RPG Conversation Editor Manual

Creative Spore

Support: support@creativespore.com

Web: http://www.creativespore.com

Unity Forums: user CreativeSpore

Twitter: @CreativeSpore

Index

1 Introduction

This is the user manual for the Unity asset RPG Conversation Editor.

RPG Conversation Editor has been designed to be user-friendly and easy to use.

This manual will teach you how to create a conversation for an NPC in an easy and fun way.

2 Asset Folder Structure

- **RPGConversationSystem**
  - **Extra**: this folder is optional and can be removed. Includes some demo examples and resources to start using RPG Conversation Editor.
    - **DemoScenes:** contains the demo scenes with example of use.
    - **DialogBackgrounds:** contains the example dialog backgrounds.
    - **DialogPrefabs:** contains the example dialog prefabs.
    - **Documentation**: contains the manual and code docs.
    - **Fonts:** contains the example dialog font.
    - **Gfx:** contains some graphics used in the example demo scenes.
    - **Scripts:** contains scripts used in the demo scenes.
    - **Sounds:** contains sounds used in the demo scenes.
  - **Scripts:** contains all the scripts necessary for this asset to work.

3 Getting Started

If you are really anxious to use the tool and have no time for reading all the manual, this will give you a fast introduction to start creating conversations for your games.

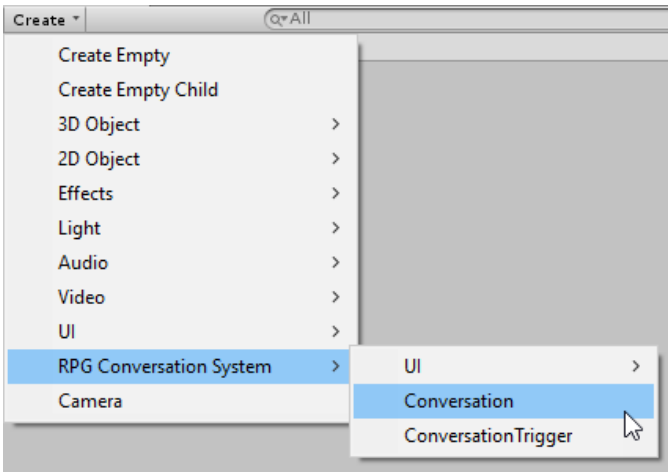
3.1 Creating a Conversation

To create a conversation, you need to create a gameObject in the Scene

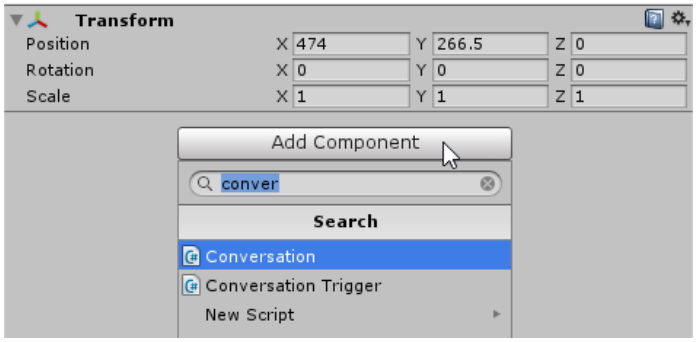
with the component ConversationController.

There are two ways to do this:

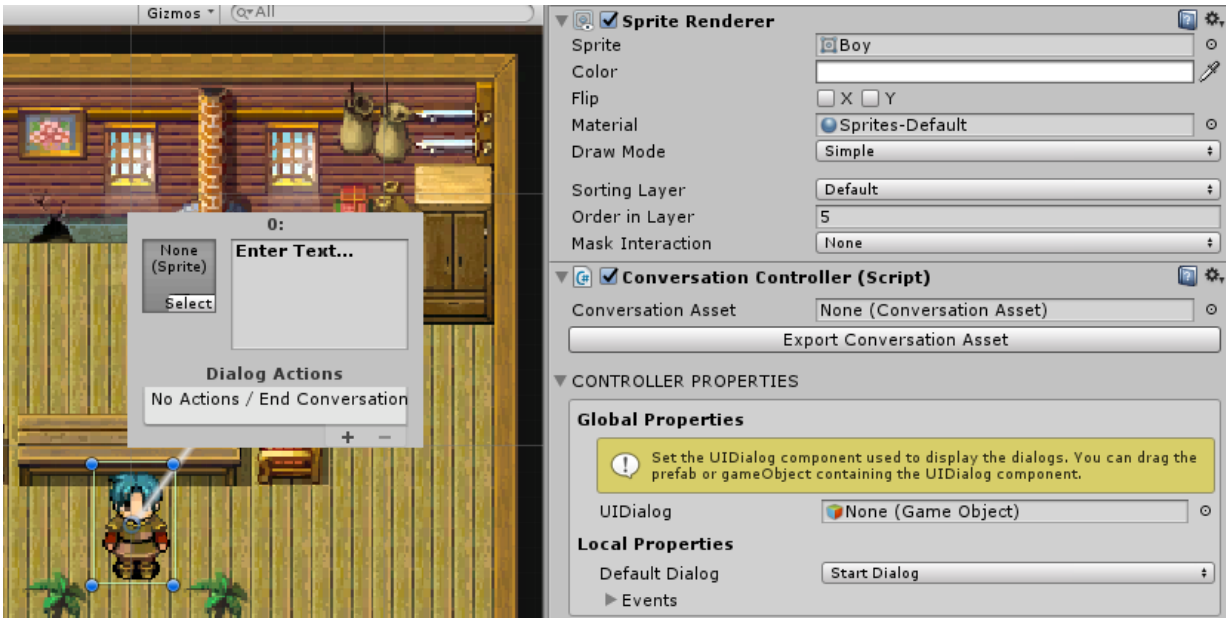
- Creating a conversation gameObject:



- Adding the ConversationController component to a gameObject:



Once you have added the component, a first conversation will be created with a single dialog.

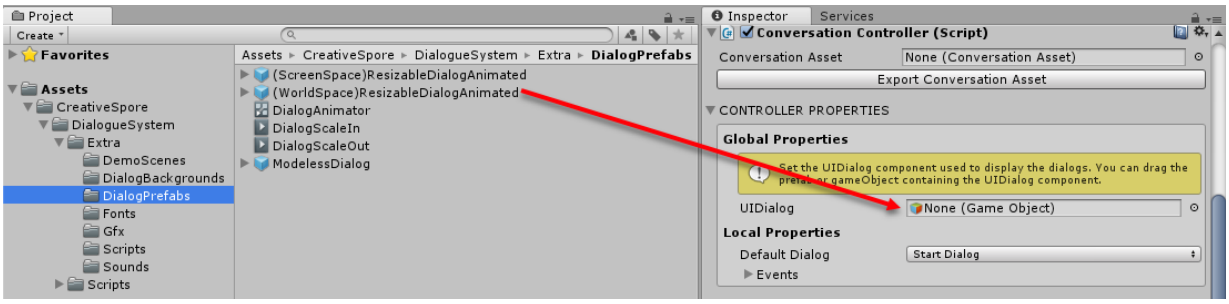


A Conversation Controller component contains a list of conversations that contains a list of dialogs that contains a list of sentences an actions (or dialogue answers).

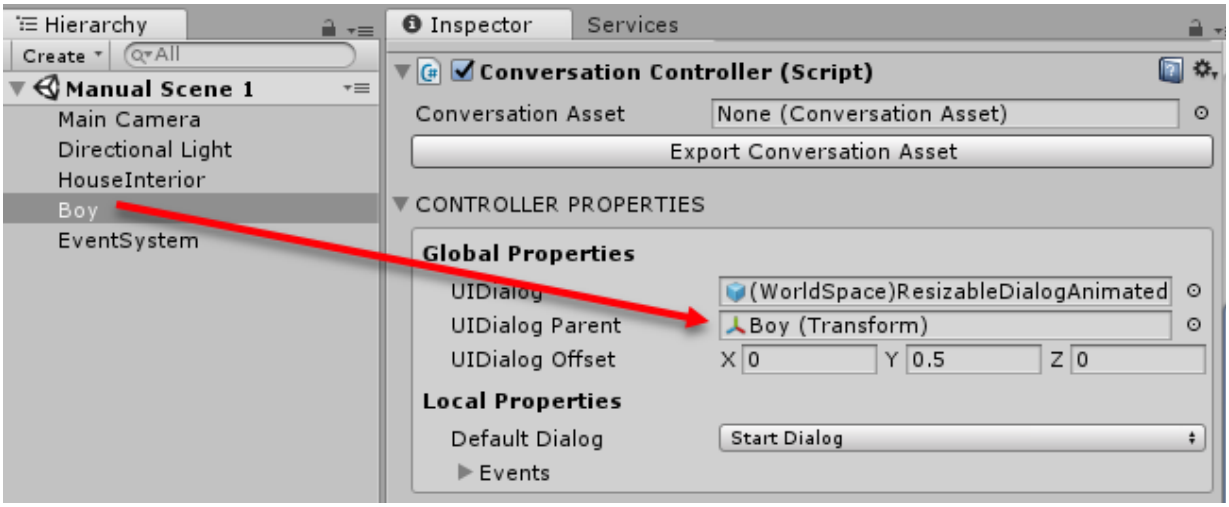
This component only holds the logic part of a conversation, not the visual part. The dialog data will be displayed in the game using a Canvas object with the component UIDialog attached that will place each dialog element in its place. So, in order to display this dialog, you need to select a prefab or gameObject in the scene with the UIDialog component in the UIDialog property. For now, we will select the included prefab dialog "**(WorldSpace)ResizableDialogAnimated**".

