Conversation Extension
Quick Start Guide

Introduction

Suppose you just wanted to put an NPC into your game that the protagonist could talk to about the weather. Here is an example transcript of what that conversation would look like using the Conversation extension:

> talk to old man
" ""Hi there old-timer!"" you tell the old man. ""The name's Janet. I just moved into town and they say you're the man to talk to about the weather in these parts.""

""Pleased to meet you, Janet,"" he replies. ""You can call me Geezer. The weather around here can be a tad fickle, but I reckon I know her better than most folks. What do you want to know?""

-- (topics) –
Geezer
mudslides

> t mudslides
"My landlord told me to watch out for mudslides because of all the rain we've been having," you tell him. "He says they can bury a car."

"True enough about the rain, but that **landlord** of yours was pulling your leg about the mudslides," he tells you. "Just take a look around lady. No hills. Should seem pretty obvious that you ain't gonna get a mudslide without any hill for it to slide down on.

> t landlord
"Sounds like you know my landlord pretty well. Anything I need to know about him?" you ask.

"I heard you was renting a cottage from Harkan," he replies. "He's a fair man, but he'll work a joke on you quick as a wink, so keep your brain going when you're around him."

There are a few things to notice about this example conversation:

- The conversation was initiated by the reader/player with the `talk to <NPC name>` command. The alternate command `tt <NPC name>` could have been used as well.
- The Conversation extension responds to the command with an extended header for a list of the topics currently available for discussion. It is important to note that the header presents the topics as if it is the protagonist that thinks of what he or she wishes to talk about as opposed to

the NPC listing what it can talk about. As a consequence, you must be sure that your topic header text is in keeping with this approach. The examples that appear later in this guide show one way to do this.

- The reader/player responds to the topic listing by indicating which topic he or she wishes to talk about with the `T` <topic> command.
- If you use an actor instance as a topic in your game, the Conversation extension will show the name of that actor as the first entry in the topic list. It does things this way because it is quite common for the protagonist to want to ask an actor to talk about himself or herself prior to talking about other topics, so it is logical for the name of the actor to be the first topic on the list.
- You can have one topic introduce another topic to create a branch on the conversation tree. In this example, the `mudslides` topic introduces the `landlord` as a new topic that was not on the list in the beginning of the conversation. This new branch could have lead to a series of new topics, and any of these new topics could have branched. Thus, you can create complex conversation trees capable of modeling a real-world conversation.

This Quick Start Guide shows you how to program a mini game that will replicate the example transcript shown here. The QuickStrtGame.alan file contains the complete source code for this example.

If you want to see a larger example that demonstrates everything the Conversation extension is capable of, look at the source code and the comments in the The_Interview.alan file.

Installing the Conversation extension

Installing the Conversation extension is very simple:

1. Place a copy of verbs.i from the Standard Library in your project directory.
2. Delete the verb definitions for `talk`, `talk_to`, and `talk_to_a` from verbs.i.
3. Place a copy of Conversation.i into your project directory.
4. Create the main Alan file (game1.alan for example) for your project and put Import statements for both these files into it:

```
import 'conversation.i'.
import 'verbs.i'.
```

5. Add import statements for the rest of the standard library:

```
Import 'classes.i'.
Import 'locations.i'.
Import 'messages.i'.
```

Creating a starting location

Since this example focuses on the NPC and the conversation, the code for the starting location is fairly minimal:

```
The field isa location
        Name Farmland
        Description
                "This lush, rich looking farmland stretches out to the horizon. There is a fence
                running along the edge of the field in front of you."
End the.

The fence isa scenery at field
        Name fence
        Description ""
        Verb examine does only
                "It is a waist-high rail fence. It's wooden rails are painted white."
        End verb.
End the.

Start At field.
```

