for that door out in the canalworks. Close the properties sheet, and set up your DoorMarker to sit in front of your door, pointing in the direction you'd be facing when you first arrive inside:



Now repeat the process for the door inside your room: choose a teleport destination (Vivec, Foreign Quarter Canalworks), hit Select Marker, and position the DoorMarker that appears to sit in front of the new door in the canalworks, pointing forward.



Great! Now, save up your ESP, and start Morrowind. Before you actually jump into the game, choose the Data File options, and make sure your ESP is checked. (You can have other plugins checked too, but in general, when you're debugging your mod, it's best to play with just your plugin checked, and to make any savegames in their own separate track, away from your usual savegames.)

That done, load up a previous save game (one that has you near or in Vivec, if possible), and visit your new room. Yes, grasshopper, *your* new room.

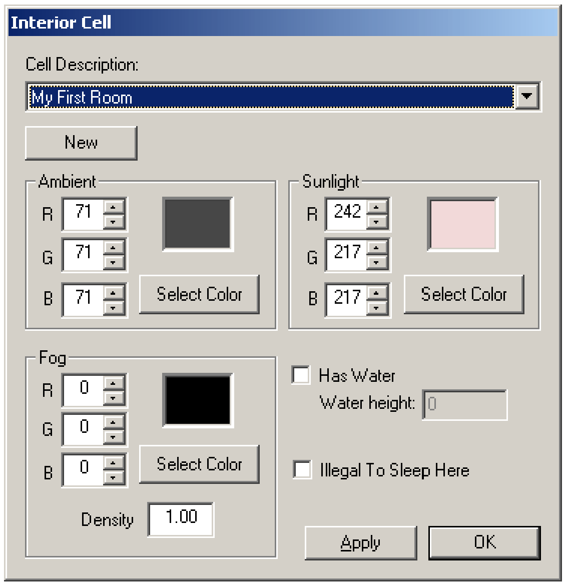
Let There Be Light

I've saved it for last in my tutorial, but you can do lighting first or at any time during the build process, really. You may even skip lighting altogether - as you were looking around your room, you may have noticed that the lighting level was already fairly adequate, a neutral level that's not too bright, not too dim. (Remember that we used the default settings for lighting, when first creating the interior cell.)

But lighting is what really sets the mood for a place, and you don't want just another drab interior, do ya? Heck no, grasshopper, you want *ambience!*

Let's tone down the background lighting, first of all, and then drag in a few candles, again according to the blueprint (those little yellow dots).

Go to the World menu, and open Interior Cell.


Interior Cell

Cell Description: My First Room

Ambient: R 71, G 71, B 71

Sunlight: R 242, G 217, B 217

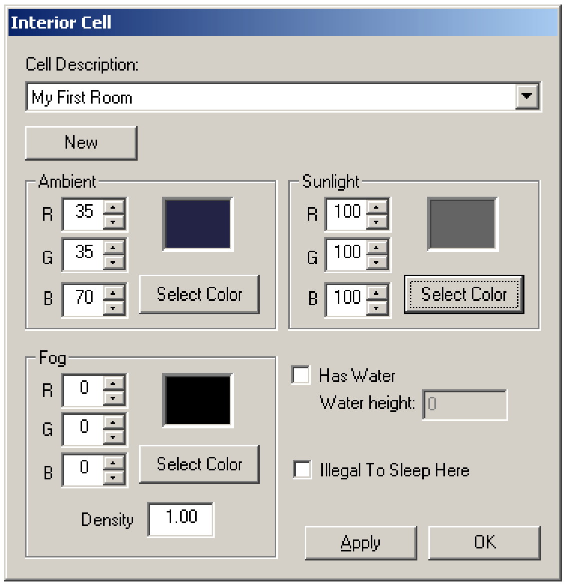
Fog: R 0, G 0, B 0, Density 1.00

Has Water: ☐ Water height: 0

Illegal To Sleep Here: ☐

The values for each of these categories are R, G, and B. That's Red, Green, and Blue. The range for each is 0 (none) to 255 (full). So R 255, G 0, B 0 would be bright red, R 0, G 50, B 50 would be a dim cyan. Yellow is red plus green. (Just trust me on this one.) Of course, you can always just use "Select Color" if you'd prefer to hunt through a pallette, instead of fiddling with the numbers.

I decided I'd give my room just a touch of blue, to complement my existing color scheme, and I've kept things pretty dim. I leave fog alone, and I also tone down sunlight to be a flat "100 grey":


Interior Cell

Cell Description: My First Room

New

Ambient: R 35, G 35, B 70 — Select Color

Sunlight: R 100, G 100, B 100 — Select Color

Fog: R 0, G 0, B 0 — Select Color

Density 1.00

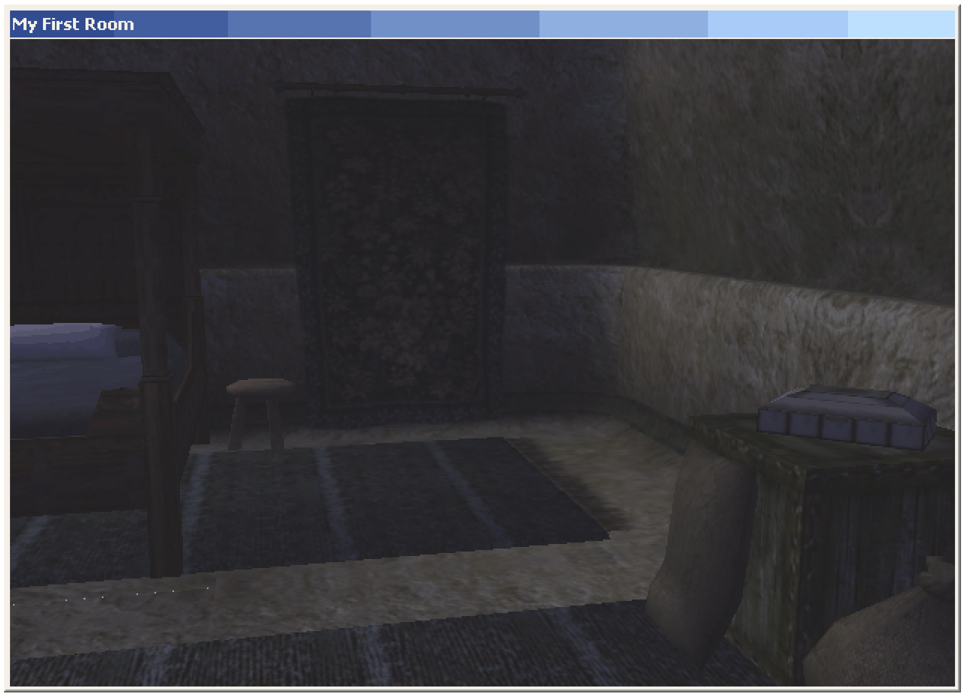
☐ Has Water

Water height: 0

☐ Illegal To Sleep Here

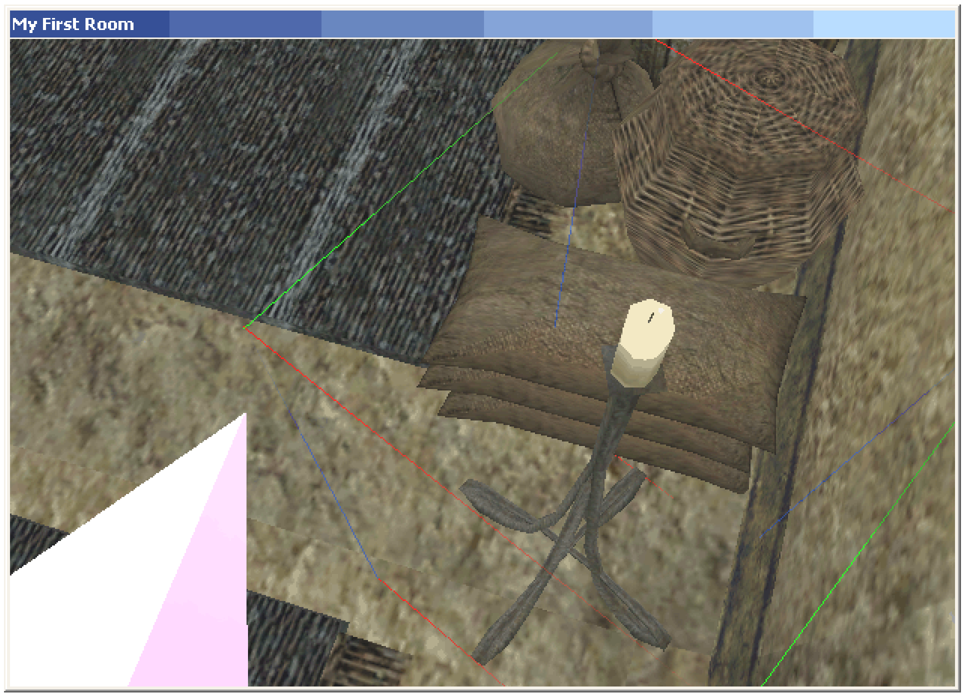
Apply OK

Now in the Render Window, to get an idea for how the new ambient lighting has taken effect, I hit A to turn off the floodlights:



This isn't an *exact* representation of how the lighting will look - at least, I've found that in-game, things still tend to be a little brighter. But this looks about like what I want, so I turn on the floods again (to make editing easier), and hit the Light tab.

I start with the major accents, and then add minor accents. The primary source of light, I've decided, is going to be a bright candle on a stand close by the containers, a little off-center for the room so as to give things a more asymmetrical feel. The light I'm using is light_de_lamp_01_128.

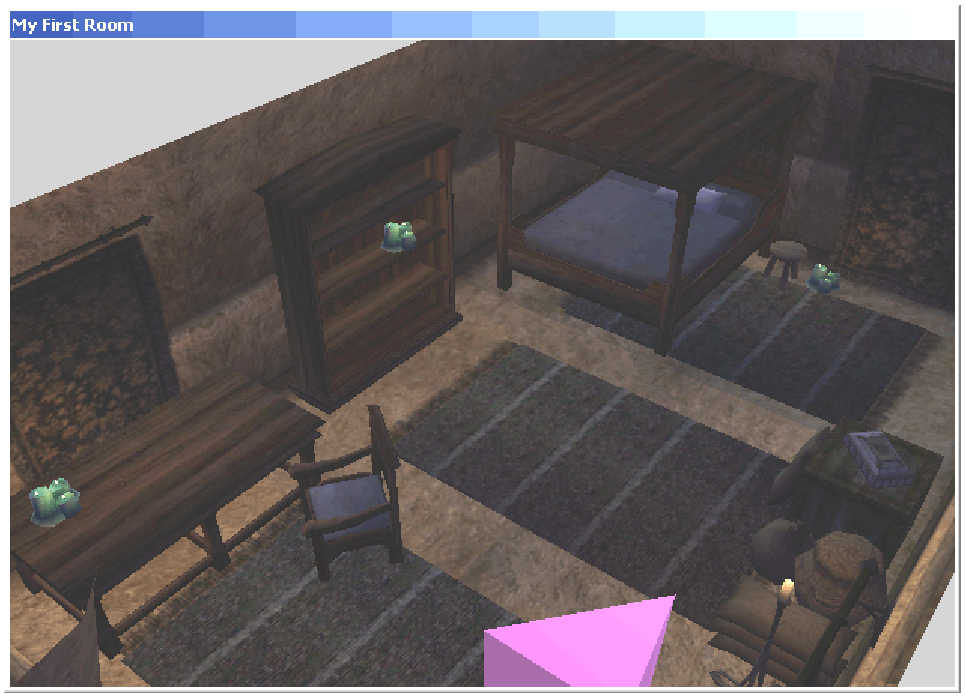


As you move around the light, you may notice that its "effect" (the flame atop the wick) does not move with it. Don't worry, it's not a separate object that you'll have to get placed precisely. It's actually just a bit of a glich in TESCS. To get it redrawn in the right spot, hit L on your keyboard (to toggle the radii for all lights), and then if you don't like all that extra information, hit L to turn it back off again. (The radii can be useful, however, to get a basic idea for how much light coverage you have.)

Now for some candles. I'm going to have one portable candle that I can carry around the room with me, while the rest will be immobile candles. For the portable, I pick out light_de_candle_blue_01_64. It's going to be just left of the door as I walk in. Rather than just drop it on the floor, I also set up another stool for myself, the same kind as I've got next to the bed. Since I've forgotten what kind that is, of course I just click on it, and check the name as it appears at the bottom-left. (Or, even better, just use Ctrl+D on the existing one over there.)

Lights annoy me sometimes because they seem to be less responsive to F (lining up with the nearest flat surface). So sometimes I have to get them positioned just right next to the surface I want, before F seems to do anything. Sometimes toggling the radii (L) seems to help, too.

After the candlestick on the stool, I set down three light_com_candle_04, one on the table, one on the middle shelf of the bookshelf, and then one more down by the nightstand.



Things still seem a little dim - but then, that's just the mood I'm looking for. Of course, for your own room, feel free to add as many of whatever other kinds of lights as you wish. Just one word of caution: do not fiddle with the values of the lights. You may well end up unwittingly changing 500 other lights in the process.

All that's left now is playtesting. Jump into the game and take a look around. Still feel too sparse? Too dark, too bright? Thinking maybe you'd like to add that downstairs addition after all? Save a copy of the room you have now, should you want to follow along with the next tutorial in this series (which will pick up from where this one leaves off), and then build to your heart's content.

The world is yours.

Level 2

or

On the Second Day

This is the second in a set of three tutorials aimed at getting a complete newbie from ground zero to a mod-buildin' wiz. In the previous episode, My First Room, we covered the basics of room building, object placement, doors, and lights. Now we're going to pick up right where we left off (with the very same ESP) and add some new architecture, talk about building a "clean" mod, make some custom (unique) items, and even throw in just a little bit of scripting, too. By the end of this tutorial, you should be comfortable enough with TESCS to accomplish some fairly ambitious work of your own. (However, it won't be until the third tutorial that we cover NPCs, and landscape editing.)

A list of all my current tutorials is available at this location.

Growing Pains

The days drift by; I've taken shelter from many a rainy sunset, stood atop towering peaks to survey pearlescent, misted fields and treetops, and walked through many a dark and barren place, always to return to the quiet and comfort of my own little room, down there in the canalworks of Vivec's Foreign Quarter.

And I'm starting to amass more loot than one room is good for.

So thought of expansion has come to me, now and then, and perhaps you've been struck with the bug to get back into the editor after your initial foray, and see what other kinds of mischief you can get into. After all, the editor seems so thorough and powerful; surely you can do more with it than just set down a room with a few pieces of furniture?

What about creating unique items, powerful spells-- and what's this "scripting" thing people ask about occasionally on the forums?

If you heeded my advice at the end of the last tutorial, you made yourself a good copy of everything we'd accomplished up to the end of My First Room, so we can pick right up from where we left off, with you and me on the same page.

But you may be thinking to yourself, "Well, I've put stuff in that room, is it gonna' show up in the editor?" or, "If I change the room and save the ESP, is that gonna' mess with anything I've got stashed in my room?" or perhaps even, "I'm saved in the room right now, can I make changes to the ESP and then load it up without crashing?"

While I would *like* to tell you to allay your worries, my doubtful grasshopper, there are substantial enough (though generally remote) circumstances in which you could get into a bad spot, that I'll recommend you go outside of your room (to the center of the four-way crossing), and save, and load from exclusively, before and during your changes to the ESP. If you're especially paranoid, you should also make a backup of your current ESP, so that in the event you *really* mess things up this round, you can always revert to it.

As an additional (highly optional) safeguard or troubleshooting measure, you can run Morrowind without your plugin selected (as a Data File), load the save I just recommended you make (where you'll now be facing a blank wall at a 3-way crossing), and save again - this time without the ESP applied. (Thereafter, load this save *with* your ESP applied, again.) This will guarantee that every time you load this save, with your newest version of the ESP, that any previous changes to the ESP will not cause you worries in the latest version. (You would *not* want to do this, however, if you've already got things stored in your room.)

