Versatile Interactive Dialogue Editor (VIDE)

By Christian Henderson
Support email: vide.editor@gmail.com

Index:

What is VIDE?

VIDE (Versatile Interactive Dialogue Editor) simplifies the creation of complex, interactive dialogues by providing the user with a simple Player-NPC node connection interface. It does not provide the user with any dedicated in-game dialogue interface, as it is, in fact, designed to be adapted to any custom dialogue interface and communicate with it. VIDE organizes data from your created dialogues and presents it to the coder in a friendlier way. Regardless, it does provide the user with a couple of examples that cover all of its available features implemented in an actual dialogue interface.

Some of the features VIDE includes:

- Create complex dialogues with the VIDE Editor. Supports dialogues between Player and NPC, Player and multiple NPCs, multiple NPCs without Player, and NPC-only dialogue.
- Optional and limitless multiple-choice comments for Player nodes.
- Tag your NPC nodes to identify the characters for your interface.
- Add extra data to a node to add special actions during a conversation, like giving an item to the player.
- Split an NPC's comment into chunks by using the
 tag.
- Node tree is not linear only. You can connect multiple nodes to a single node.
- Set the start node and end nodes of the conversation for easier handling of the dialogue interface.
- Assign the dialogues you create to your NPC game objects by using a special component, then use them to begin the conversation with them.
- All data generated from the dialogues you create are saved as interchangeable JSON files.
- Quick and simple access to all user variables and methods through VIDE_Data component. Use them in any way you want.
- When coding, simply call the BeginDialogue(), Next(), and EndDialogue() methods to automatically advance through your saved node structure.
- Control the flow of the conversation while in-game by dynamically modifying the start point and by advancing to any node you desire.
- VIDE Editor: Autosave, Undo/Redo , error detection, and file management.

VIDE is not compatible with Unity's WebPlayer platform or any Unity version below 5.

How do I get started?

For a step-by-step guide, check the **Usage** section of **The VIDE Editor** chapter next page.

To get started, it is highly recommended that you import VIDE into your project and check out the provided examples. They are located at *VIDE/Examples/*
Simply load the scenes and hit Play to see it all in action, then have a look at the scripts to understand how it all works.

Example 1: Covers player movement, interaction with NPCs, starting up conversation, checking for extra data to do special actions, NPC names, modifying the conversation's start point, and updating the in-game dialogue interface using Unity's new UI system.

Example 2: Covers the minimal setup required to get things working using the GUI class.

All example scripts are heavily commented so that you know what's going on. Make sure you also check the **Scripting API Reference** section of this document to get to know the variables and functions offered by the system.

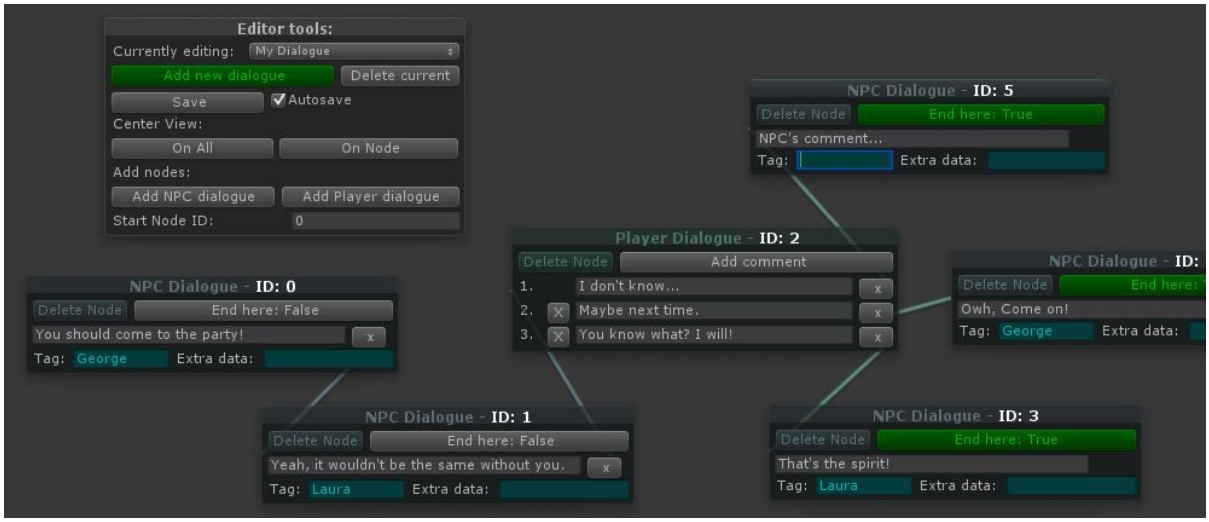
Make sure you check the **exampleUI** script. It contains various demonstrations on how to use the data that VIDE offers to modify the flow of the conversation.