Creating an NPC for the conversation

Once you have a location, you can create an NPC that the protagonist can talk to. The code for our Geezer NPC in the example transcript looks like this:

```
The Geezer Isa male at field
        Name Geezer Name old man Name old

        Description
                "An old man is standing on the other side of the fence. He seems to be waiting for
                you to talk to him."

        Verb examine
                Does Only
                "The old man is dressed in overalls. His weather-hardened face tells its own story, a
                story of a man who has devoted his life to nurturing the soil and working the land. "
        End Verb examine.

        Has topics_header1
                " ""Hi there old-timer!"" you tell the old man. ""The name's Janet. I just moved into
                town and they say you're the man to talk to about the weather in these parts.""
                $p""Pleased to meet you, Janet,"" he replies. ""You can call me Geezer. The weather
                around here can be a tad fickle, but I reckon I know her better than most folks.
                What do you want to know?"" ".

        Has topics_header2
                " ""I'd like to talk to you again Geezer.""
                $p""What's on your mind?"" he replies.".

        Has unknown_topic_header
                "After a bit of thought, you find that all you really want to talk
                to Geezer about is:".

End The.
```

As you can see, the Conversation extension has added three new attributes to the actor class that you have to provide values for:

topics_header1

When you talk to an NPC for the first time in a game, the Conversation extension uses the text in `topics_header1` as a header for the list of currently available topics. In the code example above, the text takes the approach of having the protagonist introduce herself to the old man as way to open the conversation. You are free to do it otherwise in your game.

topics_header2

After the first time you talk to and NPC, the Conversation extension uses the text in topics_header2 as a header for the list of currently available topics.

If you don't wish to make a distinction between the first and subsequent headers, you make set both values to the same text. Also you may change the value for `topics_header2` at any time during the game and the Conversation extension will display your new header from that time onward.

unknown_topic_header

It is possible for the reader/player to refer to any object in the game world with the T command, even if that object does not appear on the list of currently available topics. When that situation arises, it will use whatever text value you have assigned to the `unknown_topic_header` attribute in the NPC that the reader/player is talking to. Since each NPC has his or her own unique header, this value can help you give each actor a different personality.

Creating topics for the conversation

Now that we have a starting location for the game, and we have placed an NPC in that location, it is time to create the topics that the NPC can discuss with the reader/player.

Using an actor as a topic

If you refer back to the example transcript that appeared in the Introduction of this guide, you will notice that the Conversation extension provided a list of topics that the Geezer can talk about when the reader/player initiated the conversation:

```
-- (topics) –
   Geezer
   mudslides
```

The first topic on this list is Geezer, indicating that Geezer can talk to the reader/player about himself. What we did was to take advantage of the fact that the Conversation extension allows us to use any *thing*, be it actor or object, as a topic in a conversation.

To implement this capability, the Conversation extension added the `talkers` attribute to every thing in the game. In your code, if you include the ID of an actor in the `talkers` attribute of a thing, the Conversation extension will display the name of that thing in the list of available topics for that actor. Conversely, if you remove an ID from the `talkers` attribute of a thing, the name of that thing will disappear from the topic list for the actor whose ID you removed.

So, to get Geezer's name to appear as a topic on the topics list when the reader/player initiates a conversation, we added a line of code immediately after the mandatory START AT clause:

```
Start At field.
      Include Geezer in talkers of Geezer.
```

But what does Geezer have to say about himself? At the moment nothing, but that's easy enough to fix.

Adding text to a topic

Since you can have more than one NPC in your game that can talk about any given topic, it stands to reason that each topic object must somehow store reply text in blocks, with each block being tied to a specific NPC. The Conversation extension uses Alan's capability to override the default action of a verb to do just that.

