CONVERSATION STYLEBOOK

This manual is not about the technical aspects of writing Ultima System 7 conversations -- that topic is addressed in the HOW TO WRITE CONVERSATIONS and AGIL FOR CONVERSATIONS manuals. Rather, this book is concerned first with how to write GOOD conversations (as opposed to just writing mediocre scripts) -- and, secondly, this manual deals with some organizational rules to aid writers as they work in a team.

The manual is divided into these sections:

PRINCIPLES OF CONVERSATION -- A collections of guidelines, edicts, and rules-of-thumb regarding the writing of conversations.

CONVERSATIONS AND THE GAME -- How conversations fit into the grand scheme of game design, and how authors interact with the other members of the team.

CONVERSATION CONTROL -- Some organizational tips for members of a writing team, especially as regards the Net.

PRINCIPLES OF CONVERSATION

Dialogue Vs. Narrative

First, some definitions: "dialogue" is what the characters say, while "Narration" is what the Voice from Beyond says. For example, in the following scrap of conversation, the Narration is marked in bold --

"I am the Candle," **said the Sofa Pillow.**
"Are not," **said the Rug.**

In an Ultima System 7 game, we are trying to give the player the illusion of reality -- these are Virtual Reality games (or as close as we can get in real life). In real life, there is no narration. No invisible voice tells you what happens -- you either see it (or witness it in some other sensory way), or you don't. Thus it should also be in one of these games -- *THERE SHOULD BE NO NARRATION.*

How do you eliminate narration in a conversation? Here are some examples, with the "narrative" style of conversation to the left, and the "virtual reality" style to the right:

With Narration	**Without Narration**
"I am Rotundo the Baker," he said.	I am Rotundo the Baker.
"I'll get the key for you." Rotundo goes to the cupboard and returns with the key. "Here it is."	I'll get the key for you. *Conversation ends. Rotundo walks to the cupboard and returns with the key. Conversation resumes.* Here it is.
"Thou art a cute one." She smiles at you.	Thou art a cute one. *Character's portrait changes to a smiling face.*

Where the Virtual Reality technique breaks down is where our technology is not sufficiently advanced. For instance, what if the author wants Rotiel the Wounded Guard to become too ill to talk, after giving up a few of his clues. In a narrated dialogue, we might write:

"And the vital secret is . . ." Suddenly, Rotiel winces in pain. He cannot speak further.

We could do the changing-of-Rotiel's-portrait technique, but "wincing" is rather subtle and the player might not catch on. Or we might change Rotiel's dialogue to:

"And the vital secret is . . . [cough, cough] . . . I'm sorry, but I can speak no more."

Or we could use a third character, bringing them in to act as a quasi-narrator.

Rotiel says	And the vital secret is...
Iolo says	Milord, Rotiel has doubled over in pain. He canst speak no more.

Sound effects or music could also be used to make a point in a conversation. Possibly, there could be a "coughing" sound effect added to the game for Rotiel's conversation.

Lastly, if all else fails, narration can be used to make something clear to the player. When used, narration appears in its own screenbite, without a portrait, in a different text color. For example, this code --

```
["And the vital secret is..."]
Narrate("Suddenly, Rotiel doubles over in intense pain.");
```

would produce these words next to Rotiel's portrait --

And the vital secret is...

and then, after a mouse click, these words appear, in an area without a portrait --

Suddenly, Rotiel doubles
over in intense pain.

Spell Checking

It is impossible to overstate the importance of spell-checking every conversation. Unfortunately, spell-checking programs typically choke when they encounter all of the AGIL code we normally use. Therefore, the programmers have provided us with a utility which can "strip" all of the text out of a conversation file, and place it in a new file which can be spell-checked.

The script-stripping program is called STRIPTXT.EXE, and exists in the \Serpent\Util directory on the Net. Before using this program, you should copy it to your own C:\Serpent\Util directory. You should also make sure that your Serpent\Util directory is on your Path. (If you don't know what this means, ask your Lead Writer.)

