The Pixie QVar Tutorial (V1.0)

... Including Mission Objectives

This tutorial assumes you can set properties on objects, and can set up ControlDevice (CD) links between objects. This is covered in most basic tutorials (such as Komag's) if you are unsure. I would recommend printing out this document, then playing the mission through DromEd (though most rooms do have a poster on the wall telling you want is happening). The mission is essentially a series of rooms, each has a number, referenced in the text.

Contents

Why QVars?

A QVar (Quest Var or Quest Variable) is a game parameter that holds a number and can be changed and checked in-game. QVars are useful for all sorts of things, such as keeping count of something (like the number of secrets founds, or AIs killed), handling combination locks and timers. They are also used to handle objectives.

You cannot compare QVars to each other (you cannot check if this QVar is greater than that QVar, for example) and you cannot assign one from another (you cannot set this QVar equal to that QVar), unless you use custom scripts (see the Random Combinations section for an illustration of that). As far as I am aware, you cannot directly set other parameters from QVars, for example the speed of a Tweq or the position of an object. However, I imagine that all these things can be done with custom scripts if you can get someone to write them for you.

Creating, changing and deleting in DromEd

You can create QVars in DromEd using the *quest_create_mis* command:
quest_create_mis <qvar>, <value>
... where <qvar> is the name, and <value> is the initial value. For example:

```
quest_create_mis Counter, 50
```

You can also use the quest_create_mis command to edit it:
quest_create_mis <qvar>, <newvalue>
For example:

```
quest_create_mis Counter, 42
```

You can delete it with the quest_delete c:
quest_delete <qvar>
For example:

```
quest_delete Counter
```

Once created, you can check it by clicking *Mission Quest Data* on the Editor menu. Double clicking on the name will let you edit the value. If you try to change the name, you will get a new QVar with the new name, in addition to the original, with the old name.

QVars are also created automatically by DromEd when you set them up in objects and traps, using *Add > Trap > Quest Var*. Be aware that DromEd may assign an initial value after you first go in-game; always check it is what you were expecting. It usually defaults to 0.

QVars are case insensitive. Thief considers counter to be the same as Counter or COUNTER.

Pre-defined QVars

There are also numerous QVars that are set up in game to track loot, pockets picked, etc. Some of these you can use yourself, but others only get set at the end of the mission (eg, total_loot), so are not much use to the mission designer. Probably the most important of these QVars are map_max_page and map_min_page. See appendix 2 for a list of many more.

Also, there are some campaign QVars (all start DrSCm) that keep totals across the entire campaign. Most (but not all) of these can be seen if you go to Editors – Campaign Quest Data. It is possible to add to these through a custom script, I believe, that that is outside the scope of this tutorial.

QuestVarTraps

A QuestVarTrap (QVarTrap) can be used to change the value of a QVar in-game. This is done through the Trap --> Quest Var property. The new value is set in the format:
<op><value>:<QVar>
So, if you want the new value to be 20 the Quest Var would be set with =20:Counter

QVarTraps respond to both TurnOn and TurnOff messages.

Op	*Turn on action*	*Turn off action*
=	Set questvar equal to <argument>	Set questvar equal to zero
!	Set all of the bits of questvar which are set in <argument> [= QVar | value]	Unset all of the bits of questvar which are set in <argument> [= QVar & ~value]
+	Add <argument> to questvar	Subtract <argument> from questvar
-	Subtract <argument> from questvar	Add <argument> to questvar
*	Multiply questvar by <argument>	Divide questvar by <argument>
/	Divide questvar by <argument>	Multiply questvar by <argument>
%	Find remainder of questvar ÷ <argument>	Multiply questvar by <argument>
{	Shift left questvar by argument (essentially, multiply questvar by 2^<argument>)	Shift right questvar by argument (divide questvar by 2^<argument>)
}	Shift right questvar by <argument>	Shift left questvar by <argument>
?	Add a random amount from 0 to <argument>	Subtract a random amount from 0 to <argument>
d (T2)	Add a random amount from 1 to <argument>	Subtract a random amount from 1 to <argument>
" (T2)	Multiply by 10 and add <argument> mod 10 (so if QVar is 93, 45:QVar will change it to 935)	Divide by 10
| (T2)	As !	As !

A QVarTrap set up with =45:QVar will set QVar to 45 when a TurnOn is received, and set it to 0 with a TurnOff. A QVarTrap set up with +10:QVar will add 10 to the current value of QVar every time a TurnOn is received, and subtract 10 every time it gets a TurnOff.

QuestVarTriggers

You can react to the value in game time with a QuestVarTrigger (QVarTrigger). These are set up in exactly the same way as a QVarTrap, and will issue a single TurnOn when the calculation becomes true, and a single TurnOff when it becomes false. They are all evaluated at the start of the mission, when every QVarTrigger will issue a TurnOn or a TurnOff depending on the starting conditions.

Op	*Sends a TurnOn*	*Sends a TurnOff*
=	When the QVar becomes equal to the value	When the QVar becomes not equal to the value
<	When the QVar becomes greater than the value	When the QVar becomes less than or equal to the value
>	When the QVar becomes less than the value	When the QVar becomes greater than or equal to the value
"	When the QVar ends with these digits (which can start with a zero)	When the QVar no longer ends with these digits (T2 only)
&	When a bitwise AND operation results in a non-zero number	When a bitwise AND operation results in 0 (T2 only)

Note that only QVarTrigger and QVarTrap should have the *Trap > Quest Var* property set up as op<value>:<qvar>; other objects should just have the name of the QVar in there. If you put =1234:Combo in the *Trap > Quest Var* property for a odometer, for instance, DromEd will take all that as the name of a QVar. And

```
quest_delete =1234:Combo
```

... will be rejected by the command parser, so you could be stuck with it.

The " operator for both the QVarTrap and the QVarTrigger is a little mysterious, until you realise it is used for combination locks... Which are discussed later.

Objectives as QVars

There are a whole host of QVars that handle objectives, giving you a wide range of ways to set them up. They are created, edited and deleted just like any other QVar, but all have the form *goal_<variety>_n*, where n is the goal ID. There are 18 different goal varieties (summarised in Appendix 3, but discussed below) and you can

have up to 32 objectives, numbered from 0 to 31 (it maybe Thief 1 only that is limited to 32).

