

Compiled by **krunch**.

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

**Basic dialogue writing tutorial part 1**

**Basic dialogue writing tutorial part 2**

**Basic dialogue writing tutorial part 3**

**Basic dialogue writing tutorial part 4**

**New PC Voices Tutorial**

**Custom soundtrack tutorial**

**Adding sound and visual FX and spawning from dialogue**

**Proactive NPCs**

**Adding NPC followers**

**Transferring items**

**Adding chatter to NPC (unfinished)**

**Creating new NPCs with ToEEWB**

**Sectoring Internal Doors**

**Common Mistakes and Common Scripts**

**Adding movies**

**Complete(ish) Sectoring Tutorial**

**Flags Tutorial Part 1**

**Flags Tutorial Part 2**

**ToEEWB Tips and Traps**

**Factions, Reputations, and Reactions**

**Animations Part 1**

## 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 **00291tenant.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 let's 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 comfortable 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 associated 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](http://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 language, 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 ones, 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 separate 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 be 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.questions[#].state** (so if you are using quest number 180, that'll be **game.questions[180].state**). So, to change the state of a quest it would be:

```
game.questions[#].state = qs_mentioned  
game.questions[#].state = qs_accepted  
game.questions[#].state = qs_completed  
game.questions[#].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 Tutorial

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 this 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 MEIfRgr.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 dekle 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 knowledgeable 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 are .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 separately 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).

Ok, so go through and do that to all the lines. Dull, init?

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 (u will d/l mine won't u???)

## 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

Go and get the files **here** if you haven't already and we will look at them. There is 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 than 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 effect) 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, 00097 etc).

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

## Custom Soundtrack Tutorial

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.

Soooooooo...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 NPCs

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 has 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** Taki 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 yelllll 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 separately. 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, pursuant 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 conditional 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 attempt 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 transferring 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, because 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 tower - 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 jump point 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 transferring, 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 #,recipient)**

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 transferring, 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_DEFAULT
```

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,trigge
rer)
_____.elif
(obj.stat_level_get(stat_level_cleric) >= 1):
_____.attachee.turn_towards(obj)
_____.attachee.float_line(925,trigge
rer)
_____.elif
(obj.stat_level_get(stat_level_barbarian) >= 1):
_____.attachee.turn_towards(obj)
_____.attachee.float_line(950,trigge
rer)
_____.elif
(obj.stat_level_get(stat_level_monk) >= 1):
_____.attachee.turn_towards(obj)
_____.attachee.float_line(935,trigge
rer)
_____.elif
(obj.stat_level_get(stat_level_ranger) >= 1):
_____.attachee.turn_towards(obj)
_____.attachee.float_line(970,trigge
rer)
_____.elif (obj.stat_level_get(stat_level_druid)
>= 1):
_____.attachee.turn_towards(obj)
_____.attachee.float_line(920,trigge
rer)
_____.elif
(obj.stat_level_get(stat_level_rogue) >= 1):
```

```
_____attachee.turn_towards(obj)
_____attachee.float_line(955,trigger)
_____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 even 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. Soooooo, 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, becuae 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()`

Really, that's about it. 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! Its not that hard to make a simple

NPC, is it. 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.

Now... 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; Shortsphear; 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.

Now, if you don't use Toffee, while u have the maps folder open, 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 Modding Mistakes

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 a common, frustrating screw-up.

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.requests[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.requests[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 because 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 constitutes '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 you have already defined one of the objects in a script, then you use something abstract like 'triggerer'. Or, forgetting to put args in at all. Lemme show you 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.requests[].state = qs_accepted
game.requests[].state = qs_completed
game.requests[].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 // Groomsh
```

```
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_paladin) >= 1
pc.stat_level_get(stat_level_ranger) >= 1
pc.stat_level_get(stat_level_druid) >= 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 embedded, 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!

## Movie Tutorial

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.

