

Node based dialog system



Dialog Node Based System (Unity asset tool)

Last Updated: @07/09/2025

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.

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:

[Click here.](#)

Navigation Links

Use these links to quickly navigate to different sections of the documentation:

- [1 Node Editor](#)
- [2 Variables](#)
- [3 Sentence Node](#)
- [4 Answer Node](#)
- [5 Modify Variable Node](#)
- [6 Variable Condition Node](#)
- [!\[\]\(31b03e46ee8a80a1f1467b8c03bd76e8_img.jpg\) How to Use and Technical Part](#)
- [!\[\]\(7d9665ff04f9d2270c38081c6215a724_img.jpg\) Localization Integration](#)
- [!\[\]\(7cea648fec4dfc1e99934873e9173b69_img.jpg\) Timeline Integration](#)
- [!\[\]\(48ceb66414885cacc3f139b4fa359213_img.jpg\) Tool Bar Navigation](#)

1 Node Editor

STEP 1: How to open node editor window

1. Right-click to the **assets** folder.
2. Go to **Create/ScribableObjects/Node Graph/Node Graph**.
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

Mouse Controls

1. Left Mouse Button:

- On node: Select and move nodes
- On empty space + drag: Create selection rectangle to select multiple nodes
- On empty space + click: Deselect all nodes

2. Middle Mouse Button:

- On empty space + drag: Pan/move around the editor view
- On node + drag: Create connection between nodes

3. Right Mouse Button:

- Click: Open context menu with options like "Create Sentence Node", "Create Answer Node", "Select All Nodes", "Remove Selected Nodes", and "Remove Connections"

Variables

The variable system allows dialogs to dynamically react to game state, player choices, or external triggers by reading and modifying values during conversations.

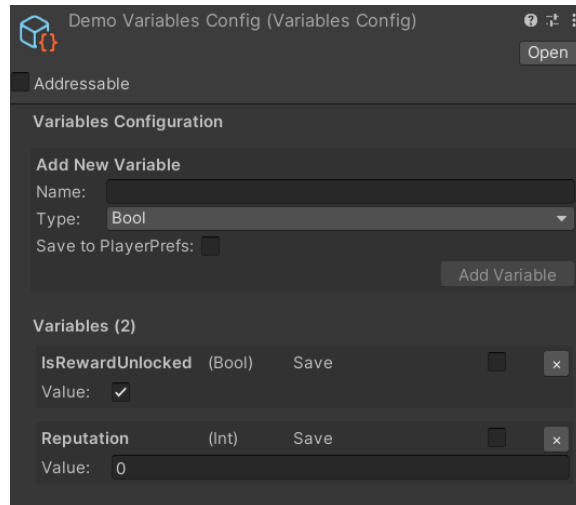
Supported Variable Types

- **Bool**
- **Int**
- **Float**
- String

Variables are stored in a **Variable Config** (ScriptableObject). You need to create it and attach it to the **DialogNodeGraph**.

Or you can create one of the variable nodes and this configuration will be added automatically.

A variable can be marked as "Save to Prefs". This means that it will be preserved when changed in PlayerPrefs.



💡 Usage in Sentence/Answer Nodes

Variables can be used inside dialog text via placeholders:

- Example: "You have {coinCount} coins!" So just write variable name inside { }.
- Handled by `DialogTextProcessor.ProcessText(text, handler)` Automatically replaced during dialog display

📡 Accessing/Getting Variables in Code

```
int coins = dialogBehaviour.GetVariableValue<int>("coinCount");
```

```
dialogBehaviour.SetVariableValue("coinCount", 15);
dialogBehaviour.SetVariableValue("isDoorOpen", true);
```

3 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 😊.

4 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.

5 Modify Variable Node

Used to **change** a variable's value.

1. Select the target variable from the drop-down menu.
2. Select target action:
 - For Boolean variables:
 - Set - Directly set the variable to true or false
 - Toggle - Switch between true and false (flip the current value)
 - For Int/Float variables:
 - Set - Directly set the variable to a specific value
 - Increase - Increase the variable by a specified amount
 - Decrease - Decrease the variable by a specified amount
3. Enter a value or set a bool status.

6 Variable Condition Node

Used to check variable values and branch dialog flow based on conditions.

1. Select the variable to check from the drop-down menu.
2. Set comparison type depending on variable type:

- For Boolean: ==, ≠
 - For Int/Float: ==, ≠, ⇒, ≤, >, <
3. Enter the value to compare against.
 4. Connect to different nodes for TRUE and FALSE outcomes.



