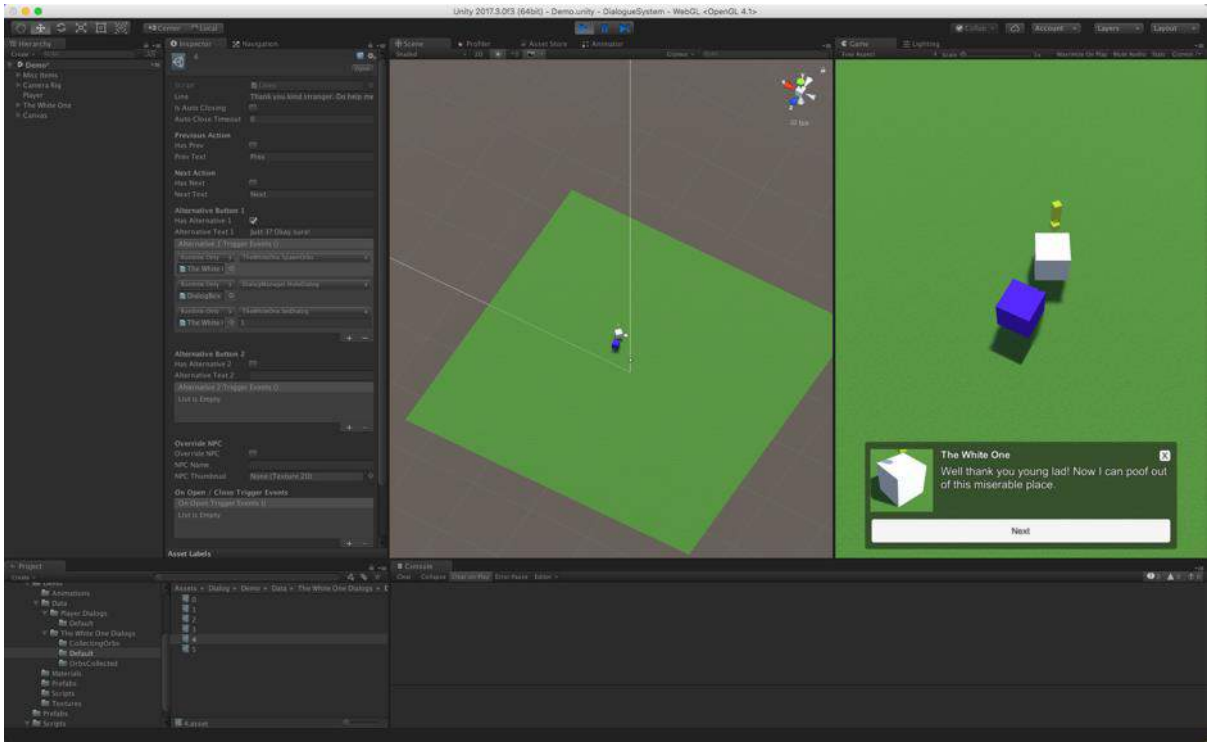
# Dialog Box

A simple to use dialog component for your Unity game.




Version 1.0.1

# Table of Contents

## Features

1) Customizable flow of dialog using drag and drop and UnityEvents.
2) Data is saved as Scriptable Objects[1]:
    a. Enabling editing data at runtime.
    b. Easy GUI-based configuration.
3) Custom callbacks to call methods inside other files.
4) Customizable GUI component using Unity GUI.
5) Auto closes the dialog box when the invoker (Player) moves away.
6) Auto closes the dialog box on time out.
7) Easily extendable to add your own animation.

## Demos

Demos can be viewed here: https://testdialog-b4340.firebaseapp.com/

### Simple Demo

Run the "`Dialog/Demo/Scenes/1. Simple`" scene in Unity Editor to view the full functionality of this package.

This demo shows a simple Show / Hide without any animations.


*Figure 1: Running simple example of the package*

The demo has two example `NPCBehaviour`. The "The White One" `GameObject` and the "Player" `GameObject` itself.

[1] Using `ScriptableObjects` does comes with it quirks. Refer the Quirks Section.

### Animated Demo

Run the "`Dialog/Demo/Scenes/2. Animated`" scene in Unity Editor to view the full functionality of this package.

This demo shows a simple Show / Hide with fade in / out animations.


*Figure 2: Running animated example of the package*

## Quick Start

1) Drag the `Dialog/Prefabs/SimpleDialogBox` prefab into your canvas.
2) Create your Lines.
3) Create your Dialog.
4) Create a class extending the `Dialog.NPCBehaviour` and attach it to your NPC `GameObject`.
5) Attach the Dialog you created in #3 to the class in the inspector.