This prefab is configured to display a UIDialog with all possible elements in the World Space.

Drag the prefab into the UIDialog property.



You can also define the parent transform for this UIDialog and the offset position it should appear when the conversation is started:

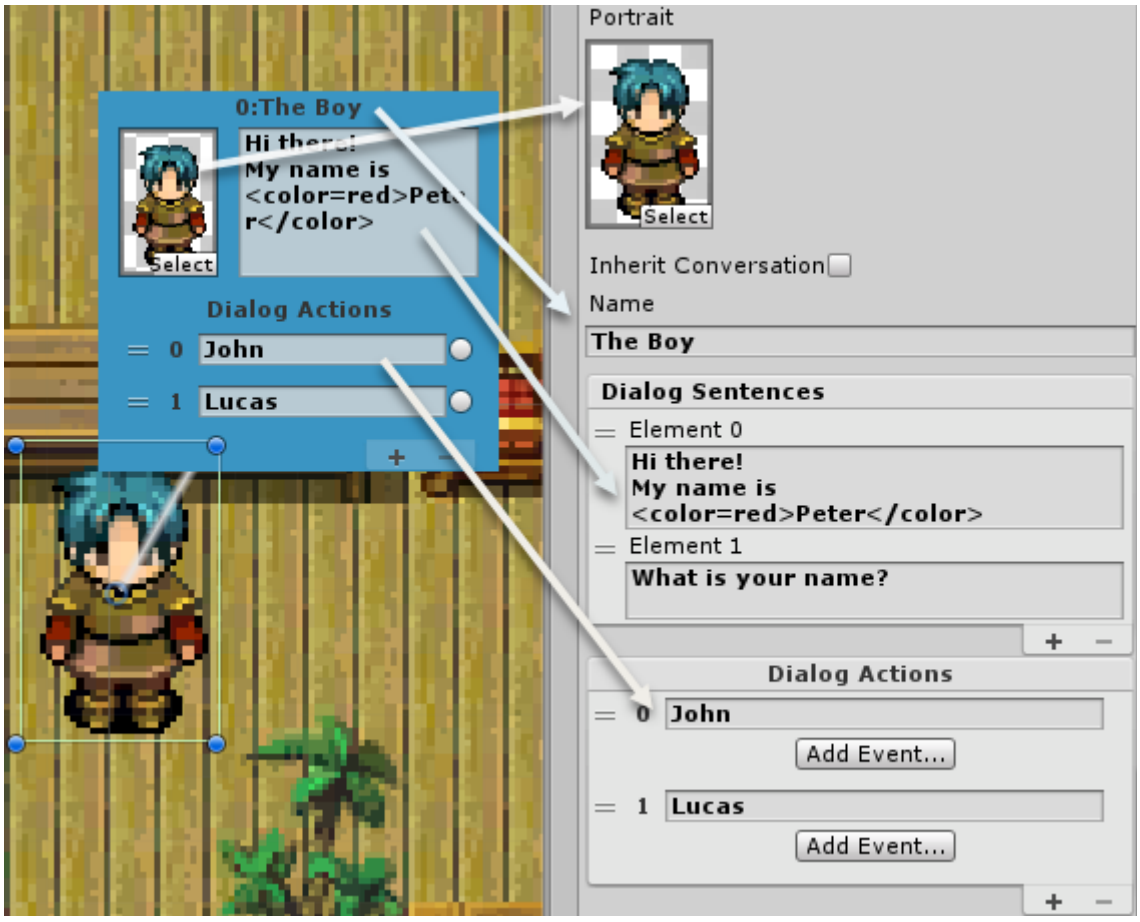


Once you have defined a UIDialog to render the conversation dialog, you will see a preview of the dialog in the Selected Dialog Properties and also in the gameObject preview:

Now, you can change the dialog data directly in the Scene View dialog or

in the Inspector:

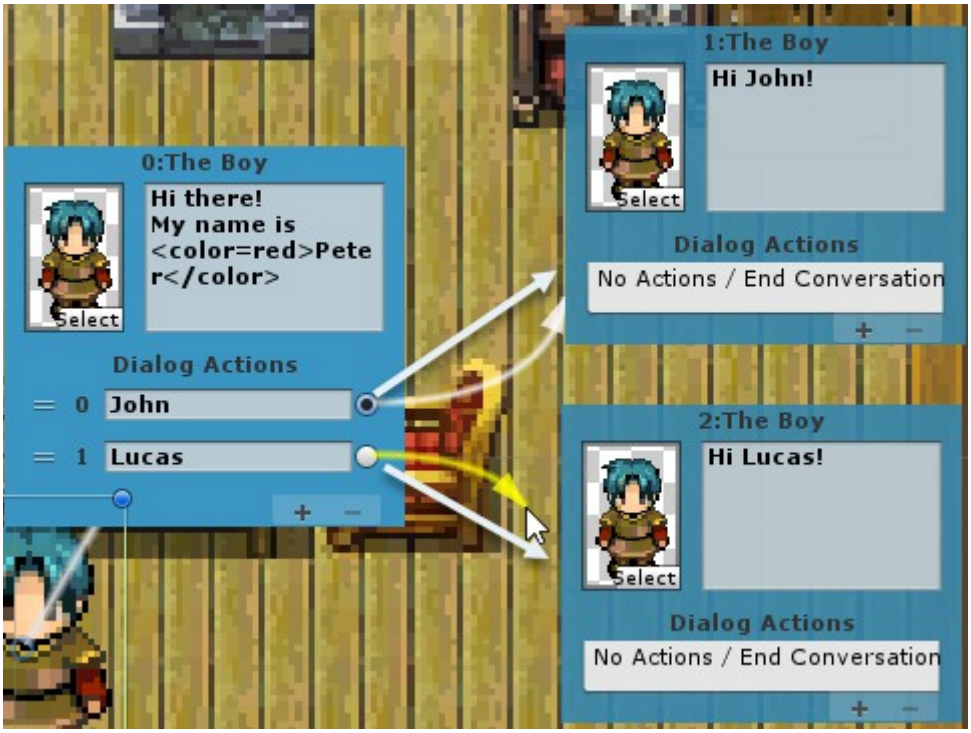
- **Portrait**: the sprite displayed for this dialog portrait
- **Name**: the name of the character (can be changed only in the Inspector). It can be inherited from the conversation name.
- **Dialog Sentences**: the sentences for this dialog. Can be more than once. In this case, each time you click over the text, the next sentence will be displayed. When the last sentences is reached, the dialog actions will be displayed.
- **Dialog Actions**: these will be the answers for this dialog that will lead to another dialog and/or trigger an event. To add or remove actions, you can use the '+' and '-' buttons.



Now, lets add two new dialogs for the two actions.

