Dialog Handler

Overview

The dialog handler is a used to create dialog that support basic branching behavior. If you want to do straight forward dialogs that have no branching at all, you might as well just call the Voice Handler directly. But if there are some branching, even simple like breaking the dialog if the player stop looking at a character, then the Voice Handler is what you want.

Basic functioning

The dialog is setup just before it is a about to get played, so there is not special init (other than what you do for the voice handler). You begin by calling Dialog_Begin() which makes the initial preparations and allows you to call the other functions. After that it is time to add a branch using Dialog_AddBranch. You can then use Dialog_AddSubject to add voice subjects to that branch. Once you add a new branch, then those subjects will belong to the newly added branch. When all branches and subjects are added, call Dialog_End to run the dialog. Dialog_End will also check which characters are used in this dialog, and by using Dialog_CharacterIsActive you can check which are currently in the middle of a dialog.

Example:

```
Dialog_Begin();
    Dialog_AddBranch("A");
        Dialog_AddSubject("AdamEnter1");
        Dialog_AddSubject("AdamEnter2");
    Dialog_AddBranch("B");
        Dialog_AddSubject("AdamLeave");
Dialog_End();
```

This will create two branches, “A” and “B”. The first contains “AdamEnter1” and “AdamEnter2”, and the other “AdamnLeave”. When Dialog_End() is called, “A” will begin playing, starting with “AdamEnter1” as it was the first added. If you want to start with some other branch, just supply its name as a parameter for Dialog_End(), like this:

```
Dialog_End("B")
```

This would start with branch “B” instead.

Also of interest is that you can make one branch go directly to another by supplying the next dialog as a second parameter, like this:

```
Dialog_AddBranch("A", "B")
```

Now when “A” is done, “B” is started. This might seem a bit pointless at first, but is really useful when you add branching events (more on that soon).

Branch Events

These events are needed in order to get the branching behaviors you are after. Events are added after having added subject and will belong that subject. They can in two types `LineEvents` and `EndEvent`. The first type is checked after each voice line has completed playing. The second is checked when the entire subject is over.

The existing functions carry this syntax:
`Dialog_Add[type]_[condition]` *(more on conditions in a few lines)*
Some examples for functions:

```
void Dialog_AddLineEvent_Callback(const tString&in asCallbackFunc, const
tString&in asNewBranch)
void Dialog_AddEndEvent_PlayerNotLookingOrOutOfRange(const tString&in
asNewBranch)
```

Note that if asNewBranch is “”, then the dialog does not jump to a new branch but just stops.

The different conditions available are:

OutOfRange
This is true if the player is out of range, as set in Voice_SetSource(..) with the PlayerListeningRange parameter. This applies to all characters in the dialog with a proper source entity and is only true if all of them are out of range.

PlayerNotLooking
This is true when the player is no longer looking at the source entity set with Voice_SetSource(..). This applies to all characters in the dialog with a proper source entity and is only true if all of them are not looked at.

Callback
This is a custom callback that can be used to let what ever code determine if there should be a branch or not. The syntax is:
`bool MyFunc(const tString&in asBranch, const tString&in asBranchSubject, int alLineIndex, const tString&in asNewBranch)`
asNewBranch is very handy when you do not want too many callbacks function to provide similar functionality. For example if the branch should be determined by the name of a cat, then you can do a function like this:

```
bool MyFunc(const tString&in asBranch, const tString&in asBranchSubject, int
alLineIndex, const tString&in asNewBranch)
{
  return asNewBranch == msSomeSavedVariableOfCatName;
}
```

The declared events can then look like this:

```
Dialog_AddLineEvent_Callback("MyFunc", "Pussy");
Dialog_AddLineEvent_Callback("MyFunc", "Shaggy");
```

Now for some full test source. Lets say that we have a dialog that should stop playing whenever the player stops looking at the character, then we do like this:

```
Dialog_Begin();
    Dialog_AddBranch("A");
        Dialog_AddSubject("AdamEnter1-2");
            Dialog_AddEndEvent_PlayerNotLooking("B");
    Dialog_AddBranch("B");
        Dialog_AddSubject("AdamLeave");
Dialog_End();
```