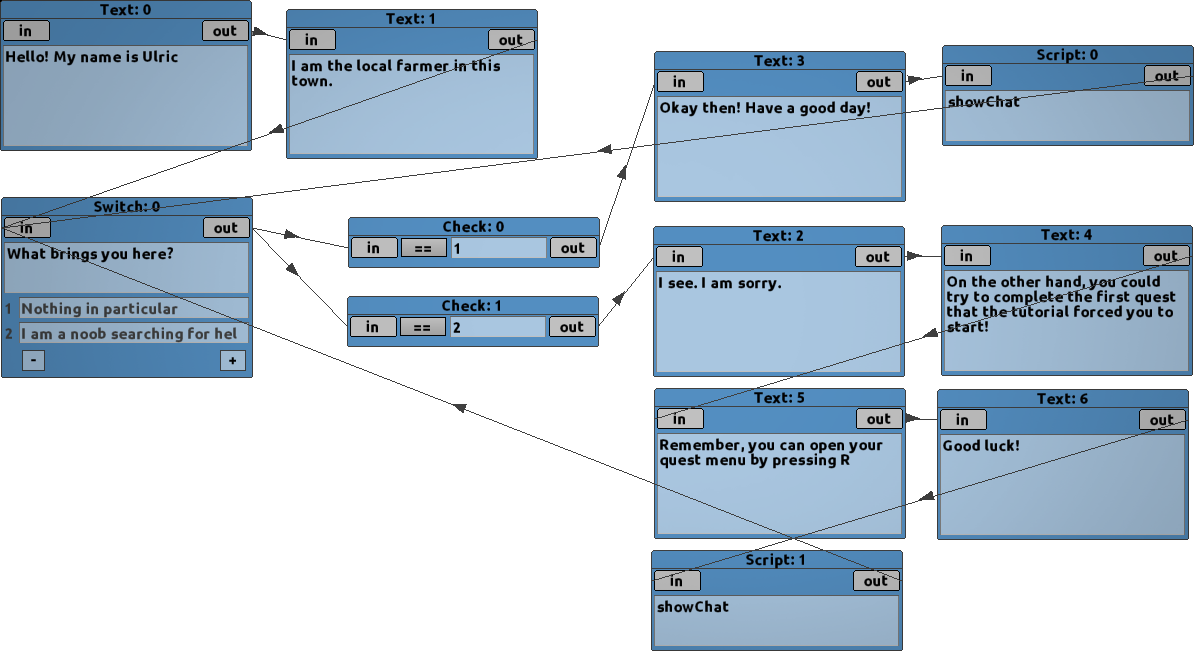
Easy Dialogue System



By NoobsWeStand

Summer 2016

Table of Contents

|  |  |  |
| --- | --- | --- |
| **1** | **Introduction** | **3** |
| 1.1 | Alpha | 3 |
| 1.2 | Requirements | 3 |
| **2** | **Interface** | **4** |
| 2.1 | Nodes | 6 |
| 2.2 | In/Out | 7 |
| **3** | **Nodes** | **7** |
| 3.1 | Text | 7 |
| 3.2 | Script | 8 |
| 3.3 | Switch | 9 |
| 3.4 | Check | 9 |
| 3.5 | Don’ts | 9 |
| **4** | **Setup** | **10** |
| **5** | **Updates** | **10** |
| 5.1 | Work Flow | 10 |
| 5.2 | BB TextBox | 11 |
| **6** | **End Notes** | **11** |

1 Introduction

This program, paired with the example, is supposed to make branching dialogue systems easy to make in Game Maker Studio. Most help that people have given on the forums is a “do it yourself” with a couple of links to things they thought might be useful. While there are some branching dialogue systems floating around there for Game Maker Studio, none of them were satisfactory for my tastes. It would be really hard for a novice to easily obtain/make a branching dialogue system. It would also take a considerable amount of time for anyone else, who wanted to easily make a branching dialogue system, to create one. This project can be both used by novices and advanced users.