I've once again done the design work for you ahead of time. I'll spare you the excrutiatingly detailed blueprints, though - I just want to re-emphasize the importance of getting your design work done *ahead of time.*

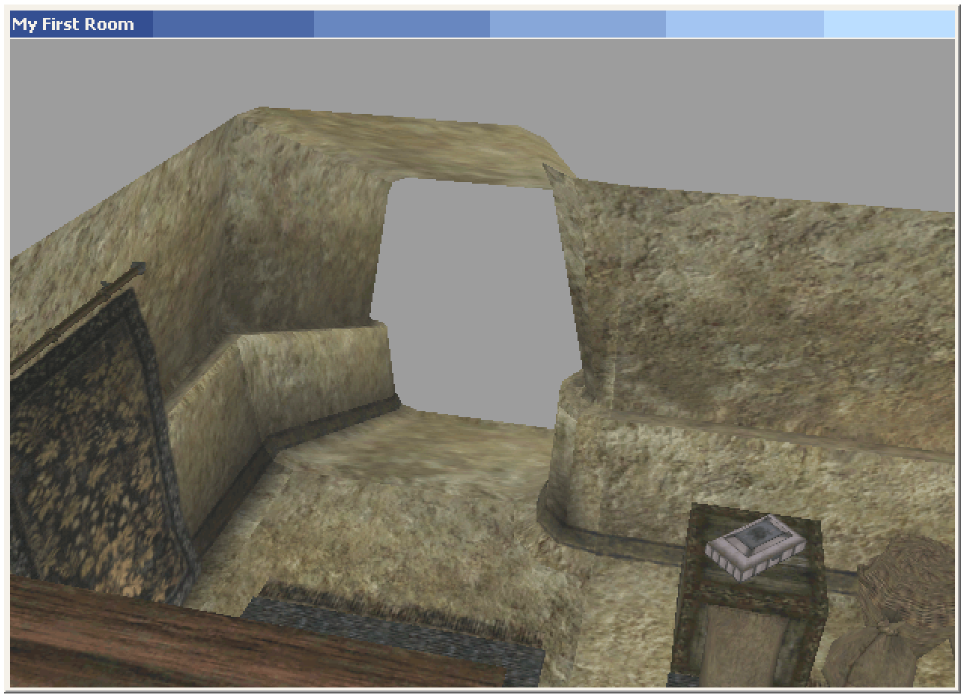
Open TESCS and open your ESP. Remember to set your ESP as the Active File when selecting it. (You'll be prompted should you forget.) Find 'My First Room' in the Cell View, double-click, turn on the lights (A), and double click DoorMarker. We're all set.

It may take you a moment to realize that your room is so stark-looking because an ESP is not a Savegame. The stuff you've stashed in this room is "stashed" in a separate file, the file for your Save, while the ESP itself just represents the architecture of the room, if you will. So you could indeed get yourself into trouble by removing a container, or even adding something - like a candle on the table that happens to be in the same spot as a ring in your Savegame, preventing you from being able to get to the ring. If you're going to be developing mods that you'll be releasing to the public, this is going to come back to haunt you. About the best advice I can give you is to test your mod thoroughly before release, so that you'll have to make as few fixes (changes) as possible afterward.

If it's just for your own private use (which is what this series of tutorials is generally aimed toward), you have a bit more freedom. In the worst case, you could always just get into your Savegame to move some loot around so as to allow for changes to the ESP.

I've been through almost an entire page now without even a single screenshot, so the following is entirely obligatory:

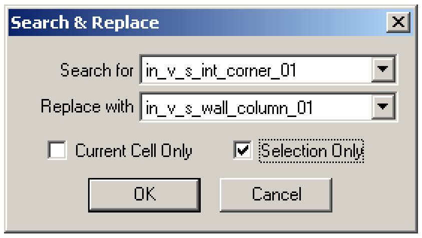
Right. Actually, I do have a useful screenshot for you:



This is how we're going to modify the existing room to lead into the new downstairs. You'll start at the northeast corner, as shown, and replace the Static there with one allowing for a connecting Static, the ramp leading down.

Since you may be still feeling a little green or maybe a bit rusty after being away from TESCS for a while, I'll walk you through it, don't worry.

Select the "lego block" that makes the northeast corner - you'll see the name of the Static, in_v_s_int_corner_01, in the bottom left corner - and now use the Edit menu to select Search&Replace. In the window that appears, choose to Search For in_v_s_int_corner_01 (if that isn't already supplied for you) and to Replace with in_v_s_wall_column_01. Make certain to check Selection Only.



Search & Replace

Search for: in_v_s_int_corner_01

Replace with: in_v_s_wall_column_01

☐ Current Cell Only ☑ Selection Only

The new Static will need to be rotated, so that the open side is facing east. Now browse through your list of Statics to find in_velothismall_ramp_01. Drag it into your cell, and then position it accordingly. Remember to use grid snapping to help you with alignment, F if you need help getting things aligned on the Z axis, and the X and Y keys to restrict movement or rotation to just one axis. And if you make a mistake you'd like to undo, use Ctrl-Z to (un)do just that.

Now, should we paint the walls blue, or green?

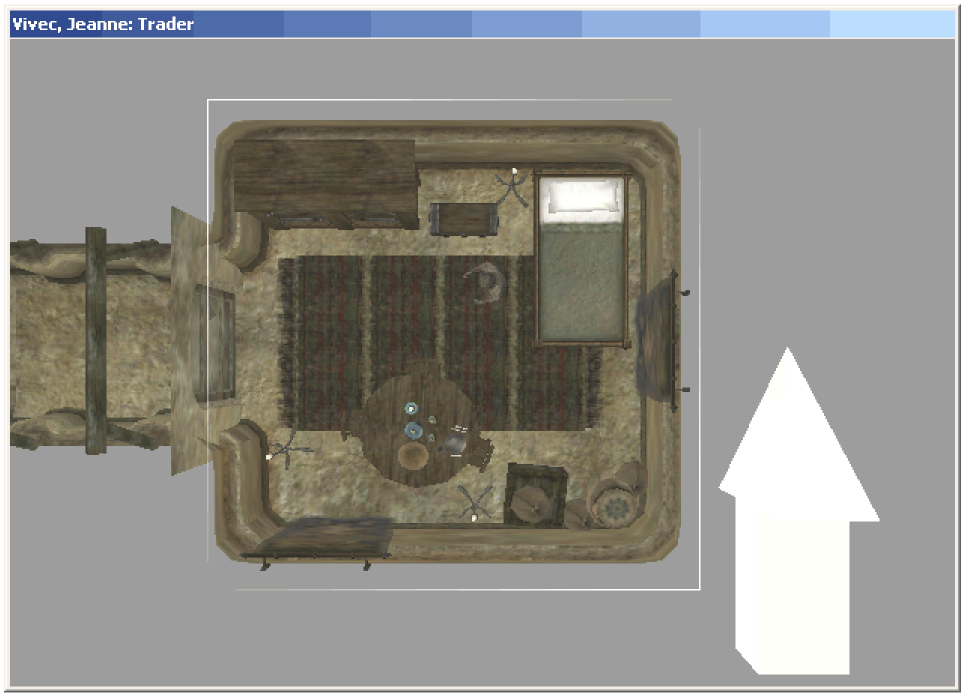
Espionage

In the previous tutorial, we laid down everything in the new room by hand, selecting each object individually, placing it, and detailing by eye. I'm going to introduce an alternative method for much faster room-building now.

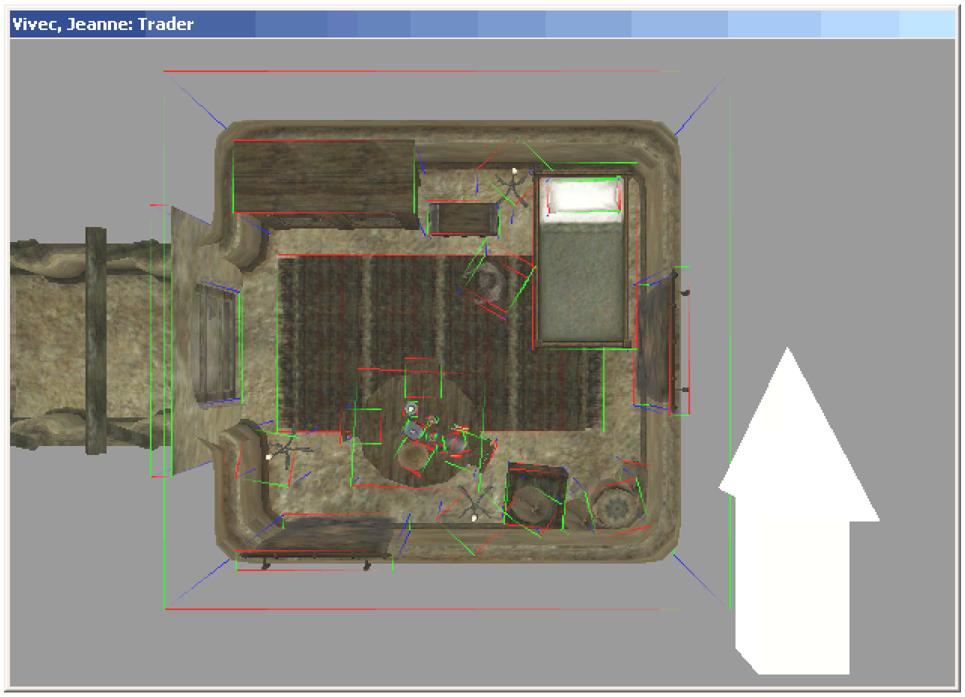
Instead of building everything from scratch, we're going to copy and existing room from somewhere else in the world, paste it into the working cell, and then customize from there. For complex rooms that need lots of objects (furniture) this is a real time-saver in getting started. For quick jobs where you're not necessarily concerned about having a room that looks exactly like someone else's, this saves you the time of having to do everything by yourself.

Since we're working in Vivec, I'm going to stick with the Vivec style, and direct you to a little shop just across the way from your cozy abode, the cell named Vivec, Jeanne: Trader.

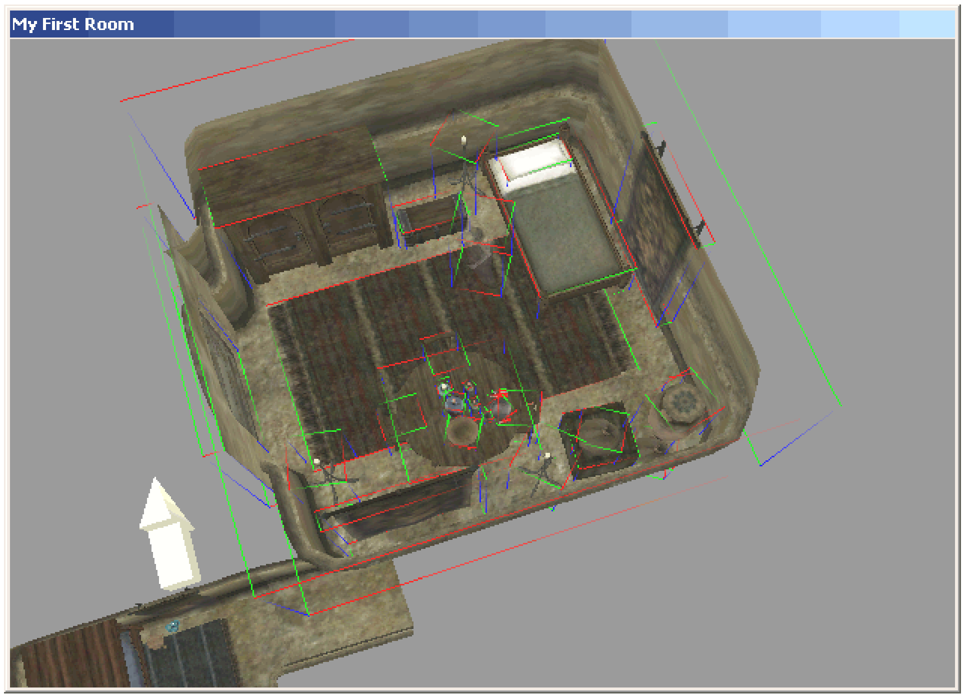
Open up that cell, and move over to a top-down view of the back room. Now use your mouse (left button) to drag a selection box around the entire back room:



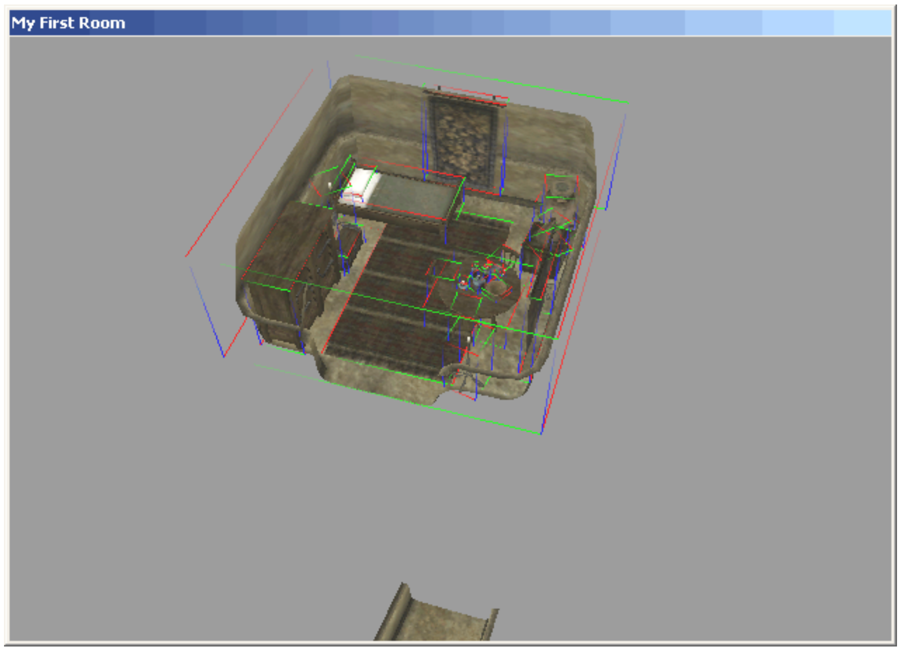
So that you select everything shown:



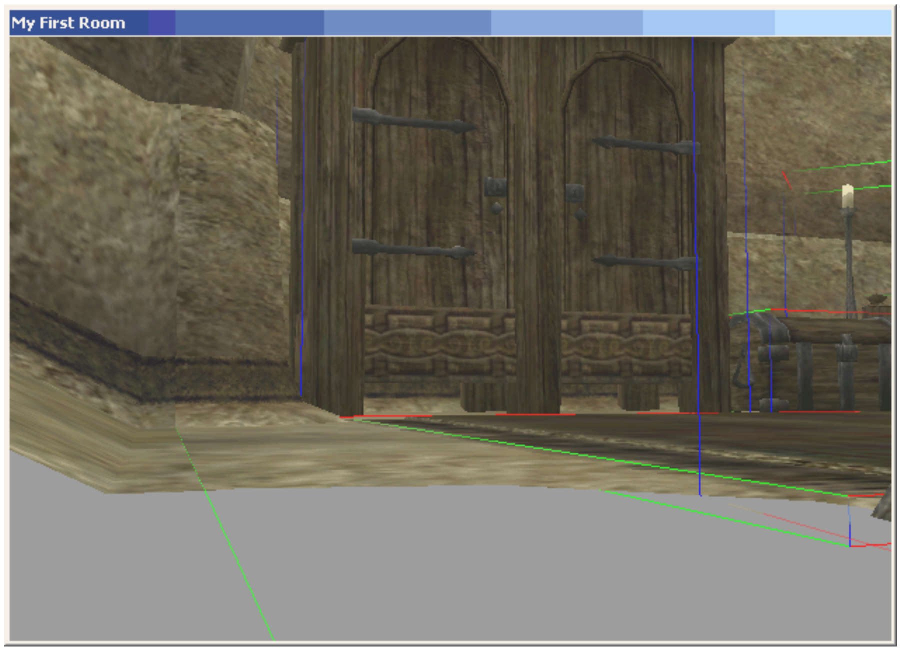
You may grab a few extra objects you don't want, but go ahead and stick with me for now. Hit Ctrl-C to copy all of the objects you have selected. Now open your cell (My First Room), give yourself a top-down view of the empty space near the bottom of the ramp, and hit Ctrl-V to paste in everything.



Let's leave everything up in the air for a moment, and get rid of the objects we don't actually want to keep, before aligning the new room with the ramp. Hit D to deselect everything, first. Start with the door and the door jam - select them each, and hit Delete. Now delete the ghost hangin' out by the bed; we certainly don't want any uninvited guests (least of all some old haunt) stinking up the place. We're going to do some work with the other objects, but now that we've got the architecture as we want (with an open entrance that can be aligned with the ramp) let's get the room set into place. Re-select everything. To make sure you only grab the new room, simply give yourself a camera angle that doesn't include your old room below:



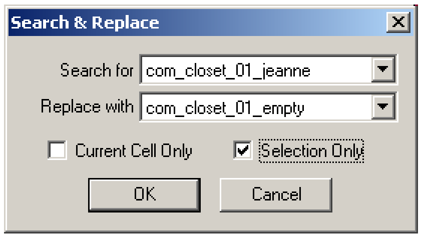
Now, with everything selected, snap the new room into place, aligning with the bottom of the ramp. Do *not* use F to Z-align, since F works on distinct objects within your selection - some objects will fall, but others won't. This part may take some patience, and you might need to set your snap increments down to ridiculously low figures (just makes sure you're always using a power of 2: 16, 8, 4, or 2). Here's the process I used: I roughed everything close to its final position (including Z position), then aligned for X and Y using a snap of 16, then 8. Finally, I aligned for Z, finding I had to go way down to 2 before I got things lined up just right. I stuck with top-down for most of the first part, then used a camera angle really close to the floor helped with the last (Z) part.



