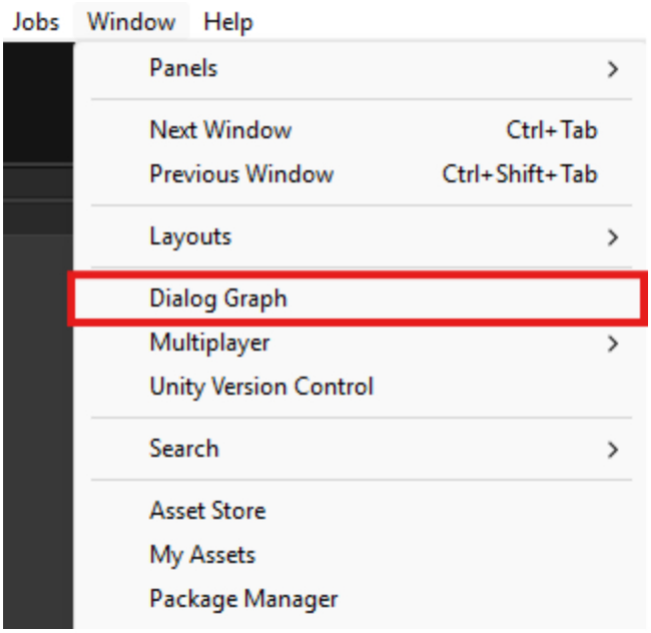
Dialog Graph Engine Guide

This guide provides a detailed and clear explanation of working with the **Dialog Graph Engine** in Unity to create and manage dialog graphs in your projects.

Getting Started

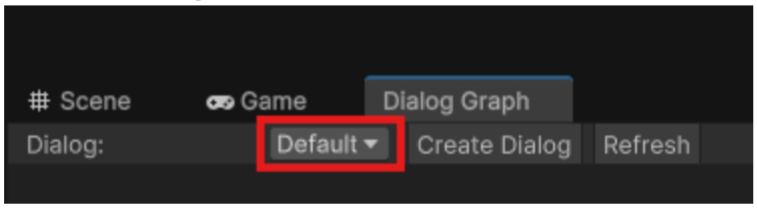
To start using Dialog Graph Engine, follow these steps after importing the `.unitypackage`:

1. Open the **Dialog Graph Editor**:
    - In the Unity Editor, navigate to the menu **Window > Dialog Graph**.

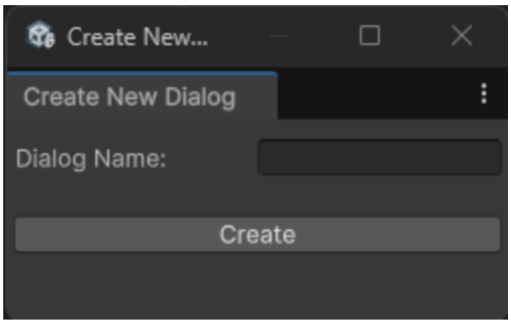


Creating and Selecting Dialogs

- **Default Dialog**: Upon opening the dialog editor, a dialog named **"Default"** is automatically created.



- **Selecting Dialogs**: Use the dropdown list in the editor to switch between created dialogs.
- **Creating a New Dialog**:
  1. Click the **Create Dialog** button.
  2. Enter a unique name for the dialog.



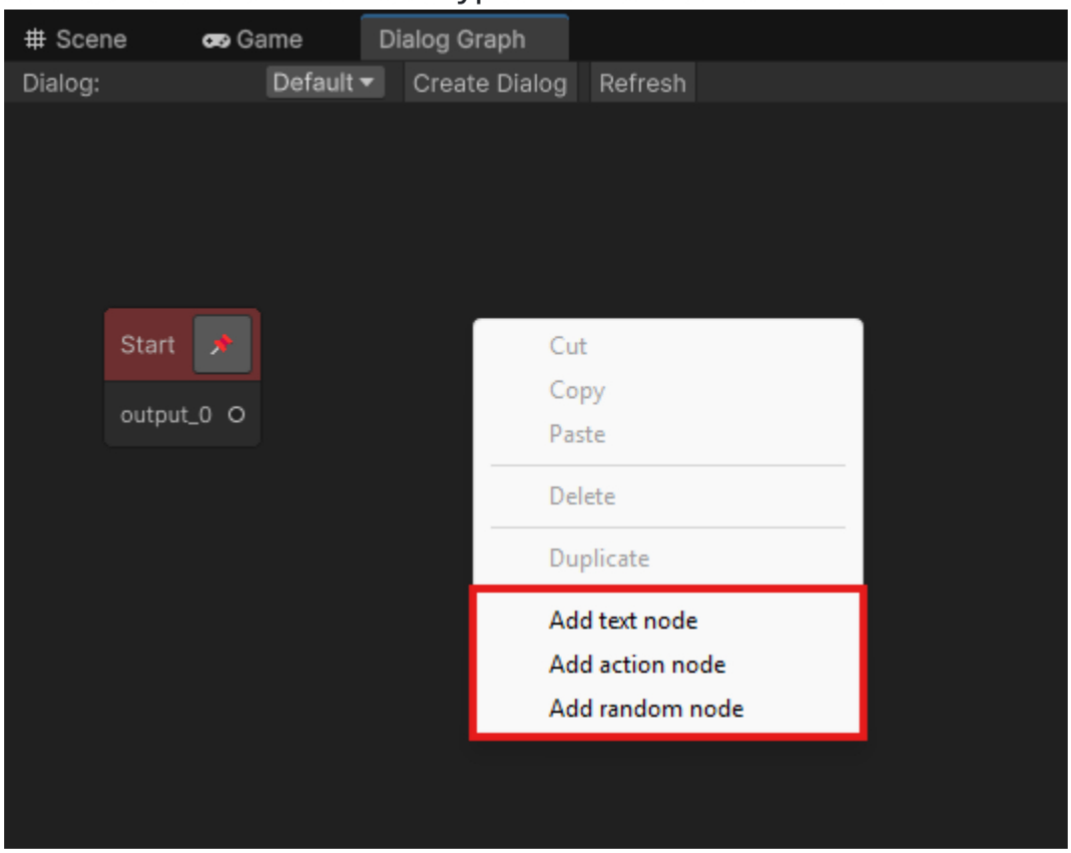
  3. The editor will automatically switch to the new dialog.

