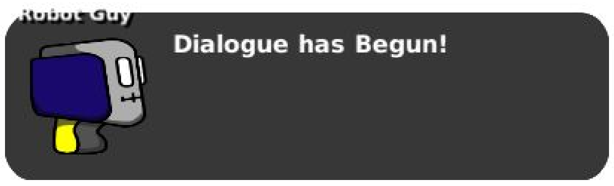
Dialog Canvas Documentation

Version 1.0



Introduction

The Dialog Canvas is a simple, intuitive way to quickly integrate **Dialogs** and **Dialogue** into your game! It uses a node-based editor and a premade UI canvas to create multi-slide message viewable in-game, with easily manageable NodeCanvas files.

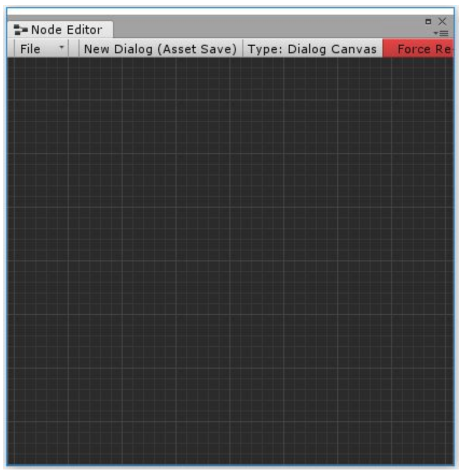
Note: Dialog is the box. Dialogue is a conversation.

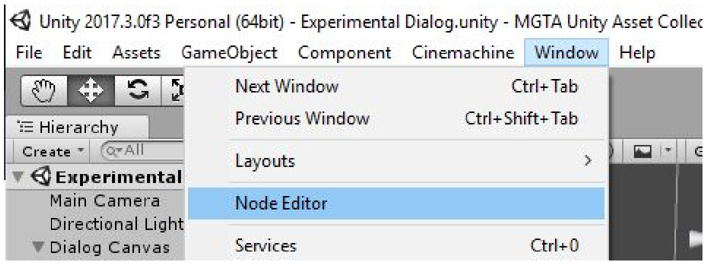
Installation

Download the MGTA Dialog Package, and open it through **Assets > Import Package > Custom Package**. Unpack the assets into your project using their default locations.

Creating a DialogNodeCanvas

Open a Node Editor window, through the **Window** menu at the top of the Unity Editor.





The window you see now should look somewhat like the one seen to the right of this text.

To create a new Dialog Canvas, hit the **File** button in this window, choose **New Canvas,** then **Dialog Canvas**.

Note: *you can use the middle mouse wheel to pan and zoom around the canvas. You can also pan by left-clicking and dragging on empty canvas space.*

Adding/Removing Nodes

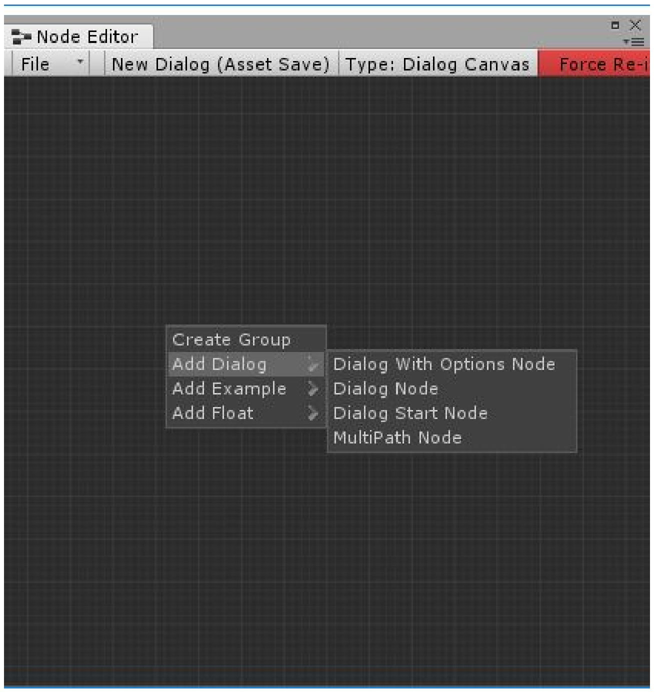
Inside the Node Editor, right click anywhere to open the **Context Menu**.