To use this program from the ordinary DOS prompt, type the name of the program, the name of the file which you want to strip, and the name of the new file where the stripped text will be put. For instance, if you want to text-strip the conversation for Brian, you might enter:

striptxt brian.use brian.out

After you hit the "Enter" button, you'll see something like this on your screen --

STRIPTXT v1.1
(C) 1991 Origin Systems, Inc - PHS -

Doin' that strip thang...

After the program does its business, there is a new file on your hard disk -- in our case, Brian.Out. It probably looks something like this --

9: weapons & armour
20: Whats going on?
102: name
102: job
102: bye
118: "Avatar, how may I serve thee?"
123: name
124: "I am Zorn."
125: name
128: job
130: "I am the blacksmith of Minoc."
131: blacksmith
131: Minoc
134: "Does thou have no respect for the dead? Indeed,
135: trying to solicit at a terrible time like this!
136: When people have been found murdered over at
137: William's sawmill!"

The number at the left is the line number from which the text comes -- information which will come useful when it's time to make corrections in the original script.

The writer then opens this text file -- in our case, Brian.Out -- in MS Word, and spell-checks it there. In the above example, we quickly learn that line 20 is missing an apostrophe! Taking note of the errors, we then switch to Sage and make the corrections in the original file -- in our case, Brian.Use.

Humor

Ultima games -- meaning those Ultima System VII games which are set in the world of Britannia -- are never "funny" or comic games, although they may have comic elements where appropriate. The NPCs should be conscious of and maintain their dignity, with these exceptions --

* Characters with no dignity by profession, such as Chuckles the Jester.
* Characters with no dignity by personality, such as someone who is known for making puns.
* Any character, when he is in an informal and relaxed circumstance -- such as the Companions are, when they are alone with the Player Character and can banter with him.

Comic moments can be designed within a plot in order to relieve the pall of doom or relentlessness of battle. In fact, it is a good idea to intersperse lighter moments between dramatic, horrific or dark episodes.

Humor which is made humorous due to references to the world beyond Britannia violates the concept of the Virtual Reality game, and must be avoided. If someone in Britannia would not understand the joke, then that piece of humor should not go into the game. These are examples of what to avoid: A sailor who happens to look exactly like Popeye. Characters who crack jokes out of Monty Python and the Holy Grail. "Inside" jokes of any kind, such as asides about other computer games.

Silliness should be limited to those characters whose personality would motivate them to be silly. If you think something might be too silly for the game, then it probably is.

Writing for Emphasis

Some lines of dialogue can be interpreted in more than one way, depending on where the emphasis in the sentence is put. In print, we can mark the emphasis by using italics --

Who is the Guardian?	[the normal question]
Who is the Guardian?	[expressing disbelief]
Who *is* the Guardian?	[suggesting a deeper question]
Who is *the* Guardian?	[implying there is only one rightful Guardian]

When writing dialogue for the Ultima System 7 games, however, the authors do not have italics to play with. Nor can we highlight words, or underline, or use bold face, or even flashing words. Capital letters are difficult to read in medieval script, and so we do not use all-capitals (Who IS the Guardian?) for emphasis.

For reasons of style, it is desirable that the authors all use the same system for placing emphasis. The chosen system is to place a dash immediately before and after the emphasized word, without spaces in-between --

Who -is- the Guardian?

This emphasis system is a bit clunky, and not all players will interpret it correctly. Therefore:

* Only use emphasis when you really need it. Skilled writing can remove the need for added emphasis.
* If it is possible to recast a sentence so that it does not require emphasis and still serves its purpose, do so.
* Avoid sentences which require multiple emphasized words. For instance, none of these look right:

What is the -Legion- -of- -Doom-?
What is the -Legion of Doom-?

The Principle of Intrusivity

When playing a Virtual Reality game, the goal is for the Player to forget that he is playing a game -- he should instead be having a Virtual Adventure. Anything which intrudes upon the Virtual Reality to remind the Player that he is indeed playing a computer-game is an Intrusion. The Doctrine of Intrusivity is rather simple: the author's job is to minimize Intrusions.

For instance, Intrusions can be caused by Keyword problems, by characters speaking out-of-character or out-of-period, or by "cute" jokes which reference other games or other worlds besides Britannia.