Here is the code (see highlighted area below) that we added to the Geezer object that contains an override for the `T` verb that stores the text for Geezer's reply:

```
The Geezer Isa male at field
      Name Geezer Name old man Name old

      Description
            "An old man is standing on the other side of the fence. He seems to be
            waiting for you to talk to him."

      Has topics_header1
            " ""Hi there old-timer!"" you tell the old man. ""The name's Janet. I just
            moved into town and they say you're the man to talk to about the weather in
            these parts.""
            $p""Pleased to meet you, Janet,"" he replies. ""You can call me Geezer. The
            weather around here can be a tad fickle, but I reckon I know her better than
            most folks. What do you want to know?"" ".

      Has topics_header2
            " ""I'd like to talk to you again Geezer.""
            $p""What's on your mind?"" he replies.".

      Has unknown_topic_header "After a bit of thought, you find that all you really want
                               to talk to Geezer about is:".

      Verb examine
            Does Only
                  "The old man is dressed in overalls. His weather-hardened face tells
                  its own story, a story of a man who has devoted his life to nurturing
                  the soil and working the land. "
      End Verb examine.

      Verb t does
            Depending on listener of hero
                  = Geezer then
                        " ""So Geezer, tell me what do you do around here that makes
                        folks think you're an expert on the weather?""
                        $pHe gives you a shy smile. ""I'm a good farmer,"" he replies.
                        ""You get to know the weather pretty well when your livelihood
                        depends on it."" "
            End depend.
      End verb.
End The.
```

Creating topics from scratch

So far we've used an actor as a topic, and you can use any existing object in your game for a topic in a conversation by using this same technique. However, it is frequently necessary to talk about something that is not already an object in your game.

The mudslides topic in this example is a good example of such an object. An object named mudslides does not exist as an object in the game world, but the reader/player still has to refer to it in order to converse about it, so the Conversation extension includes the *topic* class that allows you to create topics for use in your conversations. In the source code for this example, an instance of the topic class looks like this:

```
The mudslides isa topic
        Name mudslides

        Has Talkers {Geezer}.
        Verb t does
                Depending on listener of hero
                        = Geezer then
                                " ""My landlord told me to watch out for mudslides because of
                                all the rain we've been having,"" you tell him. ""He says they
                                can bury a car.""
                                $p""True enough about the rain, but that " Style emphasized.
                                "landlord" Style normal. "of yours was pulling your leg about the mudslides,""
                                he tells you. ""Just take a look around lady. No hills. Should
                                seem pretty obvious that you ain't gonna get a mudslide without any hill for it
                                to slide down on."" "
                End depend.
        End verb.
End the.
```

You will notice that we put the ID for Geezer in the Talkers attribute value here in the definition for the topic. We did this to show an alternate method to initialize a topic that does not use an Include statement. When you do the initialization in this manner, the topic will appear in the topic list for the NPC at the start of the game. Of course, you can still use Include and Exclude statements to modify the Talkers attribute value at anytime later on in the game to include or remove the topic from Geezer's list of topics.

Creating a branch in the conversation tree

Upon further examination of the mudslides code presented earlier, you will notice that we emphasized the word "landlord' in the text for Geezer's speech. This step was taken to clue the reader/player that we have added a new topic to the conversation list to create a branch in the conversation tree. Such cluing is not strictly required because the reader/player can always use the Topics command to periodically refresh the list, but it does remove one possible frustration he or she might otherwise have while playing your game.

To create the new branch you have to first create a landlord topic object:

```
The landlord isa topic
        Name landlord

        Verb t does
                Depending on listener of hero
                        = Geezer then
                                " ""Sounds like you know my landlord pretty well. Anything I need to know about
                                him?"" you ask.
                                $p""I heard you was renting the white cottage at the east end of town from
                                Harkan,"" he replies.
                                ""He's a fair man, but he'll work a joke on you quick as a wink, so keep your
                                brain going when you're around him."" "
                End depend.
        End verb.
End the.
```