Nodes can be **deleted or duplicated** by right clicking them, and choosing the option.

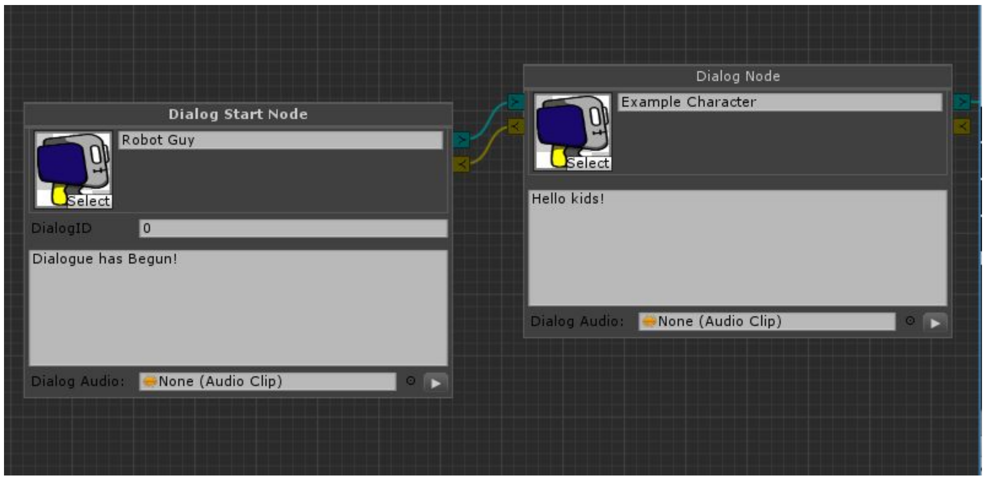
Non-dialog nodes are currently incompatible with Dialog Node features.

Setting Up a Dialog Canvas

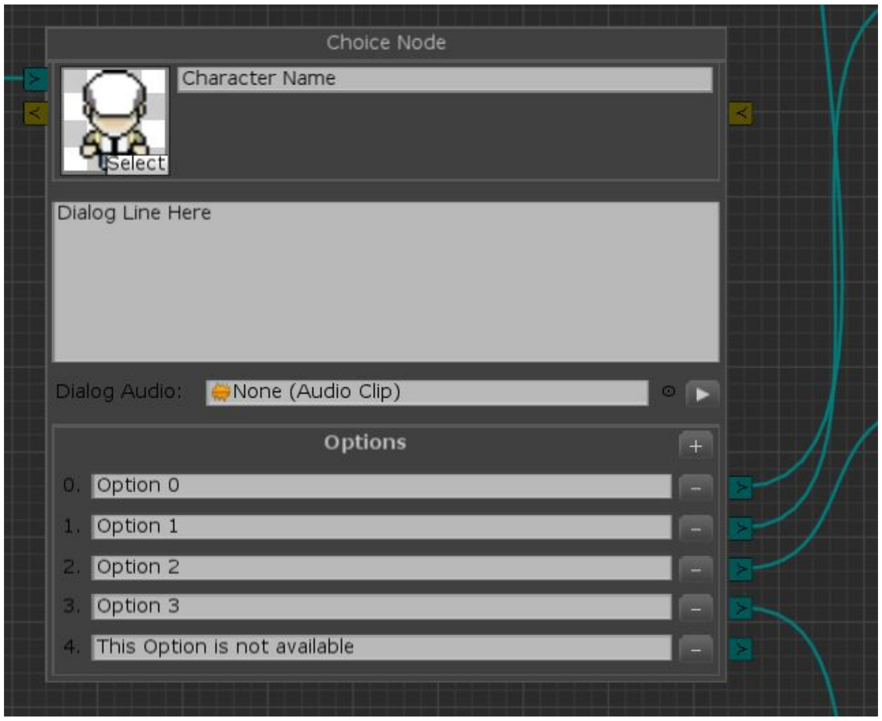
The first node you'll need is a **Dialog Start Node.** These feature a **DialogID** which is necessary to look up the proper Dialog Tree from the canvas when loading it to the Dialog Manager. Each Start Node in the canvas should be given a unique ID value.