To add a new dialog, you can right click over the Scene View and select Add Dialog:

To connect a dialog action with another dialog, click over the small circle right to the action and drag the mouse to the target dialog:

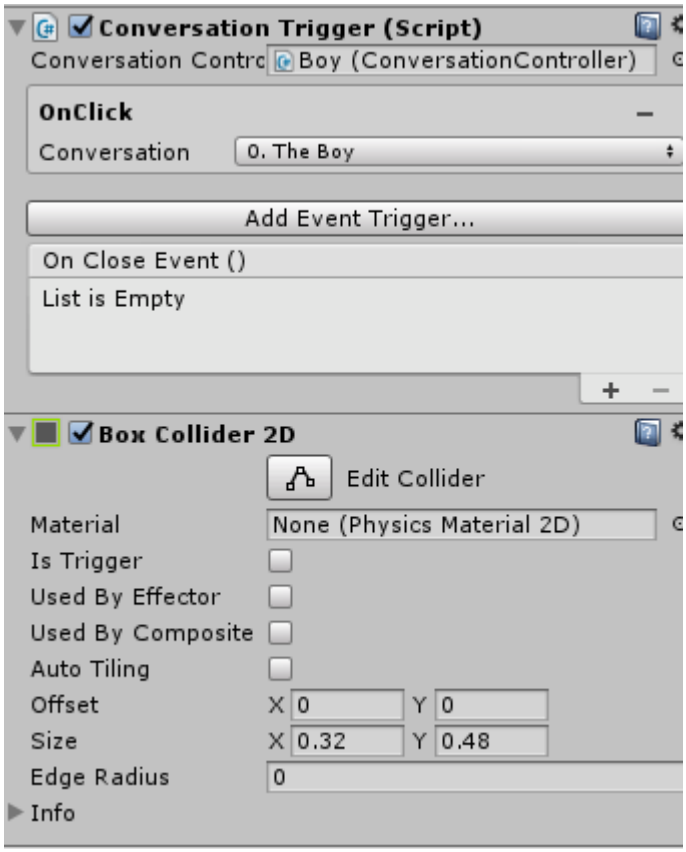


One more step before you test the scene and the conversation.

To start a conversation, you need to add a conversation trigger and select what event will start the conversation. In this case, we will use the OnClick event.

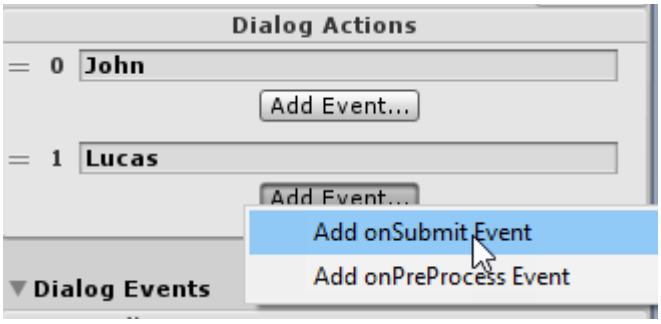
Add a ConversationTrigger component, press “Add Event Trigger...” and select Input->OnClick.

Also add a Box2D component to detect the OnClick event when clicking with the mouse or pointer.

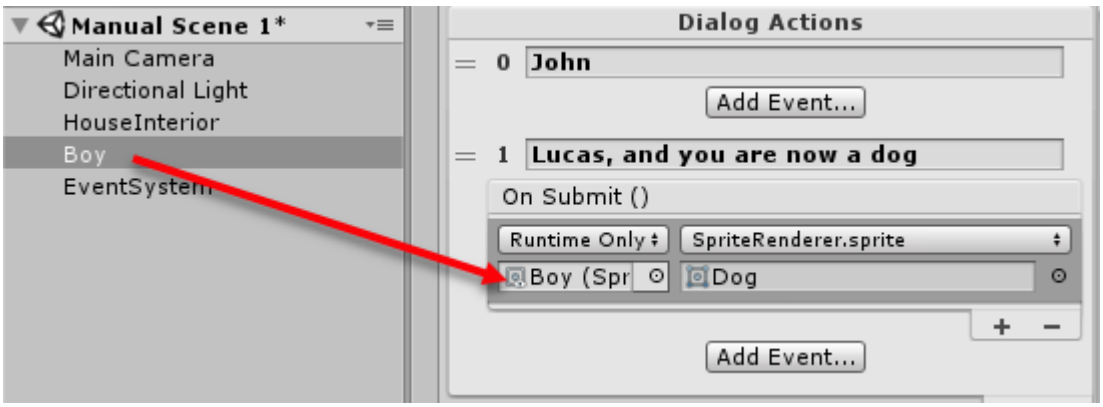


Finally, you can use different Unity Events to call a method and perform actions when a dilaog is close, open, or when clicking an action.

Let's add for example an event when clicking over one of the actions using the onSubmit event:



Now, we can change, for example, the sprite of the character for other sprite using Unity Events:



That's it. You should now be able to create a conversation in no time.

Check the section ConversationController for more information about how to create and configure conversations.

3.2 *Creating a new UIDialog for your custom dialogs*

In the previoud section we learned the logic how to create a conversation.

All the logic part of a conversation is hold by the Conversation Controller and the visual part is represented by the UIDialog component.

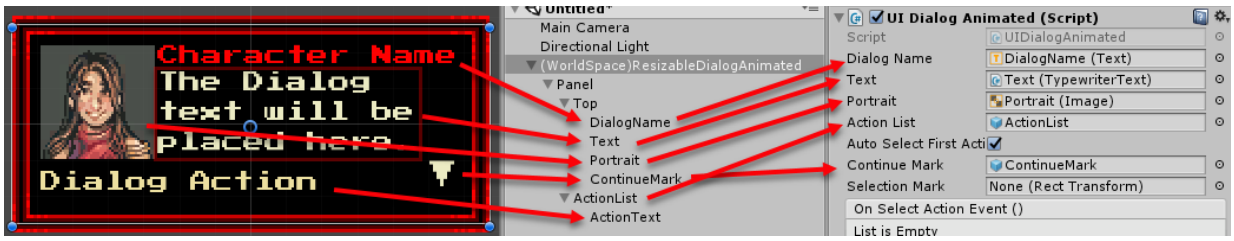
Here we are going to explain how to create your custom dialog using the included prefab.

The best way to create your custom UIDialog is to duplicate one of the included prefabs located in:

"**\CreativeSpore\DialogueSystem\Extra\DialogPrefabs**"

You can choose between the Screen or World dialogs. Screen dialogs will be placed over the screen and world dialog will be placed over the scene world. I will use here the **(WorldSpace)ResizableDialogAnimated** prefab.

The UIDialog component connects each UI element with the current played dialog.



You can create your own custom dialogs, or modify one of the prefabs creaing a duplicate (Ctr/Cmd + D while selecting the prefab).

Action List will be the parent object for the dialog action items. The templated used for these actions will be the first child item for this gameObject. Action List gameObject also uses a component VerticalLayoutGroup to organize the actions in a vertical list.

Text component is not a UI.Text but a TypeWriterText component that inherits from text and will display the characters like a typewriter. You can use here a Text component if you want.

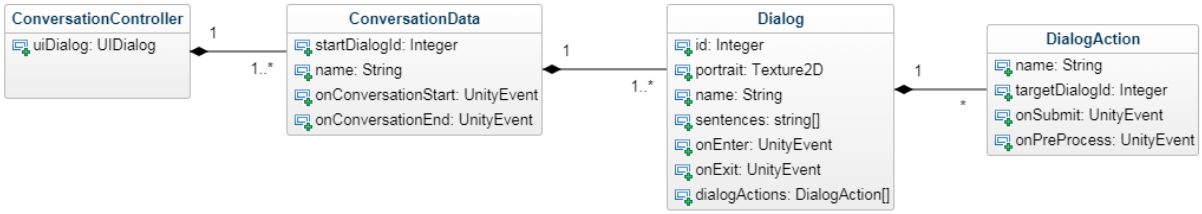
Continue Mark is an optional gameObject that will blink if the dialog has several sentences to indicate to click the text or press the submit button to

continue.

Selection Mark is another optional gameObject that will be placed over the selected dialog action, for example and arrow to select this action when using keyboard.

4 Conversation Overview

4.1 Class Diagram



Here you can see a simple class diagram with all relations between every class involved in a conversation.

ConversationController: will be the component that will contain and manage a list of conversations (held in a **Conversation Data**).

ConversationData: will contain a list of Dialogs, using a property startDialogId to define what dialog will be the first in the conversation.

Dialog: will contain the actual dialog data, like the portrait, the name of character (name), an array of sentences to split all the dialog in text chunks (sentences), two UnityEvents (onEnter and onExit) to be invoked when the dialog appears and disappears respectively, and a list of dialog actions that will be the answers of the conversation.

