Conversa Dialogue System

Intro

Conversa is a system created for Unity to allow developers to design and manage the conversations and dialogues inside their games, with flexible tools that make the experience pleasant and enjoyable. To achieve this, it uses a graphic interface, similar to the one used for ShaderGraph. Management of nodes gives to the user an easy and pleasing mechanism to create, organize and reorder their conversations at any time, without any Hassle

Creating your first conversation

To create your first conversation, open the "Create" menu, by right clicking inside the folder where you want to place your conversation, and select:

Create > Conversa > Conversation

A file will be created, containing the configuration for your conversation. Double-click the file, and the graph editor will open.

Node types

There are different kind of nodes that you can use to model your conversation dynamically, to build something beyond a static dialogue.

Message node

A message node is the most basic and important block of any conversation.

Fields

- Actor: This is a string that you can use to identify the owner of the message
- Message: This field represents the message that you want to show when displaying this part of the conversation

Ports

- Previous: This port connects to the node that was executed right before.
- Next: This port connects to the node that will be executed right after.

Choice node

This node is typically used to present the user with choices, which may lead the conversation in different directions.

Fields

- Actor: This is a string that you can use to identify the owner of the message

This node is special, because the amount of fields and port can be dynamically set. You can have as many choices as you wish. Each choice will have a field, with the text that such option represents, and a port, that will lead you directly to the next node, if that option is taken.

Booleans

A boolean node is a node that represents a value that is either true or false. This can be used together with Branch nodes, to divert the conversation in different routes, depending on external parameters.

Booleans can be set inside the main blackboard. Click on the "+" sign in the top right position, and select "Boolean". After doing this, a new item will appear in the section "Booleans". You can double click into this item to rename it.

Once the boolean is set, you can drag it into the graph, to use it. You can create multiple instances of the same boolean all over the graph.

Branch

Branch nodes can be used to divert the conversation into different directions, depending of some external conditions.

Fields

- Previous: This port connects to the node that was executed right before.
- Condition: This port connects to a boolean, that will decide which path will be taken.
- True: This port connects to the node that will be executed right after, in case the condition is true.
- False: This port connects to the node that will be executed right after, in case the condition is false

Events

Events are nodes that will help you trigger callbaks inside your game, if the conversation takes a certain path.

For example, you could create an event "SkillLearned" that will let you know to update your game, to display the available skils for a player.

Events can be set inside the main blackboard. Click on the "+" sign in the top right position, and select "Event". After doing this, a new item will appear in the section "Events". You can double click into this item to rename it.

Once the event is set, you can drag it into the graph, to use it. You can create multiple instances of the same event all over the graph.

Bookmarks and bookmark jumps

Bookmarks are a special feature that will help you organize your nodes in a cleaner way. If you have to make a connection between 2 nodes, but the nodes are too far apart, or there are too many nodes inbetween, bookmarks can help you make those transition, without having a messy layout.

There are 2 kind of nodes for this:

- **Bookmarks nodes:** This nodes are used to mark points inside your conversation where you might want to jump back in, at any time.
- **Bookmark jump nodes:** This nodes are used to jump to the bookmarks that have already been defined inside the graph.

To create a bookmark, right click inside the graph, and select the node "Bookmark". After doing this, 2 things will happen:

- A new item will appear in the section "Events". You can double click into this item to rename it.
- A new node will appear in the graph, which can be used to set up the position of the bookmark.

Once the bookmark is setup, you can place "jump nodes" inside the graph, to make the flow of the conversation return to the point where the bookmark is set All you need to do is drag the entry from the blackboard into the graph, to instantiate jump nodes. You can create as many as you wish, and place them anywhere you want.

Using your conversation in your game

The API to use the conversation is very simple. Here is an example:

```
using EMT.Conversa.Runtime;

public class DialogueRenderer : MonoBehaviour
{
    [SerializeField] private Conversation conversation;

    private void Start()
    {
        var runner = new ConversationRunner(conversation);

        // Setting default callbacks
        runner.OnMessage.AddListener(HandleMessage);
        runner.OnChoice.AddListener(HandleChoice);
        runner.OnEnd.AddListener(HandleEnd);
        runner.OnUserEvent.AddListener(HandleEvents);

        // Setting booleans
        conversation.SetProperty("Dummy", false);

        runner.Begin();
    }
```

First of all, you want to wrap the conversation object inside a `ConversationRunner`. This will give you a tool to move a cursor through your conversation, making it follow a certain flow.

After that, there are 4 important callbacks that should be set:

- `OnMessage` will set the callback to handle when a Message node gets activated.
- `OnChoice` will set the callback to handle when a Choice node gets activated.
- `OnUserEvent` will set the callback to handle when custom events get activated.
- `OnEnd` will set the callback to handle when the conversation reaches its end.

These are callbacks for each kind of event:

```
private void HandleMessage(MessageEvent e);
private void HandleChoice(ChoiceEvent e);
private void HandleUserevent(UserEvent e);
private void HandleEnd();
```

MessageEvent

Field	Description
`string Message`	Message to be displayed
`string Actor`	Actor linked to the message

Field	Description
`Action Advance`	Function to advance the cursor through the graph

ChoiceEvent

Field	Description
`List<Option> Options`	Message to be displayed

Option

Field	Description
`string Message`	Message to be displayed
`Action Advance`	Function to advance the cursor through the graph

OnUserEvent

Field	Description
`string Name`	Name of the event fired

The graph contains boolean variable that can be used to change the flow of the conversation. To set this variables, it can be done with `SetProperty`.

```
conversation.SetProperty(string nameOfProperty, bool value);
```