Ted's RPG Rant – Modding ToEE Tutorial

By: ShiningTed

Compiled by **krunch**

This ToEE World Builder (ToEEWB) tutorial is from the blogspace of **ShiningTed** and covers these ToEE modding topics.

Writing Dialogue for ToEE (Part 1)

Ok, firstly what do we need? Well, some templates will be handy! So I have made a couple. Download them **here.**
http://www.co8.org/forum/showthread.php?t=2228

These can both be opened in Notepad. But we have other things to download first. For starters you will need Phalzyr's 'Proto-Ed'. It can be found **here**.
http://www.co8.org/forum/showthread.php?t=1887

This is useful for viewing dlg files and essential for adding any new characters you want to use the dialogue files. Well, not essential, you can do both these things in ToEE World Builder as well. But it is a very different tool. Download that too! The latest version is at the last location; try the thread that accompanies it, **here**.
http://www.co8.org/forum/showthread.php?t=2021

Once you are comfortable using these tools, you will instinctively recognize when one is preferable to the other for certain tasks. One more thing to 'download', and that's a thread from the great man Phalzyr himself, where he lists many of the things that can be found in dialogue and script files: **find the thread here.**
http://www.co8.org/forum/showthread.php?t=372

This tutorial will build on that thread and explain step-by-step how to incorporate the things found in it and the pitfalls that may be encountered.

Now, lets have a quick look at the template files. Notice that the names, **00275DEMO.dlg** and **py00275DEMO.py** are (from the perspective we are concerned with) IDENTICAL. Both are '00275DEMO'. That's our first lesson, ALWAYS name them the same! Don't think an identical number is enough, I learnt that the hard way when I had a couple of files called **00291tennant.dlg** and **py00291tenant.py**. That one little typo cost me MANY hours of headaches!!!

What do they mean? Well, the .dlg file contains all the dialogue, and the .py file (short for .python) contains the scripts that control things like dialogue :-) It also controls other things which we will examine when the time comes. For now, there is only one script in our template, so lets pop it open. Open it in Notepad (you could also use Notepad++ or another text editor, Agetian also recommends something, try **here.** For now I will assume you are using Notepad because everyone has it - using other's will simply make all this easier).

You will see the following:

```
from toee import *

def san_dialog( attachee, triggerer ):
_____________triggerer.begin_dialog( attachee, 1 )
_____________attachee.turn_towards(triggerer)
_____________return SKIP_DEFAULT
```

If you have Proto-Ed, start this too. Open the game's protos.tab (if this is the first time you have ever done it, it is found in **Atari**/**Temple of Elemental Evil**/**Data**/**Rules**). You can fiddle with this all you like, just DON'T HIT SAVE!!! When you are confortable changing things and saving, make SURE you back it up. This is probably the most fundamental file in the game (well this and temple.dll. And the .dat files ;-))

In the protos.tab, go DOWN to the 14000's - behold, the prototypes of the NPCs! Now go ACROSS to col 277. Here onwards are the script file references for each character. The dialogue script pointer is in the that first col, 277 (its marked 'san_dialog'). You'll notice that this phrase is the same as in the **py00275DEMO.py** file we just opened.

What does it mean? It means that for whichever NPC whose prototype we assign that value 00275 for (in that column 277) it will go to THAT file 'py00275' to look for the dialogue script. It will ALSO automatically use the asscoiated dialogue file '00275'.

So... when you want to use these template files, you will: save them in the appropriate folders, put 00275 0 0 0 0 in the san_dialog column of the character whom you want to use this dialogue, and THAT'S IT! Not too hard, is it? Still with us? (Ignore those zeros, they are for parameters not used in dialogue stuff).

Lets reconsider the issue of why 2 files. Again, the dialogue goes in the dlg file. The .py file contains the script that controls how the game accesses that dialogue file. It can be written in such a way that the dialogue starts in different places for different circumstances. Also, the .py file (again) can control other things. Have a look at the names of the other columns in the protos.tab: san_dying, San_enter_combat, san_resurrect etc. What these mean is that when an NPC goes into combat, or dies, or is resurrected, the game goes to its prototype (recorded in protos.tab) and checks to see if there is a specific file to go to: if so, it goes to that file and executes the script in it.

So, when you click on an NPC to talk to them, the game goes to that NPC's prototype, goes along to column 277, and finds what script to execute. Lets imagine it has found '00275'. It will go to file **py00275.py** and execute the san_dialog script. Lets have a second look at that:

```
def san_dialog( attachee, triggerer ):
_____triggerer.begin_dialog( attachee, 1 )
_____attachee.turn_towards(triggerer)
_____return SKIP_DEFAULT
```

This is as straightforward as a script gets in the game, but probably looks like gibberish. Lets look carefully at each element (note that this is a 'how to write dialogues for ToEE' tutorial, not a 'how to be a Python scripting expert' tutorial. Its to tell you how to do stuff, not an explanation of the mechanics of Python - Python experts might be aghast by all this, but I am keeping it simple!)

Firstly, the parameters: attachee and triggerer.

Attachee in this case is the NPC. Think of it as whoever the script is attached to. Simple.

Triggerer is whoever has done something to trigger the script. In this case (and in many others) it is the player. You have clicked on the NPC, triggering this script as the game decides how the NPC should react.

So, the first line: triggerer.begin_dialog(attachee, 1). This tells the triggerer (the player) to go into dialogue mode: the little window comes up, the screen centres on the speaking character etc. Importantly, it tells the triggerer (the player) to go into dialogue mode with the NPC (the attachee, the 'owner' of the script) and most importantly TO START AT LINE 1. Obviously, more complicated scripts can have the dialogue start at other lines depending on the circumstances.

The next line: attachee.turn_towards(triggerer) Simple enough: it turns the NPC (the attachee) to face the player (triggerer). Always try to include this in your script, the game doesn't have it consistently and it is a bit disconcerting to have people facing the back wall when they talk to u. Another one of those things that drags you out of the game-playing moment.

Final line: return. That tells the game the script is finished, carry on, and skip the default bit (to see a default in action, click on one of the pirates wandering around Nulb).

Soooo... lets recap. Click on NPC in game, game goes to the file associated with that NPC's prototype, and executes the script there, telling it (in this case) to turn the NPC toward the PC and start dialogue from the first line in the dialogue file.

Writing Dialogue for ToEE (Part 2)

Lets jump straight into writing dialogue today. Open up **00275Demo.dlg**. Also, open up Phalzyr's thread you downloaded (or copied out somewhere) last time, or go to www.Co8.org and look at it again.

The very first thing on the dlg page is {1}. This is, not surprisingly, a line number. The game accesses the file by line numbers, and the dlg file advances the dialogue by going to different line numbers. Easy, init. You may remember that our script, in **py00275Demo.py**, will start the dialogue at line 1 each time the player clicks on this character. So lets see what line 1 is.

{1}{G:}{G:}{}{}{}{} // a greetings line from the end, as circumstances dictate.

The stuff beyond the '//' is a comment, you can erase that when you are ready to put this file into action or leave it there, it doesn't matter. Its what goes before it that is the juicy stuff that the game accesses.

So its {--LINE NUMBER--} {--COMMENT TO MALE PC--} {--COMMENT TO FEMALE PC--} {} {--LINE NUMBER AGAIN--} {} {--FUNCTION TO EXECUTE--}

Lets go through those in order.

{--LINE NUMBER--} You will notice that in this template, and in most dlg files in the game, the comments from the NPCs are in 'tens', and the replies from the player are in the unit between. Ie, the NPC comments are in line numbers 10, 20, 30 etc and the player's comments are in 11-15, 21-25, 31-35 etc. This is NOT essential, but is very handy in keeping track of what is happening. It IS necessary that the line numbers be sequential, because the game accesses the file sequentially, it does not parse the file and put all the numbers in order. If there is some catastrophic screw-up in the dlg file somewhere, the game will generally run the dlg file normally until that point, then everything after it will come up blank.

You may notice Phalzyr says, "you are limited to showing 5 PC replies at a time". This means only 5 will show up *on the screen* at any one time. You can have more than 5 replies, you can have as many as you want if you script them to occur under different circumstances (and, generally speaking, more is better!) But there is only room in the dlg box for 5 to show up at any one time. Don't believe me? Do my mod, and try taking all 8 of Lila's requested items to

her at once. When she asks you what you have for her, all 8 (plus a 'never mind' response) will trigger a comment from the PC, but only the first 5 will be seen on screen. As you hand over each item, something else will takes its place in the list until there is room for the 'never mind' response to occur.

{--COMMENT TO MALE PC--}{--COMENT TO FEMALE PC--} Ever wondered how the NPC knows to call your cute little Elven maidens "lass" and your big boofy barbarian blokes "sir"? Simple - they have seperate dlg entries. In the majority of cases they will be identical (and I can tell you from bug-hunting, the writers of ToEE used copy-n-paste a LOT!). But the option is there for them to be different, and indeed they can be wildly different and talking about two completely different things if you like (likewise, you can script the replies to only discuss certain options if the PC is of a particular gender. We will get to that shortly. Its a good example of why you might want to have a lot more than 5 replies available.)

Note: one of the most COMMON mistakes you can make in writing dlg files, is forgetting to do both of these. If there is very little variation between your male and female comments, and you are copy-n-pasting a lot (nothing wrong with that btw), you can easily forget to do it once, and since it will be surrounded on both sides by a number of single comment PC replies, the line looks normal. BUT, when you run the file, it screws up immediately after that point. Why is that?

Lets see how the game views it. The game is looking for:

{--#--}{--COMMENT--}{--COMMENT--}{}{}{}{}
{--#--}{--REPLY--}{}{--#--}{}{--#--}{}
{--#--}{--REPLY--}{}{--#--}{}{--#--}{}
{--#--}{--REPLY--}{}{--#--}{}{--#--}{} etc

What all those other brackets do we will get to in a minute.

BUT, if we leave out a comment, what does the game see?

{--#--}{--COMMENT--}{}{}{}{}{--#--}
{--REPLY--}{}{--#--}{}{--#--}{}{--#--}
{--REPLY--}{}{--#--}{}{--#--}{}{--#--}
{--REPLY--}{}{--#--}{}{--#--}{}{--#--}

When it goes for the missing NPC comment, it accesses the empty brackets in its place, and everything moves forward one. So in the next line number there is instead a comment, nothing where the next line number to go to should be (don't worry, I haven't told you where that is yet, just take my word for it) and so the game can't continue. It will probably just show nothing on the screen, and default the last (blank) line to be an 'exit' comment.

Don't worry if you make this mistake btw, we all have ;-)

Identifying these mistakes is much easier in something like Notepad++ than just normal ol' Notepad. Or, if you are like me and enjoy the spectacular simplicity of Notepad, try opening your .dlg files in Phalzyr's Proto-Ed when bug-hunting. This makes them MUCH easier to see, and also quickly picks up the other easy mistake to make, using the wrong shape brackets. If you are writing a big .dlg file and typing quickly it is all too easy to type '(' instead of '{' and screw everything right up.

Lets move on to those last 4 sets of brackets. In many .dlg files in the game these will all be blank, and the NPC comment line will end with {}{}{}{}. In other files, you may notice the line number gets repeated in the second bracket: eg for our first line, they would be {}{1}{}{}. You can do that or not as you like: I haven't met anyone who is entirely sure why it does or doesn't have to be there. I know I stopped bothering with it a while back and have noticed no ill effects whatsoever. Just be aware that sometimes the game has it.

The last set of the brackets are important. Here is where you can stick a function or script you want to execute at this point. A good example is from Liv's reworking of the tower encounter:

{366}{To arms my troops!}{To arms my troops!}{}{}{}{}
{367}{[more]}{}{1}{}{368}{}
{368}{Ready your spells! Draw your swords! Mark your targets!}{Ready your spells! Draw your swords! Mark your targets!}{}{}{}{buff_npc(npc,pc)}
{369}{[more]}{}{1}{}{370}{}
{370}{These fools killed Lareth!! Let us avenge his death!}{These fools killed Lareth!! Let us avenge his death!}{}{}{}{buff_npc_two(npc,pc); game.global_flags[833] = 1}

Ok: since this is a pretty linear 'dialogue' where the players only option is to click '[more]', Liv has not bothered spacing the line numbers by 10. For the moment, ignore the odd numbered lines (they are the players replies in this case, we will get to them next) and focus on the NPC comments. You will notice that each one starts with the line number, repeats the comment twice (once for male, once for female) and then has 3 empty sets of brackets, followed by (for the latter two) a command that should remind you of some of the scripts we have already seen in the .py file: buff_npc(npc, pc).

So... what do you suppose happens when the game executes these lines? If you answered "it goes looking for buff_npc and buff_npc_two in the .py file", you are completely correct! Have a cookie! We don't have to go into any detail here as to what these commands do, suffice it to say they execute scripts already put in the .py file. The effect should be obvious: the NPCs will buff themselves for the fight!

They also execute another command, game.global_flags[833] = 1. Again, we will look at global flags and variables in depth later. For now, it is enough to know you can set flags and variables and then check them later. Here, Liv assigns the value 1 to flag 833. There are only two values for a flag, 0 and 1, ie the flag, like a flag on a mailbox, is either up or down. Its default is 0, the flag is not set and gets ignored. Here we are 'setting' the flag (or flagging the encounter, or whatever jargon you want to call it). Later, at a specific situation (probably when you chat to Lareth again) you can have a script to check this flag. If it is 0 or unset, then you haven't been to the tower. If it is 1 or set, (as it would be in this case) then you have been to the tower encounter and your dialogue with Lareth can reflect that.

All make sense? For a list of some of the commands you can execute straight from dialogue like this, look at Phalzyr's thread under the section 'Actual Functions: Those with an * in front are known to work directly from dialog execute field.' Damn but he is a handy fella! Be aware that some only seem to function from the player's replies, not from the NPC's line. We will look at the more common ones in depth later, for now try popping some dlg files in the game and seeing how they are used. (The dialogue files, if I have not mentioned this are in **Atari/Temple of Elemental Evil/Data/Dlg**).

Now to the player replies. They look very similar but are actually quite different. They begin with a line, follow it with a comment and the last one can execute a command, but in between is where the changes happen.

Lets look at them.

{--LINE NUMBER--} {--REPLY TO NPC--} {} {--INTELLIGENCE OF PC--} {--CONDITION UNDER WHICH THIS LINE APPEARS--} {--LINE NUMBER TO GO TO--} {--FUNCTION TO EXECUTE--}

The first two are fairly obvious. The third bracket is always empty. The fourth is where the interesting stuff starts: it examines the intelligence of the PC, and displays the appropriate reply. There are 3 common parameters for this;

8 - this means anyone above Int of 8 will get this option. This is the normal conversation options.

-7 - this is for players with an Int of 7 or less. This is commonly called the 'dumb PC' option. The same thing might be said, but phrased in such a way that the PC who says it doesn't sound too bright. (In the game, this is often played for laughs).

1 - this is for anyone with Int 1 or over. This is a 'common' reply option, such as we saw with

the '[more]' option above, it doesn't matter what the Int of the PC is. If it is under 1, they would be paralyzed and unable to converse, so it covers everyone. It is used for common replies such as 'goodbye'.

You can also script specific stuff for smarter characters if you like, by adding a reply line that has, say, 15 or 17 (or more) here, so that only really intelligent PCs get the chance to explore this part of the dialogue. An example of that would be in Burne's puzzles, where only smarter PCs are given the option of explaining their correct answer (and therefore winning).

Anyway, the meaning of all this is that just as NPC comments come in male and female varieties, so there are usually TWO versions of each reply: one for normal characters, one for dumb characters. But they are shown on separate lines, and they are not compulsory for the game (as we saw the male and female ones were). Just remember if you don't have 'above 8/ under 7' options, you must have a '1' option, or you might create a situation where a character with an unexpected Int happens to be in the dialogue and there is no reply that suits them.

Moving on, the next bracket can contain wider conditions. It can also be left blank if you want that reply to always appear. Lets revisit Liv's tower encounter, the scenario has now moved on a bit, the baddies have buffed themselves and the PCs are claiming Lareth is not dead, he has merely run off:

{390}{You lie! Lareth was many things, but not a coward. You will die for your insolence!}{You lie! Lareth was many things, but a coward? No. You will die for your insolence!}{}{}{}{}
{391}{I think not. We crushed Lareth with ease. And now we will crush you and your band of paltry scum.}{}{8}{}{0}{npc.attack(pc)}
{392}{Me gonna bash yer skull!!}{}{-7}{}{0}{npc.attack(pc)}
{393}{I never said Lareth was a coward. He is waiting for us back at our hideout. Let me go get him.}{}{8}{pc.skill_level_get(npc, skill_bluff) >= 10}{500}{}
{394}{Layrest not cowherd. He wate fer us at da cave. @pcname@ brings him to ya.}{}{-7}{pc.skill_level_get(npc, skill_bluff) >= 10}{500}{}
{395}{Look you moron, Lareth is a friend of ours and he is going to be pretty damn angry with

your behavior.}{}{8}{pc.skill_level_get(npc, skill_intimidate) >= 10}{400}{}
{396}{Me Layrest friend. He gonna smash ye fer dis bad stuffs.}{}{-7}{pc.skill_level_get(npc, skill_intimidate) >= 10}{400}{}
{397}{Relax, friend. Lareth is our friend. I'll go get him and we can sort this all out. No problem here.}{}{8}{pc.skill_level_get(npc, skill_diplomacy) >= 10}{500}{}
{398}{Calm. @pcname@ is Layrest friend. Me go gets him. Yous sees.}{}{-7}{pc.skill_level_get(npc, skill_diplomacy) >= 10}{500}{}
{399}{[attack]}{}{1}{}{0}{npc.attack(pc)}

Ok: line 390 is the NPC response. After that, the even numbers are for Intelligent PC responses (note the 8) and the odd numbers are for Dumb PCs (note the -7), except for the last line (399) which is a common 'cut the crap lets just attack' response. There are attack options in the first lines (391-392) too, but it is not obvious that combat will immediatley ensue: if the player chooses this in an attempt to bluff the baddies, he or she will get a shock!

Depending on the circumstances, these may be the only ones that show up, as they are 'unconditional' (the brackets after the Int value '1' are empty), so the players will always have the option of fighting their way out.

Getting ahead of ourselves slightly, you may remember I just said that the last brackets for the PC are the same as for the NPC. They are for a command to be executed. Here, the command is 'npc.attack(pc)'. I won't bother explaining what that does! But note this is one of those that can only be executed from the player's reply, since it terminates the dialogue mode. Anything that terminates the dialogue can only be initiated from the player's reply, since if it was done by the NPC, you would not see their comment, it would flash up and terminate simultaneously.

Now, the conditions: we get a smart PC and dumb PC version of each, and they are;

pc.skill_level_get(npc, skill_bluff) >= 10
pc.skill_level_get(npc, skill_intimidate) >= 10
pc.skill_level_get(npc, skill_diplomacy) >= 10

The command 'get' tells the game to fetch the value at a certain point: it might be a critter flag (different to the global flags we just saw) or an attribute or check for a feat or a class or a gender, or other things. Here, it is checking the different skills. If the PC has a skill level equak to or greater than 10 in any of these, that option would appear. Hence if they have greater than 10 in all 3, all 3 will be given. On the screen, for a smart PC that would look like this:

Brigand leader: You lie! Lareth was many things, but not a coward. You will die for your insolence!

1. I think not. We crushed Lareth with ease. And now we will crush you and your band of paltry scum.
2. I never said Lareth was a coward. He is waiting for us back at our hideout. Let me go get him.
3. Look you moron, Lareth is a friend of ours and he is going to be pretty damn angry with your behavior.
4. Relax, friend. Lareth is our friend. I'll go get him and we can sort this all out. No problem here.
5. [attack] Obviously there will also be the different colours and the little icons to show you are using a skill.

Conditional statements, as Phalzyr says, can have operands such as == (exclusively equal), >= (greater than or equal), <= (less than or equal), > (greater than), < (less than), != (not equal).

Finally, there is the only set of brackets we have not looked at yet - line number. This is straightforward: if the player chooses this reply, then it will go to the line number in this (second last) bracket. In the examples above, it will go to a different line number depending on the skill used, as the Brigand leader reacts differently to each conversational gambit the player attempts. But the line number will be the same for both smart and dumb PCs: that is not essential btw (you could have both smart- or dumb-specific dialogue) but it is normally the

case. The line number must of course be an NPC comment, not a PC reply line, or the dialogue will terminate abruptly. A line number of '0' terminates the dialogue, whether because you are saying goodbye, entering combat, teleporting to a different location, the NPC is running off, you are moving to the barter screen or for whatever reason.

Well... that was a mouthful! And we never did write any dialogue :(We never even got to find out what the {G:} means! That's for next time. For your homework, have a look at the template, and look at some .dlg files in the game: note that they will ALWAYS conform to this pattern described, no matter what the content, ALWAYS ALWAYS ALWAYS.

Writing Dialogue for ToEE (Part 3)

Today we're actually going to write some dialogue! Pop **00275demo.dlg** again, and lets go back to line 1.

{1}{G:}{G:}{}{}{}{}

You should know what each of these are, though you might wonder why the opening line of this NPC is simply 'G:'. The brighter folks reading this will have figured out:

a) G: is a command of some sort (you don't usually see colons in dialogue ;-))
b) It is short for 'greeting' and is a shortcut to add greetings.

Or, you might have picked that up from looking at Phalzyr's thread. Whatever, them's the facts!

By using the letter 'G:', you can have the one single greeting line, and it will *change* to suit the circumstances of the greeting. If it is the first time the player has chatted to this NPC, you can have an introductory comment: if they have met before, it can be a 'welcome back' comment, if the PC has done something to anger the NPC it can be reflected in the greeting. These comments are all stored at the end in **specific line numbers**. Scroll down the template and you will see them, line numbers 10008-11004. By specific line numbers, I mean the comment

for greeting a naked PC is 10015 *in every dlg file*. Why is this? So the 'G:' command can access them of course! If you have been doing your homework, you may have noticed many .dlg files don't have these lines at the end: there are also default ones that can be used. So you don't HAVE to add them, you can still use 'G:' and the game will then supply an appropriate, if generic comment.

What benefit is there in using 'G:' if you just then have to write in the appropriate lines at the end of the file anyway? Well, it allows you to simplify the scripting that accesses the dlg file. You will remember we are starting with a very simple script:

```
def san_dialog( attachee, triggerer ):
_____________triggerer.begin_dialog( attachee, 1 )
_____________attachee.turn_towards(triggerer)
_____________return SKIP_DEFAULT
```

This always starts the dialogue at line 1. By using G: we can get away with this, always start at the same line but still have a dialogue that changes to suit circumstances.

What about the PC's comment, how do they change? Well, looking at the template this is apparent:

```
{2}{Hello, I am @pcname@.}{}{8}{not npc.has_met(pc)}{10}{}
{3}{Hi! Me @pcname@!}{}{-7}{not npc.has_met(pc)}{10}{}
{4}{K:}{}{1}{npc.has_met(pc)}{20}{}
{5}{E:}{}{1}{npc.has_met(pc)}{0}{}
```

Note that **npc.has_met(pc)** and **not npc.has_met(pc)** are in the conditional brackets as we defined them in the last tutorial: that is, these are conditions that have to be met to see these lines. Obviously, the introductory lines 2 & 3 are for when the PC & NPC have never met before: or, to use logical langaue, when the condition 'npc.has_met(pc)' is NOT true. Now, write some dialogue! Write in your own greeting. Note the '8' and '-7' for the intelligence

condition: also note the use of '@pcname@': whatever the name of the PC who does the talking, will be substituted here. This is how the game knows to call you by name, whatever that name may be.

Done a couple greetings? Well, when you feel like it, you can do all the ones down the bottom of the template as well. You don;t have to, you can always delete these when the time comes and the game will insert the generic onces, but you should at least have some idiosyncratic introductory lines at 10008 and 10009.

Perhaps you want to add a few more specific greetings: have different comments for evil or good parties, or for specific races or classes. Perhaps the NPC is an Elf and you want Elven PCs to acknowledge their kinsman, or perhaps the NPC is a priest of Hextor and you want PC Paladins or followers of Heironeous to have a few choice words to say. To do this, we add conditional lines similar to those we saw last time. I will copy Phalzyr's list here.

pc.skill_level_get(npc, skill_gather_information) == #
pc.stat_level_get(stat_deity) == #
pc.stat_level_get(stat_gender) == gender_male //**gender_female**
pc.stat_level_get(stat_alignment) == LAWFUL_GOOD (for instance)
pc.stat_level_get(stat_level_paladin) == #
pc.stat_level_get(stat_level_wizard) == #
pc.stat_level_get(stat_race) == race_orc (for instance)
pc.stat_level_get(stat_strength) >= #

Notice some have spaces and some don't - they are irrelevant, Python is a bit smarter than that. Also notice that 'stat' is used in a wider sense than D&D devotees might use it (ie just for attributes like Strength). Those above are pretty obvious since we have already seen the skills in action. 'Stat_deity' is checked by *number* according to **deity.mes** (this is in the Rules folder in **ToEE3.dat**, if you are serious about ToEE modding you HAVE to crack open those .dat files! Ask at Co8 for how).

Some other handy ones are:

anyone(pc.group_list(), "has_follower", #) This checks for a specific follower in the party (eg when doing the Preston Wetz 'sore tooth' quest, if you have Bertram in the party you are not only a worry, but you also have the option at that point of saying, hey I have a dentist right here! Or was that a ranger... :-/)

anyone(pc.group_list(), "has_item", #) This checks if any PC has a particular item in their inventory: ideal for quests, since you don't have to worry about whether the speaker has the item or whether perhaps the strongest PC carries everything but someone else does all the talking.

In both these cases, the # is from protos.tab. However, the number is from col 22 (in Proto-Ed) which is the 'name' of the object. If you open the protos.tab in Proto-Ed for, say, Elmo, you will discover his proto number is 14013, his 'name' in col 22 is 8000 and his id # is 20008! Confusing? Unfortunately, yes it is! In many cases these 3 numbers will be identical, but for NPCs (particularly followers) it can be awkward, because the game has seperate id #s for when they are a stranger and when you have met them. So if you click on Elmo from a distance, he should show up as 'drunk villager' but once u have met him, it will be Elmo. Just be aware of this.

One more condition to introduce here (that I didn't see in Phalzyr's thread but is probably there somewhere) is:

game.party_alignment == CHAOTIC_EVIL (for instance). This is for the alignment you chose for the beginning, ideal for things affecting the opening vignette quests.

Ok, that's it for now, I will try to update this with another post after work. In the mean time, get writing dialogue! Use those conditions to add lines 6, 7, 8 and 9 to the opening dialogue. Also, write some specific dialogue for the various greetings at the end: its that attention to detail (not bothering with the generic default stuff) that makes for an immersive game!

One more thing: you can group conditions with Boolean stuff like 'and' & 'or'. So for the opening lines, you might want something like:

{8}{I'll do the talking, runt.}{}{8}{not npc.has_met(pc) and game.party_alignment == CHAOTIC_EVIL}{50}{}

Note you need '==' for conditions, one '=' won't cut it! This is another very common error.

Writing Dialogue for ToEE (Part 4)

Ok this is the last major part of the Tutorial, after this you should be able to do most of the necessary dialogue writing for a module. In fact you can do a lot now, even if you don't realise it! But there will be many more tutorial chapters to come, dealing with all sorts of things... if necessary, review the previous chapters, then on with the show!

Quick review:
- We know what the various bracketed elements in each line do, and why there will always b the same number in each line no matter what is happening.
- We know how to go from one line to the next, and hence can already create great long dialogues if we like full of lots of options.
- We know how to add conditions so that the dialogue can reflect the situation, and so that certain lines only occur in certain circumstances (depending on the speaker, or the state of the storyline, or whatever).
- We know how to execute certain functions from the dialogue (not many yet ;-)).

What next? Well, lets do quests.

Quests can be activated from inside the dialogue files: indeed, this is pretty much the primary way of doing so. They can be set as **mentioned**, **accepted, botched, completed** or **unknown**. First, we need to assign the quest a number, then we can deal with it.

Numbers are assigned in **gamequest.mes** and **gamequestlog.mes**. The one accesses the other, and the relevance of the stuff in them is at the top (the 1st is the number and CR rating, the 2nd has the brief and full descriptions, numbered 200 apart). Its pretty self-explanatory. Open them up and look! Find a spare number (check the Co8 modding forum **here**, to see if anyone has claimed any new quests for an upcoming mod) and add:

1) The name of the quest (that will appear in the game in the quest log).
2) A description of the quest (ditto).
3) The CR number of the quest.

Note: these files are ornery! Take careful note where previous modders have left gaps between numbers and emulate these: if you get a CTD ingame when opening the logbook, you haven't been careful enough!

But enough cheerful thoughts, lets start adding quests to the .dlg files. The command for quests is **game.quests[#].state** (so if you are using quest nmber 180, that'll be **game.quests[180].state**). So, to change the state of a quest it would be:

game.quests[#].state = qs_mentioned
game.quests[#].state = qs_accepted
game.quests[#].state = qs_completed
game.quests[#].state = qs_botched

Of course, quests start off as **qs_unknown**. You could probably turn it back if bugtesting (or cheating, tsk tsk tsk) otherwise I can't think of any reason to do it.

Now, these commands would go in the 'execute command' section in the last brackets in a line. Comparative commands to see if a quest is accepted or completed or whatever will look like this:

game.quests[#].state == qs_mentioned

game.quests[#].state == qs_accepted
game.quests[#].state == qs_completed
game.quests[#].state == qs_botched
game.quests[#].state == qs_unknown

Note that other than the extra equals sign these are identical to the executable commands. You may find yourself using 'copy & paste' a lot for some of them: it is easier, and 'safer' than typing out lines that might then contain typos. If a line doesn't work, chances are you forgot to add the extra equals sign: I have done it many times!

There's one other:

game.quests[#].state <= qs_mentioned

I've seen this, I assume it means 'mentioned or unknown'. Interesting :-)

Comparatives can also be 'not', eg:

game.quests[#].state != qs_botched
game.quests[#].state != qs_unknown

Damn handy these! So, lets see them in action:

{10}{Do my new quest?}{Do my new quest?}{}{}{}{}
{11}{I know it}{}{1}{game.quests[120].state == qs_mentioned}{20}{}
{12}{I'm doing it}{}{1}{game.quests[120].state == qs_accepted}{20}{}
{13}{I done it}{}{1}{game.quests[120].state == qs_completed}{20}{}
{14}{I stuffed it}{}{1}{game.quests[120].state == qs_botched}{20}{}
{15}{What was that?}{}{1}{game.quests[120].state == qs_unknown}{20}{}
{20}{Just do it}{Just do it}{}{}{}{game.quests[120].state = qs_accepted}
{21}{Bite me}{}{1}{}{0}{}

Simple. Lets look now at flags and variables.

I have listed which ones I am using in the guide in my mod, and Liv has said which ones she is using. Again, check the modding forum at Co8 to see which ones people are using. Some of them are set aside for internal stuff like the random encounters. Others are simply used as counters for the various characters, flagging when you have done something they will react to or whatever. Any time a character dies for instance, it gets flagged. Just go beyond the ones Liv set aside and you should be fine.

The global flags are straight 'on / off' (0/1) thing. For instance, when Meleney dies, it gets flagged by her san_dying script. Go see Filliken, and his san_dialog script has a thing in it that if that flag is there, the conversation starts with him weeping (and finishes soon after with him attacking you). If she gets resurrected it should reset the flag to off (0).

The global variables are counters, so you can use them the same as flags, only with more than one return. For instance, with my mod, rather than wasting a whole bunch of different flags for all the various items you bring Lila, I just used a variable counter. When you brought something to Lila, the counter went up by one. When it hit 8, it meant you had finished. Of course, to stop people bringing 8 pieces of hemlock, I had to flag that one :-)

To use them, just use the following scripts: (these are examples of course)

game.global_vars[#] = game.global_vars[#] + 1 (increases variable by 1)
game.global_vars[#] = 5 (sets it to 5)
game.global_vars[#] == 0 (sees if it is 0, the default value)
game.global_vars[#] >= 2 (sees if it is greater than or equal to 2)
game.global_flags[#] = 1 (sets flag)

Again, note the difference between the comparative and executable commands.

I should mention, the game seems to save 'quest complete' etc for the .dlg files: in the .py files (the scripts) it prefers to use flags. So it might use a flag to indicate a finished quest. But you can certainly handle the flags and variables in the .dlg files as well as the .py files.

What next? Well, we should have a good look at the other default options for dialogues. We have seen {G:} the default greeting for NPCs, and {E:} the default goodbye line for PCs. Here are the others as Phalzyr lists them:

{A:} = Appreciation Response (Thank You)
{B:} = Barter Response
{F:} = Forget it response
{K:} = More Questions Response
{N:} = "No" Response
{Q:} = Crash To Desktop When Displayed.
{R:} = Any Rumors? Auto asks for money.
{S:} = Sorry Response
{Y:} = Yes Response

Straightforward? Before I go into possible complications, lets see them in action. This is from Liv's addition of the gnoll attack. (Hard woman, that Liv - the only good gnoll is a dead gnoll and if you show mercy, Liv doesn't!)

{5000}{It must have been those gnolls that we encountered back in the moathouse. Maybe paying them off wasn't such a good idea.}{It must have been those gnolls that we encountered back in the moathouse. Maybe paying them off wasn't such a good idea.}{}{}{}{}
{5001}{S:}{}{1}{}{}{}
{5002}{E:}{}{1}{}{}{}
{5003}{Y:}{}{1}{}{}{}

On the screen, this will show up as something like this:

Spugnoir: It must have been those gnolls that we encountered back in the moathouse. Maybe paying them off wasn't such a good idea.
1. I am sorry.
2. I must go.
3. Yes, of course.

Now, the details: the main one to consider is {B:}, because you need to use it to initiate bartering: but you don't have to use the default comment, you can add one.

{11}{B:Ok Lodriss, show me your goodies!}{}{1}{}{0}{}

Note the use of 0 in the line section, you are terminating dialogue and entering barter phase.

The {K:} response is used in the template, you can use it to take you to a 'default' dialogue bit where the player has a few options to talk about different things. Eg, in Jaroo's .dlg, {K:} takes you to line 80:

{80}{How can I help you, my son?}{How can I help you, my daughter?}{}{80}{}{}

{81}{Can you look at something I found?}{}{8}{pc.item_find(3000) != OBJ_HANDLE_NULL or pc.item_find(5800) != OBJ_HANDLE_NULL}{160}{}
{82}{Me find stuff. You look at it?}{}{-7}{pc.item_find(3000) != OBJ_HANDLE_NULL or pc.item_find(5800) != OBJ_HANDLE_NULL}{160}{}

{83}{I am in need of healing.}{}{8}{}{370}{}
{84}{Me have ouchies.}{}{-7}{}{370}{}
{85}{I would like to donate some gold.}{}{8}{}{40}{}
{86}{Me want to give gold.}{}{-7}{}{40}{}
{87}{I am trying to help Mytch the Miller.}{}{8}{game.global_flags[12] == 1 and game.quests[8].state == qs_accepted and game.global_flags[13] == 0}{240}{}
{88}{Me helping miller!}{}{-7}{game.global_flags[12] == 1 and game.quests[8].state ==

qs_accepted and game.global_flags[13] == 0}{240}{}
{89}{I've finally done it, Jaroo. I've gotten Terjon to agree to convert Marek, the carpenter's brother, to the Old Faith!}{}{8}{game.global_flags[17] == 1 and game.quests[5].state != qs_completed}{336}{}
{90}{Terjon say it OK for carpet man brother to join you.}{}{-7}{game.global_flags[17] == 1 and game.quests[5].state != qs_completed}{336}{}
{91}{I have some questions about the Old Faith.}{}{8}{}{200}{}
{92}{Me learn about Old Faith.}{}{-7}{}{200}{}
{93}{Why haven't you sent in your monthly report to Hrudek in the Gnarley Forest?}{}{8}{game.party_alignment == TRUE_NEUTRAL and game.global_flags[18] == 0}{318}{game.global_flags[18] = 1; game.global_flags[67] = 1}
{94}{Hrudey wants report.}{}{-7}{game.party_alignment == TRUE_NEUTRAL and game.global_flags[18] == 0}{318}{game.global_flags[18] = 1; game.global_flags[67] = 1}
{95}{Why don't you help Tarim with the Deklo Spiders?}{}{8}{game.quests[2].state == qs_mentioned}{920}{}
{96}{Jaroo, do you mind if we kill the spiders that have infested the Deklo Grove?}{}{8}{game.quests[2].state == qs_accepted}{920}{}
{97}{Jawoo care about Decklow spiders?}{}{-7}{game.quests[2].state == qs_mentioned or game.quests[2].state == qs_accepted}{920}{}
{99}{E:}{}{1}{}{0}{}

Simple enough. :-) If you have a 'default' dialogue tree, makes things easier and then you can flesh out each branch as appropriate.

Phalzyr also mentions **{C:} = Story state response**. Having scanned the .dlg files, I don't believe this is ever used. But there are still default lines for it, which as Phalzyr mentions can be found in the **mes** folder in **gd_cls_pc2m.mes** and **gd_cls_pc2f.mes**. We'll go into scripting for story state later.

For the moment, that's it! You can now create .dlg files, have them show dialogue appropriate to the situation as decreed by a wide number of parameters, keep track of things with flags and

variables, initiate and complete quests and even initiate bartering or have the NPC attack! There's plenty more, but this is the main stuff.

Enjoy, and check back regularly for more stuff. In a couple days we will look at some more complicated scripts for the .py file.

New PC Voices

Firstly, let me say the new patch is imminent! Just a few things to take care of online before I change drives and finish testing.

So while I am here, I am going to post my step-by-step on adding new PC voices. I want to post all my tutorials up on here, then create an index post and bookmark it on ths side banner so people can quickly access these things.

If you like my tutorials, feel free to check out my ads ;-)

So without further ado...

There are only 2 files you have to alter as far as I can tell to create new PC voices (though I am still keeping an eye out for others to see if i can fiddle with volume, reverb etc: but these two certainly do the job). These files are both called 'pcvoice.mes', one is in the rules folder, one in the mes folder, and both can be found in ToEE3.dat.

Crack them open! (Notepad will do, 'tools' are for sissies.) In the file from the mes folder we have the familiar names of the current PC voices, "righteous warrior", "zealous healer", "raspy", "gruff" etc. The numbers stop at 21 (or in the 50's if you have my new voices pack ;-)) but you can include up to number 63 if you want: with the instructions at the top about how the 2 pcvoice.mes files interact (the one points at the other), its an interesting indication that this was one area that was always intended to be modded. Hey Troika got something right! Imagine what might have been

Ok, so lets pretend we are adding an "ordinary bloke" voice (because I did). So, lets make him #23, we therefore add {23}{Ordinary bloke}

That's what will show up on the creation screen. Now for the other pcvoice.mes file in the rule folder.

Open it up. Here we have bizarrely named things like:
{5}{FHmnRog.mes}
{6}{FHmnSor.mes}
{7}{FHocFtr.mes}
{8}{MDwfClr.mes}
{9}{MDwfFtr.mes}
Etc. You can probably figure out what they mean, the thing to notice is the male ones start with M and the female with F. Follow this. This is why we can't give our characters cross-gendered voices (c'mon, we've all thought of it...)

So, for our ordinary bloke, lets call him 'MBlokey.mes'. That will show up as {23}{MBlokey.mes} The 23 part connects it to the previous file of course, but what is this MBlokey.mes thing?

Go back to the mes folder in ToEE3.dat and you should see a folder called pcvoice (anything to do with this exercise seems to be called pcvoice, you get that.} Open it up, and there are the files FHocFtr.mes, MDwfClr.mes etc. So of course we are going to add one called MBlokey.mes. To do this, simply copy one (lets copy MElfRgr.mes so we are both on the same page) and rename it MBlokey.mes.

Ok, open it and lets get on with the fun part. There are a number of different catagories:

// Acknowledging an order (saying yes)
// Unable to perform an action (saying no)
// Encumbered

// Death Cry
// Near Death
// Death of a party member
// Combat start
// Critical hit BY a party member
// Critical hit ON a party member
// Critical miss BY a party member
// Accidental damage
// Opening a chest with a lot of gold
// Entering a map for the first time by Area Number
// Encountering a boss monster
// Tagged scenery (special areas like Thrommel's room or the big bronze doors)
// Bored...zzz (whining when nothing is happening)
// Using the power of a deity (crying out to various Gods)

I list them all because as u can probably see, MOST do not ever appear in the game. "What a fabulous polished piece of software", you say. I agree, have a cookie. The only ones that do seem to appear regularly are saying yes, saying no, encumbrance, death gurgles, accidental damage, and opening a chest with lotsa treasure.

Why not the area ones? They work with NPC's after all... well, heres a fascinating thing. They DO appear here in the dlg files (that's what these are by the way: this stuff will show up on the screen if you switch on subtitles) but there are NO corresponding MP3s in the sound folders! Why not? I'm glad you asked.

I have added appropriate mp3s to 'ordinary bloke' and the others - I mean they are not exact, but if u have an mp3 that plays "what a nice forest" u don't need to be a member of Mensa to figure out it plays better in the deklo grove than the fire node. However, when I play-tested these, they were very erratic: the moathouse ones played the mp3 for Hommlet as often as for the moathouse, the Emridy Meadows didn't play at all, the Ogre cave mp3 played when I went into the traders establishment, and I forget what played when I went up onto the second floor

of the Wench but since nothing should have, you get the picture. Anyway, I have left them in since it is easy to 'turn them off', just recruit an NPC and they will trump the party vocals as they normally do. Also, I or someone more knowledgable may be able to fix the whole thing at a later date and it will be good to have them there.

Hopefully a fix will present itself, but I have to say, the erraticness of the playing combined with the lack of mp3 files in the PC's folders for this stuff (they stop at the 1100's) suggest to me Troika couldn't get this working at all due to time constraints and scrapped it. Ho hum. Not something a n00b hacker like myself is going to be able to fix in any hurry.

Back to the 'fun part'. Keep MBlokey.mes open, and then get out your new sound files you want to put in. Doesn't matter if they r .wav or .mp3, if they are .wav we can convert them later. You want to put them all in a folder named, u guessed it, 23. This goes in Temple of Elemental Evil\data\sound\Speech\pcvoice, so in the end you will be working in a folder called:
Temple of Elemental Evil\data\sound\Speech\pcvoice\23

Put all your .wav or .mp3 files in there. (If you want to work on them seperately then convert them and stick them in there later, its not a problem). Now the fun begins: and this is fun, but its also dull and tedious. A bit like sex: lots of fun but goes on and on and on... well at least for me ;-). Probably if there was someone else there with me it would be different lol.

Take your first file. Play it. Check MBlokey.mes. Find somewhere it matches. If it is complaining about encumbrance, it goes in the 200's, if it is just agreeing with you, put it at 0 or 1, if it is saying how they don't like deep dungeons, perhaps 1214 is the go.

Now, change the dialogue of the line number to match what you hear. Then rename the file to match the line number exactly.

Example: my Ordinary bloke has a file where he eulogises a fallen comrade with a mangling of Hamlet. So, lets put that at 500. Change the dialogue line and it will now look like this:

{500}{Alas, poor... hmmm, I didn't really know you.}

Then we change the mp3 file to match: we rename it 500.mp3.

Its that easy.

Then we do it with ALL the mp3 files.

If you have lines left over at the end, extras you have not replaced, best to delete them (if they are in the 0-1100 range). For instance, you may notice there are 5 seperate options for encumbrance (200-204). If you only had one mp3 complaining about encumbrance, stick it in at 200 obviously and then delete the rest. Otherwise, the engine will pick one of these 5 lines randomly, and if its not 200, then nothing will sound (it won't crash or anything but nothing will happen, which sorta defeats the purpose).

All that remains now is to convert the .wavs to .mp3 if necessary, and then store the other files in the game itself.

The pcvoice.mes from the mes file goes in Temple of Elemental Evil\data\mes. Also in here should be the pcvoice folder (Temple of Elemental Evil\data\mes\pcvoice) in which to put mblokey.mes. And finally the pcvoice.mes file from the rules folder of course goes in Temple of Elemental Evil\data\rules. These folders should already exist if you have the Co8 patches (but create them if u have to). Folder 23 should already have been created by u since we were working in it: Temple of Elemental Evil\data\sound\Speech\pcvoice\23, remember?

That's that! Remember to download my voices, then put yours after them

Sound and visual FX and spawning from dialogue

Last night I added a little thing at Co8 for the Lareth fight. Absurdly simple, just:

ONE mp3 copied from elsewhere in the game

TWO lines in the dlg file
ONE addition to the py file (a chunk of script)

And now if you beat Lareth down to under 50% hp, when he stops the fight and tries to talk his way out of it, if you continue fighting he will call on Lolth for aid (fully vocalised call of course, like all my NPC mods :-D) and Lolth will send him some allies, complete with summoning effects when they arrive. The point of this is to compensate for the fact you are fighting him alone, having already killed off his boyos, whereas in the original module he would of course have come out and helped and you would have had to fight the squad of evil mercenaries and the higher-than-you-lvl cleric all at once (much tougher!)

Lets go through it step by step, that will teach us:

How to add visual effects
How to add sound effects
How to spawn new critters

There are only 3.

First the dlg file. The two lines I added are:

{700}{Lolth! Protect me!}{Lolth! Protect me!}{}{}{}{game.particles('sp-Curse Water', npc); create_spiders(npc, pc)}
{701}{[attack]}{}{1}{}{0}{npc.attack(pc)}

You should be able to analyse these easily by now. First, Lareth calls out for Lolth, there is a couple things in the 'execute' field, and the second line is the PC response, which is common (1 Int), ends the dialogue (line 0) and initiates combat (npc.attack(pc)). O and i fiddled a few lines earlier to GET to line 700: namely, several lines that otherwise said something like:

{We fight now.}{}{1}{}{0}{npc.attack(pc)}

I changed to this:

{We fight now.}{}{1}{}{700}{}

Two changes, both obvious.