Numbering Objectives

Objectives have to number from 0 (the official ConVict documentation seems to be wrong on this), and there must be no gaps. If there are gaps, everything after the gap will not be treated as an objective. If you use objective IDs 0, 1, 2, 3, 5, and 6 then DromEd will start at 0 looking for goal_state_0, and will keep going up until there is no goal_state_4, and stop. Objectives 5 and 6 will not get treated as objectives. The bare minimum you require to ensure no gaps is goal_state_n. However, that will not actually do anything the player will actually see…

Objective Types

There are fives types of objectives, as set by the *goal_type_n* command. These are explored in room 6. A type 0 objective must be set though a QVarTrap. Objectives default to type 0, but it may be a good idea to set it anyway to remind yourself why it is set up like that.

```
quest_create_mis goal_state_0, 0
quest_create_mis goal_visible_0, 1
quest_create_mis goal_type_0, 0
```

In the example mission, opening the far door (with a TrigDoorOpen script on it) completes this objective.

Stealing an object is a type 1 objective. This objective requires an addition parameter, set through a QVar called *goal_target_n*, which must be set to the ID of the object.

```
quest_create_mis goal_state_1, 0
quest_create_mis goal_visible_1, 1
quest_create_mis goal_type_1, 1
quest_create_mis goal_target_1, 108
```

If you are using custom objects, be careful what you use as a base object; pick something that is as close in its properties to the new object as you can. For an item for an objective, pick something from QuestItems in the object hierarchy.

The objective to kill an AI is type 2, and again requires the *goal_target_n* QVar to set the object number of the AI.

```
quest_create_mis goal_state_2, 0
quest_create_mis goal_visible_2, 1
quest_create_mis goal_type_2, 2
quest_create_mis goal_target_2, 183
```

You can use the number of the archetype in the hierarchy (a negative number), to require the player to kill any Beast (-1327), any Mechanist (-1256) or any Apebeast (-1328), for example.

Stealing a certain amount of loot is type 3, and this requires a different parameter. In fact you have a choice of four, depending on whether you are considering the total loot, the gold, gems or goods (which is also known as art in other places). In the example mission, this is gold (because I use goods later on), but usually it is the total loot.

```
quest_create_mis goal_state_3, 0
quest_create_mis goal_visible_3, 1
quest_create_mis goal_type_3, 3
quest_create_mis goal_gold_3, 100
```

Finally there is the go to location objective, which again uses the *goal_target_n* QVar.

```
quest_create_mis goal_state_4, 0
quest_create_mis goal_visible_4, 1
quest_create_mis goal_type_4, 4
quest_create_mis goal_target_4, 111
```

This requires a "concrete" room brush. To make a room brush concrete, select the room brush, then click on Create; a dialog box will appear. Click add, and type in an appropriate name (I used Objective4); click Okay. Select your new name from the list, and click Create. Your room brush will now have a name, and a number in brackets after it. Do Build Room Database to get it all working. Any number of rooms can be the same "concrete" room brush; the player can then go into any one to complete the objective. I do not know why (and

may be it is just me), but it seems very easy to mess this up. If it is not working properly, check through all the room brushes and make sure the correct ones have got proper names and all the rest say "Default Room".

Objective Modifiers 1: No Kill

There are three objective modifiers; these change the way objectives behave. The first is *goal_reverse_n*, which reverses the sense of the goal. This is most useful with type 2 objectives; kill an AI becomes do not kill an AI.

```
quest_create_mis goal_state_5, 0
quest_create_mis goal_visible_5, 1
quest_create_mis goal_type_5, 2
quest_create_mis goal_target_5, -1325
quest_create_mis goal_reverse_5, 1
```

In the above example, rather than stating a specific AI, I have used the object number for the Bystander archetype from the object hierarchy. This covers all the bystander AIs in one go. The objective goes to failed if you kill an innocent in room 7, but it goes to completed if you complete all the other objectives (the visible ones, excluding optional and reverse objectives) first.

However, in the example mission, the player is not allowed to blackjack the innocents either, which is a bit more complicated. It is done as a QVar type objective. The TrigBrainDead script (Thief 2 only) is on each relevant AI, and there is a CD link from all of them to a QVarTrap, which is set as =3:goal_state_5.

```
quest_create_mis goal_state_5, 0
quest_create_mis goal_visible_5, 1
quest_create_mis goal_type_5, 0
quest_create_mis goal_reverse_5, 1
```

When the AI is killed or KOed, the TrigBrainDead script sends a TurnOn message to the QVar, which sets the objective to failed. Note that this is still set up as a reverse objective, even though nothing is reversed. This means that the mission can be completed, without this objective being completed, and when that happens, this objective will get set to completed.

You can test this by doing all the required objectives, and going back to the start room. If the "Get back to where you started" objectives completes, so will the "Do not kill or KO any innocents".

If you want the mission to fail if the AIs become alerted, you will have to create a blue room button to send the TurnOn message to the QVar. Each AI then needs an Alert Response to frob the button (or you set it once for the human archetype, or whatever).

Any objective that says the player must not do something (such as alert anyone) should be set up as a reverse objective.

Objective Modifiers 2

The other two modifiers are readily confused. An objective with *goal_irreversible_n* set to 1 can never go back to being uncompleted once it has been done. This is useful for items if the item can be dropped (most questy items are set up so they cannot be dropped anyway), or for a location that the player has to visit, but can then leave. In room 6, the objective to go to the far room was irreversible. Otherwise, it would go to complete when you enter the room, and go to incomplete when you leave it. The Hand Of Glory cannot be dropped, so there was no need in that case.

```
quest_create_mis goal_state_4, 0
quest_create_mis goal_visible_4, 1
quest_create_mis goal_type_4, 4
quest_create_mis goal_target_4, 111
quest_create_mis goal_irreversible_4, 1
```

Use *goal_irreversible_n* for a room the player has to visit, and then leave.

