TOOL ASSETS

Dialogue Tool Plugin Documentation

For Unity

Ryan Zagala

7/27/2020

Table of Contents

Summary:

The Dialogue Tool Plugin for Unity provides developers a way to incorporate dialogue into their games, simulations, etc. This tool has two separate canvases: one for normal On-Screen interface (Screen Space - Overlay) and one for VR (Screen Space - Camera); a canvas that will be a world space render will come soon. The resolution settings that each canvas has is 1920x1080 (16:9) setting.

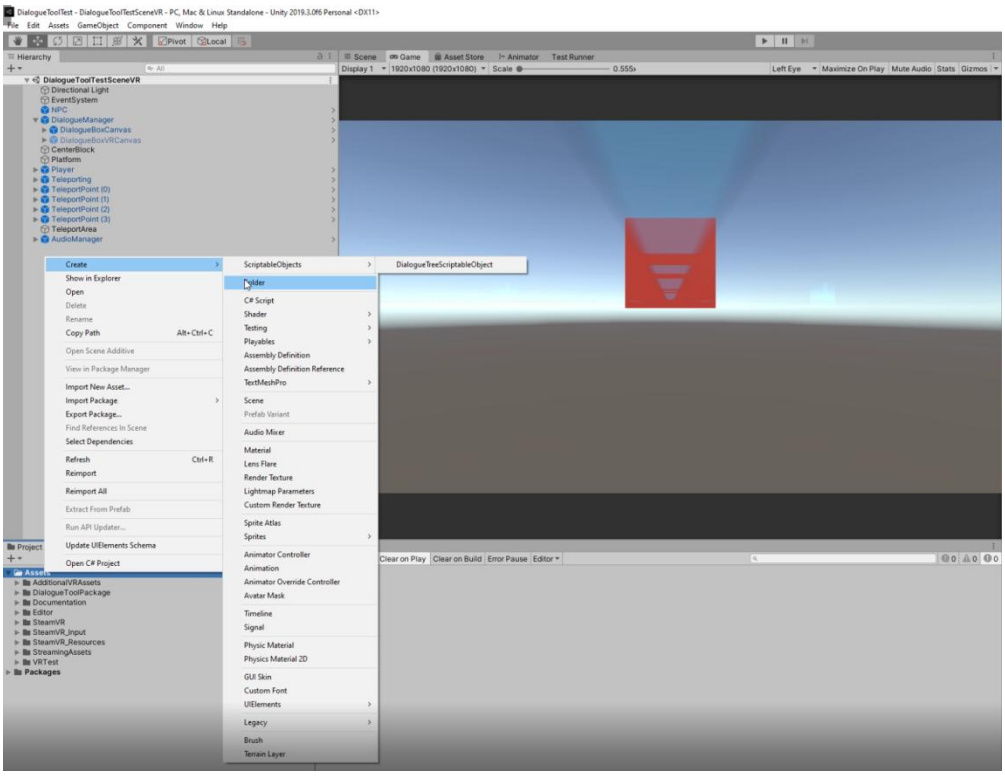
Package Contents:

1. Animations
    a. ContinueDialogueImages
        i. AutoContinueDialogueImage
            1. AutoContinueDialogueImage.controller
            2. ScrollingRight.anim
        ii. InputContinueDialogueImage
            1. Bounce.anim
            2. FadeFlashing.anim
            3. InputContinueDialogueImage.controller
    b. DialogueBoxCanvas
        i. Close.anim
        ii. DialogueBoxCanvas.controller
        iii. Open.anim
        iv. Standby.anim
2. Audio
    a. DialogueSpeech
        i. Introduction
            1. Hello there My name is Ryan.mp3
            2. I am here to tell you something.mp3
            3. Something cool.mp3
3. DialogueTreeAssets
    a. Introduction.asset
    b. Introduction_Response_A.asset
    c. Introduction_Response_B.asset
4. Images
    a. AutoContinueDialogueTemp.jpg
    b. InputContinueDialogueImage.jpg
5. Materials
    a. UI_Materials
        i. AutoContinueDialogueRenderTexture.renderTexture
        ii. AutoContinueDialogueMaterial.mat
    b. VRLevel_Materials
        i. CenterBlock.mat
        ii. Ground.mat

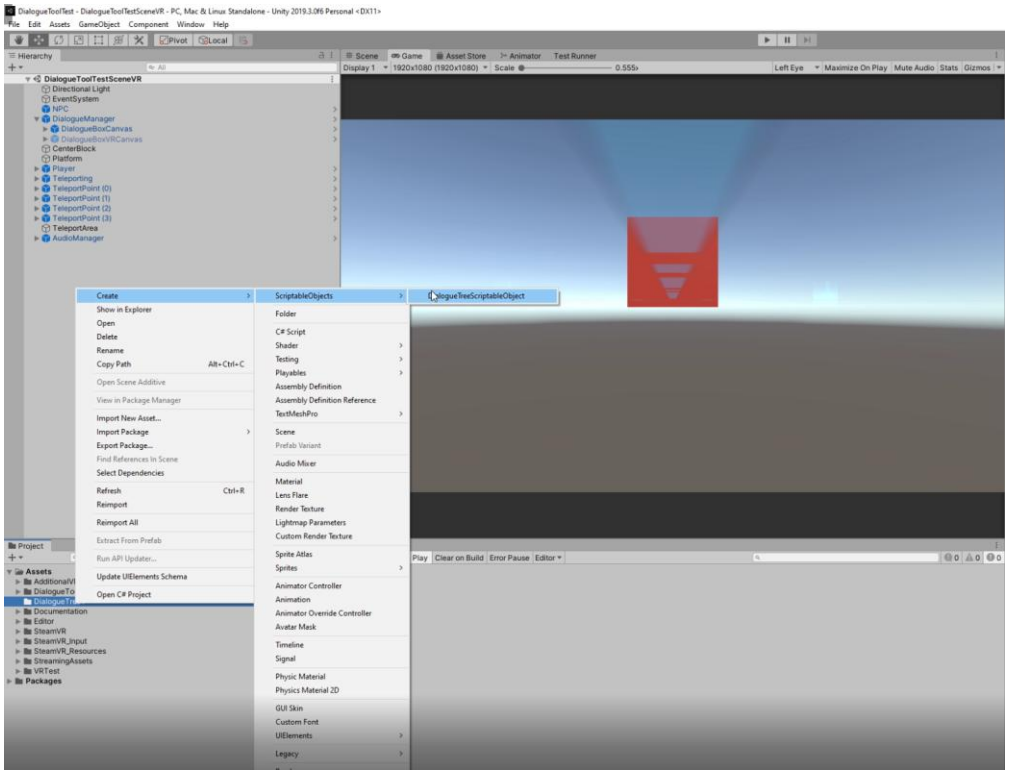
6. Prefabs
    a. AnswerButton.prefab
    b. DialogueBoxCanvas.prefab
    c. DialogueBoxVRCanvas.prefab
    d. DialogueManager.prefab
    e. NPC.prefab
7. Scenes
    a. DialogueToolTestScene.unity
    b. DialogueToolTestSceneVR.unity
8. Scripts
    a. DialogueSystem
        i. DialogueManager.cs
        ii. DialogueSystemAssembly.asmdef
        iii. DialogueTree.cs
        iv. DialogueTreeShim.cs
        v. DialogueTrigger.cs
        vi. MultipleChoiceAnswer.cs
        vii. MultipleChoiceTemplate.cs

Setup and Use:

1. **Gettings Started**
    a. Create a folder where you want to store your **DialogueTree scriptable objects** by right-clicking on the mouse in the project window. Click on **Create >> Folder**. In the project example or package, the folder used to store these scriptable objects is named something like “DialogueTrees” or “DialogueTreeAssets”; however, it is **recommended** that create your folder outside of the **“DialogueToolPackage”** folder.

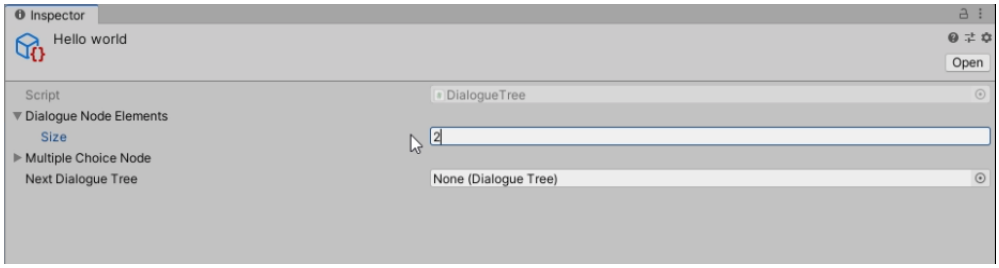


b. Created a **DialogueTree** scriptable object by right-clicking on the mouse in the project window. Click on **Create >> ScriptableObjects >> DialogueTreeScriptableObjects**. Name the **DialogueTree** scriptable object.