## Introduction

This package is consists of 4 main classes:

1) `Dialog.BaseDialogManager`
2) `Dialog.BaseNPCBehaviour`
3) `Dialog.ScriptableObjects.Dialogs`
4) `Dialog.ScriptableObjects.Lines`

The `Dialog.BaseDialogManager` class is the main script that manages the state of the Dialog UI. It comes prepackaged inside `SimpleDialogBox` prefab and you shouldn't have to attach it to any `GameObject` manually.

`Dialog.BaseNPCBehaviour` extends the `UnityEngine.MonoBehaviour`. Extend this class and attach it to any `GameObject` that needs to display a Dialog. Note that the base class already calls `Awake` and `Update` methods so should you need to override these methods just call `base.Awake()` / `base.Update()` inside of the overriding method.

`Dialog.ScriptableObjects.Dialogs` extends the `UnityEngine.ScriptableObject` class. The Unity editor uses this file to create instances of Dialogs.

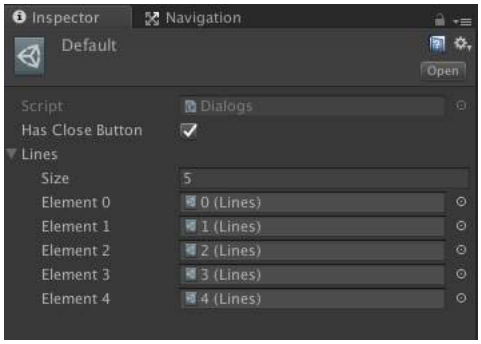



*Figure 3: The Dialogs ScriptableObject instance*

`Dialog.ScriptableObjects.Lines` extends the `UnityEngine.ScriptableObject` class. The Unity editor uses this file to create instances of Lines.

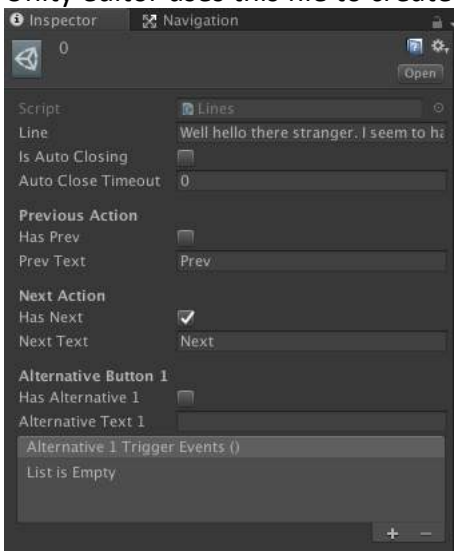



*Figure 4: The Lines ScriptableObject instance*

An `NPCBehaviour` can have multiple dialogs attached to it and a dialog can have multiple lines attached to it.

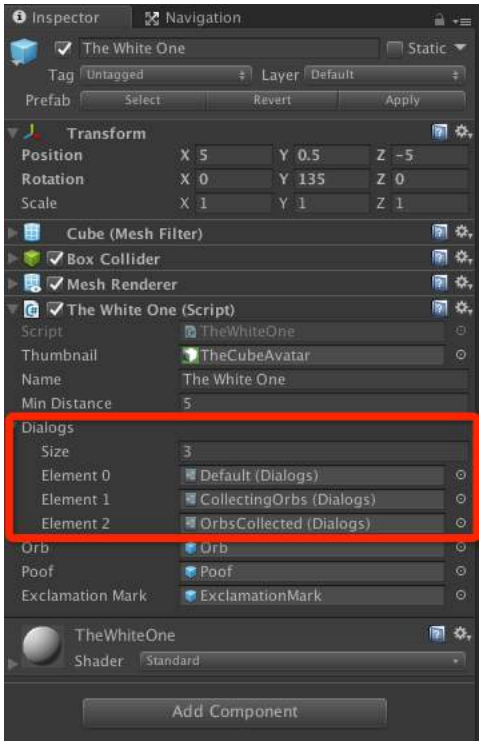



*Figure 5: An NPC can have more than one Dialog attached to it*

Only one dialog and one line can be active at any one time.

# Basic Usage

## Lines

Lines are the Dialog UI configuration when it is shown. It is attached to the Dialog.