Objective Modifiers 3: Return to location

An objective with *goal_final_n* set to 1 can only be completed if the player does the required action when all the other required objectives (but not those with *goal_reverse_n* or *goal_optional_n* set to 1, or or are not visible) have been completed already. If the objective is to get an item, then all other objectives must be completed before the item is picked up; if the item is already picked up, and then the other objectives are completed, that is no good, you would have to drop the item and pick it up again.

This modifier is usually used for the "Get back to where you started" objective, and the example mission is no different.

```
quest_create_mis goal_state_6, 0
quest_create_mis goal_visible_6, 1
quest_create_mis goal_type_6, 4
quest_create_mis goal_target_6, 90
quest_create_mis goal_final_6, 1
```

Use *goal_final_n* for a room the player has to get to after doing everything else.

Objective Modes: Bonus, Optional, Etc.

There are five modes for objectives. A normal objective – set with *goal_visible_n* – is visible at the start, and has to be completed. An optional objective – set with *goal_optional_n* – is visible at the start, but is not required to complete the mission. A bonus objective – set with *goal_bonus_n* – is not visible at the start, and is not required to complete the mission; it becomes visible when completed (getting the drill bit in Kidnap is an example of a bonus objective). An invisible objective is not visible at the start, but becomes visible later in the mission (there are invisible objectives in Framed and Blackmail). There is one of each of these in room 5, going clockwise round the room. Finally, there is an unseen objective, which the player never sees, and is exemplified by the scroll.

The normal objective was set up as before:

```
quest_create_mis goal_state_7, 0
quest_create_mis goal_visible_7, 1
quest_create_mis goal_type_7, 1
quest_create_mis goal_target_7, 81
```

The optional object has one extra bit:

```
quest_create_mis goal_state_8, 0
quest_create_mis goal_visible_8, 1
quest_create_mis goal_type_8, 1
quest_create_mis goal_target_8, 73
quest_create_mis goal_optional_8, 1
```

You can check whether this is optional by completing the other objectives, and seeing if the return home objective (which is set to final) will complete.

For the bonus objective you have to set *goal_bonus_n* to 1, and miss out the *goal_visible_n*:

```
quest_create_mis goal_state_9, 0
quest_create_mis goal_visible_9, 0
quest_create_mis goal_type_9, 1
quest_create_mis goal_target_9, 83
quest_create_mis goal_bonus_9, 1
```

The *goal_visible_9* is not required, if DromEd finds an objective with *goal_state_n*, but cannot find *goal_visible_n*, it will create *goal_visible_n* itself, and give it a value of 0. This is the only objective QVar that DromEd will create for you.

You do not need any extra commands to make your objective invisible, the objectives default to being invisible. However, you will need a QVarTrap to make the goal visible in-game. This would be set up with =1:goal_visible_n (in the example mission, press the button behind the Cultivator to trigger this). Bear in mind that the player does not have to complete this objective while it is still invisible – if he can complete the other objectives without this appearing the mission will complete.

```
quest_create_mis goal_state_10, 0
quest_create_mis goal_type_10, 1
quest_create_mis goal_target_10, 82
```

There are numerous ways you might want to trigger an invisible object, rather than pressing a button, such as an AI dying, reading a book, entering a room, opening a door or taking an object. Remember that if the QVar receives a TurnOff message, the objective will disappear again, so make sure the object cannot be dropped, or whatever.

The unseen objective is something of a hack. It has nothing to do with objectives at all, but occasionally it is useful to tap into the objective subsystem to do some trickery. If you want to trigger an event when the player

has a certain amount of loot – use an objective. The objective's *goal_state_n* will go to 1 when the amount of loot is achieved, even if the player never sees it; this can be used to trigger some event.

I have chosen a different example. If you take the scroll from the niche in Room 5, the small iron door to the left will open. Read the scroll, and the right door will open. This is not easy to set up; CD link both doors to the scroll, and they will both open at the same time. One solution would be to add and remove links when the scroll is picked up. But an alternative is to have an unseen objective to steal the scroll. When the player takes the scroll, the objective's *goal_state_n* goes to 1, which is picked up by a QVarTrigger CD linked to the left hand door.

```
quest_create_mis goal_state_15, 0
quest_create_mis goal_visible_15, 0
quest_create_mis goal_type_15, 1
quest_create_mis goal_target_15, 206
```

Special objectives: Stealing Loot

The *goal_special_n* QVar is a way to handle an objective to get a certain item of loot. The problem here is that loot items are not stored in your inventory indefinitely; when you pick up another loot item the previous one is destroyed. A steal item type objective will go to incomplete when that happens (actually, this seems not always to be the case). The way to be safe is to set the objective as a get loot objective, and to use *goal_special_n* to mark the item out.

Room 8 has a canopic jar on the right. The loot property is set to 75 art, and Special 4. The objective is set up:

```
quest_create_mis goal_state_13, 0
quest_create_mis goal_visible_13, 1
quest_create_mis goal_type_13, 3
quest_create_mis goal_loot_13, 5
quest_create_mis goal_special_13, 4
```

Because the *goal_special_13* matches the Special 4 is set on the jar, the objective completes.

That is all well and good, but what about if you want the player to steal a whole art collection - up to eight items, anyway? This requires the *goal_specials_n* QVar, and a different Special flag set on each item. In the example mission, the canopic jar on the left has Special 5 set. The objectives are set are more or less the same:

```
quest_create_mis goal_state_14, 0
quest_create_mis goal_visible_14, 1
quest_create_mis goal_type_14, 3
quest_create_mis goal_loot_14, 10
quest_create_mis goal_specials_14, 24
```

The difference is the last line; the *goal_specials_n* QVar uses a binary number to store the flags. Each Special flag has a particular value, starting with 1 for Special 1, and doubling each time, so Special 2 is 2, Special 3 is 4, and so on. The value for *goal_specials_n* is the sum of these. I am using Specials 4 and 5, which correspond to 8 and 16 respectively, a total of 24.

The player has to fulfil the loot requirement as well as the specific items to complete the objective, so set the loot requirement to no more than the value of the items.