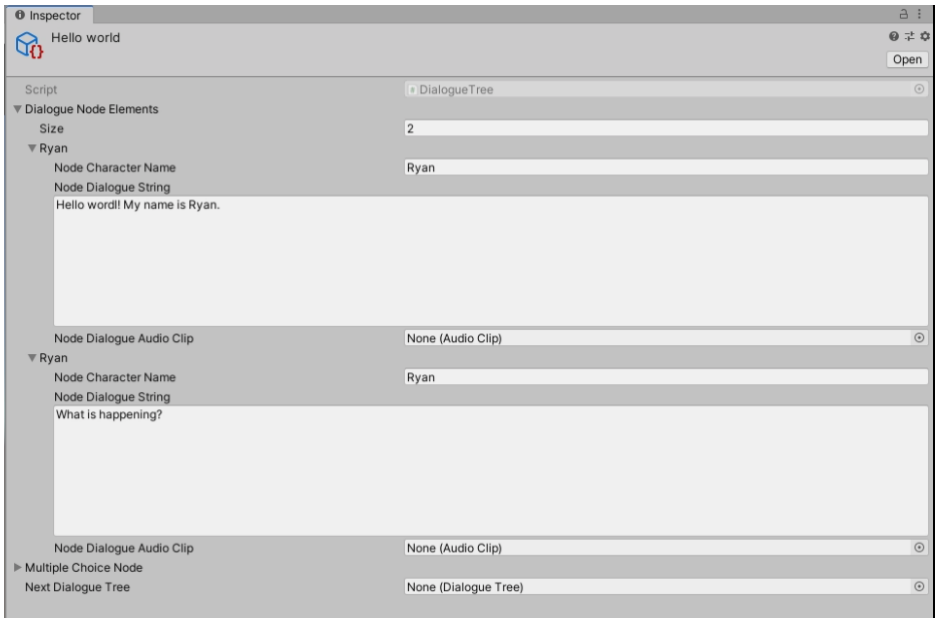


c. Click on the created scriptable object and check the inspector window. There are 3 things this scriptable object can contain: a list of Dialogue Nodes (**Dialogue Node Elements**), a special node used for multiple choice (**Multiple Choice Node**), and the next dialogue tree to play once this current DialogueTree is done playing (**Next Dialogue Tree**).

d. Click on the Dialogue Node Elements list and choose your desire size (**how many nodes you wish for this DialogueTree to contain**).

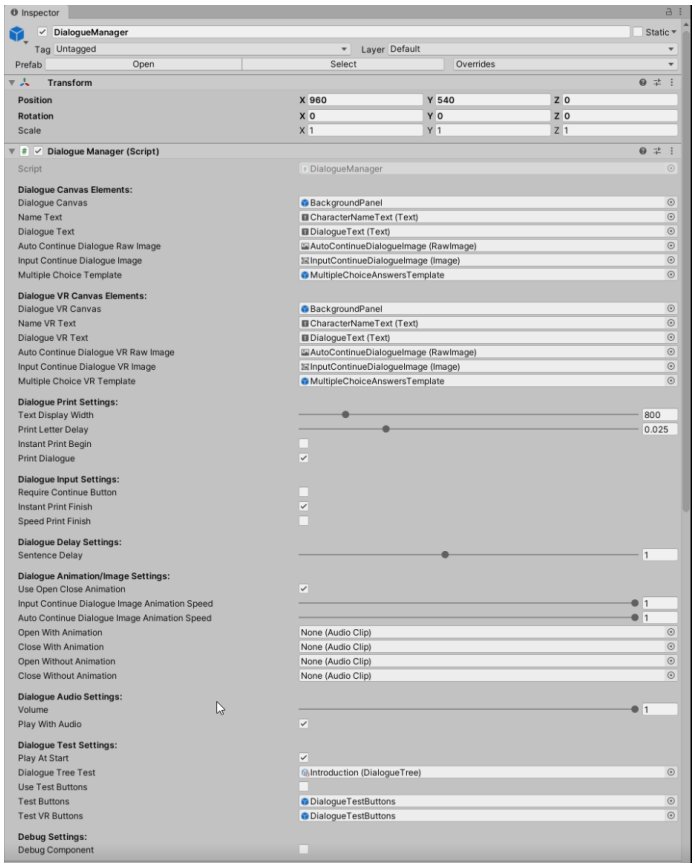


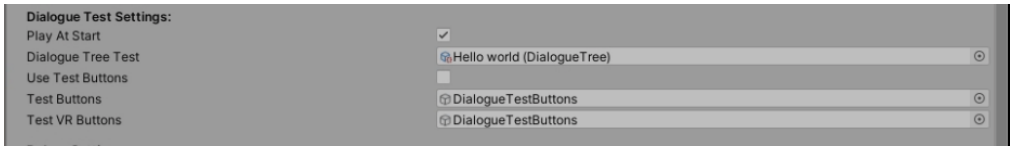
e. Each Dialogue Node contains 3 fields: the string for the character who is currently talking (**Node Character Name**), the string for the dialogue for that character (**Node Dialogue String**), and the audio clip for the dialogue (**Node Dialogue Audio Clip**). Fill in the fields with the desired strings and audio clip. **Note: The Dialogue Audio Clip is optional; you can choose to leave null.**



f. Now it's time to play the dialogue. There are two ways of starting a DialogueTree: one for quick testing and the other is to call the **DialogueManager.instance.StartDialogue (DialogueTree dialogueTree)** method from another class that derives from the Monobehavior class and runs in the scene. Refer to step g and h for these methods.

g. For quick testing purposes, drag the desired DialogueTree scriptable object into the Dialogue Tree Test field of the **DialogueManager** in the inspector and enable Play At Start Boolean. Press play and it should start shortly.



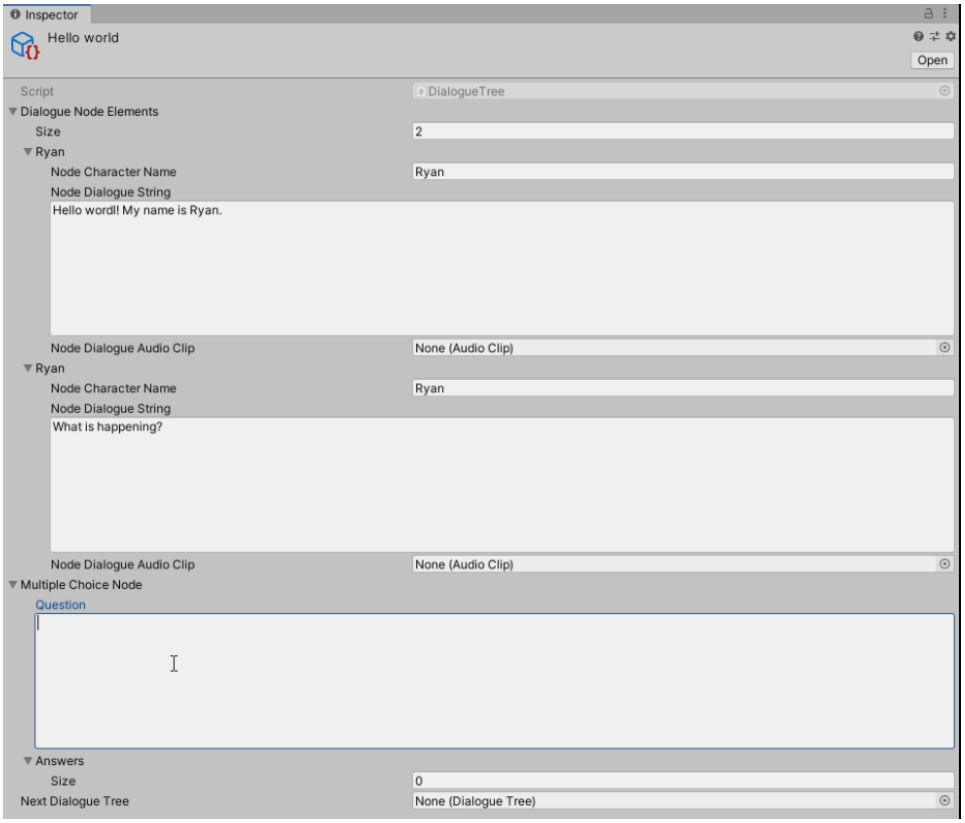


h. For the **DialogueManager.instance.StartDialogue(DialogueTree dialogueTree)** method, make sure that the Play At Start Boolean is false in **DialogueManager** in the inspector so you don't accidentally start it. Create a script that will call the method when desired by the behavior. For a quick example, call it from the Start() function that the script automatically has when created. **Note: Outside C# scripts that utilize the classes in the package contents will need to enter the namespace "using DialogueSystem".**

