Dialogue System for Unity

Textline Project Template

Updated: 2019-01-30

The Textline Project Template is a mobile-style game that plays out like a series of text messages, similar to Three Minute Games' *Lifeline* series.

This template assumes you already have basic knowledge of Unity, the Unity UI system, and the Dialogue System.

Instructions

1. Import the Dialogue System.

2. Import the Menu Framework package, available on the Dialogue System Extras web page.

3. Import the Textline package, also available on the Dialogue System Extras web page.

4. Add the scenes in Assets / Dialogue System Extras / Textline / Scenes to your project's build settings in numerical order (0, 1, 2, 3).

5. The project's dialogue database is assigned to the Dialogue Manager. It contains one conversation, titled "Main Conversation". Customize this conversation to hold your content.
   - To end your conversation, set the last dialogue entry node's Sequence to:
     `WaitForMessage(Forever)`
   - If you want to force the game to return to the main menu instead, set the Sequence to:
     `SendMessage(ReturnToTitleMenu,,Menu System)`

6. Customize the appearance of the Dialogue Manager's dialogue UI. The dialogue UI uses a customized version of the Dialogue System's Standard Dialogue UI.
   - If you want to change the heading to the conversation title (especially useful if you're using multiple conversations), assign Heading Panel > Text to the Heading Text field.

7. Customize the appearance of the scene UIs.
   - To add support for different languages (localization) to the menus, edit the Text Table found in the Data subfolder. Then add those language names to the Menu Framework's LanguageDropdown in the OptionsPanel.

Pre-Delay Settings

By default, the Dialogue Manager's Default Sequence and Default Player Sequence play a "message received" audio tone and delay for a number of seconds after showing the subtitle.

If you instead want to delay *before* showing the subtitle:

1. Inspect the dialogue UI's NPC Pre Delay Settings and/or PC Pre Delay Settings. Tick *Based On Text Length* to pre-delay based on the Dialogue Text length and the Dialogue Manager's Subtitle Settings > Subtitle Chars Per Second. Set *Additional Seconds* to delay a set duration, which may be on top of the text length delay.
2. If you left *Based On Text Length* unticked and set *Additional Seconds* to a non-zero value, set the Default Sequence (or Default Player Sequence) to:
   `Audio(ReceiveText)@2`
   where 2 is the value of *Additional Seconds*.
3. If you ticked *Based On Text Length* and left *Additional Seconds* at zero, set the Default Sequence (or Default Player Sequence) to:
   `Audio(ReceiveText)@{{end}}`
4. If you ticked *Based On Text Length* and set *Additional Seconds*, set the Default Sequence (or Default Player Sequence) to:
   ```
   Delay(2)@{{end}}->Message(PlayNow);
   Audio(ReceiveText)@Message(PlayNow)
   ```
   where 2 is the value of *Additional Seconds*.
5. If you want to show an icon during the pre-delay period, such as an animated "..." symbol to suggest that the other person is typing, assign it to the Textline Dialogue UI's NPC/PC Pre Delay Settings. The Dialogue Manager prefab is already configured to show a basic "..." icon when the NPC is "typing."

Multiple Conversations

To support multiple conversations, such as a lobby system or a menu of people to text with:

1. In the start scene, inspect Dialogue Manager > Canvas > Dialogue UI. Tick **Use Conversation Variable**.
2. Add UI buttons to your start or lobby scene. On each button, add a Dialogue System Trigger set to *OnUse*. Configure it to play a sequence like this:
   `SetVariable(Conversation,`*`title`*`);`
   `LoadLevel(3 Gameplay)`
   where *title* is the title of the conversation to play. Configure the UI button's OnClick event to call DialogueSystemTrigger.OnUse. If you are defining this UI button in the Gameplay scene, add a MenuUtility script, and configure the OnClick event to call MenuUtility.StartConversation instead; you don't need to add a Dialogue System Trigger.
3. In the 3 Gameplay scene, configure a Dialogue System Trigger to fire *OnStart*. Add a MenuUtility component to the GameObject. Set the Conditions to require that `Variable["Conversation"] == "`*`title`*`"`. Add an OnExecute() event that calls the MenuUtility component's StartConversation method, and specify the conversation title. Repeat for every conversation that you set up in step #2 above.
4. Note: In the 3 Gameplay scene, the "Start Conversation" GameObject has a Condition to require that `Variable["Conversation"] == nil`. This way it will start the Main Conversation if no conversation is specified.