Difficulty and Objectives

A common feature of missions is to have different objectives at different levels. This is illustrated in room eight. Objective 11 is only applicable for Normal and Hard, and requires 100 in goods. It was set up:

```
quest_create_mis goal_state_11, 0
quest_create_mis goal_visible_11, 1
quest_create_mis goal_type_11, 3
quest_create_mis goal_goods_11, 100
quest_create_mis goal_max_diff_11, 1
```

The important QVars are goal_min_diff_n and goal_max_diff_n. The first is the minimum difficulty at which the objective applies, in the case 0, or Normal. The second is the maximum, which is 1 for Hard.

Objective 12 is only applicable at expert.

```
quest_create_mis goal_state_12, 0
quest_create_mis goal_visible_12, 1
```

```
quest_create_mis goal_type_12, 3
quest_create_mis goal_goods_12, 200
quest_create_mis goal_min_diff_12, 2
```

To test these objectives, you can set the *difficulty* QVar, for example:

```
quest_create_mis difficulty, 0
```

This will set the difficulty to Normal, and you can test the objectives. This will not affect anything else you may have set up in your level (but will affect player parameters such as hit, that change with difficulty), for example, if you have objects that are destroyed on expert, that will not happen. However, the *difficulty* QVar will override the user setting when anyone plays the mission through Thief, so it is vital to remove it before shipping:

```
quest_delete difficulty
```

If you want to see properly what your mission will look like, you must use this command, after setting the difficulty QVar:

```
process_difficulty
```

This will permanently delete any objects that are set to be destroyed on the current value of *difficulty*. It is therefore very important *not to save* after doing this. I would suggest saving to some completely different name when you do so, in case after playing through the game and try to save by habit.

Delaying an Objective: Reading a Book

Just occasionally it is useful to delay when the "Objective Completed" or "New Objectives" text appears – principally when they are triggered by reading a book. With no delay, the text appears as the book text appears, and is never seen. There is a standard script to do this, TrapTimedRelay, but it is missing from some Thief installations. Instead, use TrapDelayer from mission 16. First load the script module (just one time for the whole mission):

```
script_load miss16
```

Create a TrapTrig from the object hierarchy, and add the TrapDelayer script. Set the Script --> Timing property to 1 (note that TrapDelayer uses seconds, not milliseconds like the rest of DromEd). For more details, see the South side of Room 3.

Optional "No Kill" Objectives

If you set up an objective to kill any innocent, then make it reversed and optional… It will not work. An optional ojective is optional to complete, but not to fail, so killing an innocent will always fail the mission. And if you do not kill an innocent, it will still be counted as not completed.

The way around this is to use a combination of two techniques already described in two objectives. The first is an unseen objective, used to detect something happening, rather than as a real objective. This is set up to complete if an innocent is killed (in the demo mission, I used the thief archetype, -2297, as I had used bystanders already):

```
quest_create_mis goal_state_16, 0
quest_create_mis goal_visible_16, 0
quest_create_mis goal_type_16, 2
quest_create_mis goal_target_16, -2297
```

The second objective is set up like the "Do not kill or KO any innocents" objective earlier. It is visible and reversed. Note that it is not optional (it just say that in the objectives screen).

```
quest_create_mis goal_state_17, 0
quest_create_mis goal_visible_17, 1
quest_create_mis goal_type_17, 0
quest_create_mis goal_reverse_17, 1
```

The "Do not kill or KO any innocents" was set up to go to failed if an AI was killed or KOed, but this objective is optional, so instead we will set the objective state to inactive (2). Set up a QvarTrigger to note when the first objective is completed (an innocent has been killed), with =1:goal_state_16. CD link this to a QVarTrap, which sets the second objective; =2:goal_state_17.

Initialising a Mission

Occasionally it is useful to have something triggered when the mission starts. One way to do this is with a QVarTrigger, with =0:goal_state_0 set up. As this starts at zero, and will never go back to zero (make sure this

is true of the objective you use), there is only one time that this will be true - right at the start. CD link it to a RelayTrap with NoOff set on it. You can CD link this to anything you like.

In the example mission, I have created a new QVar just for this purpose, called Initialiser. I know it will never change, so the Relay trap is not required. Right in front of you as you start, a door is set to open using this technique.

Combinations Locks

Combination locks are pretty easy to set up compared to objectives. They are composed of three parts; the buttons, the odometer and the lock. The only connection is that they share a QVar.

The Buttons

The NumberButton object can be found in the object hierarchy, under *physical – Gizmo – Switches – Buttons*. You will need ten, obviously. Each one must have the *Trap --> Quest Var* property set to the name of the QVar. Also each must have the Shape --> TxtRepl r0 property set to a texture with the right number on it. You can use obj\txt\button0 (note that NumberButton defaults to obj\txt16\button0, which has a problem with the palette, and the colours are messed up).

The buttons use the NumberButton script; when they are frobbed the QVar set in Trap --> Quest Var is multiplied by 10, has the first digit in the Shape --> TxtRepl r0 property added to it, and then seems to get truncated to 8 figures. This is the same as the " operator for QVarTrap, and when you watch what happens on the odometer, it makes sense. All the digits on the odometer get shifted to the right, to make room for the digit for the button you pressed.

You are not restricted to buttons, you could, for instance, set up a painting with the NumberButton script, the FrobInfo set to script for its world action, and set Shape --> TxtRepl r0 to be obj\txt16\any24. It does not matter that the texture does not exist, or that the painting does not use it. As 2 is the first digit in the texture name (after the second backslash), frobbing the painting will be like pressing button 2 (without the click).

The Odometer

You can find the odometer in *physical – Gizmo – Gauge*. Set the *Trap --> Quest Var* property with the name of the QVar, and that is all there is to it.

The Lock

The lock itself is handled by a QVarTrigger, CD linked to the door or whatever. Although the QVar is about 8 digits long, the odometer only shows the last four digits, so if you set up your QVarTrigger to, say, =1234:Combination, and the player types the combination in wrong, you would seem to need to reset it to zero before letting the player put the right one in. Far better to have "1234:Combination in your QVarTrigger. In a QVarTrigger, the " operator does a string match with the end of the QVar, so "1234:Combination will send a TurnOn message if the QVar called Combination ends in 1234. As long as you have a four digit digit code, the matching will be against what is displayed on the four reels of the odometer.