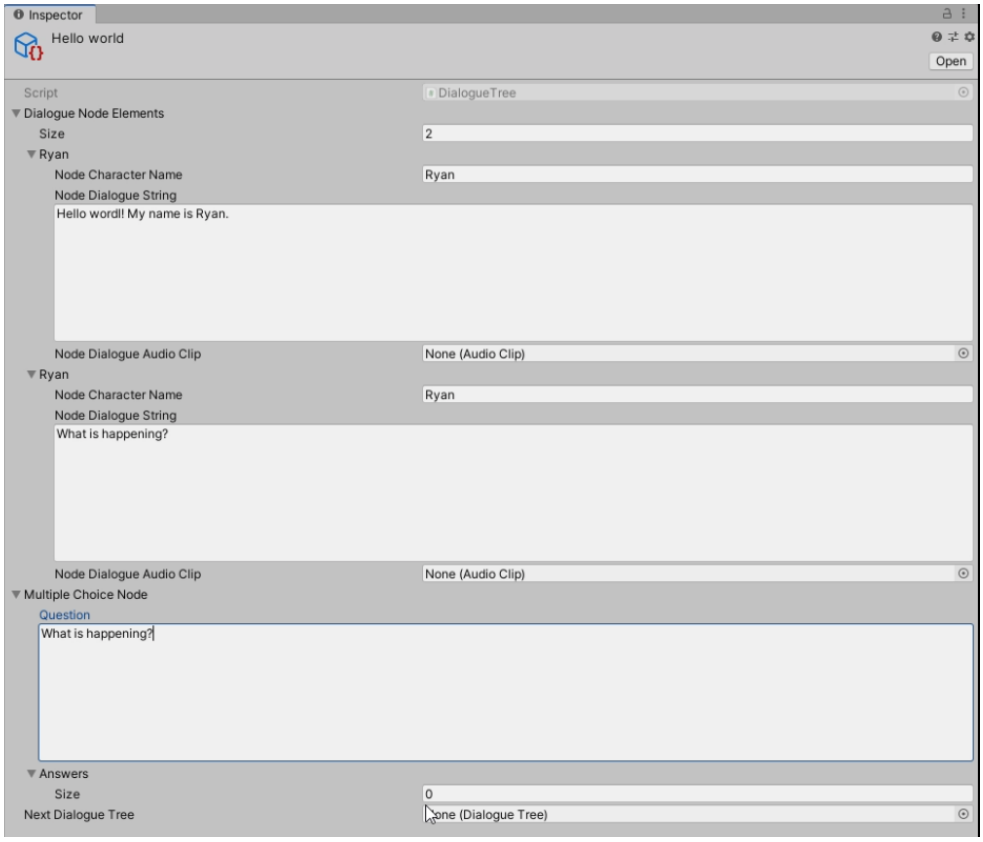
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using DialogueSystem;

0 references
public class DialogueStartExample : MonoBehaviour
{
    public DialogueTree dialogueTreeExample;

    // Start is called before the first frame update
    0 references
    void Start()
    {
        DialogueManager.instance.StartDialogue(dialogueTreeExample);
    }

