💬

Dialog Node Based System (Unity asset tool)

Last Updated: @04/23/2025 **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.

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You can check video tutorial on youtube, but this **video is little outdated**: Click here.

1️⃣ Node Editor

STEP 1: How to open node editor window

1. Right-click to the **assets folder**.
2. Go to ***Create/ScribtableObjects/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

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"

2️⃣ 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 🙂.

3️⃣ 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.

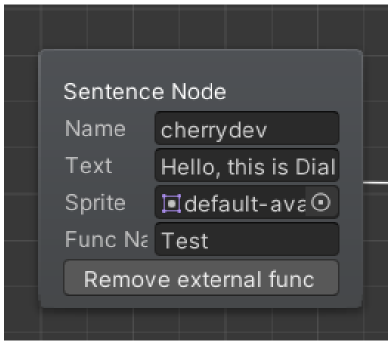
```
// Test script to call StartDialog method
public class TestDialogStarter : MonoBehaviour
{
    [SerializeField] private DialogBehaviour dialogBehaviour;
    [SerializeField] private DialogNodeGraph dialogGraph;

    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.


Sentence Node
Name cherrydev
Text Hello, this is Dial
Sprite default-ava
Func Na Test

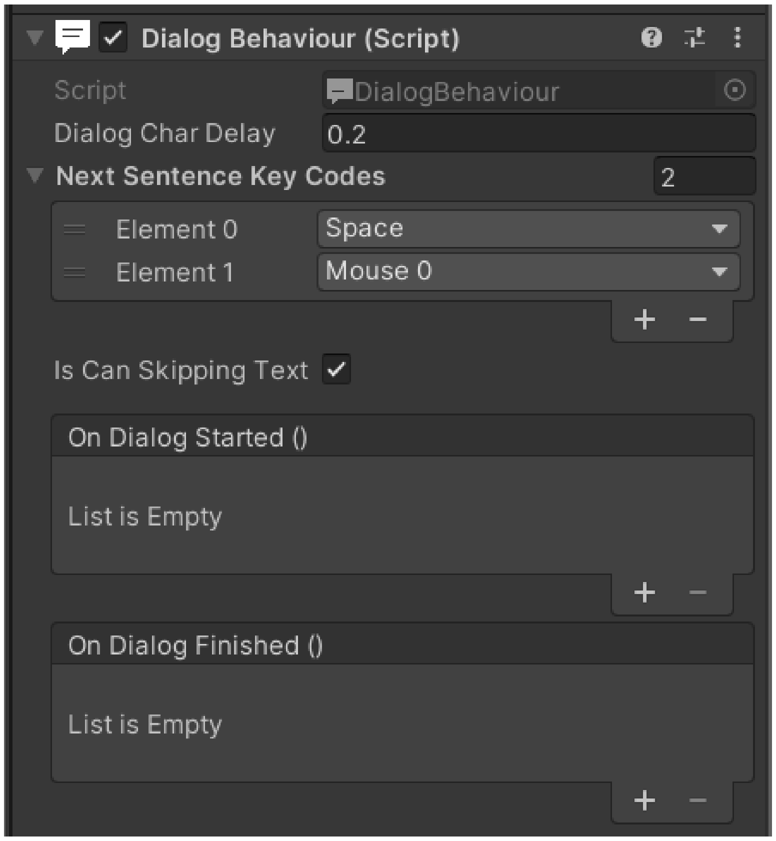
```
public void BindExternalFunction(string funcName, Action function);

--------------------------------------------------------------------------------

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 Dalay** (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



Dialog Behaviour (Script)

Script	DialogBehaviour
Dialog Char Delay	0.2
Next Sentence Key Codes	2
Element 0	Space
Element 1	Mouse 0
Is Can Skipping Text	✓

On Dialog Started ()

List is Empty

On Dialog Finished ()

List is Empty

🌐 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

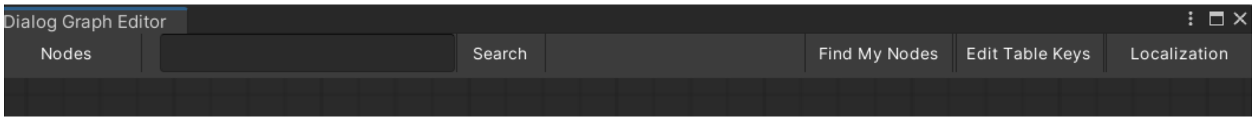
Managing Localization Keys:

- 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

🧭 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:

- `S:` for **Sentence** nodes – shows the first part of the dialog text
- `A:` 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 (×)** 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 itchio 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.