

Node based dialog system



Dialog Node Based System (Unity asset tool)

Last Updated: @02/24/2024

Publisher: cherrydev



About: Nodes-based dialog system asset is a tool that allows game developers to create branching dialog trees for their game characters easily. It is a visual editor that allows users to create and connect nodes to build conversations between characters in their game. Each node represents a piece of dialog, and the connections between nodes determine the flow of the conversation.

A

Node Editor presents an editable graph, displaying nodes and the connections between their attributes. You can create simple sentence node or node with answers to build your own dialog.

You can check video tutorial on youtube, but this **video is little outdated**: [Click here](#).

1 Node Editor

STEP 1: How to open node editor window

1. Right-click to the **assets folder**.
2. Go to **Create/ScribableObjects/Nodes/NodeGraph**.
3. Double click on your new **DialogNodeGraph** assets and you are done!
4. You can also go to **Window/DialogNodeBasedEditor**, but you still have to create **NodeGraph** SO and click on it.

STEP 2: Orientation

1. By holding the **left mouse button** you can **move** around the window.
2. **Right mouse** button opens **context menu**. It has options such as "Create sentence node", "**Create answer node**", "**Select all nodes**" and "**Remove selected nodes**".
3. To **connect** nodes together, hold down the right mouse button on the node and drag it to.

2 Sentence Node

1. **Sentence node** can only join **one** any other node (Has one parent and one child node).
2. The node has the following **parameters**: 1. **Name** 2. **Sentence** 3. **Sprite** (You can leave it **null**).
3. By clicking the "**Add external function**" button, another field appears for the name of the function that you previously added (More on this below).
4. Click the "**Remove external function**" button to **remove** external function 😊.

3 Answer Node

1. **Answer node** has an **infinite** parent node, but child nodes depend on **the number of answers** in the node (Answer node can't join to answer node).
2. By clicking the "**Add Answer**" button, you will add another answer option (The number of answer options is unlimited).
3. You can delete the **last answer** option by clicking the corresponding button.

How to Use and Technical Part

1. To use dialog system you can just drag and drop **Dialog Prefab** from the **Prefab folder** and call **StartsDialog** method from **DialogBehaviour** script that **attached** to this prefab.

💡 It is recommended that another script call this method instead of doing it in the DialogBehaviour script. For example, as in the demo script.

```
// Test script to call StartDialog method
public class TestDialogStarter : MonoBehaviour
{
    [SerializeField] private DialogBehaviour dialogBehaviour;
    [SerializeField] private DialogNodeGraph dialogGraph;

    private void Start()
    {
        dialogBehaviour.StartDialog(dialogGraph);
    }
}
```

StartDialog method from the **DialogBehaviour** script. As you can see, you need to pass **DialogNodeGraph** as a parameter.

```
public void StartDialog(DialogNodeGraph dialogNodeGraph)
{
    isDialogStarted = true;
```

```

        if (dialogNodeGraph.nodesList == null)
        {
            Debug.LogWarning("Dialog Graph's node list is empty");
            return;
        }

        onDialogStarted?.Invoke();

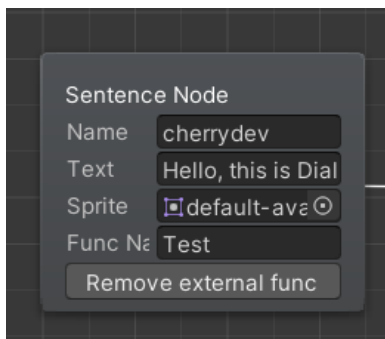
        currentNodeGraph = dialogNodeGraph;

        DefineFirstNode(dialogNodeGraph);
        CalculateMaxAmountOfAnswerButtons();
        HandleDialogGraphCurrentNode(currentNode);
    }

```

2. You can bind **external functions** to use them in **sentence node**. There is a method for this called **BindExternalFunction**. It takes as parameters the name of the function and the function itself. This method can then be used in a sentence node, it will be called along with this node.

💡 You need to bind an external function before calling it.



```

public void BindExternalFunction(string funcName, Action function)
{
    -----
}

```

```

public class TestDialogStarter : MonoBehaviour
{
    [SerializeField] private DialogBehaviour dialogBehaviour;
    [SerializeField] private DialogNodeGraph dialogGraph;

    private void Start()
    {
        dialogBehaviour.BindExternalFunction("Test", DebugExternal);

        dialogBehaviour.StartDialog(dialogGraph);
    }

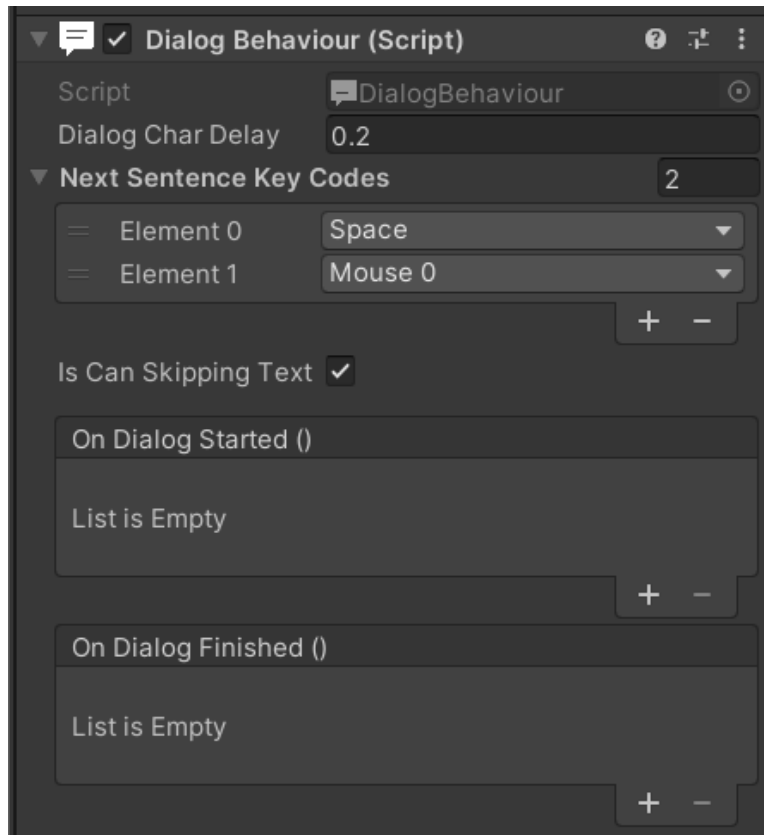
    private void DebugExternal()
    {
        Debug.Log("External function works!");
    }
}

```

3. In the inspector you can configure **parameters** such as:

- **Dialog Char Delay** (**float**) - delay before printing characters or text printing speed
- **Next Sentence Key Codes** (**List<enum>**) - keys when pressed that load the next sentence
- **Is Can Skipping Text** (**bool**) - If true - when you press the keys, instantly print the current sentence
- **OnDialogStarted** and **OnDialogEnded** events.

💡 You can assign all these parameters in code



Feel free to edit any code to suit your needs. If you find any bugs or have any questions, you can write about it to me by email, github or in reviews in the Unity Asset Store. I will also be pleased if you visit my itch.io page. 😊

Gmail: olegmrole@gmail.com

Github: <https://github.com/OlegVishnivetsky>

Itch.io: <https://oleg-vishnivetsky.itch.io/>

This file will be updated over time. If you write suggestions again.