    // Update is called once per frame
    0 references
    void Update()
    {

    }
}
```

2. Multiple Choice and Branching

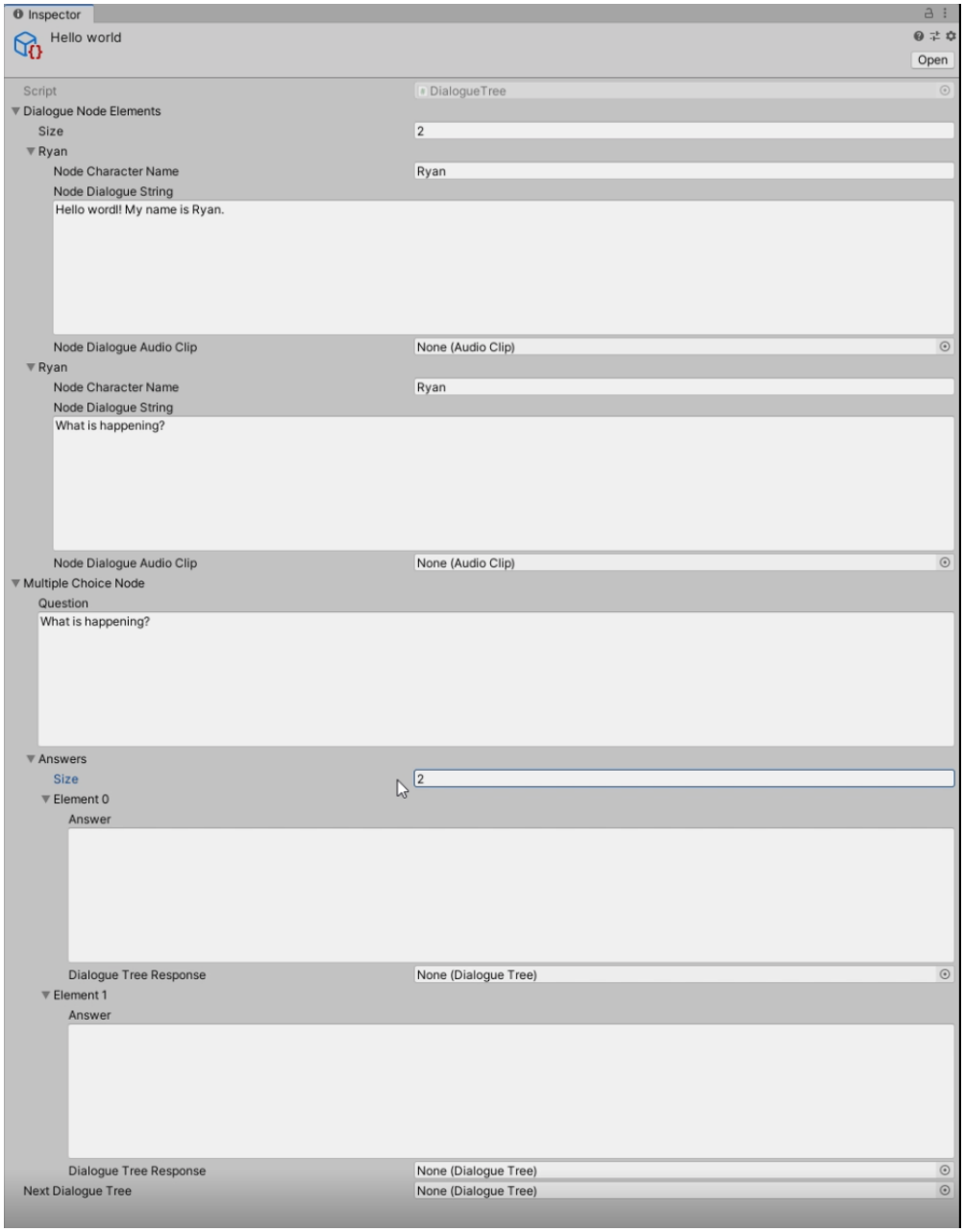
a. Click on the **Multiple Choice Node** of the scriptable object to find 2 fields: a string field to place the question or the sentence from the last dialogue node in the Dialogue Node Elements list (**Question**) and a list of answers to choose from (**Answers**).



b. Fill in the question field with a **question or sentence** that is in **context** with the associated dialogue nodes elements. **Note: It is best for flow to place the question in the last dialogue node in the Dialogue Node Elements list.**

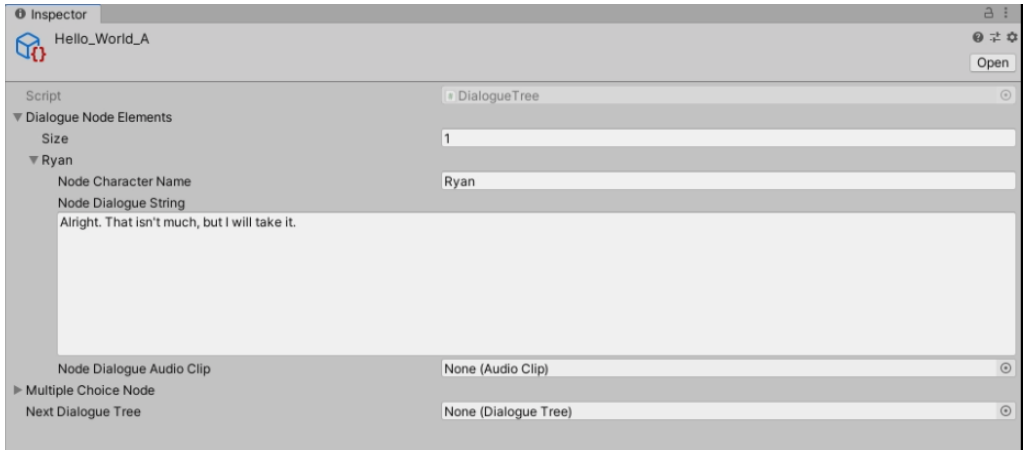


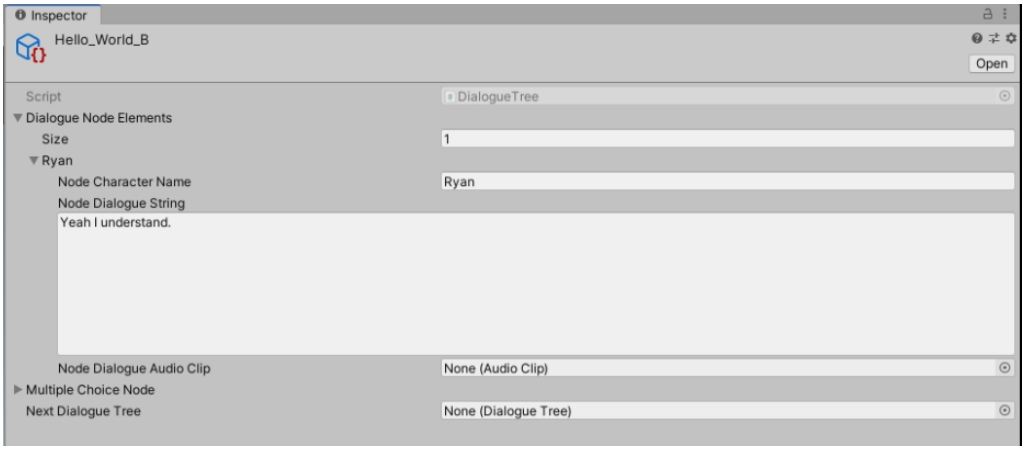
c. Click on the Answers list and **set the size to at least 2**. From there, there are 2 fields for each answer: the string field for the answer (**Answer**) and a DialogueTree field to place the next DialogueTree you want to play (**Dialogue Tree Response**). **Note: There must be at least 2 answers to select from; otherwise, the multiple choice node is treated as if it doesn't exist.**

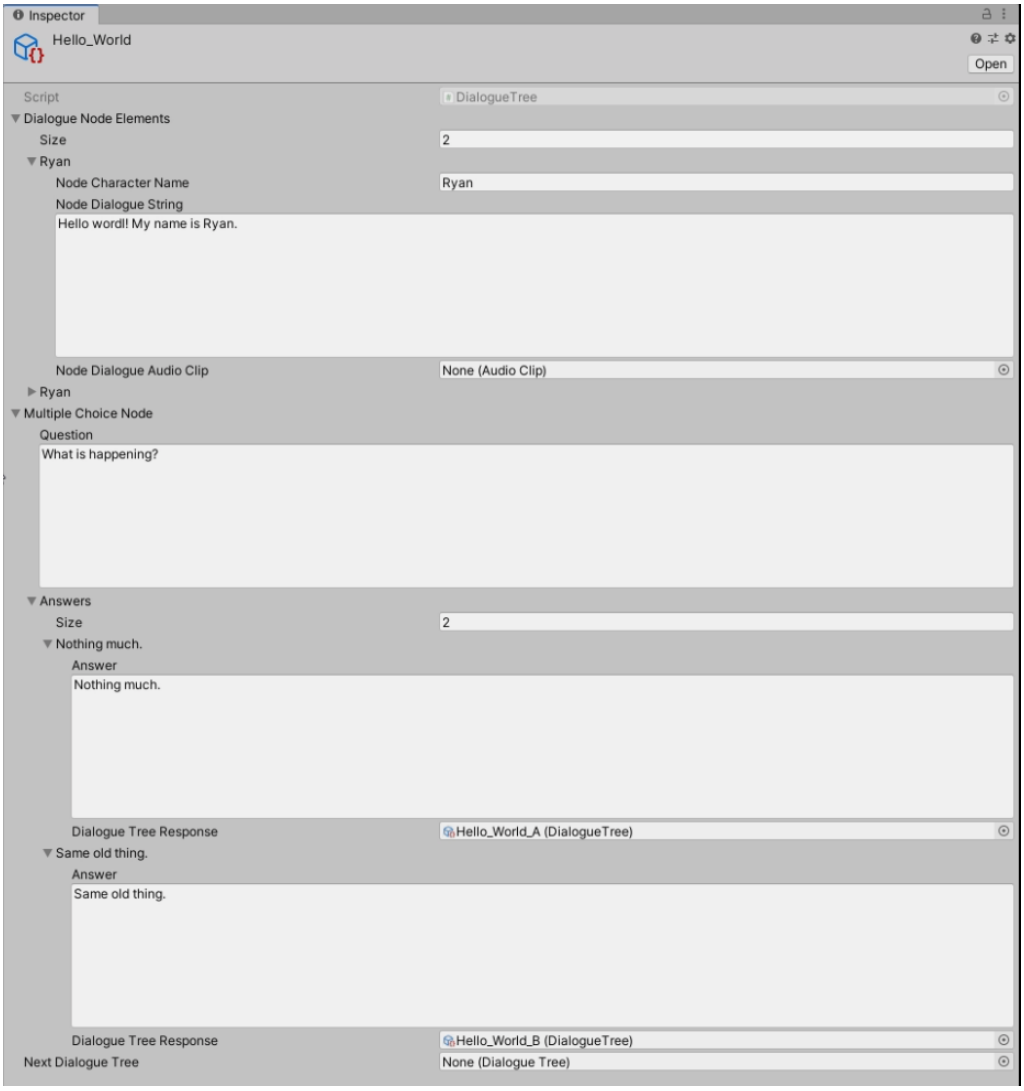


d. Set an answer or response in the **Answer** string field.

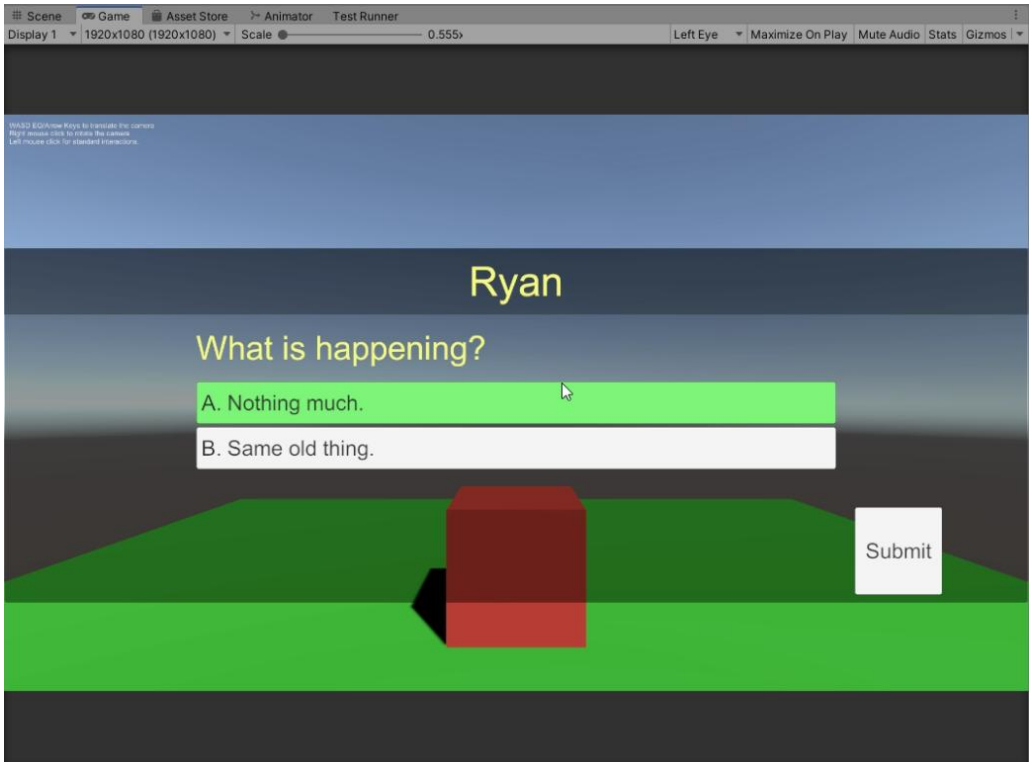
e. Create **two more DialogueTree scriptable objects** and set each in the Dialogue Tree fields: one for **answer A** and the other for **answer B**. **Note: If there is no DialogueTree set for an answer, then the DialogueTree will end and the canvas will close.**


Inspector
Hello_World_A
Open
Script
DialogueTree
Dialogue Node Elements
Size
1
Ryan
Node Character Name
Ryan
Node Dialogue String
Alright. That isn't much, but I will take it.
Node Dialogue Audio Clip
None (Audio Clip)
Multiple Choice Node
Next Dialogue Tree
None (Dialogue Tree)

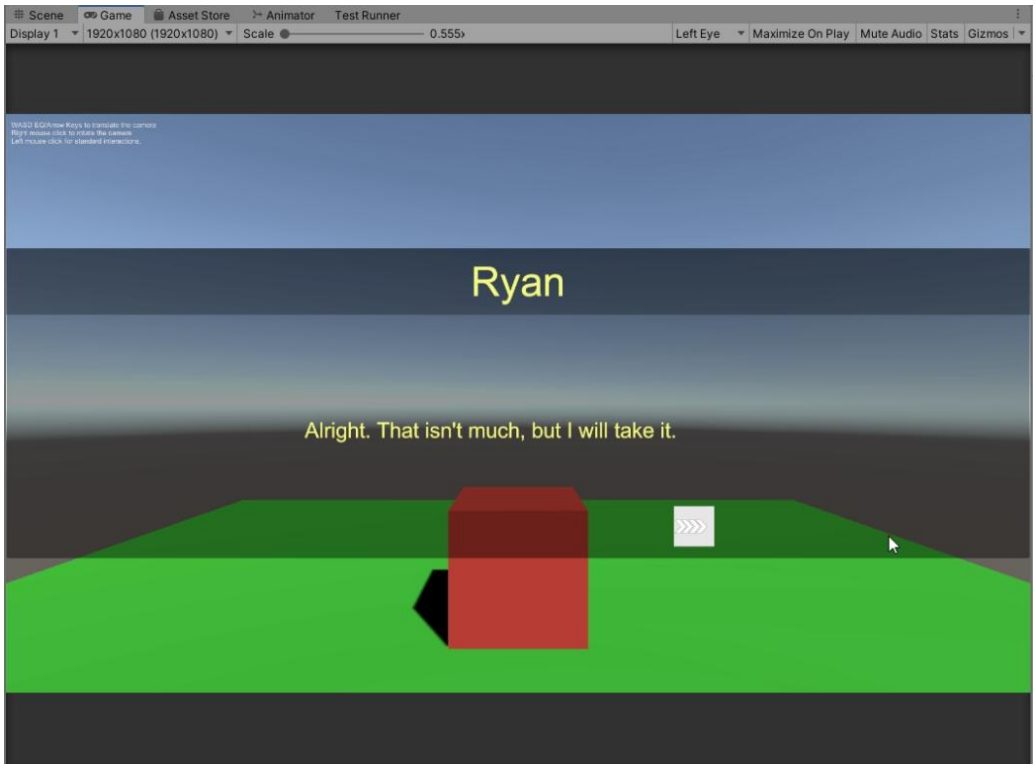

Inspector
Hello_World_B
Open
Script
DialogueTree
Dialogue Node Elements
Size
1
Ryan
Node Character Name
Ryan
Node Dialogue String
Yeah I understand.
Node Dialogue Audio Clip
None (Audio Clip)
Multiple Choice Node
Next Dialogue Tree
None (Dialogue Tree)



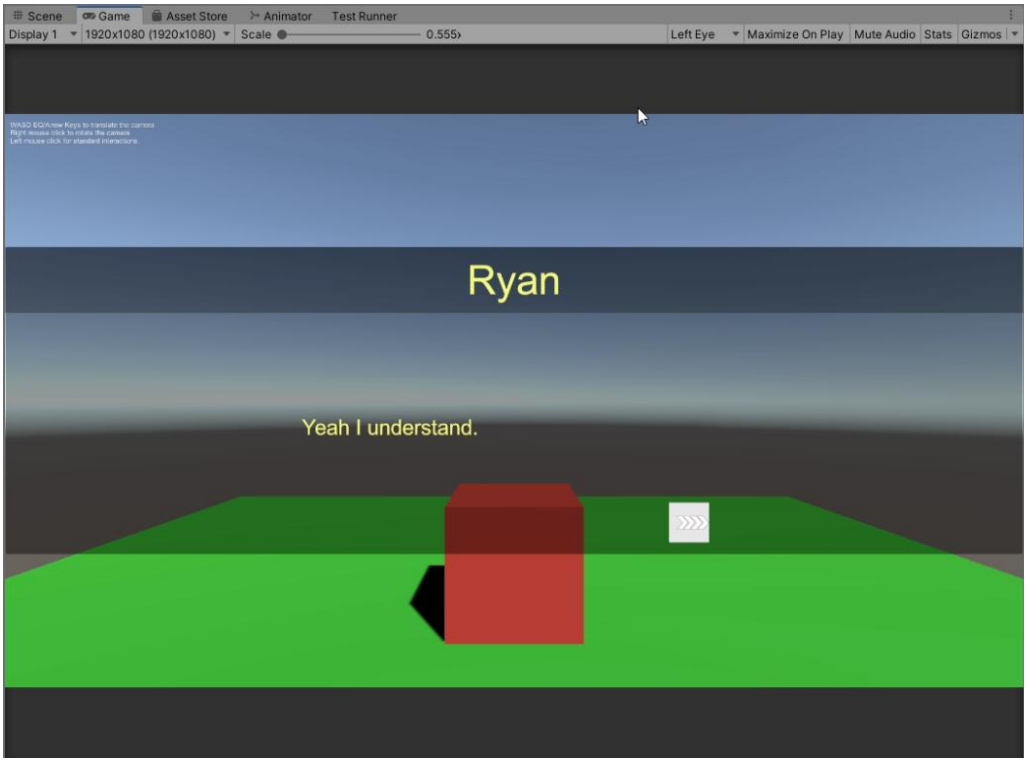
f. Test it out using one of the two methods mentioned before. Two different dialogues should play if you select different answers.
Multiple Choice Section:



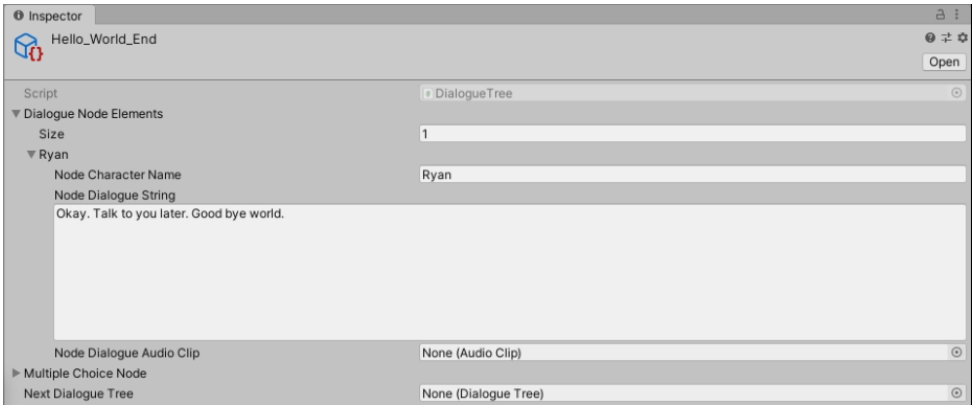
Answer A was Chosen:



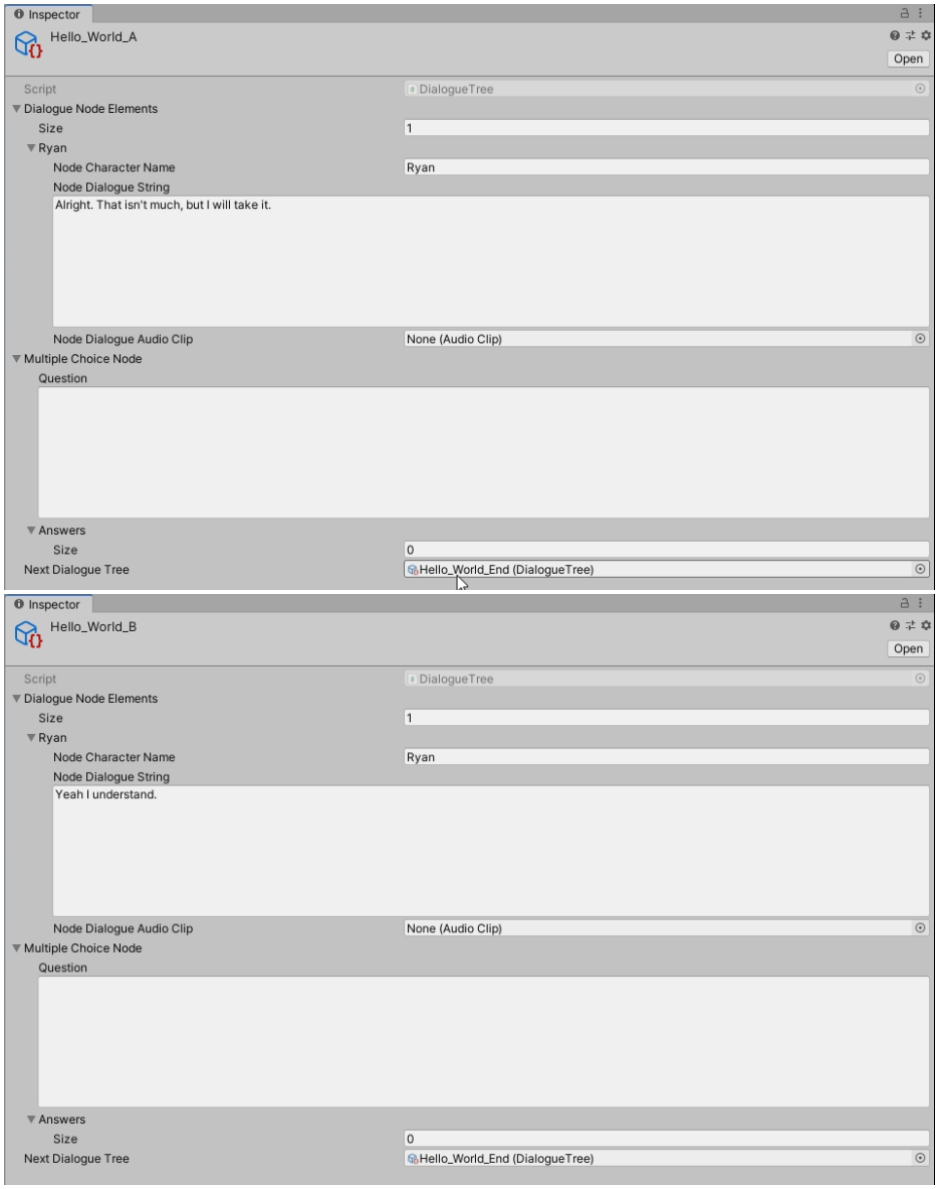
Answer B was Chosen:



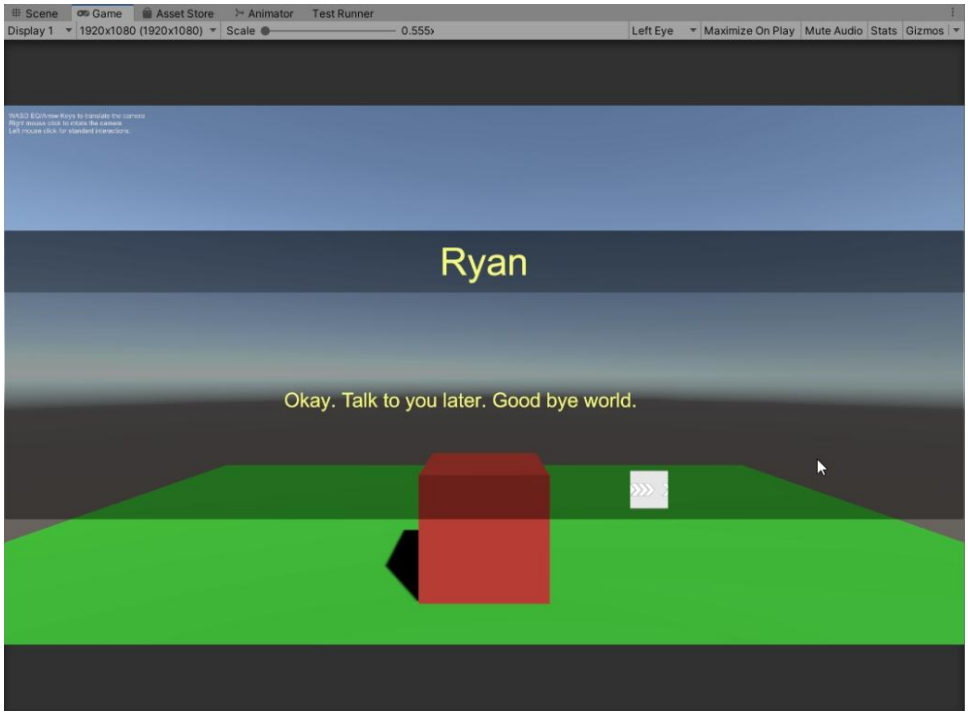
3. **Merging Dialogue Tree Branches**
    a. After following the multiple choice and branching steps, these branches can **converge into one DialogueTree** simply by setting the **next DialogueTree** to each of the branch DialogueTrees in the **Next Dialogue Tree** field. Create another DialogueTree scriptable object that marks the **end**.



b. Set this DialogueTree in the **NextDialogueTree** fields for **Answer A and B**.

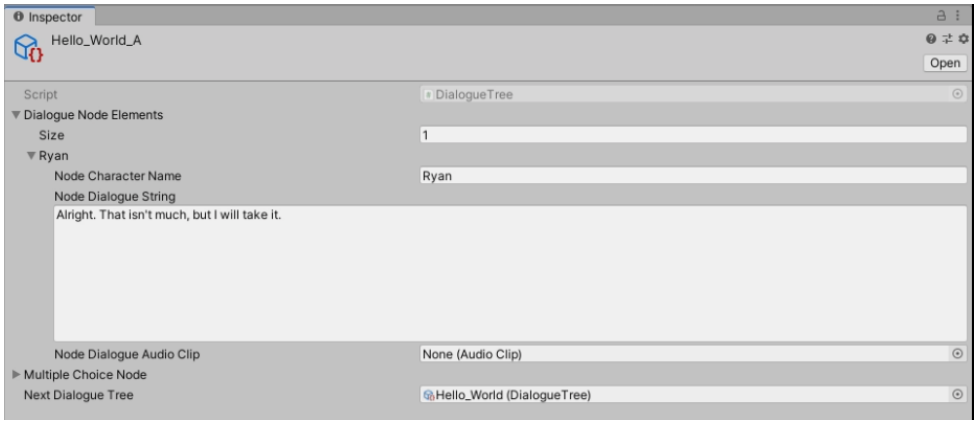