So, now Lareth calls on Lolth before the combat starts. What else happens? The first executable command is a glorious little thing, it allows us to add **game particles**. That's what we call visual effects such as spell effects, monster visual stuff (not the models but things like the Balor's fire and glowing eyes, things like that) other things, anything that looks like a visual effect rather thna a model :-) I think they are sprites but I am not sure.

What are our options for this? Well, to see them all (or even add new ones) open the files **partsys0.tab**, **partsys1.tab**, **partsys2.tab** in the **rules** folder or in **ToEE4.dat/rules** if you have unpacked it (you HAVE to unpack these if you want to mod!! For instance, partsys2.tab is not in the rules folder of your normal game - only modified files are in there normally and we have to access a lot of files modding even though we may not actually modify them). Being tab files, open them in 'Proto Ed' like you would protos.tab (makes it much easier to view them). Now you can see all the visual effects :-) Of course, there is one for each spell.

If you don't want to look at them now, that's ok, we'll keep going. Suffice it to say, **game.particles('sp-Curse Water', npc)** adds the visual effect (NOT the sound or anything, JUST the visual efffect) of casting that spell, so when Lareth calls out, a creepy looking red cross appears over him: you know he is doing more than just squealing ;-).

Notice you can just script these straight in the dlg executable thing? How cool is that?!?! Easy to add stuff.

Now... what do you suppose **create_spiders(npc, pc)** does? Hmmm...

Unfortunately, this requires a touch MORE than just the executable. This actually accesses something already defined in the .py file. So pop that file and lets find it. Go right down and you will see:

```
def create_spiders( attachee, triggerer ):
_________spider1 = game.obj_create( 14397, location_from_axis (470L, 536L) )
_________game.particles( "sp-summon monster I", spider1 )
_________spider1.turn_towards(game.party[0])
_________spider2 = game.obj_create( 14398, location_from_axis (481L, 536L) )
_________game.particles( "sp-summon monster I", spider2 )
_________spider2.turn_towards(game.party[0])
_________spider3 = game.obj_create( 14620, location_from_axis (482L, 529L) )
_________game.particles( "sp-summon monster I", spider3 )
_________spider3.turn_towards(game.party[0])
_________spider4 = game.obj_create( 14417, location_from_axis (529L, 544L) )
_________spider5 = game.obj_create( 14417, location_from_axis (532L, 555L) )
_________return RUN_DEFAULT
```

So here we define the create spiders thing. It is a variant of the scripts Calmert and Terjon use to call each other if you attack them in the church. I also used it to summon the Daemon at the end of Desperate Housewives.

As you can see it repeats a lot so we don't have to go over every line.

Firstly, lets compare:

create_spiders(attachee, triggerer) with
create_spiders(npc , pc)

In the .py file we talk in the abstracts that define the event, in the .dlg we talk about the specific characters who trigger it. You'll get used to that. Certainly you will never see 'attachee' in the

.dlg file, because the .py files are straight Python scripts that get *compiled* by the game when you access them (if they have not already). These are the identical .pyc files in the **Scr** folder. You may have some that don't have .pyc files, they haven't been accessed in your current game yet.

Now, the first line:

spider1 = game.obj_create(14397, location_from_axis (470L, 536L))

This both *creates* a spider, and defines it as a string called 'spider1' we can subsequently use in other scripts. :-) The specific spider is proto number 14397, a small fiendish spider from memory. Spider2 is a medium one, Spider3 a large one, and 4 & 5 are Black Widows for elsewhere in the game (a little surprise). So essentially we are saying, create object 14397 and call it 'spider1'. We could of course create anything and call it anything :-) We could create a Hezrou at this point and call it 'Fifi' if we have such a need.

location_from_axis is important, this says where to put it on the map, but how do we find that value? Well, when you want to add something, you go to the spot with your lead character (the leftmost one in the party lineup shown down the bottom of the screen).

Now, bring up the console (press [shift] and [~]). Type in

from utilities import * [hit enter]

This tells the game we are going to access the **utilities.py** file in the **data**/**scr** folder. That's a file well worth looking at btw.

Next, we ask the game to give us the location we are at. So type in:

location_to_axis(game.party[0].location) [hit enter]

This is a little script added to the utilities.py file by the original ToEE creators for just such a moment. Agetian improved it so you only have to type **loc**, clever fellow! Later you can look for that improved utilities.py at Co8. For now, jot down the number that you see when you hit enter: that will be the location you will later put in your creating coordinates. Game.party[0] is your leftmost character (the 'leader'), then [1], [2] etc. And note we use location_TO_axis to find it but location_FROM_axis to use it. :-)

Next line:

game.particles("sp-summon monster I", spider1)

Well, we know what that does, don't we? It creates the particle effect of the summon monster spell, and the target for it is spider1. Compare this to the previous one we saw with the curse water effect, there the location for the spell effect was npc (ie on Lareth himself). You could also script in coordinates here if you wanted it to appear nearby for some reason, or put in game.party[0] if you wanted an effect to appear on your party leader for instance.

spider1.turn_towards(game.party[0])

We've seen this one before too, it turns the spiders toward at least one member of the party , rather than facing the wall or something when they appear (that looks stupid). Might vary those later so they turn towards different member so the party, or we could even get them to turn towards the nearest, but not quite so easily ;-).

And THAT'S IT! The rest just spawns different spiders in different locations (different proto numbers and coordinates). That's all :-) Spider4 and Spider5 spawn somewhere else in the moathouse, so we don't need to turn them to face the player or show particles. Only needs to be mentioned that these spiders are already flagged 'KOS' in their protos, so we don't have to do anything to make them attack, they will wade right in.

Now, what about sound effects? It's time to do that.

For .dlg files, adding sound is easy. As it is, many characters (particularly PC followers, or at least potential ones like Lareth) come with vocal lines. So we just have to put those in.

Now, if we want, we can put in ANY mp3 file: a spell sound, a trumpet charge, geese squawking, anything. In this case, I simply added a vocal of Lareth actually yelling out "Lolth, protect me!" I could have added some spell effect sounds as well for the summoning, or layered them with a sound editing program or something, but I didn't!

Where did I get a sound of Lareth saying something? Well, if you pop 00060Lareth.dlg and scroll down to the bottom, you will see that he has a few combat sounds that really don't get used. I looked for a plea to Lolth, and sure enough, there is one at line {12080}.

So: I opened **ToEE3.dat/sound/speech**. In here you will find all the vocal lines of all the characters who have voice recordings (not all of the NPCs of course). They are divided into folders according to their .dlg/.py number, so Lareth is folder number 00060. For some reason, Ostler's stuff is just lying around loose in the speech folder. Messy damn programmers!

Open the folder. The mp3s are accessed by line number. So, for {12080} we need v12080_m.mp3. The 'm' means it is the male version of the line, this will be the default if the male and female versions are identical. If there is a seperate female version (saying 'hey lass' or something) it will be v12080_f.mp3 (but in this case there isn't).

We are adding line 700, so we simply copy this mp3, then go to our game folder and open **data/sound/speech**. We need a new folder, so we create one called 00060. Then we open it and go in, and paste that mp3 file. Then right click on it and rename it v700_m.mp3. Its that simple!

If you have my mod installed you can look elsewhere in the speech folder, everything here (at the time of writing) was added by me, because I am the only one who has done this sort of modding so far. You will find:

spell sounds fired from dialogue lines (00295, 00299 etc)
a bleating sheep (00298)
giggling kids (and a howling Cujo) (00228)
and lots of new dialogue made my blending existing rare and unused vocal tracks (00017, 00091, ooo97 etc).

Well, that's if for today! Now your npc.attack(pc) episodes can have some cunning twists ;-)

Custom soundtrack

So, you've just made a whole new map or area or whatever following Jota's "how to make a new map" thread (its a bit more fiddly than he suggests, but it works :-)). And you want to add a custom soundtrack. Well I will tell you what I did step by step.

First I made my new map - we'll call it Frank's house, because that is what it is. It will occur in that empty house north of the brewery. I am using Jinnerth the tailor's house as a template: I may modify it graphically later but I doubt it, since I just don't have much skill with those arty programs (some may have noticed...).

So... my new map is in the maps folder, its called **map1-int30-Frank**. Its name in **map_names.mes** is {5120}{Frank's house} (note I went straight on to 5120 rather than leaving a gap like I normally do, beacuse I couldn't get it working any other way - what I meant when i said it was a bit more fiddly than first thought). I changed other files of course but they are irrelevant for the music.

Anyways, lets get our new music file and put it in. It is mp3, right? Mine is an old Chicago song, I swear it was the only mp3 I could find on my PC, and the good Lord alone knows how it got there (rolls eyes). Anyway, copy it and stick it in the folder **data\sound\music**. I named mine 'testingmusic.mp3'.

Now, lets look at the music files. These are **schemelist.mes** and **schemeindex.mes**. These

are in **ToEE2.dat**.

Crack schemelist. Mine ends at 3100, future generations may go later.

Soooooo...I insert a couple new lines.

§§§ FRANK'S HOUSE §§§

{3200}{music\testingmusic.mp3 /VOL:60 /loop}

Have a good look at this file: notice you can add ambient music too if u feel the need. For now, save, and stick this in the folder data\sound, right along side the music folder you just went into.

Now for schemeindex. I added the line:

{32}{Frank's house #3200}

This, as the rem stuff at the top suggests, points straight into the schemelist.mes file you just modified. Also notice I spelled 'Frank's house' IDENTICALLY to how it occurs in map_names.mes. Dunno if that is important, but lets not tempt fate. Save and stick this file in the sound folder next to the other one.

Finally, go back to the folder in maps (**map1-int30-Frank**). Open it up and go down the bottom to the mapinfo text file. I don't know if this is called or not or just for info but again, lets not tempt fate and be consistent.

Since I copied Jinnerth's house, it will say "soundscheme: 1, 0" - that’s the basic sound u hear wandering around Hommlet.

As u may have noticed in schemelist.mes, it goes in hundreds. Ours is 3200, so of course we

change the mapinfo thing to "soundscheme: 32, 0". Save.

That's all there is to it - simple eh? Modify 3 files, stick in the music and your done.

So... I run the game, start down near the Renton's (seem to spend my whole game in that house), pop the console and enter (after the utilities bit) **game.fade_and_teleport(0,0,0,5120,489,485)** to go to my new map. Presto, there I am.

The Hommlet music fades out, and the painful strains of Chicago's 'you're my inspiration' or something begin to swell. Not too far though - the music is too soft. Obviously I can change that by fiddling the vol setting in schemelist.mes. But the main thing is, IT WORKS!

Proactive NPC characters

Welcome to part 6 of the Dialogue writing tutorial, and what a pretentious name it has!

By 'Proactive', I mean situations in which the NPCs butt in and say their piece (or react) without the PC initiating it. We have seen that generally events have to be triggered by the PC (who is defined as the triggerer in many scripts, or will be the effective triggerer) and as far as dialogue goes, it is triggered by the player clicking on the NPC to initiate dialogue (or selecting talk from the radial menu).

Many of the proactive events are no different, they are still triggered by something the PC does, its just not as blatant as a click from the mouse. An obvious one is when you move near to an NPC such as the sentinel outside the ruined Tower, who speaks as soon as you get near (or any of the NPCs in the opening vignette, again they will start talking as soon as you go near them, you don't have to click on them and you don't get the option of just walking past).

Dialogue-by-click (if we can call it that) is activated by the san_dialog script we saw earlier. This script is accessed by the game when you clearly initiate dialogue, by clicking or selecting 'talk' on the radial menu. Proactive efforts are done by another method. This is the ubiquitous 'heartbeat' method you may have heard reference to.

The heartbeat is a script that fires every couple of seconds (no surprises for guessing where the name came from). It can do pretty much anything you like, but here we are going to consider its use to start a dialogue. Since Liv always does these things the best, lets look at Kent, her kiddy at the start. (His script is py00228kids_off.py)

```
for obj in game.obj_list_vicinity(attachee.location,OLC_PC):
__________if (obj.distance_to(attachee) <= 30 and game.global_vars[702] == 0 and
critter_is_unconscious(obj) != 1):
________________if (obj.stat_level_get(stat_race) == race_halfling):
____________________________game.global_vars[702] = 1
____________________________attachee.turn_towards(obj)
____________________________obj.begin_dialog(attachee,500)
________________elif (obj.stat_level_get(stat_race) == race_halforc):
____________________________game.global_vars[702] = 1
____________________________attachee.turn_towards(obj)
____________________________obj.begin_dialog(attachee,600)
________________elif (obj.stat_level_get(stat_level_paladin) >= 1):
____________________________game.global_vars[702] = 1
____________________________attachee.turn_towards(obj)
____________________________obj.begin_dialog(attachee,200)
________________elif (obj.stat_level_get(stat_level_wizard) >= 1):
____________________________game.global_vars[702] = 1
____________________________attachee.turn_towards(obj)
____________________________obj.begin_dialog(attachee,300)
________________elif (obj.stat_level_get(stat_level_bard) >= 1):
____________________________game.global_vars[702] = 1
____________________________attachee.turn_towards(obj)
____________________________obj.begin_dialog(attachee,400)
________________else:
____________________________game.global_vars[702] = 1
```

_____attachee.turn_towards(obj)
_____obj.begin_dialog(attachee,100)

Well that looks complicated! I barely understand it myself. But it starts repeating itself pretty quickly so it won't take too long to see what it is all about.

First line:

for obj in game.obj_list_vicinity(attachee.location,OLC_PC):

Obviously words like 'for' and 'list' tell us we are going to do a loop. What we are checking for is a list of objects in the vicinity of the attachee's location (ie anything near where Kent is at that moment), and seeing if those objects are PCs. If they are, it moves on to the next line:

if (obj.distance_to(attachee) <= 30 and game.global_vars[702] == 0 and critter_is_unconscious(obj) != 1):

Note 'if' clauses always end in a colon.

In the event that a PC is detected, it checks:

Is s/he close? Actually 30 is pretty far, but remember Kent is MEANT to speak to you as soon as you appear.

Is the critter conscious? Note how it is phrased - is the 'critter_is_unconscious' flag for the object (the PC) NOT equal to 1. I am not sure why that is there: possibly you can come thru from a combat start like CE, NE or CN with noone in a state to talk. But if so, how did you win the combat? No idea, maybe Liv is just making sure or she played through an obscure situation where it happened.

And finally, *game.global_vars[702] == 0*. You will notice further down that when one of the

choices is made, it sets this variable to 1. Once this variable is no longer 0, Kent will never butt in in this manner again. It makes sure he only does it once. (Kids can still butt in though, look further down in the file and you will see they can come and annoy you a fair bit).

If all these conditions are met, it goes on to the next line passage. This is a series of conditions that establishes where the dialogue will start. Here we see why Liv specifically went for a *list* of the PCs and not just checking if there was A PC (or the leader, game.party[0], or whatever). Its so all the PCs could be checked for to see (in order):

if there is a halfling
if there is a half-orc
if there is a paladin
if there is a wizard
if there is a bard.

If that is meant to be from least to most likely - so you get to see obscurer lines first - then I would have put the bard ahead of the wizard myself. If none of these are present then you get the default line.

The lines of dialogue that start, then, are dependant on what is there.

attachee.turn_towards(obj)
obj.begin_dialog(attachee,400)

It is not just 'start at line 1' as we have been doing with simpler examples. Rather, we have the following lines depending on the PC examined:

{500}{Hey, a new kid! Where are your parents? What grade are you in?}{Hey, a new kid! Where are your parents? What grade are you in?}{}{}{}{} *halfling*
{600}{[looks frightened] Um... uh... Are you a giant? You won't eat me will you??}{[looks frightened] Um... uh... Are you a giant? You won't eat me will you??}{}{}{}{} *half-orc*

{200}{Wow! Are you a real knight!}{Wow! Are you a real knight!}{}{}{}{} *paladin*
{300}{Wow! Are you a wizard!}{Wow! Are you a witch!}{}{}{}{} *wizard*
{400}{Wow! Are you in a traveling band!}{Wow! Are you in a traveling band!}{}{}{}{} *bard*

Wonder why Liv didn't use question marks? O well ;-)

So that was all pretty straight forward: if the party goes near Kent (and they will, he is positioned so they will ;-)) then his heartbeat file will trigger an appropriate line of dialogue.

Note that if we didn't care if the party never spoke to Kent again after this, we could avoid using san_dialog all together: Kent would just show a floating line when you clicked on him (like one of the pirates wandering Nulb) but will still have his dialogue triggered by san_heartbeat. The two are completely independant.

Why do some critters have floating lines to cover a lack of dialogue and others (like chickens) don't? It depends whether they are flagged as 'mute' in their object critter flags. If they are, they show nothing: if not, they will show a generic dialogue line.

One more example, something much simpler: the goon who screams 'intruders' in my mod. Now, this is messy and complicated as only a mod can be: the goons are dlg file 00286, but they work off heartbeat py00288 (because I wanted him to speak first, but he was too far away - I should have used Liv's vicinity of 30 thing perhaps) and also contain a dialogue line for the daemon who gets summoned (to save having a whole new dlg file for it - Liv likewise stores other things in the kids' file.) Note that because the goons have their san_dialog set as 00286, when their heartbeat script fires for a dialogue (we are about to see this) it fires dialogue from file 00286 even though it is py file 00288. Interesting! I wouldn't have thought it would work like that, but it does!

The heartbeat is like this:

def san_heartbeat(attachee, triggerer):

```
_________if (not game.combat_is_active()):
___________________for obj in game.obj_list_vicinity(attachee.location,OLC_PC):
_____________________________if (is_safe_to_talk(attachee,obj)):
_________________________________________obj.begin_dialog(attachee,10)
_________________________________________game.new_sid = 0
return RUN_DEFAULT
```

First it checks there is no combat going on (this is common - you shouldn't speak during combat unless it is something very specific, such as Lareth's offer to surrender. To have an NPC want to initiate chit-chat with you while you are hitting them over the head with your flail is silly.) Then it checks for something in the vicinity, a la Kent. It checks again dialogue is possible, starts the dialogue at line 10, and finally changes the heartbeat from 00288 to 0 - the heartbeat has done its job so it is now scrapped (they can have a performance effect, so you only use them if you have to and try not to leave them running). This saves setting a flag as Liv did (well, a variable). Game_new.sid, as you will know from what Agetian posted about this, means the sid (script id, in this case 00288) is reset to a new one, in this case 0 (nothing). It could have been set to something else for some other critter, even as I am using the different py files for the Goons, the baddie and the Daemon. Note that even though Agetian hsa just explained this to us, Liv was using it aeons ago - for instance, she disabled this command in Shenshock's script to make him work better. Always one step ahead ;-)

So the heartbeat for dialogue is easy, isn't it. Since this fires the instant we walk onto the map, we could probably skip the combat and is_safe_to_talk bits since there has been no time for anything to happen. Its that easy.

What other proactive stuff is there? Well, there are the butt-ins. These are when an NPC starts talking even though you are in dialogue with another. Some examples are Monier and Sunom (if a female PC is chatting to Monier, Sunom butts in resenting it) or the back and forth between Lila and her hubby in my mod. Of course Elmo and Otis chatting away to each other when they first catch up is the same principle, or Riana and her sister reacting to each other, or switching from Elmo to Prince Thrommel if you find him in Elmo's presence (or numerous other

NPCs who react at that moment - Zert has the floating line, "quick kill him!" or something). This actually happens a lot in the game though not necessarily a lot in play if you take my meaning.

Lets see what these calls are, ummm, called in the .py files:

py00005 Calmert to Terjon: switch_to_terjon(npc, pc):
py00005 Calmert to Spugnoir: look_spugnoir(attachee, triggerer):
py00017 Meleny interrupts Mona: buttin(attachee, triggerer, line):
py00084 Sunom to Monier: argue(attachee, triggerer, line):
py00091 Elmo to Otis: make_otis_talk(attachee, triggerer, line):
py00091 Elmo to Thrommel: switch_to_thrommel(attachee, triggerer):
py00100 Riana to Jenelda: together_again(attachee, triggerer):
py00108 Bertram to Preston: buttin(attachee, triggerer, line): *(butt-in, Bertram?)*
py00109 Dala to Dick: make_dick_talk(attachee, triggerer, line):
py00112 Murfles to Y'Dey: buttin(attachee, triggerer, line):
py00121 Tower Sentinel to Lareth: talk_lareth(attachee, triggerer, line):
py00170 Take to Ashrem: switch_to_ashrem(npc,pc,line,alternate_line): *(that looks complicated!)*

Etc. I have shown these at length in order to demonstrate there is no single way of doing this, even within a .py script (like Elmo's or Calmert's) there might be more than one thing defined. But they all look pretty similar in action.

I used Monier and Sunom's carry-on as my template by and large because it goes back and forth a few times and is nice and familiar. Lets look at it in detail as a script and see how it is applied. (From **py00085**).

```
def argue( attachee, triggerer, line):
_________npc = find_npc_near(attachee,8006)
_________if (npc != OBJ_HANDLE_NULL):
____________________triggerer.begin_dialog(npc,line)
```

_____npc.turn_towards(attachee)
_____attachee.turn_towards(npc)
_____else:
_____triggerer.begin_dialog(attachee,10)
_____return SKIP_DEFAULT

Simple enough? Again, attachee is Monier, triggerer is the PC, and line is the line number of the dialogue you are GOING to (Sunom's).

Now: first of all, it looks for Sunom and makes sure she is there (the party might have snuck in, killed her, beat Monier unconscious then came back later. Stranger things have happened! Or they might have charmed Monier and be holding this conversation, not in the weavers' house, but out on the parapet of Burne's tower or some other obscure place.) Finding her is simple enough, she is id #8006 so we start with

npc = find_npc_near(attachee,8006)

As you remember from above this *defines* her presence as a variable called npc. The actual finding bit is the next line: having established this concept of 'npc' being near the attachee (Monier) it is either true or false. So the comparative line reads

if (npc != OBJ_HANDLE_NULL):

Common enough proposition: 'null' means no, basically; the search has returned a null result. That is, Sunom is NOT there. So, here we are asking if the proposition 'npc' (Sunom's presence) is NOT equal (!=) to not happening! Confused? Damn double negatives! But that's the way the game operates, you will get used to it pretty quickly.

What happens if it DOES return a null verdict - if Sunom is absent? Jump ahead to 'else', and we see that what happens is

triggerer.begin_dialog(attachee,10)

Note it is the *attachee* line 10, Monier's: his dlg line 10 is the copout:

{10}{I cannot talk now. Please come back later.}{I cannot talk now. Please come back later.}{}{}{}{}

Its good to have a copout line though! They usually do: you might want an NPC to butt in if you have a particular NPC follower in the party, (like Elmo with Otis) but because of the vagaries of the pathfinding, you often initiate dialogue on big maps while your NPCs are standing around far away wondering what to do. Initiating dialogue across sectors like that looks ridiculous and may well cause a crash. So NO SKIPPING THESE BITS.

Anyways, assuming Sunom IS there, the 'obj_handle_null' thing will return false (not equal, !=) and on we go. Next line:

triggerer.begin_dialog(npc,line)

It will go to the npc line number specified - this is the switching bit, because we have defined npc as Sunom.

npc.turn_towards(attachee)
attachee.turn_towards(npc)

This makes Sunom (the npc) and Monier (the attachee) towards each other, so instead of facing the PC when they yellll at each other, they face each other. Nice touch, but I bet you don't notice ;-)

Now, how is it fired in Monier's dialogue? Lets see:

{20}{[no response]}{I was not leering!}{}{}{}{}

{21}{And I...}{}{1}{}{0}{argue(npc,pc,40)}
{30}{[no response]}{Sunom, you are overreacting! I would never...}{}{}{}{}
{31}{I think...}{}{1}{}{}{argue(npc,pc,50)}
{40}{[no response]}{Sunom, mind your manners! She is a customer here!}{}{}{}{}
{41}{Now wait just a...}{}{8}{}{0}{argue(npc,pc,60)}

So between each line it goes off to Sunom, then she sends it back again with a similar script of her own (identical, actually).

The actual call of the script comes from the **PC's** line, because if you fired it from Monier's his would disappear before you could read it and probably they would wizz back and forth so fast you wouldn't even notice ;-) It means you have to say some little thing or have a 'continue' line in the PC's part but that can't really be helped.

It says **argue(npc,pc,40)**. Now, watch carefully: this calls the defined thing **argue(attachee, triggerer, line):**. So hear 'npc' refers to the attachee, Monier, NOT to Sunom who then gets defined as 'npc' in the script seperately. Don't get them mixed up or you will be sending people to the wrong lines!

BUT the line number is still defined as '40' so that will be SUNOM's line number. It will go to line 40 of Sunom's dialogue: lets look at it.

{40}{[no response]}{Please, your eyes nearly bugged out of your skull! And you winked!}{}{}{}{}
{41}{Excuse. . .}{}{1}{}{0}{argue(npc,pc,30)}
{50}{[no response]}{Oh shut up, you hussy! Leave my husband alone!}{}{}{}{make_hate(npc, pc)}
{51}{Hussy? Who are. . .}{}{8}{}{0}{argue(npc,pc,40)}
{52}{Hussy? Me not. . .}{}{-7}{}{0}{argue(npc,pc,40)}
{60}{[no response]}{She is no customer! And don't tell me what to do, Moneir! I have half a mind to tell my father about this!}{}{}{}{}
{61}{Well, half a mind is about. . .}{}{8}{}{0}{argue(npc,pc,50)}

The 'no response' bit for males is of course because you only get this when you start a conversation with Monier and a female PC.

That's about it. Its pretty straightforward. :-)

Adding NPC followers

This blog is being written at work on my lappie! Amazing the things u can achieve once the li'l buggers are in bed :-)

Today, persuant to something I said at Co8, I am going to explain how to add an NPC as a follower. Its ridiculously simple: you just add the following command:

pc.follower_add(npc)

But what sort of tutor would I be if I just told you that? So lets set the scene: first of all, go to **HERE** and download the zip file. It includes new .dlg and .py files for Pishella to make her an NPC follower. Note that the .py file is just to fiddle her dialogue so that when you talk to her in the party she reacts differently than when you just chat to her in Burne's tower, but otherwise you don't need any additional script to get that follower_add command to work - it really is that easy. You don't have to install this btw, I won't be offended ;-) But stick it somewhere you can look at the files.

Ok, where does it all begin? Well, go to line 30. This is where she would normally say, "Nice to see you again, I am busy" and that would be that. But, I have added lines 34 and 35 asking her if she would like to join. Then, it jumps off to line 200: she normally ends in the 170's (and some of that is added for the Burne's puzzles quest) so going to line 200 gives us plenty of room to add whatever we like.

200 is her agreeing to sign up: we then have to address the issue of the party being full. Hey,

look at that! I put the first conditional command, in line 201, in the wrong place! It should be in the middle, not at the end (in the usual place for conidiotnal arguments we have discussed before). The last brackets are for executing commands: the game will ignore **not pc.follower_atmax()** here because it doesn't execute. But the game may NOT ignore the subsequent atempt to add a follower, and if you follow this line with a full party, you will probably get a crash! (Based on what happens when NPCs such as Fruella push their way into the party when it is at max and the player is using PC Count Fiddler so the game doesn't realise it is at max). So if you do want to use this script, change that ;-)

The conditionals that fire if the party is at max or isn't are of course there to prevent such a disaster. That is really the only essential thing you need: that and a subsequent script to remove the character. We'll look at that in a minute. But first...

Just to add a bit of variety and spice (and because Cujo asked me to when I wrote this thing for him) I have added a thing whereby Burne charges you 50gp to hire her. I also added some dialogue such that Pishella explains this as Burne being a greedy sod - I think this is consistent with the image of him in the game, what with the amount of treasure he demands (I mean yes the guy is high level but he has a dragon hoard dagnabit! Well, he *reputedly* has a dragon hoard... never believed a word of that. Hey, why doesn't Burne have a beard like in the module? Never understand meaningless changes like that... like at the Argonath when they have one of the figures carrying a sword instead of an axe, the book CLEARLY says they have axes, why change it? Ok ok we are slightly off topic...

So, another new command to learn. In fact a couple, since we have to get the players to pay, AND we have to make sure they have the money! So...

pc.money_get() < 5000
pc.money_get() >= 5000

Simple, isn't it? You've probably noticed the **get** command is often used to find out the values of this or that, stats or flags or whatever. It just gets the value without changing it or anything.

So, if the party has the money, they have to hand it over. To do that we use:

pc.money_adj(-5000)

the excessively bright among you will have figured out that this command will also ADD money to the party - just leave out the minus sign! In fact it adds either way, just here it is 'adding' negative 5000.

SO what is the currency? Well, all currency transactions in the game, be they getting money or adjusting as here or the value of items recorded in the protos.tab, its always in COPPER pieces. So here, 5000 copper is 500 silver is the 50gp Burne wants. Its that simple. NOte we are not transferring the money to Burne, we are just knocking it off... we could give it to him, but transfering items is the topic for next time :-) And this way is easier ;-).

Speaking of easier, I couldn't be bothered doing a Burne butt-in where he himself asks for the money. If you really want to enhance the experience, though, you can do that yourself because we discussed this last time. And, you can even voice it, becuase he mentions 50gp somewhere! Its the amount he pays you for exposing Jayfie for memory. Add that - Burne butting in to ask for 50gp to let you hire Pishella, in the actors actual voice - and you would have an experience that you would swear blind came straight out of original game :-). Meh, that's what I'm here for.

What next? Well, there is the recruit line itself: we saw that at the beginning so I won't repeat myself (its right there in line 230) but will simply point out it fires from the NPC's command thing (does it work from the PC's? Does it matter? It works from the NPC's so use that!) Also, she joins the party without ending the conversation. She will leave that way too: that's how Liv did the 'equipment fix' thingy whereby she prevented NPCs selling stuff you gave them: in the middle of the conversation, unnoticed by the player (going on in the .py scripts) she would boot the NPC out then instantly rehire him or her. Thus, whatever they had at the time would become their 'starting equipment' that they refuse to give up. Ingenious solution! But it meant

the NPC went to the end of the party (not a problem) and forget all their spells (a problem) so it was not perfect and some people have moved on to other solutions, such as Drifter's masterful .dll hack.

That everything? Well, what about the dialogue? We have fiddled the san_dialog bit of the .py script so that when you talk to her in the party, she will have different dialogue than when she is standing around in Burne's tower. Lets take a quick look at it:

```
def san_dialog( attachee, triggerer ):
_____attachee.turn_towards(triggerer)
_____if (attachee.has_met(triggerer)):
_____________if (attachee.leader_get() == OBJ_HANDLE_NULL):
_____________________triggerer.begin_dialog( attachee, 1 )
_____________triggerer.begin_dialog( attachee, 250 )
_____else:
_____________triggerer.begin_dialog( attachee, 1 )
_____return SKIP_DEFAULT
```

I just nicked this one from Elmo and copied it over the existing one. For one thing, it now turns her to face you :-)

Ok, the thing **(attachee.leader_get() == OBJ_HANDLE_NULL)** is something you will see regularly with NPC followers. What it does is it 'gets' the condition of whether the NPC is in the party, expressed as whether they acknowledge the party leader. Don't ask me why or what that means, just take my word for it, that's how you test if the NPC is in the party or not. It tests whether this condition returns NULL, that is, if the condition is NOT present: again with the negatives! So, if it is true, if 'getting' the condition of the NPC acknowledging the leader IS null, that is, returns false - ie the NPC is NOT in the party - then it will go to line 1 as usual, and the game can then use the existing 'npc_has_met' scripts to handle where to send the dialogue (since if they have not met they go to 1 also in this script). Its a pain working to get a logical NOT using '==' rather than '!=', but take a moment to understand this since some scripts will do

it the oppsoite way and have '!=' to indicate that the NPC *IS* in the party.

If she doesn't return a null - if she IS in the party - then she goes to the appropriate dialogue I have written at line 250. Its just 'I want you to leave'. If you use this script, of course feel free to flesh all this out - in fact, I recommmend it :-)

Last thing: booting her out. Firstly, you are going to need her later so we can't just leave her down the bottom of the Temple or something. So we add a condition forcing you to be on the right map: this is

npc.area != 1
npc.area == 1

Area 1 is Hommlet: it refers to the area on the world map. So this makes sure you are somewhere in Hommlet before you can boot her. We COULD do it by specific map - see, for instance Zert's .py script where he may act in certain ways when entering certain maps. There's a san_entering_map or san_map_change or something thing that can be triggered, actually we will look at that next time becuase I am going to put that to an abstract use in KotB to improve the NPC experience, but for now it is enough to know it can be done. I don't have the whole damn game with me here at work so there is only so much i can do from memory ;-). Zert's thing will have the specific python script to check what map you are on, others will too (Mickey, for instance - he knows whether to act like he is in the brewery or in Nulb by checking the map number). Technically I guess we would have to allow Pishella to leave on a number of different maps - any level of the towr - so just saying 'area 1' is so much easier. Will she go back to the Tower on map change? I don't think she has a jumppoint to do it, so I see no reason whyb she should. So let her go IN the tower if you don't want problems later, and make sure she is standing near Burne so he can find her nearby if he needs to (though I think she does the 'destroy orb' thing all by herself).

And the actual script to boot her from the party?

pc.follower_remove(npc)

Again, it fires from the NPC's line, and doesn't terminate the conversation.

Ok, that's that. Now you know how to add Pishella, you can create some NPC followers for the Green Gryphon in the Keep! Get modding.

O and if you expected Pishella to be a wizard - not today. She is a fighter! LOL! Figure that out!

Transferring items

Ok this is a bit fiddly, by which I mean I can't always get it to work. So rather than explaining things I will simply detail methods that have worked for me and a few that haven't but are used in game or by Liv.

When I first started on *Desperate Housewives* I wanted the NPCs you were fedexing things from to actually have the gear in their possession, in their inventories, and then transfer it over to yours when the moment came: hence evil characters who couldn't be bothered running around could always say lop Myella's head off and just grab the cauldron that way. Wouldn't always have been appropriate, eg Sunom or Clarisse, but for Mandy or Paida or Glora it would have been fine.

BUT when I tried to do it, I just couldn't get it to work - the transfering, that is. Can't remember if I could even get it into the characters inventory, probably I couldn't because I wasn't mobbing it in, I was just writing it into the InvenSource.mes. Well, I was a n00b back then and didn't know what I was doing. But I had to do the far simpler method of just creating stuff in the PC's pack when the moment came.

The command for this is:

create_item_in_inventory(item #,recipiant)

Using this is relatively simple, here's a cut-down version of the one where Lila gives u her list of stuff to get:

{95}{[Lila hands you some parchment]}{[Lila hands you some parchment]}{}{}{}{create_item_in_inventory(11099,pc)}

No scripting needed in the .py file, just use it straight from the dialogue executable.

The only fiddly thing you might find is regarding the id#, if the proto number and the id# is different. Generally we do things by proto number because we tend to examine and manipulate the protos, but the id# number may differ. That is, the relevant item number may be from col 22 in Proto-Ed rather than the 'id#' column 23 which will generally match the proto number itself. Confused? Yes, its a pain, and it gets worse when trying to deal with NPCs beacuse they may have 3 different numbers Just be aware of it, and if you are making new protos, make the numbers match dagnabit!

So if you are using a script like this and it doesn't fire correctly, try using a different number. As a general rule, I would say use col 22 FIRST, and if it doesn't have a number there, add one - that is, repeat the proto number there from col 23 in col 22 (and save the protos.tab of course ;-)) If something else is already using that number somewhere, then you will get something different showing up when the transfer happens and you will know that is what the problem is :-) Then you have the painful job of trying to come up with a new number that's not used by anything else but is still somehow relevant to the item u r using. Try the search function in Proto-Ed to find a spare number, and have fun! ;-)

Could you use the **give ####** command instead? I have no idea! I don't see why not, give it a try and leave a comment here and attain modding glory! :-D

What about transferring from the PC to the NPC? Now, that bit definitely works, that's how we moved all that crap you collected for Lila over to her. Lets have a look at it in action:

{148}{So what do you have for me?}{So what do you have for me?}{}{}{}{}
{150}{Bottle from Glora.}{}{1}{anyone(pc.group_list(), "has_item", 12892)}{174}{party_transfer_to(npc, 12892)}
{151}{Hemlock.}{}{1}{anyone(pc.group_list(), "has_item", 5800) and game.global_flags[697] != 1}{160}{party_transfer_to(npc, 5800)}
{152}{Eye of newt.}{}{1}{anyone(pc.group_list(), "has_item", 12897)}{162}{party_transfer_to(npc, 12897)}
{153}{Bat's wing.}{}{1}{anyone(pc.group_list(), "has_item", 12896)}{164}{party_transfer_to(npc, 12896)}
{154}{Your small cauldron.}{}{1}{anyone(pc.group_list(), "has_item", 12894)}{166}{party_transfer_to(npc, 12894)} // Mabel
{155}{Your curette.}{}{1}{anyone(pc.group_list(), "has_item", 12895)}{168}{party_transfer_to(npc, 12895); schedule_reward(npc,pc)} // Morgause
{156}{Your distilling apparatus.}{}{1}{anyone(pc.group_list(), "has_item", 12893)}{170}{party_transfer_to(npc, 12893)} // Sunom
{157}{Your crystal skull.}{}{1}{anyone(pc.group_list(), "has_item", 12898)}{172}{party_transfer_to(npc, 12898)}
{158}{I'll just get on back to work.}{}{8}{}{0}{}
{159}{Me keep looking.}{}{-7}{}{0}{}

Here's a couple new commands:

anyone(pc.group_list(), "has_item", ####

That’s incredibly handy, makes sure you don't have to have the item in the inventory of the person doing the talking, which is annoying for the player (though you CAN do it by specific party member if it is called for, different command). The other one is our transfer command:

party_transfer_to(npc, ####)

Simple, init?

While we're describing transfering, lets have a look at the scripts that didn't work for me but do work elsewhere in the game. Again, I couldn't get them working but that may be because I wasn't properly putting things in the inventories (NPCs spawned by a mob rather than a script will have their inventories mobbed in, not from InvenSource.mes. I couldn't do that at the time). Now, as with other things, there is more than one way to do it. I tried by a couple different methods, one of which was the one Liv used to transfer stuff, and I wrote to her asking for advice and asking why she used that specific script rather than another one. Unfortunately I am such a spammer (you may have noticed) that I have long since deleted her reply (I have to keep deleting my PMs or they would go over 100 very quickly) but the gist of it was, you use what works and don't worry if there are 6 other ways to do it, if what u r doing works, then do it. Sensible girl :-)

So for starters lets look at the stuff for tranfering items from the NPC followers when you buy their starting gear. This is very handy since it deals with getting rid of items as well. Note this is NOT a transfer per se, as you will see:

```
def equip_transfer( attachee, triggerer ):
_____itemA = attachee.item_find(6011)
_____if (itemA != OBJ_HANDLE_NULL):
_____________itemA.destroy()
_____________create_item_in_inventory( 6011, triggerer )
_____itemB = attachee.item_find(6016)
_____if (itemB != OBJ_HANDLE_NULL):
_____________itemB.destroy()
_____________create_item_in_inventory( 6016, triggerer )
_____create_item_in_inventory( 7001, attachee )
_____return RUN_DEFAUL
```

You will quickly recognise what is happening here is the item is being destroyed in the NPC's

inventory (in this case, Elmo) and recreated in the triggerer's inventory (the PC). Then, Elmo gets some money - item 7001 - created in his inventory, where it has already been removed from the PC's possession in the dialogue using one of the scripts we have seen before, **pc.money_adj(-500).**

Why destroy and recreate? Well, its Elmo's stuff, you can't just transfer it! The game will resist a transfer just as it resists you trying to drag and drop from their inventory to yours. For another example: I had a funny situation (also noticed by others) where Furnok looted GKR's head. Now, when I went to talk to Glora, she did the **anyone(pc.group_list(), "has_item", ####)** business, saw it in Furnok's inventory, and we carried on quite happily, she gave me the bottle. BUT when I checked Furnok, he still had the head! Because he had resisted the transfer - in this case, the game had simply carried on and ignored this script since it couldn't be executed.

Anyways, I threw in that about Elmo to explain this and so you could see **item.destroy()** in action: its a simple enough command you use to get rid of something when u r fininshed with it. But what about a real NPC - PC transfer? Well, Black Jay does one:

{140}{Oh, please please let me have it. I will trade you for it! I have magic +1 arrows, elven boots or an elven cloak.}{Oh, please please let me have it. I will trade you for it! I have magic +1 arrows, elven boots or an elven cloak.}{}{}{}{game.quests[4].state = qs_accepted}
{141}{Give me the +1 arrows.}{}{1}{}{150}{party_transfer_to(npc, 3000); npc.item_transfer_to_by_proto(pc,5006); game.quests[4].state = qs_completed; npc.reaction_adj(pc,15)}
{142}{Give me the boots of elvenkind.}{}{1}{}{150}{party_transfer_to(npc, 3000); npc.identify_all(); npc.item_transfer_to_by_proto(pc,6057); game.quests[4].state = qs_completed; npc.reaction_adj(pc,15)}
{143}{Give me the cloak of elvenkind.}{}{1}{}{150}{party_transfer_to(npc, 3000); npc.identify_all(); npc.item_transfer_to_by_proto(pc,6058); game.quests[4].state = qs_completed; npc.reaction_adj(pc,15)}
{144}{You can just have it.}{}{8}{}{150}{party_transfer_to(npc, 3000); game.quests[4].state = qs_completed; npc.reaction_adj(pc,30)}

{145}{I don't want to trade it.}{}{8}{}{160}{}
{146}{Here, you take for free.}{}{-7}{}{150}{pc.item_transfer_to(npc,3000); game.quests[4].state = qs_completed; npc.reaction_adj(pc,30)}

Notice a couple things about this. Firstly, it uses BOTH **party_transfer_to(npc, ####)** and **pc.item_transfer_to(npc,3000)**. Why different commands? I'm sure I don't know, but be aware, again, there are different ways of doing things.

The actual NPC - PC transfer is a fabulous command:

npc.item_transfer_to_by_proto(pc,####)

No frigging around with id #'s here - it tells you straight up it is dealing with the proto number itself. Hooray! Gotta be happy with that.

Also, notice something subtle: **npc.identify_all()** before handing over the Elven boots or cloak. So when u get it, it is already identified, doesn't just turn up as a 'magic cloak' since u should already know what it is. Nice touch.

Ok, that'll have to do for the moment, I have to pack my bag! Enjoy.

Adding chatter to NPC (unfinished)

Today we have a script, its long and repetitive but it will do something rather joyous: it will very easily allow us to add chatter from (or even between) NPC followers, a la BG and such other games as have it. It is a work in progress due to a snag I hit, then shoved it on the back burner to work on the Pishella thing, but that is dealing with associated voice issues so its still slowly getting there. This will all be finished when the Starter Pack is re-released.

The way we are going to fire our chatter is from san_new_map. If a script is attached to this, it will fire each time the NPC changes maps (funnily enough). Usually this is done to check if the

NPC is to do something specific at that map - for instance, leaving the party (Thrommel, Morgan etc) or betraying them (if you've never had this happen in ToEE, it only occurs in very specific circumstances, I won't spoil it by saying when). What we are going to do is, on specific maps (the Keep itself, the wilderness map outside the Caves of Chaos and a few others - maps where the player will first off be moving for a period rather than fighting or talking to someone) a random number will be generated. If it is in the range we want (we don't need this to happen all the time, it would get very old quickly), then the nearest party member to the NPC will be examined, and the NPC will then make an appropriate comment.

Sound confusing? Well, I have done one up for a character called 'Two Swords', one of Cujo's PCs that he has graciously allowed me to use in KotB as an NPC follower. Two Swords is everything we want in a lass - skilful, voluptuous (in a toned way, of course), rough and tough etc. She likes jewelry and the furry animals that follow rangers (a bit of projection from His Dogginess perhaps?), she dislikes goody-goody Paladins and Druids babbling about balance.

IMPORTANT NOTE: The san_new_map column in ProtoEd is MISLABELLED, it is actually 2 cols to the right, labeled san_join or something.

Anyway: lets look at the script, as I said it was long and repetitive but people who like it can just copy it easily enough.

```
def san_new_map( attachee, triggerer ):
_____randy1 = game.random_range(1,12)
_____if (attachee.leader_get() != OBJ_HANDLE_NULL):
_____________if (attachee.map == 5129) and randy1 >= 10:
_____________________attachee.float_line(975,triggerer)
_____________elif (((attachee.map == 5121) or (attachee.map == 5137)) and randy1
>=9):
_____________________for obj in game.obj_list_vicinity(attachee.location,OLC_PC):
_________________________if (obj.distance_to(attachee) <= 30 and
critter_is_unconscious(obj) != 1):
```

```
_____if (obj.stat_level_get(stat_race) ==
race_halfling):
_____attachee.turn_towards(obj)
_____attachee.float_line(900,triggerer)
_____elif ((obj.stat_level_get(stat_race) ==
race_halforc) and (pc.stat_level_get( stat_gender ) == gender_male )):
_____attachee.turn_towards(obj)
_____attachee.float_line(965,triggerer)
_____elif (obj.stat_level_get(stat_level_paladin)
>= 1):
_____attachee.float_line(990,triggerer)
_____elif (obj.stat_level_get(stat_level_sorcerer)
>= 1):
_____attachee.turn_towards(obj)
_____attachee.float_line(930,triggerer)
_____elif (obj.stat_level_get(stat_level_bard) >=
1):
_____attachee.turn_towards(obj)
_____attachee.float_line(945,triggerer)
_____elif (obj.stat_level_get(stat_level_cleric) >=
1):
_____attachee.turn_towards(obj)
_____attachee.float_line(925,triggerer)
_____elif (obj.stat_level_get(stat_level_barbarian)
>= 1):
_____attachee.turn_towards(obj)
_____attachee.float_line(950,triggerer)
_____elif (obj.stat_level_get(stat_level_monk) >=
1):
_____attachee.turn_towards(obj)
_____attachee.float_line(935,triggerer)
```

```
____________________________________________elif (obj.stat_level_get(stat_level_ranger) >=
1):
____________________________________________________attachee.turn_towards(obj)
____________________________________________________attachee.float_line(970,triggerer)
____________________________________________elif (obj.stat_level_get(stat_level_druid) >=
1):
____________________________________________________attachee.turn_towards(obj)
____________________________________________________attachee.float_line(920,triggerer)
____________________________________________elif (obj.stat_level_get(stat_level_rogue) >=
1):
____________________________________________________attachee.turn_towards(obj)
____________________________________________________attachee.float_line(955,triggerer)
________return RUN_DEFAULT
```

Crikey, that's hard to read like that! O well, lets hack our way through it anyways.