I already had the base of the engine when I was developing a MMO in Game Maker 8. All I had to do was port that over to Game Maker Studio, clean up some things, and it pretty much worked. The hard part, what this whole project is about, was making an application that could write the files in such a way that the base engine could read and understand it. Branching dialogue and quests were made by hand (editing the text files) before. This was a huge pain, in that I would have to keep track of where all branching paths were going.

1.1 Alpha

This project is still in its very early stages of development. With that, come lots of bugs and problems. With only around 2 weeks of development (as of writing this), I do not feel like a full release would be the right choice.

I want this to be a community project, where people can help me by submitting errors, and bugs. I will also be releasing the source code to this, so if you want, you are allowed to look through it, and suggest improvements. The code is in no way organized to be the most readable or acceptable at this time. I will be improving that as the project moves forward.

1.2 Requirements

-Game Maker Studio

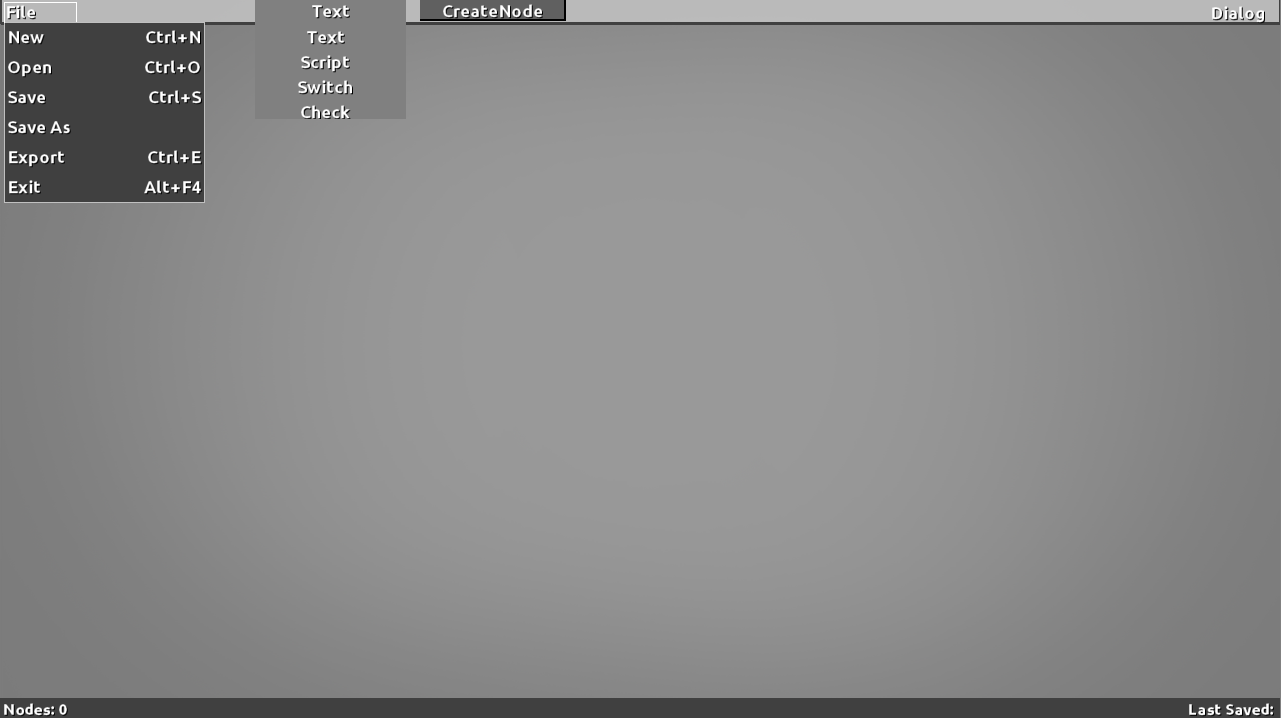
-Basic GML knowledge

-Time

1.3 Controls

|  |  |
| --- | --- |
| Middle click / Arrow Keys | Pan Camera |
| Double Left Click | Zoom into Node |
| Double Right Click | Delete Node |
| Scroll Wheel | Zoom In/Out |

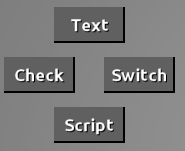
2 Interface



The interface is pretty self-explanatory

First, in the upper left hand corner is the file menu. This has the basic functions that anyone would need.