Now that you've created the topic object, you add it to Geezer's topic list with an Include statement inside the `T` verb override (see the yellow highlight below) as was done in the code example shown earlier:

```
The mudslides isa topic
        Name mudslides

        Has Talkers {Geezer}.
        Verb t does
                Depending on listener of hero
                        = Geezer then
                                " ""My landlord told me to watch out for mudslides because of
                                all the rain we've been having,"" you tell him. ""He says they
                                can bury a car.""
                                $p""True enough about the rain, but that " Style emphasized.
                                "landlord" Style normal. "of yours was pulling your leg about the mudslides,""
                                he tells you. ""Just take a look around lady. No hills. Should
                                seem pretty obvious that you ain't gonna get a mudslide without any hill for it
                                to slide down on."" "
                End depend.
                Include Geezer in talkers of landlord.
        End verb.
End the.
```

We put the Include statement inside the `T` verb override so that the topic will be added to Geezer's topic list when the reader/player reads Geezer's reply to the mudslides topic. This technique of using Include/Exclude on the Talkers attributes inside a `T` verb override allows you to set up the conversation tree of an actor so that it dynamically grows or shrinks and/or branches during a conversation depending on which topics the reader/player chooses.

Creating text for multiple NPCs

As mentioned earlier, the text that the reader/player reads for each topic is stored in the various objects you are using as topic objects. You do this storage by creating blocks of texts within an override for the `T` verb in each of the objects, and associating each block of text with a particular NPC so that each NPC can have a different opinion about a topic.

Although we don't have more than one NPC in the example for this guide, a topic in a game where multiple NPCs can each talk about his or her mother could look something like this:

```
The mother isa topic
        Name mother Name mom
        Has talkers {Geezer, Harkan, Ginny}.
        Verb t does
                Depending on listener of hero
                        = Geezer then
                                " ""Please tell me about your mother,"" you say to the old man.
                                $p""My mom died several years ago,"" he replies."
                        = Harkan then
                                " ""Your mother must be proud of you,"" you tell him.
                                $p""I am demon spawn and have no mother,"" he snarls."
                        = Ginny then
                                " ""How is you mother doing?""
                                $pGinny shakes her head and looks down at the floor without speaking."
                End depend.
        End verb.
End the.
```

The source code for this example

The QuickStrtGame.alan file contains the complete source code for the example used in this Guide.

A custom About command

It is customary to give the reader/player an About command that displays the non-standard IF commands that are required to play the game. This is one that you can use and modify as you see fit:

```
Verb 'about'
       Does
              Style normal.
              "TALK TO <name>, TT <name>, and TALK <name> all initiate a conversation with
              an NPC and display the list of topics that you can currently talk about.
              $pT <topic> displays the NPC's response for a particular topic.
              $pFor example:"
              "$n$i> tt jim
              $i ""Hey Jim,""  you call out to Jim.

              $n$i ""What?""  he replies.

              $n$i ""How can I catch crabs like you?""

              $n$i ""That's easy,"" he replies. ""What do you want to know first?""

              $n$i$t -- (topics) --
              $i$t$t traps
              $i$t$t bait

              $n$i$t -- (advice and hints) --
              $i$t$t Advice
```

```
            $n$i> t bait
            $i$t""I always use rotten chicken carcasses."" "
--          Style normal.

            "$pAs new topics become available over the course of a conversation, they
            will be "
            Style alert.
            "emphasized"
            Style normal.
            "in the current text with bold, or italics, or in some other manner,
            depending on your interpreter.
            $pNOTE: Since interpreters do not respond the same way to text formatting,
            this demo game uses ALL CAPS to emphasize the new topics just to play it
            safe.
            $pTo see the new topic, just use
            the T <topic> command with the emphasized text as the topic.

            $pYou can type either 't' or 'topics' or 't topics' when you need to remind
            yourself what topics your current listener can respond to, including the new
            topics that become available over the course of a conversation.

            $pNOTE: The ASK/TELL conversation system is NOT used in this game."
End verb.
```

Where to go next

The comments in Conversation.i are a good place to go next. It might be a good idea to read through them to see all that the extension provides before you begin using it. Also, you can look through the source code in The_Interview.alan to see examples of how to code everything the Conversation extension can do. Also, you can compile The_Interview.alan to play a demonstration game that incorporates the full range of the Conversation extension's capabilities.