First off, we generate a random number between 1 and 12. We call it 'randy1'.

Next we make a quick check to see if we are entering the pub (map 5129). If we are, and randy1 is more than 10, then Two Swords makes a quip about how its good to be back at the pub.

If we are at map 5121 (the main Keep) or 5137 (the front gate) and Randy1 is more than 9, then we have the room and opportunity to make a comment.

So, now we check the nearest party member. If it is (in no particular order):

a druid
a paladin
a barbarian
a fighter

a thief
a halfling
a male half-orc
a ranger
a monk
a sorcerer

then she makes a comment (these will later be expanded to include animal companions and wizards carrying more than 3 scrolls). If not, or if none of these are nearby (she is surrounded by wizards with no scrolls and female half-orcs) then she may make another comment, again randomly determined:

on the party alignment
on her jewelry situation.

Haven't done that bit yet. The alignment one is obvious enough, and will later be expanded to include reputations: she will respond to what they party are actually doing, not just the alignment. The jewellery one will depend on what she has in certain slots (the worn necklace and worn rings slots). If she has some expensive jewellery (established by using the get command to find the value of the item) or magical items (established by using get to find the magical flag), she will purr about it. If she has some cheap crap, she will complain about it, and if she has nothing, she will make it clear to nearby male party members they are falling down on the job.

Simple, eh?

Ok it is long and annoying but it can be copied easily enough and the arguments can be changed. So by all means use it for a new NPC! Of course, for NPCs who join in pairs (like Turuko and Kobort), or who run into other NPCs they know of or would react to in some way (like Taki and Ashrem, or Otis and Elmo), some of these lines can trigger butt-ins and exchanges (again, a la some of the stuff in BG). The hardest part will be getting voice actors:

without voices to accompany the floatlines, it really doesn't work at all.

Now, other than the expansions for other comments, what does this work in progress still have to do? I have to come up with a way to *delay* all this til the NPC is a few seconds into the map. This will allow the party to break up a bit (otherwise the NPC will ALWAYS comment to the same character, unless u change the party order - that gets old quickly). Of course, it will only be if the NPCs are free to chat and not in combat ;-). The time interval will differ for each NPC: by changing the amount of time each, erm, time something fires, even if more than one NPC gets lucky with randy1, their comments will fire at different times, rather than talking over the top of each other. I have tried to implement this using the add_time thing but have had no luck getting event the simplest script to fire from it (despite having used this successfully in the past in *DH* for Gwenno's reward - well, sorta successfully ;-)) Back to the drawing board!

Now, what exactly are Two Swords comments? You'll have to wait and see 8^P

Subsequent update: I fixed the timed chatter thing, the problem was I added a san chatter() thing but didn't realise I needed to define it with args (attachee, triggerer) because I am a n00b.

Creating new NPCs with ToEEWB

This is what ToEEWB is for, and there are instructions for it in the tutorials, but it is one HUMUNGOUS tool capable of handling almost every aspect of the game, and finding something specific in it can be a li'l daunting. In fact, I can honestly say I expected someone to ask this, because while those of us who muddled along before ToEEWB came along were able to recognise its power and start familiarising ourselves with it, (because we knew what we needed to do and saw how it helped), someone just starting could get spooked. So, today we are going to add a new character to KotB (the Provisioner, whose dlg I did while over in Malaysia) using ToEEWB. And by 'add a new NPC' I don't mean hack it into the protos.tab - you should be able to do that! - but I mean put it in the game so it spawns and can be interacted with.

But first things first. Yes, you should be able to make NPC protos, but if you can't, a quick guide:

1) Pick either ToEEWB or ProtoEd to make it in. I recommend ProtoEd, but ToEEWB is far more powerful, more accurate (some of ProtoEd's columns are wrong) and lets you do things like copy existing ones better.

2) Get your tool of choice (if u have been following my tutorials you should have both, but otherwise, ProtoEd is **HERE** and the latest ToEEWB is **HERE**).

3) Back up your protos.tab before fiddling with it!

Now, do u wanna make a new one or mod an existing one?

To make a new one, you have to open Description.mes in the data/mes folder and put it in. Stick it in the 14000's, where the NPCs go. Then, power up your tool (oo-er). To add a new line to ProtoEd, go to the appropriate number - say, 14888 (or where the appropriate number SHOULD be, in this case, between 14705 (a trader from Allyx's shopmap) and 14996 (a mystery critter for my next little "get-more-of-Cujo's-stuff-into-the-game-without-consoling" effort)). Right-Click on one of the boxes in 14996 and hit "insert row above" from the menu. Shazzam! A new row appears. Double-click the leftmost box, put in 14888, and go for it: the entries are straightforward for a basic proto, more advanced entries can be considered by looking at stuff in the General Modding thread or asking questions there. But to add a new NPC with stats, classes, spells even, feats etc, u just gotta look at what is above, its not that hard. Don't forget to save, and to copy your new protos.tab and description.mes into the root folder of ToEEWB so when u run it in a minute, it can access your new proto.

To do it all in ToEEWB, well, really, read the tutorial *Building a New Module*, section 27 onwards, where u learn the mysteries of the 'add new proto' button ;-)

Easy. Now, suppose we have our NPC all done (and u have spawned him from the console in-

game a few times to test him out). How do we get him to turn up in a game automatically?

There's a few ways. Firstly, you can have him spawn from a critter's dialogue thingy, that's the easiest. We saw that at **THIS** tutorial, where we spawned spiders from Lareth's dialogue. Simple, but obviously limited in its applications:

Orc Leader: Now you have upset me!

1. [continue dialogue to spawn rest of tribe]

Not that appropriate... but still has its moments (eg its how I bring in the Corporal and Scribe at the Gate in KotB. Must add a little postern door to the Tower for them to have come from).

The next way to do it is spawn from heartbeat. This is how we used to add things in the pre-ToEEWB days. A simple example of this is how I added the sheep and rooster for the original "Something about Mary" mod. Did I ever blog about that? Can't remember... nah I don't think I did, but I shoulda. Here's the script I used:

```
def san_heartbeat( attachee, triggerer ):
_____if (game.global_vars[697] == 0):
_____________game.obj_create(14362, location_from_axis (507, 484))
_____________game.obj_create(14365, location_from_axis (505, 478))
_____________game.global_vars[697] = game.global_vars[697] + 1
_____return RUN_DEFAULT
```

The heartbeat goes off the moment u enter the room (yes, I could have used 'first heartbeat' for it too). Then the rooster and the sheep spawn. BUT you don't want them respawning at every heartbeat, so u only do it when a certain flag is at zero, and the instant it happens u increment the flag so it is never at zero again. Simple. (And yes that could have been done by using game.new_sid = 0 instead of wasting a flag, its already on the list of things to be done for *DH2* (current release date - a long way off)).

A very handy if superseded manner of doing things. In this case, it was fool-proof - even if u entered the room with naked steel in your hand and cold murder in your heart and killed Mary instantly, there was no way u could stop that heartbeat - it would always spawn then go away. And Mary was always going to be there the first time u went in. Soooooooo, by influencing her heartbeat, u could make sure the sheep and rooster were always there too.

But what about adding a critter where there is NOTHING there to add a heartbeat or dialogue to - such as Clarisse in DH? (She, like Frank, I added partly to give you a reason to go into those otherwise empty houses - with Frank, I had to add the house interior too). Empty places cry out for NPCs in them, but also by their nature resist these two methods. And how did Mary spawn in the first place for that matter?

The answer to that is .mobs, or mobile onjects. To find out more about these, read the *Mobile and Static Objects* tutorial that comes with ToEEWB.

Mobs live in the map folder for each map, which in turn lives in the module folder (maps of course change from module to module, where other stuff stays the same). So, Mary's .mob is in

Atari\Temple of Elemental Evil\modules\ToEE\maps\Map-9-Nulb-Interior-Snake-Brothel-2nd-fl

(well, theoretically - actually, it is locked up in the .dats!)

and Clarisse's is fired from a heartbeat attached to a new .mob in

Atari\Temple of Elemental Evil\modules\Co8-4.0.0\maps\map1-int25-herdsman-house

Why make a .mob to fire a heartbeat to make a character? Well, prior to ToEEWB, that's how it had to be done: we could only make the most simple of .mobs, and we made one that was

simple (and invisible etc) and gave it a heartbeat and then spawned stuff off its heartbeat. Complicated, and the invisible .mobs, despite all our efforts, still did stuff like turning up in combat if you let off a fireball near them. One of ToEEWB's great achievements is that it allows us to easily make complex .mobs (such as NPC's with merchant inventories, flags set etc). Its other great achievement in my opinion is its ability to add sectors to maps - without it, we could NOT sector new maps (it simply can't be done my hand, I reckon). I suspect there is other stuff that I haven't even discovered yet ;-)

But on with the show. Have you still got ToEEWB powered up? Choose your proto then, and lets make a some .mobs! I will be using 14018, which in KotB will be Porteous, the Provisioner.

Firstly, we stay on the opening page of ToEEWB (the objects page) and click 'new'. Now we have a new .mob (albiet blank). Its identified by the randomly generated Object GUID, which as u can see is so damn long it can randomly generate over and over and never come up with the same thing twice.

Next, we pick our prototype. Click the menu and scroll down. Note this is alphabetical and goes by the name in the description.mes, not necesarily the name u know. That is, I will be looking for 'Provisioner' not Porteous, becuase the prototype is for a provisioner - I only learn he is named Porteous if I chat to him in the game. If you don't know how this is done (its by setting an id number different to the prototype number in col 23 of ProtoEd) then you didn't do it when u made your prototype, and you don't have to worry about it! :-) But if u r ever looking for a prototype alphabetically and get lost, go over to the tab at the top, "Proto Descriptions", and you will find them by number just like in ProtoEd and description.mes.

Find your proto and click on it - note it automatically calls itself a obj_t_npc (you can't change that even if u want to!)

Next is location. Do u know where u want to put it? If not, make sure ed.py is in your **data/scr** folder (if not, its in **ToEE World Builder\Tools\Editor's Assistant**, yet another handy little tool thanks to Agetian) then power up the map in question and walk your leader (left-most party

member, #0) to where you want to spawn the new character. Then, open the console and type in

from ed import * (then hit [enter] - that loads ed.py)

loc() ([enter])

Write down those numbers, they are the location, then put them in the X and Y things of location. Fine tuning can be done with X and Y offsets (put in 10 or -10 or something to move them slightly, requires tinkering to do).

Note: .mobing things in is FAR more locationally accurate than spawn-by-script. Take the pix of a guard by a bell u may have seen in KotB screenshots: I simply could NOT get him to stand under that bell by console-spawning, it was too close to the wall, despite the fact I had walked under there to get the location with loc(). But the mob placed him exactly where he should be.

For rotation, we can cut-n-paste from Agetian.

5.5 - northwest
5.0 - west
4.0 - southwest
3.0 - south
2.5 - southeast
2.0 - east
1.5 - east (sometimes it looks better than 2.0)
1.0 - northeast
0.5 - northeast (sometimes it looks better than 1.0)

This is the way he will face when he spawns. I'm choosing 3 for the moment.

Other than that, we just have to hit the 'dispatcher' box. Do that for NPCs (though not for other

stuff). I forget why, but that's what Agetian said :-)

Now... on the right, you should see some flags. By all means look at them, but do NOT be tempted to click the activator thing at the top (in this case, should be Object Flags) to see them better. As far as I understand (from trial and ERROR) this sets up part of the .mob to hold these flags - once it does, it will crash the game when that .mob is introduced if those flags are the wrong sort (EVEN IF U DON'T SELECT ANY OF THE FLAGS!), and you CAN'T return the .mob to its original state once it is set up for flags (by un-clicking the top box). You'll have to throw it out and start again.

Shouldn't need any Object Flags for your NPC (I don't!) so we'll move on.

Secret doors? Nah.

Scenery? Nah.

NPC/critters? Bullseye!

Does your critter have any inventory? A monster with no loot may not (or a whore in the SnakePit, pre SaM ;-)) but if they do, click "Inventory (Loot/Worn)" - distinguishing it from inventory a merchant sells - and start adding whatever you want. To do so, select the pull-down menu and find what u want to add, then click "add new" (add existing will show you a menu of stuff already in the **ToEE World Builder\Mobiles** folder that you can attach). Figured it out? That's right, every item you attach to an NPC is ITSELF saved as a .mob! It is given a GUID by ToEEWB and that is written into the NPC's .mob, so u have to keep them all together once you make them :-)

I am adding a spear, leather armour, shield and a few gp in accord with the KotB module, and some boots (a bare-foot merchant? No thanks). I set the "specific slot" thing of the spear to 220 - check the **help/special NPC inventory slots** thing to see where everything goes. It actually goes where u want automatically, but I DON'T want a shopkeeper brandishing a

weapon so I put it in a different slot (does that work? No idea! We'll see... if not, I will scrap this bit). When I want him to use it, I just do san_enter_combat attachee.item_wield_best_all()

Check the NPC flags and both the critter flags sections to see if there is anything u really need (KOS - kill on sight - is important for monsters, but should probably be in the prototype). Otherwise, click save! Later you can make all sorts of complicated stuff, with waypoints and sellable inventories, but there are plenty of tutorials provided by Agetian for that.

Having saved, go (back) to **ToEE World Builder\Mobiles**. This is where your new .mobs are! There will be one for the NPC and one for each item he or she has. I have seven.

Provisioner; Shortspear; Leather Armor; Small wooden shield; Leather boots; Gold; Silver (*I threw in a few SP even though the module didn't say - and a couple levels Ranger*).

Select all these and copy them, then find your map. If it is not one that has been fiddled before by Co8, then u will have to crack the ToEE.dat with Zane's .dat extractor and go to **ToEE/maps** and find the map you want. Then, go to **Atari\Temple of Elemental Evil\modules\Co8-4.0.0\maps** and make a folder with the identical name there (probably you should make a new folder, paste the mobs in, then copy/paste the name of the map folder from the extracted .dat file). That's it! Once your .mobs are in there, they will show up.

CLEAR THE MAP CACHE! Or your new .mob won't show up in game!!!

And finally, if you only created the prototpe with ToEEWB, don't forget to copy that protos.tab and description.mes over to your **data/rules** folder. You're done. Run the game, go to the map in question, and there's your new NPC. 8^D

Common Mistakes and Common Scripts

Today I am gonna list a few common modding mistakes (hey look, that matches the title. What a coinkidink). These are things I tend to do over and over and over, and until I started doing them regularly enough to pick them up quickly, they can be so tiny u don't notice them even

when bug-hunting and they can ruin an entire day. There is also some helpful stuff about finding and overcoming them.

Wrong Brackets: Having **{** instead of **(** or *vice versa* is common and frustrating.

To find it, open your dlg file in Proto-Ed. It will assign columns according to pairings of **{}**. If one of these is open or mistyped or something, it will show up pretty clearly. Elaborate programs like **NotePad +** also come in handy at such moments.

Open **(** brackets is a common enough thing in .py files, and a single open bracket can take down the whole file. If you try adding a heartbeat or combat script or something and the whole .py file goes down, won't even open the dialogue, this is probably your culprit. Look where they are paired: something like

_____if (((game.global_flags[368] == 1) or (game.global_flags[313] == 1)) and (attachee.reaction_get(game.party[0]) >= 0)):

Another common punctuation problem can be forgetting the colon (**:**) at the end of an 'if' script.

Wrong Operands: Is that the right word? Who cares. I mean the use of mathematical symbols, **= >= < !=** etc.

If you are doing a quest or something, it is common enough to do the 'trigger quest / set flag' bit and the 'effect it has' bit simultaneously. So in dlg you have:

{150}{So you'll do my quest?}{So you'll do my quest?}{}{}{}{game.quests[101].state = qs_accepted}

And in another dlg u have:

{152}{I'm on a quest, I'll need you to answer some questions.}{}{8}{game.quests[101].state ==

qs_accepted}{250}{}

Simple enough. And to minimise screw-ups, you make liberal use of copy-paste rather than risk typos.

Fine. But what do you forget?

You forget the change in operands between an executing function and a comparative one: namely, the fact that above in our first example we have ONE equals sign becuase we are setting the qs_accepted bit, and in the other we have TWO because we are asking if it is exclusively equal. (For anyone who is curious - yes you can use >=, <= etc in such circumstances but I am not entirely sure what consitutes 'greater than' what. It is probably obvious, 'unknown' is less than 'mentioned' is less than 'accepted' is less than 'completed' is less than 'botched', but I am not sure and so don't bother using it.)

Anyways, if your quest stuff, or anything involving an executable or comparative function ain't working, that's something to check.

Wrong Args: Here's a right PITA! This can occur where u have already defined one of the objects in a script, then u use something abstract like 'triggerer'. Or, forgetting to put args in at all. Lemme show u an example.

```
def san_heartbeat( attachee, triggerer ):
_____for obj in game.obj_list_vicinity(attachee.location,OLC_PC):
_____________if game.global_flags[3] == 0:
_____________________attachee.turn_towards(obj)
_____________________game.global_flags[3] = 1
_____________________if (attachee.has_met( obj )):
_________________________________obj.begin_dialog( attachee, 100 )
_____________________else:
_________________________________obj.begin_dialog( attachee, 1 )
```

_____return RUN_DEFAULT

Simple enough script, (makes the Tower Watch yell at you when u first go into the Bailiff's tower. Flag 3 is reset by the first heartbeat, so he only does it once each time). But didn't I have a ginormous amount of grief trying to get this to work! Because I had written the common-enough **triggerer.begin_dialog(attachee, 1)** instead of **obj.begin_dialog(attachee, 1)**. Bastard! I even got caught doing this with **attachee.float_line(945,triggerer)** at one point, and I swear I have used that line more than anyone else in ToEE's history (including the developers!)

Also, don't forget to add args if u need them. I have had many issues trying to get made-up scripts to fire because I forgot that little caveat. So, the following scripts (which I or others made) work fine:

```
def amii_dies():
_____game.quests[101].state = qs_botched
_____return RUN_DEFAULT

def room_no_longer_available():
_____game.global_vars[4] = 0
_____game.sleep_status_update()
_____return RUN_DEFAULT

def can_sleep():
_____if (game.leader.map == 5030): # inn
_____if (game.global_vars[4] >= 1):
_____return SLEEP_SAFE
_____return SLEEP_PASS_TIME_ONLY
_____elif ((game.leader.map == 5094) or (game.leader.map == 5134)): # Forest, Graveyard
_____return SLEEP_DANGEROUS
```

_____elif game.leader.map == 5123: # Caravan
_____return SLEEP_SAFE
_____return SLEEP_PASS_TIME_ONLY

The last one is, well, not complicated, but its got a bit in it (and will doubtless be huge by the time we have finished KotB). But none of these need arguments, they can all function independantly.

But throw in something like:

def chatter_box(attachee, triggerer):
_____attachee.float_line(900,triggerer)
_____return RUN_DEFAULT

and u better have that 'attachee, triggerer' bit in there, and in the script that calls it too! (Ok I know this shows what a n00b I am at Python, but then u r too or u wouldn't be reading this ;-)) This was the reason my time-thingy wasn't working, btw: all the ones I checked had no args, and when I started having trouble I reduced the script to the simplest possible thing to get it to work, but it wouldn't because I still needed args for it.

Wrong Conditional: Speaking of n00b mistakes, I may be wasting my breath mentioning this, as I may be the only person who does this! But I tend to use skill checks more than any other sort of checks in dialogue commands, and I sometimes get **pc.skill_level_get()** and **pc.stat_level_get()** mixed up even though there are a lot more stats to check than skills! For instance, if I had a dollar for every time I've errantly written:

pc.skill_level_get(stat_deity) == ##

then Yvy and I wouldn't have to worry about getting a mortgage ;-) You'll find a few screw-ups like this in the Tenant's dlg in *DH*, (to be rectified with the next release that comes out, whatever it happens to be) and a few in the Watchmen's dialogue regarding being a follower of

Kord. Annoying error. Another one like it is setting a flag using **game.global_flags[#] = 1** then using **game.global_vars[#] == 1** for ur conditional, again, just silly but it happens (well it does to me! Did it yesterday yet again!)

To help overcome this, I've attached something at the end. But first...

Wrong Name: By 'name' I am refering to the name and id columns in the protos (22 & 23 in Proto-Ed). Sometimes col 22 is missing. Sometimes it is just a copy of the proto-id. Sometimes they are all different. Its a pain. If you have trouble getting an item transfer or such to work, try putting the proto number in the id column, or using the number that is there if it differs. See **THIS** thread for more.

Bug-hunting:

What is it Hicks?
I'm Hudson sir, he's Hicks.
What is it private?
Will this be a stand-up fight or a bug hunt?

Ahhh, what a movie... anyways, if you have a bug in your .py file, and it is not killing the whole thing (ie it works to a point), one way to track it is with the ever handy command:

game.particles("sp-summon monster II", game.party[0])

Think of it like a 'break' command. It will cause the daemon-summoning pillar of flame to descend on your lead character when it executes. Hence, if you have a broken script and you are not sure if it is not being called due to args issues or if it is not working due to internal issues (or whatever), then put this line at the start. If in game u see the spell effect, u know the script at least is being called (even if it is not working). If u don't see it, then the problem is earlier than that (had this idea myself, but it also occurs in some really old threads, Zhuge or Dulcaion or someone recommends it to Liv for memory).

Ok, enough possible problems! There are plenty of them. But here is something useful (and you all have my permission to copy this into a .txt file and keep it on your desktop, like I do).

Some of these problems, like mixing up executables and conditionals, occur due to copy-pasting, and copy-pasting is done as a way of avoiding spelling errors (anyone who has spent a day going nuts because u didn't notice you mis-spelt 'neutral' will know what I mean).

Anyways, we can't eradicate the need to copy-paste, but we can eradicate the amount of subsequent typing (and possibility of typos) and we can put everything together in one place, so you don't have to go hunting through various scripts to find the right one to copy. Of course, Phalzyr's Dialogue/Python Commands thread does that, but it mixes a lot of stuff you need with even more that, generally, you don't, and the stuff you need often only has one possible parameter out of many that might all be useful (and that might all be potentially mis-spelled!)

So what I have done is accumulate a bunch of common little commands, all culled directly from the game so we know they work (hmmm we ARE talking about ToEE here... lemme rephrase that: commands that I have used in game successfully so we know they work!) So here they are! As I said, I keep a .txt doc on the desktop with these, opens instantly and everything easy to find. Feel free to copy it. Hope you all like it, and if anyone thinks of anything that is missing (or, heaven forefend, finds an error in here!) lemme know. I use these for KotB and it seems to run ok so far.

pc.skill_level_get(npc, skill_bluff) >= 7
pc.skill_level_get(npc, skill_diplomacy) >= 4
pc.skill_level_get(npc, skill_sense_motive) >= 8
pc.skill_level_get(npc, skill_intimidate) >= 4
pc.skill_level_get(npc, skill_gather_information) >= 6
pc.stat_level_get(stat_gender) == gender_male *or* gender_female

game.quests[].state = qs_accepted
game.quests[].state = qs_completed
game.quests[].state = qs_botched
game.quests[].state = qs_mentioned
game.quests[].state = qs_unknown

game.party_alignment == NEUTRAL_EVIL or game.party_alignment == CHAOTIC_EVIL or game.party_alignment == CHAOTIC_NEUTRAL or game.party_alignment == LAWFUL_EVIL *(chaotics)*
game.party_alignment == LAWFUL_GOOD or game.party_alignment == NEUTRAL_GOOD or game.party_alignment == CHAOTIC_GOOD or game.party_alignment == TRUE_NEUTRAL or game.party_alignment == LAWFUL_NEUTRAL *(non-chaotics)*
game.party_alignment == LAWFUL_GOOD or game.party_alignment == NEUTRAL_GOOD or game.party_alignment == CHAOTIC_GOOD or game.party_alignment == LAWFUL_NEUTRAL *(goodies)*

pc.stat_level_get(stat_alignment) == LAWFUL_GOOD
pc.stat_level_get(stat_alignment) == NEUTRAL_GOOD
pc.stat_level_get(stat_alignment) == CHAOTIC_GOOD
pc.stat_level_get(stat_alignment) == TRUE_NEUTRAL
pc.stat_level_get(stat_alignment) == LAWFUL_NEUTRAL
pc.stat_level_get(stat_alignment) == CHAOTIC_NEUTRAL
pc.stat_level_get(stat_alignment) == LAWFUL_EVIL
pc.stat_level_get(stat_alignment) == NEUTRAL_EVIL
pc.stat_level_get(stat_alignment) == CHAOTIC_EVIL

create_item_in_inventory(4400,pc)

pc.stat_level_get(stat_deity) == 1 // Boccob
pc.stat_level_get(stat_deity) == 2 // Corellon Larethian
pc.stat_level_get(stat_deity) == 3 // Ehlonna

pc.stat_level_get(stat_deity) == 4 // Erythnul
pc.stat_level_get(stat_deity) == 5 // Fharlanghn
pc.stat_level_get(stat_deity) == 6 // Garl Glittergold
pc.stat_level_get(stat_deity) == 7 // Gruumsh
pc.stat_level_get(stat_deity) == 8 // Heironeous
pc.stat_level_get(stat_deity) == 9 // Hextor
pc.stat_level_get(stat_deity) == 10 // Kord
pc.stat_level_get(stat_deity) == 11 // Moradin
pc.stat_level_get(stat_deity) == 12 // Nerull
pc.stat_level_get(stat_deity) == 13 // Obad-Hai
pc.stat_level_get(stat_deity) == 14 // Olidammara
pc.stat_level_get(stat_deity) == 15 // Pelor
pc.stat_level_get(stat_deity) == 16 // St. Cuthbert
pc.stat_level_get(stat_deity) == 17 // Vecna
pc.stat_level_get(stat_deity) == 18 // Wee Jas
pc.stat_level_get(stat_deity) == 19 // Yondalla
pc.stat_level_get(stat_deity) == 20 // Old Faith
pc.stat_level_get(stat_deity) == 21 // Zuggtmoy
pc.stat_level_get(stat_deity) == 22 // Iuz
pc.stat_level_get(stat_deity) == 23 // Lolth
pc.stat_level_get(stat_deity) == 24 // Procan
pc.stat_level_get(stat_deity) == 25 // Norebo
pc.stat_level_get(stat_deity) == 26 // Pyremius
pc.stat_level_get(stat_deity) == 27 // Ralishaz

pc.stat_level_get(stat_level_cleric) >= 1
pc.stat_level_get(stat_level_druid) >= 1
pc.stat_level_get(stat_level_paladin) >= 1
pc.stat_level_get(stat_level_ranger) >= 1
pc.stat_level_get(stat_level_barbarian) >= 1

```
pc.stat_level_get(stat_level_monk) >= 1
pc.stat_level_get(stat_level_rogue) >= 1
pc.stat_level_get(stat_level_bard) >= 1
pc.stat_level_get(stat_level_wizard) >= 1
pc.stat_level_get(stat_level_sorcerer) >= 1

pc.stat_level_get(stat_race) == race_orc
pc.stat_level_get(stat_race) == race_halfling
pc.stat_level_get(stat_race) == race_human
pc.stat_level_get(stat_race) == race_elf
pc.stat_level_get(stat_race) == race_dwarf
pc.stat_level_get(stat_race) == race_gnome

pc.stat_level_get(stat_strength) >= 10
pc.stat_level_get(stat_dexterity) >= 10
pc.stat_level_get(stat_Intelligence) >= 10
pc.stat_level_get(stat_Wisdom) >= 10
pc.stat_level_get(stat_Charisma) >= 10
pc.stat_level_get(stat_Constitution) >= 10

pc.money_get() >= 100
pc.money_adj(-100)

is_daytime() != 1

game.global_vars[] = game.global_vars[] + 1
game.global_flags[] == 1

game.party_size() <= 4 and pc.follower_atmax() != 1
group_average_level( game.leader ) >= 2
```

Sectoring Internal Doors

Recently you may have seen me gloating about putting doors on the KotB maps. Its an easy thing to do, but I thought I would publish some of the stuff I have been doing to help anyone trying to do something similar for any sort of mod.

Firstly, to add a door, you have to know where it goes. So, as per normal, make sure ed.py is in your **scr** folder, then go to the place on your map where you want there to be a door. Open the console and type

from ed import * (then hit *enter*, this loads ed)

loc()

and write down the coordinates.

Next, power up ToEEWB. We are gonna create a new .mob, so hit 'new'.

Now, we need a door. There are 111 of the little buggers in ProtoEd, they're at the very beginning. Which one do we use? (Don't bother with 150, for memory I made that and its a stuff-up). We'll get to choosing the right door for u in a second, lets assume u r using 1.

Now, put in the location as u have found it.

You WILL need to fiddle the offsets to get it to fit an existing doorway in a new map: to nudge it a bit down-n-left, add 10 or so in x offset. Up-n-right, have -10 in x offset, down-n-right add 10 or 15 in y offset, up-n-left try -10 (see the *Isometric Schematic View* in the **sector** bit of ToEEWB to see how the x and y values work). As for z, 10 or 15 will bring the door straight vertically down (hows that for tautology!) while -10 or so while nudge it straight up.

U also need to fiddle the rotation. Lets plagiarise Agetian one more time:

5.5 - northwest
5.0 - west
4.0 - southwest
3.0 - south
2.5 - southeast
2.0 - east
1.5 - east (sometimes it looks better than 2.0)
1.0 - northeast
0.5 - northeast (sometimes it looks better than 1.0)

Note how it goes anti-clockwise. If your door is in a wall running Nw-Sw (so a door facing out from it is perpendicular, facing SE) and is not perfectly aligned when u put in 2.5, nudge this up in the necessary direction. Ie, if it needs to be a bit more east-facing, nudge it up to 2.7. If it has to be a bit southward, try 2.3. Its trial and error but not as bad as it sounds, for instance I found the necessary value for one of Screegs maps and suspect he uses a similar camera angle so now i have it, i can type it straight in every time :-D If you are making new maps, u might wanna keep that in mind.

When u have done with your .mob, you EMBED it (not 'save', if you click save it will warn u u r doing something stupid). Of course u can save to fiddle with later (easier than embedding and extracting). So make sure u have the relvant sector (or all sectors in that map) in the **sector** folder of ToEEWB. You don't have to open then save the sector or anything like that, it will embed your door right into the sector in the folder even if u have not opened it with ToEEWB first)and you shouldn't open it, cause if u hit save later u will write your opened, doorless sector OVER the embedded sector and your door will disappear).

How to tinker the door later after embedding?

- open the sector
- go to *static objects*
- select the door and hit *extract*

- read what it says about having just made a .mob
- go back to **objects** and open the .mob created
- fiddle the .mob
- embed it
- go back to **sectors** and *load the sector again*
- now it has the new door embeded, delete the old one, and hit save!
- copy your sector back into the game, clear the map cache and test it.

Ok, that's straightforward, nothing new for seasoned modders, now for Ted's new contribution.

I spawned doors on a couple of maps and numbered them with Photoshop. When u want to get a new door, or make a map that takes doors into account, look here. To my knowledge doors CAN'T be resized with the '3d model scale' thing. I tried it once - the game crashed. Perhaps new doors can be made resized as protos in col 6 of ProtoEd, but for the moment ur stick with the doors in game as the size they r.

Here are the pix of the first 40 doors. The sliding doors have arrows indicating direction of slide. Some of them have been opened and look perpendicular to how they appear when first spawned. I left a few PCs for scale, but to get them in your game, put your PC where u want the door and in the console, enter 'create ##' and see how it looks. Rotation will be off ;-)

Hmmm... seems u have to click on the little buggers to get em to a decent size. Well, click dagnabit!

Adding movies

As you all should know, Brother Gaear has made some new .bik movies for the game and will be called upon in future to do this again for KotB. He asked me to investigate how to get them to function, so here we go.

In **ToEE**/**Movies** there is a file called **movies.mes**. To mod it, copy the FOLDER over to your module folder - **modules**/**Co8 4.0.0** in this case. Pop the file open.

Add a new line:

{1030}{NewMovie.bik;;}

where 'newmovie.bik' is, of course, the name of your new movie in the **data**/**movies** folder (with ToEEFE, keeping in mind the Toffee fiddles with these things to make them wok. Non-Toffee installs may not have this folder, I don't remember). Substitute the name of your new movie if you prefer.

Now the game can identify your new movie by this number (1030). So, to fire the movie in game, you then make a call to movie 1030. So, in the **maplist.mes** file in the MODULE'S rules folder (**modules/Co8 4.0.0/rules** in this case) you would add the call

Movie: 1030

to the map you want to fire the movie. In this case, it is something for the Moathouse, so we would mod the **maplist.mes** file to say:

{5002}{Map-7-Ruins-of-the-moathouse, 481, 552, Movie: 1030, Flag: DAYNIGHT_XFER, Flag: OUTDOOR}

Or you can get it to fire from in a dialogue line with this executable:

game.fade_and_teleport(0, 0, 1030, 5002, 481, 592)

I tested this by copying an existing .bik, renaming the copy appropriately and firing it as described. Worked very nicely once I had got the hang of where to put everything.

Complete(ish) Sectoring Tutorial

Today, we are going to sector a new map from beginning to end. This will allow people to use ToEEWB easily, where atm it may seem a scary and daunting task. Its not, if you actually take the time to RTFM and do the tutorials - ALLANON for instance threw himself into the task and come out with a very nice (and very useful) Forest map that will grace KotB and earn him unending glory. (I also want him to sector a swamp map for me when I have hacked the jpg together - I haven't forgotten dude!)

You will need:

- a new map .jpg

- the latest copy of ToEEWB
- a copy of **ed.py** in your **sec** folder
- that's about it

I am doing the ground floor of one of the towers by the Keep Gate. Afficianados will remember this has 12 men-at-arms in its garrison, 3 floors and the roof. Will all 3 floors make it in? Probably not, there is just nothing going on in there... what we ARE going to do is quarter the Corporal of the Watch here (the guy who meets you at the front Gate). The module details where the various officers live, but not this guy - it seems reasonable he should live here. Screeg has done a very useful ground floor map with his quarters, and that is what we shall be concentrating on today. The map looks like this.

First of all, we need somewhere to put the map, so we pop open **maplist.mes** in the **modules/KotB/rules** folder. We want an original Hommlet map (ie area 1 in the game) to go over, so lets use the map 5033, the Moneychanger. We'll rename it GateTower-first. The maplist.mes now reads:

{5033}{Map50-Keep-GateTower-first, 482, 485, Flag: DAYNIGHT_XFER, Flag: UNFOGGED}

The numbers are way wrong but its pretty much irrelevant: those are only there for moving by WorldMap, and we ain't gonna be doing that to get in here! We will come in by the door (or down the stairs) and will use the Jumppoint.tab to control that. But lets not get ahead of ourselves.

Now we need to name our map, so we pop **Map_names.mes** and at line 5033 we modify it thusly:

{5033}{Gate Tower - First Floor}

We can type anything here, its not important, it just shows up in a few places like your little map list in the 'WorldMap' bit of the interface.

Next we do **Maplimits.mes**. This is currently in **modules/KotB/rules**, but Agetian tells me the game is not drawing from there and it is the wrong place! Well, its a concern, but thus far it works (drawing from the current one I guess) and you'll forgive me if there are more important things to do: I WILL fix it before the end - Agetian has gone to great lengths to understand the parameters used here - and I will certainly fix it at some point. As I said though, atm it works, and that's all that matters.

What I do here is modify the limits to match the Main Floor map of the Hommlet Church. This is my 'base' map for practically everything I am doing, (a few larger maps being the exceptions) so most of the KotB maplimit parameters are identical. 5033 is now changed to match there, so it now says:

{5033}{1020,-12544,-780,-14084}

What do those limits mean? Well, I do know (because Agetian explained it to me at great length, complete with diagrams), but this tutorial is going to be long enough! Suffice it to say they set the limits where the map stops scrolling.

What next? Well, we need our new map to be accessable by the game. This involves two new folders. One has the graphics. One has the other files.

First the graphics, our new map is defined in maplist.mes as Map50-Keep-GateTower-first, so

that's what our Folders will be called. The first one goes in the **modules/KotB/art/ground** folder, the second in **modules/KotB/maps** folder. First things first.

I make a new folder in modules/KotB/art/ground called Map50-Keep-GateTower-first. This is where all the actual backround map graphics are kept. There is also a file in that folder called ground.mes - this is the control file that tells the game what maps to look out. Its done in rather a funky way, but we'll get into that in a minute. For the moment, lets just copy over the money-changer's map entry, number 52. Then we will go down to number 1052 and make the same entry. They now read:

{0052}{Map50-Keep-GateTower-first}
{1052}{Map50-Keep-GateTower-first}

These are the day and night entries. Since this is a military tower with no windows, we don't bother with day and night entries, the place is identical inside needing artificial light at all times. Some maps, such as the pub or the outdoor ones, have seperate day and night graphics to differentiate them, and Screeg has done a fantastic job for them. If we WERE to do day and night, we would call them:

{0052}{Map50-Keep-GateTower-first-day}
{1052}{Map50-Keep-GateTower-first-night}

and we would have TWO appropriately-named folders in the modules/KotB/art/ground folder, with different jpgs.

Did I mention I learnt all this bit from Jota's map tutorial in the Co8 forum? Good for him! On the shoulders of giants, I tells ya...

Next, we need our **modules/KotB/maps** folder, so we make one in there called Map50-Keep-GateTower-first. This folder will contain our sec files, mobs, those recently tamed svbs, lighting files etc, and a sexy little device called **map.prp**.

What does this file do? I have no idea! That's another Agetian question. What I have done with it is simply modify one number to get it to point at a different set of jpg images. That's what we are going to do with it now.

Grab an map.prp from a nearby folder and copy it into our new folder. Open it with a hex editor (this is where we modders earn our money). You'll see a bunch of numbers - the amount can change, I think its because the hex-editor itself will sometimes pad out the file with zeros. Some of the numbers (like c0 03) look familiar from countless hex-hacks of mob files, but in fact the only number we need is the first one. Mine is 30: I'm changing it to 34, that's 0052 in hex. Soooooooooo, lets recap:

Maplist.mes tells the game there is a new map referred to by the title Map50-Keep-GateTower-first and the number 5033. **Map_names.mes** gives the map a title for when that is necessary.

When you access this map (by teleporting to map 5033, say) the map looks for a folder in the maps folder similarly designated Map50-Keep-GateTower-first.

Inside that folder is **map.prp**: the game reads this, and hits the number 52 (hex 34). It then goes to **art/ground/ground.mes** entry 52 (or 1052 at night) to access the appropriate graphics files for the background image of the map. These, entry 52 tells it, are found in the folder Map50-Keep-GateTower-first of the ground folder. So it goes off and gets them, and thus we have our background map onscreen.

That will get our background jpg up on the screen. Lets prepare it.

First, I take the pic and I resize it for the game. In the past, this has involved getting a map from ToEE and comparing it in Photoshop to the new one, then resizing the new one and running a character around on it and seeing if it fits. Try using something like a chair or something definite (again, I use the main Floor map of St Cuthbert's church, some of the

furniture in the side room can nicely be compared to the furniture on the new map). There is some trial and error of course, and on occasion Screeg has had to redo stuff (which he has been good about).

For this map, it is on the same scale as the Bailiff's tower so I just have to match that - easy. However, so you know how to do it... the game maps are of course kept in the folders in **module/ToEE/art/ground**. If you have a look in there though, u find they are all hacked up into little jpgs of identical size. How do we combine them into one map to work off?

That's where ToEEWB comes in. Run it, and click on '2d Maps'. We want to recombine THE BIG MAP, so do this: click it (go on!) Then navigate to wherever you have unpacked ToEE.dat and find art/ground and whatever map you want to recombine. Note if you wanna recombine the main Hommlet map, u better have a lot of processing power and RAM and some time on your hands ;-) Early P4s with 512Mb (like mine), don't bother!

Now, I don't need to do that, but I DO have to have already done it (hows that for a pluperfect) for reasons we will soon see. I am doing the opposite, instead of combining a map I am splitting it for the game to read from art/ground. So, I resize my map in Photoshop to match the existing tower one and save it into the ToEEWB/2D Maps Folder.

At this point I have a surreal experience - its already there. Apparently I did all this when Screeg first sent me the map becuase I was excited to get it. Hmmmm...

Well I haven't split it, that's for darn sure! Still got ToEEWB open? Yes? Good. Click 'split'. It should bring up the ToEEWB/2D Maps folder. If your jpg isn't in there, stick it in. But don't click split yet.

The reason why I suggested recombining a map first is it will create a .txt file named from that map, explaining how wide and long the map is. When u split your map, u need such a text file to tell ToEEWB how many files to split it into and how to 'name them'. Make sense? No? Trust me... if u open **ToEE/art/ground** and pop a map, you will see they start with names like

001F001D.jpg. Without the .txt file, ToEEWB will start numbering them as 00000001,jpg, and the game will not know what to do with them. So whatever map u r 'emulating' (as I am with Map-5-Main-Floor in the church), recombine it first for the .txt file!

Now... I copy this .txt file and rename it to match my .jpg - gate_tower_ground.txt in this case (this is for the internal workings of ToEEWB, doesn't have to match the map names of the in-game folders or anything.) NOW I can click split.

Some seconds later (for a map this size) and we now have a new folder in **ToEEWB/2D Maps** called gate_tower_ground. Copy its contents over to the **modules/KotB/art/ground** folder, into our folder Map50-Keep-GateTower-first (shoulda called that GateTower-ground - o well, live and learn).

Now it should work. Power up the game. Got ed.py installed in your scr folder? Yes? If not, get it from ToEEWB. Run the game, new party (or whatever) and teleport to our new map, 5033 (go to 500,500, that's always safe). To do this for the newcomers, open the console and enter:

from ed import *

Then hit enter, that loads in the ed.py file. Now type:

tp (5033,500,500)

and hit enter, that teleports you nice n simply :-)

Done it? Yes? No? Well I just did, and it worked - woohoooo!!! After a few seconds running around, I saw this:

All looks a bit cramped to me, but the perspective is right, and after all, its a barracks - not meant to be wide open spaces or anything.

Ok, all good so far, we have a new map and we can run around on it. BUT... we can run around on it a bit too much, straight through walls etc. Lets start sectoring!

This is where ToEEWB comes into its own. With ed.py in there, we can sector in walls etc, and other unpassable areas just by running a character over them in the game. Couldn't have an easier method if u tried (and there *is* an easy method in ToEEWB itself done by painting, we'll see that in a minute).

To get ed.py to function, once again we type into the console:

from ed import *

and hit enter. Note that you only have to do this once per 'sitting', though if you reload a game u have to do it again (though it will be in the console memory and u only have to scroll up, find it and hit enter).

What we are going to do first is outline the edges of the map: ie, the walls where you want the player to stop. First we are going to activate the script that records the position of the walls as we are going to define them, then we are going to actually define them by moving the leader

(the PC in the leftmost position in your party roster at the bottom of the screen) around the places we later want to be 'walls', ie unmovable. So, the first thing to do is move the leader to the edge of a wall, thusly.

Then, we activate the script by typing in

blk_on()

This means we are turning on 'blocking' - marking areas that block all movement (you can't walk through them, fly a familiar over them, shoot through them etc). Walls, for instance. There are also areas you can fly over but not walk over, and areas that provide cover (-4 to hit). We can also define water this way.

If it worked, you should see the following in your console:

If it didn't work, you will see something about an unnamed string or something - its pretty clearly an error, and means you either didn't activate ed.py as above, or there was a typo somewhere.

Once you have it up and running, you then run the character where you want the wall to be, eg:

Be careful, u can't undo mistakes in-game, u have to manually open the sector in ToEEWB and fiddle it. We will do this next, but its a pita, so for the moment just don't make mistakes. Using this method, u should be able to do a nice square around the outside of the room. To turn off blocking, you type:

blk_off()

and should get the following reponse:

Again, u can just scroll up to reuse these commands, you don't have to type them over and over.