Second, just to the right of the file menu is the Node dropdown menu. This is where one would switch which type of Node they would create when clicking the “CreateNode” button.

Alternatively, by clicking anywhere in the work area, a pop-up display will appear.

Also, when dragging out from a Node, one can release their mouse button anywhere in the work area to get another pop-up display with all of the possible options. Once an option is picked, then the two nodes will be automatically connected

2.1 Nodes

Nodes are super important! They make up the entirety of the interface. There are currently 4 different types of Nodes. (more may be added in the future if needed)

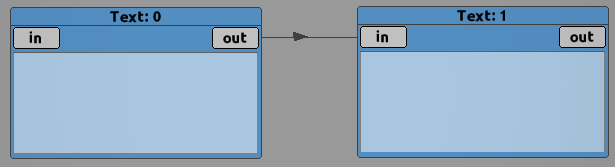
|  |  |
| --- | --- |
| Text | Script |
|  |  |
| This is the most basic type of Node.  Any kind of text can be put into this node. There is no word limit!  This box (right now) does not have a spell-checker, or very good formatting options.  It has the ability to scroll if the length of your text requires it | This node can execute simple scripts within your game. It is also able to do very simple IF statements.  For example, if you want to give the player a lemon after they complete some kind of quest involving lemons, this is where you would code that in. |

|  |  |
| --- | --- |
| Switch | Check |
|  |  |
| Arguable one of the most important Nodes for a branching dialogue system!  The Switch Node has several parts which make it unique.   1. Main text area – Used to pose the player with options. This would typically be something like, “Do you want to date me?” 2. Options – The amount of boxes changes on how many different options you want to have. With the + and - buttons, one can add or subtract the amount of options the player is given. In the boxes, one would write the available options that are given to the player. In the previous example of “Do you want to date me?”, one could allow the player to give the answers, “Take me”, “You smell”, “Get away you creep!” | The Switch and Check Node always go next to each other. This process will be explained in depth later.  Think of this Node as a simple IF statement.  There are multiple parts to this Node as well.   1. Main text area – used to compare the input to the check area, and then provide and output 2. Check Area – (currently is only limited to ==) used to check the input to various equalities or inequalities. |

2.2 In/Out

You may have noticed that each Node has an “In” and an “Out”. These are used to connect each Node to one another. With this simple yet powerful connection device, a near infinite amount of possibilities can arise!

To connect one Node to another, just click and drag on the “Out” of one, into the “In” of another.



3 Nodes

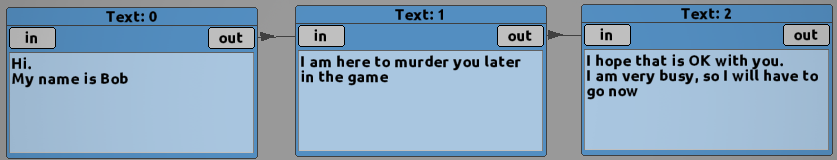
As stated before, the Nodes are the building blocks to making the dialogue system work. This section will introduce how to properly connect each Node to one another.

An important property of the Nodes is that they will attempt to display what will be saved, and what will not. In other words, if you have a poorly structured dialogue tree, each Node will determine, by its connections, whether it will be exported or not. This is displayed by simply making the poorly placed Nodes transparent. You might notice this throughout the tutorials in this manual.

