Dialogue Tool Plugin Documentation

By Ryan Zagala

**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
   1. ContinueDialogueImages
      1. AutoContinueDialogueImage
         1. AutoContinueDialogueImage.controller
         2. ScrollingRight.anim
      2. InputContinueDialogueImage
         1. Bounce.anim
         2. FadeFlashing.anim
         3. InputContinueDialogueImage.controller
   2. DialogueBoxCanvas
      1. Close.anim
      2. DialogueBoxCanvas.controller
      3. Open.anim
      4. Standby.anim
2. Audio
   1. DialogueSpeech
      1. Introduction
         1. Hello there My name is Ryan.mp3
         2. I am here to tell you something.mp3
         3. Something cool.mp3
3. DialogueTreeAssets
   1. Introduction.asset
   2. Introduction\_Response\_A.asset
   3. Introduction\_Response\_B.asset
4. Images
   1. AutoContinueDialogueTemp.jpg
   2. InputContinueDialogueImage.jpg
5. Materials
   1. UI\_Materials
      1. AutoContinueDialogueRenderTexture.renderTexture
      2. AutoContinueDialogueMaterial.mat
   2. VRLevel\_Materials
      1. CenterBlock.mat
      2. Ground.mat
6. Prefabs
   1. AnswerButton.prefab
   2. DialogueBoxCanvas.prefab
   3. DialogueBoxVRCanvas.prefab
   4. DialogueManager.prefab
   5. NPC.prefab
7. Scenes
   1. DialogueToolTestScene.unity
   2. DialogueToolTestSceneVR.unity
8. Scripts
   1. DialogueSystem
      1. DialogueManager.cs
      2. DialogueSystemAssembly.asmdef
      3. DialogueTree.cs
      4. DialogueTreeShim.cs
      5. DialogueTrigger.cs
      6. MultipleChoiceAnswer.cs
      7. MultipleChoiceTemplate.cs

**How To Use:**

1. **Gettings Started**
   1. Create a folder where you want to store your DialogueTree scriptable objects. In the project example or package, the folder used to store these scriptable objects is named “DialogueTreeAssets”; however, it is recommended that create your folder outside of the “DialogueToolPackage” folder.
   2. 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.
   3. 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).
   4. Click on the Dialogue Node Elements list and choose your desire size (how many nodes you wish for this DialogueTree to contain).
   5. 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.
   6. 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.
   7. 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.
   8. For the DialogueManager.instance.StartDialogue(DialogueTree dialogueTree) method, make sure that the Play At Start Boolean is false 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.
2. **Multiple Choice and Branching**
   1. 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). 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.
   2. 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 and a DialogueTree field to place the next DialogueTree you want to play. Note: There must be at least 2 answers to select from; otherwise, the multiple choice node is treated as if it doesn’t exist.
   3. Set an answer or response in the Answer string field and set a DialogueTree scriptable object in the Dialogue Tree field. Note: If there is no DialogueTree set for an answer, then the DialogueTree will end and the canvas will close.
3. **Merging Dialogue Tree Branches**
   1. After following the multiple choice and branching steps, these branches can converge into one DialogueTree simply by attaching the next DialogueTree to each of the branch DialogueTrees.
4. **Loops**
   1. Simply attach a previously used DialogueTree to the Next Dialogue Tree field in any DialogueTree.

**3rd-Party Support:**

1. **VR Laser Pointer (not currently in use)**
   1. 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)**
   1. 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
   1. Using a single object that contains all DialogueTrees 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
   1. 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
   1. There might be a situation where you want to activate an event or call a method at a certain point in the dialogue.
4. Less Dependency from 3rd-party support
   1. 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.