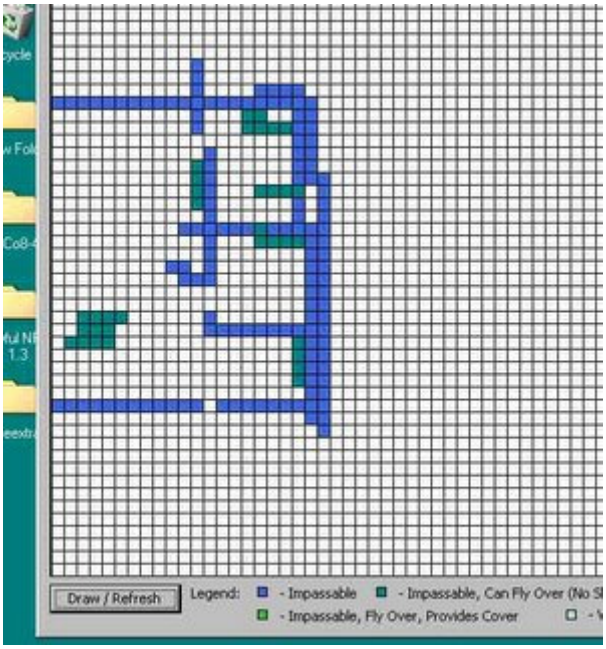
If you want to add a single tile, instade of painting an area, just type in

blk()

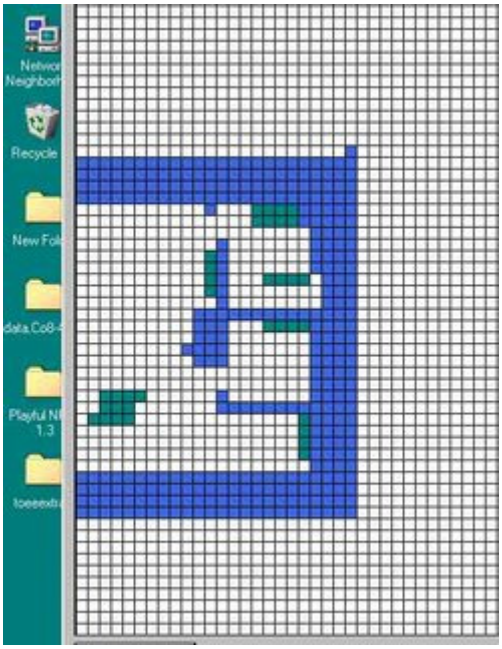
The other commands are fly() and cov() for fly-over and cover respectively, ditto fly_on(), fly_off(), cov_on() and cov_off(). Run around the map making the desks and beds etc fly-over (or providing cover, if u r nasty enough ;-)). Add puddles (if your map has the graphix for it) with wtr().

Pretty soon, your console will look something like this:

Now we can have a look at what we have done in ToEEWB. The files have automatically been writing themselves in the /**sectors** folder. To see what you have done, click on the Sectors tab in ToEEWB (second from left), then open - there should be two new sectors in there. Open the first one, then click *tools/visual sector analyzer*. When it comes up, hit refresh down in the bottom corner: there's a visual image of what you have just made! So easy to use... the blue bits are blocked, light green are providing cover, dark green are fly over etc. Mine looks like this:



I then clean it up a bit, using the painting command. To paint, on the right click on 'paint mode' then what you want to paint ('fully impassable' ie blocked, 'fly over' etc). Note that 'fully passable' can be used now to correct errors. Doorways need to be two tiles wide to get through, btw. Once cleaned up, it looks like this:



After you have cleaned it up or made whatever changes you want to make in the painting tool, don't forget to SAVE THE SECTOR! You can switch off the painter bit without losing anything

btw, but save the sector on the ToEEWB interface proper.

Now, to test it, I copy it to the **modules\KotB\maps\Map50-Keep-GateTower-first** folder: the folder that atm should only have map.prp in it. Just copy it, don't move it - that way while you are testing, if there is anything you want to change, the files are right there in the sectors folder to be easily modified.

Run the game and don't forget to clean the map cache! Then teleport to the map and run around it.

You can't - there are walls to stop u! So teleport this time to 508,487: that should put you comfortably in the room.

Ok... now things get funky. We are going to use the .svbs to make our characters grey out when they are behind walls (so we have a '3d'(ish) interactive environment not just a painted background our characters walk over).

If you haven't read the tutorials in the ToEEWB about svbs, do so - they are very short, and provide one brilliant idea. That is, to see where you want to do your svbs, you should paint them on the map using ed.py by marking them as *water*.

See, these different things are stored in different files. Sectoring stuff like we have done is stored in the .sec files, svbs in the .svb file, and water in the .hsd file. So, u run around marking everything as water where u want the model to grey out, then in ToEEWB you paint it up as svbs, then presto, just delete the .hsd file (or never copy it over to the map folder) and the water is gone.

We have to do this because svbs can't be painted using ed.py, they can only be added with ToEEWB; so we need a frame of reference so we know where we are putting them. So paint them as water, and you're set.

So what do we do? Well, on the map, put your leader somewhere that s/he should be covered by a wall. Then enter:

wtr_on()

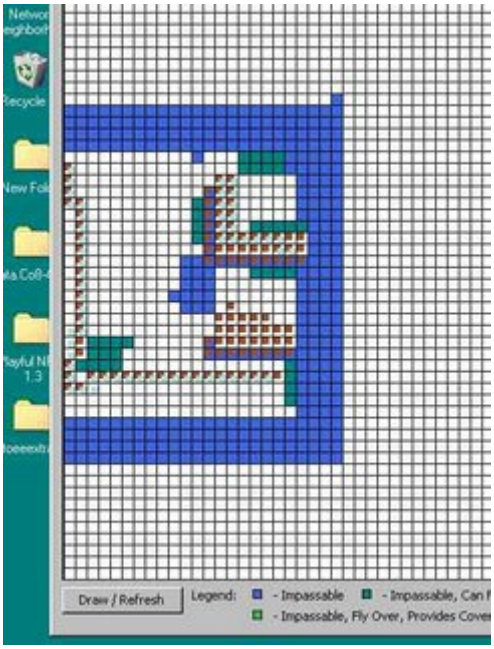
and run around behind the wall, everywhere. When you are finished, do wtr_off(). To do individual tiles, use wtr(). Get out and open the file in ToEEWB. Now, (make sure ur ToEEWB is up to date, only the most recent versions support this!!!) where it says

Sector Visibility Blocking (will be painted over existing tile data)

go into paint mode and click on + extend +. Paint OVER your water tiles (the ones with little light blue squares) and notice you are now ADDING little brown squares to indicate that the .svb dfile is acknowledging this tile (the file will now be created of course). You are not replacing what is already there.

When u have marked up all the 'water' tiles, and any others you may hjave missed or wish to add, you should ALSO do ALL the relevant walls that the player is behind, that is, the wall that is blocking players visibility. Svbs don't just grey out the models so they look like the player is behind a wall, they also (when the map is properly lit) prevent the far side of the wall being lit / seen, to prevent u seeing through walls, very handy.

Here is a pic of what I have done for my map. Note the water tiles I have painted over (and in some additional cases where I have allowed for the leader not being able to 'run' as accurately as I would have liked) and how the southern walls in each room have also been svbed as well as the areas the player runs around in.



I have also done the 'ceilings' (the long black bits indicating the tops of walls - the long lines in each direction).

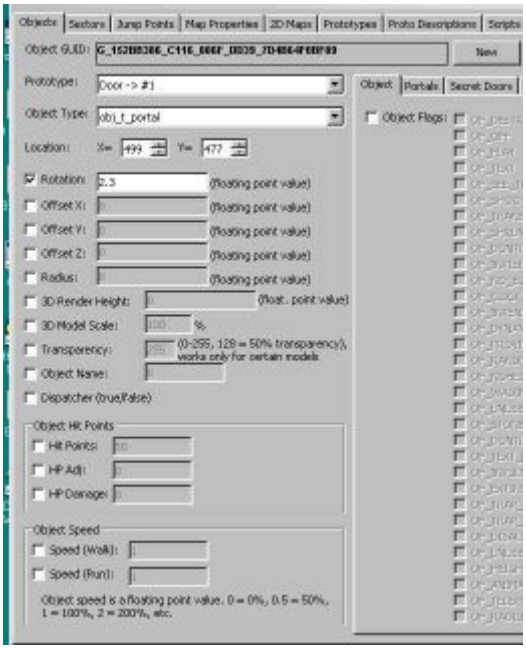
And here is an image of it all done, with greyed out models in place:

Bit hard to tell from the pic, but when you go from 'normal' to 'behind wall', it is quite noticeable. Ok, lets add an internal door. {Edit: Forgot I did 'internal door sectoring' when I uploaded the pix of the doors! Thought it all felt familiar... o well, you've had the theory, here it is in practise 9^D } First we have to decide where to put it, so we power up the game, go to the map, run our leader to where we want the door to appear, and (with ed.py activated of course) type:

loc()

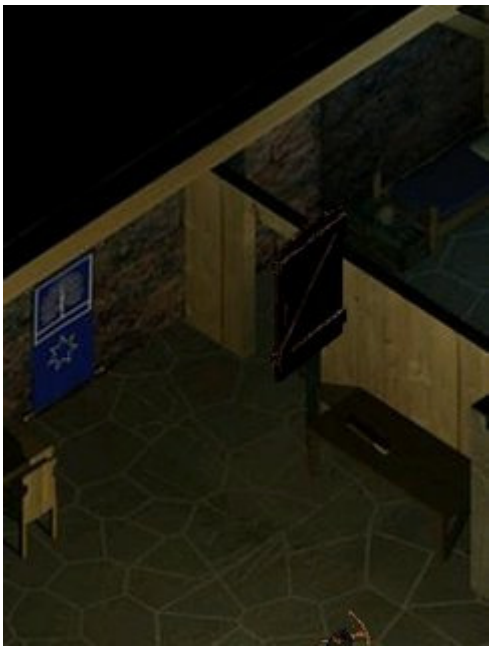
in the console and hit enter. (You probably knew to do that by now, its a common command). Apparently my door is going at 499, 477.

Now go to ToEEWB and prepare to create a new object. We need a door, so check out **THIS** tutorial and find one to suit you. I am gonna try one of the first couple. The rotation will be the same as any other mob, lets call it 2.3 for the moment. Lets leave it simple. We'll fiddle it momentarily. Here's all you need:



Your GUID will be different of course ;-) When u r finished, click 'embed'. This writes the door INTO the .sec file (it doesn't save it as a .mob). You CAN save it as a mob if u wanna work on it later, but its not gonna run in the game that way. O and u don't have to click 'save' if u have the sector open or anything: in fact, if you DID have the sector open, click embed, then click save, you will save your already open sector OVER the one with the newly embedded door and it would disappear (done that more than once while fiddling exact door locations. We'll do that in a minute).

Ok, I am gonna copy mine into the game's maps folder and see what it looks like in the game.



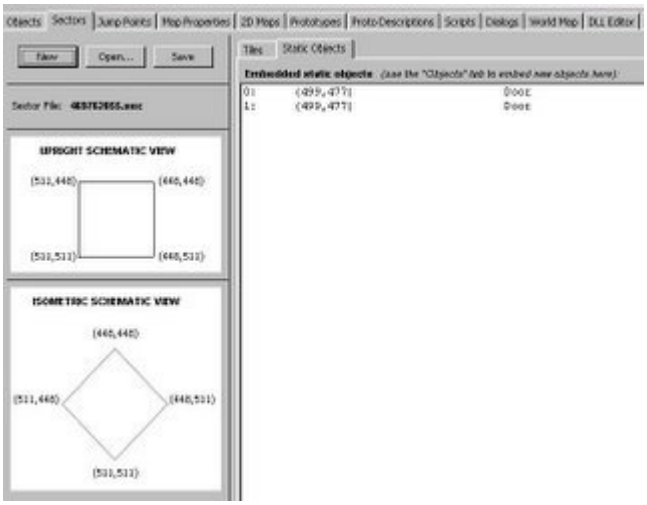
Oops... got the rotation WAY wrong... must've been thinking of something else entirely. Lemme see, where have I used a perpendicular door to this? The Guild building? Can't remember... pity, because Screeg's maps tend to be the same camera-angle, so once u have fiddled around getting the rotation perfect once, u can transport that number to other doors (I have that luxury, yay Screeg!).

O well, trial and error time. Lets get the rotation correct before we nudge it into the right location.

I am trying 3.9 this time (shifting the door from SE facing to SW facing). Ok, embed the new

door.

Trouble is, now we have two! So, we open the sector file, and click on 'static objects'. There are our 2 doors.

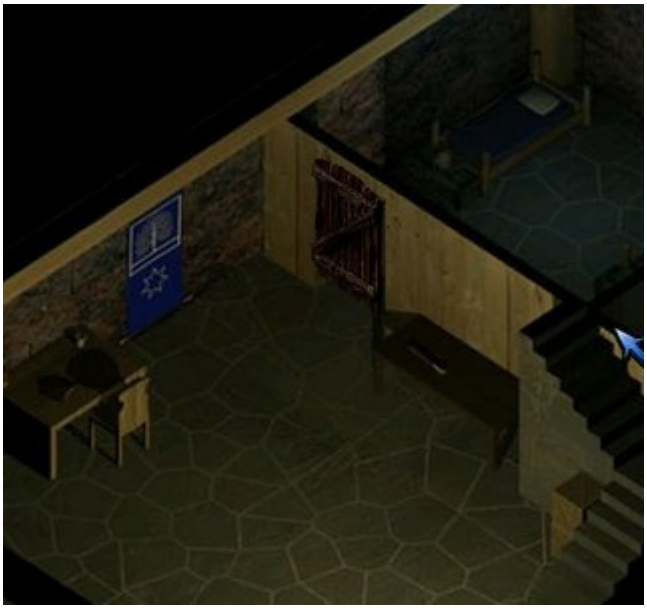


So delete the top one. Now we have one door, our latest one. SAVE the modified sec file, and copy it over to the game.

Run the game (clear the map cache!) and lets take a look.

Ooo nice! Before we see if it needs any minor rotational adjusting, lets get it in the doorframe

properly. Have a look at your 'Isometric Schematic View' in the sector section (Agetian thinks of everything ;-)) We want to move our door slightly forward on the x axis, and slightly back on the y. So, we now use the offsets, and I'll do 'em both by 10 for starters. Btw, if u keep the door as the open object in ToEEWB (as I am here) then u can keep fiddling it and re-rembedding it. If not - if u go back and change an embedded object later - u have to extract it as a temporary mob to play with. ToEEWB talks you through it though, its easy enough.



Ooooooooo, so close! Its just a weeee bit too far up the y axis, and sitting off the ground (at least from our perspective, not really of course!) Lets tweak the y by 2 and move it down the z axis by 10.

Don't forget to open the sec file after u have embedded the changes and delete the old door, then save again! Bet u forget to do that at least once... if there is more than one door on the map, that's why! If you go to copy a tinkered .sec file over to the game and it has grown in size even though u have only fiddled the door, not added more stuff, that's a dead giveaway you forgot to delete the old one.

Anyways, that last tweak'll do nicely! (Though I had a screenie of it but apparently not.) The door doesn't actually fit the frame perfectly - slightly too short, tiny gap at top - but seriously, there's nothing we can do about that. The one time I tried to resize an embedded obj like you can resize a mob (to turn lizards into dragons, or quasits into Daemons etc) the game CTDed, alas. An ill-fitting door is a small price to pay though: at least its straight. I AM gonna move the door up ever so slightly to spread the gap AROUND it not just have it at the top like that, but otherwise I think its fine.

Lets leave it there. Later I will probably make the door lockable, give the key to the Corporal, etc - I will also add door and stair icons to allow you to leave the room ;-) and get the Corporal to head off to the pub by night and be here by day (and get rid of him from the front gate!) But that's for later.

TUTORIAL: Secret doors.

NOTE: I assume that you've read the module building tutorial, so the phrase "copy the file 469762055.sec from the Tutorial Map 1 in extracted files to the Sectors folder of your World Builder installation" makes perfect sense to you.

1. Let's add a secret door to Tutorial Map 1 to (505,457). Copy the corresponding sector file (469762055.sec) to the Sectors folder of your ToEE World Builder installation, and launch ToEEWB.
2. In the object editor, create a new object.
3. From the prototypes box, select "Secret Door Icon -> #2035". Define the coordinates (X=505, Y=457).
4. Click the "Object Name" property and assign value 1602 to it. If you ask me, I don't know what this is for, but its set for all original secret doors, so I assume it's somehow useful.
5. In the "Secret doors" tab, click on the Secret Door DC property and type 10 (well, you may use any difficulty class value you want).
6. Also, click on the "Secret Door Flags" property and then enable the flag OSDF_UNK1 (you may try doing it without actually enabling OSDF_UNK1, but it may not work). Do NOT touch other flags manually; otherwise you will corrupt the secret door DC J
7. On the "Scenery" tab, click on the Use as a teleport with destination flag and assign a jump point 262 to it (so that our secret door will teleport us to tutorial map 2).
8. Embed our secret door into a sector.
9. Copy the file back to the game and try it out. Enjoy!

NOTE: If you reload the secret door into the ToEEWB you'll notice that extra flags will be set in Secret Door Flags that you didn't set up manually. This is NORMAL and INTENTIONAL, do NOT unset or set those flags manually or you'll mess things up badly!

Flags Tutorial (Part 1) updated

Well some at least... I will describe some of the basic ways of using flags, ones that come from inside the game and thus we know work. In some cases they are things I have discovered

myself - be sure I will crow long and loud about those! In most cases they are things found by others such as Liv, Agetian and Dulcaion. Also, at some point we will look at individual variables (as opposed to the global kind) based on some stuff done by Agetian and Cerulean the Blue. Most importantly, this tutorial will be done in parts and will be upgraded at various points as I think of new stuff.

What are flags? Well, I am not talking the global.flags and global.vars, I am speaking about the various object flags, critter flags, npc flags etc that can be put into the prototype or mobbed into specific critters using ToEEWB. If you have used ToEEWB and opened a mob, or made a new one, you will doubtless have noticed long lists of flags. Lets look at them now:

Object Flags (OF_*)

OF_DESTROYED
OF_OFF
OF_FLAT
OF_TEXT
OF_SEE_THROUGH
OF_SHOOT_THROUGH
OF_TRANSLUCENT
OF_SHRUNK
OF_DONTDRAW
OF_INVISIBLE
OF_NO_BLOCK
OF_CLICK_THROUGH
OF_INVENTORY
OF_DYNAMIC
OF_PROVIDES_COVER
OF_RANDOM_SIZE
OF_NOHEIGHT
OF_WADING

OF_LOOTED
OF_STONED
OF_DONTLIGHT
OF_TEXT_FLOATER
OF_INVULNERABLE
OF_EXTINCT
OF_DISALLOW_WADING
OF_HEIGHT_SET
OF_ANIMATED_DEAD
OF_TELEPORTED
OF_RADIUS_SET

Item Flags (OIF_*)

OIF_IDENTIFIED
OIF_WONT_SELL
OIF_IS_MAGICAL
OIF_NO_PICKPOCKET
OIF_NO_DISPLAY
OIF_NO_DROP
OIF_NEEDS_SPELL
OIF_CAN_USE_BOX
OIF_NEEDS_TARGET
OIF_LIGHT_SMALL
OIF_LIGHT_MEDIUM
OIF_LIGHT_LARGE
OIF_LIGHT_XLARGE
OIF_PERSISTENT
OIF_MT_TRIGGERED
OIF_STOLEN
OIF_USE_IS_THROW

OIF_NO_DECAY
OIF_UBER
OIF_NO_NPC_PICKUP
OIF_NO_RANGED_USE
OIF_VALID_AI_ACTION
OIF_DRAW_WHEN_PARENTED
OIF_EXPIRES_AFTER_USE
OIF_NO_LOOT
OIF_USES_WAND_ANIM
OIF_NO_TRANSFER
OIF_FAMILIAR

Critter Flags (OCF_*)

OCF_IS_CONCEALED
OCF_MOVING_SILENTLY
OCF_EXPERIENCE_AWARDED
OCF_FLEEING
OCF_STUNNED
OCF_PARALYZED
OCF_BLINDED
OCF_HAS_ARCANE_ABILITY
OCF_SLEEPING
OCF_MUTE
OCF_SURRENDERED
OCF_MONSTER
OCF_SPELL_FLEE
OCF_ENCOUNTER
OCF_COMBAT_MODE_ACTIVE
OCF_LIGHT_SMALL
OCF_LIGHT_MEDIUM

OCF_LIGHT_LARGE
OCF_LIGHT_XLARGE
OCF_UNREVIVIFIABLE
OCF_UNRESSURRECTABLE
OCF_NO_FLEE
OCF_NON_LETHAL_COMBAT
OCF_MECHANICAL

Critter Flags 2 (OCF2_*)

OCF2_ITEM_STOLEN
OCF2_AUTO_ANIMATES
OCF2_USING_BOOMERANG
OCF2_FATIGUE_DRAINING
OCF2_SLOW_PARTY
OCF2_NO_DECAY
OCF2_NO_PICKPOCKET
OCF2_NO_BLOOD_SPLOTCHES
OCF2_NIGH_INVULNERABLE
OCF2_ELEMENTAL
OCF2_DARK_SIGHT
OCF2_NO_SLIP
OCF2_NO_DISINTEGRATE
OCF2_TARGET_LOCK
OCF2_ACTION*_PAUSED

NPC Flags (ONF_*)

ONF_EX_FOLLOWER
ONF_WAYPOINTS_DAY
ONF_WAYPOINTS_NIGHT

ONF_AI_WAIT_HERE
ONF_AI_SPREAD_OUT
ONF_JILTED -
ONF_LOGBOOK_IGNORES
ONF_KOS
ONF_USE_ALERTPOINTS
ONF_FORCED_FOLLOWER
ONF_KOS_OVERRIDE
ONF_WANDERS
ONF_WANDERS_IN_DARK
ONF_FENCE
ONF_FAMILIAR
ONF_CHECK_LEADER
ONF_CAST_HIGHEST
ONF_GENERATOR
ONF_GENERATED
ONF_GENERATOR_RATE1
ONF_GENERATOR_RATE2
ONF_GENERATOR_RATE3
ONF_DEMAINTAIN_SPELLS
ONF_BACKING_OFF
ONF_NO_ATTACK
ONF_BOSS_MONSTER
ONF_EXTRAPLANAR

This list is, of course, nicked from the forum **HERE**. Now, it is not the purpose of this blog to explain the meaning of these - thats what the thread is for - but rather to explain how to access them in-game. I will do this by demonstrating one of each type using an example from the game I know works. Lets start with a look at the most obvious:

OF_OFF

This is the most frequently used flag in scripts. Simply put, it turns the critter on or off: when off, the critter can still fire its scripts (eg heartbeat scripts) but will not be drawn or turn up on the radar for find_npc_near or anything like that.

To access the flag in a script, use the following commands:

_____attachee.object_flag_set(OF_OFF)

_____attachee.object_flag_unset(OF_OFF)

Simple, init? And so it goes for the various other Object Flags (OF_...). Or so I assume, no time to test them all! But note that a .mob set OF_OFF canNOT be unset by this flag! (Thanks Cerulean the Blue). I will keep an eye out for what can be over-ridden in the .mobs by scripts and what can't - I don't know if this is the only one or if that is a blanket thing. Annoying if it is - but I don't think so, because, (as you will see below) KOS_OVERRIDE in a script will speak over a .mobbed KOS flag (which is not the same as unsetting it... hmmm... wonder if the override exists precisely *because* you can't unset a .mobbed flag? More testing needed!)

Moving right along for the critter flags:

OCF_MUTE

This was sorta discovered by Liv and Dulcaion, I don't think it turns up in scripts in the original ToEE but is found a lot nowdays. It allows monsters to speak under very specific circumstances who would normally be mute (if they were not mute, they would make random comments if you snuck up and clicked on them, like NPCs without specific dlg files do. Having dire rats say, "good morning sir" when u click on them is less than necessary). So where the OCF_MUTE flag may be set by the prototype or the mob, it can be overriden and switched off in specific circumstances by the script, then reactivated when you are done talking to it. (The

thread dealing with all this is
http://www.co8.org/forum/showthread.php?t=1375&highlight=Update
and is well worth reading for the issue at hand. It is also well worth reading for several other nostalgic reasons but a tutorial is no place to go into it. Note that I disagree with Dulcaion's assertion that unsetting a flag breaks the script - that sounds like the classic symptom of a minor cockup, I've had many. More testing is needed.)

So what do the scripts look like? Here they are, hot from the .py files:

```
def san_first_heartbeat( attachee, triggerer ):
_________if (attachee.map != 5080 or game.global_flags[813] == 1):
____________________attachee.critter_flag_set( OCF_MUTE )
_________game.new_sid = 0
_________return RUN_DEFAULT
```

Now... note that previoulsy I said it was ***flagS*** where the previous (object) and following (npc) flags are in the singular. I have now REVERSED that opinion, based on more extensive reading of Dulcaion's tests, and also some of my own experience. Thus that little script above has been **altered** from the one in-game, which is, again based on my experience, broken. Took me a while for it to sink in. (Thats a really perverse use of game.new_sid too, in context). That flag is meant to mute trolls found elsewhere than dungeon level 4 (where u can try to beat a password for the trap out of them. They don't actually know it - damn Liv was a champ! A red herring no less!) However, long-term readers of the forum will remember a funny pic I posted where I clicked on a troll and he said, "me no see you" because my guy was invisible. Now, my troll was on dungeon lvl THREE, in Thrommel's room (a random encounter caused by sleeping). It has always niggled at the back of my mind, why wasn't he mute when he wasn't on lvl 4? Leaving aside the issue of whether muting affects floating text (Liv pretty conclusively showed it does, Dulcaion disagreed) I think that shows the code I originally posted (saying **attachee.critter_flag*S*_set(OCF_MUTE)**) was probably broken. Also there is the common sense element that it should not be in a different form than the other flags (which require a singular flag thing) and finally, in Dulcaion's final post on this issue he made exactly this point.

So barring conclusive testing otherwise, thats gonna be my official position.

Damn I hate it when I post something wrong in my tutorials! Hate it... please understand, I try to stick to things I have personally done or used, but periodically use scripts I have come across by Liv but not personally seen in action (like here), or in the game that I know werk. Let me at least assure you I NEVER just make stuff up and post it on the basis that it SHOULD work: amyone who has modded knows u hit problems with things that 'should' work all the time, thats why we need tutorials. Anyways, my apologies for the error.

So... npc flags :-)

ONF_KOS

Here's a common flag, and well worth being able to manipulate since changing critters to KOS (or not) depending on the player's actions makes for a nicely interactive game. Setting the flag in the prototype or mob is straightforward enough, but what about changes in game? One way is to use the **san_will_kos** thing in the protoype. That can look something like this:

```
def san_will_kos( attachee, triggerer ):
_____if (game.global_flags[840] == 1 and attachee.map == 5065):
_____________return SKIP_DEFAULT
_____return RUN_DEFAULT
```

Simple enough, and speaks volumes about the difference between skipping and running the default scripts.

What about changing it on the fly from scripts? I had to discover this one myself, and what a pita it was!!! But it was damn useful since of course it will work for all NPC flags. Here is a snapshot of the upcoming Moathouse respawn:

```
_____________bear1 = game.obj_create( 14052, location_from_axis (429L, 437L) )
```

```
_____bear1.rotation = 2
_____bear1.npc_flag_set( ONF_KOS )
_____bear2 = game.obj_create( 14053, location_from_axis (427L, 447L) )
_____bear2.rotation = 3
_____bear2.npc_flag_set( ONF_KOS )
```

I spawned some bears wandering around outside the walls of the moathouse, but since they were usually animal familiars, they had no KOS flags set. Can't have summoned animals attacking the party ;-)

Now... this brings us to the KOS_OVERRIDE script. I added one of these to KotB as I have a guy who runs amok under certain conditions, but unless u have met those conditions you should never get near him. So, I flagged him KOS with a thought to maybe altering it later. Well, later is fast approaching ;-) so I left him .mobbed as KOS and added the following script:

```
def san_first_heartbeat( attachee, triggerer ):
_____if game.quests[4].state == qs_unknown:
_____attachee.npc_flag_set( ONF_KOS_OVERRIDE )
_____return RUN_DEFAULT
```

Damn backwards ;-) should leave him normal then make him KOS when the time is right, but meh I say. It'll do for now - one reason I often do things backwards or in weird ways is to learn how to do new stuff. This worked for that end, so stop complaining!

Speaking of first_heartbeat files, here's a reminder of something I also put in the *Well Whaddya Know?* thread: The first time you enter a map, all the first_heartbeat files will fire everywhere. However, if you have been there before, they only fire as you enter their sector. So if you are adding to a first_heartbeat file, as well as using a save from off the map (I made that mistake with the file above, then wondered why it didn't work ;-)) you may have to enter the sector to get it to work, EVEN THOUGH if you have mobbed something new in it will work straight away whatever sector its in. That can be confusing if u r using little tells like having

game.particles fire to let you know certain internal effects are working: when some work and others don't, you of course assume you have screwed up somewhere. Try walking into the sector first.

Ok, thats our basic NPC flags dealt with (OCF2 still to do). What about item flags? Mr Blue mentioned the other day that item flags look like this, as you might expect:

obj.item_flag_set(OIF_FLAG)

Again, its just common sense :-)

Soooooooo... what about reading flags? Here's something complicated by Dulcaion that will probably tell you more about reading flags than you want to know!

- First I collected a list of all the critters in the vicinity of my party (that includes the party members), calling it a

>>> a = game.obj_list_vicinity(game.party[0].location, OLC_CRITTERS)

- I then examined a and found out that the Murderous thief was item 5 (a[5]) with simply

>>> a

- I then examined OCF_MUTE to find out its binary value (it's 8192)

>>> OCF_MUTE

For reference, 8192 in binary is
0000 0000 0000 0010 0000 0000 0000

- Then I checked his critter flags

>>> a[5].critter_flags_get()

This returns 134488064, which is (in binary):
1000 0000 0100 0010 0000 0000 0000

So you can see that the mute flag IS set (the 0010 column lines up with a 1 between the two numbers in binary)

- Then I unset the mute flag with

>>> a[5].critter_flags_unset(OCF_MUTE) *(Ted says: don't believe that flag**S**!!)*

- I checked the results

>>> a[5].critter_flags_get()

this time, it was 134479872, or
1000 0000 0100 0000 0000 0000 0000

so you can see that the mute flag WAS cleared by the command.

I know its a pain me posting things and adding, "Btw I haven't tested it but its wrong" but again this was by Dulcaion and he basically added an errata saying as much himself. Anyways, I think that is well worth reading all that, because the flags themselves are of course stored as binary within the .mobs. This is why you can't click, say, the 'items' thing in ToEEWB on a .mob for an NPC: at the appropriate part of the .mob, clicking that will add

0000 0000 0000 0000 0000 0000 0000

to the .mob and kaboom, the damn thing doesn't work because NPC .mobs aren't meant to have item flags and the game can't read what it sees. It also explains no matter how many

flags you set or unset, the .mob file doesn't change size, only adding a new TYPE of flag (or indeed all the flags of that type for the same size-price) will change the size. We don't all have to be Agetians, but every little bit of understanding helps. :-)

So, lets go back and see about the flag-reading thing. Here's another example from the upcoming moathouse spawn: and let me assure you this works, because I tested my a$$ off on it, complete with many private discussions with Mr Blue, and many different formats. Turned out I was doing the right thing all along - it was the fact my first_heartbeat files weren't firing on subsequent trips til I entered the sector that was making me think nothing was working. I sent C-Blue a screaming ranting (but also exulting) email about the evils of first_heartbeat files when I finally got to the end of it, and he hasn't spoken to me since ;-)

```
def san_first_heartbeat( attachee, triggerer ):
_____y = (attachee.critter_flags_get() & OCF_MUTE)
_____bandit1 = find_npc_near(attachee,14070)
_____bugbear2 = find_npc_near(attachee,14172)
_____gnolls1 = find_npc_near(attachee,14079)
_____gnolls2 = find_npc_near(attachee,14066)
_____if (attachee.map == 5005 and game.areas[4] == 1):
_____________if game.global_vars[52] >= 2:
_____________________game.obj_create( ??? )
_____________________game.obj_create( ??? )
_____________________game.obj_create( ??? )
_____________________game.obj_create( ??? )
_____________________game.global_vars[52] = 0
_____________if y == 0:
_____________________game.particles( "sp-summon monster I", game.party[1] )
_____________________if (bugbear2 == OBJ_HANDLE_NULL):
_____________________________game.obj_create( ??? )
_____________________________game.obj_create( ??? )
_____________________________game.obj_create( ??? )
```

```
_____________________________________game.obj_create( ??? )
_____________________________________game.obj_create( ??? )
_____________________________attachee.destroy()
___________________else:
_____________________________game.particles( "sp-summon monster I", game.party[0] )
_____________________________if ((gnolls1 == OBJ_HANDLE_NULL) and (gnolls2 ==
OBJ_HANDLE_NULL) and game.global_flags[288] != 1):
_____________________________________game.obj_create( ??? )
_____________________________________game.obj_create( ??? )
_____________________________________game.obj_create( ??? )
_____________________________________game.obj_create( ??? )
_____________________________attachee.destroy()
_________else:
___________________return SKIP_DEFAULT
_________return RUN_DEFAULT
```

Note the game.particles are merely there for testing purposes, I have left out what is spawned so there will be some surprises, and there is lots more to this (all the spawning stuff for the upstairs and outside). They were simple enough, make a .mob, stick it somewhere where it could tell if the players have been through or not (eg the outside one is in the middle of the bandits) then do a quick nearby NPC check to seee if the players went that way and wasted all the monsters. If so, spawn - if not, don't spawn. 'Game.area[4] == 1' is to check if the players have found the temple yet, thats the trigger for this new lot of content - if they have, then they have been gone long enough for the bodies to rot (dead monsters still show up on find_npc_near checks!) and they are a healthy level to come back and try this (ok, they may have just gone to Nulb, got the location from Otis, then raced back here, but if they do it that quick the bodies may not have rotted, so nothing will spawn - EVER - and they have ruined their added content. Ne'er mind - at that level they would not have survived these encounters anyways ;-))

So much for the upper levels (keep in mind there are not new critters in every room, just a few

new encounters here and there to show what the new owner of the place is up to - namely, adding undead as she can). BUT for the lower levels, there are a number of encounters the players may never have done - if they use the secret stairs to go via the Zombies and Ghouls (skipping the much tougher Lugbash / Gnolls / Bugbears encounters) they could easily have gone straight through to Lareth, killed his men (one web and a couple charms and then sit back and enjoy :-)), dealt with him, gone out the back door, then come back later. Can't just add a bunch of new monsters to the gnolls' or bugbears' rooms under such circumstances, they will set on the gnolls and bugbears! (The gnolls have been respawned as skeletal gnolls, as you might expect - something rather more nasty has moved into the bugbears' old apartments). So, I need multiple .mobs running **find_npc_near** in different places (or, a counter added to their san_dying files, but if said bugbears and gnolls in the protos.tab turn up somewhere else, eg in random encounters or in new content added by people saying "hey those protos look ideal", this could be triggered wrongly.)

Multiple prototypes? Nah, too much effort ;-)

Multiple .mobs of one prototype? But then they will all be using the same first_heartbeat script. That means a lot of global flags, dunnit?

In the past, yes! But today, nah... u should know now how much I hate using global flags if I can help it. And as it turns out, I can help it here :-)

What I did was add OCF_MUTE to one of the mobs but not the other, and differentiated between them that way :-) Thats what the first variable 'y' does, it checks to see whether the attachee (that specific .mob) has the flag set, by logically ANDing it with the flags as read. If it does, its the .mob sitting in the middle of the gnolls (OF_OFF of course.
game.global_flags[288], btw, is set by u bribing the gnolls: it would mean they would not show up as there, but of course they shouldn't have been lying around as corpses waiting to be reanimated as skeletal gnolls either, so in that case they don't show).

If it returns a zero on the AND, meaning the flag was not set, I know I am dealing with the

bugbears' .mob, and it can test if they are around (I had a bugbear1 test as well but he would pick up one of the ones hiding in the little rooms behind the doors. People could easily forget them or not bother with them - I know I have done both in the past - so I am not going to care if they are there or not, as long as the rest have been dealt with, thats ok. If people want to pull the hiding bugbears into combat at their rear while dealing with the new monsters, thats their business).

Either way, they only fire once then are destroyed. The first to fire also deals with the issue of the main inhabitant (who is set up in Lareth's old rooms of course, not much of a spoiler that!), thats the first little spawny thing. **game.global_vars[52]** was put in the san_dying thing of Lareth's leiutenant and seargant, two unique critters who should not be showing up elsewhere, and as long as they are both dead this will fire (then it is reset to 0, only fires once). If you **suggested** one of them, then u r in trouble - but you shouldn't have **suggestion** at that level anyways!

Quick recap: to find if a critter flag is set, use:

attachee.critter_flags_get() & OCF_FLAG

Note that yes it IS flag***S*** when getting but not when setting!!! That is tested and undisputed!

Flags Tutorial (Part 2)

Ok, time for object flags. Rather than do an update, here is a new post (cause we already have great long lists in part 1) ;-).

These are flags (or 'identifiers') that are attached to specific objects (funnily enough), not just to prototypes (which might occur several times on a map. For instance, you might spawn or .mob in several goblins or skels or whatever in a dungeon, or a couple helms in an armoury). Last time you saw a manner in which I faked a way around it: I had 2 .mobs on a map of identical protos and (therefore) with identical scripts, and distinguished between them by setting different OCF flags in their .mobs. Today we will look at the real deal. These are internal flags

so they can be read more than they can be set ;-) Also, they can happily crash a game if u stuff up by setting an item flag for an NPC or whatever, just like a mob (probably like a prototype too but never tried). And indeed I don't know if some of them won't crash the game anyway, if they are not supported (Arcanum anyone?) In this racket, you gotta get used to trial-and-error people, its the only way.

Now, first a word from C-Blue on this subject, then a list of ALL the flags. I like lists. 8^D For those who want some downloadable versions, check this thread **HERE**.

Ok, over to ol' Red-Eye:

All the fields can be read, but not all of them can be set. The " _f_" fields are floating point and the " _i_" fields are integer. The fields with 64 in them are 64 bit, the fields that end in "as" are arrays, and the fields that end in "idx" are indexes for arrays. Don't mess with those. The get_int function only reads 32 bit integers. That is pretty much all I know about these fields. Trial and error has shown me what I can set and what I can't. I can't remember off hand what all is what though. That's why I have the spreadsheet now, so I can record this stuff.

We'll lean on him later for this list: you're gonna have to get used to these people, we are gonna use them a LOT for KotB, where global flags and variables will be left for GLOBAL events, not just to check if your project NPC has had his breakfast or not. Why set a global flag for an object when you can just read it straight from the object's internal flags? Madness, I tells ya. For now, we'll look at some flags that can be used as internal variables. But first, the list:

obj_f_begin=0
obj_f_current_aid=1
obj_f_location=2
obj_f_offset_x=3
obj_f_offset_y=4
obj_f_shadow=5
obj_f_overlay_fore=6

obj_f_overlay_back=7
obj_f_underlay=8
obj_f_blit_flags=9
obj_f_blit_color=10
obj_f_blit_alpha=11
obj_f_scale=12
obj_f_light_flags=13
obj_f_light_aid=14
obj_f_light_color=15
obj_f_overlay_light_flags=16
obj_f_overlay_light_aid=17
obj_f_overlay_light_color=18
obj_f_flags=19
obj_f_spell_flags=20
obj_f_blocking_mask=21
obj_f_name=22
obj_f_description=23
obj_f_aid=24
obj_f_destroyed_aid=25
obj_f_size=26
obj_f_hp_pts=27
obj_f_hp_adj=28
obj_f_hp_damage=29
obj_f_material=30
obj_f_scripts_idx=31
obj_f_sound_effect=32
obj_f_category=33
obj_f_rotation=34
obj_f_speed_walk=35
obj_f_speed_run=36
obj_f_base_mesh=37

obj_f_base_anim=38
obj_f_radius=39
obj_f_3d_render_height=40
obj_f_conditions=41
obj_f_condition_arg0=42
obj_f_permanent_mods=43
obj_f_initiative=44
obj_f_dispatcher=45
obj_f_subinitiative=46
obj_f_secretdoor_flags=47
obj_f_secretdoor_effectname=48
obj_f_secretdoor_dc=49
obj_f_pad_i_7=50
obj_f_pad_i_8=51
obj_f_pad_i_9=52
obj_f_pad_i_0=53
obj_f_offset_z=54
obj_f_rotation_pitch=55
obj_f_pad_f_3=56
obj_f_pad_f_4=57
obj_f_pad_f_5=58
obj_f_pad_f_6=59
obj_f_pad_f_7=60
obj_f_pad_f_8=61
obj_f_pad_f_9=62
obj_f_pad_f_0=63
obj_f_pad_i64_0=64
obj_f_pad_i64_1=65
obj_f_pad_i64_2=66
obj_f_pad_i64_3=67
obj_f_pad_i64_4=68

obj_f_last_hit_by=69
obj_f_pad_obj_1=70
obj_f_pad_obj_2=71
obj_f_pad_obj_3=72
obj_f_pad_obj_4=73
obj_f_permanent_mod_data=74
obj_f_attack_types_idx=75
obj_f_attack_bonus_idx=76
obj_f_strategy_state=77
obj_f_pad_ias_4=78
obj_f_pad_i64as_0=79
obj_f_pad_i64as_1=80
obj_f_pad_i64as_2=81
obj_f_pad_i64as_3=82
obj_f_pad_i64as_4=83
obj_f_pad_objas_0=84
obj_f_pad_objas_1=85
obj_f_pad_objas_2=86
obj_f_end=87

obj_f_portal_begin=88
obj_f_portal_flags=89
obj_f_portal_lock_dc=90
obj_f_portal_key_id=91
obj_f_portal_notify_npc=92
obj_f_portal_pad_i_1=93
obj_f_portal_pad_i_2=94
obj_f_portal_pad_i_3=95
obj_f_portal_pad_i_4=96
obj_f_portal_pad_i_5=97
obj_f_portal_pad_obj_1=98

obj_f_portal_pad_ias_1=99
obj_f_portal_pad_i64as_1=100
obj_f_portal_end=101

obj_f_container_begin=102
obj_f_container_flags=103
obj_f_container_lock_dc=104
obj_f_container_key_id=105
obj_f_container_inventory_num=106
obj_f_container_inventory_list_idx=107
obj_f_container_inventory_source=108
obj_f_container_notify_npc=109
obj_f_container_pad_i_1=110
obj_f_container_pad_i_2=111
obj_f_container_pad_i_3=112
obj_f_container_pad_i_4=113
obj_f_container_pad_i_5=114
obj_f_container_pad_obj_1=115
obj_f_container_pad_obj_2=116
obj_f_container_pad_ias_1=117
obj_f_container_pad_i64as_1=118
obj_f_container_pad_objas_1=119
obj_f_container_end=120

obj_f_scenery_begin=121
obj_f_scenery_flags=122
obj_f_scenery_pad_obj_0=123
obj_f_scenery_respawn_delay=124
obj_f_scenery_pad_i_0=125
obj_f_scenery_pad_i_1=126
obj_f_scenery_teleport_to=127

obj_f_scenery_pad_i_4=128
obj_f_scenery_pad_i_5=129
obj_f_scenery_pad_obj_1=130
obj_f_scenery_pad_ias_1=131
obj_f_scenery_pad_i64as_1=132
obj_f_scenery_end=133

obj_f_projectile_begin=134
obj_f_projectile_flags_combat=135
obj_f_projectile_flags_combat_damage=136
obj_f_projectile_parent_weapon=137
obj_f_projectile_parent_ammo=138
obj_f_projectile_part_sys_id=139
obj_f_projectile_acceleration_x=140
obj_f_projectile_acceleration_y=141
obj_f_projectile_acceleration_z=142
obj_f_projectile_pad_i_4=143
obj_f_projectile_pad_obj_1=144
obj_f_projectile_pad_obj_2=145
obj_f_projectile_pad_obj_3=146
obj_f_projectile_pad_ias_1=147
obj_f_projectile_pad_i64as_1=148
obj_f_projectile_pad_objas_1=149
obj_f_projectile_end=150

obj_f_item_begin=151
obj_f_item_flags=152
obj_f_item_parent=153
obj_f_item_weight=154
obj_f_item_worth=155
obj_f_item_inv_aid=156

obj_f_item_inv_location=157
obj_f_item_ground_mesh=158
obj_f_item_ground_anim=159
obj_f_item_description_unknown=160
obj_f_item_description_effects=161
obj_f_item_spell_idx=162
obj_f_item_spell_idx_flags=163
obj_f_item_spell_charges_idx=164
obj_f_item_ai_action=165
obj_f_item_wear_flags=166
obj_f_item_material_slot=167
obj_f_item_quantity=168
obj_f_item_pad_i_1=169
obj_f_item_pad_i_2=170
obj_f_item_pad_i_3=171
obj_f_item_pad_i_4=172
obj_f_item_pad_i_5=173
obj_f_item_pad_i_6=174
obj_f_item_pad_obj_1=175
obj_f_item_pad_obj_2=176
obj_f_item_pad_obj_3=177
obj_f_item_pad_obj_4=178
obj_f_item_pad_obj_5=179
obj_f_item_pad_wielder_condition_array=180
obj_f_item_pad_wielder_argument_array=181
obj_f_item_pad_i64as_1=182
obj_f_item_pad_i64as_2=183
obj_f_item_pad_objas_1=184
obj_f_item_pad_objas_2=185
obj_f_item_end=186

obj_f_weapon_begin=187
obj_f_weapon_flags=188
obj_f_weapon_range=189
obj_f_weapon_ammo_type=190
obj_f_weapon_ammo_consumption=191
obj_f_weapon_missile_aid=192
obj_f_weapon_crit_hit_chart=193
obj_f_weapon_attacktype=194
obj_f_weapon_damage_dice=195
obj_f_weapon_animtype=196
obj_f_weapon_type=197
obj_f_weapon_crit_range=198
obj_f_weapon_pad_i_1=199
obj_f_weapon_pad_i_2=200
obj_f_weapon_pad_obj_1=201
obj_f_weapon_pad_obj_2=202
obj_f_weapon_pad_obj_3=203
obj_f_weapon_pad_obj_4=204
obj_f_weapon_pad_obj_5=205
obj_f_weapon_pad_ias_1=206
obj_f_weapon_pad_i64as_1=207
obj_f_weapon_end=208

obj_f_ammo_begin=209
obj_f_ammo_flags=210
obj_f_ammo_quantity=211
obj_f_ammo_type=212
obj_f_ammo_pad_i_1=213
obj_f_ammo_pad_i_2=214
obj_f_ammo_pad_obj_1=215
obj_f_ammo_pad_ias_1=216