- **Storage**: All dialogs are saved as node sets in the `Resources/DGEDialogs` directory, with each dialog in a separate folder.

Creating Nodes

To add nodes to the dialog graph:

1. Right-click in an empty area of the graph.
2. Select one of the three node types from the context menu:



- **Text Node**
- **Action Node**
- **Random Node**

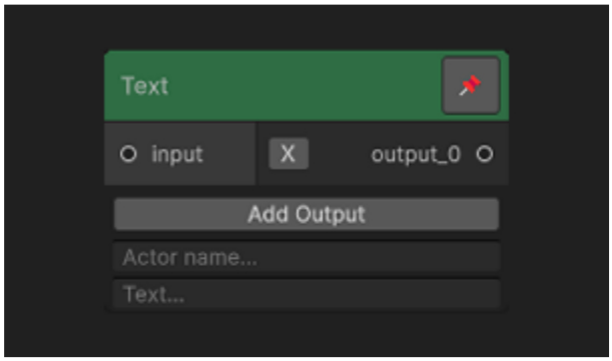
To view the `.asset` data file of a specific node in the project hierarchy, use the **Go to Data** button in the node's header on the graph.

Node Types and Their Features

Dialog Graph Engine supports three node types, each with unique properties and connection capabilities:

1. Text Node

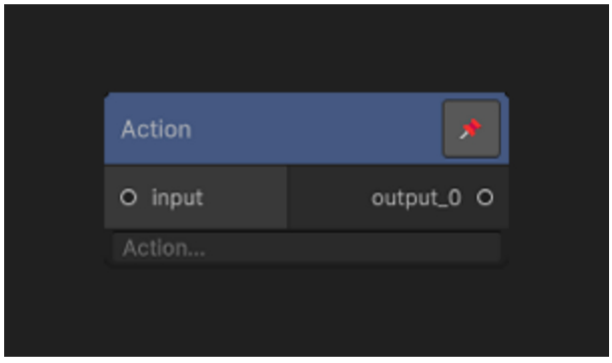
- **Purpose**: Contains a dialog line or phrase.



- **Fields**:
  - **Source**: The name or identifier of the speaker.
  - **Text**: The dialog text itself.
- **Input Port** (Many-to-One):
  - **Allowed Connections**: Action Node, Random Node.
- **Output Ports** (One-to-One):
  - **Allowed Connections**: Action Node.
  - Use the **Add Output** button to create output ports.

2. Action Node

- **Purpose**: Represents response options (e.g., buttons) for a phrase from a text node.

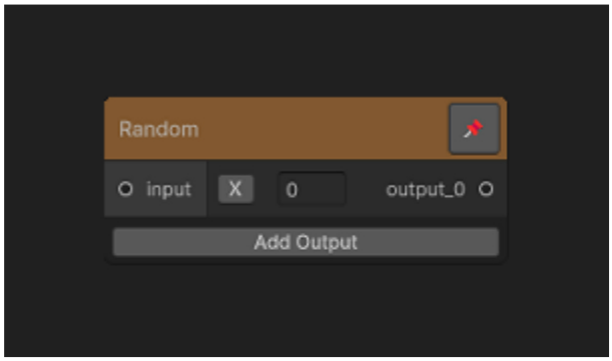


- **Input Port** (One-to-One):
  - **Allowed Connections**: Text Node.

- **Output Port** (One-to-One):
  - **Allowed Connections**: Text Node, Random Node.