The Limitation of Keywords

Many of the Intrusivity problems deal with the proper use of Keywords. The use of Keywords is a concession to our current game-technology -- in a proper Virtual Reality environment, the Player should be able to say whatever he wants to the NPCs. However, we don't have voice-recognition in our games, and there is no other convenient way for the Player to input entire sentences into the game (typing is too slow). Therefore, we have the menu selection system which involves letting the Player pick from a list of options (which we call Keywords).

The **First Error** which a writer can make is to not give the Player the full range of options which he should have. If a Player comes to a point where he wants to say something which his Keyword options do not allow him, then the lack of proper Keywords becomes an Intrusion. The author must make every effort to anticipate what the Player might want to say, and allow him these options.

For instance, in the following conversation scrap, the powerful King Gollop asks the Player to serve him as a Knight:

```
["Wilt thou serve me loyally as a Knight, <PCName>?"];
                Answer = ~AskYesNo();
```

Let's imagine that this decision happens to be important in the game, and that the Player knows almost nothing of this Gollop the King. A Player, reaching this point in the conversation, might feel trapped -- he doesn't know if he wants to say Yes or No. What if he wants to think about it? Or can he ask the King for more information, first?

A better script, therefore, might read like this --

```
  ["Wilt thou serve me loyally as a Knight, <PCName>?"];
Answer = ~AskList(<<"Yes", "No", "not sure", "ask for more
                    information">>);
```

Does this mean that the author should never use ~AskYesNo()? No. There are many times when "yes" or "no" are all the options which the Player will want. The author just needs to be aware of the Player's viewpoint when writing his script.

The Second Error which authors can make when using Keywords is to assign inappropriate Keywords to actions. In other words, when a Player chooses a Keyword, he should not be surprised by the immediate consequences -- if he chooses "job," his character should say something like "What is thy job?" and not "Canst I have a job?" or "I bet that I canst guess thy job" or some other surprise.

A less blatant case of inappropriate Keywords comes when the author has thought of something particularly clever or witty which he wants the Player Character to say. Unfortunately, the Player might not want to be so witty or clever (especially if the line can get him in trouble). For instance, let's say the author wants the PC to say --

"So, art thou a jester, or dost thy mother dress thee funny?"

Nice line, no? But you couldn't just put this with the Keyword "job" -- yes, the PC is asking what the NPC's job is, but the Player would feel imposed upon by the game if his attempt to simply ask "What is thy job?" turned into an insult.

However, let's say that we write the conversation in such a way so that the first time Chuckles makes the Player feel stupid or irritated (which is what Chuckles is for, after all), the Player gets a special choice:

```
Attitude = ~AskList("be nice", "be nasty");
```

Then we can have the Keyword group for "job" look like this --

```
key "job"
   if (Attitude = "be nice")
      {
      PCSays("What is thy job");
      }
   else
      {
```

```
PCSays("So, art thou a jester, or
   dost thy mother dress thee funny?");
}
```

The Rules of Dialogue

The Ultimate Prescript of Dialogue Writing is this: the characters should only say those things which are consistent with their motives and personality. This is what is known as "staying in character."

All of a character's dialogue should be consistent with his character -- even his worktype barks, and his friendly banter (if he's a Party member).

Staying " in Period." Characters should sound as if they belong to the genre and age in which the game is set (the rare exception being those characters whose origin lies outside that game's setting). For most Ultima VII System games, this means that the characters should speak in pseudo-medieval jargon.

See Sheri Hobbs' report on Medieval English for the rules of using "thee"s and "thou"s.

The corollary to this is that characters should *not* use modern expressions or slang. The comic fat guy, for instance, should not quip about needing to go on a diet -- that's a modern concept. He might say something to the same effect, but he wouldn't use the word "diet" in this context. Instead, he'd be more likely to quote the old precept about "moderation in all things."

Furthermore, you can't just "medievalize" modern slang. These, for instance, sound wrong --

"How is it going, milord?"	(Too informal, too modern)
"Hold thine horses, sirrah!"	(Too different from the slang "hold your horses" for the reader to recognize the expression)
"Thou better believest it!"	(Sounds like modern language rudely medievalized)
"Holdest it right there, gent."	(Clearly modern.)