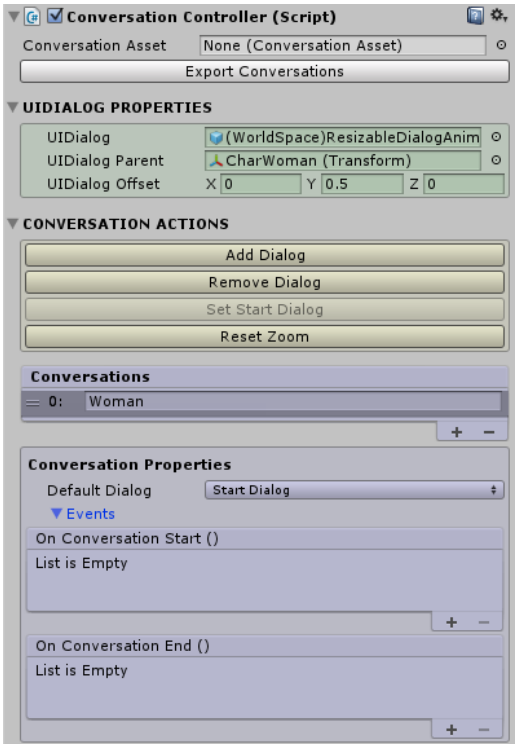
DialogAction: this will contains the action name (the text appearing as answer), a target dialog to open when this action is chosen, and two UnityEvents (onSubmit and onPreProcess). OnSubmit will be invoked when the action is submitted, and the OnPreProcess will be invoked when the action is being processed to appear in the UIDialog. You can use OnPreProcess to change the visibility, change the text, or any other property of the action before it is processed by UIDialog. This could be used to create conditional dialog actions, for example.

5 ConversationController (Component)

This component is the one containing all the conversations and is the one in change of starting the conversations.

This component also allow to edit the conversation dialogs in the scene, adding, removing and modifying dialog data.

5.1 ConversationController (InspectorView)



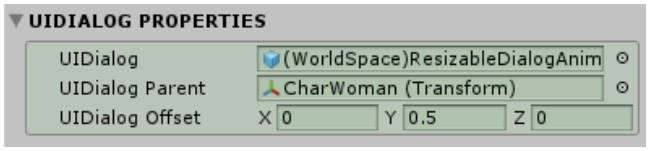
This is the inspector view for the ConversationController. Here you can change some global properties, affecting all the conversations like the UIDialog prefab or gameObject used to display the dialogs, or Conversation properties like the default dialog or some events.

Conversation Asset: you can export your conversations to an asset to be shared by other ConversationControllers. In this case, this property will refers to that asset in the project.

- **Export Conversations**: will export the conversations in this controller as an asset.
- **Embed Conversations:** will embed the conversation asset inside this

controller breaking the link with the conversation asset (but not removing it).

UIDialog Properties:



- **UIDialog:** the UI Dialog component used to draw this conversation dialogs. It can be a component in the scene or inside a prefab.
- **UIDialog Parent:** the parent gameobject that will be assigned to the UIDialog instance when the conversation starts. Usually used for World Space Canvases.
- **UIDialog Offset:** The local position set to the UIDialog instance. It will be relative to the UIDialogParent object.

Conversation Actions:

- **Add Dialog**: adds a new dialog in the selected conversation.
- **Remove Dialog**: removes the selected dialog.
- **Set Start Dialog**: sets the selected dialog as the start dialog of the conversation.
- **Reset Zoom**: sets the zoom to 100%

Conversations List: this is a reorderable list where you can add, remove or change the order of all the conversations.

Conversation Properties

- **Default Dialog:** this is the default dialog used when a conversation is started when the start dialog is not specified.
  - **Start Dialog**: the default option. The dialog sets as start dialog will be used.
  - **Random**: a random dialog will be used
  - **Random Non-Repeating**: a random dialog will be used but trying to avoid repetitions with previous chosen dialogs.

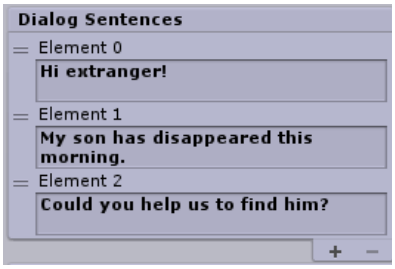
Selected Dialog Properties:

- **UIDialog Preview:** this is a preview of the UIDialog filled with the dialog data. You can also see a preview in the inspector preview.

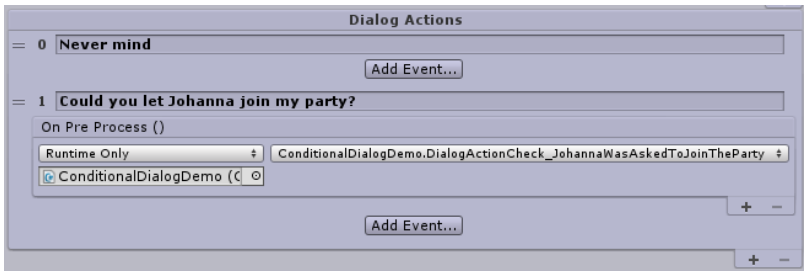
- **Position / Size / Color**: these are properties only for the dialog, not the UIDialog instantiated when the dialog is played.

- **Portrait**: this is the texture used for the dialog portrait.

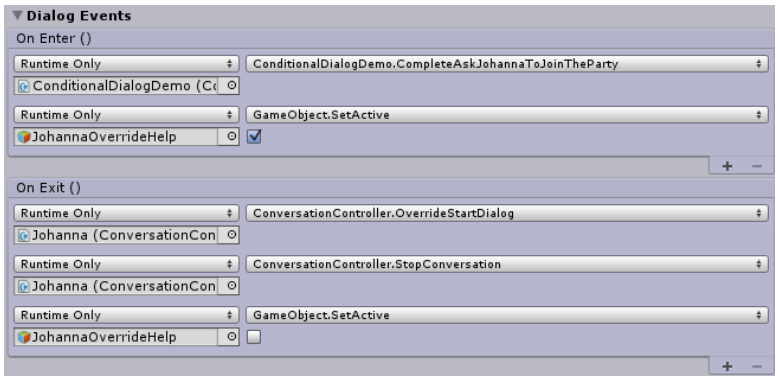
- **Inherit Conversation Name:** if this is checked, the conversation name will be used as name of the character for this dialog. Otherwise you can set the name in a name property.
- **Dialog Sentences:** here you can add/remove the sentences for this dialog. They will be displayed one by one when the dialog is played.



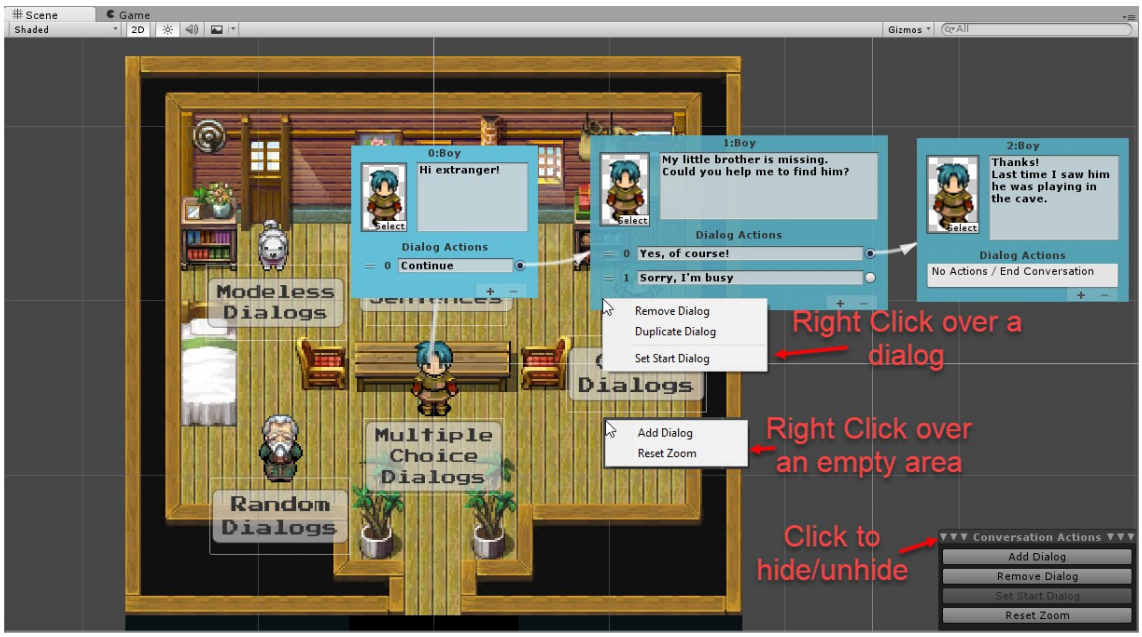
- **Dialog Actions:** the dialog answers will be added here. You can also add some events to these actions:
  - **onPreProcess:** when the UIDialog is about to process an action, this event is called. You can use this event to change, for example, the visibility according some requirements. Ex: `UIDialog.processedDialogAction.visible = AskJohannaToJoinTheParty;`
  - **onSubmit:** this event is called after the action is selected by clicking it or activating the submit input.



- **Dialog Events:**
  - **onEnter:** will be invoked when the dialog is played.
  - **onExit**: will be invoked when the dilaog is closed.