obj_f_ammo_pad_i64as_1=217
obj_f_ammo_end=218

obj_f_armor_begin=219
obj_f_armor_flags=220
obj_f_armor_ac_adj=221
obj_f_armor_max_dex_bonus=222
obj_f_armor_arcane_spell_failure=223
obj_f_armor_armor_check_penalty=224
obj_f_armor_pad_i_1=225
obj_f_armor_pad_ias_1=226
obj_f_armor_pad_i64as_1=227
obj_f_armor_end=228

obj_f_money_begin=229
obj_f_money_flags=230
obj_f_money_quantity=231
obj_f_money_type=232
obj_f_money_pad_i_1=233
obj_f_money_pad_i_2=234
obj_f_money_pad_i_3=235
obj_f_money_pad_i_4=236
obj_f_money_pad_i_5=237
obj_f_money_pad_ias_1=238
obj_f_money_pad_i64as_1=239
obj_f_money_end=240

obj_f_food_begin=241
obj_f_food_flags=242
obj_f_food_pad_i_1=243
obj_f_food_pad_i_2=244

obj_f_food_pad_ias_1=245
obj_f_food_pad_i64as_1=246
obj_f_food_end=247

obj_f_scroll_begin=248
obj_f_scroll_flags=249
obj_f_scroll_pad_i_1=250
obj_f_scroll_pad_i_2=251
obj_f_scroll_pad_ias_1=252
obj_f_scroll_pad_i64as_1=253
obj_f_scroll_end=254

obj_f_key_begin=255
obj_f_key_key_id=256
obj_f_key_pad_i_1=257
obj_f_key_pad_i_2=258
obj_f_key_pad_ias_1=259
obj_f_key_pad_i64as_1=260
obj_f_key_end=261

obj_f_written_begin=262
obj_f_written_flags=263
obj_f_written_subtype=264
obj_f_written_text_start_line=265
obj_f_written_text_end_line=266
obj_f_written_pad_i_1=267
obj_f_written_pad_i_2=268
obj_f_written_pad_ias_1=269
obj_f_written_pad_i64as_1=270
obj_f_written_end=271

obj_f_bag_begin=272
obj_f_bag_flags=273
obj_f_bag_size=274
obj_f_bag_end=275

obj_f_generic_begin=276
obj_f_generic_flags=277
obj_f_generic_usage_bonus=278
obj_f_generic_usage_count_remaining=279
obj_f_generic_pad_ias_1=280
obj_f_generic_pad_i64as_1=281
obj_f_generic_end=282

obj_f_critter_begin=283
obj_f_critter_flags=284
obj_f_critter_flags2=285
obj_f_critter_abilities_idx=286
obj_f_critter_level_idx=287
obj_f_critter_race=288
obj_f_critter_gender=289
obj_f_critter_age=290
obj_f_critter_height=291
obj_f_critter_weight=292
obj_f_critter_experience=293
obj_f_critter_pad_i_1=294
obj_f_critter_alignment=295
obj_f_critter_deity=296
obj_f_critter_domain_1=297
obj_f_critter_domain_2=298
obj_f_critter_alignment_choice=299
obj_f_critter_school_specialization=300

obj_f_critter_spells_known_idx=301
obj_f_critter_spells_memorized_idx=302
obj_f_critter_spells_cast_idx=303
obj_f_critter_feat_idx=304
obj_f_critter_feat_count_idx=305
obj_f_critter_fleeing_from=306
obj_f_critter_portrait=307
obj_f_critter_money_idx=308
obj_f_critter_inventory_num=309
obj_f_critter_inventory_list_idx=310
obj_f_critter_inventory_source=311
obj_f_critter_description_unknown=312
obj_f_critter_follower_idx=313
obj_f_critter_teleport_dest=314
obj_f_critter_teleport_map=315
obj_f_critter_death_time=316
obj_f_critter_skill_idx=317
obj_f_critter_reach=318
obj_f_critter_subdual_damage=319
obj_f_critter_pad_i_4=320
obj_f_critter_pad_i_5=321
obj_f_critter_sequence=322
obj_f_critter_hair_style=323
obj_f_critter_strategy=324
obj_f_critter_pad_i_3=325
obj_f_critter_monster_category=326
obj_f_critter_pad_i64_2=327
obj_f_critter_pad_i64_3=328
obj_f_critter_pad_i64_4=329
obj_f_critter_pad_i64_5=330
obj_f_critter_damage_idx=331

obj_f_critter_attacks_idx=332
obj_f_critter_seen_maplist=333
obj_f_critter_pad_i64as_2=334
obj_f_critter_pad_i64as_3=335
obj_f_critter_pad_i64as_4=336
obj_f_critter_pad_i64as_5=337
obj_f_critter_end=338

obj_f_pc_begin=339
obj_f_pc_flags=340
obj_f_pc_pad_ias_0=341
obj_f_pc_pad_i64as_0=342
obj_f_pc_player_name=343
obj_f_pc_global_flags=344
obj_f_pc_global_variables=345
obj_f_pc_voice_idx=346
obj_f_pc_roll_count=347
obj_f_pc_pad_i_2=348
obj_f_pc_weaponslots_idx=349
obj_f_pc_pad_ias_2=350
obj_f_pc_pad_i64as_1=351
obj_f_pc_end=352

obj_f_npc_begin=353
obj_f_npc_flags=354
obj_f_npc_leader=355
obj_f_npc_ai_data=356
obj_f_npc_combat_focus=357
obj_f_npc_who_hit_me_last=358
obj_f_npc_waypoints_idx=359
obj_f_npc_waypoint_current=360

obj_f_npc_standpoint_day_INTERNAL_DO_NOT_USE=361
obj_f_npc_standpoint_night_INTERNAL_DO_NOT_USE=362
obj_f_npc_faction=363
obj_f_npc_retail_price_multiplier=364
obj_f_npc_substitute_inventory=365
obj_f_npc_reaction_base=366
obj_f_npc_challenge_rating=367
obj_f_npc_reaction_pc_idx=368
obj_f_npc_reaction_level_idx=369
obj_f_npc_reaction_time_idx=370
obj_f_npc_generator_data=371
obj_f_npc_ai_list_idx=372
obj_f_npc_save_reflexes_bonus=373
obj_f_npc_save_fortitude_bonus=374
obj_f_npc_save_willpower_bonus=375
obj_f_npc_ac_bonus=376
obj_f_npc_add_mesh=377
obj_f_npc_waypoint_anim=378
obj_f_npc_pad_i_3=379
obj_f_npc_pad_i_4=380
obj_f_npc_pad_i_5=381
obj_f_npc_ai_flags64=382
obj_f_npc_pad_i64_2=383
obj_f_npc_pad_i64_3=384
obj_f_npc_pad_i64_4=385
obj_f_npc_pad_i64_5=386
obj_f_npc_hitdice_idx=387
obj_f_npc_ai_list_type_idx=388
obj_f_npc_pad_ias_3=389
obj_f_npc_pad_ias_4=390
obj_f_npc_pad_ias_5=391

obj_f_npc_standpoints=392
obj_f_npc_pad_i64as_2=393
obj_f_npc_pad_i64as_3=394
obj_f_npc_pad_i64as_4=395
obj_f_npc_pad_i64as_5=396
obj_f_npc_end=397

obj_f_trap_begin=398
obj_f_trap_flags=399
obj_f_trap_difficulty=400
obj_f_trap_pad_i_2=401
obj_f_trap_pad_ias_1=402
obj_f_trap_pad_i64as_1=403
obj_f_trap_end=404

obj_f_total_normal=405
obj_f_transient_begin=406
obj_f_render_color=407
obj_f_render_colors=408
obj_f_render_palette=409
obj_f_render_scale=410
obj_f_render_alpha=411
obj_f_render_x=412
obj_f_render_y=413
obj_f_render_width=414
obj_f_render_height=415
obj_f_palette=416
obj_f_color=417
obj_f_colors=418
obj_f_render_flags=419
obj_f_temp_id=420

obj_f_light_handle=421
obj_f_overlay_light_handles=422
obj_f_internal_flags=423
obj_f_find_node=424
obj_f_animation_handle=425
obj_f_grapple_state=426
obj_f_transient_end=427
obj_f_type=428
obj_f_prototype_handle=429

Try and have those memorised by Tuesday ;-)

Ok, enough bamboozling you, lets pick a specific one and play with it.

We are gonna use **obj_f_npc_pad_i_5=381** as a variable that will be changed according to the ongoing state of what is happening to the character. We are using that particular integer because Blue used it for the shop-keeper-reinventorising and I know it works ;-) Go and have a look at Burne's .py file, he uses things in a more integrated way than I bother with (I am keeping it simple).

My guinea-pig for today is Endarire's NPC Ronald. You may remember (but probably won't) that he already has a story-state variable dependant on what the party have done to him (like dump him to recruit Lareth, or other pleasant actions). If necessary, refamiliarise yourself from this tutorial **HERE** (hint: its not necessary for this tutorial).

Sooo... what do we want to do? Well:

Read the integer.

Set the integer.

Simple! For what purpose?

Well, its a story-state variable. The first time it will appear is when Ronald hits level 17 and gains the clerical power to take care of something that he has had on his mind throughout the entire game. So firstly, we want it to trigger an event when Ron becomes lvl 17 and then advance the variable accordingly. But where to put it? First_heartbeat is an option, but Ron's is already a fiasco. Soooo, lets put it for starters in san_new_map. At the moment, Ron's looks like this:

```
def san_new_map( attachee, triggerer ):
_____if (attachee.map == 5004):
_____________attachee.float_line(650,triggerer)
_____________game.new_sid = 0
_____return RUN_DEFAULT
```

What this does is have Ron make a quirky remark when he enters the Moathouse, then sets it to zero so it never happens again (Endarire not only recorded dialogue for all the stuff the NPCs can say in the game, like "my pack is full" and "hey look Emridy Meadows", bur also a fair bit for stuff they don't normally say, and left me to script workarounds. Here is one).

Instead of setting the bugger to zero, lets see if you can set it to some other number (of course theoretically we all know you can, but has anyone ever tried?) I am setting it to 283, my ioun stone .py file, in the happy knowledge it will never be asked to use a new_map thing (in fact, doesn't even have it in its prototype). Liv used to do this a lot, stick scripts in other scripts that wouldn't get called by their original owner: for instance, the OCF_MUTE thing for the trolls we looked at last time is actually in the script for that woman in Hommlet, ummm the one married to Filliken's brother (Percy?) who sets you on the whole love-quest roundabout.

NPC: Your just as sweet as sugar.
PC: Just doing my job, ma'am.

Something like that - and quality dialogue for once! Ahhh, didn't that run out quickly... anyways, first time I saw that thing in her file, I thought, "crikey, THATS who's got the password? She must be *suggest*ed or something... i wonder how we would know?" But of course it was nothing to do with her, just using her script. The "Peter, I've been murdered by gnolls!" interlude is likewise hidden in Kent's stuff.

But I digress. The point is, diverting a character to this or that script, while having inherent dangers (remember when I tried to run all the san_dying things off one script and Otis etc yelled out in Spugnoir's voice when they died?) is damn handy. Will be doing this for KotB, having different scripts for the guards to switch between depending on how they feel about the party.

Anyways, the altered script from py00277Ron.py looks like this:

```
def san_new_map( attachee, triggerer ): # after moathouse, sets to 283
_____if (attachee.map == 5004):
_____________attachee.float_line(650,triggerer)
_____________game.new_sid = 283
_____return RUN_DEFAULT
```

Painless. Now what about the accompanying file, py00283ioun.py? That now looks like this:

```
def san_new_map( attachee, triggerer ): # Ron after moathouse
_____game.particles( "sp-Magic Stone", triggerer )
_____st = attachee.obj_get_int( obj_f_npc_pad_i_5 )
_____if (st == 0 and (attachee.stat_level_get(stat_level_cleric) == 8)):
_____________attachee.obj_set_int( obj_f_npc_pad_i_5, 1 )
_____________game.party[0].begin_dialog( attachee, 2000 )
_____else:
_____________game.particles( "sp-summon monster I", game.party[0] )
_____return RUN_DEFAULT
```

The first and last game.particles things are just there to let me know it is firing (I had a shocker getting it to work, because I had a stupid typo in the script, but thought I had made some other sort of error - thought maybe i needed to do a second dlg at 00283 (I didn't) - but finally got it going). Otherwise, its the other stuff that is of interest.

Now, this is just to initiate the first "I have turned lvl 17" dialogue. Note I am using lvl 8 for the moment because I couldn't be bothered entering 'levelup' enough times just yet to get him to lvl 17 (I know its just typing it in, then 'up' + 'return' several dozen times, but I'm a lazy, lazy man Roger...) will do that when I am satisfied the script works. When that happens, there will be additional stuff here and in the san_dialog thing to advance the story through the various parts of the endgame, but thats for later. This is a flags tutorial, not a Ronald tutorial ;-)

So, reading and setting the flags? Here they are again:

st = attachee.obj_get_int(obj_f_npc_pad_i_5) (where 'st' is the name of the variable of course)

attachee.obj_set_int(obj_f_npc_pad_i_5, 1)

Note that in the second one, we are setting the value at 1 (for the moment). Thats whay the 1 is for. We could set it at something else, and will later. A bit different to how we set global variables or OCF_FLAGS etc.

Well, there you go, now your objects can have personal flags that will never clash with anyone else's - and for the moment you know as much about them as I do ;-) Later I will update this with stuff about how to use the personal flags to see if another NPC has done something or other (not sure how I will do that, but we'll see ;-) and how to read the internal health of your NPC:

obj_f_npc_who_hit_me_last=358

Mmm, roleplaying... me RIKEY!!!

O and while we're at it...

obj_f_npc_substitute_inventory=365

Change number in event of Masterwork quest? Hmmm, must test this, and there are several other contenders...

ToEEWB Tips and Tricks (and Traps)

Well I am off work for a couple days with a bit of a flu so a nice time to do a tutorial.

There have been a huge number of changes to ToEEWB lately, as usual Agetian is adding stuff faster than we can even figure it out ;-), and while a lot of it (such as the whole existance of Sector Sort) has been widely talked about, in a few weeks or so it may all be forgotten and newcomers to ToEEWB may still make some basic errors. So here are a list of the things I do to avoid mistakes, and a list of the common mistakes I still do every time but know how to remedy. I will probably update this or add a parts 2, 3, 4 etc as time goes by. And yes I may simply be repeating certain things found in the tutorials and documentation, but I'm fine with that 8^D

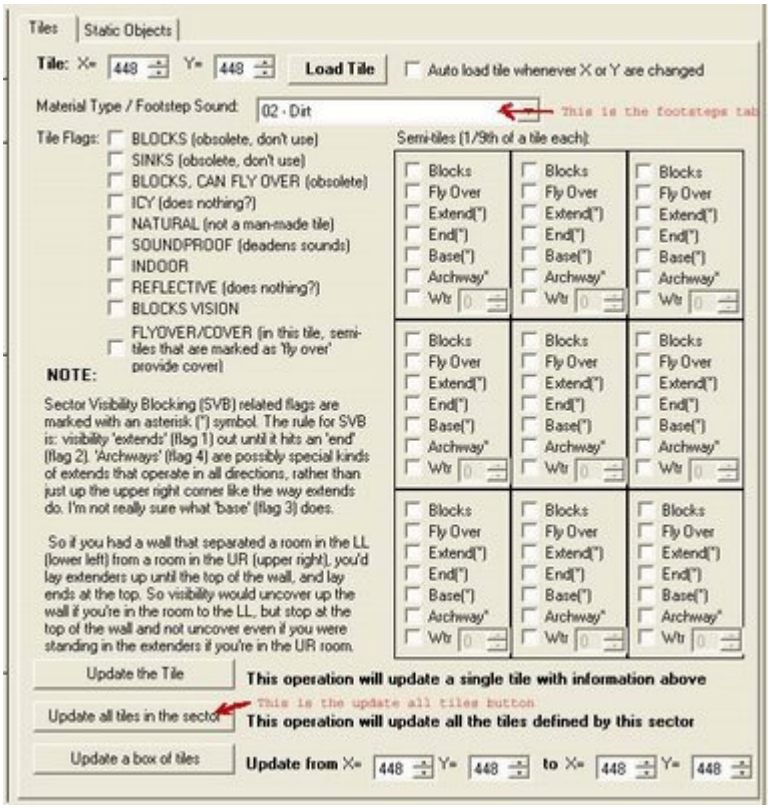
Tip 1: Do the footsteps first!

Damn I can't emphasise this enough people. If you are making an indoor map, or an outdoor map on stone, or anything where the dominant footstep sound will not be 'dirt' (the default one), the first thing u wanna do is flag every tile on the map as the dominant sound: stone, or wood for indoor floorboards, or ice (that works well for carpets) - they are about the only ones that work I think. Must see if we can hack some more in, the flags are there to be used :-).

How do u do that? Well I'm glad you asked. Simple enough: open your new sector, select the

little dooby marked 'Material Type / Footstep Sound' and select whatever you want. Then, update the WHOLE SECTOR, using the button 'Update all tiles in the sector'. This may take a while - it surely does on my 1.6Ghz P4. Then SAVE THE SECTOR.

Here's a pic:



Afterwards, you can run around on the sector, paint your walls or whatever, and you don't have to worry about going back and adding footsteps here and there - thats a pain. And yes, I wish I practised this all the time! About 90% of the maps in KotB so far I forgot to do it, in fact it is only a handful of maps here n there I remembered. Thats gonna be a pain 2 fix later, let me tell you.

Now: make sure you don't put anything in those little sub-tile boxes before you do this, cause if u mark so much as one subtile as blocked or something, you will mark every lawt tile as blocked and u will have to start again (u should just delete the file in that case).

O yeah - how do u have a file in the first place? Well the easiest thing is to go onto the map in question, mark one tile as blk() or fly() or whatever, that will create the sector file, then just do as above and it will copy over that one tile.

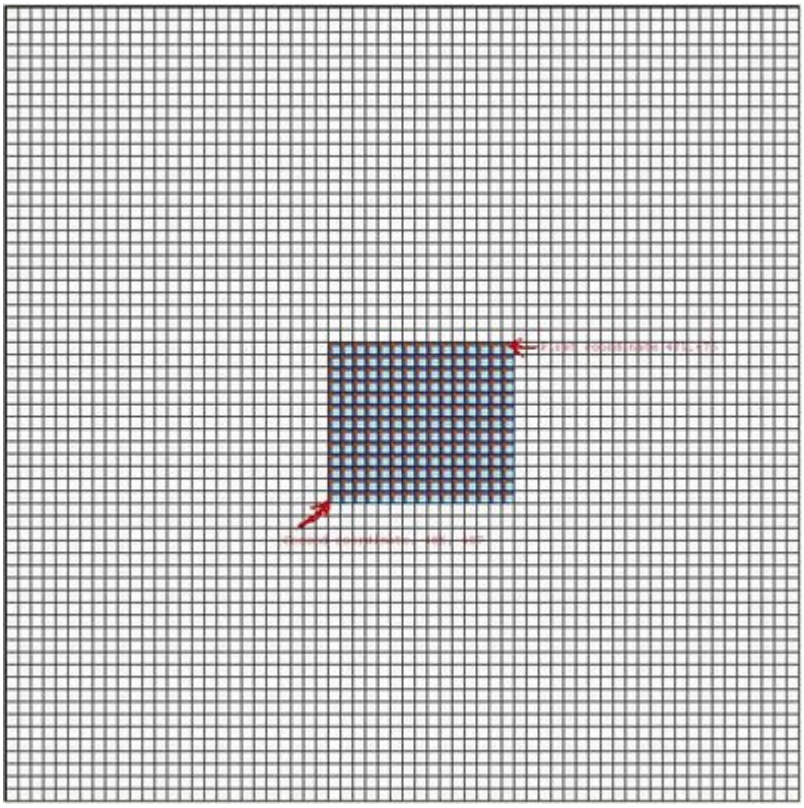
Tip 2: Getting the damn 'Update a box' thing to work

I screwed this up so often I eventually told Ag, "hey this doesn't work" and he said, "yeah it does, works fine for me" (this was a while back). It works logically rather than intuitively.

Have another look at the pic, the 'Update a box' button is at the bottom. There are coordinates next to it: you put in coordinates and the rectangle thus created forms a box to update.

Now, since there are only 2 coordinates, to create a box u have to put in the opposite corners, right? Absolutely, its common sense. BUT, if u intuitively put in the upper left / bottom right corners, you will get an error message. You have to put in top right / bottom left, because the tiles count forward in both x and y directions that way. Makes sense logically, but u have 2 think about it.

So for the following box I have marked the ones you do first and second. Note that I only marked the middle subtile as blk(), wtr() and extend svb to get this effect: if any part of a tile is marked, the whole tile will turn up in the Visual Sector Painter like that.



Tip 3: Don't add object flags to embedded items

Embedded suff - that is, scenery, doors, trees, ladder icons etc - shouldn't have certain instinctive flags added. I found this out when I tried to add 'OF_PROVIDES_COVER' to a tree, since it seemed reasonable enough that firing through its foilage would provide cover (of course, you sector the base as blk() so people can't fire through the trunk or run through it). What it did was make the tree 'interactive', so when I tried to run behind it, it instead took this as clicking 'on' the tree and my PC just ran up to it. Heck, you might be able to talk to trees this way, I don't know, must try that :-) But since it wasn't what I wanted (and a tree has a heck of a built-in size, there was this huge circle selected around the base when I clicked on it and nothing else could be added in its vicinity) it was a P.I.T.A. As a general rule of thumb though, you add effects to embedded items by sectoring them in, since they aerembedded in the sector file: to make them NO_BLOCK just leave them alone and you can walk through them, to make the SHOOT_THROUGH flag the tile as fly() etc. Leave the flags for the mobs.

What object flags do you need to add to a mob? If you do look at a mob from the game, you will find all sorts of object flags set: typically, OF_HEIGHT_SET, OF_RADIUS_SET, maybe OF_FLAT (for items), maybe OF_SEE_THROUGH, OF_SHOOT_THROUGH, OF_CLICK_THROUGH etc. I don't really bother with any of these except OF_SEE_THROUGH, because you don't have to set the radius, height etc in the mob - that is all in the prototype. Of course, if you WANT to, sure you can fiddle with these :-) Note that click through means just that, your 'clicks' go right through it, so don't set this on a monster with a dialogue file (or that is going into combat or anything).

Hmmm... maybe I should have made the trees 'OF_PROVIDES_COVER' *and* 'OF_CLICK_THROUGH'. Heh, only just thought of that. Well there u go ;-)

Anyways, the only flags I really bother with are OF_SEE_THROUGH and OF_PROVIDES_COVER(since you don't want monsters obscuring their surroundings that much, and critters should of course provide cover for critters behind them). By all means use more, but don't get spooked by them all. An incomplete list of their meanings can be found **HERE**.

Tip 4: You don't have to draw things for them to have an effect

This specifically refers to the infamous green fog ball. To get this happening, u flag it as 'OF_DONTDRAW'. Then you don't see it - but it still makes fog. Yippee :-) Here is a quick look at it in action in the woods of the mad hermit in KotB:

Fire in the sky... ;-)

For the record, I figured this out pretty quickly, its not the reason the ball showed up on revisits. That was due to the embedded items being out of order. To fix this, you use Sector Sort: recent readers will know this, future readers may not.

Tip 5: Use Sector Sort

Reminds me of an Aussie actor in an American accent declaiming, "Wear Sunscreen". The boss at work has been playing that a lot lately ;-)

Anyways, Sector Sort is available in the new 'add-ins' section of ToEEWB (its at the top with File, Help etc). It is easy to use, in fact just clicking on it will cause it to sort every item in your Sectors folder and then gives you the option of sorting stuff in other folders.

When to use it: Well really, any time you make a new sector ;-) But specifically, any time you find new items in sectors aren't working properly. This is particularly the case if you find the first time you go to a sector your embedded items work fine, but the second time they cause issues (I have had it with the green fog ball showing up, and with internal doors). Click Sector Sort, and wah-lah! O and no, you don't have to click save after, because it does it automatically to all the files whether you open them or not.

Tip 6: Making your background interactive

This is more of a KotB thing, but I think I will get this put into ToEE since people are adding so much new content nowdays.

Cuchulainn made a model with a transparent mesh (1 single pixel, to be precise). To use it, simply make a proto of whatever you want, an NPC, a container or whatever it is painted on your map you want to interact with (eg the talking tree mentioned above, this would be another way to make it), then use this model in the proto (its 10171 in KotB, {10171}{Weapons\handle_t} to be specific though it is not in the current release). You will interact with the model as far as the game engine is concrened, but you will *see* the painted background (and hovering text thing) so you are, to all intents and porpoises, interacting directly with what you see onscreen. Nice easy workaround for our lack of modelling capacity 8^) Here's a pic in action:

Yeah, thats right, I've added HOLLOW STUMPS. And this one has a MAGIC AXE in it. Whaddya gonna do about it?!?

[Ahem] Sorry, don't know what came over me. Anyway, that'll do for now. Next time, lights: like Ag said, it will all look better with lights (for the record, I HAVE been taking that into account - only have to look at how sweet that Gnarley Forest map came out :-))

Factions, Reactions, and Reputations

Today we are going to deal with the issue of reputations and reactions. These were rather poorly handled in ToEE, despite having a very good framework within which to operate - thats the story of the game, I guess. We saw back in ummmm tutorial 1 or 2 that you can start a dialogue with the generic greeting, {G:}{G:}{}{}{}{}. This would give a greeting based on various factors, including the reputation. Good stuff. However, beyond this there was no change, you went straight into the standard dialogue tree - "can I ask you some more questions" etc, and generally speaking they reacted identically. Sure, Black Jay or Filliken might hate you and refuse to talk to you for a while, but that was changed by doing definite things for them and flagging specific things such as quests etc. It was not based on reputation.

And yet, when we look at the **gamerep.mes** (in the **Rules** folder), we find that there is a very good framework for handling the whole reputation issue and allowing it to *automatically* have a real impact on the game, such that players actions could (if followed up) significantly affect their outcomes - ie, roleplaying. Lets do a cut-paste of what it says.

```
// Game reputations start at 1
// Each reputation starts with 3 faction numbers, indicating the three
// factions that will treat the reputation owner as being part of their
// faction. One or all of these faction numbers can be zero, which is
// the empty faction.
// Each reputation can then have up to 5 effects, separated by commas
// Each effect is 2 numbers, separated by spaces
// The first number is a reaction adjustment
// The second number is a faction (0 means ANY faction)
// Examples:
//
// 0 0 0, -20 0
// The reputation grants no faction to the PC, but he has a -20
// reaction from any NPC
```

```
//
// 1 10 0, -10 0
// Any NPC from factions 1 and 10 will treat the PC as being on that
// faction, and every NPC responds with a -10 reaction
//
// 1 0 0, +10 1
// Any NPC from faction 1 will treat the PC as being in that faction,
// and all NPC's belonging to faction 1 respond with a +10 reaction
//
// 0 0 0, +15 1
// The reputation grants no faction, but an NPC belonging to faction 1
// gets a +15 reaction to the PC
```

So we can see that we can have a player get a reputation (one of the things in the logbook - 'Prison Liberator', 'Moathouse Cleaner' and 'Butcher of Hommlet' all come to mind) and it can be set so that whole FACTIONS automatically react. That is, be mean to a guard and ALL the gaurds (if they are in the one faction) can be set to react nastily to you.

Brilliant! But we want to expand the reaction beyond just a nasty greeting. So, I would suggest scripting whole seperate dialogue options based on whether the player loves you or hates you or whatever. that sounds like a pain, doesn't it. BUT when you tihnk about it, its really just a question of the number of options:

- NPC is ambivalent to party (just met or whatever): has various options
- NPC loves party (they have done something for him, a quest or got a great rep somehow): has additional options, he is willing to do more for them
- NPC hates party (they have pissed him or his faction off somehow): has seriously restricted options, possibly no more than telling them to piss off or being drawn into some sort of dialogue stream that allows the party to mend the situation (ideally, there SHOULD be something to allow the party to remedy things, make amends or whatever: either through

attaining a FACTIONAL rep that improves that NPCs personal one, or through finishing a quest).

See? Common sense, and not too much work: a li'l cut-n-paste, a few additional lines and some scripting so that the dialogue tree available reflects the rep (starting with an individualised greeting, of course!) Really, its not that different to how things are handled now - if u do quests for people, there are more options next time u talk to them. What I am suggesting is this be expanded to include simple things to nudge the rep in one direction or another.

For instance, if the player is regularly haughty and contemptuous to the guards, (roleplaying an evil sod to be sure) the rep will slowly nudge over toward hate, such that the players might now need to do something big like a quest just to get the guards to treat them normal, and whatever bonus stuff was available from doing the quest might now have to be earned by getting a rep of 'guard's buddy' or something. Of course, from the evil perspective, snivelly characters who respect strength and intimidation will get a positive reaction toward the characters if they are frorceful, as might straightforward NPCs, and this will also be reflected: smarmy paladins who try to shmooze everyone will pay the penalty.

Anyways, thats the idea: a way of improving roleplaying by giving more consequences for actions. Lets see how the scripts actually work.

For the big reps, we've just seen that you fiddle the **gamerep.mes**, to match the reputations asigned in gamereplog.mes. ToEE tends to work in 10s and 20s: for instance, some generic positions look like this:

```
def make_hate( attachee, triggerer ):
________if ( attachee.reaction_get( triggerer ) >= 20 ):
____________________attachee.reaction_set( triggerer, 20 )
________return SKIP_DEFAULT
```

```
def make_worry( attachee, triggerer ):
_____if ( attachee.reaction_get( triggerer ) >= 40 ):
_____________attachee.reaction_set( triggerer, 40 )
_____return SKIP_DEFAULT
```

10s and 20s are fine by me. Of course, certain acts or tasks can still have a massive influence of say +/- 50 - if u save someone's life, all past indiscretions might be forgiven, while if u butcher their kids in front of them, no amount of politeness will redeem you. But thats a case by case thing obviously. The 'butcher of Hommlet' rep is an obvious example, that gives a -89 beating to anyone of faction 9 (ordinary Hommletians).

At this point I should make 2 things clear:

NPCs start at reaction level 50, so when we talk about going up 10 or 20 or whatever (better reaction) or going back -10, -20 etc (negative reaction) we are talking about starting from 50. So above, a person who really shits the NPC (the teamster in this case) will go straight to 20, and his generic greetings will be contemptous, while a preson who annoys him a bit will go straight down to 40 and get worried 'not sure about you' greetings.

On the up side, an example: if you barter Black Jay's wife's ring back to him, you get a reaction of +15, but if u just give it to him outright and refuse a reward, you get +30.

Below zero, NPCs attack on sight. So -89 for the Butcher of Hommlet rep means the townsfolk, by and large, are all going to have at you on sight. The game DOES include the possibility some folk had such a high opinion of you that they don't attack you even if you have that rep, they will greet you with generic comments such as:

{1}{Your murderous reputation does not intimidate me!}

This reminds us that when it comes to handing out negative reps, you have to be careful: just because ur player is slightly unpleasant doesn't mean folks in a bastion of Order like the Keep

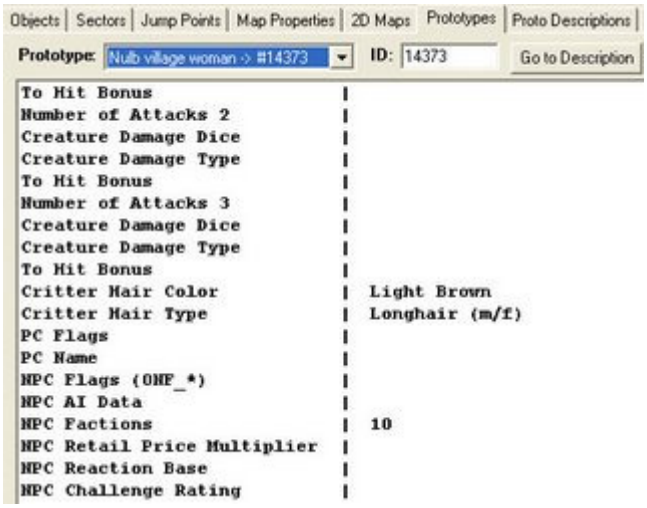
will start trying to kill them (or it shouldn't). So no negative rep things in the standard dialogue tree perhaps, that can be acccessed over and over (or if there are, the NPC might have a thing in their san_dialog script that below a certain point, say 20, they refuse to talk at all and thus the person can't push them below zero. In that case, to talk to them (and say sorry perhaps) the player will either have to raise their entire factional reputation, or fulfill a quest that will then open up a new dialogue branch and of course raise their rep).

Confused? No matter, lets start again and be methodical this time.

Factions:

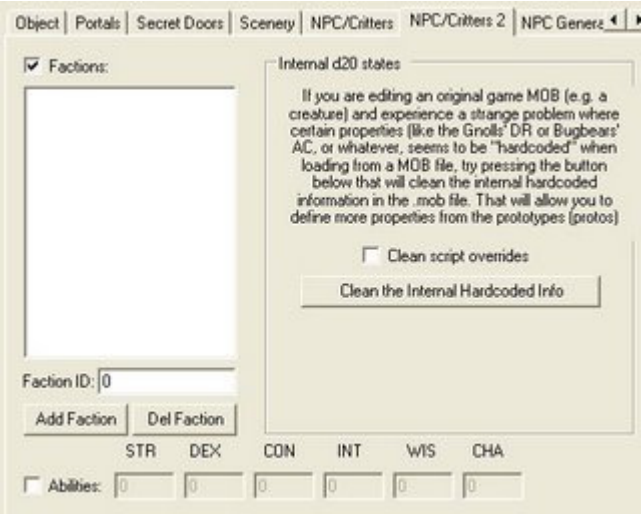
Factions are set in col 154 of **Protos.tab** (by ProtoEd numbering, or just look for **NPC Factions** in ToEEWB). Here are pix of the respective shots.

Line #	Col 151	NPC Flags	Col 153	Col 154	Col 155	Col 156	CR/EXP	Base Refle
14143		ONF_KOS0					3	-2
14144				10				
14145				10				
14146				10				
14147				10				
14148				8				
14149				10129				
14150				10				
14151				1014				
14152				1014				
14153				4				
14154				4				
14155				5				
14156				4				
14157				8				
14158		ONF_KOS		3			2	4



Lets concentrate on the first one, which is from the Protos.tab in patch 2, I think. Note that a critter can have more than one faction. Thus for instance all the folks in the Welcome Wench will have the same faction - 9, Hommletians - but might have their own factions on top of that (such as being guards, or evil Hextorites, or something). They all have the Hommletian faction - despite the fact that Elmo and Turuko, for instance, might have diametrically different loyalties - because if the party attack someone in the bar, noone is going to just stand around - EVERYONE is going to be in on it. And thats in a nut-shell what factions do, they make sure groups act together, so, one in, all in. Attack one member of the earth temple, and everyone in noticing range will jump to their defense: it doesn't have to be scripted.

Factions can be set independantly in .mobs by ToEEWB (not sure this is the latest version, but what the hey):



Since you can have more than one faction, I don't know whether this REPLACES anything set in protos.tab for that character, or adds the factions to it. Thats one for Ag to answer.

To set factions in ToEEWB, you apparently have to add trailing zeros between them (and after them). So, as Krunch put it:

If a prototype has a monster or NPC listed as having a faction of 8 and you want to make a new mob for a map where the monster or the NPC will continue to have a faction of 8, plus you also want the monster or NPC to have an additional faction of 19, in ToEEWB you can add both the 8 and 19 as two factions for the monster or NPC. However, to do this in ToEEWB, you must add four separate entries for the two factions. You must add a trailing zero after each faction. In this example, you would add a faction of 8, add a trailing 0, add a faction of 19, and add a trailing 0. Treat adding a trailing zero as like entering a faction of zero, just remember that it is a trailing zero - not a faction. And, do not forget to save the changes.

four entries in the factions list window in ToEEWB:
8
0
19
0

Make the entries in that order, remembering a trailing zero follows each faction to make a monster or NPC have factions of 8 and 19.

Hmmm, that seems to suggest that .mob factions DO replace and override protos.tab factions - well, there you go. Doesn't surprise me, .mob settings are pretty consistent that way. Anyways, well done Kap'n K for picking that up, and Ag for explaining it.

The factions themselves are listed in **faction.mes** (surprise!) in the **Oemes** folder. Here they are as of Co8-5.0.1:

{0}{An unimportant faction}
{1}{Moathouse brigands}
{2}{Moathouse captives}
{3}{Air temple}
{4}{Earth temple}
{5}{Fire temple}
{6}{Water temple}
{7}{Greater temple}
{8}{Non-affiliated temple}
{9}{Hommlet villager}
{10}{Nulb villager}
{11}{St. Cuthbert member}
{12}{Old Faith believer}
{13}{Brothel Member}
{14}{Pirate}
{15}{Forces of Lareth}
{16}{Men-at-arms captives}
{17}{Farmer captives}
{18}{Elven consortium prisoners}
{19}{Orc prisoners}
{20}{Female prisoners}

{21}{Ashrem's bandits}
{22}{Scorpp's rebels}
{23}{Bugbear deserters}
{24}{Ogre cave dwellers}
{25}{Traders}
{26}{Skole's goons}
{27}{Ikian's adventuring party}
{28}{Caravan survivor - vignette}
{29}{Dead woman - vignette}
{30}{Moathouse gnolls}
{31}{Mayor faction}
{32}{Hextor faction}
{33}{Hrudek faction}
{34}{Rat faction}
{35}{Black Jay and his farm animals}
{36}{Verbobonc citizen}
{37}{Scarlet Brotherhood}
{38}{Hickory Branch}

Now: PCs themselves can be recognised as parts of factions (though it doesn't mean that in combat critters are going to come to your aid - I mean they might, but there is a problem in ToEE with cooperative fighting in that when you kill all the 'baddies', the combat won't end - every critter in combat has to be dealt with, even if they came in on your side or were attacked innocently through friendly fire or area-effect stuff by the enemy, they will still have to be killed / subdued etc to end combat). You may have noticed this in the bit above about reputations - adding a reputation can get the party members recognised as members of this or that faction. Let me repeat it:

// Game reputations start at 1
// Each reputation starts with 3 faction numbers, indicating the three
// factions that will treat the reputation owner as being part of their

// faction. One or all of these faction numbers can be zero, which is
// the empty faction.

That is (to spell it out again), when you get a reputation, it can cause up to 3 factions to recognise you as a member of their own faction (plus give reaction adjustments, positive or negative to the same or other factions). Here's where the fun begins.

Reputations:

Reputations are found in **gamereplog.mes** in the **mes** folder and are adjusted according to **gamerep.mes** in the **rules** folder (I think I said that above). Lets have a look at them for ToEE:

{1} {Butcher of Hommlet}
{2} {St. Cuthbert Disciple}
{3} {Old Faith Donor}
{4} {Master of the Crude Insult}
{5} {Beggar Maker}
{6} {Tatterdemalion Tutelary}
{7} {Defiler of Women}
{8} {Master Ravager}
{9} {Purveyor of Swift Justice}
{10} {Member of the Air Temple}
{11} {Member of the Earth Temple}
{12} {Member of the Water Temple}
{13} {Member of the Fire Temple}
{14} {Champion Brawler}
{15} {Moathouse Cleaner}
{16} {Prison Liberator}
{17} {Liberator Extraordinaire}
{18} {Friend of Lareth the Beautiful}
{19} {Grud Buddy}

{20} {Victor of the Drinking Contest}
{21} {Assassin of Lodriss}
{22} {Knight of Furyondy and Veluna}
{23} {Target of Revenge}
{24} {Initiate of Assassin's group}
{25} {Assassin}
{26} {Dragonslayer}

I'll spare us all the effects and descriptions. Lets look at the first one in detail, 'Butcher of Hommlet'. Gee, I wonder what that means? ;-)

The effect in **gamerep.mes** for number one looks like this:

{1}{0 0 0, -89 9} //Butcher of Hommlet

The 3 zeroes are the 3 factions that this rep COULD make you a member of, but it doesn't make u a member of any, so they are zeroes (obviously). The -89 to faction 9 (Hommletians) as i mentioned above means just about any Hommletian you encounter is going to want to kill you on sight with this rep.

Heres an interesting one:

{22}{0 0 0, +40 9} //Knight of Furyondy and Veluna

Get knighted by Thrommel and the average person will have a +40 reaction (ie from 50 to 90) so getting the 'Butcher of Hommlet' rep SHOULD reduce your reaction to 1 (if you haven't done anything else to piss them off, which is pretty rare in the game - Black Jay maybe, ummm Sunom the weaver if u make eyes at Monier, a few others). So the Hommletians should hate your guts and express it in initial greetings BUT not attack you 8^D Must test that one day. Lets face it, if the titled gentry want to murder a few peasants, the peasants have to damn well cop it on the chin - thats how fuedalism works!

Lets look at a slightly different one, 'Member of the Air Temple' (# 10) - like anyone would seriously throw in their lot with that loser Kelno (or whatever his name is).

{10}{3 0 0, +30 3 -20 4 -20 5 -20 6} //Member of the Air Temple

So now members of faction 3 (the air temple) regard you as one of them, and get a +30 reaction to you - they love you as much as Black Jay does if u return his wife's ring to him and ask for no reward. BUT the members of the earth, water and fire temples all take a -20 reaction to you.

Now... something which I just tested: reputations continue to effect your NPCs even in the party. So for instance I tried consoling in the 'Knight of Furyondy etc' rep, and sure enough, Elmo and Fruella's opinion of me rocketed up to 90 (Fruella's was 50 - go figure). Then I murdered the woodcutter and...

...nothing happened - his wife didn't even join in the fight. Weird. Obviously someone forgot to tag him faction 9!

But when I murdered the wife, sure enough, I was the butcher of Hommlet, and Fruella's opinion of me was a measly 1 (thats more like it!) I then went to Filliken's house and chatted to his family, they gave the 'I don't talk to the likes of you' spiel at the opening line, then went into the standard dialogue tree (Meleney was so offended by my murderous ways she flirted with me. [Yawn]). Here's a pic of the console output.

The 50's mentioned there are interesting: those are reactions to me joining all 4 of the temples. So, that tells us that even though the party may be recognised as being a member of this or that faction, the individual NPCs in your party do NOT appear to have the faction numbers attached to them (Elmo and Fruella did not get either the +30 factional reaction bonus for joining a temple, nor the -20 beaty for joining a different temple). I had a theory that it was these reaction penalties that were causing NPCs to go postal when they leave the party (join all 4 temples and you will take at least 3x20 points reaction penalty, so that would push an NPC on 50 down below zero and explain everything) but alas, the testing does not bear this out. O well :-)

What about removing reputations? I mean you can get one as a Knight of Furyondy, but what if you decide to throw in your lot with the Greater Temple, butcher all of Hommlet and call the Viscount of Verbobonc an inbred pillow-biter? Some cunning modder might decide you should lose the rep, what happens then?

Well people who keep up with the *Common Issues and Solutions* thread will remember Blue posted a console fix to get rid of the Butcher of Hommlet rep if u have got it inadvertantly. So yes, remove the rep and u remove the reaction number. Why is this so? It says a lot about how the game works: it doesn't have some set number somewhere for each character that gets moved up and down accordingly, but constantly calculates and recalculates on the fly (Drifter actually proved this some time back regarding the Constitution bonus issue). So remove the

reactionary element, and the game doesn't think, "ooo I better knock that bonus 40 off the reactions then" (that would be cunning indeed!) but rather the next time it recalculates, it will come up with a new number. Here is a pic of it in action, where I added the Knight rep, checked Elmo's reaction (game.party[5]) to my PCs (game.party[0]) - it was 90, 50 start +40 for the rep of course - then removed the rep and checked again, back to 50.

Before we get into individual reactions, one more thing: certain reputations will AUTOMATICALLY cause a paladin to fall. They are as follows:

{1} {Butcher of Hommlet}
{4} {Master of the Crude Insult}
{6} {Tatterdemalion Tutelary}
{7} {Defiler of Women}
{8} {Master Ravager}
{10} {Member of the Air Temple}
{11} {Member of the Earth Temple}
{12} {Member of the Water Temple}
{13} {Member of the Fire Temple}
{18} {Friend of Lareth the Beautiful}
{20} {Victor of the Drinking Contest}
{21} {Assassin of Lodriss}

No surprises in any of that (well maybe #4 instead of #5 - I know which I regard as the greater sin - and we won't go into the drinking one here). The other assassin ones (25 & 26) being later Co8 additions don't have this automatic element, but you have to be an evil group to initiate them I think. (Maybe thats why Tarim has no faction, so he can be silently assassinated without u getting the 'Butcher of Hommlet' rep. Is he one of the intended victims? I have never played that scenario).

Anyways, these anti-paladinial reps seem to be hard-coded - certainly there is nothing in **gamerep.mes** to cause a Paladin to fall, and indeed I tested these in KotB (where I have a party with a Paladin) despite the fact that I have only done the reps up to number 13 there (they just showed up blank in the Logbook), but STILL my Paladin fell. So now I am going to have to go back and renumber some of the KotB ones, lest joining the Merchant's Guild makes your Paladin fall (theres a moral argument - root of all evil, and all that ;-): needless to say, doing quests for the monsters in the Caves of Chaos etc will cause a Paladin to fall instantly, as will joining the Thieves' Guild, getting Hextor's blessing etc. Are there any other instant, automatic results of getting this or that rep? Not that I have noticed. And no, being knighted does not raise a fallen Paladin (nor should it).

Aside: interesting, and rather unfortunate, the amount of stuff Troika put in the engine rather than in mes files etc where we can hack it easily. Why would specific reputations be hard-coded to cause a Paladin to fall, rather than it being one more triggered effect in gamerep.mes? It says to me (along with the many other examples like it), there was NEVER any intent to do a follow-up game to ToEE.