From there, you can start making connections to **Dialog Nodes** and **Choice Nodes,** which will make up the rest of the Dialog Tree. **Blue** paths represent *forward* movement in the dialog, whereas **Yellow** paths represent *backwards* movement in the dialog. **For now, only Blue paths are 100% supported, and Dialog Audio is disabled.**



To connect nodes together, left-click on a **connection point** and drag to another node's corresponding endpoint. When the dialog runs, it will progress from the start node to whichever node is the last in this chain.



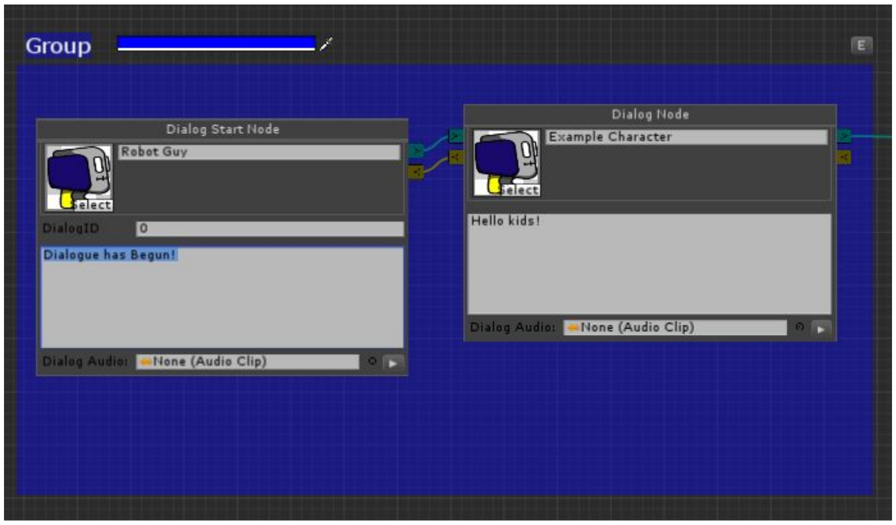
Choice Nodes, as seen above, allow for multiple branching paths during a Dialog. **At this time, only up to four options are supported (0-3)**. Fewer options are supported as well.

Character Portrait

The Character Portrait is completely optional. The Dialog UI actually features two similar layouts, one that uses the image, and one that does not. They can be customized independently.

Dialog Groups

You can also create Groups through the **Context Menu.** They are mainly to give a colorful backdrop to help visually separate parts of your canvas. Drag the sides to resize. Color and name can be edited by hitting the **E** button in the top right corner of the group.



Saving Your Dialog

Once you are finished editing, use the Node Editor **File Menu** to save your canvas. It will tell you to **Save As** if you haven't previously saved it. It will be stored as a .asset file in your project.

If the Gizmos folder was included in the package import, it should have a fancy custom icon!