Dialog Events
On Enter ()
Runtime Only
ConditionalDialogDemo.CompleteAskJohannaToJoinTheParty
ConditionalDialogDemo (C
Runtime Only
GameObject.SetActive
JohannaOverrideHelp
On Exit ()
Runtime Only
ConversationController.OverrideStartDialog
Johanna (ConversationCon
Runtime Only
ConversationController.StopConversation
Johanna (ConversationCon
Runtime Only
GameObject.SetActive
JohannaOverrideHelp

5.2 *ConversationController (Scene View)*



In the scene view you can see and edit the selected conversation.

In the bottom right corner you can see the same conversation actions that you can see in the inspector, but closer.

You can **zoom in-out** with the mouse wheel.

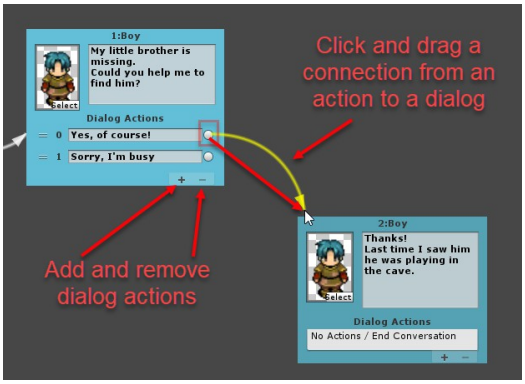
If you **right click over an empty area**, you can Add a new dialog or reset the dialog zoom to 100%

If you **right click over a dialog**, you can remove it, create a duplicate or set it as the start dialog in the conversation (there is a line from the owner gameObject to the start dialog).

You can **resize the dialog** moving the mouse to the dialog borders (check highlighted borders in the above image), holding the left mouse button and dragging.

Drag the dialog to any position by dragging the dialog header bar. If you hold Ctrl/Cmd while dragging a dialog, all dialogs will be dragged at once.

Edit the Dialog properties like Portrait, main text, and dialog actions.

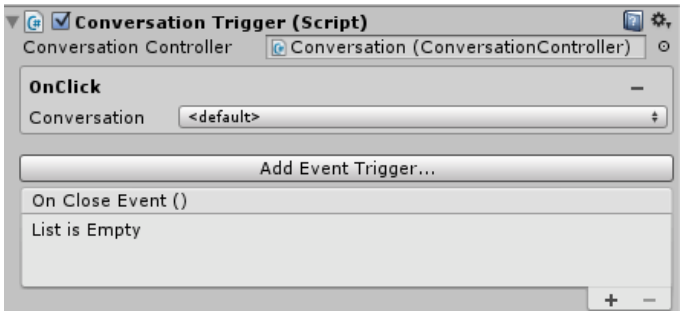


And **create connections** between a dialog action and another dialog by dragging a line from the right action circle to another dialog.

You can also **rearrange the dialog actions** dragging the left icon.

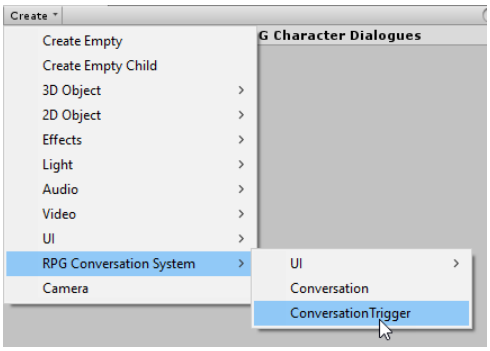
Remember that you can also change the dialog name, add more sentences to the text and add dialog events, and dialog actions events in the inspector view.

6 ConversationTrigger (Component)

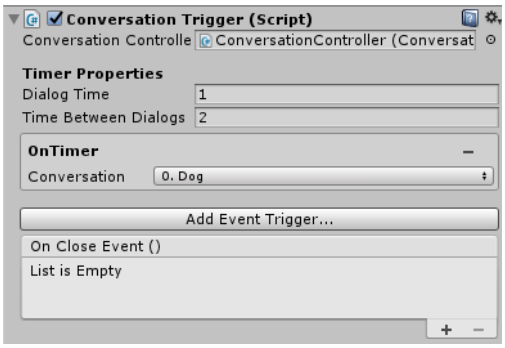


A conversation trigger is used to start a conversation when an event is triggered.

You can create a conversation trigger throught the menu "**GameObject->RPG Conversation Editor->ConversationTrigger**"

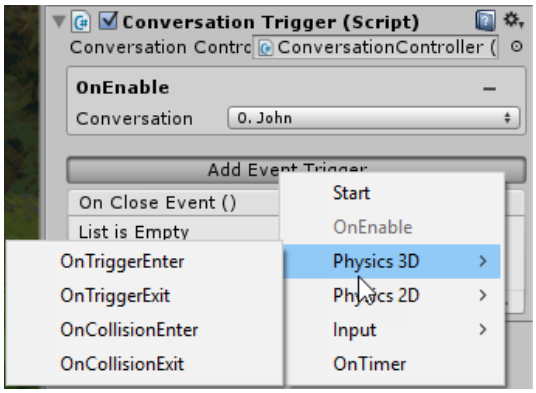


Or adding the component directly in a gameObject.



You need to select what conversation controller will be the target for this trigger.

Then press the “Add Event Trigger...” button to add an event that should start a conversation from the conversation controller.



Some events will add additional configuration properties, like onTimer.

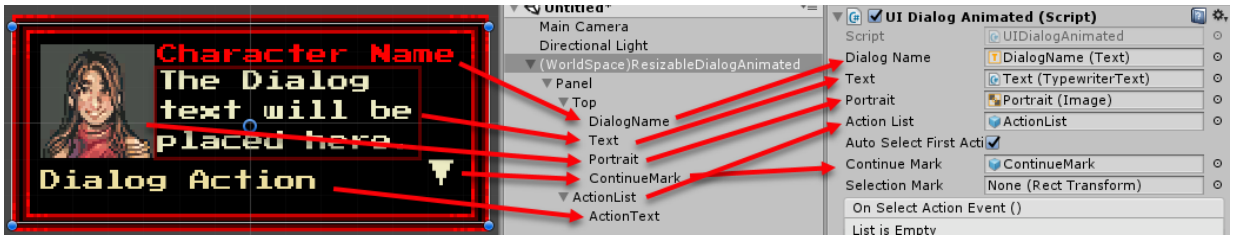
And other events will need another components to be added, like onClick.

Once you add an event, you can choose what conversation start when this event is triggered. If you choose default, the active conversation will be started.

OnCloseEvent will be invoked when the conversation is closed.

7 UIDialog (Component)

The UIDialog component is managing the visual part of a conversation dialog. In responsible of playing the dialog using a Canvas and UI elements for each dialog attribute.



There is another component called **UIDialogAnimated**. This component works the same as UIDialog, but also supports to play an fade-in/fade-out animation.

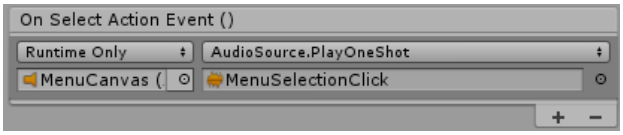
You can skip any property in case you don't need it and leave it empty.

UIDialog Properties:

- **Dialog Name:** a Text component used to display the dialog name (usually the name of the character).

- **Text:** a Text component used to display the dialog sentences.
- **Portrait:** a Image component used to display the portrait texture.
- **Action List:** this would be a parent gameObject, normally with a LayoutGroup component used to group the dialog actions in your preferred layout. The first child of this component should have a Text component to display the action name and also to be used as a prefab to create the needed extra actions.
- **Auto Select First Action:** check this to make the first action automatically selected when the dialog is played. Useful if you use keyboard to select the dialog actions.
- **Continue Mark:** this gameObject will be flashing (due activation/deactivation) when a dialog has several sentences to indicate there more text to be displayed.
- **Selection Mark:** this gameObject will be placed in the currently selected action position. Ex: an arrow pointing to the active action.

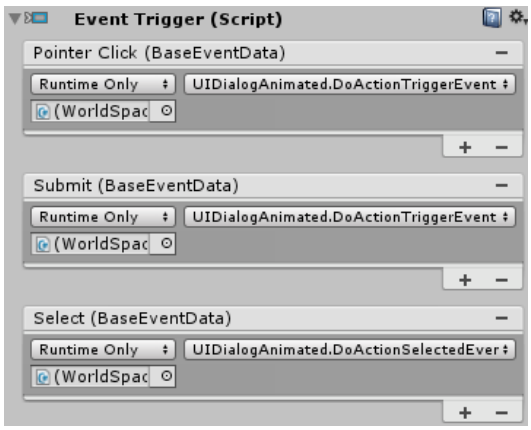
- **On Select Action Event:** this event will be invoked when an action is selected. Use this, for example, to play a sound.