Anyways, I just realised, I have not actually spelled out the scripts for achieving all this! They are as follows:

triggerer.reputation_add(#)
game.party[0].reputation_has(#)
pc.reputation_remove(#)

obj.reputation_has[#] == 1

where I have mixed up the objects for various moments: note when using the console u have to use something like game.party[0], because it won't recognise triggerer etc.

I haven't tested **obj.reputation_has[#] == 1** but can assure you **pc.reputation_has(#)** works as a conditional in a dlg file, its already in KotB.

Note that Phalzyr lists the following also:

free_rep(npc, pc)
make_hate(npc, pc)
make_like(npc, pc)
make_worry(npc, pc)

He calls these 'procedures' requiring 'additional scripting'. What they are, is just names the game uses for little scripts to achieve these things. As we saw above, 'make_hate' is simply a little script to set the NPCs reaction straight to 20 if it is equal to or above it, and likewise 'make_worry' and 'make_like' will set to specific numbers if appropriate. Calling these by themselves, without adding the additional scripting, won't do anything: but now you have seen the actual functions u can write your own little scripts and call them anything u like :-)

So, finally we can move on to the important part of this tutorial, handling individual reactions.

Reactions:

Well we saw the functions for individual reactions in 'make_hate' and 'make_worry', lets have them again:

attachee.reaction_get(triggerer) >= 50
attachee.reaction_set(triggerer, #)

attachee.reaction_adj(triggerer, +#)

where reaction_set creates an absolute number and reaction_adj adjusts the existing number by the specified value (which can be positive or negative).

So u can create reactions for individual NPCs within factions, as well as the reputations for the whole faction. Damn this is a powerful tool! I love it, and think it is a pity it is so under-utilised in ToEE. Being able to so simply and easily adjust a large number of characters' reactions to u via reputations, but still maintain individual control of each NPC via the reactions, what more could you want for role-playing? Anyways, lets look at it in action. Here's a snippet from the current thing I am working on, busting a guy out of the cells in the Keep fortress. In the last update it was bare-bones but since getting back from Malaysia I have fleshed a lot of it out. Lets have a look:

{60}{[The rogue smiles ambiguously.] I think I will try my chances in the cell. Now get the hell out of my face before I call the guard myself.}{[The rogue smiles ambiguously.] I think I will try my chances in the cell. Now get the hell out of my face before I call the guard myself.}{}{}{}{npc.reaction_adj(pc,-20)}
{61}{Fine, stay here and rot.}{}{1}{}{0}{}
{62}{Big mistake.}{}{1}{}{0}{}
{63}{Whatever.}{}{1}{}{0}{}

This is when you handle the whole thing so badly the dude thinks it is a trap and elects to stay in his cell rather than get rescued by you. You take a reaction beaty.

{10}{[He brightens fractionally, but is still wary.] Who sent you?}{[He brightens fractionally, but is still wary.] Who sent you?}{}{}{}{}
{11}{Do you want to get out or not?}{}{1}{}{60}{}
{12}{Milson, the head of the Guild.}{}{1}{}{40}{npc.reaction_adj(pc,-10)}
{13}{A mutual friend, I won't mention his name here.}{}{10}{}{16}{npc.reaction_adj(pc,10)}

If you are silly enough to blurt out sensitive information when there may well be listening devices around (you are breaking this guy out precisely to stop him from being questioned), you take a small reaction penalty. If you show some tact, you get a small reaction bonus.

There are a few other mitigating factors - pulling off the rescue will earn a reputation that will make his reaction go up, stopping to pester him with questions in the middle of the rescue will make his reaction go down, etc. The payoff? When it is all over, and you ask him what he was in for:

{400}{Good to see you again, brother.}{Good to see you again, sister.}{}{}{}{}
{401}{You too. Keeping your nose clean?}{}{1}{}{410}{}
{402}{I wanted to ask you, how did you end up getting arrested anyway?}{}{8}{npc.reaction_get(pc) >= 50}{420}{}
{403}{I wanted to ask you, how did you end up getting arrested anyway?}{}{8}{npc.reaction_get(pc) < 50}{430}{}

If you have made a small *faux pas* along the way, the reputation will erase it and raise his reaction above 50 anyways. But if u have pissed him off a number of times, well, not surprisingly, he does not feel like sharing rather confidential information with you.

This is my first effort at using reactions so it is fairly straight-forward: be nice and tactful and u get a good reaction, be nasty or stupid and you get a bad reaction. Of course there is more to dealing with folks than that: for instance, both the Bailiff and the Blacksmith have little asides in them (to be scripted in later) that if u try to be overly nice to them, they take it as smarmy and you get a negative reaction rather than a positive one - they react well to bold and forceful people who are not time-wasters. People will have to really role-play their characters in KotB (hopefully - well that is the ideal): and of course, sometimes you are simply going to piss people off. Thats life.

One thing: I have NOT been able to get negative reaction adjustments to work when the NPC is IN the party, though positive ones do: that is, THIS works:

npc.reaction_adj(pc,10)

THIS doesn't:

npc.reaction_adj(pc,-10)

when the NPC is in the party. This is doubtless due to the different way the game handles NPCs in the party to normally (not firing certain scripts etc). For instance, you can piss an NPC off to a negative reaction (below zero) in the party but they won't attack you, because they are under your control - once they leave the party, though, they go for your throat.

How can u piss them off if the reaction thing doesn't do negative numbers while they are in the party? By reputation: a reputation that bestows a negative reaction on that NPC follower's faction will still have a negative effect on that NPCs reaction, its just the individual reactions that don't seem to go backwards.

Another example of this sort of scripting not working in the party is trying to set the reaction manually using **npc.reaction_set(pc, #)**: this likewise doesn't work. For instance I tried the following script as a workaround:

```
def neg_rep( attachee, triggerer ):
________a = attachee.reaction_get( triggerer )
________if a >= 11:
____________________a = a - 10
____________________attachee.reaction_set( triggerer, a )
________return SKIP_DEFAULT
```

This works fine if the NPC is NOT in the party, but not when he has joined up. To get him to have a negative reaction to you stopping for chit-chat in the middle of the escape, I had to use the horribly crude method of booting him for the duration of the exercise.

```
def neg_rep( attachee, triggerer ):
________triggerer.follower_remove(attachee)
________a = attachee.reaction_get( triggerer )
________if a >= 11:
____________________a = a - 10
____________________attachee.reaction_set( triggerer, a )
________triggerer.follower_add(attachee)
________return SKIP_DEFAULT
```

Ewww, I feel dirty just thinking about it! But it works: it will take it down by 10 every time you are crass enough to interrupt the escape for idle chit-chat, but won't allow it to go below zero - he is not going to enter mortal combat over this, he has his eyes on the goal even if you don't! Also, it is pretty damn seamless in the game, unless you have moved him up the order.

Come to think of it, I coulda booted him and just used **npc.reaction_adj(pc, -10)**.

Anyways, how many people actually talk to their characters while they are in the party? Lets face it, other than to deal with equipment issues or make them leave, it is pretty rare (hint: talk to Ronald! He has the second-biggest dlg file in the game and MOST of it is in-party banter! Endarire really put in a huge effort there).

Well thats it! Use this wisely people, we don't need folks attacking you every 5 minutes just because u said something silly. Its not how lawful societies function. And if a major plot point becomes hidden because the player pisses of an important NPC, remember to add some way of repairing it!

Animations (Part 1)

This starts as a tutorial, but descends into a series of observations and suggestions on that most fun of subjects, animation!

Not that there will be any cartoons or anything here. I simply want to catalogue a few of the animations that can be accessed in ToEE, as usual drawing on the ones I have done myself so u can see things that actually work and not just theoreticals (though we will have some of them too).

Animations are, imho, a very important part of fantasy and RPG games (I wrote 'essential' but crossed it out because Ultima 2 didn't have them and still delivered the experience, and likewise Ultima 3 also didn't have any to speak of other than monters moving their arms and legs up and down in unison and looking very fake - but again, it delivered the goods in bucketloads. Damn that game had atmosphere!)

In ToEE, we have the agony and the ecstasy. On the one hand we have quite a glorious little engine that animates everything from the attacks and running around and all that usual stuff, some gorgeous spell effects, drinking potions, picking locks, people tripping and slumping over when bezapped (even Balor's can be tripped - well, the model can) and some elegant little touches, from fireflies and cloaks swirling as you run past due to the breeze created, to spiders twitching their legs some seconds after u kill them. Often these involve many layers of particles, 3d models and 2d sprites and a lot of work on the part of the developers across various fields. PCs cleaning their weapons or practising their 'crane technique', blacksmiths a-smithing, ogres getting bored, zombies eating, goblins dancing, evil clerics worshipping, there's a lot in there.

And yet... we also have the nonsense. Characters standing there saying "I am lying in bed, u know". No sitting. No lying down (unless smacked down). And no easy way to access much of them.

Sooo... the average encounter in Hommlet involves various characters **just standing there**. Not plying their trade, not sitting or sleeping in the middle of the night, not doing a damn thing. (And with their weapons in hand, [sigh]). If someone like Jayfie gets arrested, u don't see it happen, u just get told about it: many of the characters didn't even face u when u speak to them until the modders fixed it (probably there are some who still don't). Even the small

number of secret doors in Hommlet proper got removed, perhaps because they made the place too exciting. It really tests the suspension of disbelief - indeed, I think it is a fair comment to say that u never really get the feeling u r in a living town, its just a place where stuff happens if u instigate said stuff, and NOTHING happens if u don't.

Indeed other than the blacksmith who hammers away (all night, alas...) the only people who actually seem to be DOING anything in Hommlet are the Badgers. They SHOULD have their weapons there, and they make their patrols. Hallelujah!

So, todays first lesson: waypointing (I don't think I have covered this before).

WAYPOINTING

Waypointing involves getting a character to move around a series of waypoints (surprise!). Its how the Badgers do their patrols - they have waypoints to follow, and they follow them. Easy.

Waypoints can be introduced in TOEEWB for mobs in the **NPC** / **Critters** tab. Here's a pic of the Corporal's from KotB in a shot from the upcoming 'day in the life of the Corporal'.

You hit the Waypoints check-box, then go to town putting in the X & Y values of the waypoints you want the character to follow. There are no offsets, but then defining things that precisely would be no end of tedious - ugh!

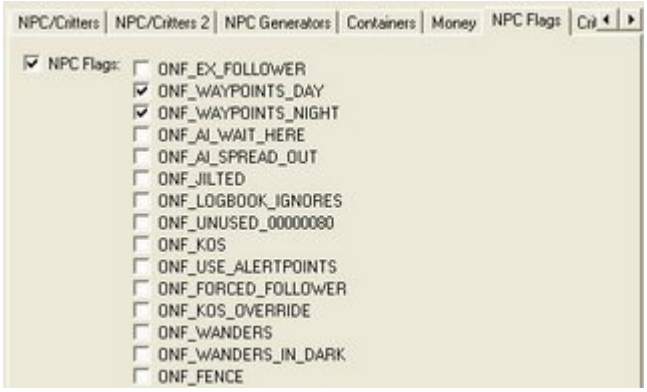
Note the rotation and delay values. Yup, at each waypoint u can have the guy stand there for a predetermined amount of time, facing any given direction, if u so desire. Very handy, and not

just for patrols - other people move around, u know! Just don't forget to hit the check-box for each if u r using them. The delay is measured in milliseconds (1000 will cause a 1 second delay) while the rotation is as per usual (borrowing from the ToEEWB manual again):

5.5 - northwest
5.0 - west
4.0 - southwest
3.0 - south
2.5 - southeast
2.0 - east
1.5 - east
1.0 - northeast
0.5 - northeast

Any downsides? Well, u can't insert a waypoint if u screw up, u can only 'add' or 'delete'. So you would have to delete back to where you screwed up and then add the ones that come after it. (Of course u can put in the fixed one, click on the good ones that came after it to bring their values up, click 'add' for each, so u have the full set after the screw up, then go back and delete what is in between. Its not too hard).

The other thing to remember - and this is essential - is to activate the ONF_WAYPOINT_DAY or ONF_WAYPOINT_NIGHT flags in the **NPC Flags** tab.



You can use both or either, depending on when u want your NPC to wander around, but if u

don't check either of them, don't think ur guy is going anywhere: he'll stand around just as if u didn't bother setting the waypoints, because u didn't activate them.

A thought: presumably if u want your guy to stand around until a given moment THEN perform his waypoints, you can just wait til the triggering event then activate the flag with

attachee.npc_flag_set(ONF_WAYPOINT_DAY)

Never actually tried that :) I have a character like that (the Corporal, no less) but I made a new mob for him under the circumstances for a variety of reasons.

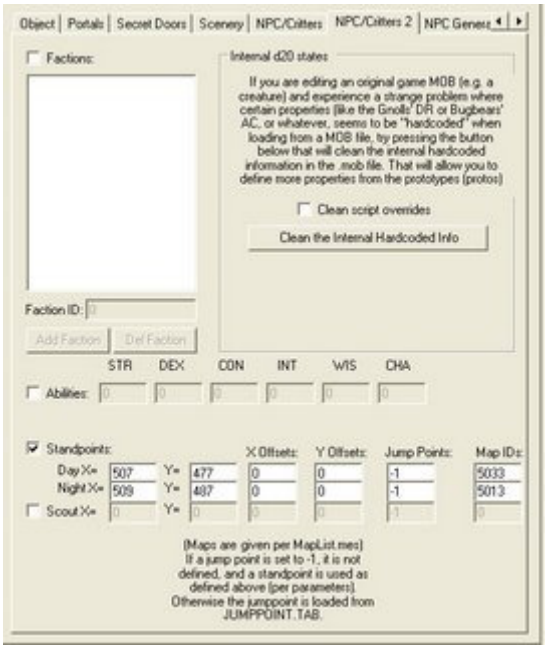
Note that afaik there is no way to get SEPERATE waypoints happening for day and night. Pity, because of course if u r gonna have the day / night transition u r gonna have people doing different things at those times. But at least we can switch one of them off, have the NPC wander around his fields by day, say, and stand in his bedroom at night.

This leads to the question, how do we move them around? I'm glad u asked.

STANDPOINTS

Click on the **NPC** / **Critters 2** tab to access the standpoints. Click on the checkbox. Unless u really want to, I see no reason not to leave the jumppoint thing at -1 and just write the map number straight in, rather than writing it in another file then calling to it. (Jumppoint.tab is great for many things but this aint one of them).

Using the standpoints, u can set your locations where u want the NPC to stand by day and by night, complete with offsets. If they are flagged to follow waypoints, they will do so. Great, init? Have a pic.



There is only one drawback (a biggy) - if u r using Standpoints across different maps, u have to edit the daynight.nxd file. This ain't so bad. What is bad is, u have to start a new game to get those to take effect (thats why we all had to start afresh to play Verbo-Mod when it first came out, afaik.)

If u r using Standpoints across the same map, u don't have to do that, just write them in here in the mob and thats all there is to it. Easy. There is a reasonable amount of scope in that; for instance, in KotB the guards stand at different points by day and by night (so people carrying out nefarious activities will have a window of oppportunity for their plans - and one of getting caught!) while I just recently put in a builder who roams the building site in the NW corner by day, then stands by the nearby fire by night. Likewise the castle builders in TOEE are at the building site by day and the labourers' camp by night (where they wander around, if I am not mistaken). Or Nera Melubb, who from memory stands in different parts of his shop by day and by night: u can have a shopkeeper stand in the shopfront by day and in his private quarters at the back after sundown.

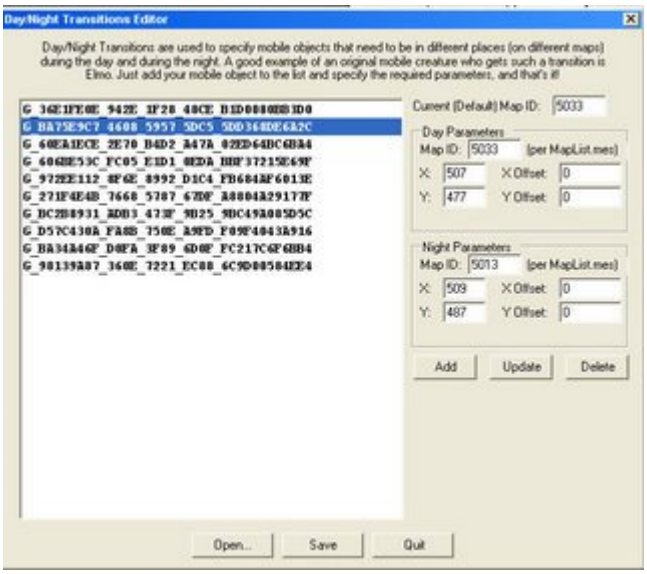
This is important, because moving a merchant across maps BREAKS THE LINK TO HIS INVENTORY BOX (Blue discovered this iirc and I have certainly had it happen to me). Otoh simply moving him around on the same map doesn't cause this (u might have to keep him in

the same sector but properly used a sector is a decent size of real estate - I make no claims a shopkeep can access his box from one end of Hommlet to the other). If you do want to move merchants, you will have to have seperate mobs, with a heartbeat triggered to switch them on and off according to time of day (OF_OFF of course) - again something Blue did (or similar to what he did) to allow u to take Otis, Burne etc adventuring with u, just switched off the one attached to the box and created a new one to adventure with, then visa versa when u brought him home. Previously, signing him up then leaving the map broke the link, and that was that.

Now, lets have a look at daynight.nxd in action. If u have never used it before it lives in your **module/.../rules** folder (**module/Co8 5.0/rules** or **modules/KotB/rules** or whatever). Make a copy and drop it straight in your ToEEWB root folder, next to protos.tab and Invensource.mes and all the other funky files in there.

To access it, click **Tools** / **Day/Night Transitions Editor**, then 'open' one (o look, didn't have to copy it into the ToEEWB folder at all, my bad). Well at least u made a copy of it that way - shame on you for trying to edit an un-backed-up file! Have I taught u nothing?

Anyways, here's a pic of the bugger in action.



There's our Corporal again! By day he stands in his office in the Gate Tower (map 5033), working away at his desk (too stupid to sit in the chair) while by night he is in the pub (map

5013) leering at Two Swords and drowning his sorrows. Again we have our X & Y locations, offsets, and the ability to update things this time - woohooo!!! But then they don't have to be in any order either (afaik).

What happens if u put one set of Standpoints in the mob and different ones in the daynight.nxd? Which wins? I have no idea, why would u try? Sheesh! Althought it might get a little 2-waypoint action going between them I guess (handy for the off-waypoint time of day), but frankly I have never had the need for it so I don't care atm.

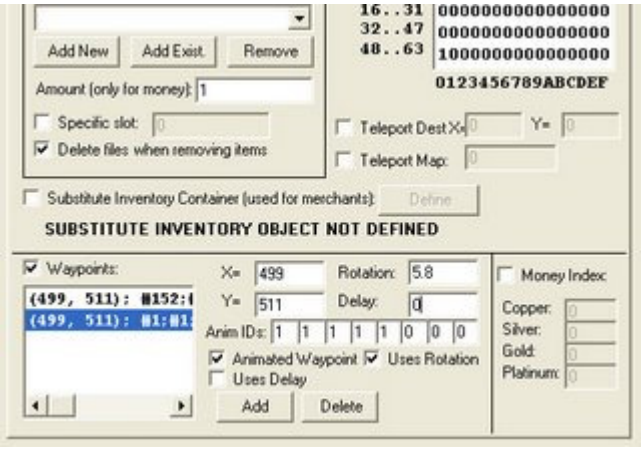
The main thing to notice about using daynight.nxd is the concept of the default map ID, the first box on the right. *This is the map where the mob is stored!* So in this case, the Corporal's mobs are stored in **Map50-Keep-GateTower-first**, not **Map50-Keep-int07-pub**: and no, u don't need 2 sets of mobs, one in each map (the whole point of this is to save that happening. Anyways, 2 mobs would be treated as 2 different objects, for the purposes of remembering if the NPC had met u, if u had attacked him or something).

Pretty straight-forward really. Just remember to save changes and to copy the file back into the game folders: and start a new game!

So, onto the juicy bit: animations. What if we want the guy to do more than just walk up and down?

Go back to the **NPC** / **Critters** tab and down to the Waypoints. You have to set waypoints to do the animations (ones from the mob, anyways) but if u just want the NPC to stand there and carry out his animations, well of course u only do one waypoint, the same as his standpoint, and he will do it in place.

Lets start with an easy one: the Blacksmith animation. Here is the blacksmith from KotB.



Now, keep in mind his model is 504 - it is NOT standard model 100 (human male). So, animation number 1 is him whacking away with his hammer. He does it 9 times - thats his models special thingy, you can see that in

ToEE1.dat\art\meshes\NPCs\Brother_Smith\NPC_Brother_Smith.txt

if u desperately need to (this folder only has him and Elmo (for the drunk stagger) in it, so that'll tell u how many peculiar animations there are for NPCs - not many!!) Here's what it says, though it is irrelevant for our purposes.

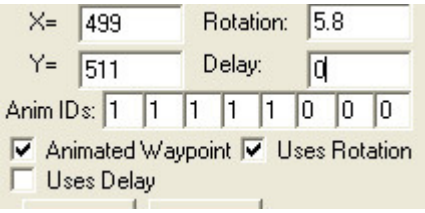
animation "sword_unarmed_special1" "sword_unarmed_special1.mae"

{
event 22 "script" "game.sound_local_obj (4007, anim_obj)"
event 72 "script" "game.sound_local_obj (4007, anim_obj)"
event 122 "script" "game.sound_local_obj (4007, anim_obj)"
event 172 "script" "game.sound_local_obj (4007, anim_obj)"
event 222 "script" "game.sound_local_obj (4007, anim_obj)"
event 272 "script" "game.sound_local_obj (4007, anim_obj)"
event 322 "script" "game.sound_local_obj (4007, anim_obj)"
event 372 "script" "game.sound_local_obj (4007, anim_obj)"
event 422 "script" "game.sound_local_obj (4007, anim_obj)"

}

All this tells us is that if he has a swinging weapon ('sword') on one hand and is unarmed in the other, he can perform his special move (hammering away) and get his nice little special move sound to boot (hammer-on-anvil, which I think is too tinny but Blue tells me is accurate: like I've never worked on an anvil before. Sheesh!)

Enough looking into the innards of the game - not a lot we can do about them atm til we have a model editor. Suffice it to say, thats where the 9 animations come from: so at the bottom, when we see the row of 1's and zeros:



each 1 will cause 9 of these little hammerings to occur, so all up if u stand there and stare at Brother Smith (and I copied him exactly for my blacksmiths, because it was my first go and I wanted to get it right) he will hammer 45 times in groups of 9, then have a pause.

Now... the zero at the end. In this case there is 3, but there only has to be one. BUT... there HAS to be one. You have to end this little row of 8's with a zero, or the game will CTD (so I found in testing, several times in a row).

So thats the Blacksmith's special animation solved. But what about normal models (100 and 101), what can we get them to do? Well, if we look at the NPC_Brother_Smith.txt, the non-special stuff (ie the regular stuff he can do) we see all sorts of things, and more to the point, we see how dependant they are on what he has in his hand (not surprisingly). Lets look at some of them:

animation "unarmed_unarmed_rattack" "unarmed_unarmed_rattack.mae"
animation "unarmed_unarmed_lattack" "unarmed_unarmed_lattack.mae"
animation "unarmed_unarmed_fhit" "unarmed_unarmed_fhit.mae"

animation "unarmed_unarmed_lhit" "unarmed_unarmed_lhit.mae"
animation "unarmed_unarmed_rhit" "unarmed_unarmed_rhit.mae"
animation "unarmed_unarmed_bhit" "unarmed_unarmed_bhit.mae"
animation "unarmed_unarmed_rcriticalswing" "unarmed_unarmed_rattack.mae"
animation "unarmed_unarmed_lcriticalswing" "unarmed_unarmed_lattack.mae"
animation "unarmed_unarmed_fidget" "unarmed_unarmed_fidget.mae"
animation "unarmed_unarmed_fidget2" "unarmed_unarmed_fidget.mae"
animation "unarmed_unarmed_fidget3" "unarmed_unarmed_fidget3.mae"
animation "unarmed_unarmed_sneak" "unarmed_unarmed_sneak.mae"
animation "unarmed_unarmed_panic" "unarmed_unarmed_panic.mae"
animation "unarmed_unarmed_combatfidget" "unarmed_unarmed_combatfidget.mae"
animation "unarmed_unarmed_special1" "unarmed_unarmed_special1.mae"
animation "unarmed_unarmed_special2" "unarmed_unarmed_special2.mae"
animation "unarmed_unarmed_special3" "unarmed_unarmed_special3.mae"
animation "unarmed_unarmed_fdodge" "unarmed_unarmed_fdodge.mae"
animation "unarmed_unarmed_rdodge" "unarmed_unarmed_rdodge.mae"
animation "unarmed_unarmed_ldodge" "unarmed_unarmed_ldodge.mae"
animation "unarmed_unarmed_bdodge" "unarmed_unarmed_bdodge.mae"
animation "unarmed_unarmed_rthrow" "rthrow.mae"
animation "unarmed_unarmed_lthrow" "lthrow.mae"
animation "unarmed_unarmed_lsnatch" "unarmed_unarmed_lsnatch.mae"
animation "unarmed_unarmed_rsnatch" "unarmed_unarmed_rsnatch.mae"
animation "unarmed_dagger_rturn" "unarmed_unarmed_rturn.mae"
animation "unarmed_dagger_fhit" "sword_sword_fhit.mae"
animation "unarmed_dagger_lhit" "sword_sword_lhit.mae"
animation "unarmed_dagger_rhit" "sword_sword_rhit.mae"
animation "unarmed_dagger_bhit" "sword_sword_bhit.mae"
animation "unarmed_dagger_rcriticalswing" "dagger_dagger_lattack.mae"
animation "unarmed_dagger_lcriticalswing" "dagger_dagger_lattack.mae"
animation "unarmed_dagger_fidget" "Dagger_dagger_fidget.mae"
animation "unarmed_dagger_fidget2" "unarmed_dagger_fidget2.mae"

animation "unarmed_dagger_combatfidget" "sword_sword_combatfidget.mae"
animation "unarmed_dagger_special1" "unarmed_dagger_special1.mae"
animation "unarmed_dagger_special2" "unarmed_dagger_special2.mae"
animation "unarmed_dagger_special3" "unarmed_dagger_special3.mae"
animation "unarmed_dagger_fdodge" "sword_sword_fdodge.mae"
animation "unarmed_dagger_rdodge" "sword_sword_rdodge.mae"
animation "unarmed_dagger_ldodge" "sword_sword_ldodge.mae"
animation "unarmed_dagger_bdodge" "sword_sword_bdodge.mae"
animation "unarmed_dagger_rthrow" "rthrow.mae"
animation "greatsword_greatsword_combatfidget" "greatsword_greatsword_combatidle.mae"
animation "greatsword_greatsword_special1" "greatsword_greatsword_special1.mae"
animation "greatsword_greatsword_special2" "greatsword_greatsword_special2.mae"
animation "greatsword_greatsword_special3" "greatsword_greatsword_special3.mae"
animation "greatsword_greatsword_fdodge" "greatsword_greatsword_fdodge.mae"
animation "greatsword_greatsword_rdodge" "greatsword_greatsword_rdodge.mae"
animation "greatsword_greatsword_ldodge" "greatsword_greatsword_ldodge.mae"
animation "greatsword_greatsword_bdodge" "greatsword_greatsword_bdodge.mae"
animation "greatsword_greatsword_rthrow" "rthrow.mae"

Throw that greatsword! But anyways, that is a SMALL selection of the various animations there are: subtly different (or possibly, frequently identical) for each variant of weapon and weapon combo (small, medium, large, shield or unarmed in each hand, plus bows, spears etc). As well as fidgeting, sneaking, critical hits, dodging blows, and all the other stuff models have to be able to do.

So, can we access all these and mod them in to occur on command? I really have no idea, this tutorial isn't anywhere near that complicated! But I can tell you this - yes, we can get the basic combat actions to occur on command: swing, parry, dodge, shield bash (wish we had the matching feat!) duck, sway, they can all be done.

Lets have a look at the Corporal: he can be our test model 100. I ran him thru with numbers 1-

12 and this is what we got:

Hard to picture animations just by looking at still shots, but suffice it to say we have:

The worship one leading off (thats anim 1, so to get it u put a 1 in the box like the Blacksmith's hammering was anim 1)

Attacking and swinging a sword (anim 2)

Parrying (thats the second last one pictured on the right - anim 5 or so for memory)

A bunch of swaying and generally avoiding blows with body movement and shield (anims 3, 4, 6, 7, 8 or so)

A bunch of sheild bashes (the later ones, anims 9f)

The animations for the Corporal (with longsword and shield equipped) went up to 12. Then at 13, he just stopped dead - no more animations, no more waypoints. I guess I broke him.

Good? Bad? I think it is quite handy. You see, as you may have gathered from the waypoints

tutorial, waypoints are on a loop. The guy will walk his waypoints, then at the end go back and walk them again.

BUT what if we don't want him to? What if we are trying to get him to play out a specific series of events and then stop, thats it?

"Ahhh yes, your reward" said the Castellan. He walked slowly around the table and over to his desk, where he picked up a long glowing rod. "Here it is".

There is a number of ways we can do it - well least 3, one of which is destroying the waypointed mob and creating an identical one in its place with no waypoints. (Has to be done on occasion, as with the switching the merchants on and off so as not to break their link to their inventory, but a crude and inelegant way of doing it).

Secondly, we can switch off the Waypoint flag when he arrives: but we have to have a script that fires at that exact moment. If we have a trigger (the glowing rod in this case, which can be transfered from the desk - or better still, from a chest where we might get the nice chest-opening animation as well - into the Castellan's inventory, whereupon it could fire a script) then we can do this fairly straightforwardly - but if we are just getting the NPC to move into position somewhere, then we have to come up with some fiddly way of doing things, almost certainly with a timed script. Not too bad, since we can set a long train of delay waypoints to make sure there is a nice fat window for the timer to go off while the NPC is at the right spot, but anyone who has ever noticed an inactive or frozen Badger standing around doing nothing will know, these animations are not 100% reliable. Likewise anyone who has not had Scather show up will know the timed scripts aren't either.

Of course, there WILL be moments that these are appropriate (especialy if we want to turn the waypoints back on later). But I have found, simply inserting animation 13 (in this case) does the same thing: stops him dead in his tracks, problem solved. Again it is only going to be used in specific situations, but was perfect for what I wanted (getting the Corporal to walk into his bedroom, turn around, then stop there).

So, we've got some combat animations on tap: good 8^) Needless to say the first thing I did with this was set up a little sparring routine between a couple of the Watchmen in their barracks, so when u walk in, instead of seeing guys just standing there waiting for the adventurers to come over and make their life have some meaning, u see guys sparring away quite oblivious to your existance (indeed, they are quite busy and don't WANT to know u, even if u ask!) Quick montage:

O and we have the worship thing: I might stick that on the Curate or one of the other clerics, but its a bit overkill I think. Will probably save it for the Temple of Evil Chaos.

Another one that has come up several times is simply falling down prone. This came up once with the Corporal when I simply ran all his animations one after another - he tripped and fell over halfway through almost as though jumping from one combat move to the next disoriented him! But I wasn't able to replicate this when testing the animations individually. Likewise, when I first made Reece the Cobbler hammer away at his shoes (with a light hammer this time), I forgot to change his model to 504 and using anim 1 for him simply made him fall over, no worshipping: peculiar. It goes to show you really have to test for your specific model and set-up.

However all is not lost, since falling down prone can be accomplished very simply by a script:

Terjon: {[short] This is the Church of St. Cuthbert. You appear to be lost. The door out is down the stairs and through the study.}{[short] This is the Church of St. Cuthbert. You appear to be lost. The door out is down the stairs and through the study.}{}{20}{}{}

PC: {[Kick Terjon square in the codpiece.]}{}{1}{}{0}{npc.condition_add_with_args("prone", 0, 0)}

Really must mod this in one day, at least for Evil parties - have to track down an appropriate sound effect though. Anyway, fire this little beauty and Terjon goes down like he's been, well, kicked square in the codpiece. (I plan on giving the party about 3 seconds to make it to the stairs before he goes KOS). Unfortunately, he gets straight up again: changing the args to big numbers (rather than 0) doesn't stop that, either. Pity.

What else can we script in? Here's an interesting one from Toee_list.txt:

Q_Play_Critical_Hit_Anim

that would be nice to have on command.

PC: Here is the evil artefact from the demon's tomb.

[Drop artefact]

[Turn Castellan to face artefact]

With one swing of his mighty axe, the Castellan smashes the evil relic to oblivion! [Play Critical Hit Anim and Game Particles]

If anyone figures out how to do it, let me know! Suffice to say trying to add it like I did with 'prone' just made the game CTD (don't know I have CTDed the game from a dlg file before, that was a first :-))

But enough of what we can't do: here is one I call the bitch-slap: we'll use a scene straight out of the Green Gryphon for this.

{455}{No comment.}{I'll be here. [He leers and makes a clumsy, drunken attempt to pinch your behind as you turn to leave.]}{}{}{}{}
{456}{[Let him, then wiggle seductively away.]}{}{1}{}{0}{npc.reaction_adj(pc, 5)}

{457}{[Dodge him, then skip giggling away.]}{}{1}{}{0}{npc.reaction_adj(pc, 10)}
{458}{[Slap the grub.]}{}{1}{}{470}{damage(npc, pc)}

where 'damage' is our bitch-slap:

```
def damage(attachee, triggerer):
_________dam = dice_new( '1d2' )
_________attachee.reflex_save_and_damage( attachee, 20, D20_Save_Reduction_Half, D20STD_F_NONE, dam, D20DT_SUBDUAL, D20DAP_UNSPECIFIED, D20A_CAST_SPELL, 0 )
_________return
```

It could probably be done with half the scripting, but meh I say. What it does is cause 1d2 of non-lethal damage (half with reflex save vs dc20) to the victim: this automatically causes the model to:

- stagger as though just hit

- go into a combat crouch

BUT because we have identified the 'aggressor' as the attachee themselves, it won't spark combat, and they will get over the need to stand in a combat crouch soon enough. This shouldn't be overused of course, but it can make for far more interesting Intimidate checks (a scene in Nulb where you can slap Mickey around comes to mind): one second the guy is reeling, the next he looks ready to knock your block off if u don't back down. Good stuff :-)

What else can we script? Well, frankly that is about the end of my bag of tricks! I intend to keep looking - we should be able to force the 'use wand animation' at the very least through creating a dummy object in the recipient's inventory that has the oif_use_wand_anim flag and an insert_item script (so the theory goes) and likewise I am hoping we can force the use_potion

anim as well. Plus there are all the other monster models to catalogue (starting with those dancing gobbos :-)).

Scripting for Proto Values

Well I stayed up very late working on KotB last night, but I very quickly got sidetracked into writing this blog. There's nothing in here that I have discovered myself (though I did pen all the KotB scripts I am using as examples) but rather it is a compendium of things I have had to go look for when scripting for this or that, which I now present so you folks won't have to hunt so hard 8^D

So, lets say you want to recognise an object, or an NPC, not just by their number but according to some setting in the protos.tab. Here's some examples of how.

Firstly, let me say, you can't always get what you want. For instance, picking critters by faction: damn that would be useful. But we can't. When u try to read the faction flag, you get a changing number (certainly NOT what is in col 154 of ProtoEd) and if you try to script for it, it crashes. The reason for this (so I am told) has to do with how the .dll interprets that particular flag, and it doesn't just read it as an integer. Indeed, there is nothing in the .dll to output the contents of the flag (obj_f_npc_faction (363) for memory, it was a while back when I tried this) in a way we can understand, or so I was reliably informed when I pestered some gurus like Ag, S-S and C-Blue to help with this, so we can't just read it and say, "ahhh, yonder critter is faction X, lets manually remove him from the fight". I mean we CAN manually remove critters from a fight, but not based on faction.

Never fear! There are many things we CAN read. Some we already have seen, such as the flags: I wrote a whole tutorial on flags somewhere. Here's a quick recap from KotB:

```
_________y = (attachee.critter_flags_get() & OCF_MECHANICAL)
_________if attachee.map == 5033: ## office
____________________if ((is_daytime() != 1) and (game.global_vars[19] == 1) and (y != 0)):
_____________________________attachee.object_flag_unset(OF_OFF)
```

_____game.leader.begin_dialog(attachee, 600)

We read the specific critter flag by logically ANDing it with the whole flag byte: if the OCF_MECHANICAL bit is set, it returns a value of 1 to y, otherwise it returns a 0. This way I can have 2 different mobs for a single NPC in the same room and access them independantly. Under the right circumstances (nighttime, and the trigger variable [19] having been set) then the mob flagged OCF_MECHANICAL (which I presume is a leftover from Arcanum and means nothing in ToEE, but in any case is simply a handy 'nothing' flag that the game still recognises) will get its OF_OFF flag unset (ie it will switch on instead of OFF) and it will kick in. Of course, there is an accompanying thing so that the mob WITHOUT the OCF_MECHANICAL flag will then switch off.

Easy, and you can do it for any of the flags: object flags (OF), item flags (OIF), NPC flags (ONF) etc.

What about some of them other columns in protos.tab? Well, lets go through in no particular order...

NAME: Ok, this is a simple but important one: the item name. Item name is col 22 of ProtoEd: it can therefore differ from the prototype number. An example of this in action is find_npc_near: lets have a look at the script whereby Otis checks to see if Elmo is nearby:

```
def make_elmo_talk( attachee, triggerer, line):
_________npc = find_npc_near(attachee,8000)
_________if (npc != OBJ_HANDLE_NULL):
____________________triggerer.begin_dialog(npc,line)
____________________npc.turn_towards(attachee)
____________________attachee.turn_towards(npc)
_________else:
____________________triggerer.begin_dialog(attachee,410)
_________return SKIP_DEFAULT
```

We've seen this before so I won't dwell on it: suffice it to say, the game looks for Elmo's NAME (8000) not his prototype number (14013) or even his IDed number (col 23, which in this case is 20008 - the line in Description.mes where the game gets the word "Elmo" so it can call him that after you have met him, instead of "drunk").

So how exactly do you read the name for moments like this? Well we could go thru Utilities.py and deconstruct the "find_npc_near" command that just got called, but I will leave you to do that for yourself (checking out the innards of that file is well worth doing for any modder). For now, I will simply say it is attachee.name == xxxxx. For instance:

```
________if attachee.name == 14696:
____________________etc...
```

Easy :-) And you can see how important it is that everything have a name, which is why Darmagon went through and added a name (based on protoype # if not already set) to everything in the protos.tab - so spells, scripts doing things by name would always find a handle.

MATERIAL: No, not the material thing for meshes, I mean the material an object is made from. Want your Warp Wood spell to only warp wood? Of course you do ;^) Here is another little thing from KotB that shows how to find a wooden item: in this case, it is going by slot, so you get a 2-for-1, how to find an item by material AND by slot :-)

```
def unequip( slot, npc):
________for npc in game.party:
____________________item = npc.item_worn_at(slot)
____________________if item != OBJ_HANDLE_NULL:
________________________________if (item.obj_get_int(obj_f_material) != mat_wood):
____________________________________________holder = game.obj_create(1004, npc.location)
____________________________________________holder.item_get(item)
```

_____npc.item_get(item)
_____holder.destroy()
_____return

Item slot numbers can be found in ToEEWB: I will post them here though for better tutorial bang-for-your-buck.

0 = helmet
1 = necklace
2 = gloves
3 = primary weapon
4 = secondary weapon
5 = armour
6 = primary ring
7 = secondary ring
8 = boots
9 = ammo
10 = cloak
11 = shield
12 = robe
13 = bracers
14 = bardic item
15 = lockpicks

So to check for an item worn at this or that slot, just check:

npc.item_worn_at(slot)

where 'npc' and 'slot' get replaced by the appropriate handles. For instance, lets say you want to add a script so Terjon checks if u actually walk up to him wearing his dad's ultra-masculine

pendant thingy (prototype 6139) and if so, says something about it without waiting for u to raise the issue. The pendant is flagged

OIF_WEAR_RING_SECONDARY OIF_WEAR_RING_PRIMARY
OIF_WEAR_RING_SECONDARY OIF_WEAR_NECKLACE

I have never understood those "flag it multiple times" things, but Protos.tab does it a lot. Anyways, lets interpret this to mean "it can be warn in either ring slot or the necklace slot" because I think that is what it is probably trying to say. So we would check all 3 slots:

```
________item1 = triggerer.item_worn_at(1)
________item2 = triggerer.item_worn_at(6)
________item3 = triggerer.item_worn_at(7)
```

then test by item name (which for some reason is 3004):

```
________if (item1.name == 3004) or (item2.name == 3004) or (item3.name == 3004):
____________________triggerer.begin_dialog( attachee, 990 )
```

And then have some appropriate dialogue at 990. (note, there may be a more elegant way of doing it than checking them individually like that, but thats ok :-)

Getting back to the material flag, we are checking the internal object flags again (we have used some of these before also) and in the case of material it is, to repeat it:

item.obj_get_int(obj_f_material) != mat_wood

where 'item' could be 'attachee' or 'triggerer' or something else depending on the circumstance. Mat_wood could of course be mat_metal, or mat_cloth, or mat_flesh, mat_glass, mat_force etc.

Do we use this to check whether we are talking to a living creature? Well we could (sort of - living or dead creature would be more accurate, but then you can't 'talk' to a dead thing, except in very special exceptions like Clarisse ;) but a better way would be by checking object type. This is the very first column in protos.tab.

OBJ_TYPE:
_____for pc in game.party:
_____if pc.type == obj_t_npc:
_____return 0

This checks to see whether a member of the party is a PC or NPC follower. Again, this could be attachee.type or triggerer.type or whatever. And it could be obj_t_npc, or obj_t_armor like the pendant, or obj_t_scroll or obj_t_generic (all the items in the 12000s) or obj_t_weapon or whatever.

Here is another quick example of obj.type and flags (portal flags this time - they don't show up too often) in action: the Knock spell.

_____if target.obj.type == obj_t_portal:

_____if (target.obj.portal_flags_get() & OPF_LOCKED):
_____target.obj.float_mesfile_line('mes\\spell.mes', 30004)
_____target.obj.portal_flag_unset(OPF_LOCKED)

SIZE: Back to NPCs. Need to fine-tune your specific NPC rather than just by whether it IS one or not? Well, size is good :-) Easy, too:

_____if attachee.get_size < STAT_SIZE_LARGE:

Nice how they go sequentially: you don't have to say '== small or == medium or == diminutive' etc, just '< LARGE'. A few things go that way - quest outcomes ('if game.quests[19].state <

qs_botched') for instance - but it is not always easy to see the progression, so normally I just do it the long way.

Still worrying about whether it is dead or not?

_____if critter_is_unconscious(attachee) != 1:

want him not only conscious but on his feet?

_____and not attachee.d20_query(Q_Prone):

ok, I am getting carried away ;-) those are not prototype settings. Lets try to stay focused.

But what if we are still looking for a more specific type of NPC than just one who is smaller than LARGE? Race is easy enough, I have talked about that before:

pc.stat_level_get(stat_race) == race_Halfling
pc.stat_level_get(stat_race) == race_Human
pc.stat_level_get(stat_race) == race_Elf
pc.stat_level_get(stat_race) == race_Dwarf
pc.stat_level_get(stat_race) == race_Gnome etc

but we also have the CATEGORY value in col 163 (not to be confused with the one in col 30, from category.mes):

_____obj.is_category_type(mc_type_humanoid)

where 'obj' could again be attachee or whatever handle you grab it by. 'mc_type_humanoid' could be mc_type_animal or mc_type_outsider or mc_type_undead or mc_type_giant or whatever.

SUBTYPE, on the other hand is this:

_____obj.is_category_subtype(mc_subtype_fire)

which is not that surprising ;-)

ALIGNMENT you already know:

_____pc.stat_level_get(stat_alignment) == LAWFUL_GOOD // individuals

_____game.party_alignment == NEUTRAL_EVIL or game.party_alignment == CHAOTIC_EVIL or game.party_alignment == CHAOTIC_NEUTRAL or game.party_alignment == LAWFUL_EVIL // parties

but here is a method for checking alignment as a flag (easier in some circumstances):

_____alignment = target_item.obj.critter_get_alignment()
_____if (alignment & ALIGNMENT_EVIL) or ((alignment & ALIGNMENT_NEUTRAL) and not (alignment & ALIGNMENT_GOOD)):
_____etc...

Note it is checking any evil, any good etc. You can also check the Law - Chaos line:

_____alignment = target_item.obj.critter_get_alignment()
_____if (alignment & ALIGNMENT_CHAOTIC) or ((alignment & ALIGNMENT_CHAOTIC) and not (alignment & ALIGNMENT_LAWFUL)):
_____etc...

You know DEITY:

_____pc.stat_level_get(stat_deity) == 1 // Boccob

and ATTRIBUTES:

_____pc.stat_level_get(stat_strength) >= 10

and GENDER:

_____pc.stat_level_get(stat_gender) == gender_male (or gender_female)

and CLASS:

_____pc.stat_level_get(stat_level_cleric) >= 1

but you won't have come across HIT DICE before (not in my tutorials, anyways!):

_____if obj.hit_dice_num <= 10:

What about PORTRAITS? Yep, we can even read and change them (I can't for the life of me think of many moments to do so, but I guess it might be fun to change the portrait of a character to all bloodied and battered if he is yanked out of combat by falling to low HP):

_____attachee.obj_set_int(obj_f_critter_portrait, 6500)

(That one's from 'Flaming Sphere', which I assume was done by Darmagon and is just a masterpiece of scripting). To read a portrait, I would assume it would be:

_____x = attachee.obj_get_int(obj_f_critter_portrait)

but that one is untested.