You'll probably want to save your ESP at this point, and if you'd like, jump into the game to have a careful look around. Visual defects are generally more obvious in-game than in the editor. Just make sure you carry a bright light with you (or use the Eye of Night enchantment) so as to help you scrutinize. (Note that you may get a warning if you load a Save that was previously using your ESP, before you updated it. Don't worry, it's safe to continue.)

You may notice while you're sniffing at all of your new furniture that the containers are for the most part populated. The containers we sto- er, copied from Jeanne were already set up to have some random contents. Unless that's what you really want, we're going to swap out these containers for their empty equivalents.

This is accomplished simply as a series of Search&Replace edits. Start with the closet, which is at present com_closet_01_jeanne. You're going to replace it with com_closet_01_empty:



Search for: com_closet_01_jeanne
Replace with: com_closet_01_empty
[ ] Current Cell Only
[✓] Selection Only

Repeat the process for the remainder of the containers:

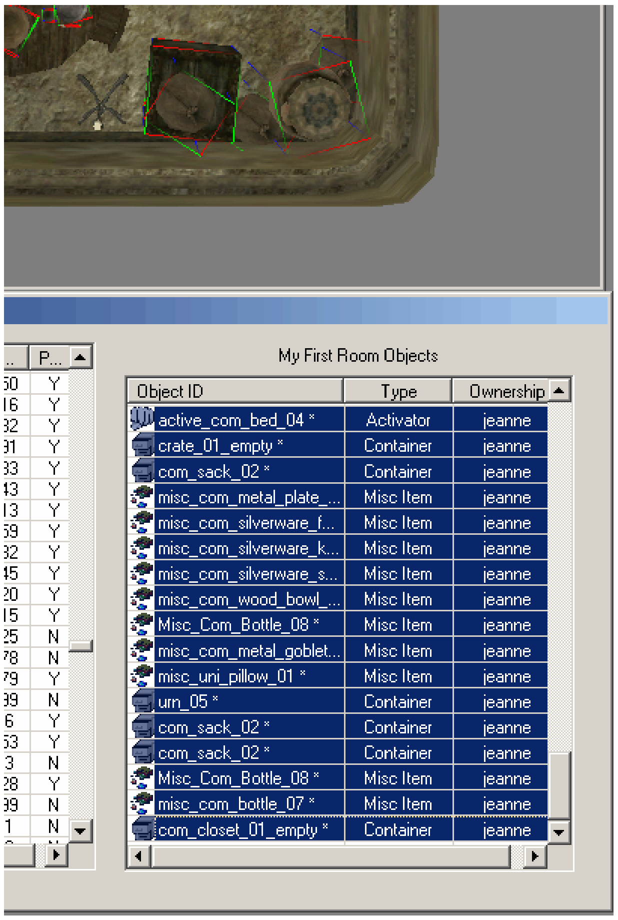
com_chest_02_jeanne_u	com_chest_02_empty
crate_01_random_pos	crate_01_empty
com_sack_02_ingred	com_sack_02_pos
com_sack_02_chpfood3	com_sack_02_pos
urn_05_food	urn_05

Make certain you select the option for "Current Selection Only" when performing your S&R, and of course make certain you've got the container you want to replace currently selected. (The exception is with the com_sack_02_chpfood3 - there are two of them, so you could select both of them, and perform a single S&R on the two of them at once; you'd still use "Current Selection Only", of course. Use the Ctrl key, held down, to select multiple objects one mouse click at a time.)

The beauty of Search&Replace is that the object maintains its position and rotation, so everything is still nicely arranged for you. There's a drawback to this, however. Because the object maintains its "individuality" (its Reference Data, to be precise), it also maintains such information as its "owner". Since we grabbed all of this stuff from Jeanne's place, much of it is still set with Jeanne as the owner.

Next we're going to clear the ownership on all of the objects, which is accomplished with a nifty little trick shown me on the forums. Why is it important? Well, say you just so happened to remove some of these objects and try to sell them to Jeanne, in particular. If she were still the owner of said objects, she would complain about the items being sold her having been hers in the first place, then the guards get involved, things get messy... take it from meow, you just don't want to go there.

Let me start by showing you a separate trick for selecting multiple objects. In your Cell View window, on the right side (all of the objects in your cell), sort the list by Ownership. Now you'll have all of Jeanne's objects lined-up for you. Start at the top, click on the first one. Now scroll to the end of the list of Jeanne's objects, hold down Shift, and click the last one. (You could also use Ctrl to select, of course, one at a time.) You'll fine that all of those objects have become selected, in the Render window.



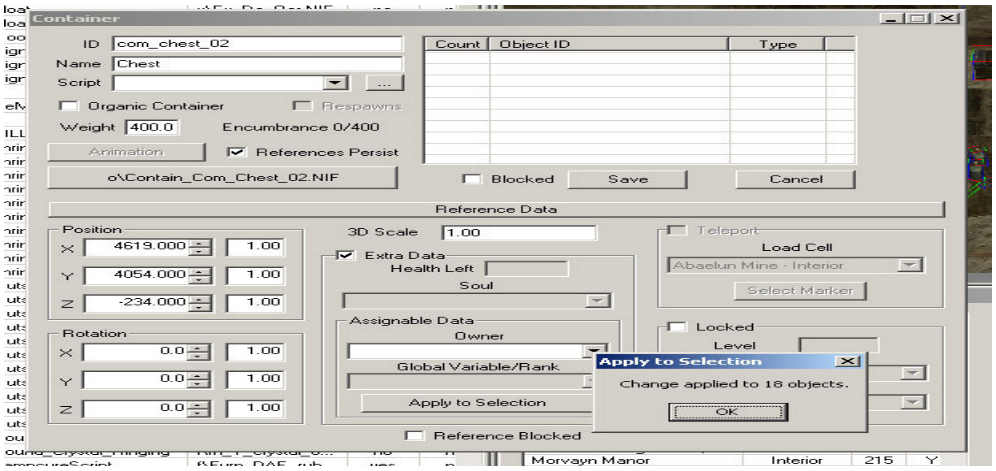
..	P...
50	Y
16	Y
32	Y
31	Y
33	Y
43	Y
13	Y
59	Y
32	Y
45	Y
20	Y
15	Y
25	N
78	N
79	Y
39	N
6	Y
53	Y
3	N
28	Y
39	N
1	N

My First Room Objects

Object ID	Type	Ownership
active_com_bed_04 *	Activator	jeanne
crate_01_empty *	Container	jeanne
com_sack_02 *	Container	jeanne
misc_com_metal_plate_...	Misc Item	jeanne
misc_com_silverware_f...	Misc Item	jeanne
misc_com_silverware_k...	Misc Item	jeanne
misc_com_silverware_s...	Misc Item	jeanne
misc_com_wood_bowl_...	Misc Item	jeanne
Misc_Com_Bottle_08 *	Misc Item	jeanne
misc_com_metal_goblet...	Misc Item	jeanne
misc_uni_pillow_01 *	Misc Item	jeanne
urn_05 *	Container	jeanne
com_sack_02 *	Container	jeanne
com_sack_02 *	Container	jeanne
Misc_Com_Bottle_08 *	Misc Item	jeanne
misc_com_bottle_07 *	Misc Item	jeanne
com_closet_01_empty *	Container	jeanne

In the Render window, double-click on any one of said objects. In the drop-down box for Owner, change the entry to the 'blank' owner (at the very top of the list). You can get there quickly by hitting Home, once the list is open.

Now click the button (immediately below Owner and Global Variable/Rank) 'Apply to Selection'. You should get a confirmation that your change was applied to 18 objects.



Container

ID: com_chest_02
Name: Chest
Script:
Organic Container
Respawns
Weight: 400.0
Encumbrance 0/400
Animation
[✓] References Persist
o\Contain_Com_Chest_02.NIF

Count	Object ID	Type

Blocked
Save
Cancel

Reference Data

Position
X: 4619.000 | 1.00
Y: 4054.000 | 1.00
Z: -234.000 | 1.00

Rotation
X: 0.0 | 1.00
Y: 0.0 | 1.00
Z: 0.0 | 1.00

3D Scale: 1.00
[✓] Extra Data
Health Left
Soul
Assignable Data
Owner
Global Variable/Rank
Apply to Selection
Reference Blocked

Teleport
Load Cell: Abaelun Mine - Interior
Select Marker
Locked
Level

Apply to Selection
Change applied to 18 objects.
OK

Morvayn Manor | Interior | 215 | Y

Click the X in the top-right corner. *Don't click Save.*

You may briefly wonder if you should (or could) set the Owner of these objects to yourself (the player). It's not necessary. Ownership is designed to keep you from selling objects back to their owners and making a killing. Since this problem doesn't really apply in reverse (NPCs swiping your own loot and selling it back to you), there's no need for setting the ownership of an object to you, the player. (Who, by the way, is actually the NPC named "player"!)

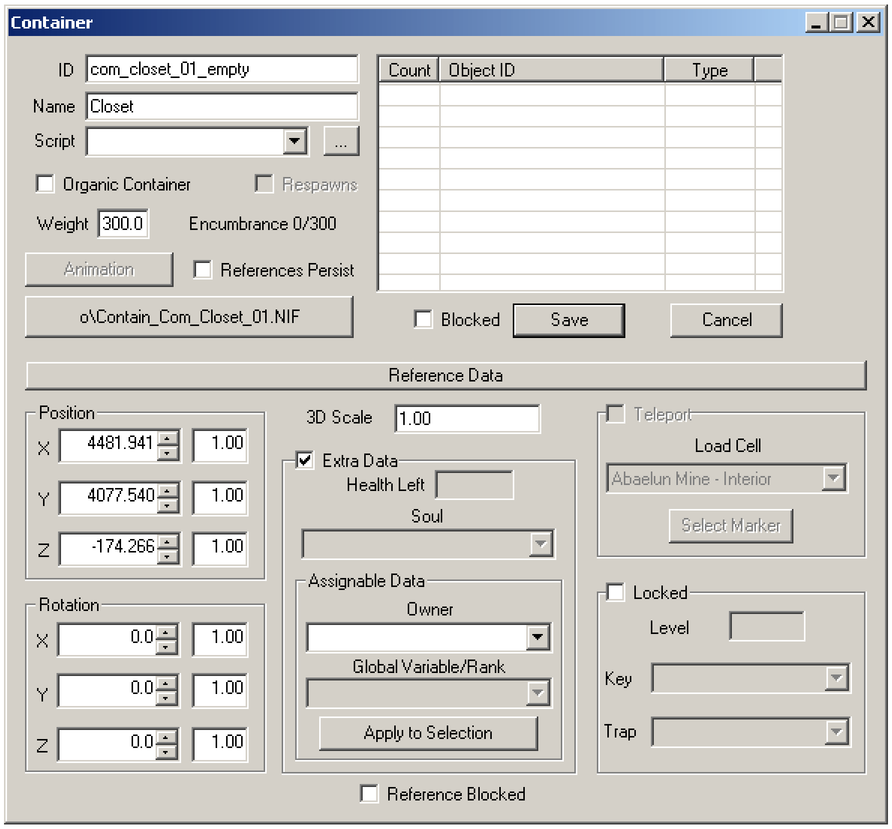
All done? Good - save up your ESP, and take a look around in the game if you like, just to make sure everything's to your satisfaction. If you want to add a few extra details or change the existing ones, go ahead. Just leave the bed alone. We're going to be giving it some "special attention" toward the end of this tutorial.

Multiplicity

Let's take a break from editing for a while and talk about something that's become more and more at issue as you continue to advance in your skills as an "ESPer" (ke ke ke), my young Padawan.

I've been repeatedly telling you not to venture near that Save button, for one thing. You may have heard the use of words like "clean" and "dirty" on the forums, if you hang out there. You may have even peeked at my other tutorial, "Building a Cleaner ESP", which is a boiled-down and high-level version of what we're about to cover here, presently.

The concept of IDs and References is at the heart of understanding why and how a clean mod should be made. Open the properties for the new closet you've got downstairs, the one named com_closet_01_empty. For the first time, we're going to take a really close look at what *all* of those fields mean.



ID: com_closet_01_empty
Name: Closet
Script:
[ ] Organic Container
Weight: 300.0
Encumbrance 0/300
[ ] References Persist
o\Contain_Com_Closet_01.NIF

Position
X 4481.941 1.00
Y 4077.540 1.00
Z -174.266 1.00

Rotation
X 0.0 1.00
Y 0.0 1.00
Z 0.0 1.00

3D Scale: 1.00
[✓] Extra Data
Health Left:
Load Cell: Abaelun Mine - Interior

Starting at the very top of the window, you'll have the object's base type: Container, in this case. This is the tab under which you would find this object, in the Object window.

Immediately beneath, in the top-left region of the window, you'll have the object's *ID information*, which is what *makes this object unique from any other object in the list of that object's type*. The type, remember, is the tab under which the object appears. This object, a container, is "com_closet_01_empty", which is to say that it is unique and distinguished from any other container; for instance, com_closet_01_jeanne, or com_closet_01. They are distinguished *by the ID assigned to each of them*.

They are ***not***, however, unique to your ESP!

I said *unique from any other object in the list of that object's type*. Your ESP does not actually contain any information about "com_closet_01_empty", the ID - rather, it just specifies that a copy of this particular object (ID) happens to exist at position 4481.941, 4077.540, -174.266 (based on my example screenshot). The "copy", or "instance" of that ID, that exists in your ESP, is called a *reference*.

The difference between an ID and a reference can be thought of as the difference between a master template document, and photocopies made from that template, with pertinent information (such as position and ownership) filled in, in the blanks.

That reference information, the "blanks filled in", is what appears below the separator, **Reference Data**. Everything above the separator is ID information - *template* information - while everything below the separator is only specific to the one particular instance you are currently looking at. Reference data can be unique for every copy of an object you have in your ESP. The ID information, on the other hand, *is the same for all copies of that object*.

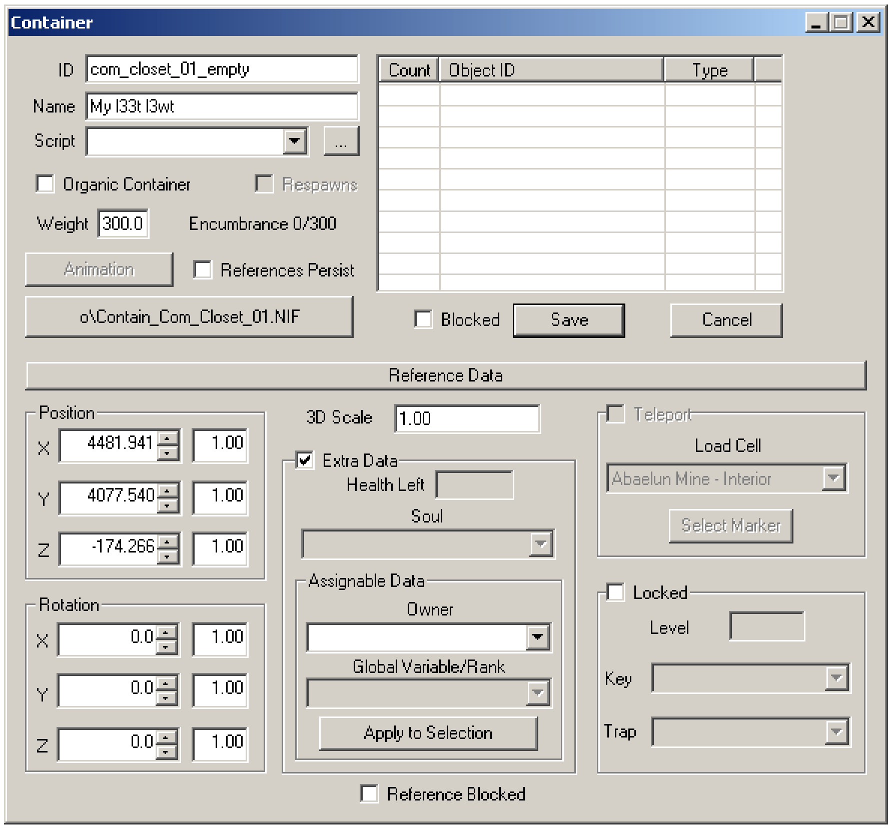
Going back to the ID information at top-left, then, we see that, in the case of a container type object, the ID, Name, Script, Organic Container toggle, Respawns toggle, Weight (limit), Animation, Persist toggle, model (NIF), Blocked toggle, and contents (of the container) are *all* ID information. ***Change any one of these fields, and you have changed that information for all instances of that object, everywhere in the world!***

You make changes to these fields with the Save button. You do *not* need to use the Save button when changing Reference Data. (Just click the X at top right.) If you make changes to the ID information that you decide you do not want to keep, you click Cancel.