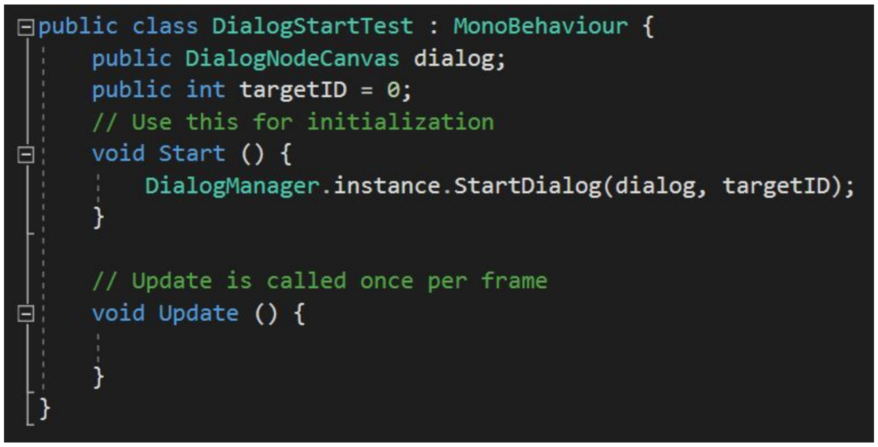
Using The Dialog Manager

Once you are done creating your Dialog Canvas, it's time to load it into a UI Canvas. Yes, the use of 'Canvas' is confusing.

Since retrieving the data from each node can get a bit complicated, I've created helper scripts and a UI Canvas prefab that will load up the data for you. They're located in the **MGTA Assets** Folder that should have been included when importing.

Add this prefab to your scene, and make sure you have a UI **Event System** in there as well. You will likely need it if you are using Choice nodes to select options. (You can add an event system through GameObject > UI > EventSystem)

Note: ONLY ONE Dialog Canvas / Dialog Manager is needed for your scene!

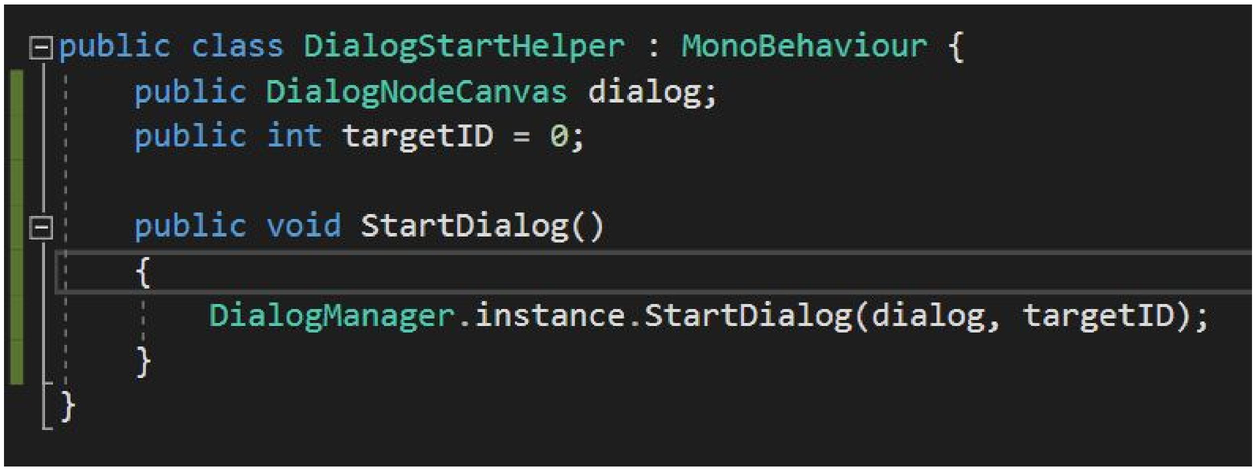


```
public class DialogStartTest : MonoBehaviour {
    public DialogNodeCanvas dialog;
    public int targetID = 0;
    // Use this for initialization
    void Start () {
        DialogManager.instance.StartDialog(dialog, targetID);
    }

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

    }
}
```