Avoiding Game-Speak. The characters should never be aware that they are in a game. Iolo, for instance, would not understand such terms as hit points, alignment, stats, levels and experience points. He would never say this --

"Young Barney is of too low a level to be of any use to us."

Instead, Iolo might say --

"Young Barney is too inexperienced to be of any use to us."

Linearity and Game Design

The concept of Linearity is all about choices. Our conversations are designed so that a Player has choices in how he can respond to an NPC's previous statement.

A conversation is said to be **Linear** if the Player only has one choice. A script with fewer choices is more Linear that a conversation with many choices. Linearity is bad to the extent that it causes an Intrusion (see "The Limitation of Keywords," in this document). Linearity is also bad if the Player begins to feel that his role in a conversation is irrelevent, due to a lack of meaningful decisions.

A good conversation is **Multilinear**, meaning that it provides many choices to the Player. Furthermore, the choices need to be viable options. For instance, the following scrap of conversation code provides several choices, but one is a duplicate and another is an unlikely (or silly) selection --

```
["I am the mistress of Castle Ogront. What dost thou wish of me?"];
        TurnOn (<<"food", "weapons", "a meal", "a date">>);
```

As a rule of thumb, each Keyword which leads to additional Keywords -- in other words, any Keyword not at the Bottom of the Logic Tree (see "Tree Structure Doctrine") -- must lead to at least two Keywords, and optimally to as many as four or five. Furthermore, each of the choices should have unique, interesting consequences.

Interruption Doctine

Imagine that all of the characters in a conversation are divided into two teams: the Player Character's team, and the Other Team. No character can belong to both teams, or switch between teams during any conversation. These teams have nothing to do with alignments, or being in the Party, or anything other than this particular rule of style.

Once you have the Teams in mind for the conversation, write it in such a way that no team member ever interrupts or speaks after someone else from his team. Ever. Think of this as tennis, and the conversation as the ball hit over the net -- no two team members ever hit the ball sequentially, do they?

For instance, let's say that the Player Character's Team consists of the PC and Dupre. The Other Team includes Rotundo the Baker, and Iolo. Our interruption rules tell us that Dupre can interrupt Rotundo or Iolo, but never the PC (who is on his team). After Rotundo the Baker speaks, either the PC or Dupre can say something, but not Iolo.

Why do we have this apparently arcane rule of thumb? Because adherence to Interruption Doctrine means that during a conversation, a speaker's portrait will always appear in the same place. Everyone on the PC's Team will appear at the bottom of the screen, and everyone on the Other Team will always appear at the top. No character will appear once at the top, then bounce down and appear at the bottom later in that same conversation.

When you need to add a character to a conversation, do not put him on the same team as the person he is most likely to be interrupting or speaking after. For instance, if in Rotundo the Baker's conversation Iolo has the role of verbally defending the PC, then he should be on the PC's Team in order to rebutt Rotundo's unkind remarks.

Tree Structure Doctrine

The conversations for new Ultima VII System games are to context-controlled through a strict tree structure. **Context-Control** means that the author designs the conversation in such a way that the PC only has Keyword selections which are pertinent to his current topic. For instance, after asking about the local inn and hearing a reply, the PC cannot suddenly blurt "job" -- his choices (Keywords) might be something like "Joe the Barkeep," "Brunisian ale," and "change the subject." In order to get back to an old Keyword such as "job," the Player would have to choose "change the subject."

In order to write this sort of conversation, the author needs to sketch out his conversation's flow -- its **Logic Tree**. A small conversation's Logic Tree might look like this (reading from left to right) --

```
        |---Red
Name ---|---Flame
        |             |---Poison
        |---Dagger---|---Foul Nondo
```

```
                    |---Waterfall

        |---Panderer
Job  ---|---Masseur
        |---Lover

Bye
```

In other words, if the Player selects "Name," the NPC says something which gives the Player three new Keywords -- Red, Flame and Dagger. "Red" does not lead to anything new, but "Dagger" brings the new Keywords "Poison," "Foul Nondo," and "Waterfall."

Keywords are said to be "on top" of the tree if they are the first Keywords the Player encounters (in the example above, those words to the far left -- Name, Job, Bye -- are **Top Keywords**). Top Keywords must be "Turned On" before entering the Converse Loop -- in other words, the TurnOn statement for those Keywords should be placed before the Converse statement.