Let's introduce a practical example of the difference between ID and reference. **Save your ESP** before you continue - we're going to revert to this save later, since the following changes you're about to make *are changes you do not want to keep,* in your final ESP.

Open up the properties for that closet, if you don't still have them handy. Note how the Name of the container is simply, "Closet". Let's say you got it in your head to change the Name to "My l33t l3wt". Go ahead and plug that into the Name field.

Click the X in the top-right. Open the properties for the closet again, and note how the change you made didn't stick. Change the Name again, *and this time hit Save.* (Then hit the X to close the properties.) Now when you go back to the properties for the closet, you'll see that the change has been made permanent.



Container

ID: com_closet_01_empty
Name: My l33t l3wt
Script:
[ ] Organic Container [ ] Respawns
Weight: 300.0 Encumbrance 0/300
Animation [ ] References Persist
o\Contain_Com_Closet_01.NIF

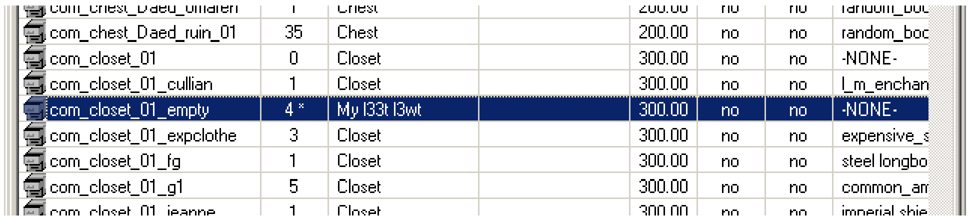
Count	Object ID	Type

[ ] Blocked Save Cancel

Reference Data

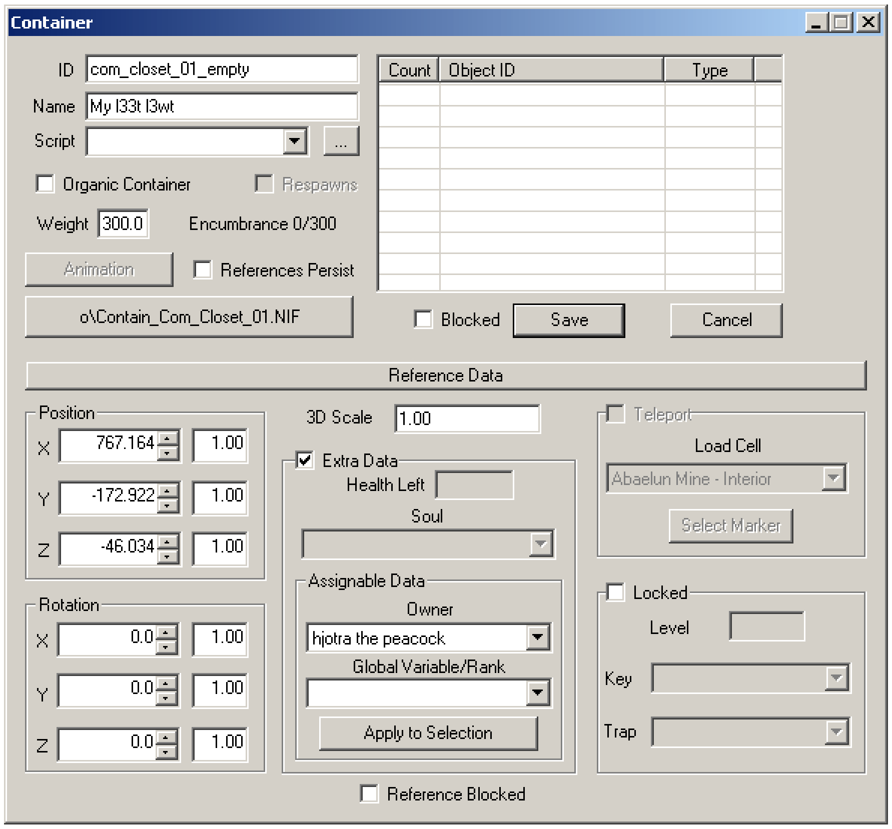
Position: X 4481.941 1.00; Y 4077.540 1.00; Z -174.266 1.00
Rotation: X 0.0 1.00; Y 0.0 1.00; Z 0.0 1.00
3D Scale: 1.00
[✓] Extra Data — Health Left: ; Soul:
Assignable Data — Owner: ; Global Variable/Rank:
Apply to Selection
[ ] Teleport — Load Cell: Abaelun Mine - Interior; Select Marker
[ ] Locked — Level: ; Key: ; Trap:
[ ] Reference Blocked

Now head over to the Object window, and on the Container tab, find com_closet_01_empty. Note how there are *four* references to this object, throughout the world.



com_chest_Daed_ulmaren	1	Chest		200.00	no	no	random_boo
com_chest_Daed_ruin_01	35	Chest		200.00	no	no	random_boc
com_closet_01	0	Closet		300.00	no	no	-NONE-
com_closet_01_cullian	1	Closet		300.00	no	no	l_m_enchan
com_closet_01_empty	4 *	My l33t l3wt		300.00	no	no	-NONE-
com_closet_01_expclothe	3	Closet		300.00	no	no	expensive_s
com_closet_01_fg	1	Closet		300.00	no	no	steel longbo
com_closet_01_g1	5	Closet		300.00	no	no	common_arr
com_closet_01_jeanne	1	Closet		300.00	no	no	imperial shie

Now go to the Edit menu, up top, and run a Find. Search for com_closet_01_empty. You'll get a hit in the cell "Dagon Fel, End of the World Renter Rooms", and you'll be zoomed to that hit. (The view is covered up by a tapestry, you'll need to rotate the camera.)



Container

ID: com_closet_01_empty
Name: My l33t l3wt
Script:
Organic Container: ☐   Respawns: ☐
Weight: 300.0   Encumbrance 0/300
References Persist: ☐
o\Contain_Com_Closet_01.NIF

Count	Object ID	Type

Blocked: ☐

Reference Data

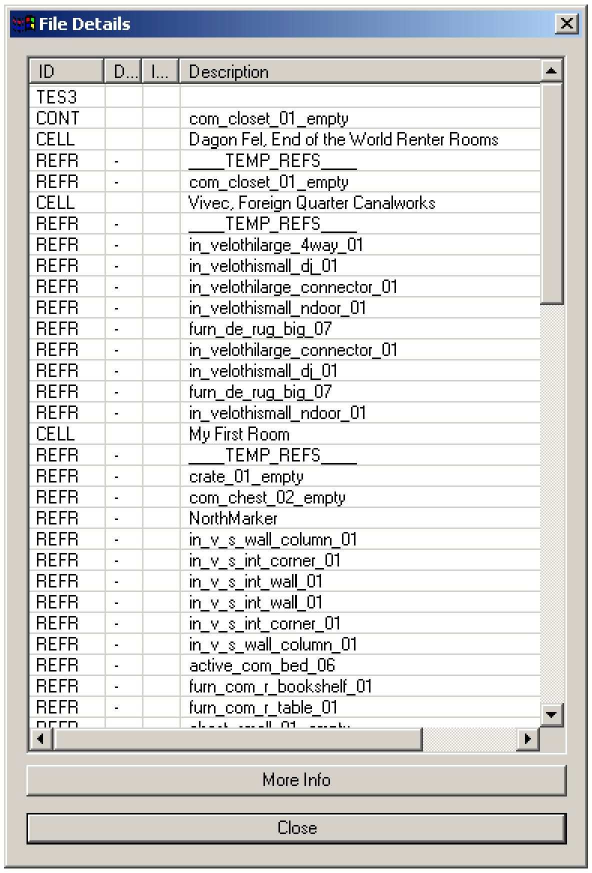
Position		
X	767.164	1.00
Y	-172.922	1.00
Z	-46.034	1.00

Rotation		
X	0.0	1.00
Y	0.0	1.00
Z	0.0	1.00

3D Scale: 1.00
Extra Data: ✓
Health Left:
Soul:
Assignable Data
Owner: hjotra the peacock
Global Variable/Rank:
Teleport: ☐
Load Cell: Abaelun Mine - Interior
Locked: ☐
Level:
Key:
Trap:
Reference Blocked: ☐

Woah there! Some closet all the way in the middle of nowhere has the same name as yours? That's right - because you changed the *ID information*, which is the same for *all* copies (references) of that object, **everywhere!**

Let's take an even closer look at the problem. You'll have to save the ESP you're working on to do so. That means you'll want to rename the ESP you'd saved first thing - rename it to "MyCleanRoom.ESP", if you like. (You'll use the usual means - Windows Explorer - to do so.) Now save your current working ESP, which will be the new "MyFirstRoom.ESP". Finally, let's head back to our old friend "Details", under File, Data Files - for the dirty ESP, MyFirstRoom.



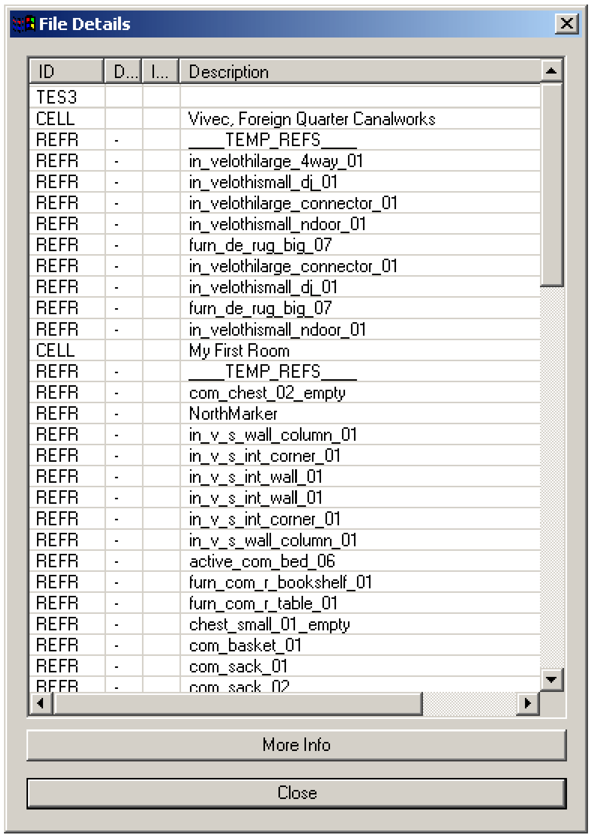
File Details

ID	D...	I...	Description
TES3			
CONT			com_closet_01_empty
CELL			Dagon Fel, End of the World Renter Rooms
REFR	-		____TEMP_REFS____
REFR	-		com_closet_01_empty
CELL			Vivec, Foreign Quarter Canalworks
REFR	-		____TEMP_REFS____
REFR	-		in_velothilarge_4way_01
REFR	-		in_velothismall_dj_01
REFR	-		in_velothilarge_connector_01
REFR	-		in_velothismall_ndoor_01
REFR	-		furn_de_rug_big_07
REFR	-		in_velothilarge_connector_01
REFR	-		in_velothismall_dj_01
REFR	-		furn_de_rug_big_07
REFR	-		in_velothismall_ndoor_01
CELL			My First Room
REFR	-		____TEMP_REFS____
REFR	-		crate_01_empty
REFR	-		com_chest_02_empty
REFR	-		NorthMarker
REFR	-		in_v_s_wall_column_01
REFR	-		in_v_s_int_corner_01
REFR	-		in_v_s_int_wall_01
REFR	-		in_v_s_int_wall_01
REFR	-		in_v_s_int_corner_01
REFR	-		in_v_s_wall_column_01
REFR	-		active_com_bed_06
REFR	-		furn_com_r_bookshelf_01
REFR	-		furn_com_r_table_01

More Info

Close

Compare this with the details for your clean copy, MyCleanRoom:



File Details

ID	D...	I...	Description
TES3			
CELL			Vivec, Foreign Quarter Canalworks
REFR	-		____TEMP_REFS____
REFR	-		in_velothilarge_4way_01
REFR	-		in_velothismall_dj_01
REFR	-		in_velothilarge_connector_01
REFR	-		in_velothismall_ndoor_01
REFR	-		furn_de_rug_big_07
REFR	-		in_velothilarge_connector_01
REFR	-		in_velothismall_dj_01
REFR	-		furn_de_rug_big_07
REFR	-		in_velothismall_ndoor_01
CELL			My First Room
REFR	-		____TEMP_REFS____
REFR	-		com_chest_02_empty
REFR	-		NorthMarker
REFR	-		in_v_s_wall_column_01
REFR	-		in_v_s_int_corner_01
REFR	-		in_v_s_int_wall_01
REFR	-		in_v_s_int_wall_01
REFR	-		in_v_s_int_corner_01
REFR	-		in_v_s_wall_column_01
REFR	-		active_com_bed_06
REFR	-		furn_com_r_bookshelf_01
REFR	-		furn_com_r_table_01
REFR	-		chest_small_01_empty
REFR	-		com_basket_01
REFR	-		com_sack_01
REFR	-		com_sack_02

More Info

Close

You may recall performing a similar exercise in the first tutorial. Back then, I told you that you'd probably want to start over to get rid of the "unclean" entries - in this case, com_closet_01_empty. At this point, though, starting over from scratch is probably not the first thing that would come to your mind.

Young Padawan is *probably* thinking, instead, "No, look, I'll just go back to the object and change the name back to how it's supposed to be." And young Padawan would find, if she chose to do so, that this would not in fact fix the problem - at all. The Details for your dirty ESP will still list the extra, "changed" copy of com_closet_01_empty. That's because *two wrongs don't make a right*; the editor doesn't understand that you're trying to undo your changes to the ID information. In fact, there is no easy way to revert an object back to its original state. Instead, you have any one of the following not-so-easy alternatives:

1. You can start over.

2. You can be more careful in the future not to make changes to ID information. The simplest way to make sure you never do is to develop the habit of not using the Save button, *unless you know what you're doing* (and we're getting there).

3. You can use a tool like TESAME, found at http://sphosting.com/scarabus/tesame.html (this link may become outdated). TESAME is a utility that opens your ESP and gives you fine control over all of the objects saved in it - so you could, for instance, remove the "dirty" com_closet_empty_01 and the cells which were added to your ESP so as to reference the dirty copy. This is a godsend for advanced modders that make a mistake or two and are prepared to make amends, but it does *not* give you license to be lazy, and I don't provide any instruction for how to use TESAME, here.

To get back to the original problem, though - you want the closet *in your room* to be named 'My l33t l3wt'. You don't want to affect any of the closets around the rest of Vvardenfell, you don't want it to be a "dirty" closet - shouldn't something so innocent as changing the name of just one object have a simple solution?

Well, in fact grasshopper, it does. The solution is to create your own unique object, and give it whatever name you please.

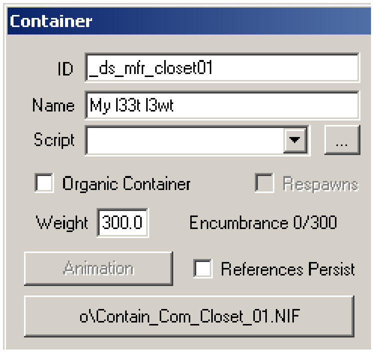
My First Closet

Let's start by getting back to your "clean" copy of the ESP. Close TESCS, and delete the dirty ESP, which should be MyFirstRoom. Then rename MyCleanRoom back to MyFirstRoom. Open TESCS again, set your data files and active file, and head back to your cell. In your Cell View, double-click com_closet_01_empty, and open its properties.