c. Test it out using one of the two methods mentioned before. The two different dialogues branches should lead to play the **same DialogueTree regardless which answer you choose in the Multiple Choice part**.
The Last DialogueTree playing:

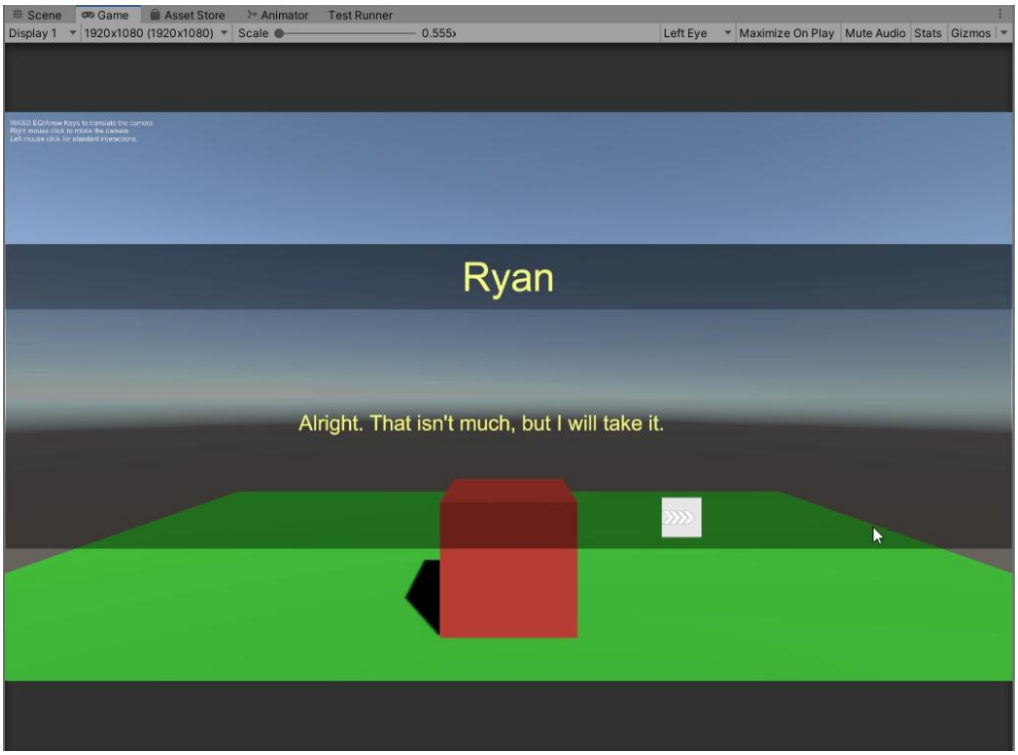


4. Loops

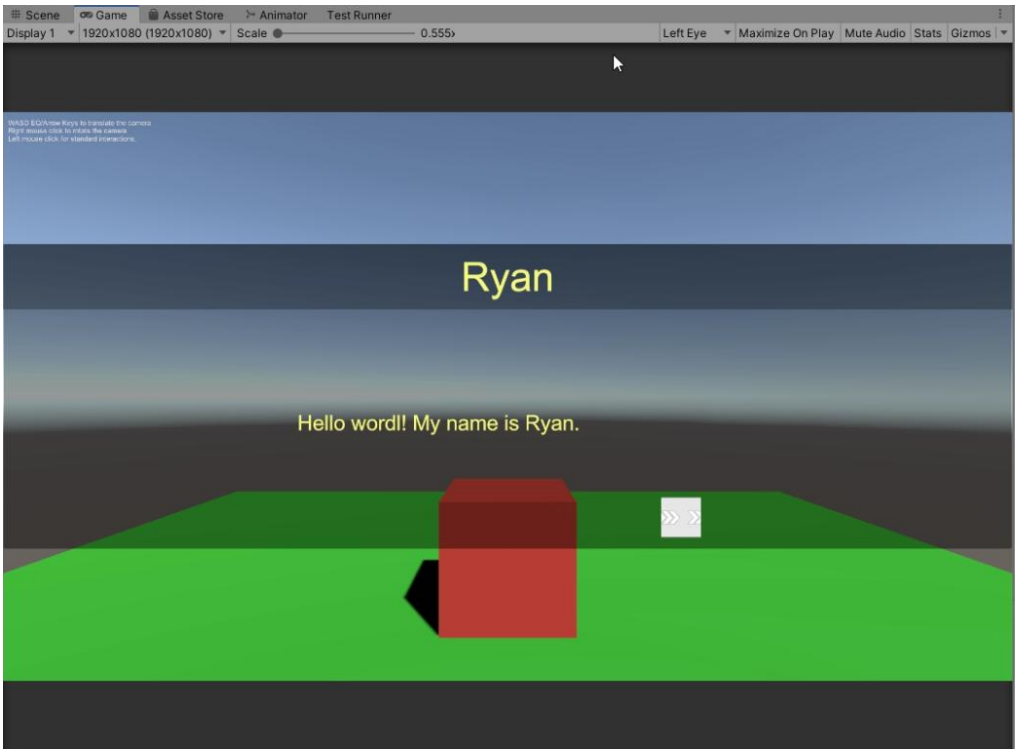
a. Simply attach a **previously used DialogueTree** to the **Next Dialogue Tree** field in **any DialogueTree**. As an example, use the **first DialogueTree** and place it in the **NextDialogueTree** field of **answer A**.



b. Test it out using one of the two methods mentioned before. If **answer A is chosen** in the Multiple Choice part, then it should **play the Dialogue Tree that played previously**. After Answer A is chosen:



The previously played DialogueTree plays again:



Animations:

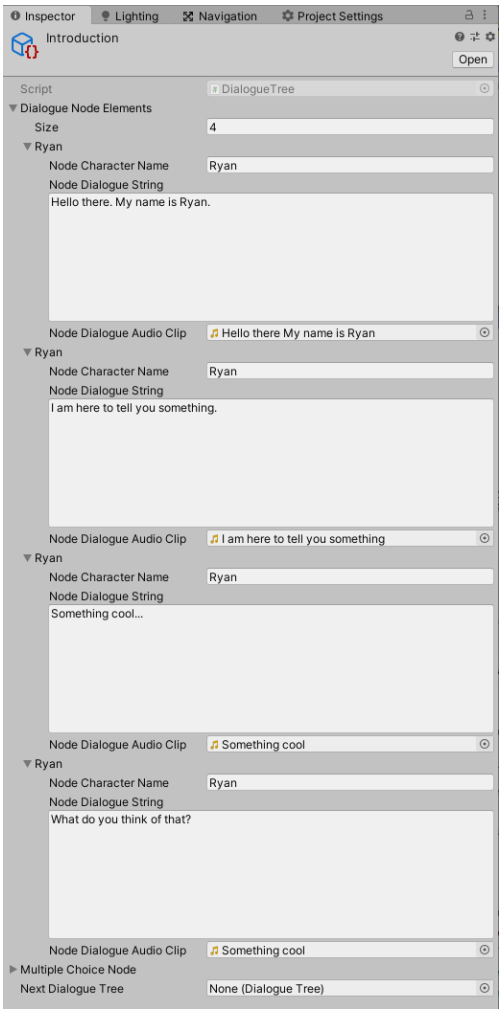
1. Continue Dialogue Images
    a. Auto Continue Dialogue Image
        i. **AutoContinueDialogueImage.controller**
            1. The animator controller for the AutoContinueDialogueImage.
        ii. **ScrollingRight.anim**
            1. The animation for the AutoContinueDialogueImage which scrolls its GUI horizontally.
    b. Input Continue Dialogue Image
        i. **Bounce.anim**
            1. One of the animations for the InputContinueDialogueImage which bounces the GUI up and down.
        ii. **FadeFlashing.anim**
            1. One of the animations for the InputContinueDialogueImage which fades the GUI from solid to transparent and vice versa.
        iii. **InputContinueDialogueImage.controller**
            1. The animator controller for the InputContinueDialogueImage.
2. Dialogue Box Canvas
    a. **Close.anim**
        i. The animation for the DialogueBoxCanvas which will shrink its y-value scale size to 0.
    b. **DialogueBoxCanvas.prefab**
        i. The prefab of the DialogueBoxCanvas.
    c. **Open.anim**
        i. The animation for the DialogueBoxCanvas which will expand its y-value scale size to 1.
    d. **Standby.anim**
        i. The animation for the DialogueBoxCanvas that will keep its y-value scale size constant.

Audio:

1. Dialogue Speech
    a. Introduction
        i. **Hello there My name is Ryan.mp3**
            1. Audio clip of a dialogue line “Hello there. My name is Ryan.”
        ii. **I am here to tell you something.mp3**
            1. Audio clip of a dialogue line “I am here to tell you something.”
        iii. **Something cool.mp3**
            1. Audio clip of a dialogue line “Something cool.”

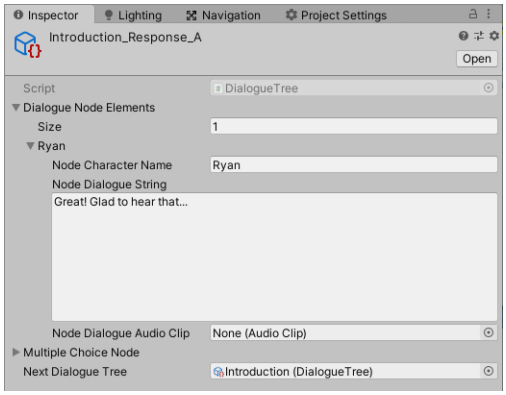
Dialogue Tree Assets:

1. **Introduction.asset**



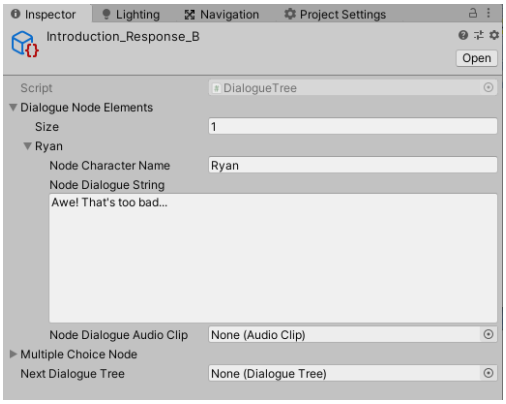
   a. The DialogueTree scriptable object of the introduction.

2. **Introduction_Response_A.asset**



   a. The DialogueTree scriptable object of the response to the first choice of the question in Introduction.

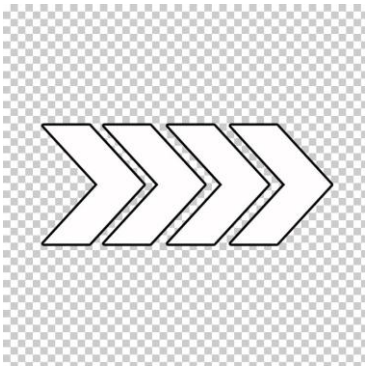
3. **Introduction_Response_B.asset**