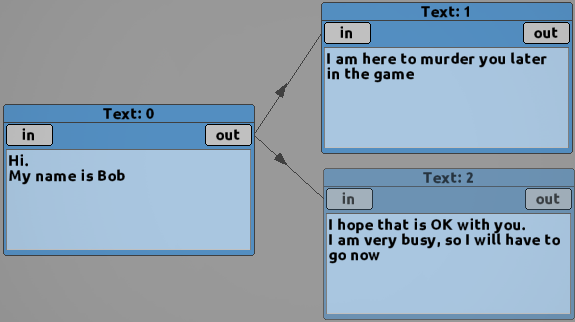
3.1 Text

The text Node is one of the simplest nodes. It is the fundamental building block to any dialogue. One could make a whole conversation with just text Nodes.

When connecting text Nodes to each other, it is important that the dialogue is linear. If you connect one text to two other texts, the dialogue system will not know which one to choose. It is important that when building a dialogue, that the text Node only has 1 input and 1 output (unless connected to a script Node)

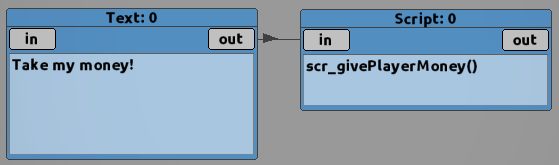


This is an example of a well-structured text Node layout.

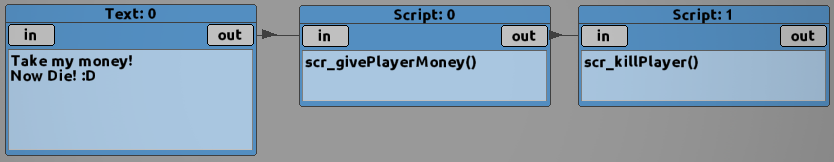


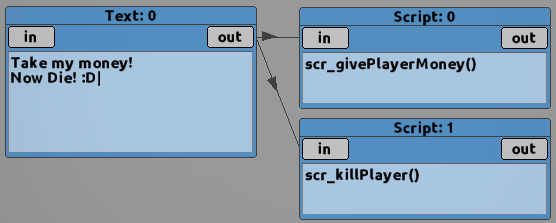
This, however, is not properly structured, and might not export the way you would expect.

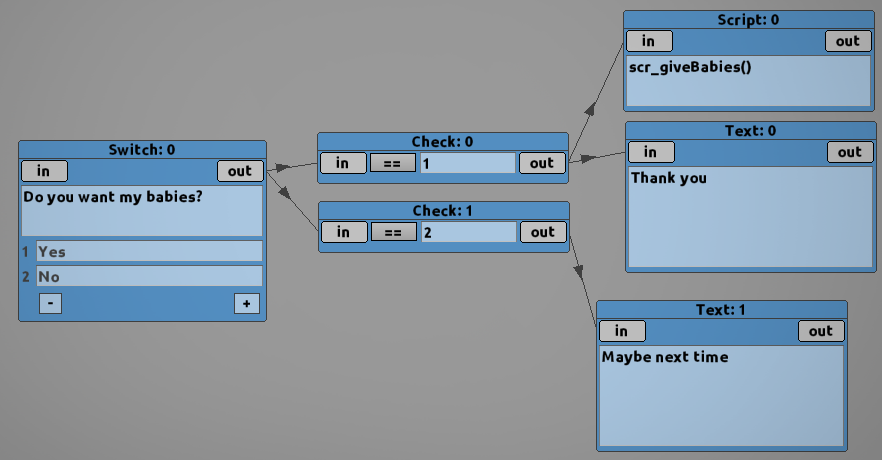
3.2 Scripts

 The script Node was made to allow the user more control of how the dialogue system works. The script can be attached, or stacked on top of pretty much any other Node.

This would be a valid configuration, as long as you had a script called scr\_givePlayerMoney() within your game

If you want to do more than 1 script, then you can stack the script Node on top of itself.

It does not matter how you do this. Either of the above configurations would work just fine.

You can also apply them to the Switch Node (check next section for more details)!

