Visual Novel Engine for Unity

By Michael Long

Summary

A light weight code base suitable for visual novels and simple cut scenes. Useful for any game that has dialogue between characters. Created because all of the VN frameworks I tried for Unity weren't suitable for my work on my upcoming game Triple-M.

Built using Unity 5.1+ using Unity's UI. Does not rely on any external tools or code.

Setting Up

1. Import all the assets into your project.
2. Right click anywhere in the Hierarchy inspector (the inspector where all of your gameobjects in your scene are) and mouse over the 'VN Engine' sub menu.
3. Select 'Create DialogueCanvas'. Every scene that uses the VN Engine requires a DialogueCanvas.
4. Select 'New Conversation' from the 'VN Engine' sub menu. This will create a new gameobject that will serve as the basis of our conversation.
5. Select the Conversation gameobject, right click and navigate the 'VN Engine' sub menu again and click on elements to add to this conversation.
6. Expand the 'DialogueCanvas' object and select 'SceneManager'. Drag the newly created Conversation into the 'Starting_Conversation' field.
7. Play the scene and your conversation should run.

DialogueCanvas

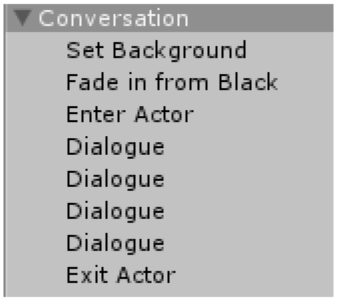
All information and interactions with this VN Engine is done through the DialogueCanvas. Feel free to user your own different canvas for other parts of your game. Be careful when changing the DialogueCanvas. If something breaks after a change, try analyzing the original DialogueCanvas, or in a worst case scenario start fresh from the DialogueCanvas prefab. You will most likely want to change the look of the UI.

Conversations

All VN Engine instructions are carried out through Conversations. Any gameobject with a ConversationManager script attached to it is a Conversation. All children of this base Conversation object may contain directions. Any Node type script may be placed as children of Conversations.

Nodes are scripts that inherit the Node class. View Node.cs for more information. Nodes are in the individual instructions carried out in a conversation. Consider them like stage directions. Have an actor appear, have them say their lines, change background, etc.

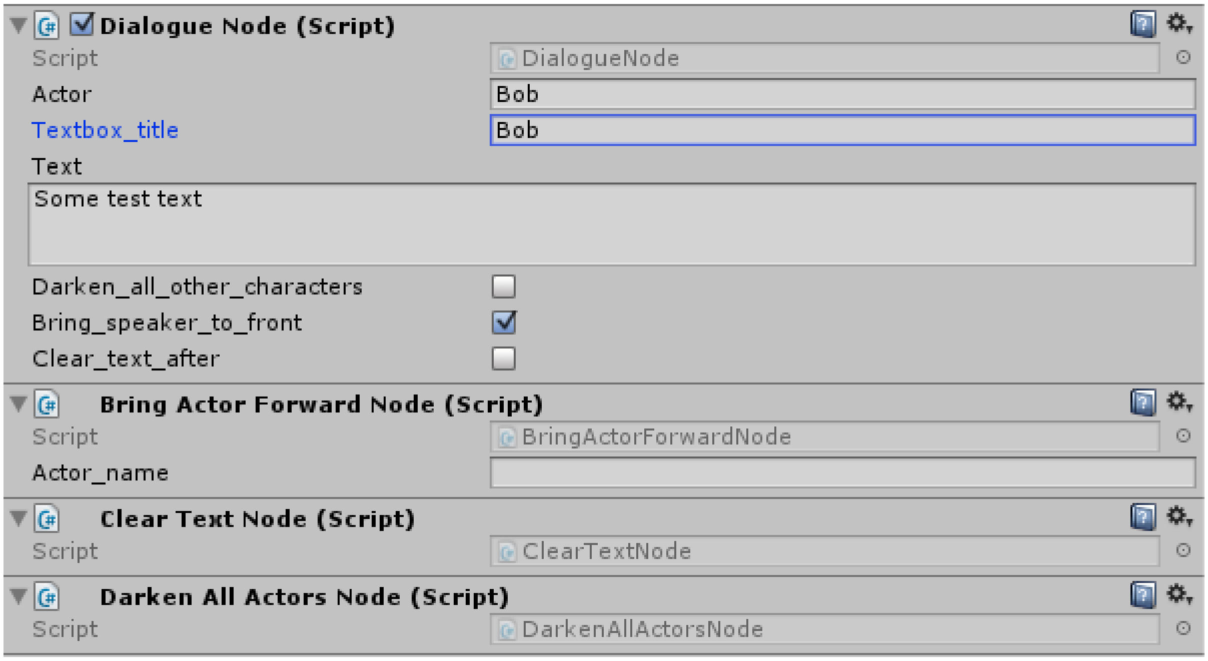
Ex:



▼ Conversation
Set Background
Fade in from Black
Enter Actor
Dialogue
Dialogue
Dialogue
Dialogue
Exit Actor

ConversationManager uses Unity's GetComponentsInChildren to find all scripts that inherent Node in its children. This means you can either place Nodes as their own child gameobjects, or add multiple Node scripts to the same object. In this case, instructions are executed going top to bottom.

Ex:



☑ Dialogue Node (Script)

Script: DialogueNode

Actor: Bob

Textbox_title: Bob

Text: Some test text

Darken_all_other_characters □

Bring_speaker_to_front ☑

Clear_text_after □

Bring Actor Forward Node (Script)

Script: BringActorForwardNode

Actor_name:

Clear Text Node (Script)

Script: ClearTextNode

Darken All Actors Node (Script)

Script: DarkenAllActorsNode

The order of execution of the above example would be: Dialogue, Bring Actor Forward, Clear Text, Darken All Actors.

Actors

Dialogue is normally spoken by characters who are on-screen. These character images are called Actors. Actors are stored in: Resources/VN Engine/Actors. This package includes several actors. The key component of an Actor is the Actor script attached to it. Actors have an Actor script and an Image. The name of an Actor in the Actor script is very important. Ensure that the Actor name in the Actor script is the same as the gameobject's name.

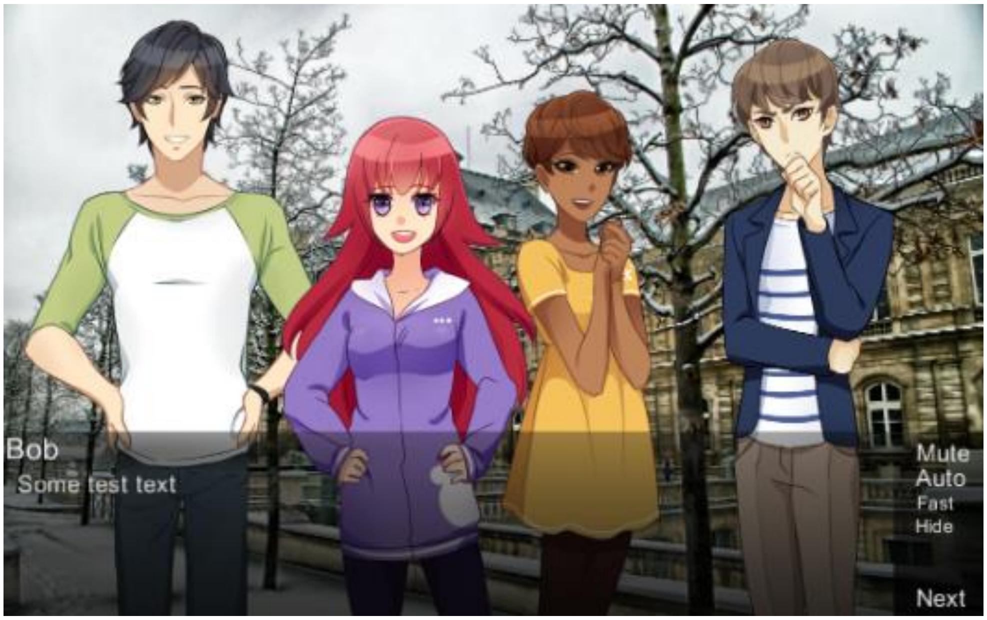
To create your own Actor, copy one of the existing Actors, change the image and scale appropriately, and change the Actor's name in the Actor script and its gameobject name.

NOTE: Image size is important, and should be considered carefully when deciding how many characters you want on-screen at the same time. It is recommended to use the 'Set Native Size' button on the Image of the actor, and then setting the Rect Transform's scale so it fits properly on the screen.