a. The DialogueTree scriptable object of the response to the second choice of the question in Introduction.

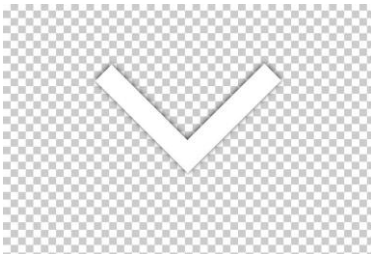
Images:

1. **AutoContinueDialogueImageTemp.png**



a. The image that is used in the DialogueToolCanvas when the requiresContinueButton option is disabled in the inspector of the DialogueManager.cs.

2. **InputContinueDialogueImage.png**



a. The image that is used in the DialogueToolCanvas when the requiresContinueButton option is enabled in the inspector of the DialogueManager.cs.

Materials:

1. UI_Materials
    a. **AutoContinueDialogueRenderTexture.material**
        i. A material used to render the AutoContinueDialogueMaterial.

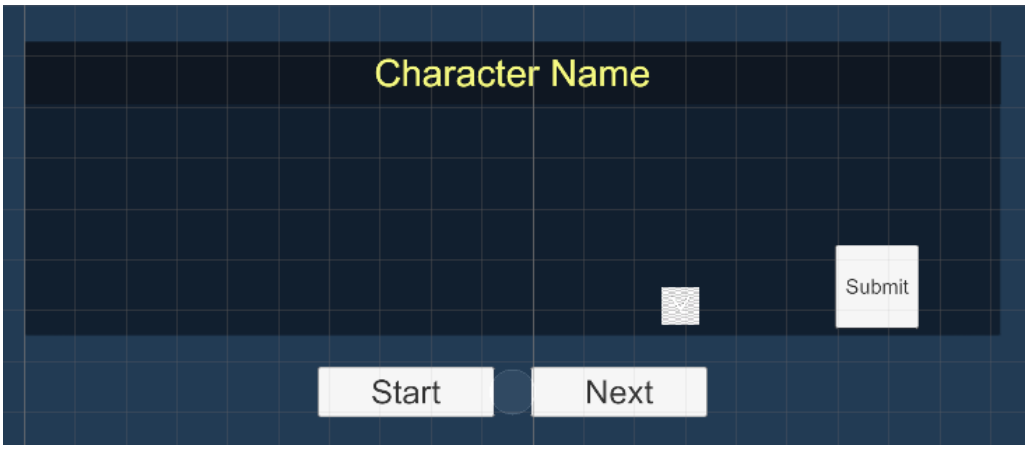
b. **AutoContinueDialogueMaterial.material**
    i. A material used to render the AutoContinueDialogueImage.

2. VRLevel_Materials
    a. **CenterBlock.material**
        i. A basic material used for the center block in the test scenes.
    b. **Ground.material**
        i. A basic material used for the ground in the test scenes.

Prefabs:

1. DialogueSystem
    a. **AnswerButton.prefab**

        i. The button that contains all the answer data when instantiated.
    b. **DialogueBoxCanvas.prefab**



        i. The main standard gameobject that is the parent for the dialogue display.
    c. **DialogueBoxVRCanvas.prefab**
        i. A VR version of the DialogueBoxCanvas gameobject that is rendered in Camera Space in front of the VR headset. The answer choices can be interacted with the used of the index finger of the hand. (Must be setup with Additional VR Assets' TouchDetectionXRAddon.cs or TouchDetectionSteamVRAddon.cs).
    d. **DialogueManager.prefab**
        i. The prefab is the core of the plug-in that contains the different types of canvases for standard and VR gameplay. Each canvas is structured the same way in the Hierarchy window. Some settings might differ in each canvas.
    e. **NPC.prefab**
        i. The gameobject that has the DialogueTrigger.cs component attached to it. Each NPC can carry its own unique DialogueTree scriptable object.

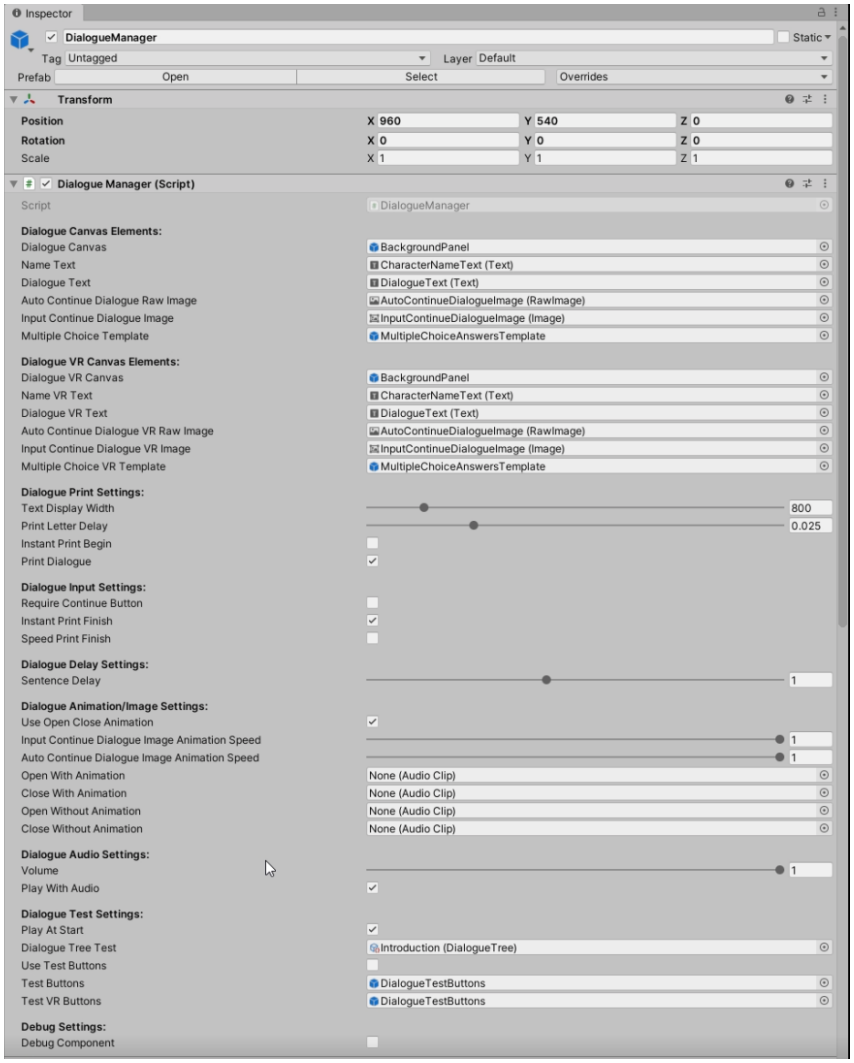
Scenes:

1. **DialogueToolTestScene.unity**
    a. The test scene to test the DialogueToolCanvas on Start(). The Render Mode for the canvas is in ScreenSpace – Overlay.
2. **DialogueToolTestSceneStreamVR.unity**
    a. The test scene to test the DialogueToolVRCanvas on Start(). The Render Mode for the canvas is in ScreenSpace – Camera. Requires SteamVR to be installed and Additional VR Assets' TouchDetectionSteamVRAddon.cs.
3. **DialogueToolTestSceneUnityXR.unity**
    a. The test scene to test the DialogueToolVRCanvas on Start(). The Render Mode for the canvas is in ScreenSpace – Camera. Requires UnityXR to be installed and Additional VR Assets' TouchDetectionXRAddon.cs.

Scripts:

DialogueManager

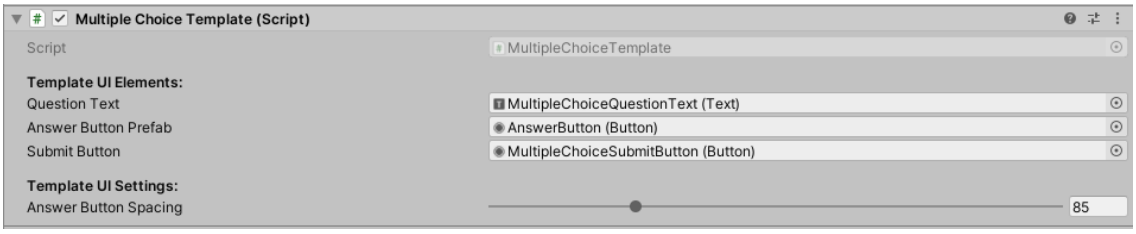
- **DialogueManager Prefab/GameObject**