## Sectoring Tutorial (mostly complete)

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 Jumpoint.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 accessible 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 background 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. Sooooooooooooo, 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:

# Ted's RPG Rant – Modding ToEE Tutorial

<http://rpg-rant.blogspot.com/2005/11/tutorial-list.html>

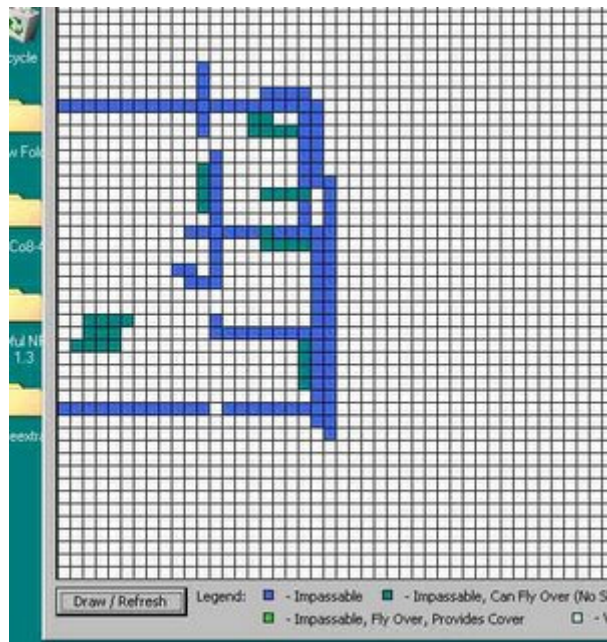
By: ShiningTed

Page 96 of 185

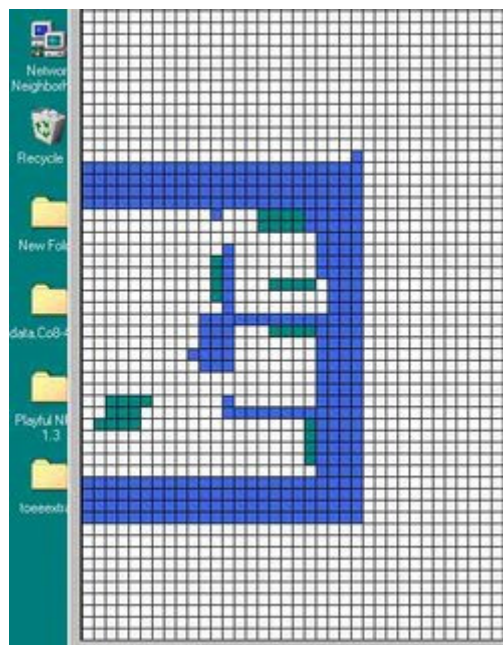


Now we can have a look at what we have done in ToEEWB. The files have automatically been writing themselves in the **/sectors** folder. To see what you have done, click on the Sectors tab in ToEEWB (second from left), then open - there should be two new sectors in there. Open the first one, then click *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()
```

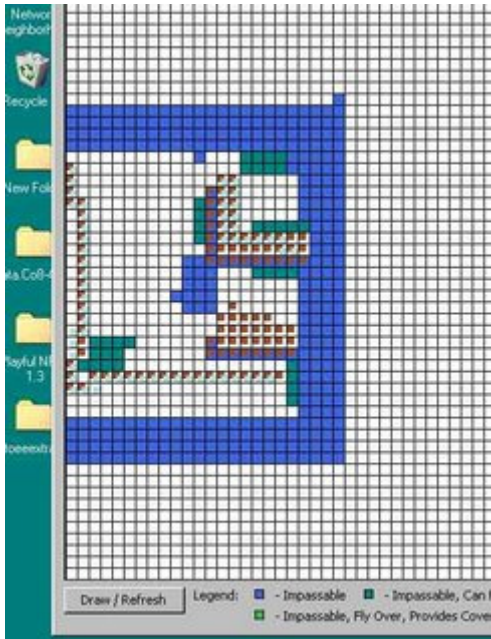
and run around behind the wall, everywhere. When you are finished, do `wtr_off()`. To do individual tiles, use `wtr()`. Get out and open the file in ToEEWB. Now, (make sure ur ToEEWB is up to date, only the most recent versions support this!!!) where it says

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 hja ve 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

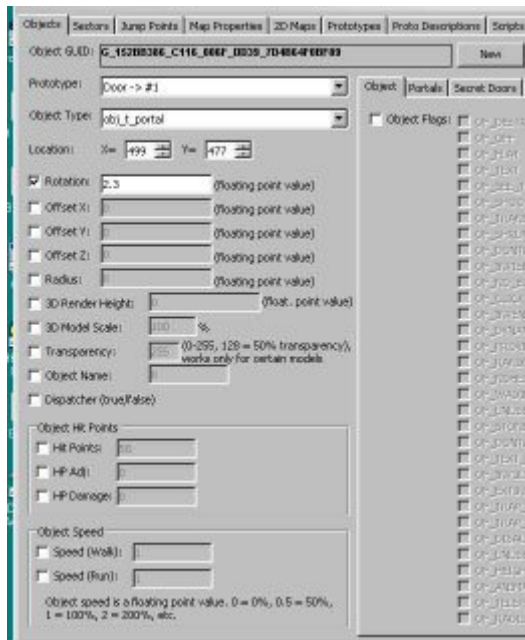
ed.py activated of course) type:

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.



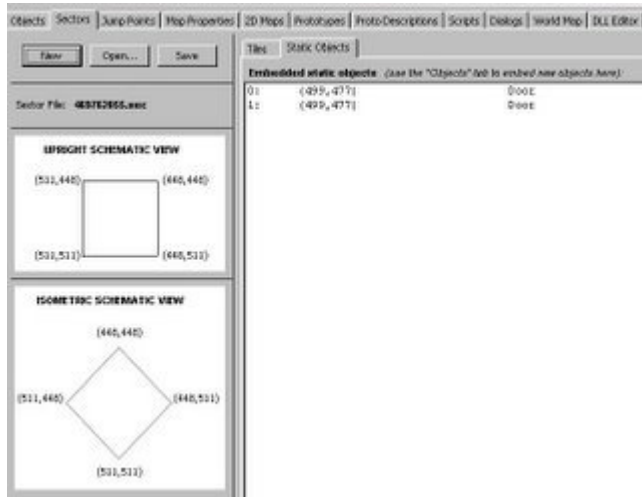