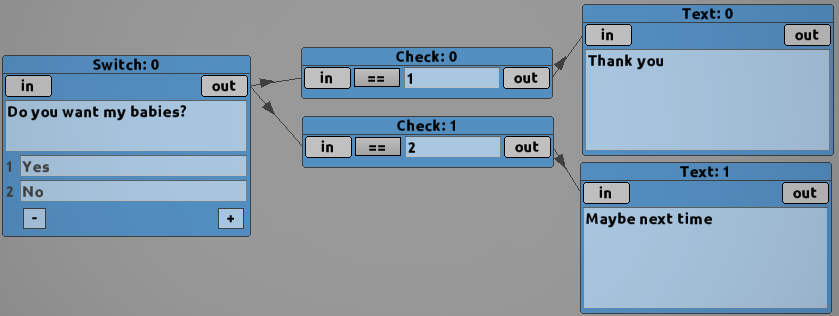
As seen in the diagram, the scr\_giveBabies() will be called if the player clicks the “Yes” option.

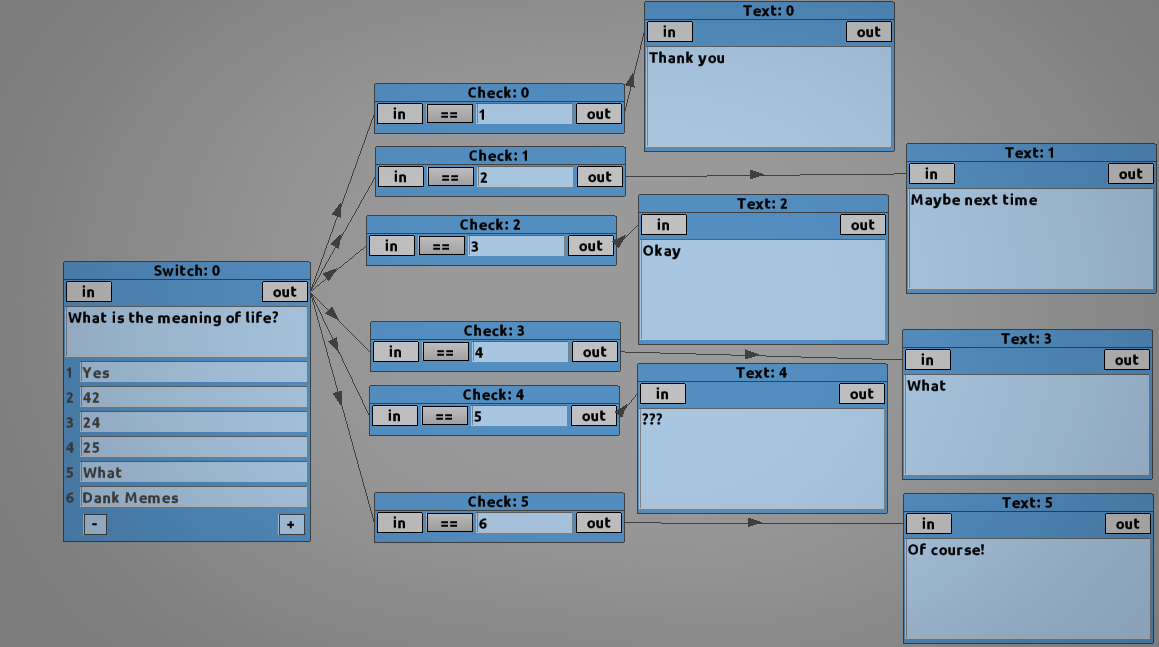
**Note: When attached to a Text Node, the scripts will be executed AFTER the player clicks the “continue” button!**