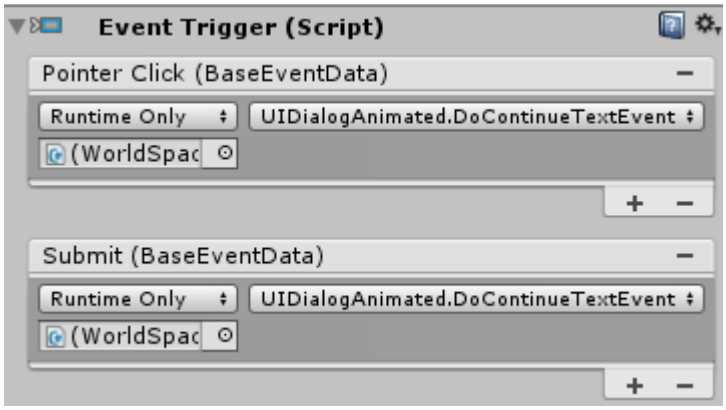
- **UIDialog Animated Properties** (only for **UIDialogAnimated** component):
  - **DialogAnimIn**: animation played when the dialog is played.
  - **DialogAnimOut**: animation played when the dialog is closed.

- **Add Default Event Trigger to Dialog Actions:** this will

automatically add some needed trigger events to the dialog actions. They are used to trigger the dialog action when the action is clicked or submitted(pressing the submit button) and also to trigger the action selected event when the action is selected.



- **Add Default Event Trigger to Text Component**: this will automatically add some needed trigger events to the text component. They are used to continue in the conversation when the text is clicked of the submit button is pressed and no action is displayed for this dialog.



8 TypewriterText (Component)

There is an special Text component included in the asset. It works like a **UnityEngine.UI.Text** component but adds an typewriter effect.

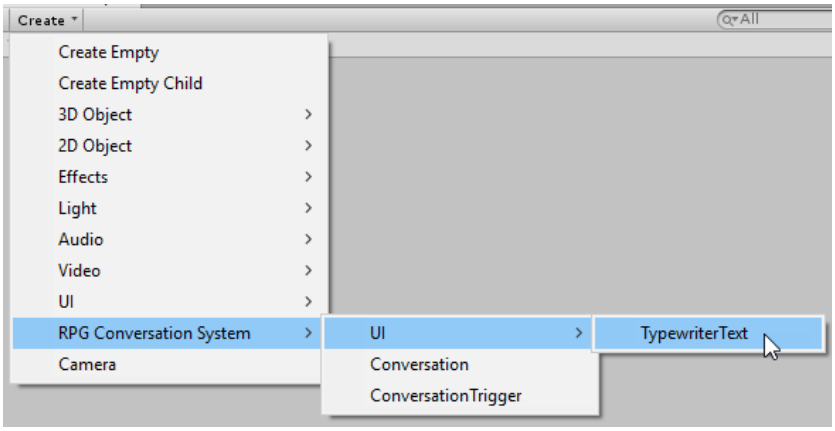
It inherits from **UnityEngine.UI.Text** so it can also be dragged to the UIDialog Text property.

This Text component will display the text (even with rich text) character

by character and its integrated with the UIDialog so when you click over the text, the full text will be automatically completed and only after it is complete the dialog will move to the next dialog.

It will also play a sound while the text is being typed and stops it after all text is displayed.

You can add a TypewriterText component to any canvas through the menu "**GameObject->RPG Conversation Editor->UI->TypewriterText**".



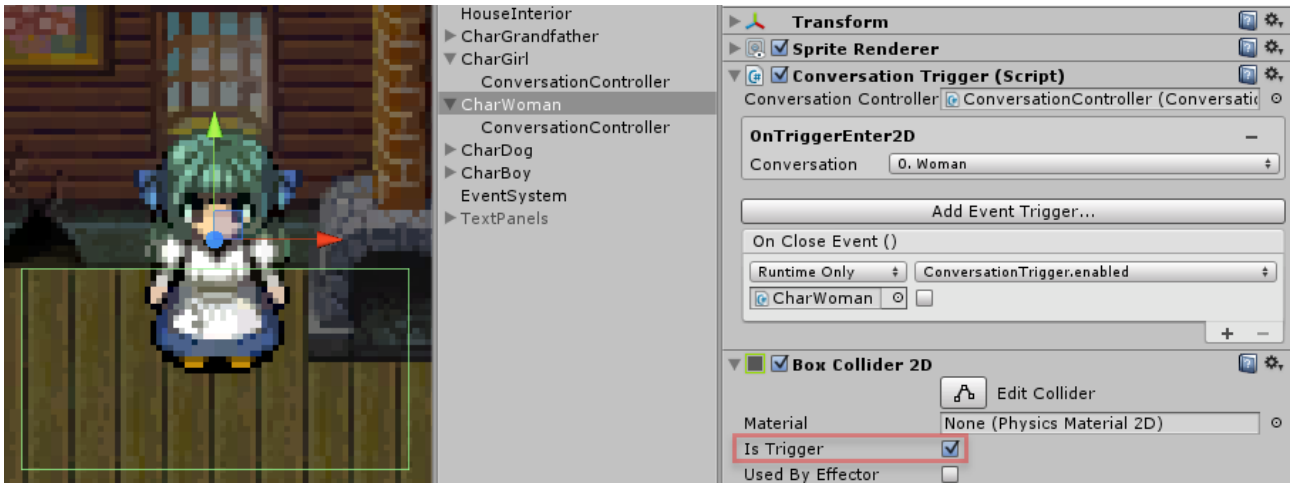
TypewriterText Properties:

- **Typing Speed:** the speed in characters per second.
- **Fill Amount:** The text fill amount percent with 0 for 0% and 1 for 100%.
- **Typing Sound:** the sound to be played in a loop while the fill amount is less than 1. If no AudioSource is added to the gameObject and this property is set, an AudioSource component will be added automatically.

9 Conversation Use Cases

Here there is a list of use cases in a game where you could use RPG Conversation Editor.

9.1 Start a conversation when the player is closed to an NPC

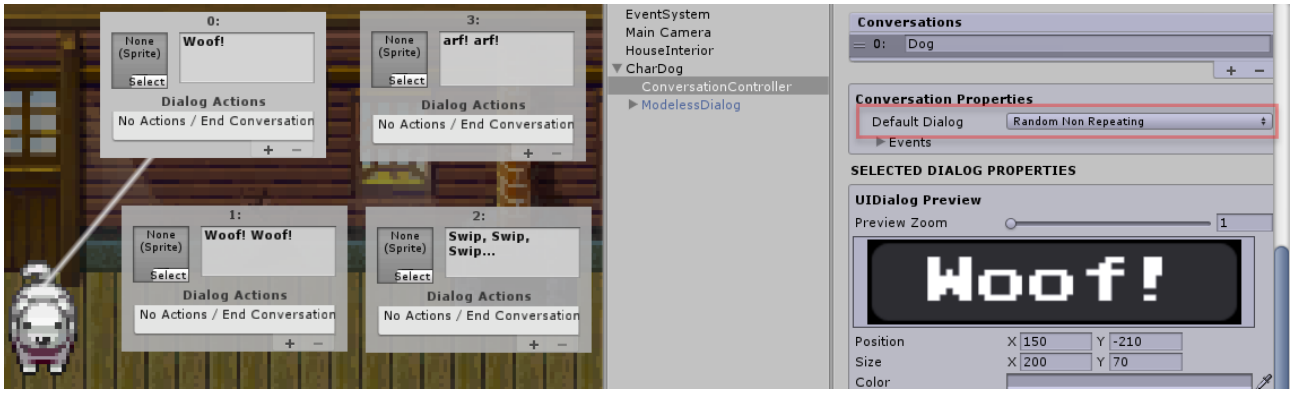


Add a **Conversation Trigger** to the character and also a conversation controller, like in the image.

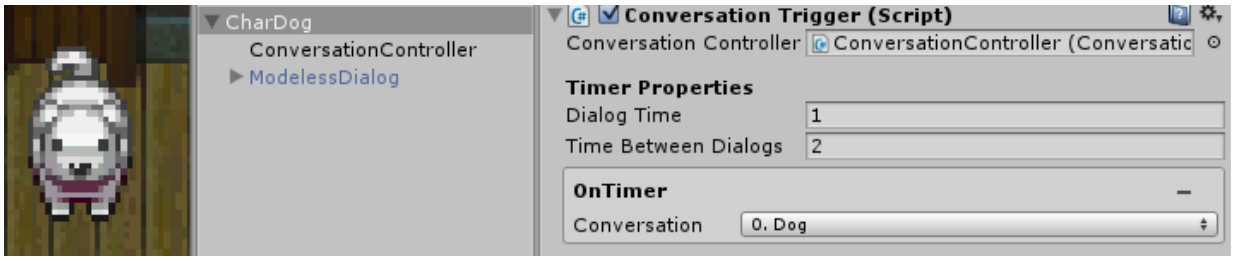
Add an event OnTriggerEnter2D to trigger the conversation when other collider enters the BoxCollider2D.