3. Random Node

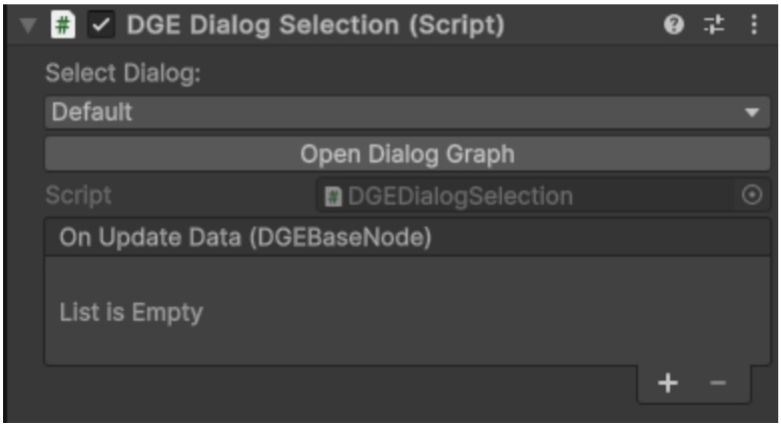
- **Purpose**: Randomly selects one of the connected output nodes with a specified probability (0–100%). Probabilities are normalized (e.g., two outputs at 50% are equivalent to two outputs at 100%).



- **Input Port** (One-to-One):
  - **Allowed Connections**: Action Node.
- **Output Ports** (One-to-One):
  - **Allowed Connections**: Text Node.

Using Dialogs in a Scene

The **DGEDialogSelection** script manages dialogs in a Unity scene.



Setup

1. **Adding the Script**:
    - Attach the `DGEDialogSelection` component to an object in the scene.

2. **Subscribing to Events**:

- Use the `onUpdateData` event to receive dialog node updates:

```
public UnityEvent<DGEBaseNode> onUpdateData;
```

- Use the `onDialogEnd` event to handle dialog completion:

```
public UnityEvent onDialogEnd;
```

- The `onUpdateData` event passes nodes of type `DGEBaseNode`, starting from the node following `StartNode`.

```
public DGEDialogSelection dialogSelection;

// Subscribe to dialog updates
dialogSelection.onUpdateData.AddListener(UpdateDialog);
// Subscribe to dialog completion
dialogSelection.onDialogEnd.AddListener(DialogEnd);

private void UpdateDialog(DGEBaseNode node)
{
    // Iterate through connected nodes (dialog response button
    foreach (var actionNode in node.outputs)
    {
        // Use the SelectAction() method to proceed to the nex
        dialogSelection.SelectAction(actionNode);
    }
}

private void DialogEnd()
{
    // Your code after dialog completion
}
```

3. **Starting a Dialog**:

   - Call the `StartDialog()` method to begin the dialog process:

```
dialogSelection.StartDialog();
```

Customizing Nodes

You can extend nodes with additional data to meet your project's requirements.

Steps for Customization

1. **Accessing the Node Script**:
    - Use the **Go to Data** button in the node's header to locate its `.asset` file.

2. **Modifying the Script**:

   - Open the script for the desired node type (e.g., `DGETextNode`) and add custom fields. Example:

```
using UnityEngine;

[CreateAssetMenu(fileName = "TextNode", menuName = "Nodes/Text
public class DGETextNode : DGEBaseNode
{
    public enum ActorType
    {
        Actor_0,
        Actor_1,
        Actor_2,
        Actor_3,
        Actor_4,
        Actor_5
    }

    public ActorType actorType;
}
```

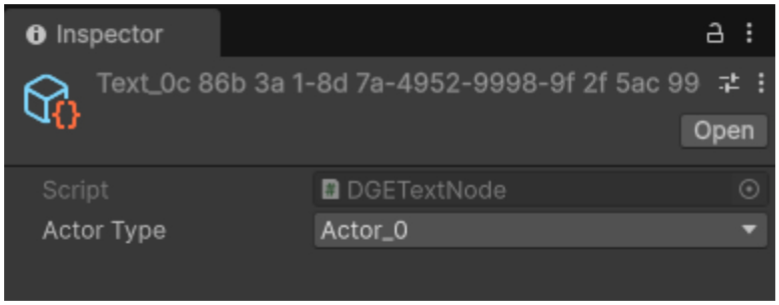
3. **Accessing Custom Data**:

   - In the `onUpdateData` event, cast the node to the desired type to access its data:

```
switch (node)
{
    case DGETextNode textNode:
        var actorType = textNode.actorType;
        break;
}
```

Notes

- Custom fields added to a node are displayed in the dialog editor for that node.



Conclusion

Dialog Graph Engine provides a flexible and intuitive system for creating complex dialog chains in Unity. By using **Text Nodes**, **Action Nodes**, and **Random Nodes**, you can create interactive dialogs with branching and random outcomes. The `DGEDialogSelection` script integrates the dialog system into your scenes, and node customization allows you to adapt the system to your project's specific needs.