Now change the Name to 'My l33t l3wt', as before, and then change the ID to something of your own choosing, as well. If you want to use SANC (the Safe Accessible Naming Convention, a pet project of mine) for naming your object, name it something along the lines of,

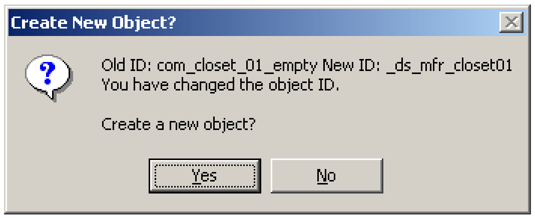
_ds_mfr_closet01

Replace 'ds' with your call-sign (your unique identifier, something like your initials), 'mfr' with the name of your mod (I've just abbreviated 'My First Room' here, so you could probably leave this alone), and 'closet01' with whatever unique ID you'd prefer this object have. ('closet01' is fine by itself, actually.) The underscores are used for separation - and the first underscore to make the object sort to the beginning of a list, to make it easier to find. You don't *have* to use SANC, and SANC is still a young, in-development standard (so it's not really a standard yet), but the point is to try and name your IDs something that no one else would ever name their IDs, in different ESPs. (Otherwise, the two objects will conflict, and only one of them will actually show up in the running game.)


Container

ID: _ds_mfr_closet01
Name: My l33t l3wt
Script:
Organic Container: ☐
Respawns: ☐
Weight: 300.0
Encumbrance 0/300
References Persist: ☐

Now hit Save. The one time you *will* want to use Save is when you're creating a unique object. You've made this object unique by giving it a unique ID. TESCS isn't sure if you want to create a unique object or just change the ID of the existing object, though, so you'll be prompted:

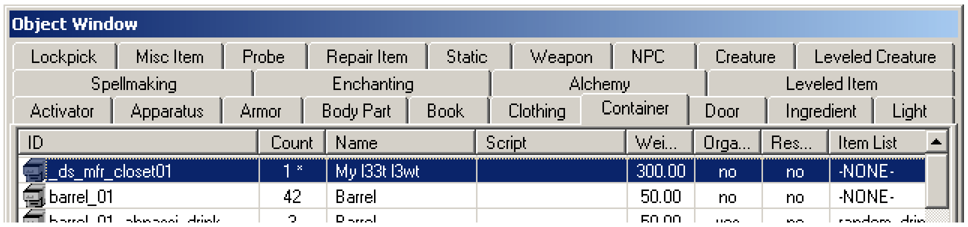

Create New Object?

Old ID: com_closet_01_empty New ID: _ds_mfr_closet01
You have changed the object ID.

Create a new object?

Click Yes.

And now you've created your very own closet, to do with however you please.

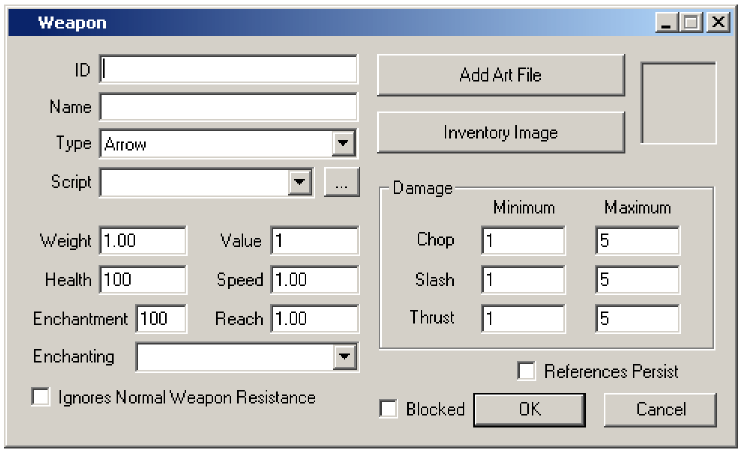


ID	Count	Name	Script	Wei...	Orga...	Res...	Item List
_ds_mfr_closet01	1 *	My l33t l3wt		300.00	no	no	-NONE-
barrel_01	42	Barrel		50.00	no	no	-NONE-
barrel_01_abnassi_drink	3	Barrel		50.00	yes	no	random_drin

But creating a unique *closet* isn't really that exciting, is it? Let's create something a little more befitting the lifestyle of a Master Thief with her own crib snug down in the canalworks of Vivec.

How about a signature dagger with a few custom effects, maybe even its own custom look?

Hit the Weapon tab in the Object window. Right-click anywhere on the list, and choose New.



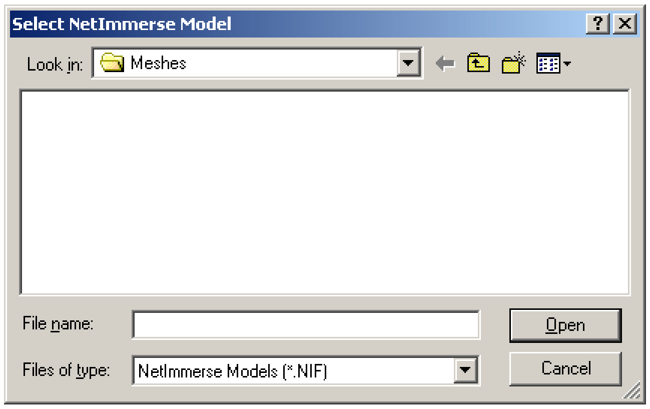
ID	
Name	
Type	Arrow
Script	
Weight	1.00
Value	1
Health	100
Speed	1.00
Enchantment	100
Reach	1.00
Enchanting	
Ignores Normal Weapon Resistance	☐
References Persist	☐
Blocked	☐

Damage	Minimum	Maximum
Chop	1	5
Slash	1	5
Thrust	1	5

For the ID, I'm sticking with the SANC rules, and naming it "_ds_mfr_spiderdagger". The Name itself is, "The Spider Dagger". And that's *The* Spider Dagger, cuz' there's only one! (And it's *all mine!*) The type of weapon is ShortBladeOneHand, and for now, there's no Script attached to it. I leave the Weight at 1.00 (its excellent craftsmanship causes it to be lightweight - not any added enchantments), set the Value to 300, the Health to 200 (due to its spindly craftsmanship, it is unfortunately quite delicate), and the Speed to 1.50 (faster than any other dagger from the game). I'm not quite sure what the Enchantment value should be just yet, so I leave it at the default 100. The Reach is 1, same as any other short, one-handed weapon.

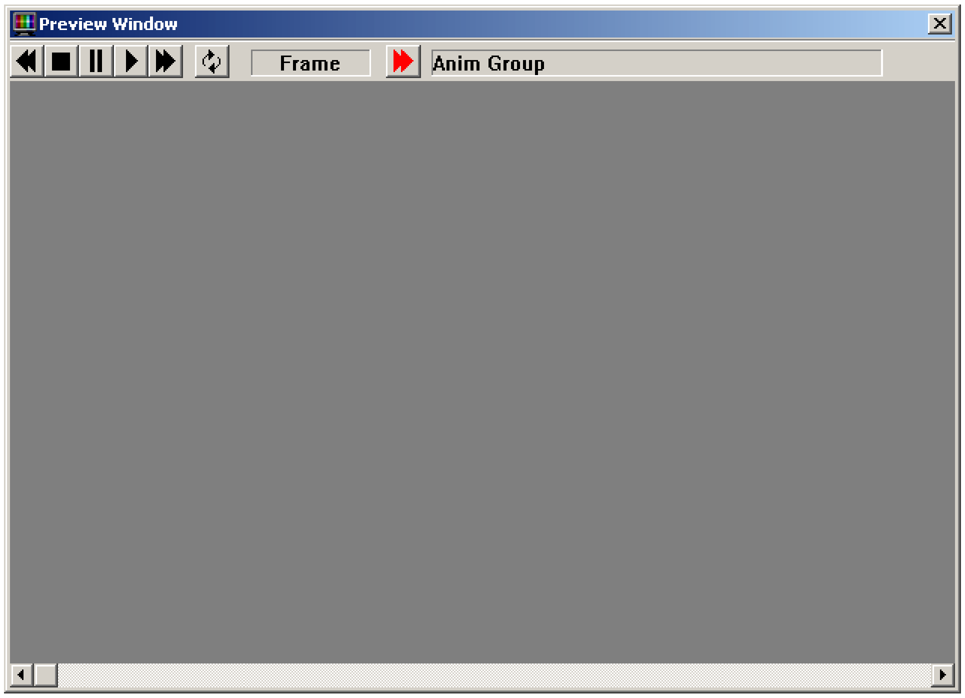
Now for the damage values! I want the dagger to be fairly insidious on its own, without the help of extra enchantments (though I do plan on giving it two or three), so I set them to 3-4, 4-5, 9-10. These aren't incredible, but then I'm not designing an uber-slayer here, I do want something realistic - the point is to make a signature item, not to unbalance the game for myself.

I tick off 'Ignores Normal Weapon Resistance' (it can be used against certain magical creatures upon which normal weapons are ineffectual), and then head for the two buttons to set the weapon's model (Art File) and icon (Inventory Image).

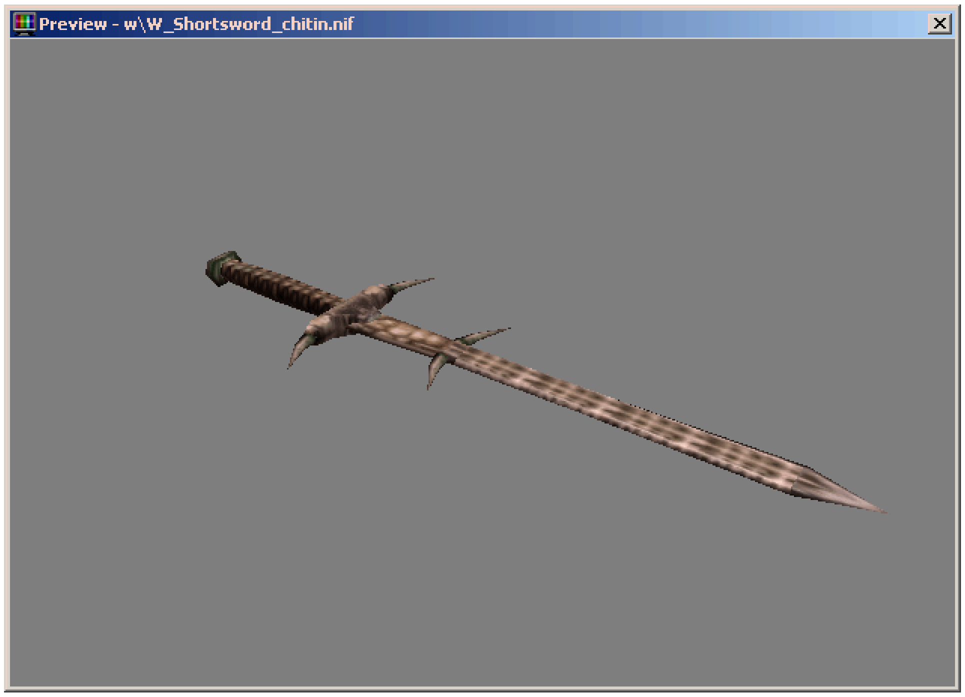


Hmm. Well, this isn't so promising. But then I remember I don't have the Construction Set CD in my drive. Remember way back when I said you don't need the CS CD to run the editor? You don't - not until you're creating custom items, and you need to pick models and icons for them. Get your CS CD loaded, wait for it to spin up, and come back to this dialogue (Add Art File). Now browse to the CD, open the Meshes directory, and here you'll find all of the NIFs you have to choose from. Well, shoot, I have to Cancel again, because I don't know which NIF I want my dagger to have the look of.

There's an easy way to preview NIFs without having to drag each and every dagger into your room, and that's the Preview window, which until now you've not even had open. You access it from the View menu, up top. It will replace your Render window, while active.



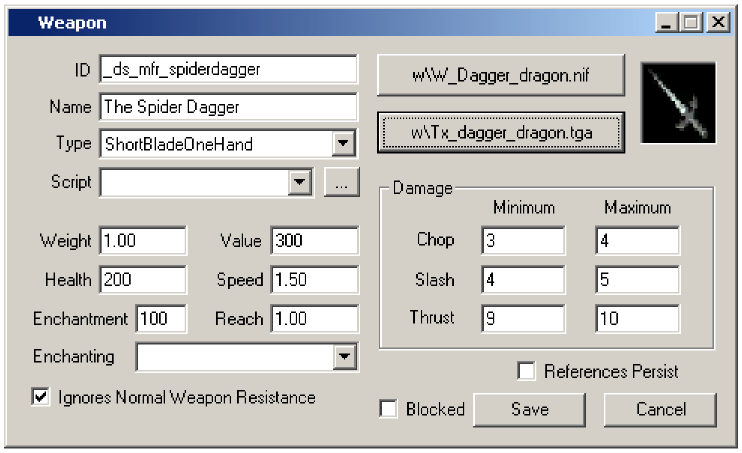
There are two ways to operate the Preview window. You can right-click and Load Model, and specify exactly which NIF you want to look at, or just select an object in the Object window, and its NIF will be shown for you. Click the 'chitin dagger', for instance. You'll start with a top-down view of the object. You won't use your mouse to move it around here, though, you'll instead use keys on the numpad (with Numlock turned off) to roll and turn the object. You'll also find that the view will "grey out" when you switch applications, but by giving the object a little nudge (rotating it in any direction on the keypad) it comes back into view.



I browse through a few of the daggers, window shopping for the perfect look. Of course, every time I hit a new one in the Object window, I have to click the Preview window again to make it active, before my numpad keys work. I decide on the model used for the steel dagger (and numerous 'vipers'), and take note of the path and name at the top of the Preview window: w/W_Dagger_dragon.nif.

Now I return to the Render window, by going back to View, and selecting Render Window. I find The Spider Dagger on the object list, open its properties, and for the Art File, I browse on the CD to the Meshes directory, the w directory, and select W_Dagger_dragon. I take a quick look at the Inventory Image the steel dagger uses, w\Tx_dagger_dragon.tga, and use that for the Inventory Image for my own dagger, too. (This time, it's under the Icons directory, not the Meshes directory.)

Et voila!


Weapon

ID _ds_mfr_spiderdagger
Name The Spider Dagger
Type ShortBladeOneHand
Script

w\W_Dagger_dragon.nif
w\Tx_dagger_dragon.tga

Weight	1.00	Value	300
Health	200	Speed	1.50
Enchantment	100	Reach	1.00

Enchanting

Damage

	Minimum	Maximum
Chop	3	4
Slash	4	5
Thrust	9	10

[✓] Ignores Normal Weapon Resistance
[ ] References Persist
[ ] Blocked

In the next section, I'm going to go over how to give the dagger its own customized appearance, by 're-skinning' it. This makes use of a photo editing program, and a hex editor. It's not incredibly involved, but if you don't feel the need to go there just yet, or you don't have the programs you'd need handy, you can just skip past it, to Extra Oomph.

The Spider Dagger

Let me start with a brief overview of the format of a NIF file - a NetImmerse Format file. NIF is a proprietary format which is not commercially availble: that means you can't "Save As" or "Open" a NIF using any known modeling application. You won't be able to use 3DSMax to open NIFs. There is an "Exporter" available that allows you to build a completely new and original model, and Export it to NIF, but that is beyond the scope of this tutorial.

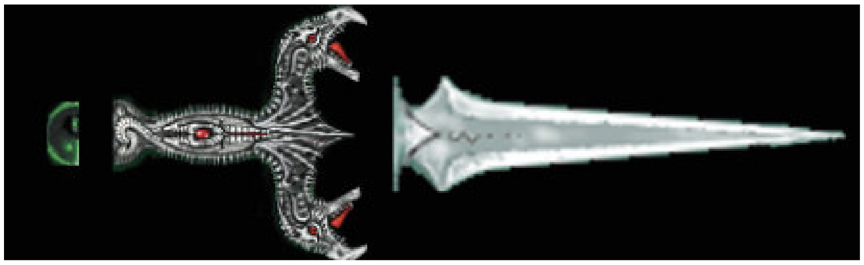
A NIF file consists largely of unencrypted data. That means you can open a NIF in a hex editor, and read much of the information as "plain text". Were you a guru with a hex editor, you may even be able to modify the actual mesh (shape) of an object (NIF) with just a hex editor. I am certainly no kind of guru with a hex editor.

What I'm leading up to, here, is that, casually, you will probably not be able to look forward to designing your own custom *shapes* for your items. Not unless you're already pretty good at modeling and texturing, and have access to and experience with 3DSMax (or, purportedly, Maya). What you *can* do, more realistically, is "re-skin" existing objects, to give them a signature look and feel. In my case, I'm going to re-skin the steel dagger model for my custom Spider Dagger. I'm going to take the bright steely appearance of the steel dagger, and darken it down to a matte quality, something more befitting an expert in the field of stealth and, erm, "troubleshooting".

You'll need two applications to get working with re-skinning. One for photo editing - to tweak the actual appearance of the texture files that are applied onto the mesh of the object. One for hex editing, to tweak at the NIF file itself, to point to your new modified textures, instead of the original textures. Currently, hex editing is the only way to re-skin a NIF. However, it's feasible that at some future date, long after this tutorial is written, there will be utilities available that expedite this process for you. For our purposes, I'm going to show you how to do it the "old-fashioned" way, though.