|  |  |
| --- | --- |
| **Command** | **Usage** |
| teleNPC(x,y) | Teleports the NPC to a specific x,y position |
| telePlayer(x,y) | Teleports the Player to a specific x,y position |
| showChat | Toggles the text box off. So, if you are done with some dialogue, and want the conversation to end, just slap this on the end. |
| line(on0,on1) | Used to jump to a specific dialog box. When you export, or test your project, on0 and on1 can be found on the top of the Node.  These numbers will probably change as you change your dialog tree!  Do not rely on this command, and only use sparingly |
| quest(questID,questStep) | “objControl.quest[questID]=questStep”  That is the literal code. Just sets the quest, with the given questID, to the questStep. This will affect how the “convo” system works |
| convo(questID,questStep) | This jumps to the “convo” with the two arguments  https://i.gyazo.com/5c392366cfdcc436dee83742152d5f07.pngIn this example, if you have a script with “convo(0,0)”, it will jump to the first Node within that convo group—in this case “Hello there!” |
| label(str) | This will set a pointer in the dialog tree, so when you call “goto(str)”, it will automatically find the label, and jump to that Node |
| goto(str) | This will jump to the corresponding label. If the label does not exist, the editor will tell you |

3.3 Switch

This is arguably one of the most important Nodes for building a branching dialog system! This gives the player a chance to decide for themselves, and allows games more interesting options!

 For every option, there must be a corresponding Check Node attached. The Switch Node and Check Node go hand-in-hand. Each option is numbered, starting at 1.

Notice how the text in the Check Nodes corresponds with the numbering of the options in the Switch Node.

3.4 Check

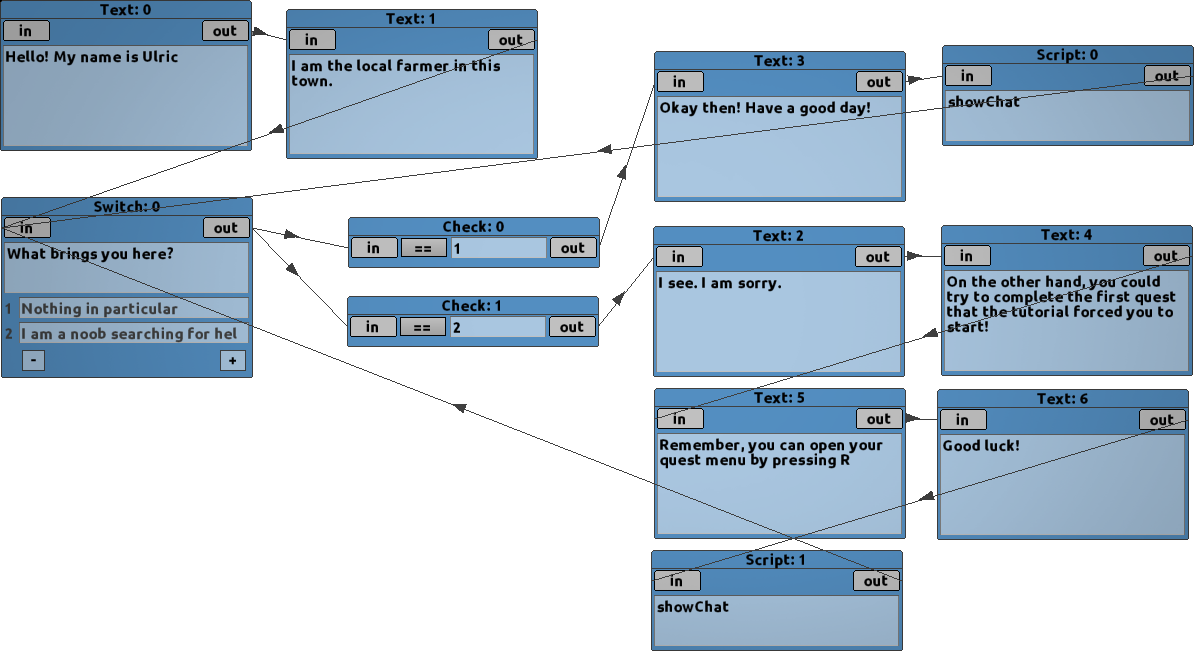
Refer back to Switch Node.

This Node is specialized in that it really only has 1 purpose.

There is also a deeper element to the Check Node. There is a drop-down menu that will allow the user to switch how it functions with certain types of input (currently not in!).