In the example mission, room 1 illustrates this. The keys are obvious enough, but you can also use the painting instead of button 2. The safe on the left has the combination 4567, the other two use 0123. The one on the right has a QVarTrigger set up with =0123:Combination, rather than "0123:Combination, so if you get he combination wrong it is necessary to put in some zeros to clear the old values properly.

Note that the " operator is a string match, so 0123 is treated as four digits.

I think there is something in the official documentation that states you can only have one combination lock; this is not true, as the mission shows. Also, if you look in the OM *Shipping and Receiving*, you will see a ScriptParam link to or from every odometer and associated QVarTrigger to a marker called "God". This was probably for debugging, and is entirely superfluous; you can delete "God", and it still works fine (do not ask me about the theological implications of that, though).

See appendix 4 for some suggestions on how to use combinations.

Random Combinations

It is possible to have a different random combination to a vault every time the game is played, though you do need custom scripts (see appendix 5 for how to use custom scripts). You need 2 QVars, one to hold the number the player is typing, the other to hold the actual combination.

```
quest_create_mis RandomComboTry, 1000
quest_create_mis RandomComboSet, 0
```

I used RandomComboSet for the actual combination. Using the technique described in the "Initialising a Mission" section, a QVarTrigger sends a TurnOn message to a QVarTrap when the mission starts. The QvarTrap is set up: ?8999:RandomComboSet. This adds a random number from 0 to 8999 to RandomComboSet, to give a number from 1000 to 9999; this is your combination (a combination of, say 456, would only need three figures to be matched, so the range starts at 1000).

A poster scroll to the right of the safe has the custom script NVOnscreenText, and is set not to inherit other scripts (you do not want the StdBook script trying to do stuff at the same time). The book --> text property is set up as normal for a book, with rcombo, referring to the file books\rcombo.str. This is set up (read the script documentation for further details):

```
page_0:"The combination to the safe is "
qvar:"RandomComboSet"
page_1:". Do not let anyone find it."
```

So now we have a random combination for the safe, and a way to communicate it to the player. The next step is to let the player try a combination. This is done with ten number buttons and an odometer, exactly as was done in the previous section, but using the QVar RandomComboTry.

Finally, we need to compare the actual combination, with the player's attempt. This is done with a TrapTrig object with the custom script NVTrigQVar on it (in the S --> Script property). This object will behave more-or-less like a QVarTrigger, except that you can use QVar names instead of numbers – thus "RandomComboSet:RandomComboTry instead of "1234:Combination. Note that the actual combination goes first. This is CD linked to the door.

Maps

Thief uses two QVars to handle the maps, and you can use that to change a map or add new ones as the player progresses through the mission. It is not that flexible; you can only add maps in a specific order. If you have a changing map, you can only have the one.

The example mission has a changing map. Maps can start at any number, and i have arbitrarily started at 11. My map is in the intrface\miss18 folder, and called page011.pcx, and was set up with:

```
quest_create_mis map_min_page, 11
quest_create_mis map_max_page, 11
```

That is exactly what one does for an ordinary map, but if you go to room 2, and frob the lever on the left, the map will change. The lever is connected to two QVarTraps, set up with +1:map_min_page and +1:map_max_page. After the lever is frobbed, Thief goes looking for page012.pcx (which looks the same, but with notes added to it). Frob the lever again, and it sends TurnOff messages to the QVarTraps, reversing the effect, and you are back with the original map.

Customising debrief.str

The debrief.str file is where Thief determines what to display in the Stats screen, once the mission has been completed. It can be found in the intrface folder; if it is not there copy it from res\intrface.crf, which you can open with WinZip.

If you type ":debrief" (no quotes, colon at start) in game mode whilst in DromEd, you can see what the debrief screen will look like (click on "replay" to get back to the game).

There are a number of reasons to edit your debrief.str file, the most obvious being that you do not want to report useless facts. If your FM is a one-off, the various campaign totals are rather pointless.

How Thief Handles Stats

If you open debrief.str, you will find a set of words for your language ("minute", "false", etc.) that can

probably be left as they are. After that there is the data for what will appear on the screen. Each section is made up of three parts:

```
stat_n: "MyTotal"
text_n: "Number of times: "
format_n: "%d"
```

Just like objectives, these have to number from zero, and if you miss any numbers out, no subsequent stats will be displayed. The *stat_n* indicates the QVar name (*MyTotal* in the above), *text_n* is the text that will go before the value (and should ideally end with space so the value is not flush with the writing). The format tells the software how to display the number. The *%d* indicates the value of the QVar is to be displayed as a number (as opposed to time, Boolean, etc.). Here are some further examples to give an idea of how these work:

```
format_3: "@c%d"
```

Display the QVar as a number, then continue on the same line for the next stat (often used for the "out of" bit).

```
format_8: "@+%d"
```

Display the QVar as a number, if greater than zero, otherwise skip this stat.

```
format_14: "@<03@c%d"
```

Skip this stat if the mission number is less than 3. Otherwise, display the QVar as a number, then continue on the same line for the next stat

Appendix 1 gives more details.

Setting totals for secrets and pockets to pick

One reason for editing debrief.str is because the total number of secrets or the total number of pick pockets is sometimes miscalculated by DromEd. One solution is to create new variables, say SecretsTotal and PickTotal.

```
stat_3: "DrSPocketOK"
text_3: "Pockets Picked: "
format_3: "@c%d"

stat_4: "PickTotal"
text_4: " out of "
format_4: "@c%d"

stat_17: "DrSSecrets"
text_17: "Secrets found: "
format_17: "@c%d"

stat_18: "SecretTotal"
text_18: " out of "
format_18: "%d"
```

The QVars were set up with:

```
quest_create_mis SecretTotal, 1
quest_create_mis PickTotal, 2
```

It is possible to hard code these values into debrief.str, but this way allows you to use the same debrief.str for multiple missions.