How about SKILLS? Well you should be able to read them already:

_____pc.skill_level_get(npc, skill_bluff) >= 7

But what about manipulating them? Here is how Darmagon implemented Analyse Dweomer - a +20 on Spellcraft so you would recognise anything going on:

_____spell_obj.item_condition_add_with_args('Skill Circumstance Bonus', skill_spellcraft, 20)

This powerful command can be used to add other bonuses besides circumstance ones, as Darmargon also discovered - but they don't come off in a hurry, so use at your peril!

_____game.party[0].condition_add_with_args('Shield Bonus',8,0) // has the effect of raising the armor class of the left-most party member by 8!

As for checking for FEATS:

_____if (not target.has_feat(feat_evasion)) and (not target.has_feat(feat_improved_evasion)):

where, for the last time, 'target' could be 'attachee' or 'triggerer' or however else you define it.

Well, that just leaves ROTATION, col 31 :-) We'll end this with a little script that gets quite a workout in KotB: it causes the NPC to turn and face the other way, whatever their initial rotation.

```
def away(attachee):
_____x = attachee.rotation
_____x = x+3
_____if x >= 6:
_____________x = x - 5.99
```

_____attachee.rotation = x
_____return

Creating Notes

Since both Zeb and Allyx have asked this in the last week, here's a kwik tutorial on how to create those little notes you see around the place. This is being posted from work (long dull night-shift) so no pictorial aids alas, you have to use your imagination ;-)

There are two type of notes: ones that open an interactive dialogue thing (like Lareth's diary) and ones that create a big floating box, a la the tutorial: ummm the little note Alira has with a map to the Temple comes to mind, or Zaxis' sister Irma's note from down on level 4 (Liv added that, afaik - yes, she discovered this too).

Both type of notes live in the 11000s section of protos.tab. I am not sure thats essential, but they do. Lets see some of them.

{11000}{Book}
{11001}{Note}
{11002}{Note from Smigmal Redhand}
{11003}{Lareth's Diary}
{11004}{Romag's Diary}
{11005}{Burne's Moathouse Map}
{11006}{Alira's Temple Map}
{11007}{Ragged Note to Terjon}
{11008}{Tattered Piece of Paper}
{11050}{Book of Heroes} // Arena Mod
{11097}{Otis' Letter}
{11098}{Letter from Witch}
{11099}{Lila's List}
{11100}{Parchment of Challenge}

{11300}{Cerulean's Note}
{11301}{Ragged Parchment}
{11302}{Ragged Parchment}
{11303}{Drow Note}

There ya go, thats what my Co8 5.0.x description.mes has (may not be up to date). All the usual suspects are there, and if you don't recognise them all, well, at least one of them is an easter egg I personally inserted which to my knowledge has never been found (well, no-one has mentioned it: though I did send a couple of the modders off to check it to make sure it was there and working). Just a tiny thing.

Once we crack open the relevant parts of protos.tab, we discover that these notes have one of two things that make them work: either...

- an number in column 92, or
- a san_use value.

A few have both, all have one or the other.

We'll start with column 92. This refers to the file written_ui.mes in the data/rules folder. Lets have a look at it in its entirety:

{0}{TAG_TUT_LOCKED_DOOR_REMINDER}
{1}{TAG_ASSASSIN_NOTE}
{2}{TAG_BURNE_MAP}
{3}{TAG_ALIRA_MAP}
{4}{TAG_TERJON_NOTE}
{5}{TAG_TATTERED_NOTE}
{6}{TAG_RAGGED_PARCHMENT}
{7}{TAG_ZUG_POEM}
{8}{TAG_EXTRAPLANAR_NOTE}

{9}{TAG_PARCHMENT_OF_CHALLENGE}
{10}{TAG_OTIS_LETTER}

Its pretty obvious how these match up with some of the existing notes, and so no surprise that, say in proto 11097 (Otis' letter) the number in column 92 is 10, or in proto 11007 the number is 4. But where is the text pointed to by these handles?

This, complicatedly enough, is in the help.tab in the data/mes folder: presumably because the 'bring up a floating box to say stuff' is generally the domain of asking for help. Hence when we go to help.tab we find a line called TAG_TERJON_NOTE (which is called by the handle in written_ui.mes, obviously) which in turn has a 'heading' entry saying Ragged Note to Terjon and then some text:

Dear Terjon, Please accept this Copper Starburst Pendant of St. Cuthbert on your promotion to Chief Cleric of Willip. May St. Cuthbert always guide your path, Your Father

The little boxes you will see in the help.tab but which may or may not appear here in the blog, are carriage returns.

Fairly straightforward, really. Make a prototype, assign it a spare number in written_ui.mes, call it something, then add that to help.tab. You open help.tab with ProtoEd, btw, same as protos.tab. Go open it and have a look around :-)

Now, what if we want something to happen, like marking a new location on your map (as Alira's or Burne's) or an interactive text (like Lareth's diary)? Thats where san_use comes in.

The maps just have something to mark a certain are on your map: I'm gonna make up a script now because those files are not in Co8 5 (they have never had to be modded). They would be something like this:

def san_use(attachee, triggerer):

```
_________game.areas[13] = 1
_________return SKIP_DEFAULT
```

Simple behind the scenes thing: note there is no need for a script to fire the text box, because that is what the call in column 92 to written_ui.mes does. Since not all the notes that function this way have a need to mark a map or anything, not all the notes even have a san_use script: some just fire the text box and leave it at that.

Ok, what about the interactive ones? They actually create a critter (proto from the 14000s) that you 'talk' to, then disappears afterwards. I'll use Lila's list as an example:

First, the san_use script: this creates the critter and initiates dialogue with it:

```
def san_use( attachee, triggerer ):
_________loc = triggerer.location
_________npc = game.obj_create( 14697, loc )
_________triggerer.begin_dialog(npc,1)
_________return SKIP_DEFAULT
```

Simple, eh? The other end comes in the dialogue file:

```
{1}{[You unroll the parchment.]}{[You unroll the parchment.]}{}{}{}{}
{2}{Ok, what's next?}{}{1}{game.global_vars[699] >= 1 and game.global_vars[699] < 7}{10}{}
{3}{Let's see what we got here...}{}{1}{game.global_vars[699] == 0}{10}{}
{4}{Let's see what the last thing is...}{}{1}{game.global_vars[699] == 7}{10}{}
{5}{That was a looong quest.}{}{1}{game.quests[98].state == qs_completed}{0}{npc.destroy()}
{6}{I've done everything Lila asked, I should find out why she wanted it
all.}{}{1}{game.global_vars[699] > 7 and game.quests[98].state !=
qs_completed}{0}{npc.destroy()}
{10}{SUNOM (weaver) -> distilling apparatus,
MORGAUSE (Nulb) -> curette,
```

MYELLA (Church) -> cauldron,
PAIDA (away) -> eye of newt,
MANDY (missing?) -> bat wing,
CLARISSE (dead) -> crystal skull,
GLORA (Inn) -> fresh blood, and
HEMLOCK.
Thanks dearie! [signed Lila]}{SUNOM (weaver) -> distilling apparatus,
MORGAUSE (Nulb) -> curette,
MYELLA (Church) -> cauldron,
PAIDA (away) -> eye of newt,
MANDY (missing?) -> bat wing,
CLARISSE (dead) -> crystal skull,
GLORA (Inn) -> fresh blood, and
HEMLOCK.
Thanks dearie! [signed Lila]}{}{}{}{}
{11}{Ok, I've got 1. Well, it's a start.}{}{1}{game.global_vars[699] == 1}{0}{npc.destroy()}
{12}{Ok, I've got 2. Better get back to it.}{}{1}{game.global_vars[699] == 2}{0}{npc.destroy()}
{13}{Ok, I've got 3. Better get back to it.}{}{1}{game.global_vars[699] == 3}{0}{npc.destroy()}
{14}{Ok, I've got 4. Better get back to it.}{}{1}{game.global_vars[699] == 4}{0}{npc.destroy()}
{15}{Ok, I've got 5. Better get back to it.}{}{1}{game.global_vars[699] == 5}{0}{npc.destroy()}
{16}{Ok, I've got 6. Getting there...}{}{1}{game.global_vars[699] == 6}{0}{npc.destroy()}
{17}{Ok, I've got 7. Just one more!}{}{1}{game.global_vars[699] == 7}{0}{npc.destroy()}
{18}{It's good to be finished.}{}{1}{game.global_vars[699] == 8}{0}{npc.destroy()}
{19}{I've got a looong ways to go. [sigh]}{}{1}{game.global_vars[699] == 0}{0}{npc.destroy()}

You can read that or not as you desire: the important part is npc.destroy() whenever there is a line 0 (leave dialogue), this eradicates the critter after you have finished chatting to it. Otherwise, it hangs around and funky things happen if you go into combat or cast area-effect spells: early players of DH will remember this, because I didn't consistently have this for the critter created to interact with the Sandalwood Box. Minor hilarity would periodically ensue.

To see these critters, go to protos 14697 (as suggested by the san_use file above) or 14687. They are flagged OF_SHOOT_THROUGH OF_SEE_THROUGH OF_DONTDRAW, with the last one being the money shot: it means the critter isn't drawn onscreen, which would ruin the moment (though they might still cast a subtle shadow if you look closely). I usually just use a model of something tiny like a shuriken anyway. Note that the critters have their own .py files to join to their dlg files, with throw-away san_dialog entries (of course, you don't initiate the conversation by clicking on the crtitter, activating the san_dialog script: it is done through the san_use script. So its just there to link up the relevant dlg file). Hence Lareth's diary works from py00222larethdiary_book.py (the san_use file called by the diary object) while the critter that this summons works from py00221larethdiary_critter.py (whose san_dialog entry seems to be a copy of the san_use script. Well, there you go).

KotB Scripts tutorial (Part 1)

The following is a list of some of the new scripts being used in KotB. These may be updated in the future.

First and foremost, there are npc flags. Folks who read Flags Tutorial 2 will remember it dealt with object flags - not OF_OFF and such, but individual flags for individual objects. The one we particularly played with was

attachee.obj_get_int(obj_f_npc_pad_i_5) &

attachee.obj_set_int(obj_f_npc_pad_i_5, 1)

which was a simple way of storing a variable attached to the object. Cerulean used it for keeping track of recruitable shopkeepers as part of his fix that if you let them go, they returned to being shopkeepers (by having two - one to adventure, the other to remain attached to the shop inventory).

Anyways, in KotB I have written a very simple script to allow you to easily use these integers: it is in Scripts.py. It simply uses them as on-off flags (like game.global_flags[]) and is accessed in the following manner:

Firstly, put

from scripts import *

in the py file of the character that is going to use it, much the same way you use 'from utilities import *'.

Secondly, to set one of the flags, just use

npc_1(npc) or npc_1(attachee) depending on whether it is used from a dlg or py file.

There are 3 of these flags, called (funnily enough) npc_1, npc_2 and npc_3. Now, if you set them with npc_1(npc) (or npc_2(npc)...) you read them with

get_1(npc), get_2(npc) and get_3(npc)

Want to set one back to 0? Use

npc_1_undo(npc)

I wonder why I did it that way and not just undo_1(npc)? Heh... must have had a reason.

Anyways, here is an example of it in action from of one of the Watchmen in the Outer Bailey, who makes sure you have your weapons sheathed:

{750}{Citizen! Sheathe those weapons - it is an offense to show naked blades in the Keep.}{Citizen! Sheathe those weapons - it is an offense to show naked blades in the Keep.}{}{}{}{npc_1(npc); npc.reaction_adj(pc, -5)}

{760}{Citizen! I have told you before, the carrying of naked blades is an offense in the Keep. If this happens again I will have to take further measures.}{Citizen! I have told you before, the carrying of naked blades is an offense in the Keep. If this happens again I will have to take further measures.}{}{}{}{npc_2(npc); npc.reaction_adj(pc, -5)}

{770}{Citizen! Once again I find you walking the Keep with weapons in hand. This cannot continue. The standard fine is 10 gold: perhaps this will help you to remember to obey the laws.}{Citizen! Once again I find you walking the Keep with weapons in hand. This cannot continue. The standard fine is 10 gold: perhaps this will help you to remember to obey the laws.}{}{}{}{npc_3(npc); npc.reaction_adj(pc, -5)}

{790}{Citizen! I am at a loss... once again you stand here with weapons drawn. Do you wish the whole garrison to fight you? Enough is enough. You are under arrest.}{Citizen! I am at a loss... once again you stand here with weapons drawn. Do you wish the whole garrison to fight you? Enough is enough. You are under arrest.}{}{}{}{}

So in 750, 760 and 770 we see each of the 3 flags being set. And what subsequent effect does setting these flags have? Well, if we check the heartbeat file we see that depending on which of the flags are set, the Watchman will automatically escalate his response:

```
if sheathed(attachee, obj):
_________if get_3(attachee):
____________________obj.begin_dialog( attachee, 790 )
_________elif get_2(attachee):
____________________obj.begin_dialog( attachee, 770 )
_________elif get_1(attachee):
____________________obj.begin_dialog( attachee, 760 )
```

```
________else:
____________________obj.begin_dialog( attachee, 750 )
```

Obj is defined as any PC in range. This is a fragment of the heartbeat file, of course.

By doing it this way, it means Watchmen can react as individuals: if that particular guard has busted you before, he will remember it and be suitably unimpressed. If it is someone new who doesn't know you, they will treat it as a first offence. All this without tying up a bunch of global flags.

By now you are wondering precisely what 'sheathed' is: this is a script written for me by Darmagon, for memory. This one is in guards.py, so you put 'from guards import *' in your .py file if you want to use it.

```
def sheathed(attachee, triggerer):
________for pc in game.party:
____________________if pc.type == obj_t_npc:
_____________________________return 0
____________________else:
_____________________________weap = pc.item_worn_at(3)
_____________________________if weap != OBJ_HANDLE_NULL:
_________________________________________if (weap.obj_get_int(obj_f_material) != mat_wood):
_____________________________________________________return 1
________return
```

Obviously the point of this is to check if you have your weapons sheathed. So we see firstly it only deals with folks in the party (not mercenaries, who can't be unequipped). Secondly it checks if the party member is a PC or an NPC follower, and if it is an NPC it just ignores it (I forget why, but again I had a reason). If it is a PC, it checks if the PC has something equipped at their primary weapon slot: slot 3, as detailed by Darmagon in the Well Whaddya Know? thread:

Item slot as per item_worn_at

0-hat/eyeglasses
1 necklace
2 gloves
3 primary weapon
4 secondary weapon / big shield
5 armor
6 primary ring
7 secondary ring
8 boots
9 ammo
10 cloak
11 2nd shield spot/ buckler spot
12 robe
13 bracers
14 bard's instrument
15 thieves tools

If there is something there in the slot (ie it does NOT return a null finding - gotta love those double negatives) then it checks if it is made of wood. If not, (finally) it returns a 1, so our initial query from the guard:

if sheathed(attachee, obj)

having returned a 1, goes ahead and fires: otherwise it doesn't. (The 'wood' thing, btw: most wooden weapons - crossbows, heavy flails, spears, staves, longbows etc - cannot be 'sheathed' in the convetional sense, so are exempt. I thought it was a distinction worth making).

Now, what happens if the guard finds you have got something equipped? Other than the dialogue, an unequip script will run (since asking the player to go around and do it manually is pretty ordinary). That is also in the guards script, and calling it looks like this:

{755}{Make sure it does not happen again.}{Make sure it does not happen again.}{}{}{}{unequip(3, pc); unequip(4, pc)}

The script being called looks like this (again it is a Darmagon):

```
def unequip( slot, npc):
    for npc in game.party:
        item = npc.item_worn_at(slot)
        if item != OBJ_HANDLE_NULL:
            if (item.obj_get_int(obj_f_material) != mat_wood):
                holder = game.obj_create(1004, npc.location)
                holder.item_get(item)
                npc.item_get(item)
                holder.destroy()
    return
```

This little script unequips the party members by creating a spare container ('holder'), moving the weapon in the slot over to the container, moving it back to the party member (where-in it will simply reappear in their inventory and not be equipped) then the holder is destroyed. Funky.

Note that by calling this twice for slot 3 then slot 4, it unequips both the primary and secondary slots. Also note that that it doesn't distinguish between NPCs and PCs - it unequips everyone in the party for thoroughness. So your NPCs are set up to behave like everyone else, but if they re-equip through some heartbeat moment (if they are kicked out then rejoined for soem internal reason, or some such thing) the party are not penalised by having this show up as a weapon-bearing moment to the guards.

KotB Scripts tutorial (Part 2)

Ok, todays tutorial is about Climb / Use Rope, which has a somewhat complicated scripting process that I still get muddled with myself on occasion. I am quite proud that we have added new skills to the game. It really allows people to customise their characters. Not long ago, someone posted a thing in the 'Temple of Elemental Evil' forum on Co8 (I think it was that forum, I am not going back to look) asking what skills to bother giving a cleric. It was a fair point - once you've got Concentration, what would you bother with? Spellcraft, just because? A bunch of cross-class stuff?

Hopefully in years to come we will be turning out modules with all the Knowledge skills implemented. For now we aren't, so a couple new skills give people the chance to at least have some more choice, something useful in-game to apply their skill points, and maybe a difficult choice or two, where an RP-choice has to be made between competing needs.

That reminds me - we really have to have a combat at some point straight off a rope, so people who take their armour off before climbing are at a disadvantage (as they should be).

But back to the scripts. Climb is implemented in two separate ways in KotB: one through a dialogue where you get to climb down a hole (I won't be bother with that one here), the other through using the rope item.

Rope has always been in the game at 12011. Whether it was ever meant to do anything, I have no idea, but its nice to have it already there. I've made a few alterations, I think I'll go through them one by one since this is a neglected part of my tutorials: making new spells and such (I am by no means an expert but I have done maybe 7 new spell effects for KotB). Essentially what we are doing is setting the rope up as a 'wand' that casts a specific spell, 'Use Item'. The reasons for this will be outlined as we go along.

The changes to the rope prototype (12011) don't kick in until col 50, the item flags. They now read:

OIF_EXPIRES_AFTER_USE OIF_NO_NPC_PICKUP OIF_USES_WAND_ANIM

Straightforward. Firstly, you use it then it expires - secondly, NPC followers don't loot it (added to pretty much all items by an early incarnation of Co8 to prevent weird follower looting) - thirdly, it uses the wand animation. This means when you use the rope, you do so with a little 'throwing' animation like flicking a wand: very pleasing effect! :-)

Chugging along we hit col 59, which says our 'wand' has one charge before expiring: so we get to use each rope once. Why does it expire? Well, it is assumed when you tie a rope and climb down it then it is left there. Keep in mind we haven't just implemented 'Climb' but 'Use Rope' as well. So when you climb down a rope, it checks to see whether you have successfully tied a slip-knot (or whatever its called) that can then be released from the bottom. If so, you get your rope back (a new one is created) - if not, your rope is left there, albeit that spot is flagged so in future you can access it again without using up another rope.

There's nothing more until col 168. This is where items keep their 'powers' or bonuses. Armour bonuses, shield bonuses, weapons being chaotic or holy, skill circumstance bonuses, you name it and it appears in here somewhere. Our bonus is simply that it is a Usable Item. This means (essentially) it appears in the radial menu: stick it in your inventory and you can 'use' it through the radial menu (I don't know what the 0 and 1 parameters do precisely, but suffice it to say they are necessary and I tried a lot of variations on this setup to finally get the rope to work like this). Scoot over to col 312 and we get our spell list, which has the 'spell' Use Item: this is set up as 'Domain Special' so you don't have to be a wizard or whatever to use it. When we use our 'wand' through the radial menu (to spell it out one more time) this spell effect gets cast. What we see, though, is we choose 'rope' through the radial menu and our little fella onscreen does a throwing animation and we get teleported to wherever climbing the rope takes us.

We'll have a look at this in a minute but first, you might be asking, why use a spell? Why not just use a san_use script attachment?

I know I've ranted about this before but its my blog so I will say it all again. We should be able to do it that way: the game would be massively enhanced if we could just add a simple san_use script to generic items without having to frig around with new spells etc. San_use it not broken, per se - we can use it on written items to create little dialogue boxes, as we saw in the 'creating notes' tutorial (http://rpg-rant.blogspot.com/2007/06/creating-notes.html). We can use it to make doors work on command, or not - I did it with the ladder in the Inn (which works perfectly) and the doors into the Thieves' Guild and Warehouse (which work clumsily, but Liv has come up with a better way that I will hopefuly get a chance to implement soon).

BUT too often we are reduced to making new spells to implement scripts that should be done directly through the san_use command. Its annoying, and its prohibitive, and I don't like it.

For the rope, we're stuck with it. So lets get back to it. We have wandered down a dungeon corridor and come to a hole: we want to climb down with our rope. We select rope through the radial menu and the animation fires, and behind the scenes our 'spell' is cast: now what?

Opening Spell783 - Use Item.py and we find the usual conglomerate of OnBeginSpellCast(spell), OnSpellEffect(spell) and OnEndSpellCast(spell). These occur in every spell, and I won't go into it here.

The important bit is in the spell effect area: the whole script is fairly repetitive but I will go through it here so that if anyone ever wants to add a climbing point of their own, they can easily understand what to do. Here it is:

```
________if game.leader.map != 5038 and game.leader.map != 5069 and game.leader.map != 5001 and game.leader.map != 5066:
____________________spell.caster.float_mesfile_line( 'mes\\narrative.mes', 1001 )
____________________create_item_in_inventory(12011,spell.caster)
```

```
_________elif game.leader.map == 5001:
____________________for obj in game.obj_list_vicinity(spell.caster.location,OLC_GENERIC):
_________________________if obj.name == 12797:
_____________________________npc = find_npc_near(spell.caster,14144)
_____________________________if (npc != OBJ_HANDLE_NULL) and
(spell.caster.distance_to(npc) < 40):
_________________________________spell.caster.begin_dialog(npc, 350)
_________________________________create_item_in_inventory(12011,spell.caster
)
_________________________________npc.turn_towards(spell.caster)
_____________________________else:
_________________________________if (spell.caster.distance_to(obj) > 9):
_____________________________________spell.caster.float_mesfile_line(
'mes\\narrative.mes', 1002 )
_____________________________________create_item_in_inventory(12011,s
pell.caster)
_________________________________else:
_____________________________________spell.caster.float_mesfile_line(
'mes\\narrative.mes', 1006 )
_____________________________________game.timevent_add(teleportrope,
(), 1300 )
_________else:
____________________for obj in game.obj_list_vicinity(spell.caster.location,OLC_GENERIC):
_________________________# rope down
_________________________if obj.name == 12797:
_____________________________spell.caster.turn_towards(obj)
_____________________________if (spell.caster.distance_to(obj) > 9):
_________________________________spell.caster.float_mesfile_line(
'mes\\narrative.mes', 1002 )
_________________________________create_item_in_inventory(12011,spell.caster
)
```

```
_____________________________________else:
_____________________________________________if (game.leader.map == 5038 and
game.global_flags[54] == 1) or (game.leader.map == 5001 and game.global_flags[53] == 1):
_________________________________________________spell.caster.float_mesfile_line(
'mes\\narrative.mes', 1009 )
_________________________________________________create_item_in_inventory(12011,s
pell.caster)
_____________________________________________else:
_________________________________________________spell.caster.float_mesfile_line(
'mes\\narrative.mes', 1006 )
_____________________________________________game.timevent_add(teleportrope, (), 1300 )
_____________________________# rope up
_____________________________elif obj.name == 12793:
_________________________________spell.caster.turn_towards(obj)
_________________________________if spell.caster.item_find(12792) ==
OBJ_HANDLE_NULL:
_____________________________________spell.caster.float_mesfile_line(
'mes\\narrative.mes', 1007 )
_____________________________________create_item_in_inventory(12011,spell.caster
)
_________________________________elif (spell.caster.distance_to(obj) > 9):
_____________________________________spell.caster.float_mesfile_line(
'mes\\narrative.mes', 1002 )
_____________________________________create_item_in_inventory(12011,spell.caster
)
_________________________________else:
_____________________________________spell.caster.float_mesfile_line(
'mes\\narrative.mes', 1006 )
_____________________________________game.timevent_add(teleportrope, (), 1300 )
_____________________________________create_item_in_inventory(12011,spell.caster
)
```

_____else:
_____spell.caster.float_mesfile_line('mes\\narrative.mes', 1001)
_____create_item_in_inventory(12011,spell.caster)
_____spell.spell_end(spell.id)

What a mouthful! Firstly, going through the 'sections', these can be divided up into:

_____if game.leader.map != ####...

_____elif game.leader.map == 5001:

_____else: # rope down

_____# rope up

_____else:

Only five sections: thats easier.

Section one checks what map you are on. Only certain maps have rope points: a list of maps is checked, and if you are not on any of them, then a floatline fires (# 1001, {Not here!}) and a rope is created in the the 'spell.caster's' inventory - remember, the rope expires on use, so any time it is not used properly, it has to be replaced. (This is the reason why I haven't implemented silk rope, if anyone has been keeping track - people would be legitimately aggrieved if they spend the money on a silk rope for its +2 bonus, try it somewhere likely-looking, get the 'not here!' floatline, and have their nice silk rope replaced with a normal one. A more complicated script could check for it, but I haven't done it yet :^)).

If you are on a viable map, then it moves on to section two and checks if you are on map 5001. This is the Keep Outer Bailey map and it is singled out for a reason: if you try to use your rope

to climb into the well, and there is a watchman around, he will quite legitmiately ask what the heck you are doing and prevent you. Good for him. Also, this will be probably the only map where there will be multiple rope points: the current one is the well, there is a rope-up point planned for climbing up onto the wall (mentioned by one character late in the Demo but not yet done), and a rope-across point that I won't go into here.

Lets ignore the rest of the 5001 option and move straight onto the Rope Down option. The Rope Up and Rope Down sections are prefaced by the money line - the one that makes it all work:

for obj in game.obj_list_vicinity(spell.caster.location,OLC_GENERIC):

Climbing works by searching for a 'rope point' object, a generic item (12000s) which can be one of 3 types:

{12791}{Rope Across Hook}
{12792}{Grapple}
{12793}{Rope Up Hook}
...
{12797}{Rope Down Hook}

I threw the grapple (or 'grappling hook' if you prefer) in there because thats about to come up and I may as well show it now.

So.... having quickly assessed what generic items are lying around nearby (usually very few) the game next tests if there is a Rope Down hook. If so, the distance is calculated: you can only use your rope if you are right near your rope hook (ie near the hole, or whatever the hook is attached to). If you are too far away (further than 9 - thats pretty close) then you get another floatline:

{1002}{Move closer!}

I might speak a little more about what exactly these rope hooks are. Model-wise, they are leather armour - yup, a piece of leather armour lying there. Popping the prototypes of 12793 and 12797, we find they are both OF_CLICK_THROUGH. So you never get to touch them or pick them up (or be able to feel them there). thats not much... I must have added the OF_DONTDRAW manually in the mobs. Now here's something I never mentioned before (but have encountered a few times) - if you add something manually in the mobs, because of the way flags are set up in the mobs, it will OVERRRIDE the prototype flags, so you have to add the flags all over again. The mob saves a flag register for every type of flag you select for it, so if you just have OF_DONT_DRAW yiour flag register for the Object flagsd might look something like:

0000 0000 0010 0000

so you have zeroed out the OF_CLICK_THROUGH! Now, this may not be a big deal in this case (I doubt you can click on something that is not drawn), but be aware of it: I added a NO_TRANSFER item flag to a mob, and this negated the DRAW_WHEN_PARENTED flag in the prototype which meant instead of drawing the item in the NPCs hands, it drew it at his feet, and when he moved around, it just stayed there on the ground rotating to match him! Damn weird and a lot of stressing trying to figure it out.

While I am warning you about this, I will issue the same warning for the script override tool. If you use this on a mob to add a heartbeat, where the prototype has, say, a san_dialog attachment, you have to add the san_dialog all over again in the script override tool. Obviously saves them in the mobs the same way.

Ok... lets say we are indeed climbing down somewhere and the game has found the Rope Down hook and determined we are in range. Next it does a by-map-number flag check to see if you have used this hole (or whatever) before and left a rope dangling there, because you failed your Use Rope check when you got to the bottom (think Sam and Frodo in the Emyn Muil, of

course). Since there are a finite number of ropes in the game, we can't have you running out because you keep leaving them everywhere and rolling 1s on your Use Rope check.

If the relevant flag is checked, it says:

{1009}{Climbing the rope tied here!}

and replaces your rope. If not it just says {1006}{Using rope!}

Then comes the actual moving of the party, via a timed event, teleportrope(). It is done as a timed event to give you a moment to read whatever floatline appears and to see your little 'throw the rope' animation, which really is quite effective.

Before I speak about teleportrope(), lets do Rope Up.

The main difference is, after Rope Up checks if there is a rope hanging there, it then checks for a grappling hook: unless you've tied a rope from the top, you won't go far without a grappling hook! If you don't have one, then it says:

{1007}{Needs grapple!}

Otherwise Rope Up functions exactly like Rope Down. Note it adds a rope to your pack - you never lose a rope climbing up, and don't have to make a Use Rope check. We also have a Rope Across hook but as I said above it isn't use yet.

So... straightforward, really. Now comes the tricky part.

We have to do the Use Rope check AND the check to see if you have fallen and sustained damage AFTER the map change. This is a problem: we can't trigger a time event to happen after we cross the map, since there is nothing there on the new map to trigger it: no 'attachee', if you will. Our attachment points have been generic items, which don't get to fire heartbeats or

san_new_maps. Even if our rope hooks were OF_DONTDRAWn critters, rather than generic items, they still wouldn't be there on the new map and the scripts wouldn't cross.

We have 3 options:

1) Put OF_OFFed or DONTDRAWn critters at the landing points, set a flag each time the rope is used, and have these critters have a first_heartbeat file that checks for that flag and if present, does the necessaries. I didn't do this (too clunky, all the critters everywhere).

2) Use Spellslinger's Persistant Data mod to handle things across the map change, since that is pretty much what it is designed for. I didn't do that either (don't really understand how it works) but eventually this is probably how it will be done: the most elegant and least disruptive method.

3) The current method: manufacture an NPC and attach him to the party temporarily, then fire all the scripts off his san_new_map attachment.

Hence we find this in the teleportrope() script:

```
def teleportrope():
_____if game.leader.map == 5038:
_____________rope_item = game.obj_create(14591, game.leader.location + 1)
_____________game.leader.ai_follower_add(rope_item)
_____________game.fade_and_teleport( 0,0,0,5039,492,479 )
_____elif game.leader.map == 5066:
_____________rope_item = game.obj_create(14591, game.leader.location + 1)
_____________game.leader.ai_follower_add(rope_item)
_____________game.fade_and_teleport( 0,0,0,5069,503,388 )
_____elif game.leader.map == 5001:
_____________rope_item = game.obj_create(14591, game.leader.location + 1)
_____________game.leader.ai_follower_add(rope_item)
```

```
____________________game.fade_and_teleport( 0,0,0,5038,505,469 )
_________return
```

We can see that the sole difference between the various sections is where the teleport lands you: otherwise in each, a critter is created and attached to the party as an ai follower (like a commanded undead or pet). This may beg the question, what if the party is full? Well, one of these days I will write a san_heartbeat addition for the Watchmen to tell people not to fill their party (and more to the point, to check if the party is already full). If it is, they will subtly tell the player there is a problem to be fixed. The thing is, the two instances of co-operative combat in the game are done by adding the person you are fighting alongside to the party - again, this may change if we can ever figure out how to check factions and do it that way (or if anyone figures out any other way to do it) but for now we just surreptitiously add the person being defended to the party. So you can well see keeping one slot in the ai section open has a couple of uses, and will just have to be done (there are 10 slots anyway. Even if all your characters have familiars / pets, thats still only 8 - and thats pretty unlikely).

Anyway, lets assume the teleport has taken place and you are on the new map. Now your critter's san_new_map fires, lets see what it does (its in py00520RopeUse.py):

```
def san_new_map( attachee, triggerer ):
_________if game.global_vars[36] == 1:
____________________game.global_vars[36] = 0
____________________falls(attachee, triggerer)
_________elif game.global_vars[36] == 4:
____________________game.global_vars[36] = 0
____________________climbs_sans( attachee )
_________elif (attachee.map == 5069):
____________________climbsup(attachee)
_________else:
____________________climbsdown(attachee)
_________game.leader.ai_follower_remove(attachee)
```

```
_________attachee.destroy()
_________return RUN_DEFAULT
```

The first lot - checking for variable 36 - is from the 'climb via dialogue' option we mentioned at the very beginning of the tutorial but never went back and looked at. I'm not going to look at it now either - the purpose of this tutorial is to explaion how rope works so people can add fully functioning rope hooks if they want. So lets skip to the two subroutines climbsup(attachee) and climbsdown(attachee): and note after these run, the game returns, removes the ai follower critter, and destroys it (not removing it before destroying will give you one of those annoying 'blue circle in my party' moments).

```
def climbsup(attachee):
_________t = 1
_________for clown in game.leader.group_list():
____________________roll = game.random_range(1,20)
____________________effort = clown.skill_level_get(attachee, skill_climb)
____________________result = roll + effort
____________________if result >= 5:
_____________________________t = t+500
_____________________________game.timevent_add(comment4, (clown), t )
____________________else:
_____________________________ring1 = clown.item_worn_at(6)
_____________________________ring2 = clown.item_worn_at(7)
_____________________________if ring1.name == 6650 or ring2.name == 6650:
_________________________________________t = t+500
_________________________________________game.timevent_add(comment5, (clown), t )
_____________________________elif clown.stat_level_get(stat_level_monk) >= 4:
_________________________________________t = t+500
_________________________________________game.timevent_add(comment7, (clown), t )
#_____________________________elif clown.type == obj_t_npc:
#_________________________________________for m, n in flying_squad.items():
```

#_____if clown.name == n:
#_____t = t+500
#_____game.timevent_add(comment2, (clown), t)
#_____else:
#_____t = t+500
#_____game.timevent_add(comment6, (clown), t)
_____else:
_____t = t+500
_____game.timevent_add(comment6, (clown), t)
_____return

Sooooo... firstly, there is the little command t = 1. This is essential: t is going to be a counter, going ahead by 500 (milliseconds) to stagger the reports for each member of the party. 'for clown in game.leader.group_list()' checks the whole party (including ai followers) and reports what happens, and these reports are staggered by 500ms (1/2 a second) so they cascade on the screen rather than all come up at once. So you can read them, if you want to.

Now, here's the important part... if you DON'T set t as something like 1, but just assume the computer will assume that t is 0 (which I might add I think it SHOULD) then the whole thing doesn't work. Don't ask me why. You have to set t as something.

O - and why do I call them clowns? Because I wrote the 'jump down without rope' dialogue climb thing first, and copied the rest off that. Only a clown would jump down without climbing.

So, within the 'for...' routine, each member of the party has to make their climb check. This is done via a simple random 1-20 roll, and comparing it to DC 5. If you make it, then a time event fires, timed according to the currrent value in t, stating that you climbed the rope succesfully, t advances by 500 for the next party member, and thats that.

Why DC 5? Because climbing a rope is DC 10, and you have to screw up by 5 or more to actually fall. If you screw up by less than 5, you just don't progress, though two fails in a row also make you fall. To be honest, I should probably implement that, but for the moment I have this dumbed-down version whereby you either fail catastrophically and fall, or you just succeed. And even if you fall, you only fall once, then you are assumed to have made the climb. I figured no-one wanted to see their fighter in full plate, making his checks at -8 or whatever, take 6d6 damage just from failing half a dozen checks in a row before succeeding. Thats silly: characters could die that way without the player having any input, and thats just plain bad.

What if you do fall? Just take 1d6 damage? (All distances in KotB atm are 10 feet). Noooo... what about 4th level monks, or characters wearing a new ring of feather fall? Both of these are implemented at this point. Note there is also a new ring of climbing, but this just adds +5 to your climbing skill and doesn't have to be checked for right now. Item slots 6 & 7 are your left and right hands, of course (for rings, not weapons or shields). After this a monk level above 4 is checked for. (Yes I am also going to have to add a check for Half_knight's new Wings of Pain, but believe me, I don't mind! 8^D)

Next, there is a bit that is currently remarked-out with '#': the check for the flying squad. The flying squad are those familiars that can fly - hawks and doves and bats and such. Since we check for the whole group and not just the 8-man party, we have to keep in mind that some familiars etc can fly and wouldn't fall. Makes me think I better check for those who have a decent climb score and make sure it is in the prototype (familiar spiders and such).

Its remarked out because I have not tested it yet - I don't use summonings or familiars at the best of times, never have. Don't know why, just don't. Anyway, the flaying squad has to have been previously defined: it is, looking like this:

```
#flying_squad = {
#small_air_elemental: 14380,
#dire_bat: 14390,
```

#fiendish_bat: 14407
#}

Not finished, obviously! Not tested either, but I do seem to remember at least getting it into a format that didn't break the .py file. Anyways, for anyone wanting to know how to handle arrays like this, I thought I would throw it in. A working version is found in pc_start.py (dealing with all the possible weapon feats that people could have that would lead to them getting a weapon when they start the game).

So what happens if any of these conditions are met? Well, a floatline appears explaining what happened: here are some examples.

{100}{Fell safely due to Monk abilities!}
{101}{Fell 10 feet, taking damage!}
{102}{Climbed the rope but fell, taking damage!}
{103}{Climbed the rope safely!}
{104}{Climbed down safely!}
{107}{Flew down safely!}
{108}{Fell but unharmed due to Monk abilities!}
{109}{Fell but unharmed due to Ring of Feather Fall!}
{110}{Fell while climbing, taking damage!}

Some are for the 'climb by dialogue' option of climbing without the rope. The DC is 10 then instead of 5. Where appropriate there is damage, and if that happens, you are assumed to have fallen so you wind up prone. (Thats in 'climbsdown' but not in climbs up, since if you fell while climbing up, it is assumed you got back up and climbed up again). Adding damage looks like this:

clown.damage(OBJ_HANDLE_NULL,D20DT_UNSPECIFIED,dice_new("1d6"),D20DAP_NORMAL)

while making the character prone looks like this:

clown.condition_add_with_args('prone', 0, 0)

Another thing we find in climbsdown is the 'get rope check', called thusly:

```
_________t = t+500
_________game.timevent_add(get_rope, (attachee), t )
```

which is the very last thing done. This check sees if you can 'recall' your rope and not leave it tied there. This is a DC 15 check of the Use Rope skill.

```
def get_rope(attachee):
_________w = 1
_________for roper in game.party:
____________________roll = game.random_range(1,20)
____________________effort = roper.skill_level_get(attachee, skill_use_rope)
____________________result = roll + effort
____________________if result >= 15:
_____________________________w = w+500
_____________________________game.timevent_add(comment8, (roper), w )
_____________________________create_item_in_inventory(12011,roper)
_____________________________if game.leader.map == 5066:
_____________________________________game.global_flags[47] = 0
_____________________________elif game.leader.map == 5038:
_____________________________________game.global_flags[53] = 0
_____________________________return
_________w = w+500
_________flagging()
_________game.timevent_add(comment9, (roper), w )
_________return
```

Note the resetting of the flags that check if you have left the rope there: you don't get a choice to leave it there for later, its assumed you always try and take it with you. Deal with it.

Now, this is a really dumbed down little routine: it just runs the Use Rope skill of everyone in the party. Its a bit like Appraise or Survival - you don't have to worry about who is getting checked, because everyone does. If you succeed, you get a rope in your inventory and the flag that says a rope is there is set to 0 (in case you were climbing down a rope that was left behind before). That can make for a weird moment, if your rope and grapple are always in the thief's hands and its the druid who makes his Use Rope check - you have to go hunting for where the rope is. Small price to pay. If you fail, a small routine called 'flagging()' is run, to set the appropriate flag that leaves a rope at the spot.

Then, it hits return. There's been a few of those. Where are we returning back to?

We go right back to the san_new_map() call: once all these subroutines have run, our little interloper removes himself from the group and detroys himself. Job done: you have changed maps to whereever you climbed to, got the rope back if you had the skill, taken damage if you fell. Finished!

So, finally, how do you add a rope point in the game then? Well...

1) You add the Rope Up or Rope Down hook at the point where you want it to function (where you want to climb from, obviously - where you want a character using his rope to get a response).

2) You add the map number to the first line of the Spell783 - Use Item.py script (this is just a little performance thing - by checking immediately whether there is a viable use for the Rope on that map, rather than going through the whole 'look for a nearby object' routine, it should improve the performance on any map where there are objects around but not rope points. Just

a little thing, but considering we are modding, and mods invariably come with performance issues, well, every little bit helps).

3) You add the arrival bit (where the rope is going to take you) to the teleportrope() subroutine in Spell783 - Use Item.py.

4) You pop py00520RopeUse.py and alter the get_rope() and flagging() subroutine with a new flag to allow for the presence of the rope, if someone leaves it there. Then go back and factor this in to the Rope Up section of Spell783 - Use Item.py.

Adding New Meshes

This is a blog I have been meaning to write for a long time, not because I have anything to add to this subject - I generally leave these things up to the artists, and the groundbreaking stuff has been written about and explained previously by the folks who discovered it such as Cujo, Cuchulainn and Allyx - but because I often get tied up in knots myself and forget some important element, so I thought I would do a step-by-step idiot-proof guide for myself and other folks who are interested in this area of modding could then get into it easily.

Today I am going to add some new meshes to KotB: new boots, armour etc made by Half Knight. He has done the hard yards changing the graphics and I am going to put them into the game and record how that is done. I won't attempt to explain model-making, bones, skeletons, alpha-channels or any of the other weird stuff that the folks who make new things talk about: apart from technical nous, that sort of thing also requires artistic talent, something I don't have. What I will explain is simple techniques for making new meshes and the various files that have to be altered to see them in the game.

What is a mesh? By that, I mean the texture files that you see on screen on the various models - NPCs, weapons, clothes etc: the .jpgs or .tgas that wrap over the top of the models. Today we will specifically be looking at clothes. Different sorts of clothes can be made from one model by using different meshes: thus there is one model for female 'mystic garb' - the saucy

little number worn by Elisteria in *DH* - but it can come in different colours, by putting different meshes on the models. The game currently has 'blue' (which is greenish) and 'red' (which is pink). Fabulous. Half Knight has already made some new ones: blue and red that are just that, and black and white as well. Lets start by adding the new prototypes for them as a base, and as we fill in each column, we can have a look at where each file goes and how they interact with each other.

In KotB, we'll put the new stuff at 6340 onwards, since there are some new robes there. So we add the following lines:

{6340}{Red Mystic Garb}
{6341}{Blue Mystic Garb}
{6342}{White Mystic Garb}
{6343}{Black Mystic Garb}

We also alter 6211 and 6213 to 'Cyan' and 'Pink' to better reflect the reality of their colours.

Then we back up protos.tab and add 4 new lines at 6340-6343. I'll assume you can do the basics like adding the obj type in column 1, the Id# in col 23, size_medium in 24 etc. If you aren't sure, just go back up to 6211 and copy those sorts of things (weight, cost etc too).

Material is an interesting one: its mat_cloth. No surprises there, but the more knowledgable or observant will have noticed that in data/rules, next to the protos.tab file that you just backed up before adding stuff to it, there is a file called 'materials.mes'. The materials column does NOT refer to this file, but we WILL be playing with this file in making our new meshes. Later.

Category is '1', same as 6211. The first column of real interest is col 34, 'model'. This is very important, it tells the prototype exactly where to find the model to display the item on screen. The 5 digit number (usually in the 12000s) that you see in the other armour and clothing prototypes (this section, the 6000s, includes pretty much everything wearable, such as boots, belts, goggles, shields, amulets and headbands, as well as armour and clothes), well this

'model' number refers to meshes.mes. Thats way down in the innards of the art folder, so lets go there:

data\art\MESHES\meshes.mes

Open it up and you will see thousands of entries - thats every damn model and mesh in the game, right there! Ducking back to prototype 6211 we see that the original blue (now Cyan) mystic garb refers to line 12131, which in turn says

{12131}{armor\sorcerer\sorcerer_blue_ground}

This, obviously, refers to a file or folder location. Going back to the meshes folder, we find an armor folder, and opening that we find... well, lots of stuff, including folders named marauder, gladiator, leather_scale and other stuff made by Co8ers (Cujo and Maggit in those cases). We don't find a **sorcerer** folder, though, since no-one has ever made new mystic garb for the game prior to this (or at least they have never posted it). To see the original, we have to go to the dats, so going to ToEE1.dat we find

ToEE1\art\meshes\Armor\Sorcerer

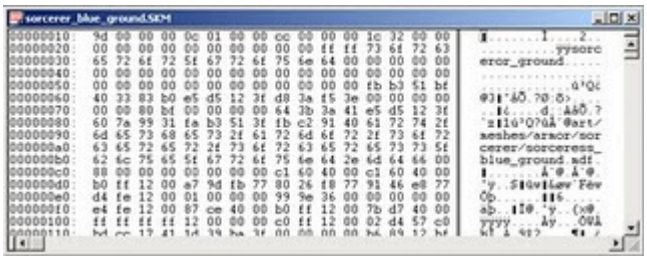
and popping that open, we find a huge number of files. Before we start dissecting those, lets put our new entries in meshes.mes: since both Half Knight and Cuchulainn are currently making all sorts of stuff, I will pick some obscure numbers that shouldn't get in anyone’s way and stick this particular bunch of new stuff at 12299, working backwards. Since they will all be going in the sorcerer folder we can copy that bit from 6211, but we have to rename the destination file since the game already recognises sorcerer_blue_ground, sorcerer_red_ground and sorcerer_green_ground. We'll call ours sorcerer_nblue_ground (for 'new blue'), sorcerer_nred_ground ('new red'), sorcerer_black_ground and sorcerer_white_ground. There is actually a logic to this, which I will reveal in a minute. In any case, the following lines will now appear in meshes.mes:

{12296}{armor\sorcerer\sorcerer_nred_ground}
{12297}{armor\sorcerer\sorcerer_nblue_ground}
{12298}{armor\sorcerer\sorcerer_white_ground}
{12299}{armor\sorcerer\sorcerer_black_ground}

Quickly going back to protos,tab, then, in column 34, 'model' we now put 12296, 12297, 12298 and 12299.