Textures for objects are saved in the BMP file format. Pretty standard, so just about every photo editor available will allow you to work with them. Because textures are generally being applied to fairly small objects, a scrutinous level of detail really isn't necessary, so you shouldn't be concerned if you don't have a really expensive, feature-rich photo editor. Even a freeware editor will probably suit the extent of your re-skinning needs. (And, anyhow, it's the quality of the *artist* using the editor that really makes the difference in the end.)

To get started, you're going to need to know what files you're going to work with, right? Before we get there just yet, though, let me point out one last property of NIF files: they can have multiple "parts". The steel dagger, for instance, is comprised of three parts: the pommel (the jewel on the butt-end), the hilt (being the grip, and the gargoyle-design on the crossbar), and the blade itself. Each part has its own individual texture. For a complete re-skin of the steel dagger, then, you'll be editing three separate BMPs, roughly previewed below:



Now - which BMPs are they? Where are they? To answer that question, you're going to need the hex editor before you actually use the photo editor. If you've never used a hex editor before, and you're hesitant, let me assure you that basic hex editing is really not that difficult. Certainly no more difficult, analytically, than using a photo editor. Let me qualify that, though: a *good* (easy to use) hex editor makes hex editing easy. A poor hex editor makes hex editing a nightmare.

I'm going to direct you to my own personal preference in hex editors, though you're free to use any you like. If you're already familiar with a particular hex editor, then by all means use that one in place of mine. Otherwise, allow me to point you to Hex Workshop, available from BreakPoint software, www.bpsoft.com. I'm not going to give you a full dissertation on the use and benefits of Hex Workshop. I will say that it's not free, and your use of it is limited to a 30 day trial period. Afterwards, I would encourage you, if you find you enjoy and make frequent use of the application, to register it, for $50 (single license). The unregistered version is unrestricted. So downloading it for just this tutorial won't hurt a bit.

I'm going to describe how you would go about the particulars of hex editing for re-skinning using specific commands and screenshots from Hex Workshop. But since you may be using a different hex editor (which you should already be quite familiar with), I'll try and be generic where applicable, so that you can still follow along.

Firstly, you'll need to make a copy of the NIF you're re-skinning, and it needs to be located in a rather particular place: putting files in the correct locations is vital for a re-skin to work! But the logic of it is simple: just make sure your files are in *the same place* on your local installation of Morrowind, as they are in the corresponding directory structure on the Construction Set CD.

If you browse the CS CD, you'll see that there's a Data Files directory in the root. You also have a Data Files directory in your Morrowind installation. That's where your ESP files go, for one. You'll also have several directories branching off of Data Files: like meshes, textures, icons, etc. You'll also have the same directories in your own Data Files directory, in your installation. You'll notice that those directories are empty on your hard drive, though - whereas they're populated on the CD. There are also more levels of directories on the CD - like "w" under the meshes directory.

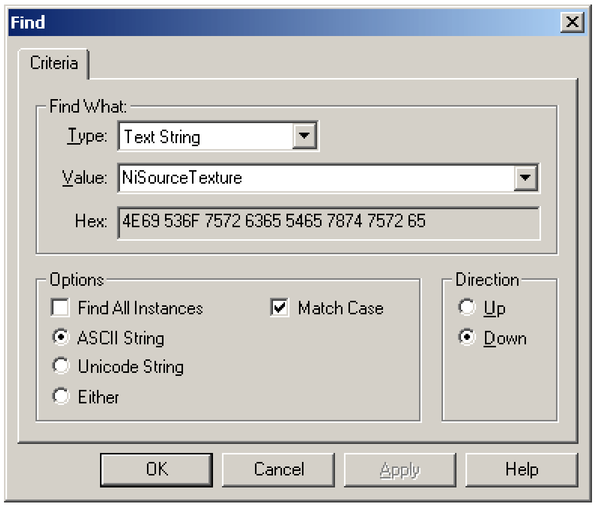
So, when you copy a NIF from the CD to your hard drive, just make sure that its location on your hard drive mirrors its location on the CD. In this case, we're copying meshes\w\W_Dagger_dragon.nif. So, from your own installation of Morrowind, open the Data Files directory, then the meshes directory. Create a folder named 'w'. In this folder, place a copy of W_Dagger_dragon.nif. You may rename it to whatever you please - I'm sticking with SANC, and naming mine _ds_mfr_spiderdagger.nif. (Other tutorials have specified that you shouldn't mess with the name. I've found that you can name the file however you like, it doesn't really matter, except for keeping with the existing convention, or using a different convention - like SANC.)

Finally, the file will often have its "Read Only" attribute set after being copied from CD (which is Read Only), so you may need to open the file's properties (in Windows Explorer) and turn the Read Only bit off.

We needed to find what texture files we need to work with. This is a simple matter of running a Search through the NIF which is being re-skinned, looking for the particular text, "NiSourceTexture".

Open up Hex Workshop, and go to File, Open. Browse to your NIF, and open it. Your screen will be filled with lots and lots of numbers, more numbers, and on the right side, what looks like text mixed up with all sorts of other strange miscellaneous characters. Don't be overwhelmed, I'll walk you right through what you need to do.

Go to Edit, and choose Find. Set the Type to Text String, and enter 'NiSourceTexture' in the Value field. Leave the rest of the options at default.

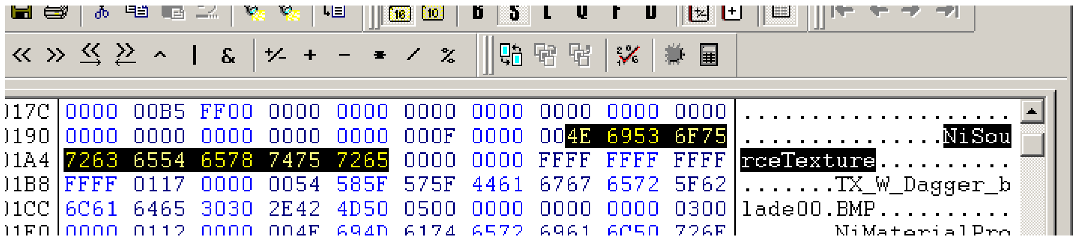


Find What:
Type: Text String
Value: NiSourceTexture
Hex: 4E69 536F 7572 6365 5465 7874 7572 65

Options
Find All Instances
[✓ Match Case]
[✓ ASCII String]
Unicode String
Either

Direction
Up
[✓ Down]

Hit OK to find the first match.



```
17C  0000 00B5 FF00 0000 0000 0000 0000 0000 0000 0000  ...................
190  0000 0000 0000 0000 0000 000F 0000 004E 6953 6F75  ..............NiSou
1A4  7263 6554 6578 7475 7265 0000 0000 FFFF FFFF FFFF  rceTexture.........
1B8  FFFF 0117 0000 0054 585F 575F 4461 6767 6572 5F62  .......TX_W_Dagger_b
1CC  6C61 6465 3030 2E42 4D50 0500 0000 0000 0000 0300  lade00.BMP.........
```

For now, just pay attention to the information in the window on the right. (What's on the left is the hex equivalent of what's on the right - not that useful to us, right now.)

Going down (right to left, top to bottom) from where NiSourceTexture was found, look for the next text string - it should have '.BMP' on the end of it. This is one of the textures that the NIF uses. In this case, the first texture used is "TX_W_Dagger_blade00.BMP". Write this down.

Hit F3 to run the previous search again. You'll get another hit, and another texture, this time "TX_W_Dagger_jewel00.BMP". Hit F3 again, for one more hit, "TX_W_dagger_hilt00.BMP". An F3 after that will give you a message indicating Hex Workshop searched to the end of the file (from your current location) without finding any more matches. (You can hit Cancel to this message.) We've found three textures that this NIF uses. Those are the three you will need to edit, in order to re-skin this NIF.

Now where are those textures? Well, they're in the "Textures" directory, on the CS CD, of course. (Textures being under Data Files, that is.) And in order to make unique versions, you'll need to copy them into your own Textures directory. (Again, you may also need to toggle the Read Only bit on each of these files.)

Now go ahead and open each (or all) in your photo editor. I won't recommend any particular photo editor, since everyone seems to have their own personal preference. But I hear a lot of good things about PaintShop, if you have no idea which one to pick: www.jasc.com for that one. I'm also not going to guide you through exactly how to photo edit these, either - since that's certainly well beyond the scope of this tutorial! Try a few things and get the pictures to look how you like. This may be something as simple as adjusting the coloration, giving the dagger a gold or (in my case) black look. You may go all out and change the entire appearance from scratch! There's only one rule: the "shape" (outline) of the skin can't change, since the underlying model (mesh) for the object itself won't change. You can't map a square picture onto a dagger-shaped model - it just won't look right. Sorry, this is one time you're not allowed to draw outside of the lines.

I saved each BMP with its own unique file name, as well, and I've given you links to each of them, below. You can use mine, if you prefer, or if you're stumped with your own.

Pommel
Blade
Hilt

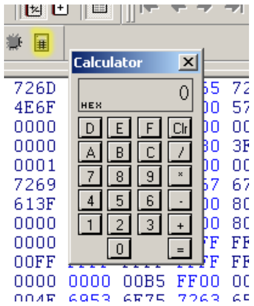
Now for the really terse part - hex editing the NIF to point to these new textures of yours, instead of the three texture names you wrote down previously.

Hope back into Hex Workshop, and go back to the top of the file (the Home key is a shortcut for doing so). Hit F3 again to find that first match, which should still read, "TX_W_Dagger_blade00.BMP". This is where things get tricky. You see, you can't just delete what's there and plug in what you've got. The NIF file format includes information for *exactly how long the text string* is, that names each BMP. In this case, the string "TX_W_Dagger_blade00.BMP" is 23 characters (bytes) long, and the NIF file says so. It says, "For the next 23 bytes, expect the name of this NiSourceTexture - no more, no less!"

So you have to change that 'bit' that indicates how many bytes (characters) long your file name is, as well as specifying your filename. You could get around that by trying to keep your files inside the exact same number of characters as the original BMPs, or you may even be tempted toward the Dark Side, my young Padawan, and think not to change the original file names at all! *Don't go there!* You will have succumbed to the influences of a *dirty* mod, and the last thing you or the community wants floating around out there are *dirty* mods, right?

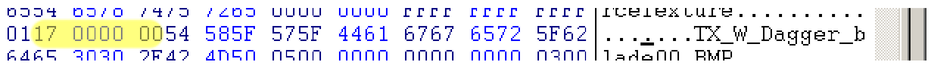
Really, this won't hurt - that much. You just need to count how many characters are in the name of your files, including the dot for the extension, and including the extension. In the case of my first file, "_ds_mfr_spiderdagger_blade00.bmp" (which is the replacement for "TX_W_Dagger_blade00.BMP"), I have 32 characters. Go ahead and count them for yourself, to make sure you see what I mean.

Now you need to convert that number into hex, since you're editing in hex, after all. What exactly is "hex"? Well, it's just a different numbering system, which is base 16, instead of base 10 (decimal). But before I get off on a tangent about the symbology of numbering systems, let me just point you to the built-in calculator Hex Workshop provides:



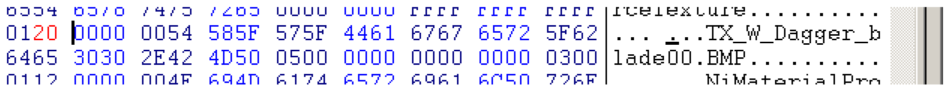
Click the button I've highlighted for you in the image. Now click on the word 'HEX' in the top display, to change it to 'DEC'. Now type in the number '32'. Click 'DEC' again to change back to 'HEX', and your decimal number, 32, will be converted into its hexidecimal equivalent, '20'. Most standard calculators allow you to perform this conversion, just in case your hex editor doesn't include this functionality - even Windows Calc.

So now I know that I need to change "TX_W_Dagger_blade00.BMP" to read "_ds_mfr_spiderdagger_blade00.bmp", and somewhere else I need to tell the NIF to expect 32 bytes, instead of 23. That 'somewhere else' is just a few bytes back from the file name - exactly four bytes back, to be precise:



```
0117 0000 0054 585F 575F 4461 6767 6572 5F62 |.......TX_W_Dagger_b
```

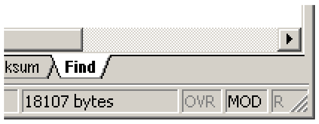
I've highlighted all four bytes, because you can specify any length up to four bytes long - which could be a pretty big number. You'll generally only need to worry about setting the first 'byte' of those four bytes, though - the one that says '17' in the screenshot. '17' is 23, in hexidecimal. So I set my cursor in the left window, right where it says 17, and I type in '20':



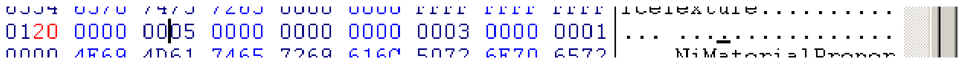
```
0120 0000 0054 585F 575F 4461 6767 6572 5F62 |... ...TX_W_Dagger_b
6465 3030 2E42 4D50 0500 0000 0000 0000 0300 |lade00.BMP..........
```

The change I made *overwrote* the 17 that was there before, but I'm actually assuming being in 'overwrite' mode, which is the default mode for Hex Workshop. If the change you made just added '20' in front of the '17', then delete the 17. You don't want it. You want your len to read '20 00 00 00', no more, no less!

For the next operation, though, you *don't* want to be in overwrite mode. You toggle overwrite mode with the Ins (Insert) key, and an indicator, "OVR", in the bottom-right of Hex Workshop will light up or grey out to tell you which you're currently in:



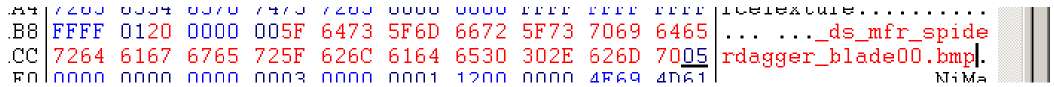
Now I move over to the file name, and use Delete to remove what's currently there. *Just the 23 characters that specify the file name! No less, no more!* (Make sure you don't accidentally delete the other three 'bytes' of 0s after your '20', and certainly not any of the bytes after the end of the file name!)



```
0120 0000 0005 0000 0000 0000 0003 0000 0001 |... ...-.............
```

Note carefully my cursor position, and the bytes on the left and the right.

Now, to enter in text data, I move back to the right pane. Just click behind where you already see your cursor (the underline) and make certain your position remains the same, as that shown in the above screenshot. I still have '20 00 00 00' immediately to my left (with '05' underlined in the left pane). Then I just type in the name of my file, all 32 characters of it:



```
.B8 |FFFF 0120 0000 005F 6473 5F6D 6672 5F73 7069 6465 |... ..._ds_mfr_spide
.CC |7264 6167 6765 725F 626C 6164 6530 302E 626D 7005 |rdagger_blade00.bmp.
```

Although your file name may be different, compare your changes against those shown above. You should still see the same bytes (numbers) on the left and right sides of your changes.

Now hit F3 to run that same search, for 'NiSourceTexture'. Repeat the same procedure with the next texture: first note the texture you're replacing (the 'jewel', pommel texture in this case), then record how many characters your new file name uses, edit the 'len' to the left of the old file name, delete the old file name, and then type in your new, original file name. Hit F3 once again to get this done one last time, for the hilt texture.