Counting Hidden Objectives and Other Things

Let us suppose there are three hidden objectives; you might want the player to know that so he will go back and play the mission again. There are also five volumes of a book to find and read.

```
stat_3: "HiddenCount"
text_3: "Hidden objectives completed:"
format_3: "@c%d"

stat_4: "HiddenTotal"
text_4: " out of "
```

```
format_4: "%d"

stat_5: "VolumesCount"
text_5: "Volumes read:"
format_5: "@c%d"

stat_6: "VolumesTotal"
text_6: " out of "
format_6: "%d"
```

I have a slightly different set up for the Room 5 objectives. There are four of these, and a count is kept of how many the player has completed.. These are only displayed if at least one has been completed:

```
stat_19: "Room5Count"
text_19: "Room 5 objectives: "
format_19: "@s@c%d"

stat_20: "Room5Total"
text_20: " out of "
format_20: "%d"
```

The @*c* code tells Thief to keep on the same line, so the player would see something like "Room 5 objectives: 3 out of 4". The @*s* code tells Thief not to display anything at all on the line in Room5Count is zero (if I had used @+, the player might see "out of 4" if he had failed to get any of them).

The QVars were set up with this:

```
quest_create_mis Room5Count, 4
quest_create_mis Room5Total, 4
```

Four QVarTriggers are used, one for each objective, set up to trigger when the *goal_state_n* goes to one, eg, =1:goal_state_7. These are all linked to a QVarTrap, set up +1:Room5Count. Whenever a Room 5 objective is completed *Room5Count* is incremented.

Remember that the state of every QVar is set when the game starts, and as all those *goal_state_n* QVars get set to 0, each of the QVarTriggers will send a TurnOff message to the QVarTrap. *Room5Count* starts at 4, therefore, because as the game starts it get decremented 4 times.

Mission Specific Comments

Use the @*!* code in debrief.str to restrict a comment to a single mission. Note that the format has no *%D*, so the QVar is not reported.

```
stat_21: "goal_state_0"
text_21: "Now on to mission two..."
format_21: "@!18"

stat_22: "goal_state_0"
text_22: "Now on to mission three.."
format_22: "@!19"

stat_23: "goal_state_0"
text_23: "Hurray, the world has been saved"
format_23: "@!20"
```

QVars as Timers

You can use a FlickerTrigger to send TurnOn messages to a QVarTrap at regular intervals, so the QVar tracks time. For a countdown, create the QVar:

```
quest_create_mis Countdown, 0
```

Set your FlickerTrigger to a period of 500 so it will send a TurnOn, at 500 ms, send a TurnOff, wait 500 ms, etc. CD link this to a Relay trap with NoOff set, which will allow the TurnOn messages through, but not the TurnOff (otherwise each TurnOff will make the QVarTrap add 1 to the countdown as often as it subtracts 1). CD link the relay to the QVarTrap, and set the Trap --> Quest Var property of the QVarTrap to -1:Countdown. Set up a QVarTrigger to =0:Countdown, and CD link it to whatever you want to happen when the countdown ends. You

could also put in an odometer, with the QVar set to Countdown, so the player can see how much time is left.

Go to Room 2 to see a countdown in action. Pressing the button sends a TurnOn to a QVarTrap set with =20:Countdown. There is a QVarTrigger with >0:Countdown, CD linked to a Flicker trap, CD linked to a NoOff relay, CD linked to a QVarTrap with -1:Countdown. When the button is pressed, Countdown is set to 20, which is more than 0, so the QVarTrigger sends a TurnOn message to the Flicker trap. The Flicker trap sends TurnOn messages to the final QVarTrap every second, and Countdown gets decremented. When it gets to 0, the QVarTrigger sends a TurnOff message to the Flicker trap. The QVarTrigger is also connected to the safe door, so this opens during the countdown, and closes at zero.

You can pick up the odometer in this room. This could be used to let the player keep a running score as the game proceeds, or to indicate time remaining to complete the mission (though be aware that some players object to that sort of mission). The FrobInfo for World is set to Move, and Inventory --> Type is Item. I have also set a name to indicate what the number means.

Room 4 has an event timer; press the button to the left of the window and see what happens. The button starts a Flicker trap, connected to a QVarTrap as before, but this one is counting up. Various things around the room are operated by QVarTriggers that go on and off at certain times. These are connected to Relay traps with NoOff or NoOn, which are connected in turn to the things in the room. The whole set up needs about 50 traps...

FlickerTriggers have a rate limit of 65535, and lots of them in a mission can adversely affect frame rate.

Changing the Contents of a Book

QVars can be used to change the text in books, so you can make a single book show more than one thing. The StdBook script (and StdScroll) read the Trap\Quest Var property. The value of the quest variable there is appended to the name of the book. So if you have Book-->Text set to "mybook", and the quest variable "magic_book" is 2, then the name of the book file that will be displayed is "mybook02.str".

Note that the appended number is taken as two digits (so you could have 100 different ones), and the caching seems to work so that if it fails to find foo02.str it may well assume all the others are missing too.

In the example mission, in Room 3, the relevant QVar is called BookVar. There are six number buttons, one to six, which will multiple BookVar by ten, then add their number. There is also a QVarTrigger that reacts when BookVar goes over 9, linked to a NoOff Relay, and then to a QVarTrap that takes the mod 10 of BookVar (%10:BookVar); that ensures BookVar can only have values 0 to 9, and the limited number buttons further restrict that to 1 to 6. The value is appended to book, the value in Book --> Text for the poster scroll, so the player will see the text in book01.str to book06.str.

One example use might be to let you randomise where some treasure is hidden, another use might be for a vault combination which could have several values. However, in either case, I would guess it would be easier and use less objects to have several books, each with its own text. You would require destroy traps anyway, either to destroy the unused stashes or the QVarTriggers with the wrong combination; far easier to my mind to link the destroy traps to the associated book.

Letting the Player Keep Count

Room 2 has an odometer you can pick up. This could be used to let the player keep a running score as the game proceeds, or to indicate time remaining to complete the mission (though be aware that some players object to that sort of mission), as is done here. The FrobInfo for World is set to Move, and Inventory --> Type is Item. I have also set a name and made it undroppable.

At the end of the corridor there are four more odometers. The top one counts the number of pockets picked (DrSPocketOK), the next counts objectives completed in Room 5 (Room5Count), the next is the number of secrets found (DrSSecrets). The bottom displays the value of total_loot, just in case you were thinking of using this. total_loot is not calculated until the player goes to the Stats screen, so this sticks at zero.