### Lines Creation

1. In the Project View, navigate to a folder where you want to store the data (can be any folder inside the Assets folder).
2. Right click -> Create -> Dialog: Create Line


*Figure 6: Create Line*

## Lines Configuration

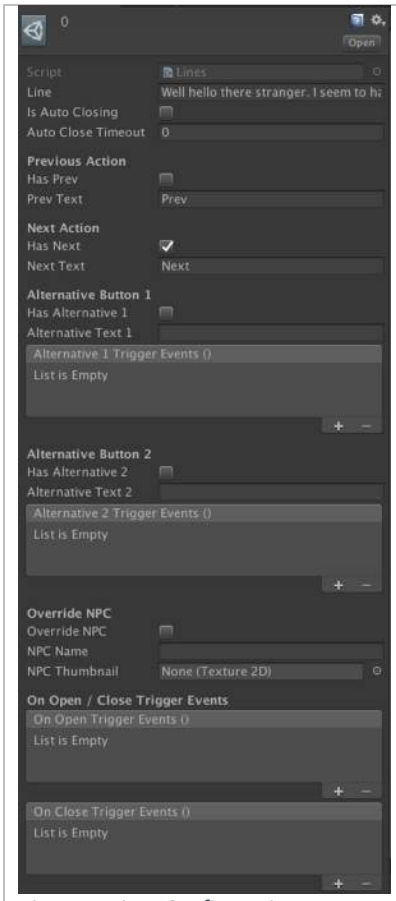



*Figure 7: Lines Configuration*

| Property | Explanation |
|---|---|
| Line | The text that is shown in the dialog. |
| Is Auto Closing | If this is checked then the dialog will be closed automatically on timeout. |
| Auto Close Timeout | Number of seconds the dialog will wait before auto closing. |
| Has Prev | If this is checked the “Previous” button will be shown. |
| Prev Text | The text to be shown inside the “Previous“ button. |
| Has Next | If this is checked the “Next” button will be shown. |
| Next Text | The text to be shown inside the “Next“ button. |
| Has Alternative 1 | If this is checked an alternative button will be shown. |
| Alternative Text 1 | The text to be shown inside the “Alternative 1“ button. |
| Alternative 1 Trigger Events | Events to be triggered on click of the “Alternative 1” button[2]. |
| Has Alternative 2 | If this is checked a second alternative button will be shown. |
| Alternative Text 2 | The text to be shown inside the “Alternative 2“ button. |
| Alternative 2 Trigger Events | Events to be triggered on click of the “Alternative 2” button. |
| Override NPC | If this is checked the Name and Thumbnail will be overridden in the Dialog. |
| NPC Name | The Name used to Override. |
| NPC Thumbnail | The Thumbnail used to Override. |
| On Open Trigger Events | Events to be triggered when the line is shown. |
| On Close Trigger Events | Events to be triggered when the line is closed. |

## Dialog

Dialogs are basically to a list of Lines that the NPCBehaviour will show.

[2] To call custom MonoBehaviour methods, make sure it’s attached to a Prefab and reference the Prefab in this field. Refer the Quirks section.

## Dialog Creation

1. In the Project View, navigate to a folder where you want to store the data (can be any folder inside the Assets folder).
2. Right click -> Create -> Dialog: Create Dialog

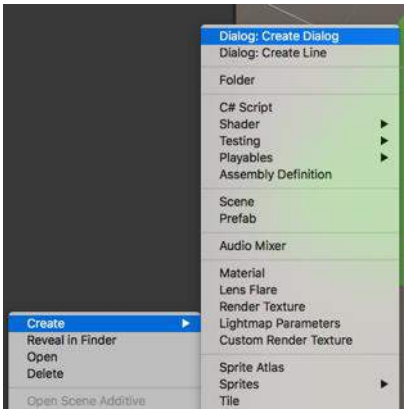



*Figure 8: Create Dialog*

## Dialog Configuration

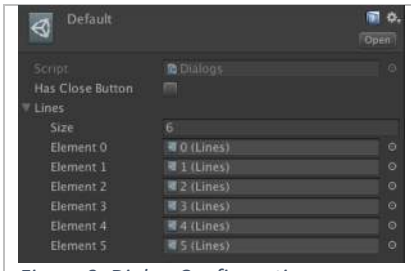



*Figure 9: Dialog Configuration*

| Property | Explanation |
|---|---|
| Has Close Button | If this is checked a small close button will be shown on the top of the dialog. |
| Lines Size | Number of Lines to attach to this dialog. |
| Lines Element | The attached Line element. |

## BaseNPCBehaviour

The `BaseNPCBehaviour` script is extended and attached to a `GameObject` that shows dialogs (usually NPCs).