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();

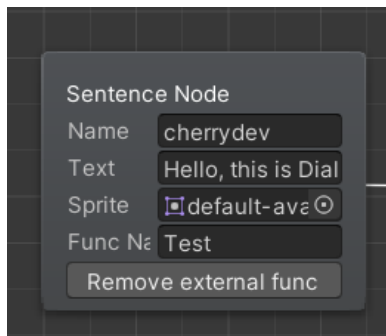
    currentNodeGraph = dialogNodeGraph;

    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);
}

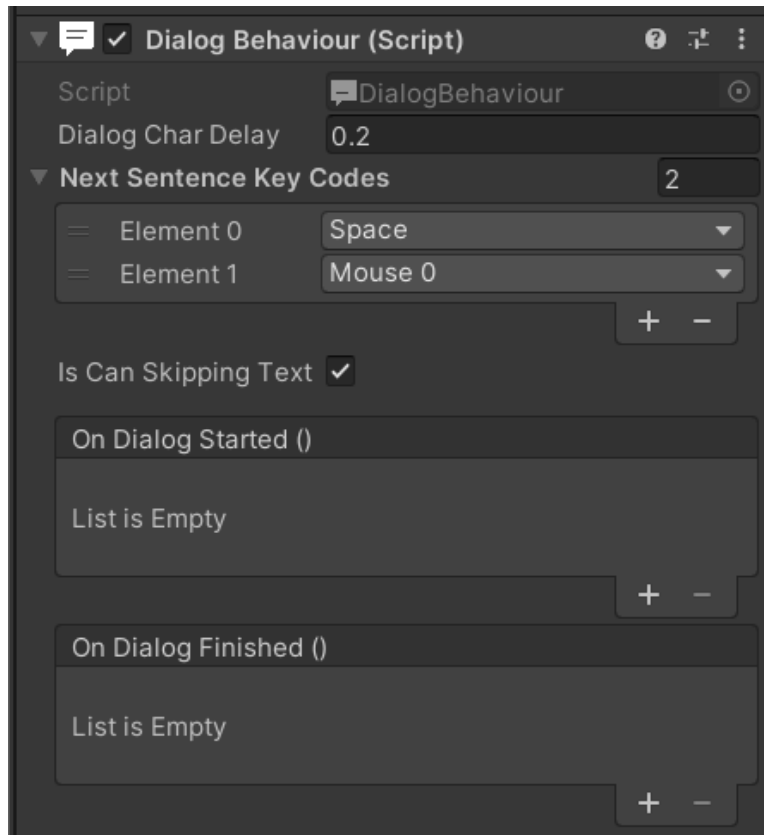
private void DebugExternal()
{
    Debug.Log("External function works!");
}
}

```

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



Localization Integration

This asset integrates with the Unity Localization system for easy multi-language support:

💡 The asset includes Unity Localization as a dependency. If you don't need localization, you can delete this package.

Setting Up Localization:

1. **First**, set up localization in **Player Settings**:
 - Create **Local Settings**
 - Set up your desired **Locales**
2. In the **Dialog Node Editor**, click **Localization** → **Set Up Localization**:
 - This creates a **Localization** folder in **Assets**

- All localization data for each graph will be stored here
- A **localization table collection** is automatically created with pre-configured entries

Edit Your Translations:

- Add text for other languages in the generated **table collection**
- The system works with **auto-generated keys**, but you can customize them

Managing Localization Keys:

- Click **Edit Table Keys** button to toggle key editing mode
- Auto-created keys are random and might not be readable
- You can **edit keys** to be more descriptive
- Click **Localization → Update Keys** to apply your custom keys

✅ Auto-generated keys work fine if you prefer not to customize them

Timeline Integration

Dialog Behaviour can now be used with Unity's Timeline system for cutscenes and scripted sequences:

Setting Up Dialog in Timeline:

1. **Create Dialog Behavior Track:** Add a new track in your Timeline and assign your Dialog Behavior component
2. **Add Sentence Clips:** Create sentence clips on the track and assign sentence scriptable objects
3. **Control Typing Speed:** The character typing speed is determined by the clip length - longer clips mean slower typing

External Functions in Timeline:

- Use **Call External Function Clip** to trigger bound functions at specific points in your timeline
- Make sure to **bind methods** before using them in timeline clips
- Always use the **exact method name** in the timeline clip that was used when binding



Tool Bar Navigation

The editor toolbar provides quick access to various functions to enhance your workflow:



Nodes Dropdown

Access a list of all nodes in your graph. Each node is prefixed with:

-  for **Sentence** nodes – shows the first part of the dialog text
-  for **Answer** nodes – shows the first answer option

Click on any node in the list to instantly **center** and **select** it in the graph.

Search Functionality

Quickly find specific dialog content:

- Type text into the **search field** to locate nodes containing that text
- Click the **Search** button to find and center on matching nodes
- As you type, matching nodes are **automatically highlighted**
- Click the **Clear (x)** button to reset your search

Find My Nodes

Automatically centers the view on all nodes in your graph.

Helpful when nodes are **scattered** across the canvas.

Localization Tools

Integrates with Unity's **Localization** package:

- **Edit Table Keys** – Toggle editing mode to customize localization keys
 - **Localization** – Access all localization options from the dropdown
-



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: olegmroleg@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.