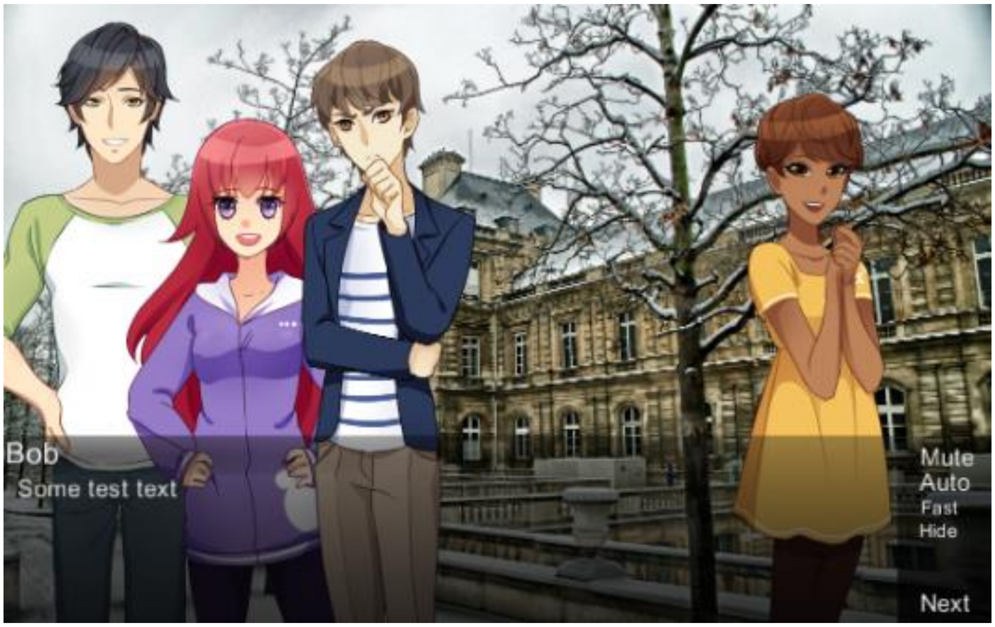
Actor Positions

Actors are positioned on either the LEFT or RIGHT side of the screen. You may place as many characters on either side of the screen as you like, but any more than 3 on a side may look crowded and start overlapping. It is recommended to have 6 or fewer characters on screen at the same time.

Actors will automatically adjust their positions when Actors enter or exit the scene. They will try to provide as much spacing as possible between them, while attempting to stay on their half of the screen.



2 characters on the LEFT, and 2 characters on the RIGHT



3 characters on the LEFT, 1 character on the RIGHT

Escape Menu

Hit the ‘Escape’ button on your keyboard to see different levels of volume for sound effects, and the speed at which characters are displayed when characters are speaking their lines.

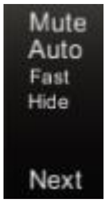
You may access the Volume in code by looking at AudioManager.cs. You may access the text display speed (text scroll speed) by looking at SceneManager.cs. You will most likely want to change the look of this settings menu.

Extending the Engine

To add more types of Nodes, copy and paste NodeTemplate.cs, change the name and fill out the 3 methods. View Node.cs for more information. Look at the other Node scripts for examples. This is meant to be extremely simple.

To change what keys are used for proceeding through dialogue, Select the SceneManager in the DialogueCanvas and change the next_button’s and superspeed_button field.

Sidebar



Menu shown at the bottom right hand side of the text window.

Mute: Mutes all audio by setting the main camera’s AudioListener volume to 0.

Auto: Automatically proceed through dialogue when the text is done displaying AND/OR the voice clip associated with the dialogue is done.

Fast: Hold down the ‘Fast’ button to quickly fast forward through dialogue.

Hide: Hides all UI in the DialogueCanvas until the user clicks any button.

Next: Starts the next piece of dialogue. Same as pressing ENTER, left clicking on the main text window.

Credits

Coding: Michael Long

Character Sprites: konett, shared under CC-By 3.0 https://creativecommons.org/licenses/by/3.0/ No changes have been made to the sprites.

http://lemmasoft.renai.us/forums/viewtopic.php?f=52&t=24893

If you wish to use these sprites, you must attribute the above author and provide a link to the CC-By 3.0 license.

UI Design: William Selby

Contact:

Michael Long, foolishmortalsgames@gmail.com, http://foolish-mortals.net