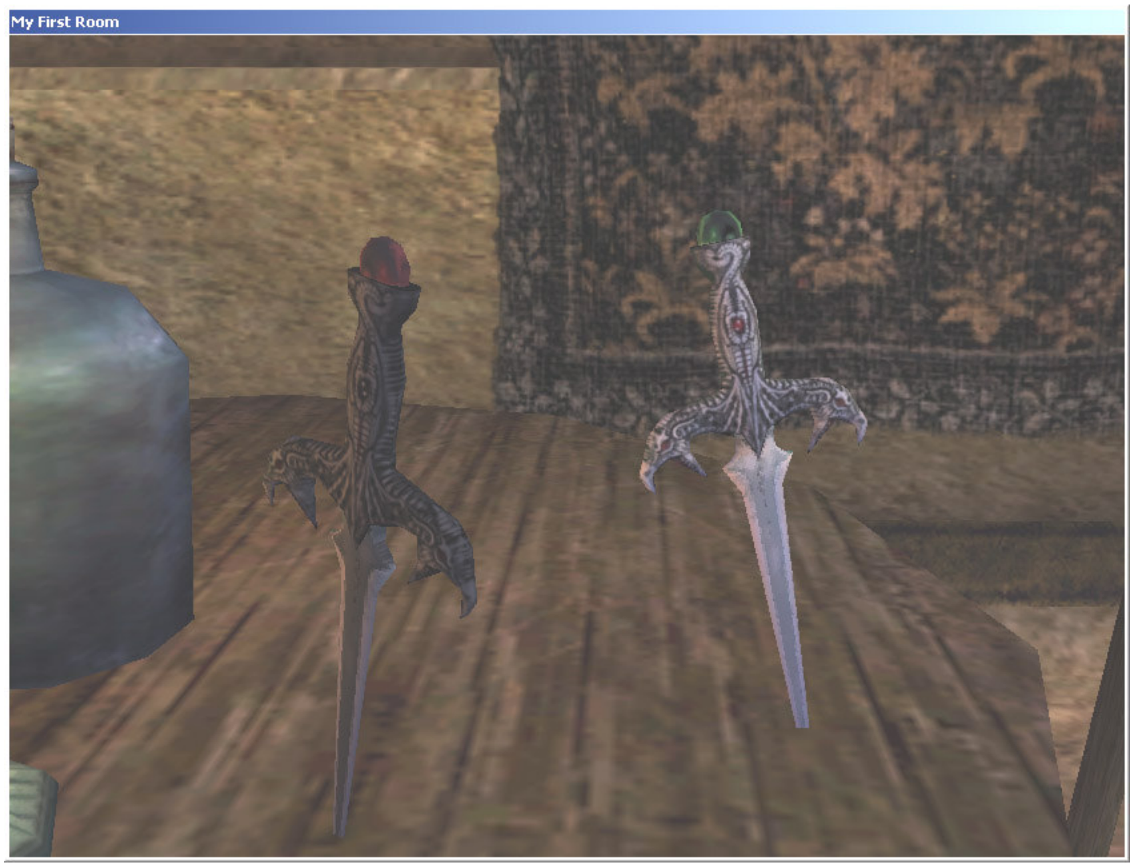
If you're having trouble with getting this done just right, or you'd like to just take it easy for now, you can use or compare against mine directly. Download it right here. Note that my NIF references my BMP files, so you'll also need to download those (from the above links) if you wish to use my NIF.

Finally, make sure everything is in the proper directory. Your NIF is in the meshes directory, in the w sub-directory. Your textures (BMPs) are in the Textures directory.

Now get back into TESCS, and open the properties for The Spider Dagger once more. Hit the top button, that currently lists W_Dagger_dragon.nif, and browse to your hard drive (not the CS CD), and point to your NIF. If you've done everything correctly, you *won't* get any errors. If you missed even one byte while you were hex editing, though, you will get an error. Or two, or three, or six. If you get completely stuck, try using my NIF and my BMPs, and make sure everything's in the correct directories!

(One more note: when you use the Preview window, as opposed to the Render window, it seems to want your texture files in the same directory as the NIF, rather than in the proper location - the Textures directory. Odd.)

Here's my Spider Dagger, juxtaposed with the original (a plain old steel dagger) for contrast:



Now that wasn't so bad, was it?

(NB: I haven't covered editing the 'icon' (Inventory Image) for the item, here, but you should be able to figure that out pretty easily - it's just a TGA, another common image format, which you can copy from the CD to the appropriate (mirrored) directory, and edit to your heart's content, then assign to the object.)

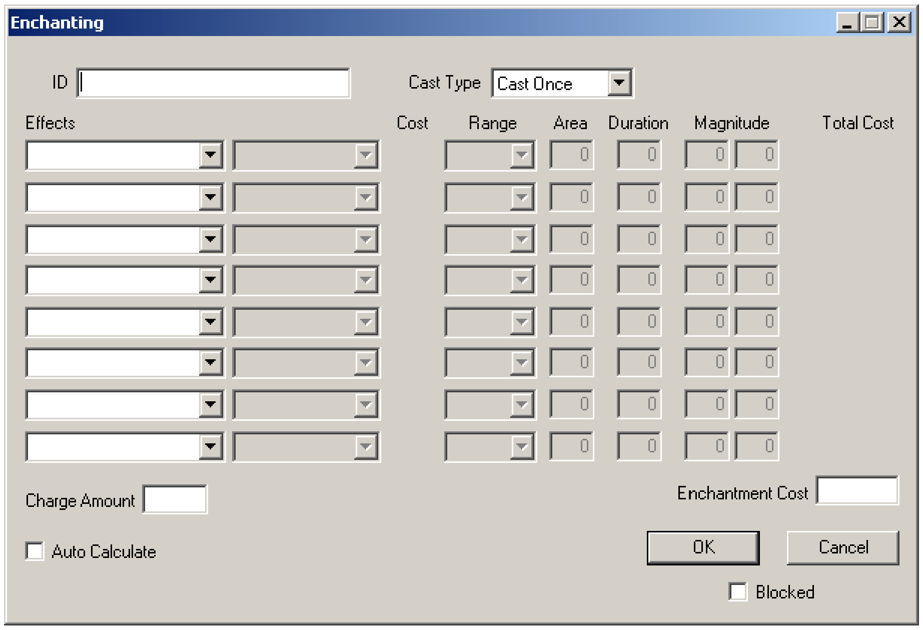
Extra Oomph

I'm going to put some extra details onto my dagger, before I wrap up. Specifically, I'd like to give it a few enchantments, so as to make it worthy of its owner, the Silent Paw of the South.

You may notice that the drop-down list for Enchanting (back in the properties for the ID) lists a whole bunch of odd-sounding entries, "And how the heck do you add more than one enchantment onto this thing?" young grasshopper complains in wonderment.

The 'Enchanting' list is actually expecting you to specify a "pre-cooked" enchantment, which could actually be a whole bunch of individual enchantments combined into one. You design new entries for this list under the 'Enchanting' tab.

Hit the Enchanting tab, and once again right-click in the list to choose New.



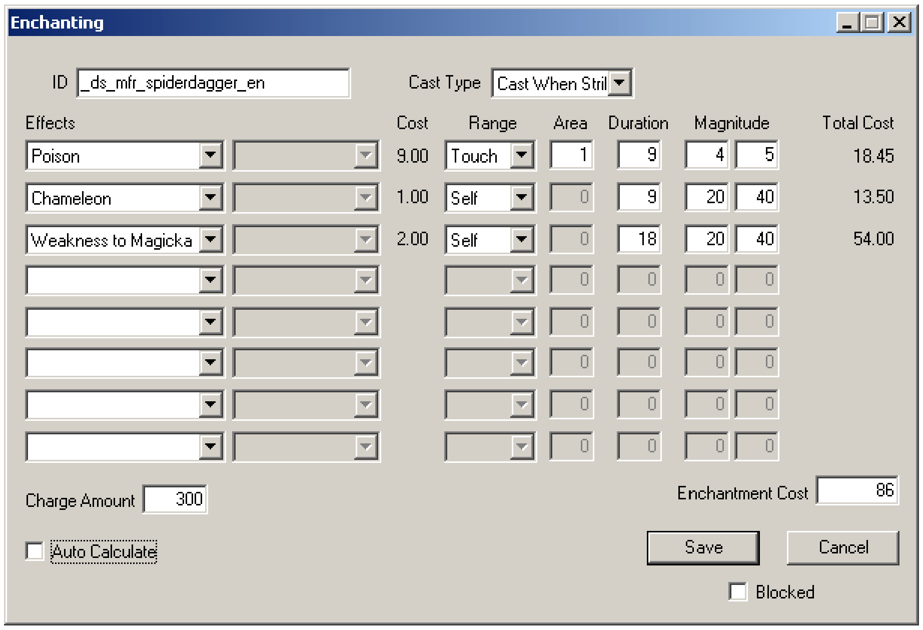
I start with an ID for my new (custom) enchantment. Once again I rely on SANC (and you don't necessarily have to), choosing '_ds_mfr_spiderdagger_en' as my ID. I set the Cast Type to Cast When Strikes - the enchantment(s) will be activated when the dagger successfully lands a blow.

For my first (and primary) effect, I select Poison. This is the main purpose of the Spider Dagger, after all - its other effects are tertiary (and so will be somewhat low-power). The range will be 'Touch' (remember that this is a 'Cast When Strikes' style enchantment, so 'Target' is unnecessary), the Area 1 (no 'splash' damage), the duration 9 (a whopping nine seconds!), and the Magnitude 4 through 5 (damage). If you're not entirely familiar with what all of these values mean, you should get into the game and head to an Enchanter, and play around with the interface for enchanting a weapon or item.

For my second effect, I select Chameleon. After the dagger makes its venomous strike, the assassin wielding it should fade away as if into the very surroundings, facilitating either a quick retreat - or an advantageous position to watch as the victim reels in the slow torture of the potent primary effect. The range is Self, of course, and I keep the other values low-power: just 9 (seconds) for the duration (not that long for Chameleon, actually), with a magnitude of 20 to 40 (with Chameleon, this means '20% to 40% chance of obscurity'). Note that this effect won't stack even after multiple strikes - each new strike just resets the effect.

Finally, I pick a third effect, a negative effect this time: 'Weakness to Magicka', which aligns with the sinister history of the dagger and the demise of its original owner (due to a cunning flaw the dagger's creator gave to the weapon, anticipating its ultimate use by the assassin against the wizard creator himself!). I set the Range to Self, the Duration to 18 (!), and the Magnitude to 20-40 (!). This won't stack, either, of course.

I tick Auto Calculate, to fill in the Charge Amount and Enchantment Cost values for me. Then I untick it, and set the Charge Amount down to just 300. The dagger (enchantment) has fairly limited use, again, to keep it (me) from being overpowered. The Enchantment Cost is, you guessed it, the 'cost per use' of the enchantment. So with a full charge (300), the dagger would only be able to land three successive strikes, before it would need to begin charging again. (There's more to it than that, actually - the base cost of 86 is reduced via the Enchant skill.)


Enchanting

ID _ds_mfr_spiderdagger_en

Cast Type: Cast When Stril

Effects		Cost	Range	Area	Duration	Magnitude		Total Cost
Poison		9.00	Touch	1	9	4	5	18.45
Chameleon		1.00	Self	0	9	20	40	13.50
Weakness to Magicka		2.00	Self	0	18	20	40	54.00
				0	0	0	0	
				0	0	0	0	
				0	0	0	0	
				0	0	0	0	
				0	0	0	0	

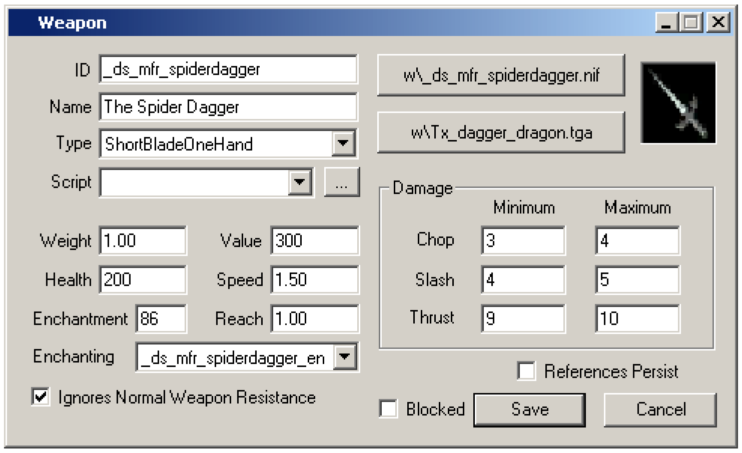
Charge Amount 300

Enchantment Cost 86

☐ Auto Calculate

☐ Blocked

Back on the Weapon tab, I again open the properties for my dagger, and now I assign it the 'enchantment' I've just created, and set the 'Enchantment' value to 86. This is the maximum amount of enchanting the object can accept. Because the cost of the enchantment I've created is 86, this means no new enchantments can be added to the dagger. If I'd left the Enchantment at 100, there would be 14 extra points worth of new enchantments that could be added to the dagger - and 14 is enough for some fairly substantial extra spells. (I want the dagger to remain exactly as powerful as I've designed it to be.)


Weapon

ID _ds_mfr_spiderdagger

Name The Spider Dagger

Type ShortBladeOneHand

Script

w_ds_mfr_spiderdagger.nif

w\Tx_dagger_dragon.tga

Weight 1.00 Value 300

Health 200 Speed 1.50

Enchantment 86 Reach 1.00

Enchanting _ds_mfr_spiderdagger_en

☑ Ignores Normal Weapon Resistance

Damage	Minimum	Maximum
Chop	3	4
Slash	4	5
Thrust	9	10

☐ References Persist

☐ Blocked

Save up your ESP and head into the game to have a look at your shiny new custom dagger.

Actually, it's a little *too* shiny for my taste (it has the 'shimmery' effect of all enchanted items), unfortunately there's apparently no easy or straightforward way (to the best of my knowledge) to remove the glow on any one particular item - though there are mods out there that will remove the glow on *all* magic items. (Which I'm certainly thinking about installing, after all that hassle with re-skinning my Spider!)

At this point you should have enough knowledge to figure out what you'd need to, in crafting other kinds of unique, custom objects. Have at it. There's one last thing I'm going to cover in this tutorial, and that's how you can use some basic scripting to add secret passage to your new downstairs room - leading to the Master Catpaw's underground hidden lair!

To the Batmobile, Robin!

Scripting is perhaps the one element of modding that many would consider the most advanced or complex aspect of TESCS, though in fact the language TESCS uses is fairly simple and (if you're already a programmer) very easy to pick up. That said, I'm not going to teach you the scripting language inside and out - rather, I'm going to cover some of the basic concepts involved in scripting, and show you how one simple script works. You can just copy my scipt and nod your head if you feel scripting isn't really your thing - or maybe you'll get enough of a feel for it that you'll decide to look through some of the scripts included with the game, to learn what scripting's really capable of.

A very common use for scripting is the old, 'hit this switch to open this door' scenario. And that's exactly how we're going to employ it. The bed downstairs is in fact just a prop, concealing a secret door beneath it, which can be used to access an underground chamber beneath the room! To move the bed aside to access the door, one must have in their possession a specific 'key' item. Normally when you 'activate' the bed (target it, and 'Use' it), you're prompted to rest. With the 'key' in your possession, however, the bed will instead move aside (or back into place). Tricky, no? Certainly, as befits one as cunning and catty as meow. (No, I am not going to spare you any more of my dry wit any time soon, grasshopper.)

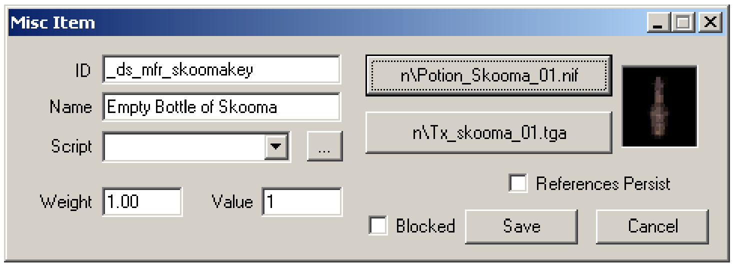
To get started, I need to delineate what 'items' I'll need for the script to work. I'll need the 'key' object, and I'll need a special bed object, not just the one I've got now. Finally, I'll need a new door, under the bed, to lead to a new cell, which I'll also have to create and populate.

The key and bed will be unique items, because I'm going to have to change aspects of their ID information, so I'll create custom objects for both of them.

Let's start with the key. It's not going to be *a key* of course, silly Padawan, but it's actually going to be what appears as a harmless, pointless, useless empty bottle of skooma.