Oops... got the rotation WAY wrong... must've been thinking of something else entirely. Lemme see, where have I used a perpendicular door to this? The Guild building? Can't remember... pity, because Screeg's maps tend to be the same camera-angle, so once u have fiddled around getting the rotation perfect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



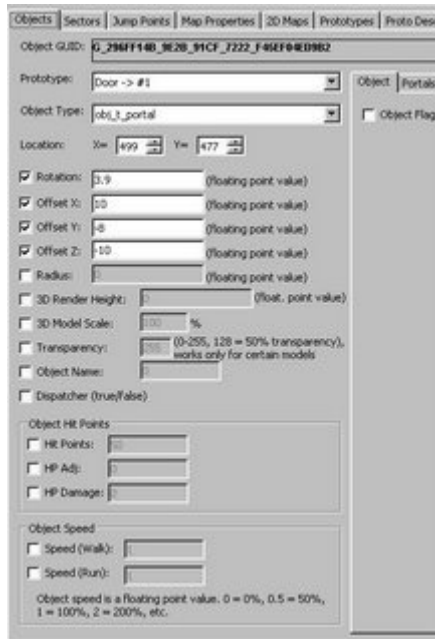
Ooo nice! Before we see if it needs any minor rotational adjusting, lets get it in the

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.



Ooooooooooooo, 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 nowadays. 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 overridden 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 previously 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. (That's 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\_flagS\_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, that's 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: anyone 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 :-)

Sooooooooo... 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 flagS!!)
```

- 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 see 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 Lughash / 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 spawnny thing. **game.global\_vars[52]** was put in the san\_dying thing of Lareth's lieutenant and seargent, 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 *suggested* 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). That's why 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 existence 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:

Tiles: Static Objects

Tile: X= 448 Y= 448 Load Tile ☐ Auto load tile whenever X or Y are changed

Material Type / Footstep Sound: 02 - Dirt ← This is the footsteps tab

Tile Flags:

- ☐ BLOCKS (obsolete, don't use)
- ☐ SINKS (obsolete, don't use)
- ☐ BLOCKS, CAN FLY OVER (obsolete)
- ☐ ICY (does nothing?)
- ☐ NATURAL (not a man-made tile)
- ☐ SOUNDPROOF (deadens sounds)
- ☐ INDOOR
- ☐ REFLECTIVE (does nothing?)
- ☐ BLOCKS VISION
- ☐ FLYOVER/COVER (in this tile, semi-tiles that are marked as 'fly over' provide cover)

**NOTE:**

Sector Visibility Blocking (SVB) related flags are marked with an asterisk (\*) symbol. The rule for SVB is: visibility 'extends' (flag 1) out until it hits an 'end' (flag 2). 'Archways' (flag 4) are possibly special kinds of extends that operate in all directions, rather than just up the upper right corner like the way extends do. I'm not really sure what 'base' (flag 3) does.

So if you had a wall that separated a room in the LL (lower left) from a room in the UR (upper right), you'd lay extenders up until the top of the wall, and lay ends at the top. So visibility would uncover up the wall if you're in the room to the LL, but stop at the top of the wall and not uncover even if you were standing in the extenders if you're in the UR room.

Semi-tiles (1/9th of a tile each):

<input type="checkbox"/> Blocks	<input type="checkbox"/> Blocks	<input type="checkbox"/> Blocks
<input type="checkbox"/> Fly Over	<input type="checkbox"/> Fly Over	<input type="checkbox"/> Fly Over
<input type="checkbox"/> Extend(*)	<input type="checkbox"/> Extend(*)	<input type="checkbox"/> Extend(*)
<input type="checkbox"/> End(*)	<input type="checkbox"/> End(*)	<input type="checkbox"/> End(*)
<input type="checkbox"/> Base(*)	<input type="checkbox"/> Base(*)	<input type="checkbox"/> Base(*)
<input type="checkbox"/> Archway*	<input type="checkbox"/> Archway*	<input type="checkbox"/> Archway*
<input type="checkbox"/> Wtr 0	<input type="checkbox"/> Wtr 0	<input type="checkbox"/> Wtr 0

Update the Tile This operation will update a single tile with information above

Update all tiles in the sector ← This is the update all tiles button This operation will update all the tiles defined by this sector

Update a box of tiles Update from X= 448 Y= 448 to X= 448 Y= 448

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.

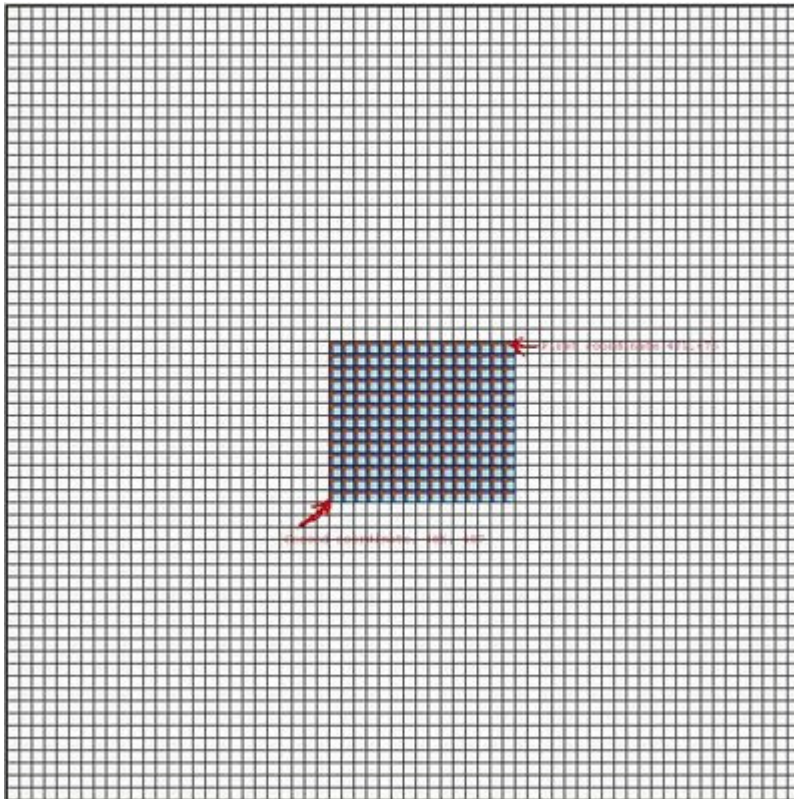
## Tip 2: Getting the damn 'Update a box' thing to work