    - The prefab is the core of the plug-in that contains the different types of canvases for standard and VR gameplay. Each canvas is structured the same way in the Hierarchy window. Some settings might differ in each canvas.
- **DialogueManager.cs Component**



- **Dialogue Canvas Elements**
  - GameObjects in the DialogueBoxCanvas prefab that are controlled and updated by the DialogueManager.
- **Dialogue VR Canvas Elements**
  - GameObjects in the DialogueBoxVRCanvas prefab that are controlled and updated by the DialogueManager.
- **Dialogue Print Settings**
  - Adjusts how the dialogue manager handles printing characters on the DialogueText part of the canvases.
- **Dialogue Input Settings (needs to be programmed for all forms of input.)**
  - Determines how input controls the dialogue being printed.
- **Dialogue Delay Settings**
  - Determines the interval length in between sentences.
- **Dialogue Animation/Image Settings**
  - Determines if animations are used for opening and closing the canvases.
  - Controls animations for input or automatic progression indications.

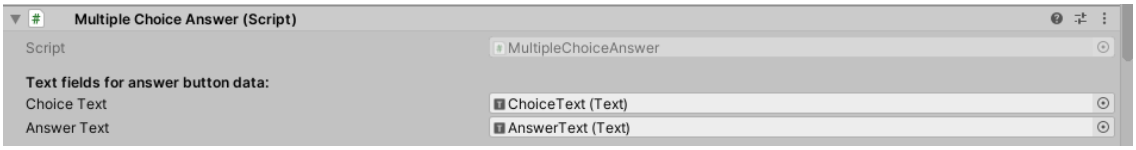
        - Allows for optional audio clips for starting and ending dialogue trees.
    - **Dialogue Audio Settings**
        - Controls volume of the audio sources that play the audio clips in each DialogueTree node.
        - Controls if audio should be played for each DialogueTree node or not.
    - **Dialogue Test Settings**
        - Can only be used if PlayAtStart Boolean is used.
        - Allows for fast testing that plays a DialogueTree on a Monobehavior Start() method.
    - **Debug Settings**
        - If debug component is true, then you can see Debug.Logs associated with the behavior of the DialogueManager.

MultipleChoiceTemplate



- The core of the multiple choice parts of a dialogue. The template is set up using the **SetTemplate()** method. The DialogueManager already takes care of that.
- The **SetTemplate()** method is used when there are two or more answers set in the DialogueTree that is currently playing.

MultipleChoiceAnswer



- This gameobject is instantiated when the **SetTemplate()** method is called from **MultipleChoiceTemplate.cs** gameobject.
- Answer Data can be extracted from the **MultipleChoiceAnswer.cs** gameobject when button is selected and set in the MultipleChoiceTemplate using the **SetChoice()** method.

DialogueTree

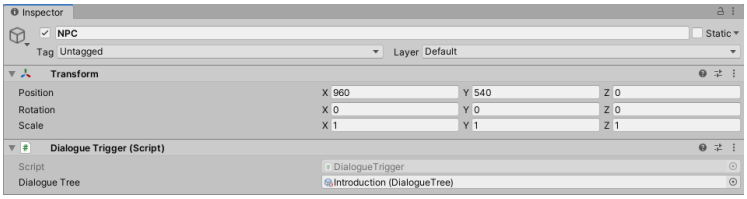
- The data container for dialogue. It is a scriptable object that can be created by right-clicking on the mouse in the project window, and then clicking on **Create >> ScriptableObjects >> DialogueTreeScriptableObjects**.
- It contains a list of dialogue nodes (**Dialogue Node Elements**), a multiple choice node (**Multiple Choice Node**), and a reference to the next dialogue tree (**Next Dialogue Tree**).
- Dialogue Nodes consist of a character name (**Node Character Name**), dialogue string (**Node Dialogue String**), and a reference to the dialogue's audio clip (**Node Dialogue Audio Clip**).