Optionally, disable the Conversation Trigger component when the dialog is closed to avoid repeating the conversation again.

9.2 Start a random NPC non-modal dialog automatically



Add some dialogs to the character conversation and set the **Default Dialog** to Random or Random Non Repeating.



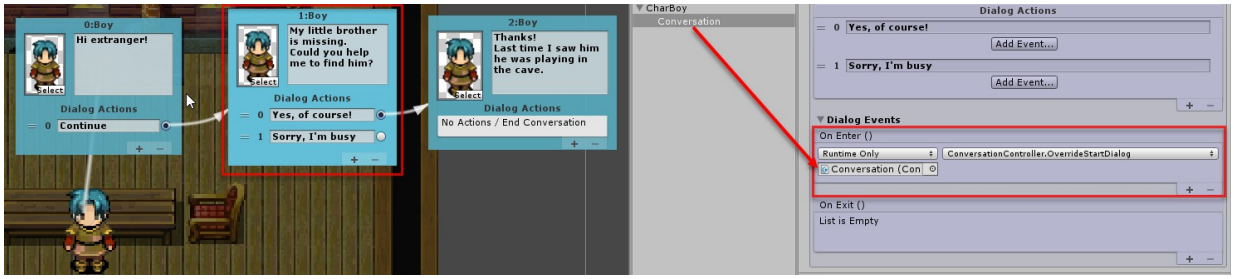
Add a **Conversation Trigger** to the character and add an onTimer event to start the conversation using a timer.

Each time the conversation starts, it will display a random dialog.

This use case is implemented in "**[RPG Conversation Editor] 1. RPG Character Dialogues**".

9.3 Continue a conversation from a previous dialog

For example, if you start a conversation and the NPC greets you, but the next time you don't want it to greets you again.

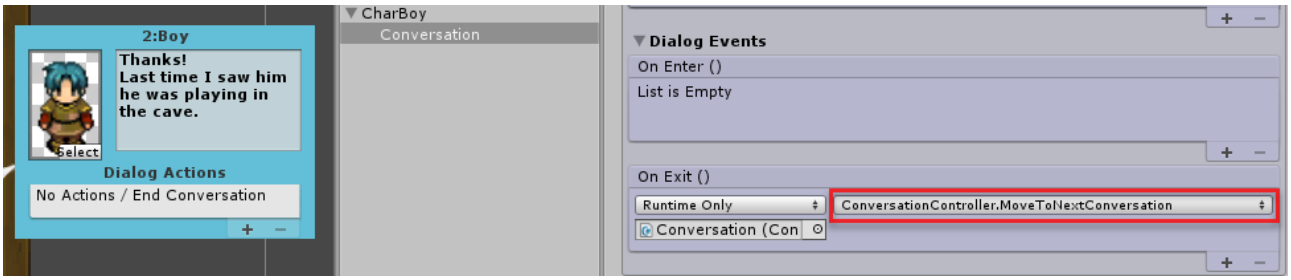


Call the method OverrideStartDialog from the onEnter event of the dialog from which you want to start the next time.

This use case is implemented in "**[RPG Conversation Editor] 1. RPG Character Dialogues**".

9.4 Move to the next conversation in the list when reaching a certain dialog

For example, if you have an NPC that is asking you to accept a quest. Once you accept the quest you could change the conversation to a different conversation where it is asking you if you have completed the quest.



You can move to the next conversation in the list calling the method MoveToNextConversation in the onExit event of the dialog or action where you accept the quest.

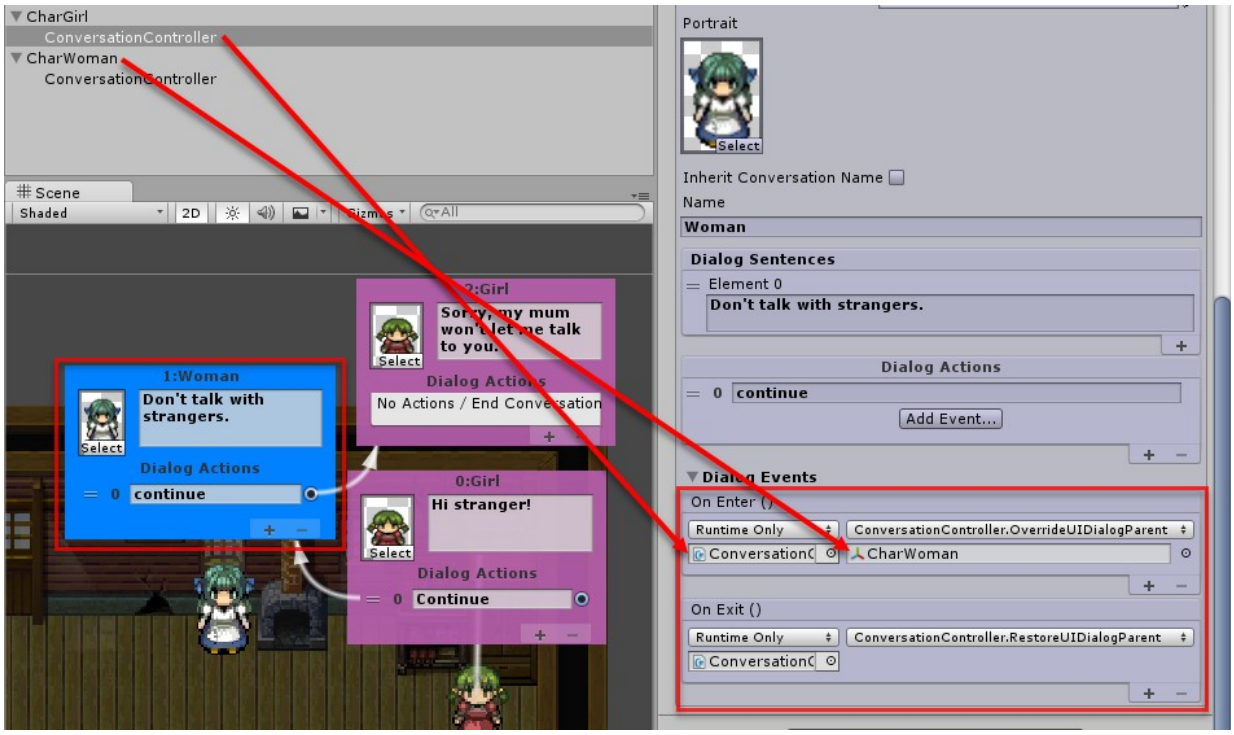
This use case is implemented in "**[RPG Conversation Editor] 1. RPG Character Dialogues**".

9.5 Change the parent of the UIDialog in a multiple character conversation

If you have a conversation with multiple characters and you want to place the UIDialog over different characters during the conversation, you can call the conversation controller method OverrideUIDialogParent to chage its position relative to another gameObject.

You can do this using the Dialog events **onEnter** and **onExit**.