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 foliage 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 are embedded 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 nowadays.

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 concerned, but you will see the painted background (and hovering text thing) so you are, to all intents and purposes, 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 guards (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 separate 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 think 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 forceful, 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 accessed 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.

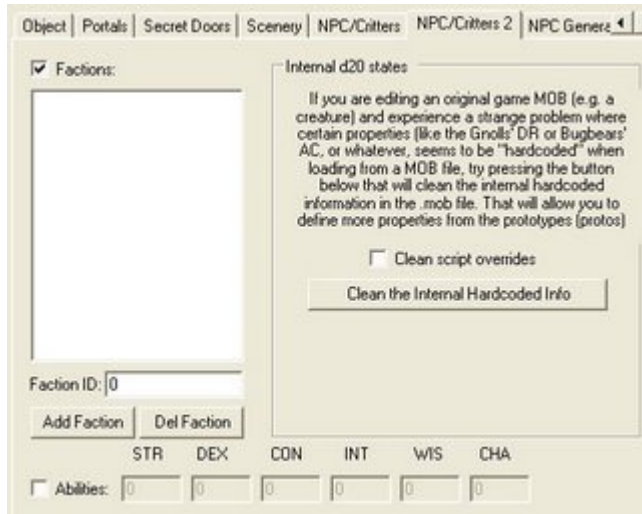
Line #	Col 151	NPC Flags	Col 153	Col 154	Col 155	Col 156	CREXP	Base Refle
10143		ONF_KOSD					3	-2
10144				10				
10145				10				
10146				10				
10147				10				
10148				8				
10149				10 12 9				
10150				10				
10151				10 14				
10152				10 14				
10153				4				
10154				4				
10155				5				
10156				4				
10157				8				
10158		ONF_KOS		3			2	4



Objects	Sectors	Jump Points	Map Properties	2D Maps	Prototypes	Proto Descriptions
Prototype: Nub village woman -> #14373						ID: 14373
						Go to Description
To Hit Bonus						
Number of Attacks	2					
Creature Damage Dice						
Creature Damage Type						
To Hit Bonus						
Number of Attacks	3					
Creature Damage Dice						
Creature Damage Type						
To Hit Bonus						
Critter Hair Color						Light Brown
Critter Hair Type						Longhair (m/f)
PC Flags						
PC Name						
NPC Flags (OHF_*)						
NPC AI Data						
NPC Factions						10
NPC Retail Price Multiplier						
NPC Reaction Base						
NPC Challenge Rating						

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. That's 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}{Hommllet 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 NPC's 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)**. O well, its done now and it works :-)

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 monsters 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, today's 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!). It's 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 you can have the guy stand there for a predetermined amount of time, facing any given direction, if you so desire. Very handy, and not just for patrols - other people move around, you know! Just don't forget to hit the check-box for each if you're 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

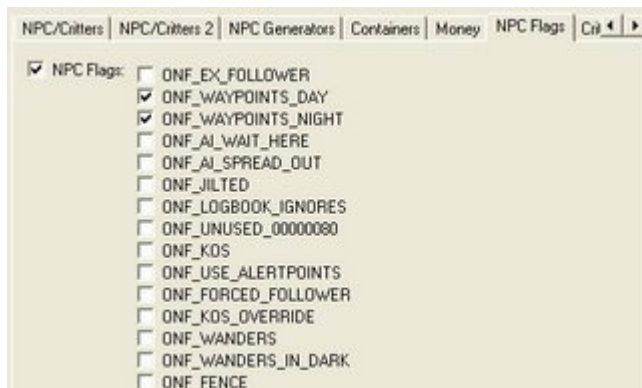
1.5 - east

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.