But it all starts by creating your own dialogues! You can edit the dialogues in the examples at will. Just fire up the VIDE Editor and choose the dialogue you're going to modify! See next page for reference.

IMPORTANT*!*
If you want, you can move the VIDE folder to another folder within your project, but to let the software know of your change, you will have to open the VIDE_Editor.cs script and edit the ***pathToVide*** *variable by adding the extra path to the new location of the VIDE folder. If you don't do this, the VIDE Editor will not work.*

The VIDE Editor

This is where you will be creating your dialogues by adding **comment nodes** and connecting them. You can make **Player->NPC** dialogues, **Player->NPCs** dialogues, **NPC->NPC** dialogues, and **NPC->NPCs** dialogues.



The editor consists of three main window types:

1) **Editor Tools window:** Create, modify, delete, and save dialogues. From here, you also have buttons to create NPC and Player Nodes, and you can specify the conversation's default start node. The conversation can start either with a Player node or with an NPC node. Also, you have options to center the view.

2) **NPC comment node:** This node contains an NPC's comment. If you want to flag that comment as a conversation endpoint, you need to declare so by setting the 'End here' button to true, otherwise, it needs to be connected to a Player or NPC node. You can also fill the 'Extra data' field to pass relevant information to *VIDE_Data.nodeData,* like "item", for example. The Tag will help you identify the NPC.

3) **Player comment node:** Unlike the NPC node, you can add more than one comment to this one. Each comment will lead to a different NPC node (or to the same one). There cannot be any unconnected comments.

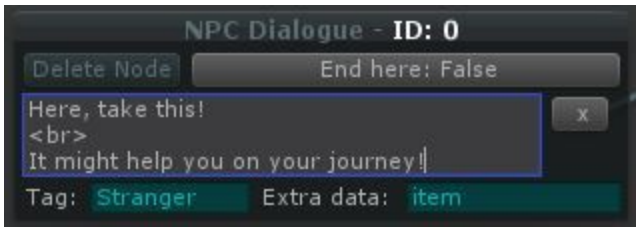
Every comment node will have a unique ID that you can use to set the start point and to read from *VIDE_Data.nodeData* to meet certain conditions in your code.

Usage:

1. In Unity, go to *Window* > *VIDE Editor* to open up the editor. You can also open it from the VIDE_Assign component.
2. Click “Add new dialogue”. Name your new file and click ‘Create’.
3. Click “Add NPC Node” or “Add Player Node” to create your first node.
4. Click and hold the Link button, then drag the cursor to an empty space and release. This will create the corresponding new node and automatically connect the current node to it. You can also release while on top of another node to connect them. If you want to connect an NPC node to another NPC node, use the “Add NPC Node” to create it, then connect the nodes.
5. Continue building your conversation, making sure there are no unconnected comments (Unless it is a single NPC node flagged as End Here: true).
6. Make sure you specify the Start Node in the Editor Tools window with an existing node ID.
7. Save your dialogue by clicking the ‘Save’ button. Even if autosave is ticked, it won’t work unless you have saved it at least once. Autosave saves on every change you make to the dialogue, excluding moving nodes and deleting them.
8. In your game Scene, attach a **VIDE_Assign** component to your NPC game object.
9. In the Inspector for the VIDE_Assign component, select the desired dialogue for that NPC using the dropdown box. You can also add a custom name for the dialogue.
10. Now, have or add a script that will manage all UI-related stuff in your desired game object. Within the script, make sure you add a variable of type VIDE_Data, then attach a component for it. With this, you will be able to read all of the Node data to create your UI. Check the **How it works** chapter for a more detailed explanation.