Those Keywords which lead to no new Keywords are at the bottom of the tree, and are **Bottom Keywords** -- such as Red, Flame, Poison, Foul Nondo, Waterfall, Panderer, Masseur, and Lover from the above example.

Following the Key statement for any Bottom Keyword, in the key code-group pertaining to that Keyword, there should be a TurnOff() statement for that Keyword. When following proper style, the author places this TurnOff() at the end of the key-group. For example --

```
key "Lout"
   {
   PCSays("Why do they call thee a lout?");
   ["'Cause I art one."];
   TurnOff("Lout");
   }
```

Following the Key statement for any Keyword which is NOT a Bottom Keyword, at the end of the key-group, there should be the following statements ***in precisely this order:***

PushKeys()	To save the current Keyword list.
TurnOn()	To set up the Keywords for the next conversation level.
AllowChange()	To add the keywords for changing the subject.

For instance --

```
key "who"
   {
   PCSays ("Who are the people who call thee such names?");
   ["Just my friends -- Gentrid, Lanni, and Foul Nondo."];
   PushKeys();
   TurnOn(<<"Gentrid", "Lanni", "Foul Nondo">>);
   AllowChange();
   }
```

In other words, thanks to Tree Structure Doctrine, every Key Statement code-group has either a TurnOff() or a TurnOn(), but never both. If there is a TurnOff(), this Keyword must be at the bottom of the conversation's Logic Tree. If there is a TurnOn(), then this Keyword leads to other Keywords.

Adding a New Subject

Current conversation style allows for the NPC to introduce new subjects, and for the Player to change from one subject to another (see "???" in this article for more information). But what if the Player wants to bring up a new topic of conversation?

Obviously, we're limited by Virtual Reality and the Keyword system. (Technically, we could allow the Player to type in his own Keywords, but this sort of defeats the purpose of the current system.) We can't let the Player add Keywords of his own design to the conversation.

The next best solution is to design the game so that it is aware of what topics the Player might want to bring up in a conversation. We'll call these **Global Subjects**, since they apply to more than one conversation. For example, in *Ultima VII, Part One,* the Player quickly learns about the murder in Trinsic -- "murder" is an example of the Global Subject concept, since the PC will want to discuss this topic with everyone he meets in Trinsic.

There is usually a Global Flag associated with each Global Subject. The Flag is set to "True" when the Subject becomes active, or in other words, when the Player would want to start asking people about the topic. The Flag should be reset to "False" when the Player has finished with the topic -- for instance, once the PC has solved the murder mystery in Trinsic, the MurderInTrinsic flag should be set to "False" again.

The new AGIL routine, AddGlobalSubject(), should be placed at the start of most conversations -- just before the first TurnOn() statement in the Source = _Action section. If any Global Subjects are active, this code causes the Keyword "Bring up a new subject" to be added as one of the **Top Keywords.** Selecting this Keyword allows the Player to select from Keywords for all of the Global Subjects.

Furthermore, every Converse Loop must include Key statements and code in order to handle every possible Global Keyword. Fortunately, there are very few Global Keywords, and most characters probably would simply say something like "I know nothing of that."

KeyThoughts

KeyThoughts are a new concept which are reflected here in the Stylebook, but which (due to time and schedules) have not yet been incorporated into the AGIL FOR CONVERSATIONS document. The basic idea is to use thoughts rather than single words. For example --

Keyword	***KeyThought***
job	I wonder what his job is.
name	I wonder what his name is.
murder	Perhaps I should ask him about the murder.

In order to reduce the change of typographical errors in writing conversations, KeyThoughts will be defined as Variables at the start of the conversation. The Variables will then be used just as the Keywords themselves have been used in the past. For instance:

```
JOB = "I wonder what his job is.";
TurnOn (JOB);
Key JOB
   {
   [Stuff goes here];
   }
```

Max number of Keythoughts

CONVERSATIONS AND THE GAME

Art.
Stats.
The Character's Environment.
The Character's Schedule.

CONVERSATION CONTROL

Global Flags

PVCS and the Net