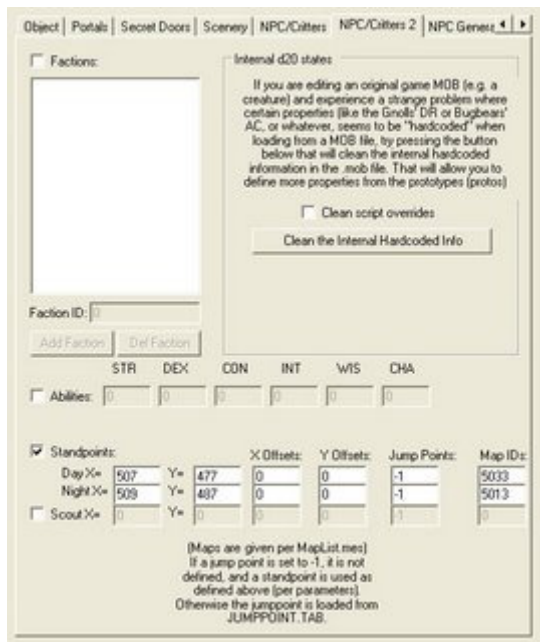
Note that afaik there is no way to get SEPERATE waypoints happening for day and night. Pity, because of course if u r gonna have the day / night transition u r gonna have people doing different things at those times. But at least we can switch one of them off, have the NPC wander around his fields by 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.

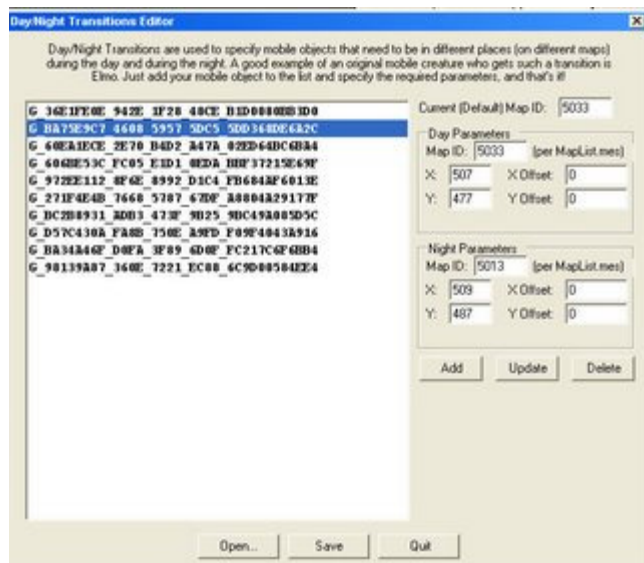
This is important, because moving a merchant across maps BREAKS THE LINK TO HIS INVENTORY BOX (Blue discovered this iirc and I have certainly had it happen to me). Otoh simply moving him around on the same map doesn't cause this (u might have to keep him

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! Although 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

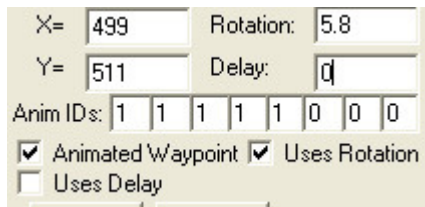
if u desperately need to (this folder only has him and Elmo (for the drunk stagger) in it, so that'll tell u how many peculiar animations there are for NPCs - not many!!) Here's what it says, though it is irrelevant for our purposes.

**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:



The screenshot shows a configuration window with the following fields and options:

- X= 499
- Rotation: 5.8
- Y= 511
- Delay: 0
- Anim IDs: 1 1 1 1 1 0 0 0
- ☒ Animated Waypoint
- ☒ Uses Rotation
- ☐ Uses Delay

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 (animations 3, 4, 6, 7, 8 or so)

A bunch of shield bashes (the later ones, animations 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 transferred 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 (especially 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 existence (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 :-) ).