Tips and good-to-knows:

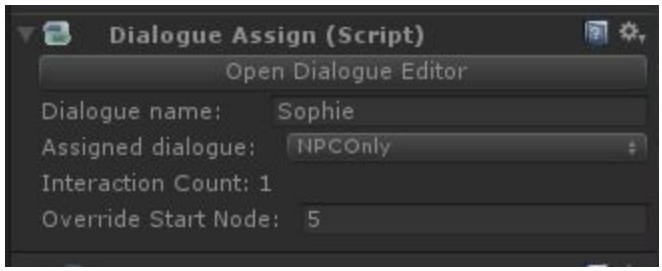
- For NPC comments, you can use the **
** keyword to split the comment into a series of comments.
- Right click on empty space to invoke the Editor Tools window.
- Click and drag on empty space to drag all of the nodes.
- Click and drag on a node to drag it, also works with Editor tools window.
- Release a connection line on empty space to create a new node and connect automatically.
- VIDE will not automatically center your dialogues when you load them to the Editor. If you can't see the nodes you've created, center the view by clicking the "On All" button in the Editor Tools window.
- You can have more than one conversation in a single dialogue file, but remember you can only have one default start point. To start in a different node instead, use the SetNode() method at the beginning of the conversation, or set **Override Start Node** field in the Inspector for the **VIDE_Assign** component to a node ID. Check the Example 1 **exampleUI** script for reference on how this can be achieved.
- You will not be able to save or autosave if you have errors in your dialogue:
    - Disconnected comments/nodes
    - Start Node ID is nonexistent.
- You can duplicate your dialogue and save it with a new name if you click the 'Save' button.



How it works

Initially, you'll require 3 things to get the system working:

- Saved dialogues made with the **VIDE Editor**.
- Your custom dialogue interface with a ***VIDE_Data*** component attached.
- ***VIDE_Assign*** component attached to NPCs or to whatever game objects you want. You will need this component to start the conversation by calling BeginDialogue() on the VIDE_Data component.



The VIDE_Data component contains a variable called **nodeData**. This variable of type **NodeData** stores all of the current node's data. At the beginning of the conversation, the current node is equal to the Start Node you selected in the VIDE Editor (unless you set the Start Node Override). When you call Next() method, the conversation will go one step forward, depending on your connections. nodeData will then be populated with data from the next node which will be now the current one.

Essentially, you're constantly reading the contents of *VIDE_Data.nodeData* to do whatever you want in your dialogue interface using its data. All you need is a reference to the VIDE_Data component:

```
public class dialogueUI_template : MonoBehaviour
{
    [System.NonSerialized]
    public DialogueData dialogue;
```

1. User calls *VIDE_Data.BeginDialogue()* method on the VIDE_Data component to begin the conversation with the NPC. The method requires the user to send the **VIDE_Assign** component attached to an NPC or other game object. This will populate the *VIDE_Data.nodeData* with data from the first Node that begins the conversation.
2. User uses the data in VIDE_Data.nodeData to customize the in-game dialogue interface.
3. User calls *VIDE_Data.Next()* on the VIDE_Data component to populate VIDE_Data.nodeData with the data from the next Node in the conversation.
4. User uses the new data in *VIDE_Data.nodeData* to customize the in-game dialogue interface.
5. Step 3 & 4 repeats until user reads the *nodeData.isEnd* variable to know when to call the *VIDE_Data.EndDialogue()* method and clean the dialogue interface.

It is very important that you check the **Scripting API Reference** next chapter in order to understand the methods and the contents of **NodeData** class. Once you get to know the variables and methods offered, you'll know how to incorporate them to your UI script.