- Multiple Choice Nodes consist of a question string (**Question**) and a list of answers for the question (**Answers**).
- Answers consist of the answer string (**Answer**) and the next dialogue tree that will be in response to the question (**Dialogue Tree Response**).

DialogueTreeShim

- A **static class** that serves as a **shim** to pass **DialogueTree data** from **Non-Monobehavior objects** to the **DialogueManager**.

- In this case, you would have to create a **new DialogueTree in a C# script** and pass it to the **SetAndDisplayDialogueNodeContent(DialogueTree dialogueTree, bool changeCharacterName = false)**.

DialogueTrigger



- This Monobehavior script can be attached to gameobjects such as **NPCs** to start a DialogueTree using the **TriggerDialogue()** method.

3rd-Party Support:

1. **VR Laser Pointer (not currently in use)**
    a. A tool that incorporates the SteamVR plugin for Unity. It is used with DialogueBoxVRCanvas's button elements. The laser pointers can be activated in each hand. Pressing the select button on either hand controller while the laser is hovering over a UI button will select that option. Please refer to the VRLaserSystem.cs script which is included in the project folder.
2. **VR Touch Detection (currently in use)**
    a. A tool that incorporates the SteamVR plugin for Unity. It is used with DialogueBoxVRCanvas's button elements. There are sphere colliders located at the tips of the index finger of each hand and are only enabled when index fingers are released and not rested on the hand controller triggers. Please refer to the TouchDetection.cs script which is included in the project folder.

Future Revisions:

1. Easier DialogueTree Referencing
    a. Using a single object that contains all NPCs with DialogueTrigger components for the particular level/scene. This will be easy to locate if it is not hidden within a parent and is easy to reference from.
2. World Space Canvas
    a. There might be situations in the VR world space where the player might feel nausea when look at the DialogueBoxVRCanvas using a Screen Space – Camera mode on the Canvas settings. Having DialogueBoxVRCanvas converted to a world space object would help fix that.
3. UnityEvents called at particular Dialogue Moments

    a. There might be a situation where you want to activate an event or call a method at a certain point in the dialogue.
    b. At the very least have one unity event for the start of the DialogueTree and one for the end of the DialogueTree that can be set within a DialogueTrigger.
    c. Using a list of structs that contain a method name and object value for a parameter could be used in SendMessage(string methodName, object value) method to mimic UnityEvent Invoke() method.
4. Less Dependency from 3rd-party support
    a. This needs to stay consistent. Everything in the package contents should transfer to any project without having to rely on any 3rd party support and references in code.