## NPCBehaviour Configuration

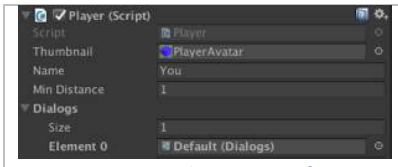



*Figure 10: NPCBehaviour Configuration*

| Property | Explanation |
|---|---|
| Thumbnail | The Thumbnail shown in the dialog. |
| Name | The Name shown in the dialog. |
| Min Distance | The minimum distance to close the dialog once the invoker has moved away |

| | | |
|---|---|---|
| | | from the attached script transform. |
| | Dialogs Size | The number of Dialog elements attached. |
| | Dialogs Element | The Dialog element attached to the script. |

## Scripting API Overview

### Abstract Dialog.BaseNPCBehaviour Class

```
void Dialog.BaseNPCBehaviour.SetDialog(int index)
```

Set the current dialog to the index.

```
void Dialog.BaseNPCBehaviour.OpenDialog(
    UnityEngine.Transform invokerTransform,
    int lineIndex,
    Dialogs dialog
)
```

Set the current dialog to “dialog”, sets the current line to “lineIndex” and watches the invoker position to auto close when it moves away. Opens the dialog afterwards.

```
void Dialog.BaseNPCBehaviour.OpenDialog(
    UnityEngine.Transform invokerTransform,
    int lineIndex
)
```

Sets the current line to “lineIndex” and watches invokerTransform position to auto close when it moves away. Opens the current dialog afterwards.

```
void Dialog.BaseNPCBehaviour.OpenDialog(
    UnityEngine.Transform invokerTransform
)
```

Watches invokerTransform position to auto close when it moves away and sets the currentLine to 0. Opens the current dialog afterwards.

```
void Dialog.BaseNPCBehaviour.ShowLine(int index)
```

Shows the line based on the `index`.

```
void Dialog.BaseNPCBehaviour.CloseDialog()
```

Closes the active dialog.

### Abstract Dialog.DialogManager Class

```
void Dialog.BaseDialogManager.OpenDialog(
    NPCBehaviour NPC,
    Dialogs dialog,
    int lineIndex
)
```

Sets the current NPC to “NPC”, sets the current dialog to “dialog” sets the current line index to “lineIndex” then calls `Dialog.BaseDialogManager.ShowLine()`.

```
void Dialog.BaseDialogManager.ShowLine()
```

Calls `Dialog.BaseDialogManager.ShowLine(currentLineIndex)` to show the line.

```
void Dialog.BaseDialogManager.ShowLine(int index)
```

Sets the current index to “index” and preps the dialog box. Then it shows the current line in the dialog box. Calls any callbacks attached to the line’s “On Open Trigger Events”.

```
void Dialog.BaseDialogManager.HideDialog()
```

Closes the active dialog. Calls any callbacks attached to the line’s “On Close Trigger Events” `Dialog.BaseNPCBehaviour.CloseDialog` should be preferred when closing a dialog.

```
void Dialog.BaseDialogManager.ShowNextLine()
```

If there is a next line in the active dialog line list, it fires any callbacks defined in the “On Close Trigger Events” on the current line then shows the next via `Dialog.BaseDialogManager.ShowLine(CurrentLineIndex + 1)`.

```
void Dialog.BaseDialogManager.ShowPrevLine()
```

If there is a previous line in the active dialog line list, it fires any callbacks defined in the “On Close Trigger Events” on the current line then shows the previous line via `Dialog.BaseDialogManager.ShowLine(CurrentLineIndex - 1)`.

## Quirks

This package uses `System.Reflection` to access Unity’s private classes. Reflection is used because the behaviour of `UnityEvent` inside `ScriptableObjects`: in which it can only refer to methods that are attached to `Prefabs`.

If `UnityEvent.Invoke` is called directly (without reflection), the call will be targeted to the `Prefab` rather than the `Prefab instance` in the Scene (as one would expect).

The way the `UnityEvent` is invoked is by searching the scene for `GameObject` matching the `Prefab` "`name`" or `Prefab` "`Name(Clone)`" then calling the `SendMessage` method on it and passing any variables that one might have set inside the inspector. Though `SendMessage` is slower compared to other methods of executing, it does come with the benefit of ease-of-use and extra flexibility for the user. Because these methods are only called on show / hide / click custom button on the dialog, performance impact should be negligible. Performance might suffer if you have lots of `GameObjects` in the scene (like thousands).

It's a tradeoff decision between usability vs best practice that I had to make. I chose to implement the more user-friendly way for users to attach UnityEvents to the dialog. Furthermore, chances are, the user is already familiar with the UnityEvent UI and accustomed to the way it works.

## Contact Information

You can contact me at zulfajuniadi@gmail.com to report bugs or support queries.