All these values get displayed in the debrief stats.

Miscellany

When you are setting up any trap system it can be helpful to use buttons and levers to test the system. A CD link from the book to a lever would let you test the book is set up correctly (if, say, reading the book should throw the lever), and a button CD linked to the QVarTrap will let you test the QVarTrap (frobbing the button should make the new objective visible). For QVars, set up an odometer with the QVar property set to the number of the QVar of interest, and you can see its value as it changes in game. If you want to keep these in for a while, set the difficult --> destroy property to yes on 0, 1 and 2, and when you play the game through Thief/Darkloader the testing buttons, levers and odometers will all get deleted from the game, and the player will never know. You could put an odometer in your inventory, or you could put the TrigInvFrom script on the compass and use that to trigger test messages to your trap systems.

If you have a number of QVars to set up in your mission, put all the quest_create_mis commands in a file, and run the file. It is easy to copy-and-paste each one, and change appropriately, you can check back at what you have done, and easily modify it. For this tutorial, I used a file called qvarobj.cmd. In DromEd I typed *run qvarobj.cmd* in the command box. The Thief Objectives Wizard produces a file like this, which you could use as a base. I have included qvarobj.cmd in the package; you do not have to. There is also a delete.cmd, which will remove all your objectives (up to 23 anyway) if you run it, so you can start again (incidentally, the .cmd extension seems to be optional - that is, the file does not need it, but if the file has it, you must type it).

If your QVars are not working right, think about whether there are TurnOff messages you have not accounted for. Also, it is worth checking the starting values in Editors - Mission Quest Data, as these occasionally seem to get set in game made, and the values stick (I guess this just happens the first time you use a new QVar). This happened to me with DrSPocketOK, a built-in QVar.

When trouble-shooting a hidden objective, set *goal_visible_n* to 1, and see what happens. And if objectives are not working at all, make sure the StartingPoint has the VictoryCheck scipt, and there are no missing objectives at lower numbers.

If you have a whole load of CD links to set up, you can do them all from one object; keep clicking Add, Control Device will already be selected, just change both the From and To values. Much quicker, *a little more risky*...

If DromEd assigns an object a value of 10 or less, it will probably change, which will mess up any objective to get that object. This is to do with transient objects (such as Player) that are created in game mode. Save the mission and reload it to be safe, before getting the object ID.

I discovered – eventually – that a script will not work if there is a space before or after the name, but are case insensitive.

By the way if you want back slashes and double quotes in your readables, use C-style escape characters, for backslash use \\, for double quotes use \".

Credits

A big thanks to Telliamed for many comments and suggestions for improvements (five pages of them), and to ffox for (as ever) much proof-reading, and beta-testing. Also to Nameless_voice, who wrote the scripts that allowed me to do random combinations (with help from Telliamed again).

References

Telliamed:
http://www.ttlg.com/forums/showthread.php?t=75907
http://www.ttlg.com/forums/showthread.php?t=51512
http://www.ttlg.com/forums/showthread.php?t=98605

Orkin man:
http://www.ttlg.com/forums/showthread.php?t=14197

R Soul (delete.cmd, optional no kill)
http://www.ttlg.com/forums/showthread.php?t=98097
http://www.ttlg.com/forums/showthread.php?p=1225642#post1225642

Comments and Corrections

Please e-mail the author at the_pix@hotmail.com, or PM him via TTLG.

Appendix 1 – Codes for Debrief.str (by Telliamed)

Terminal format codes determine what is displayed. Only one should be used at a time and it should be the last code in the string.
@b - boolean. Displays either "False" or "True".
@q - quantity. Displays either "None" or "Some".
@t - time. Displays "hh Hours mm Minutes nn Seconds"
%d - regular number. Really just a c-style format string. I haven't tested whether all of the options of printf will work. (Probably will.)

Codes that affect formatting are:
@n - forced newline.
@x - ??? I haven't figured out what this does, though it shows up in the standard debrief.str.
@c - continue line. Don't move to the next line after printing this stat, so the next stat will be on the same line.
@: - Displays ": "
@/ - Displays "/"

Value conditionals:
@+ - positive. Only show this stat if it's not zero. It will still move to the next line unless you use "@c".
@s - secret. If the value is zero, then hide this stat and all the rest of the stats on this line. You should only use this on a stat that starts a line.

Mission conditionals:
These codes are followed by exactly 2 digits. The stat will be ignored if the condition is true.
@< - less than.
@> - greater than.
@= - equal to. (or, don't show on this mission)
@! - not equal to. (or, only show on this mission)

Appendix 2 – Possibly Complete List of Pre-defined QVars

DrSAerials	Aerial knock-outs performed
DrSBackStabs	Back stabs performed
DrSBodyFound	Bodies found by enemies
DrSCmDmgDeal	Damage dealt in total over campaign
DrSCmDmgTake	Damage taken in total over campaign
DrSCmKills	Total kills (does that include innocents?) over campaign
DrSCmLoot	Total loot picked up over campaign
DrSCmReload	Total number of reloads over campaign
DrSCmTime	Total elapsed time for campaign
DrSDiscovery	Number of AIs alerted maybe
DrSDmgDealt	Damage dealt
DrSDmgSelf	Damage inflicted on self, eg by falls (I guess)
DrSDmgTaken	Damage taken
DrSGameCode	
DrSGassed	Number of AIs knocked out by gas arrows (I guess)
DrSHealing	Healing taken
DrSInnocent	Innocents killed
DrSKills	Non-innocents killed
DrSKnockout	Knockouts performed
DrSLkPickCnt	Locks available to pick
DrSLockPick	Locks picked
DrSLootTotal	Total loot in the mission available to be picked up
DrSObjDmg	
DrSObjKilled	
DrSPocketCnt	Pockets available to be picked (including arrows from archers, keys, etc.)
DrSPocketFail	Count of failed picks (what is a failed pick)
DrSPocketOK	Pockets picked
DrSReloads	Number of reloads
DrSRobotsDeactivated	Robots deactivated
DrSRobotsKilled	Robots destroyed
DrSScrtCnt	Secrets available to find
DrSSecrets	Secrets found
DrSSuicides	Count of exploding frogs?
DrSTime	Time taken to complete the mission (I have a feeling this is only calculated at the end)
difficulty	This is the difficulty, 0 for normal, etc. Discussed in *Difficulty and Objectives*
InitStatsBuilt	I guess this goes to 1 as the mission starts, and does not change.
map_locs_visited	Used for automaps
map_max_page	Map images number up to the number
map_min_page	Map images number up from the number
map_oob_loc	Used for automaps I guess
map_oob_page	Used for automaps I guess
mission_complete	I imagine this is 1 when the mission is complete, and 0 otherwise
schplay_%s	?
schplayid_%d	?
total_loot	Loot picked up by player in the previous mission (used for the equipment store, then discarded when the game starts)