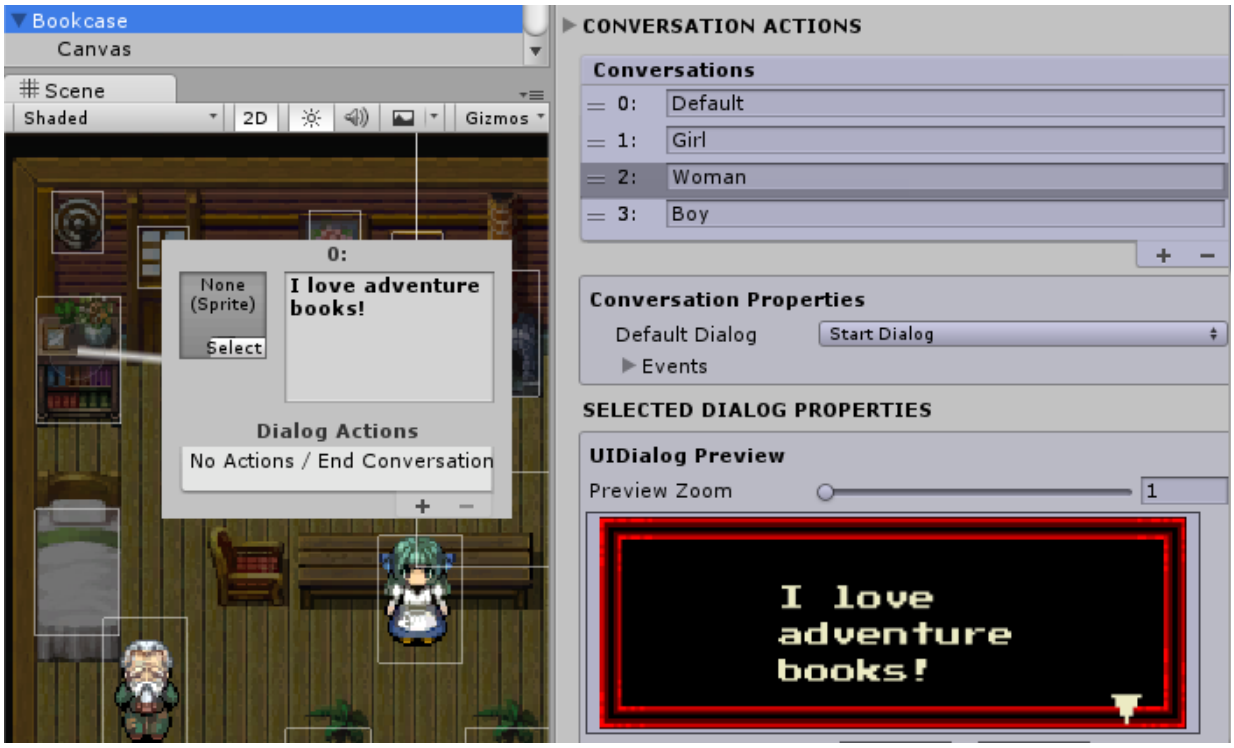
In the onEnter event you will call the conversation controller method OverrideUIDialogParent and to restore it back to the original parent, you can call the method RestoreUIDialogParent.



This use case is implemented in "**[RPG Conversation Editor] 1. RPG Character Dialogues**".

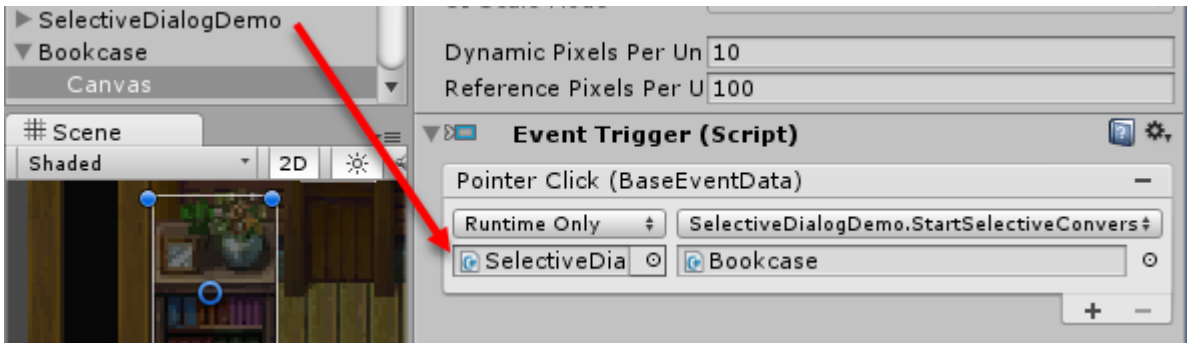
9.6 Display a selective dialog depending on the character played by the player

Sometimes, in your games, you have a party of characters and you can choose witch one to play with, and depending of your choose you want to display a different dialog. To do this, you can create a conversation with the name of the character and an special name for a default conversation like "Default":



As you can see, this bookcase has a conversation for each character and a Default conversation in case the character has no special conversation.

To start the conversation you can use a canvas with an onPointerClick event that calls a script that will start the conversation according to your selected party member name:



You can see an example of this case in the demo "**[RPG Conversation Editor] 2. Selective Dialogues**".

In this example, StartSelectiveConversation method received the conversation controller to be started and call the StartConversation method with the name of the current character that in this demo is the

name of the selected sprite:

```
public class SelectiveDialogDemo : MonoBehaviour
{
    public SpriteRenderer PlayerSpriteRenderer;

    public void StartSelectiveConversation(ConversationController convCtrl)
    {
        string conversationName = PlayerSpriteRenderer.sprite.name;
        if (convCtrl.FindConversationByName(conversationName) == null)
            conversationName = "Default";
        convCtrl.StartConversation(conversationName);
        // if no portrait is set, overwrite the portrait and name with the current character
        if (!convCtrl.uiDialogInstance.portrait.sprite)
        {
            convCtrl.uiDialogInstance.portrait.sprite = PlayerSpriteRenderer.sprite;
            convCtrl.uiDialogInstance.portrait.gameObject.SetActive(true);
            convCtrl.uiDialogInstance.dialogName.text = PlayerSpriteRenderer.sprite.name;
        }
    }
}
```

9.7 *Make conditional dialog actions appear only when certain condition is met*

Imagine that an NPC ask you to do something to complete a quest. You could need to display a dialog action like “Here you have what you asked for” only when you already complete the quest.

You can use the action event **onPreProcess** to change action properties before the action is displayed, like the visibility.

Create a onPreProcess event for the action you want to modify according to other parameters. In the method you are calling using this event, you can access the current processed action like this:

Dialog.DialogAction *action* = ***UIDialog.****processedDialogAction;*

And change the visibility like this:

UIDialog*.processedDialogAction.visible = isQuestCompleted;*

For example:

```
[Header("Conditional Dialog Actions")]
public bool JohannaQuestAchieved = false;
public bool AskJohannaToJoinTheParty = false;

/// <summary>
/// Called from Johanna's conversation dialog action event onPreProcessed will hide or unhide the action according to the state of the mission.
/// </summary>
public void DialogActionCheck_JohannaQuestAchieved()
{
    UIDialog.processedDialogAction.visible = JohannaQuestAchieved;
}
```

You can see an example with conditional dialog actions in the demo "**[RPG Conversation Editor] 3. Conditional Dialogue Actions**".

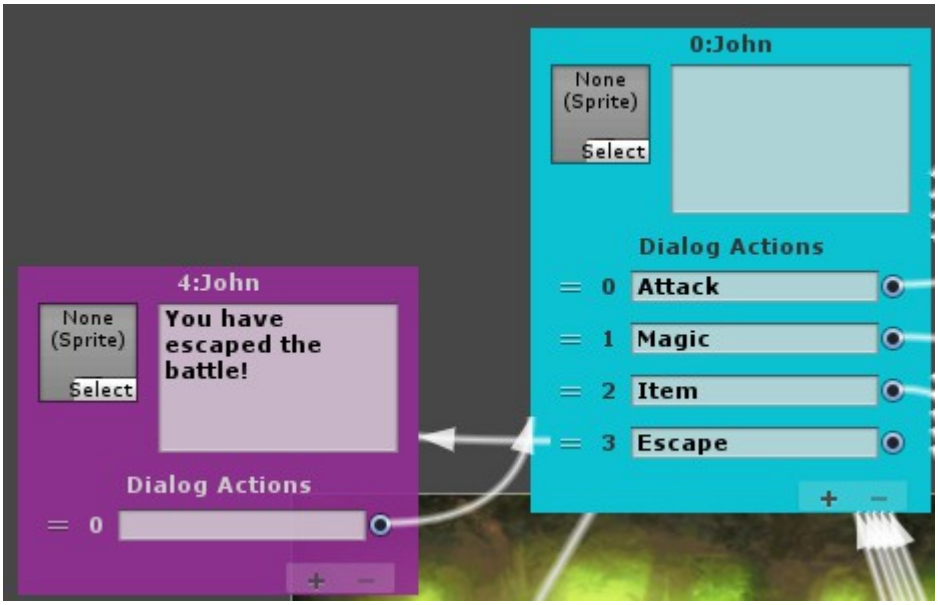
9.8 Use special tags in the dialog text like player name, money, etc

Like in the previous conditional dialog actions use case, you can use the event **onPreProcess** to change the action text (called name) to rename a tag to another value.

```
public void ProcessActionTags()
{
    Dialog.DialogAction action = UIDialog.processedDialogAction;
    action.name = action.name.Replace("<player_name>", "Michael");
}
```

9.9 Specify the next dialog in a conversation without any dialog action

An action of a dialog box can lead to another dialog box when the action is selected. But if you don't want to display an action, like "continue", you can create an action leaving the name empty. This way, the action won't be displayed and when you click over the text or press the submit button, the next dialog will be displayed.



You can see an example of this case in the demo "**[RPG Conversation Editor] 4. Menu Dialogue**".