Ok, so now when we hit 'give 6340' in the console, the game knows to display this prototype it has to create an item from model #12296, and by going to meshes.mes and looking at line 12296 it knows the relevant files for that model are found at **data\art\meshes\Armor\Sorcerer** in something called sorcerer_nred_ground. Sweet! But what are those files?

Before we create file sorcerer_nred_ground and friends, lets look at an existing one in the dat: specifically sorcerer_blue_ground. Lets open it up. Eek, there's two of him! sorcerer_blue_ground.ska and sorcerer_blue_ground.skm. One is 1kb and the other is 15kb. We'll have a quick look inside the bigger one. Opening it with HexEdit, it looks like this:

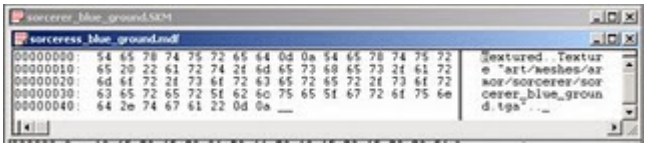


Ugh, what a mess! Obviously this is not meant to be viewed as a text file. What .skm means, afaik, is 'skeletal model' - this is the actual bones of the animated 3d models we see in the game. We are not going to make anything sensible out of it without looking at it with a model viewer, but we're modders - that won't stop us hacking it and hacking it good!

One thing in all that mess was intelligible - the bit that says

art\meshes\Armor\Sorcerer\sorceress_blue_ground.mdf

Hmmm... note it says 'sorceress'. Keep in mind, the male and female versions of this model are radically different: we have to create male and female files. Ok, what does the .mdf file look like? Opening it we see it says:



Ok, thats legible! It is pointing to the actual tga graphic that is the texture that gets wrapped around the model - thats the one we are going to change to produce our new clothes. So now we can see how the files interact:

protos.tab gets the model from meshes.mes
meshes.mes gives the location of the files
.skm gives the model, and looks to .mdf for the mesh
.mdf gives the name of the mesh graphic file

Easy! If you are wondering what .ska does, as .skm means 'skeletal model' I think .ska means 'skeletal animation' - it handles the model animations (how the mystic skirt flaps as the player runs, etc). They don't change from one type of mystic garb to the next so we just have to copy that.

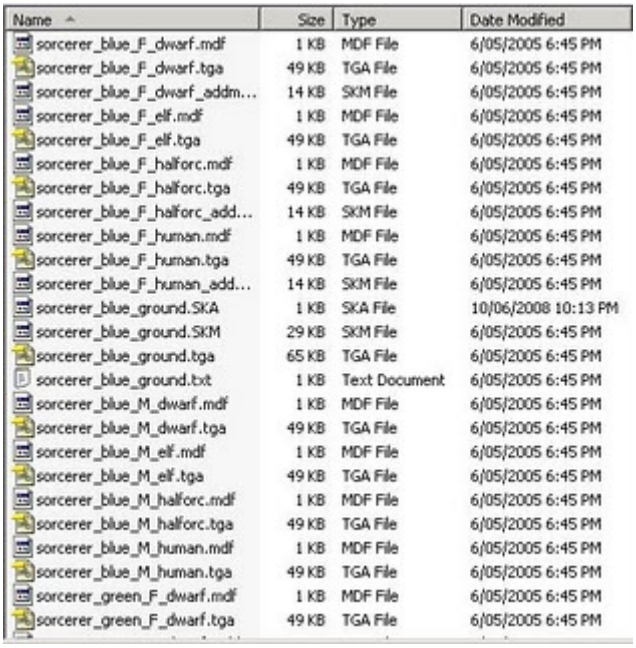
Thats our next job, we are going to copy all these files for our new stuff. We already decided what to call them: prototype 6340 will look for model sorcerer_nred_ground. So we have to have some .ska, .skm and .mdf files named accordingly (not to mention the tga files we see onscreen, which is the point of it all!)

So: we are going to take the sorcerer_blue_ground files we have been looking at and copy them and rename them sorcerer_nred_ground. Note we are using the BLUE ones - we are

NOT using sorcerer_red_ground or sorcerer_green_ground. ONLY USE BLUE. The reason why will become very clear in a minute.

When I say 'copy', we have no 'sorcerer' folder in our KotB art/meshes folder, so the first thing we do is make a new folder in there called 'sorcerer' being VERY careful to name it correctly. As you can now tell, all the files are looking for very specifically named things, and a single typo will bring an instant CTD in the game when you try to display the model, so be thorough. I am simply going to copy the whole sorcerer folder over from ToEE1.dat then rename and cull everything as necessary to prevent typo issues.

Egad, there is a lot of them...



Name	Size	Type	Date Modified
sorcerer_blue_F_dwarf.mdf	1 KB	MDF File	6/05/2005 6:45 PM
sorcerer_blue_F_dwarf.tga	49 KB	TGA File	6/05/2005 6:45 PM
sorcerer_blue_F_dwarf_addm...	14 KB	SKM File	6/05/2005 6:45 PM
sorcerer_blue_F_elf.mdf	1 KB	MDF File	6/05/2005 6:45 PM
sorcerer_blue_F_elf.tga	49 KB	TGA File	6/05/2005 6:45 PM
sorcerer_blue_F_halforc.mdf	1 KB	MDF File	6/05/2005 6:45 PM
sorcerer_blue_F_halforc.tga	49 KB	TGA File	6/05/2005 6:45 PM
sorcerer_blue_F_halforc_add...	14 KB	SKM File	6/05/2005 6:45 PM
sorcerer_blue_F_human.mdf	1 KB	MDF File	6/05/2005 6:45 PM
sorcerer_blue_F_human.tga	49 KB	TGA File	6/05/2005 6:45 PM
sorcerer_blue_F_human_add...	14 KB	SKM File	6/05/2005 6:45 PM
sorcerer_blue_ground.SKA	1 KB	SKA File	10/06/2008 10:13 PM
sorcerer_blue_ground.SKM	29 KB	SKM File	6/05/2005 6:45 PM
sorcerer_blue_ground.tga	65 KB	TGA File	6/05/2005 6:45 PM
sorcerer_blue_ground.txt	1 KB	Text Document	6/05/2005 6:45 PM
sorcerer_blue_M_dwarf.mdf	1 KB	MDF File	6/05/2005 6:45 PM
sorcerer_blue_M_dwarf.tga	49 KB	TGA File	6/05/2005 6:45 PM
sorcerer_blue_M_elf.mdf	1 KB	MDF File	6/05/2005 6:45 PM
sorcerer_blue_M_elf.tga	49 KB	TGA File	6/05/2005 6:45 PM
sorcerer_blue_M_halforc.mdf	1 KB	MDF File	6/05/2005 6:45 PM
sorcerer_blue_M_halforc.tga	49 KB	TGA File	6/05/2005 6:45 PM
sorcerer_blue_M_human.mdf	1 KB	MDF File	6/05/2005 6:45 PM
sorcerer_blue_M_human.tga	49 KB	TGA File	6/05/2005 6:45 PM
sorcerer_green_F_dwarf.mdf	1 KB	MDF File	6/05/2005 6:45 PM
sorcerer_green_F_dwarf.tga	49 KB	TGA File	6/05/2005 6:45 PM

Each new .tga has its own mdf file that points to it, and each race has its own .tga. While the elf model is the same as the human one (just with paler skin and thus its own .tgas), half-orcs and dwarves (that should be *dwarfs*, thank you Mr Tolkien) have their own models, thus their own .skms. We have a LOT of files to hack...

Lets rename them. I rename every blue one 'nred' at the relevant bit (keeping in mind the

game still gets the relevant blue ones, for the original cyan-coloured garment, from the .dats). I don't rename the .tgas - they will be replaced by the ones H_K has done, and I rename those appropriately. Here's the new human red male next to the old pinkish one:

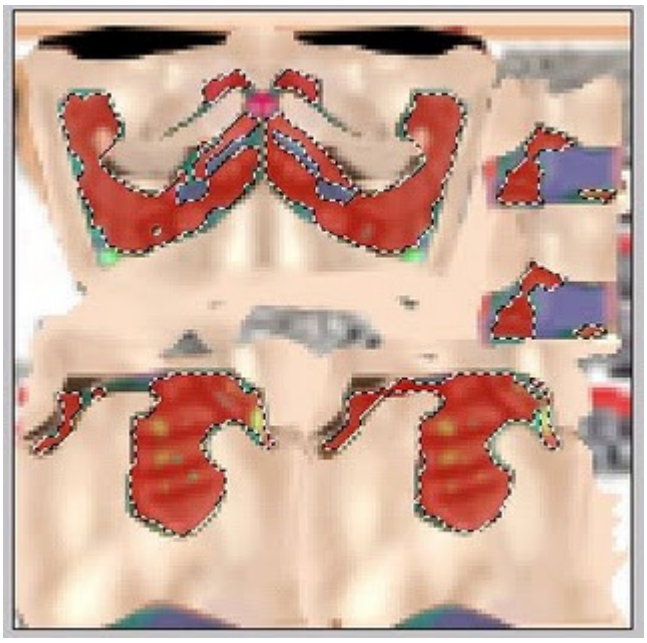
Not a lot of difference; its mainly in the female ones. Where are they? I don't see any new red ones: H_K did say he hadn't done both sexes yet, so lets do that one ourselves. ;-))
Popping the 4 female ones (sorcerer_nred_F_human.tga, sorcerer_nred_F_halforc.tga, sorcerer_nred_F_elf.tga and sorcerer_nred_F_dwarf.tga) we see they look like this:

while the old pinkish ones look like this:

Ewwww... and they call that red. Lets fix it. As well as the 4 tgas, there is a 5th one simply called sorcerer_nred_ground and a matching .mdf file: we'll modify that too.

Look at the above pic. We see cyan with some blue bits. I will change them to bright red with gold highlights. I select the nred_ground.tga and all the green-cyan bits with magic wand. Set tolerance at 15: it doesn't try to grab the stencil-blue parts. Hold shift, for those who don't use magic wand, selects new sections while keeping the old ones selected. It looks like this:

I then go Enhance / Adjust Colour / Colour Variations and I press the 'increase red' button SEVEN times. Thats important, because I have to do this with 5 different .tgas and want to remember what I did with each one. That gives me something like this:

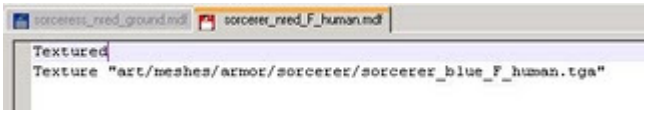


Well thats something, isn't it? I mean, its red, not pink 8^P

Next I select the blue highlight parts, and do much the same, only in the enhancing bit I change them to 'decrease blue', which makes them glow yellow. Nice :-)

Next, I labouriously do the other 4 .tgas for the 4 racial models. And thats the basic artwork done. We now have male and female versions.

Time to see them in game? Not quite. We may have renamed the new files with variations on 'nred', but internally they are still pointing at the old ones, so we have to crack open the .skm and .mdf files and hack them. I will use Notepad++ this time for the .mdf file, since it makes it that little bit easier (I will still use the hex one for the .skm since either way you get a screen full of gibberish and a few intellible words here and there). Here's what we see in sorcerer_nred_F_human.mdf:



So we change the bit saying

art/meshes/armor/sorcerer/sorcerer_blue_F_human.tga

to

art/meshes/armor/sorcerer/sorcerer_nred_F_human.tga

Now, listen carefully, this is the most important thing I can tell you. The rest of it is just renaming files and using common sense - here is where hours of frustrated hacking by previous modders comes into its own.

One of the reasons I opened these files in a Hex Editor is so you can remember the game is handling files in hex format, not text like a .mes file. So you can't just remove a byte or add one - that shifts everything one byte to the right or left and bang, the whole thing is ruined. You have to copy byte by byte.

Therefore, if we are going to point a file to a new file called **sorcerer_nred_F_human.tga**, we have to use an existing file that can be altered to say that IN EXACTLY THE SAME NUMBER OF BYTES. In this case, that means the same number of letters. Hence we have NOT copied all the files called sorcerer_red_F_human and simply introduced an 'n' - that would be adding a byte. Rather, we have changed the ones called sorcerer_blue_F_human (and variations) to sorcerer_nred_F_human.

Think about it carefully. Then, open a Hex editor and try it. As you type over the letters, watch how the numbers change. We want all the other numbers to stay the same! I mean they will, but to get our new file reference to work we have to change all the numbers in the relevant part of the old one, and there is no room to add new ones, so we can't 'introduce' an 'n' to 'nred' but we *can* type 'n-r-e-d' over the top of 'b-l-u-e'.

I hope thats clear. If not, I'm afraid you'll be going for a walk in CTD land.

Ok, so I go through and use the Notepad++ to rename all the male and female .mdf files, and then I use my Hex Editor to do sorcerer_blue_ground.skm: these model files sometimes take some searching through to find the relevant bits but you have to do it. If you later run the game and the object you have made looks wrong, or looks like a mish-mash of something you made and something else, or just plain looks like the original file you were modding instead of the new file you made, then chances are this is the problem: you forgot something in the .skm. This is more of a problem with monster meshes, though, and in this particular case, the mystic garb, there is only one entry in the skm to change.

So now our new files look at each other in their innards as well as their outtards: can we run it now and have a look? Absolutely not! We have only done half the prototype!

Going back to protos.tab, just keep moving along. Do the weight, cost and inventory icon the same as 6211, and the flags too (cols 50 and 61). Also go over to col 79, “Max Dexterity”, and set that to 100 (the basic setting for any item of clothing or wearable item that shouldn’t affect the Dexterity bonus). We now have one more column to do: 62, ‘colour’ in ProtoEd. I am not sure what ToEEWB calls it, but I can tell you now, it refers to our aforementioned materials.mes. 6211 has an entry of 9000, so go to data/rules and open materials.mes (in Notepad).

Entry 9000 onwards looks something like this:

// Sorcerer Outblue Fit blue

// 9000-9099 sorcerer outblue_Fit blue
{9000}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_M_human.mdf}
{9001}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_F_human.mdf}
{9002}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_M_elf.mdf}

{9003}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_F_elf.mdf}
{9004}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_M_halforc.mdf}
{9005}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_F_halforc.mdf}
{9006}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_M_dwarf.mdf}
{9007}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_F_dwarf.mdf}
{9008}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_M_dwarf.mdf}
{9009}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_F_dwarf.mdf}
{9010}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_M_human.mdf}
{9011}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_F_human.mdf}
{9012}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_M_human.mdf}
{9013}{CHEST:art\meshes\armor\sorcerer\sorcerer_blue_F_human.mdf}

This does two things: it tells the game where to apply these new meshes – to the chest of the PC / NPC model where it replaces the existing one (we'll prove that in a moment) – and it also formally tells the game which of the many .tgas (which are individually named in the .mdf files, remember) to apply for the male and female of each race. You might think, "well that's obvious, 'blue_F_Dwarf" is for the female dwarf model!" but remember, we could name them anything at all (as long as it was the right number of bytes, and we cross-referenced it the same way internally). Its like naming a new weapon "Longsword +3" – you have to still put the +3 bonus and the weapon type in protos.tab to make it what it sounds like, you can't just call it that in description.mes and leave it at that.

Take a look at the size of that materials.mes file, btw – if someone (probably Cujo) ever did add a new race like Drow or Snervfleblin, then someone else (probably me or Gaear ;-) would have to go through and add an extra male and female line to EVERY damn entry here. Even if it just said blue_M_human / blue_F_human all over again, as it does here for many of them (since the Halfling, Half-Elf and Gnome races are just human models somewhat shrunken). If you want a whole new model, then that's all new meshes and mdfs for EVERY garment. Who wants to do that? Any takers?

Enough of that. We have to now make a new bunch of entries for all the variations. I am

putting them at 12900, since there are big gaps among the temple Robe entries for some reasons (they should be differentiated in the hundreds but among the robes they do it in the thousands, leaving 9 hundreds to play with if you take my meaning). Once I have renamed the blue entries to 'nred' it looks like this:

// remeshed and new mystic garb

// 12900-12999 sorcerer proper red
{12900}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_M_human.mdf}
{12901}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_F_human.mdf}
{12902}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_M_elf.mdf}
{12903}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_F_elf.mdf}
{12904}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_M_halforc.mdf}
{12905}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_F_halforc.mdf}
{12906}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_M_dwarf.mdf}
{12907}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_F_dwarf.mdf}
{12908}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_M_dwarf.mdf}
{12909}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_F_dwarf.mdf}
{12910}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_M_human.mdf}
{12911}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_F_human.mdf}
{12912}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_M_human.mdf}
{12913}{CHEST:art\meshes\armor\sorcerer\sorcerer_nred_F_human.mdf}

Easy, and didn't take long thanks to copy-paste. We then place the entry 12900 back in col 61 of protos.tab.

One last thing: we now open another file in the Rules folder: addmesh.mes. It has identical numbering to materials.mes, so looking again at 9000 (where the original blue mystic garb, proto #6211, kept its racial mesh information in materials.mes) we see the following:

// 9000-9099 sorcerer blue oufit blue

{9000}{}
{9001}{art\meshes\armor\sorcerer\sorcerer_blue_F_human_addm.SKM}
{9002}{}
{9003}{art\meshes\armor\sorcerer\sorcerer_blue_F_human_addm.SKM}
{9004}{}
{9005}{art\meshes\armor\sorcerer\sorcerer_blue_F_Halforc_addm.SKM}
{9006}{}
{9007}{art\meshes\armor\sorcerer\sorcerer_blue_F_dwarf_addm.SKM}
{9008}{}
{9009}{art\meshes\armor\sorcerer\sorcerer_blue_F_human_addm.SKM}
{9010}{}
{9011}{art\meshes\armor\sorcerer\sorcerer_blue_F_human_addm.SKM}
{9012}{}
{9013}{art\meshes\armor\sorcerer\sorcerer_blue_F_human_addm.SKM}

Its easier to see what addmesh does than explain it, so for now we'll just rename these to match our newly named files, and also put it at 12900:

// 12900-12999 sorcerer proper red

{12900}{}
{12901}{art\meshes\armor\sorcerer\sorcerer_nred_F_human_addm.SKM}
{12902}{}
{12903}{art\meshes\armor\sorcerer\sorcerer_nred_F_human_addm.SKM}
{12904}{}
{12905}{art\meshes\armor\sorcerer\sorcerer_nred_F_Halforc_addm.SKM}
{12906}{}
{12907}{art\meshes\armor\sorcerer\sorcerer_nred_F_dwarf_addm.SKM}
{12908}{}
{12909}{art\meshes\armor\sorcerer\sorcerer_nred_F_human_addm.SKM}

{12910}{}
{12911}{art\meshes\armor\sorcerer\sorcerer_nred_F_human_addm.SKM}
{12912}{}
{12913}{art\meshes\armor\sorcerer\sorcerer_nred_F_human_addm.SKM}

Ok, lets power it up! I run the game, clear the map cache (to be sure), it loads – good – then I load a save and 'give 6340' to my female human rogue. It doesn't CTD – woohooo!! I then click on it to get her to wear it, and see this:

Ooops! There's the red-and-gold outfit, but the bit and the shoulder-pads are blue-green!

Those are the addmesh right there: I have altered their names and pointed the addmesh.mes at them, but I didn't do the addmesh.skm file and they are still pointing at the old files.

I left this error in not only because I actually made it, but because it nicely demonstrates what an addmesh is. We saw in materials.mes that the mesh that our new mesh replaces is the chest mesh on the PC / NPC model: and it does replace it, not just cover it, because it includes the skin element and if you leave it out, or alpha channel it into transparency, you won't see something underneath (this is an ongoing problem with trying to make things that allow other things to show, such as tabards).

An addmesh, on the other hand, does not simply provide a new mesh for a part of the existing model, but adds a new little modelled bit on top. Hence while the existing chest mesh can essentially be replaced with a new body-hugging garb, there are no shoulder-pads on the human model to re-mesh, these have to be provided afresh by an added mesh (hence Addmesh. But you figured that out 6 paragraphs back, I dare say).

Anyways, addmeshes go in addmesh.mes at the same place as the remeshing in materials.mes. Some things have both, some have only one or the other: for instance, male mystic garb has no addmesh, its just repainted trousers and tattoos (all for the chest) while helmets are all addmesh and nothing else, so in materials.mes we see this for the helmets bit:

// 4000-4007 AsianHat
// 4100-4199 Barbarian Helmet
// 4200-4299 Chain Coif
// 4300-4399 CircletNice
// 4400-4499 CircletSimple
// 4500-4599 Generic Helm
// 4600-4699 Great Helm
// 4700-4799 Plumed Helm
// 4800-4899 Sorceror Helm
// 4900-4999 Wizard hat
// 5000-5099 Leather Cap

While we see something similar for addmesh.mes at the areas for gloves and boots

// 5700-5799 Hedrack Full boots
// 5800-5899 Hedrack Full gloves
// 7900-7999 Fullplate2 Gloves
// 8000-8099 Fullplate2 Boots

They’re just placeholder remarks, so you don’t make your brand new pride and joy – a

multicoloured helmet with glowing horns - hack it into materials.mes at what looks like a free spot, then go mad wondering why it comes out with a skirt because there was an addmesh already at that spot that you knew nothing about and hadn't checked for. Make use of those remarks. Here are some more:

//21100-21199 Cuchulainn
//21200-21299 Cuchulainn
//21300-21399 Cuchulainn
//21400-21499 Cuchulainn
//21500-21599 Cuchulainn
//21600-21699 Cuchulainn
//21700-21799 Cuchulainn
//21800-21899 Cuchulainn
//21900-21999 Cuchulainn

That'll save name-calling for later! Lets fix our addmesh. Its sorceress_addm_red.mdf: open and hack it to point at the nred file. Thats it. Let's run the game again and see if it works.

And there's our finished product! Simple, eh? No? Contrary to what it looks like, it really is just copying files, making a few changes here and there to match them all up, and having a nice tga to introduce something worth doing it all for. Not hard, just tedious, and once you have done a few, not even that.

To be thorough I will introduce one more thing that Half Knight has done for these new garments: new icons. Probably most people who read this will already know how to add new icons but since I don't think I have ever mentioned it in a tutorial, I will say something now.

Icons, as we saw in protos.tab earlier, is at col 53, which is called Inventory Icon in ProtoEd: spot on. We put in the number 242 based on 6211: lets give them all individual icons. 242 is not just an unfulfilled UN resolution, its also a number in inventory.mes (don't you love how easy these files are to remember? At least they don't call it something weird or in latin or something – *graphica_minor.mes*) which is found in data\art\interface\inventory, not tooo far from the meshes folder. Also in the folder, as well as the .mes file, are all the inventory .tgas: too easy. Lets look inside the .mes file:

{00239} {mandolin_icon.tga}
{00240} {bone_helm_icon.tga}
{00241} {bone_armor_icon.tga}
{00242} {sorcerer_outfit_blue.tga}
{00243} {sorcerer_outfit_green.tga}
{00244} {sorcerer_outfit_red.tga}
{00245} {Familiar_Bat.tga}
{00246} {Familiar_Cat.tga}
{00247} {Familiar_Frog.tga}

There it is! Heading further down, we see there is a gap at 573-579 where some clothes will fit nicely, but I have to add a bunch of hats and a helmet and there are already hats and helmets in that bit (568-70 and 580-85) so I will put those in there and move this stuff back to before 455 (which has the worn and grubby boots – I am adding some new boots Half Knight made too, so I will put the mystic garb and boots there). So lets just put it at 444 onwards:

{00444} {sorcerer_outfit_nred.tga}
{00445} {sorcerer_outfit_nblue.tga}
{00446} {sorcerer_outfit_white.tga}

{00447} {sorcerer_outfit_black.tga}

Looking again, Half Knight has only done the later two (black and white) since he intended blue and red to be replacements for the current pink and cyan ones, not new ones: I had other ideas ;-) So I go back into ToEE1.dat, find the blue icon, 5 shots of enhance / colour variations, and I have a nice red icon. Lets see it all together:

Well I have no idea where that damn left-click led us - something bizarre in longdescription.mes: some left over from ToEE, perhaps. But here you can see the icon, the new garb and its stats in protos.tab (such as it is - very little stat-wise for simple clothing) all in one pic. Finished!

Maps, Areas and Quests

Today's tutorial addresses some issues with maps: which maps belong to which game.areas, which map has which quests associated with it, that sort of thing.

GAME AREAS

The following little example from dear old Zert will demonstrate what 'game area' refers to.

{120}{Yes, lad?}{Yes, lass?}{}{120}{}{}
{121}{I think you should leave the group.}{}{8}{npc.area != 1 and npc.area != 3}{130}{}
{122}{You leave group!}{}{-7}{npc.area != 1 and npc.area != 3}{130}{}
{123}{I think you should leave the group.}{}{8}{npc.area == 1 or npc.area == 3}{140}{}
{124}{You leave group!}{}{-7}{npc.area == 1 or npc.area == 3}{140}{}

{130}{What? Leave here? No, lad, that won't do. I'll only go my way in a village.}{What? Leave here? No, lass, that won't do. I'll only go my way in a village.}{}{130}{}{}
{131}{[exit]}{}{1}{}{0}{}

{140}{Okay lad, I'll take my leave. Watch your back, though. It's a dangerous world.}{Okay lass, I'll take my leave. Watch your back, though. It's a dangerous world.}{}{140}{}{pc.follower_remove(npc)}
{141}{E:}{}{1}{}{0}{}

Soooooo... Zert can tell whether or not he is in a village based on the game area: 1 or 3 are apparently villages. 1 of course is Hommlet, so naturally 3 is Nulb.

There are in fact some 15 different game areas, corresponding to places on the worldmap. Lets consider the worldmap for a moment: here are the names you will see displayed on the worldmap, as recorded in data / mes / worldmap_location_names_text.mes:

{010} {Emridy Meadows} ; WORLDMAP_MEADOWS_LOC
{020} {Moat house} ; WORLDMAP_MOATHOUSE_LOC
{030} {Moat house cave exit} ; WORLDMAP_MOATHOUSE_CAVE_LOC
{040} {Nulb} ; WORLDMAP_NULB_LOC
{050} {Imeryds Run} ; WORLDMAP_NULB_RIVER_POOL_LOC
{060} {Cave Lair} ; WORLDMAP_CAVELAIR_LOC

{070} {Temple of Elemental Evil} ; WORLDMAP_TEMPLE_LOC
{080} {Burnt Farmhouse} ; WORLDMAP_TEMPLE_WELL_LOC
{085} {Deklo Grove} ; WORLDMAP_DEKLO_GROVE_LOC
{090} {Hommlet - North} ; WORLDMAP_HOMMLET_LOC_NORTH
{100} {Hommlet - South} ; WORLDMAP_HOMMLET_LOC_SOUTH
{110} {Hommlet - East} ; WORLDMAP_HOMMLET_LOC_EAST
{120} {Temple Ruined House} ; WORLDMAP_TEMPLE_HOUSE_LOC
{130} {Temple Broken Tower} ; WORLDMAP_TEMPLE_TOWER_LOC
{140} {Verbobonc} ; WORLDMAP_VERBOBONC_LOC
{150} {Hickory}
{160} {Placeholder}
{170} {Placeholder}
{180} {Placeholder}
{190} {Placeholder}
{200} {Placeholder}

This is how it looks since Co8 had a hack at it and added Verbobonc, Hickory Branch and a bunch of placeholders for future expansion (clever Spellslinger). These are the little name boxes that appear when you hover over a spot on the worldmap, and which appear in the list on the right. But what about the numbers of the game areas themselves? They look something like this:

1 _____ Hommlet
2 _____ Moathouse
3 _____ Nulb
4 _____ The Temple
5 _____ Emridy Meadows
6 _____ Imeryds Run
7 _____ Temple Well
8 _____ Moathouse Back Door
9 _____ Ogre Cave

10 ____ Deklo Grove
11 ____ Temple Tower
12 ____ Temple House

Thats only 12: I thought we would get up to 14 on the Worldmap when we remember that Hommlet has 3 Worldmap locations, but 14 is Verbobonc. Whats 13? It took me forever to figure out that in the list above (that has the 2 Hommlet maps) the Deklo Grove comes in at 85, so there is in fact 16 names there: the 12 areas, Verbobonc and Hickory, and the 2 extra Hommlet worldmap spots. Heh. The fact Deklo Grove seems to have been sorta jemmyed in there may explain why there is no 13 - I can't think what else there is.

So how do we use this connection? Well, if we want to have a new area on the worldmap open up to the players, we activate the pertinent game area. For instance, lets have a look at Spugnoir, who will mark both Emridy Meadows and the Moathouse on your map. (Note this starts mid-conversation with the PC asking about locations)

{33}{Where is this moathouse?}{}{8}{game.areas[2] == 0}{70}{game.areas[2] = 1}
{34}{Me want to go to this moaty place.}{}{-7}{game.areas[2] == 0}{70}{game.areas[2] = 1}
{37}{Do you know where Emridy Meadows are?}{}{8}{game.areas[5] == 0}{130}{}
{38}{Where be Emridy Meadows?}{}{-7}{game.areas[5] == 0}{130}{}

{70}{Here, I will show you where the moathouse is located on your map.}{Here, I will show you where the moathouse is located on your map.}{}{70}{}{}
{75}{Thank you. I will head there now. [travel immediately to the moat house]}{}{8}{}{0}{game.worldmap_travel_by_dialog(2)}

{130}{Here, I will show you where Emridy Meadows is located on your map.}{Here, I will show you where Emridy Meadows is located on your map.}{}{130}{}{game.areas[5] = 1}
{135}{Thank you. I will head there now. [travel immediately to Emridy Meadows]}{}{8}{}{0}{game.worldmap_travel_by_dialog(5)}

Notice a couple of things there, compared to Zert's:

- the command **game.worldmap_travel_by_dialog(#)** where # corresponds to the game area
- that you check for whether a PC does or doesn't know the area (ie already have it available on their map) using the comparatives **game.areas[#] == 0** and **game.areas[#] == 1**
- that you set the game area as known (ie mark it on the map) with **game.areas[#] = 1**
- that you check whether you are *in* this or that game area with **npc.area == #** or **attachee.area == #** (as we saw Zert do above). I can tell you from tragic experience, **game.area == 1** does NOT work!

Another thing to mention, though you will have to take my word on this, is that if you make your way to an area some other way, it will still automatically mark on your map. For instance, the 3 'satellite' maps to the Temple - the broken tower, the ruined house and the well (which is your back door when coming out from the area where Smigmal is) can all be accessed map-to-map by just going through doors / ladders, but will then show up on your worldmap. Note also that as with the 3 locations in Hommlet, there are no 'red dot' paths between these areas, or random encounters: so if you are making a new module with a new worldmap, you place these next to each other.

So, lets say you use **game.worldmap_travel_by_dialog(5)** to get to Emridy Meadows - where do you turn up? Which is to say, why turn up at that exact spot? In the past we have seen teleport commands such as **game.fade_and_teleport(0,0,0,5008,465,481)** which will teleport you to the top of the Welcome Wench, to the exact coordinates scripted. But where does **game.worldmap_travel_by_dialog(5)** (or clicking on the worldmap in general) get its coordinates from?

The answer to that is found way away from the various map names, in the modules/rules folder. Inside you will find the maplist.mes file. It contains all the map names, plus their starting coordinates and whether they are to be regarded as outdoors or not, or whether they are to be fogged. I'll give you a few examples: go look at this file for yourself, its a very important one.

{5066}{Map12-temple-dungeon-level-1, 548, 589, Flag: DAYNIGHT_XFER}
{5067}{Map13-dungeon-level-02, 488, 569, Flag: DAYNIGHT_XFER}
{5068}{Map-Area-6-Imeryds-Run, 478, 483, Flag: DAYNIGHT_XFER, Flag: OUTDOOR}
{5069}{Map-Area-10-Decklo-Grove, 485, 487, Flag: DAYNIGHT_XFER, Flag: OUTDOOR}
{5070}{Random-Scrub, 480, 480, Flag: OUTDOOR}
{5071}{Random-Forest, 480, 480, Flag: OUTDOOR, Flag: UNFOGGED}
{5072}{Random-Swamp, 480, 480, Flag: OUTDOOR, Flag: UNFOGGED}
{5073}{Random-Riverside, 490, 475, Flag: OUTDOOR, Flag: UNFOGGED}

Note how the maps are named. Those are not idle descriptions, like those for the worldmap names we saw above that turn up on your worldmap (which could be just as easily changed), or the map names in map_names.mes that are at the side of your worldmap and could likewise be changed quite easily and have no impact on the game. The map names in the maplist.mes are the names of the folders that the game looks for when it goes to that map. So when you execute the aforementioned

game.fade_and_teleport(0,0,0,5008,465,481)

it is to this file that the game looks, to line 5008, to see which folder to look at in the modules / Co8 5 / maps folder. Then, inside that are the mobs, the sectoring, and the map.php which in turn points to the background map number etc. About the only thing that doesn't come from this folder is the townmaps - they are arranged by map number (5001ff) no matter what the map is called or what folder is used for it.

To answer the previous question, this is also where the game gets coordinates from if you just do a **game.worldmap_travel_by_dialog(#)** command to this or that area, or if you add a new area to the Worldmap like Verbobonc or HB: those co-ordinates in the maplist.mes map reference are where the game looks (and can be changed accordingly). Otherwise, if you go via a script or a door, then you get the coordinates from jumppoint.tab or written into the script or whatever (and they over-ride these ones, of course).

We've probably been through all that before. What is worth mentioning in this context is that some maps CALL themselves Area 9 or Area 12 or whatever - well, those ARE just names, an area doesn't become 'Area 6' because you name the folder that, its controlled in the .dll which has already assigned specific areas to the maps. So our next question is, which maps correspond to which areas? Which is to say, if you add a new map (such as Frank's house) will it still be in area 1?

After much slavish effort, here are all the maps in ToEE and their areas.

{5000}{}

Area 1 {5001}{Hommlet}
Area 2 {5002}{The Ruins of the Moathouse}
Area 2 {5003}{The Moathouse Tower}
Area 2 {5004}{Upper Level of the Moathouse Ruins}
Area 2 {5005}{Dungeon Level of the Moathouse Ruins}
Area 1 {5006}{Inn - Cellar}
Area 1 {5007}{Inn - First Floor}
Area 1 {5008}{Inn - Upper Floor}
Area 1 {5009}{Trader's Barn}
Area 1 {5010}{Trader's Store}
Area 1 {5011}{Church of St. Cuthbert - Upper Floor}
Area 1 {5012}{Church of St. Cuthbert - Main Floor}
Area 1 {5013}{Church of St. Cuthbert - Lower Floor}
Area 1 {5014}{Guard Tower - Cellar}
Area 1 {5015}{Guard Tower - Lower Level}
Area 1 {5016}{Guard Tower - Main Hall}
Area 1 {5017}{Guard Tower - Parapet Level}
Area 1 {5018}{Guard Tower - Upper Hall}
Area 1 {5019}{Guard Tower - Parapet Interior}
Area 1 {5020}{Prosperous Farm Cottage}

Area 1 {5021}{Prosperous Farm Cottage - Upstairs}
Area 1 {5022}{Woodcutter's Cottage}
Area 1 {5023}{Well-Kept Farm}
Area 1 {5024}{Well-Kept Farm - Upstairs}
Area 1 {5025}{Prosperous Farmhouse}
Area 1 {5026}{Leatherworker's House}
Area 1 {5027}{Blacksmith's Shed}
Area 1 {5028}{Weaver's House}
Area 1 {5029}{Weaver's House - Upstairs}
Area 1 {5030}{Tailor's Cottage}
Area 1 {5031}{Average Farm Building}
Area 1 {5032}{Weatherbeaten Building}
Area 1 {5033}{Moneychanger's Establishment}
Area 1 {5034}{Small House}
Area 1 {5035}{Small House - Upstairs}
Area 1 {5036}{Potter's Cottage}
Area 1 {5037}{Brewhouse}
Area 1 {5038}{Modest Cottage}
Area 1 {5039}{Cheesemaker's Cottage}
Area 1 {5040}{Mill}
Area 1 {5041}{Reclusive Cottage}
Area 1 {5042}{The Grove}
Area 1 {5043}{Herdsman's Barn}
Area 1 {5044}{Wheel and Wainwright's Shop}
Area 1 {5045}{Walled Manor House}
Area 1 {5046}{Walled Manor House - Upstairs}
Area 1 {5047}{Carpenter's House}
Area 1 {5048}{Town Hall}
Area 1 {5049}{Stone House}
Area 0 {5050}{MileStone 10 Map}
Area 3 {5051}{Nulb}

Area 3 {5052}{Boatmans' Tavern and Nulb Market}
Area 3 {5053}{Mother Screng's Herb Shop}
Area 3 {5054}{Mother Screng's Herb Shop - Second Floor}
Area 3 {5055}{Mother Screng's Herb Shop - Third Floor}
Area 3 {5056}{Smithy}
Area 3 {5057}{Snake Pit - Main Floor}
Area 3 {5058}{Snake Pit - Second Floor}
Area 3 {5059}{Snake Pit - Top Floor}
Area 3 {5060}{The Waterside Hostel}
Area 3 {5061}{The Waterside Hostel - Upper Floor}
Area 4 {5062}{The Temple of Elemental Evil}
Area 1 {5063}{Modest Farmhouse}
Area 4 {5064}{The Temple of Elemental Evil - Main Floor}
Area 4 {5065}{Tower Ruins}
Area 4 {5066}{Temple - Dungeon Level One}
Area 4 {5067}{Temple - Dungeon Level Two}
Area 6 {5068}{Imeryds Run}
Area 10 {5069}{Deklo Grove}
Area 0 {5070}{} // Random
Area 0 {5071}{} // Random
Area 0 {5072}{} // Random
Area 0 {5073}{} // Random
Area 0 {5074}{} // Random
Area 0 {5075}{} // Random
Area 0 {5076}{} // Random
Area 0 {5077}{} // Random
Area 4 {5078}{Slaughtered Caravan}
Area 4 {5079}{Temple - Locked Level}
Area 4 {5080}{Temple - Dungeon Level Four}
Area 4 {5081}{Air Node}
Area 4 {5082}{Earth Node}

Area 4 {5083}{Fire Node}
Area 4 {5084}{Water Node}
Area 3 {5085}{Nulb Building}
Area 3 {5086}{Nulb Building - Second Floor}
Area 3 {5087}{Nulb Building - Third Floor}
Area 3 {5088}{Nulb Building - Fourth Floor}
Area 0 {5089}{} // Moathouse Demo
CTD {5090}{} // Menagerie Demo
Area 8 {5091}{Moathouse Hidden Exit}
Area 7 {5092}{Temple - Escape Tunnel}
Area 7 {5093}{Ramshackle Farm}
Area 5 {5094}{Emridy Meadows}
Area 9 {5095}{Ogre Cave}
Area 1 {5096}{} // Opening Vignette
Area 1 {5097}{} // Opening Vignette
Area 1 {5098}{} // Opening Vignette
Area 1 {5099}{} // Opening Vignette
Area 1 {5100}{} // Opening Vignette
Area 1 {5101}{} // Opening Vignette
Area 1 {5102}{} // Opening Vignette
Area 1 {5103}{} // Opening Vignette
Area 1 {5104}{} // Opening Vignette
Area 0 {5105}{} // Temple Screenshot
Area 4 {5106}{Circular Shaft}
Area 0 {5107}{} // Temple Demo
Area 0 {5108}{} // Temple Demo
Area 0 {5109}{} // Air Node
Area 0 {5110}{} // Demo Hommlet
Area 0 {5111}{} // clipping test
Area 11 {5112}{Temple House}
Area 12 {5113}{Temple Tower}

Area 9 {5114}{Ogre Cave Interior}
Area 1 {5115}{Herdsman House}
Area 0 {5116}{} // Tutorial map 1
Area 0 {5117}{} // Tutorial map 2
Area 0 {5118}{} // Tutorial map 3
Area 0 {5119}{} // Arena of Heroes
Area 0 {5120}{Frank's House}
Area 14 {5121}{Verbobonc}
Area 0 {5122}{Lord Viscount's Office}
Area 0 {5123}{Caravan}
Area 0 {5124}{Inn - First Floor}
Area 0 {5125}{Inn - Second Floor}
Area 0 {5126}{Gnome Living Quarters}
Area 0 {5127}{Temple Antechamber}
Area 0 {5128}{Temple}
Area 0 {5129}{Drow Tunnel}
Area 0 {5130}{Dragon Cave}
Area 0 {5131}{Dragon Cave Exit}
Area 0 {5132}{Lord Viscount's House - First Floor}
Area 0 {5133}{Lord Viscount's House - Second Floor}
Area 0 {5134}{Chapel}
Area 0 {5135}{Church of Hextor}
Area 0 {5136}{Bazaar of the Bizarre}
Area 0 {5137}{Gnarley Forest}
Area 15 {5138}{Hickory Branch}
Area 0 {5139}{Hickory Branch Crypt}
Area 0 {5140}{Temple - Dungeon Level Three}

It won't come as a shock to anyone that the new bits we have added - Frank's House, the Caravan, Verbobonc, Huckleberry Branch etc - all come in as Area 0: the game wasn't designed to finesse those places so just gives them a zero. But look at the Verbobonc and HB

main maps - they are areas 14 and 15! That comes from Spellslinger's efforts to hack them into the .dll so they could be accessed from the World Map.

Next are the quests.

QUESTS BY AREA

You will have noticed there are different tabs in your logbook's quest page for different areas: for Hommlet, Nulb and the Temple. As you get quests, they show up in relevant parts of the logbook accordingly: quests received in Hommlet turn up under the Hommlet tab, quests received in Nulb... well, you get the picture. Those names are from logbook_ui_quests_text.mes, which looks like this:

--- logbook_ui_main_tab_window text---
{11} {Hommlet & Verbobonc}
{12} {Nulb}
{13} {Temple}

--- text above acquired window ---
{15} {Quest Location}

--- text above detail window ---
{20} {Quest Information}

--- label text in detail window ---
{30} {Status:}

--- text used in the acquired text buttons
{31} {Month: }
{32} {Day: }

--- Text of quest states ---
{40} {Mentioned}
{50} {Accepted}
{60} {Completed}
{70} {Botched}

--- No Quest text ---
{100}{No Quests Available}

The canny will already have figured out that area 0 quests default to Hommlet, which is why all the Verbo-mod ones turn up there, and Allyx then added '& Verbobonc' after attempts to get a 4th tab going didn't work.

In KotB, the logbook tabs look like this:

--- logbook_ui_main_tab_window text---
{11} {Keep}
{12} {Caves}
{13} {Wilderness}

The Caves of Chaos ravine will be based on the Nulb worldmap location (game area 3), and the Caves themselves will be the various houses and places of Nulb. Everything else is the wilderness, where-in you actually get very few quests atm: just the bonus quest, actually :-).

Of course, all this means we can change all the other stuff in the logbook if we wanted. We could change 'botched' to 'screwed up', or 'completed' to 'you go girl!' But I should like to think we won't.

Now, precisely which 'quest locations' tabs are associated with which map areas? Here they are:

0 _____ Hommlet
1 _____ Hommlet
2 _____ Hommlet
3 _____ Nulb
4 _____ The Temple
5 _____ Hommlet
6 _____ Hommlet
7 _____ Hommlet
8 _____ Hommlet
9 _____ Hommlet
10 ____ Hommlet
11 ____ Hommlet
12 ____ Hommlet
14 ____ Hommlet
15 ____ Hommlet

Which is a fancy way of saying that everything that isn't set to Area 3 (Nulb quests) or Area 4 (Temple quests) are set for the Hommlet tab: they default there, same as area 0. Note that the other areas *around* the Temple on the World Map - the Well, Burnt Farmhouse and Tower - will NOT show their quests as being for the Temple (not that they have any quests - indeed, only the Tower has any mobs, they are tragically empty maps) because they have seperate game areas to put them on the World Map: interesting side effect, as it were.

Also note that it has nothing to do with the quest number: if you consoled the Temple Hight Priests into the middle of Hommlet and got all the Temple quests off them there, they would appear as Hommlet quests. Likewise if you had a quest from a recruitable NPC (and we probably should have a few of those) then any quest they gave you would appear in the logbook depending on where you happened to be when they gave it to you. Hope that makes sense - I think I've laboured the point enough that it should ;-) But just to prove it, here are some screenies where I ended up with the Nulb and Temple quests in the first tab because I consoled them on other maps:


Quest Location
Hommlet & Verbobonc
Nub
Temple
Cupid's Arrow
Day: 15 Month: 4 15:12
Flirting With Disaster
Day: 15 Month: 4 15:12
Agent Revealed
Day: 15 Month: 4 15:12
Romag 3: Assassinate Belsornig
Day: 15 Month: 4 15:12
Grud's Fish Story
Day: 15 Month: 4 15:12
Belsornig 1: Oohlgrist For Water
Day: 15 Month: 4 15:13
Belsornig 2: Blame Game
Day: 15 Month: 4 15:14
Belsornig 3: The Setup
Day: 15 Month: 4 15:14
Quest Location
Hommlet & Verbobonc
Nub
Woodcutter's Problem
Day: 15 Month: 4 15:10
Quest Location
Hommlet & Verbobonc
Nub
Black Jay's Dead Sheep
Day: 15 Month: 4 15:11