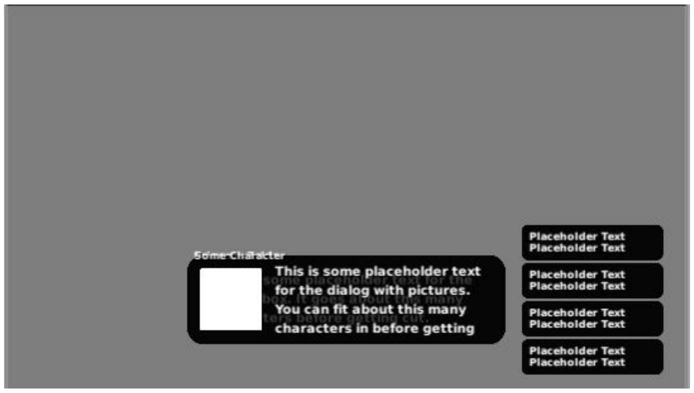
The **DialogStartTest** script, when added to a GameObject, will call the StartDialog() method from whatever DialogManager exists in your scene. Feed it a reference to a NodeCanvas and tell it which ID to start from, and the DialogManager should handle the rest.

The StartDialog method can be called from **anything!** The DialogManager uses the singleton pattern, maintaining a static reference to itself during gameplay. Feel free to write additional scripts that use this method.

You can also use the similar **DialogStartHelper** when working with Unity Events, since they can't handle multiple parameters for some reason.



```
public class DialogStartHelper : MonoBehaviour {
    public DialogNodeCanvas dialog;
    public int targetID = 0;

    public void StartDialog()
    {
        DialogManager.instance.StartDialog(dialog, targetID);
    }
}
```


Some Character

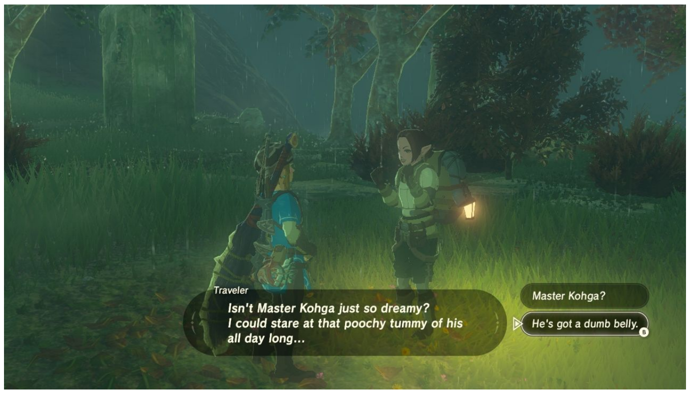
This is some placeholder text for the dialog with pictures. You can fit about this many characters in before getting

Placeholder Text
Placeholder Text

Placeholder Text
Placeholder Text

Placeholder Text
Placeholder Text

Placeholder Text
Placeholder Text


Traveler

Isn't Master Kohga just so dreamy?
I could stare at that poochy tummy of his all day long...

Master Kohga?

He's got a dumb belly.

Customizations

The appearance of the dialog, by default, resembles the UI used in Legend of Zelda, Breath of the Wild.

However, you can customize many of its features!

- You can change the key used by the Dialog Manager to progress Dialog.
- The background sprites (*On the Dialog Panel and Picture Dialog Panel Objects)* can be replaced and recolored.
- Text Fonts can be swapped out and recolored on all objects.
- The **Dialog Handler** script attached to the Panels can have its **continuation prompt** text swapped out for a different key, if you choose not to use the space bar to progress dialog. You can also change the speed at which letters are "written" into the box.
- You can reposition any of the objects to wherever you would like on-screen.

WARNING: Do not delete any objects from the Dialog Canvas, or remove components. The scripts enabling it during gameplay rely on references to each of its parts. Deleting or removing objects will break the scripts.

Controller Support

Work In Progress

Credits

The Dialog Canvas is an extension of the Seneral Node Editor Framework found here: https://github.com/Seneral/Node_Editor_Framework

Developed by Tyler van Vierssen for the Mason Game and Technology Academy, 2018