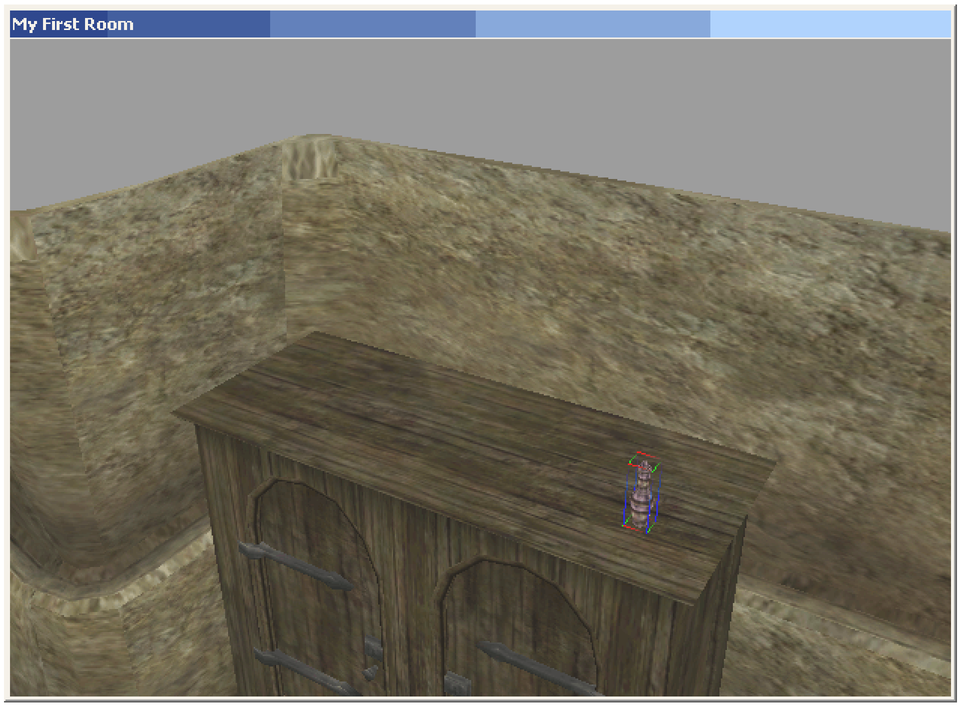
I'm creating a custom Misc Item type object in this case, but I need to do a little research before I get started with it. First, I hit the Alchemy tab, and look up Skooma (potion_skooma_01). What I want to know is its NIF and icon (TGA), so I can produce a clone Misc Item that looks the same - yet doesn't apply any kind of enchantment (since it's empty). The NIF is "n\Potion_Skooma_01.NIF", and the icon, "n\Tx_skooma_01.tga".

I hit the Misc Item tab, and right-click in the list, to choose New. I set it up like so:



ID: _ds_mfr_skoomakey
Name: Empty Bottle of Skooma
Weight: 1.00
Value: 1
n\Potion_Skooma_01.nif
n\Tx_skooma_01.tga

I drag a copy of the bottle out into the world, setting it atop my closet cabinet.



Now for the bed. I've already got a bed right in the spot I want, so I just open the properties for that bed, and change the ID to "_ds_mfr_falsebed". When I hit Save, I'm prompted to create a new object - I answer Yes to this, of course. I realize at this point that the pillow on top of the bed is either going to have to go, or is going to have to move with the bed, as the bed moves out of and back into place. There's the potential for some scripting bugs, here, since the pillow could feasibly be picked up and, say, set back down onto the table. The pillow certainly should not move if its on the table! I can get around this, however, with another little scripting trick, that prevents the pillow from behind removed from the bed. A bit of a kluge, but I won't be entirely perfectionist, at this level.

It is, however, going to need to be a unique pillow. So I open the pillow's properties, change the ID (to "_ds_mfr_pillow"), and again create a new object when prompted. There's one other thing you'll need to do with the pillow, which is rather important. Check the box for 'References Persist'. For a different object's script to be able to operate upon this object (the bed's script will need to move the pillow), this box needs to be checked.

And finally the trap door. This needs to go under the bed, so you can either move the bed (and pillow) out of the way while you work, then put them back into place when done, or work with a really awkward camera angle if you don't want to mess with the bed/pillow's position. I'm opting for the first.

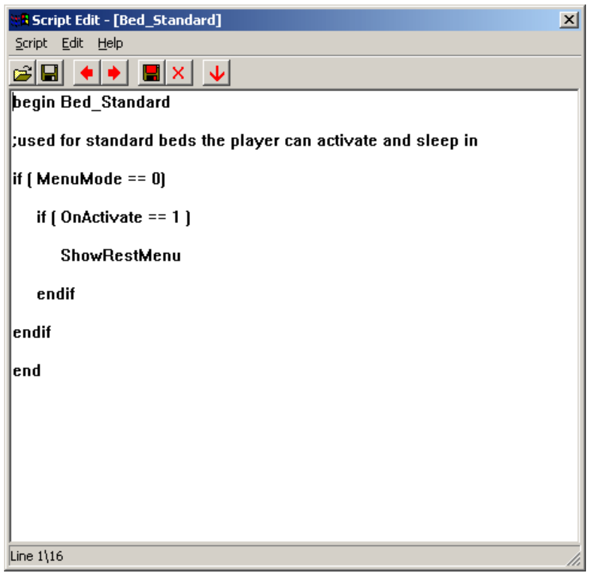
I want a door already designed for the 'trap door' look, so on the Door tab, I just sort by Name, and look through all of the choices named 'trap door'. I decide ex_h_trapdoor_01 is the one I want. Except that it's too big. So when I drag out a copy, I get into the *Reference Data* for the door, and set the 3D scale to .60. (Scaling objects is restricted from .5 to 2.0.)

My First Room

Now I move the bed back into place. The goal is to conceal the door just enough so that, even though it may be vaguely visible at a given distance and angle, you won't actually be able to 'Use' it with the bed in place. It helps, of course, that the player can't crouch or lie down! Just to make sure, I save up the ESP, and playtest a bit, in game. It takes a few extra nudges, but eventually I get the door right where it needs to be.

Right now the door doesn't lead anywhere, and we won't be able to set that up until we've got the new cell ready, so that will wait for last. At this point, we're ready to get to the real meat of it - the script!

Get into to the properties for the bed again. Note how the Script value is *ID information*, so assigning a script to an object entails creating a custom object! Right now the script is set to Bed_Standard, the script all beds use. Hit the ellipses (three dots) button next to Script, to open that script in the Script Editor.



```
begin Bed_Standard

;used for standard beds the player can activate and sleep in

if ( MenuMode == 0)

    if ( OnActivate == 1 )

        ShowRestMenu

    endif

endif

end
```

Well that doesn't look so tough, right? Much of the scripting language is so intuitive that you could probably figure out what it's doing just looking at it, and making intelligent guesses. The script has a header (and footer) that 'begin' and 'end' the script, it has a line that's a comment, specified by a semi-colon at the beginning of the line, and it has two "if" clauses, one inside the other, which ask (respectively), "Is MenuMode equal to 0? (Not 'true'?)" and "Is OnActivate' equal to 1? ('True'?)".

The first question is asking whether or not the player has the menu screen open. You can't use a bed while you have your menus open, you have to return to the game first. Next, the question is asked, "Did the player just 'activate' this object?" This needs to be asked because the script for this object is *constantly* running, so long as the player is in its vicinity, so unless the player has 'activated' (used) the bed, the script shouldn't do anything.

If the bed *has* been activated, then the function 'ShowRestMenu' is called, which as you might guess, brings up the dialogue for resting. Using a little intuition, you might wonder if this same script could be applied to, say, a fork, or a wall. You could indeed apply this script to a fork. You could not apply it to a wall - for one thing, you can't 'use' (activate) a wall, but for another, Static objects cannot have scripts attached to them, at all. (Although you could easily create a custom object, of type Misc Item, which uses the same model as a wall - and in that case, you could attach this script to it!)

A long time ago, I explained that there's a basic difference between objects of type Activator, and Misc Item. Well, now we finally get down to what that basic difference is. A Misc Item, when 'used', will be added to the player's inventory. This doesn't require a script. An Activator, when used, will by default *do nothing*. It's not until you attach a script to an Activator, and check for 'OnActivate', that it will actually 'do' something. Note that you could, of course, add a script to a Misc Item, check for 'OnActivate', and have it do something *besides* (or in addition to) being added to the player's inventory. (In fact, we'll be using just such a trick, with the pillow!)

But back to the bed. The current script isn't what we need - not quite. We need an additional if-clause, another question that asks, "Does the player have the Empty Bottle of Skooma in her inventory?" And if the answer is Yes, then we need to tell the bed to start moving. Then we'll need a section of the script that handles the bed's moving (and stopping). Altogether, it looks like this:

```
begin DS_mfr_falsebed

; Acts like a normal bed, until the player possesses
; the Empty Bottle of Skooma. Then the bed scoots
; aside to yield access to the trap door beneath!

short Moving
float Timer
short toggleSound

if ( MenuMode == 1 )
        return
endif

if ( OnActivate == 1 )
        if ( player->GetItemCount "_ds_mfr_skoomakey" > 0 )
                if ( Moving == 0 )
                        set Moving to 1
                        set toggleSound to 0
                elseif ( Moving == 2 )
                        set Moving to 3
                        set toggleSound to 0
                endif
        else
                ShowRestMenu
        endif
endif

; Moving states:
; 0  At rest in 'set' (covering door) position
; 1  Moving from 'set' to 'open'
; 2  At rest in 'open' (revealing door) position
; 3  Moving from 'open' to 'set'
if ( Moving == 0 )
        return
elseif ( Moving == 2 )
        return
elseif ( Moving == 1 )
        if ( toggleSound == 0 )
                PlaySound "Door Stone Close"
                set toggleSound to 1
        endif
        MoveWorld Y, -10
        "_ds_mfr_pillow"->MoveWorld Y, -10
elseif ( Moving == 3 )
        if ( toggleSound == 0 )
                PlaySound "Door Stone Close"
                set toggleSound to 1
        endif
        MoveWorld Y, 10
        "_ds_mfr_pillow"->MoveWorld Y, 10
endif

if ( Timer > 3 )
        set Timer to 0
        if ( Moving == 1 )
                set Moving to 2
        else
                set Moving to 0
        endif
        return
endif

set Timer to ( Timer + GetSecondsPassed )

end
```

Quite a doozy, eh? But I'll break it down for you, at least at a high level, section by section - so that even if you don't feel like you could write something like this, on your own, you'll at least have an understanding for what the script you're using *accomplishes*.

```
begin DS_mfr_falsebed
```

We start with the header, which define's the script's name. I'm using SANC, specifying my call-sign (DS, dragonsong), with the module's name, and the particular identifier for this script, which is just named after the object it's assigned to.

```
; Acts like a normal bed, until the player possesses
; the Empty Bottle of Skooma. Then the bed scoots
; aside to yield access to the trap door beneath!
```

Next we have some comments. These are for your benefit, as well as mine. If I come back to this script after a few months, I want to be able to quickly get an idea for what it does, without looking at all the code to figure it out on my own. The comments at the top greatly facilitate this. It's called "keeping code maintainable".

```
short Moving
float Timer
short toggleSound
```

As in one other well-known language *cough*C*cough* (or, on the other paw, *unlike* in one other well-known language *cough*Perl*cough*), all variables must be pre-declared, along with their type, before they can be used. The variables I need for this script are 'Moving', 'Timer', and 'toggleSound'. Moving will keep track of the bed's movement. Timer is a timing mechanism which I use to track when 3 seconds are up (the time it takes for the bed to move), just like a stopwatch. Finally, toggleSound is used to keep track of whether the sound effect for the bed moving has played. (It only needs to play once, throughout the whole moving process.)

```
if ( MenuMode == 1 )
        return
```

```
endif
```

The script shouldn't run if you have your menus open. A script can be told to finish processing prematurely with the 'return' function. This is just a different way of accomplishing what the original script for the bed did - checking for whether MenuMode was 0, and proceeding with the rest of the code, if so. (I prefer my method - it lends to less clutter with indentation.)

```
if ( OnActivate == 1 )
        if ( player->GetItemCount "_ds_mfr_skoomakey" > 0 )
                if ( Moving == 0 )
                        set Moving to 1
                        set toggleSound to 0
                elseif ( Moving == 2 )
                        set Moving to 3
                        set toggleSound to 0
                endif
        else
                ShowRestMenu
        endif
endif
```

Here's the first really hefty chunk of code, here. The outter-most if-clause is asking the question, "Has the player activated the bed?" So everything inside of this block deals with what happens when the player 'uses' the bed.

The next question in, using the GetItemCount function, asks whether player has "more than 0" (at least 1) of the object "_ds_mfr_skoomakey" (note how the object's ID is used, not its Name!). This is the qualifier for the player possessing the 'key' that moves the bed. If you pay attention to the indentation, you'll see that this particular if-clause also has an 'else' alternative. So if the player *does* have "more than 0" of the key, then the block immediately after the 'if' question is processed. In any other case (which is to say, the player has "none" of the key), the block after the 'else' alternative is processed: which is the function 'ShowRestMenu', the normal behavior you would expect from a bed.

Should the player possess the key, the next (and last) question in deals with the bed's current "Moving" state. If that state is currently 0, we change the state to 1, to reflect that the bed is now moving (outward), and we also toggle the variable that indicates a sound should be played. On the other hand (elseif) the Moving state is currently 2, we change the state to 3, to reflect that the bed is now moving (inward), and again we also toggle the variable that indicates a sound should be played. There are no other alternatives for this question - that is, nothing at all is done if the Moving state is 1 or 3. That's because the bed shouldn't be interrupted while it's already in the process of moving (state 1 or 3), so nothing happens when you 'acitvate' it in one of these states.

```
; Moving states:
; 0  At rest in 'set' (covering door) position
; 1  Moving from 'set' to 'open'
; 2  At rest in 'open' (revealing door) position
; 3  Moving from 'open' to 'set'
if ( Moving == 0 )
        return
elseif ( Moving == 2 )
        return
elseif ( Moving == 1 )
        if ( toggleSound == 0 )
                PlaySound "Door Stone Close"
                set toggleSound to 1
        endif
        MoveWorld Y, -10
        "_ds_mfr_pillow"->MoveWorld Y, -10
elseif ( Moving == 3 )
        if ( toggleSound == 0 )
                PlaySound "Door Stone Close"
                set toggleSound to 1
        endif
        MoveWorld Y, 10
        "_ds_mfr_pillow"->MoveWorld Y, 10
endif
```

Another seemingly byzantine block of code, heralded with an innocuous paragraph of comments, outlining what each state for Moving means, to the script. The outter-most question, this time, has to do with what Moving is currently set to. There is one block of code beneath for each possible state.

If the state is 0, the script simply returns. Nothing needs to happen, while the bed is at rest.

If the state is 2, it's the same deal as with 0. If you're already a programmer, you may be wondering, "Can't you state, 'if (Moving == 0 || Moving == 2)'?" Alas, you can't. There are no logical || or && operators, in the Morrowind scripting language.

Next we check to see if the state is 1. If so, there's another question asked, first of all: is toggleSound set to 0 (its initial state)? If so, then play the sound "Door Stone Close", and then set toggleSound to 1, so that the sound won't play again any time soon. That question handled, we next perform the MoveWorld function twice - the first time is on the current object itself (so the function is called by itself), the second time, on the object "_ds_mfr_pillow", which would be the pillow atop the bed. The MoveWorld function moves an object on a particular axis, a particular distance (in the span of 1 second). It differs from the Move function in that it doesn't check for how an object may be rotated (a rotated object's axes change based on the object's rotation - so the 'X' axis points in an entirely different direction when an object is facing a different direction), it just uses the 'absolute' axis specified: the Y axis, in this case. (Was that a mouthful.) The skinny: the bed and the pillow move down by 10 units on the Y axis.

And finally, state 3. This is precisely the same as state 1, except that movement on the Y axis is positive, instead of negative (moving in the other direction).

```
if ( Timer > 3 )
        set Timer to 0
        if ( Moving == 1 )
                set Moving to 2
        else
                set Moving to 0
        endif
        return
endif
```

Ahh, a much more peaceful-looking section, yes? What we're checking for here is if 3 seconds (based on our 'stopwatch' variable) have passed. If so, then reset the stopwatch (for its next use), and then update the Moving state to reflect that the bed is now at rest, in its arrival position. Finally, the script returns, since we don't want to process the next line after 3 seconds are up:

```
set Timer to ( Timer + GetSecondsPassed )
```

This updates the stopwatch, using the GetSecondsPassed function. It's very last, and in timer-based scripts, should remain very last, or else Bad Things(tm) could happen.

```
end
```

And that's all, folks!

There's just the one extra detail of the script that the pillow needs. This one's so simple, that I'll leave it to you to see if you can figure out what all of the code does:

```
begin DS_mfr_pillow

if ( MenuMode == 1 )
        return
endif

if ( OnActivate == 1 )
        messagebox "Why don't we just leave that where it is, 'kay?"
endif

end
```

Right! With the scripts saved up and assigned to the two objects, save your ESP and head into the room to check things out. You should find that the bed tends to act just like a bed when you don't have the Empty Bottle of Skooma in your possession, while with the key in your grubby little paws, the bed moves back and forth at your whim. The pillow also cannot be removed from atop the bed. And there's the trap door, beneath it... leading nowhere, still.

And you were probably thinking that, at this point, I'd walk you through setting up the secret lair cell, beneath the bed, mm? In fact, grasshopper, it's time you set out on your own. We've arrived at the end of a long and arduous tut- journey, that is, and I can see that you have grown, in wisdom and in confidence. I'll leave it up to you to design your own secret lair however you see fit. Go back and brush up on the steps we took in the first tutorial if you need a reminder on how to get a new cell started. Be creative, look around at some of the other cells in Vivec, and beyond. Or break from convention entirely and try something new and challenging!

The world is yours.