Appendix 3 – Summary of Objective QVars

Variety	*Values*	*Notes*
goal_state_n, x	0 = not complete 1 = complete 2 = inactive 3 = fail	
goal_visible_n, x	0 = invisible 1 = visible	Gets added with a default value of 0, if put in.
goal_type_n, x	0 = QVar controlled 1 = steal an object 2 = kill person 3 = steal loot 4 = go to location	
goal_target_n, x	number of item (to steal, kill or get to)	
goal_min_diff_n, x	0 = normal 1 = hard 2 = expert	If 1 then goal n will be for hard and expert only.
goal_max_diff_n, x	0 = normal 1 = hard 2 = expert	If 1 then goal n will be for normal and hard only.
goal_optional_n, x	0 = normal 1 = optional	Mission can be completed without completing this objective
goal_reverse_n, x	0 = normal 1 = reverse	Reverses the goal_target, eg kill guard, would be, do not kill guard
goal_satisfied_n, x		Used in-game to note whether a final objective is satisfied..
goal_bonus_n, x	0 = normal 1 = not visible	Objective becomes visible if completed; mission can be completed without completing this objective
goal_special_n, x	special number	For an objective to get a certain loot item
goal_specials_n, x	binary value of special flags	For an objective to get a certain set of loot items
goal_final_n, x	0 = normal 1 = last goal	This objective will only complete if all the other necessary objectives have been completed
goal_irreversible_n, x	0 = normal 1 = true	Once the objective is completed if will never become not complete, even if the conditions change, eg if an item is required, but is subsequently dropped goal remains completed
goal_gold_n, x	total value of gold	
goal_gems_n, x	total value of gems	
goal_goods_n, x	total value of goods	
goal_loot_n, x	total value of loot	gold, gems, items combined

Appendix 4 – Creative Combinations

A standard combination lock is just a slight variation on the key and lock; find the combination, use the combination to open the door. How much better if the player has to think a bit?

In my MechBank, the combination to a magic box is set with a series of levers, each adding (or subtracting) a certain amount to the combination. The player has to determine what each lever does, then work out which ones are required to get the correct value (all much easier to set up with QVarTraps than for the player to solve, I suspect).

Alternative ways to give the combination

Here is another possibility, with the combination in the form of a riddle:

> If the contents you would get
> All the numbers you must set
> Each is secret, but here are clues
> Never the combination you will lose
>
> The number of towers is my first
> From the finest to very worst
> The age of Mary is the second
> When she felt the goddess beckoned
>
> So far so good, how have you fared?
> The next is simple, tis three squared
> The number of ways to skin a cat
> Is the fourth, now that is that.

The player can count the towers, perhap find a book called "The Five Ways to Skin a Cat", and a diary that mentions Mary was nineteen when she became a nun.

How about a note from the locksmith:

> The vault system has now been installed, as per your instructions; the combination is the year you were born, plus the year your husband was born.

And a diary somewhere:

> 20th July, 1365
> Mary is making a big deal about it being my birthday today. A little unfair, I feel, it is only 2 years until she is forty.

A bit of thought should give 2652 as the combination. In Rosaries are Red, by Tainted, the player has to visit four places to get each of the four digits - and there are numerous ways one could display them at each location, from a number of the wall (like in the corridor of this mission) to the shape of the room itself. I would advise either making the numbers obvious, or the locations obvious.

Alternative uses for combinations

- A combination could be found in the control room of a Mechanist complex. One combination deactivates this robot, another deactivates that SecCam, allowing the maintenance crew to repair them... or Garrett get into areas he should not.
- In Shipping... and Receiving, one combination system opened several doors. A variation on that might allow Garrett to alter patrol routes, perhaps getting AIs on opposing teams to meet up.
- The Mighty Morphing Machine requires a lump of Mighty Morphing Matter in the hopper, a combination set, and a button pressed. A handful of combinations produce useful items, anything else produces slag.
- Use the combination to set the light colour in a room; under certain frequencies, different texts can be read.
- Type in the coordinates for a teleportation device. In a fantasy scenario, you might like to say these relate to distances along ley lines
- And of course, combinations can lock elevators, machines, drawbridges... I suppose you could even use one to lock the door to a vault.

Appendix 5 – Using Custom Scripts

This was my first experience with custom scripts, and I realised that, one, they have a huge potential, and, two, they are pretty easy to use. Scripts are hed in groups called scripot modules, and the script module I using was NVScript, by Nameless_voice. This is what I had to do:

Download the zip file from the internet, and unzip the file so the script module (the file with the .osm extension, in this case NVScript.osm) is in the Thief root directory.

Open you mission up in DromEd and load the script module into the mission with the script_load command:

```
script_load nvscript
```

You should see “Module Loaded” in the status bar at the bottom.

You can now use each script as often as you like, just as you would a normal script like TrapConverse.

When you come to package your mission fir distribution, remember to include the .osm file (and to thank the script author and include any required copyright in the mission info).

Telliamed's custom scripts (tnhScript 1.5.4) can be downloaded from this page:
http://www.thiefmissions.com/telliamed/

If you unzip it you will see a lot of files in there. Most of these are the source code (in C++), and unless you want to write your own scripts, you can ignore them. Just make sure you get the script modules (with a .osm extension) and the documentation.

Nameless_voice (http://www.geocities.com/nameless_voice/) has some scripts here:
http://www.geocities.com/nameless_voice/Downloads/Tools/NVScript.zip

For the random combinations thing, you need at least version 1.0.7 of NVScript.