Scripting API reference

VIDE_Data

Script component attached to your dialogue UI game object. This component will store all the data regarding the currently loaded dialogue. Store a reference variable of it within your UI script to access the following:

Functions:

public NodeData **BeginDialogue(**VIDE_Assign diagToLoad**);**
Loads up the dialogue just sent. Populates the **nodeData** variable with the first Node based on the Start Node. Also returns the current **NodeData** package.

public void **EndDialogue();**
Wipes out all data and unloads the current VIDE_Assign, raising its **interactionCount**. Do not call BeginDialogue() again if you haven't called this yet.

public NodeData **Next();**
Populates **nodeData** with the data from next Node based on the current **nodeData**. If current **nodeData** belongs to a Player Node, make sure **nodeData.selectedOption** is correctly set before calling Next(). If current nodeData belongs to an NPC Node with multiple comments (when using
), calling Next() will advance through those comments before getting to the next Node. If current **nodeData** was flagged as 'End Here', calling Next() will set its **isEnd** variable to true and will not get any new Node. Also returns the current **NodeData** package.

public NodeData **SetNode(**int ID**);**
Ignores current **nodeData** state and jumps directly to the specified Node, be it Player or NPC Node. Make sure the ID exists.

Variables:

public bool **isLoaded;**
Is there a dialogue currently loaded?

public int **startPoint;** (ReadOnly)
The ID of the default Start Node set in VIDE's Editor Tools window. Returns 0 if there's no dialogue currently loaded.

public VIDE_Assign **assigned;**
Reference to the currently loaded VIDE_Assign component. Variable is null when no dialogue is currently loaded.

public NodeData **nodeData;**
Variable containing all of the current Node data you'll need to set up your dialogue interface.

NodeData class

This class stores all of relevant variables of your current node.

public int **nodeID;**
The current Node´s ID.

public bool **currentIsPlayer;**
Is this current Node a Player Node?

public bool **isEnd;**
Is it the end of the conversation?

public string[] **playerComments;**
An array of strings with all of the Node's player comments.

public string[] **npcComments;**
An array of strings with all of the Node's NPC comments. If not using
, the size of this array will always be 1.

public int **npcCommentIndex;**
The current index of the focused NPC comment when there're more than one. It is always 0 if not using
.

public int **selectedOption;**
The index of the currently selected player comment. When calling Next() on a Player Node, the method will read this variable to know where to go next.

public string **extraData;**
The string of extra data declared on the VIDE Editor.

public string **tag;**
The tag you set for the Node on the VIDE Editor.

VIDE_Assign

Script component attached to the game objects you want to have a dialogue loaded from. Either by using **VIDE_Data.assigned** or by creating a reference to the component yourself, you can have access to the following:

Functions:

public string **GetAssigned();**
Returns a string with the name of the currently assigned dialogue (Not the custom name assigned on the Inspector).

public bool **AssignNew(**string dialogueName**);**
Assign a different dialogue to this VIDE_Assign. The dialogue you're going to assign must exist, otherwise the method will return false. Doing this is the same as selecting it from the Inspector. Do not include the file extension for the dialogue name.

Variables:

public int **interactionCount;**
This variable begins on zero. Everytime you call EndDialogue() on VIDE_Data while having this VIDE_Assign currently loaded, interactionCount will increment by 1. In the end, it keeps track of how many times you've interacted with this game object in particular.

public string **dialogueName;**
The custom name for this dialogue. Can be set from the Inspector.

public int **overrideStartNode;**
Default is -1. When changed, the assigned dialogue's Start Node will be ignored and will use this one instead. Make sure the ID exists. This is an in-game change only, it does not modify the actual dialogue's Start Node. Set it back to -1 to use original Start Node. You can also set it from the Inspector.

Changelog

Version 1.0.1

- Fixed issue with titleContent to add support for Unity 5.0
- Fixed VIDE_Assign component not loading the dialogues correctly when importing the asset.
- Fixed NPC texts not properly clearing (ExampleUI.cs)
- Changed Canvas Scale Mode to constant pixel size for better consistency between aspect ratios.
- Renamed all classes/scripts to prevent duplicated definitions.
- Cleaned some scripts.
- Updated documentation

If updating from the initial release, backup your Dialogues folder (VIDE/Resources/Dialogues), update, and replace the Dialogues folder. You might have to set the VIDE_Assign (DialogueAssign) components again.

Version 1.0

Initial release.