3.5 Don’ts

With a wide variety of possibilities, there are going to be certain combinations of connections that do not make sense.

Try to avoid loops. If your dialogue loops, there may not be a clear beginning, and then the program will not know where to start

This dialogue has a clear beginning at the Text Node that contains, “Hello! My name is Ulric”.

If that the two beginning Nodes did not exit the program would not know where to begin, and therefore not export.

4 Setup

What is the point of all of this if you don’t know how to put it into your game!

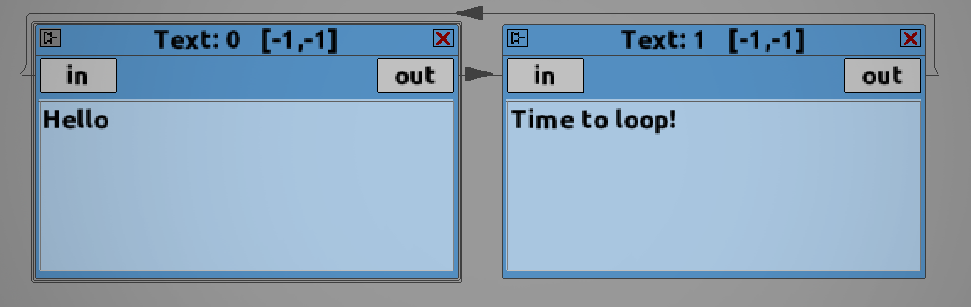
This manual will be working off of the provided example that can be found [here](https://www.dropbox.com/s/qh7clp7xda97ikv/Example1_2.gmz?dl=0)

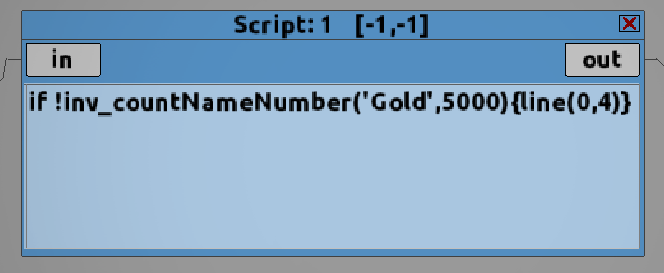
5 Updates

Due to this project being in the alpha stages, there are going to be many updates!

Here is where the new functionality will be explained!

**Text Looping!**



You can now make a dialogue tree with no clear beginning node! By clicking on the little pin on the right hand corner on the Node, it is now possible to tell the program while dialogue box you want the beginning of the conversation to start at!

**Smart Scripts!**

With smart scripts, it is now possible to test your dialogue trees without actually putting them in your game first!

If the program runs over a script it does not have, then it will ask you for what it might return, and then continue down the dialogue tree.

**BB Text Boxes**

You can now add cool effects with simple inline text commands. Here is a list of commands:

|  |  |
| --- | --- |
| [color=c\_red] | Changes the color of the text. You can either use GM’s c\_ values, or put your own hex with $ infront of it (eg. $4455ed) |
| [font=fnt\_name] | Changes the font. The font must exists in your game as an asset |
| [alpha=0.75] | Changes the alpha value of the text |
| [speed=0.1] | Changes the speed of the typewriter effect |
| [waveA=1] | Changes the wave amplitude |
| [waveF=1] | Changes the wave frequency |
| [waveO=25] | Changes the offset of the wave effect per character |
| [shake=1] | Shake amount in both X+Y |
| [shakeX=3] | Shake amount in only the X direction |
| [shakeY=2] | Shake amount in only the Y direction |
| [hsv=1] | Shifts through the HSV values at the given rate. This will override the [color] tag |
| [var=xxx] | Replace “xxx” with whatever variable that you have created inside the editor, and it will insert whatever is inside the variable |

6 End Notes

This project is in Alpha, and will contain bugs, errors, and typos! If you find any, please tell me via the GMC: [NoobsWeStand](https://forum.yoyogames.com/index.php?members/noobswestand.81/)

Thank you for checking this out!