Tutorial Designer Documentation

Written by Michael Bukmaier, Rebound Games

Version 1.3, 03/2018

Table of Contents

1 Quickstart

Jump in at this chapter if you would like to get right to the point. Import the “TutorialDesigner” folder with all its contents into your Assets folder. It should be in the top level of your project:

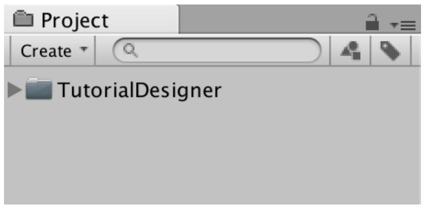


Illustration 1: Projectfolder

That’s it! Now you can open the Editor window by clicking
“Window” → “Tutorial Designer” → “Tutorial Editor”

1.1 Demos

You’ll find a little example Game under TutorialDesigner/Demo-Game/Scenes/Tutorial.unity. It covers all the basic features. Another short showcase comes with version 1.3, including all new added features: Tutorial-Designer/Demo2/Demo.unity. Please look in the “**Demo2 - FIRST README.txt**” file for Instructions to extract this scene.

2 Get started

Starting at this section, all details of the Asset will be explained in the right order. After you importet the Asset, like described in Illustration 1: Projectfolder, make sure you have the Editor window opened. You could just dock it into the top bar like this:

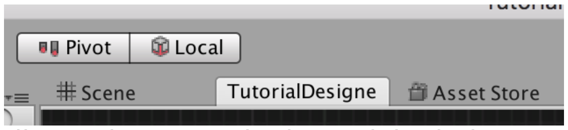
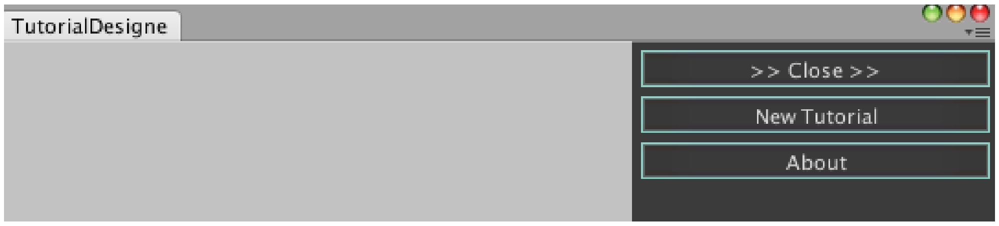


Illustration 2: Docked Tutorial Window

You can achieve this by drag & drop the top of the Editor window between the other windows above the Sceneview.

2.1 Editor window

After you opened the TutorialEditor, this is all you see.

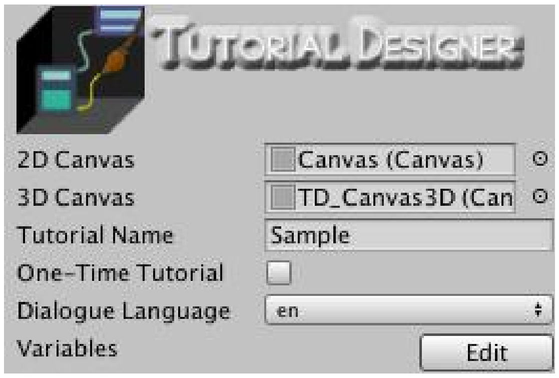


The dark area on the right is the Sidewindow. You can hide / open it by clicking the "Close" Button on the top.

If you click the "New Tutorial" Button, the Editor will become active. Also a Gameobject is created in the Hierarchy, called "TutorialSystem". This object must **NOT** be changed, renamed or removed! Otherwise the TutorialEditor loses its reference.

2.2 TutorialSystem Gameobject

After you created a new Tutorial, the TutorialSystem Gameobject tries to get a Canvas object located in the Hierarchy, as reference for Dialogue objects. If you don't have a Canvas yet, TD creates one. Same goes for an EventSystem GameObject, which is necessary for the Unity UI.



2D Canvas	Canvas (Canvas)
3D Canvas	TD_Canvas3D (Can
Tutorial Name	Sample
One-Time Tutorial	☐
Dialogue Language	en
Variables	Edit

Illustration 3: Tutorial System GameObject

"2D / 3D Canvas" - 2D Canvas on which the Dialogues will appear. 2D will be picked automatically if there exists one in the scene. If not, a new one will be created. In case you have 3D Dialogues, it will always create an own 3D Canvas under the TutorialSystem GameObject.

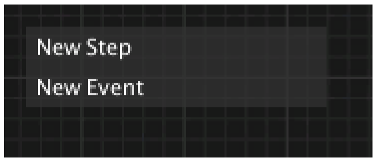
"Tutorial Name" - Used for storing PlayerPrefs, in case of One-Time Tutorial

"One-Time Tutorial"- Defines if this Tutorial should appear only once. Can be set for a particular scene or the whole project. See One-Time Tutorial

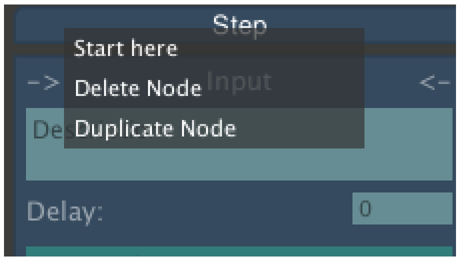
"Dialogue Language" is related to SmartLocalization. If you use multiple languages in your Tutorial, they can be switched here. Also during runtime. See SmartLocalization

"Variables" are defined to have dynamic parts on Dialogue Texts. See Variables

2.3 Working with Nodes in the Editor

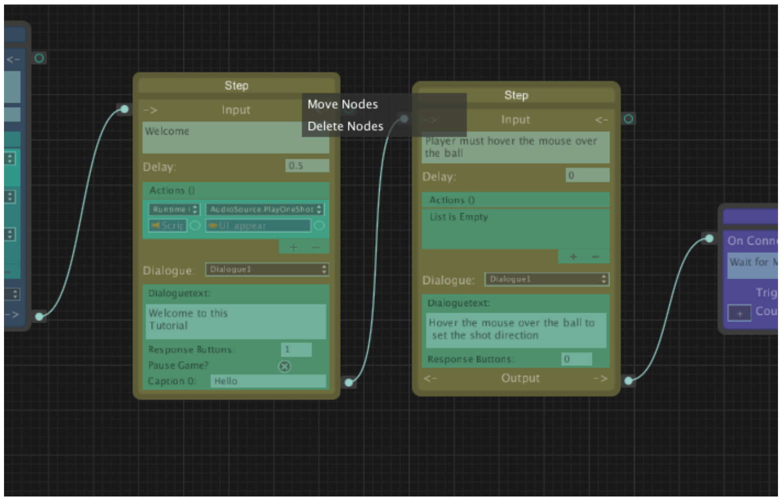


Right-click anywhere in the Editor. A Step Node or an Event Node can be created by this Contextmenu.



Right-clicking a Node opens a Contextmenu for deleting or duplicating a Node. “Start here” sets the clicked Node as Start Node for the Tutorial. After starting the Game, the Default Working Paths start from there.

You can also select multiple Nodes by holding down the “Shift”-key and clicking them. Releasing the “Shift” key and clicking a single Node, ends the Multi-Selection mode.



With multiple Nodes selected, there are two options: “Move Nodes” and “Delete Nodes”. Deletion can be undone.

If both Nodes are Step Nodes with active Dialogue, those Dialogues can be edited at once: Multi-Editing Dialogues

2.4 Sidewindow

After you created a new Tutorial, the Sidewindow looks like this:

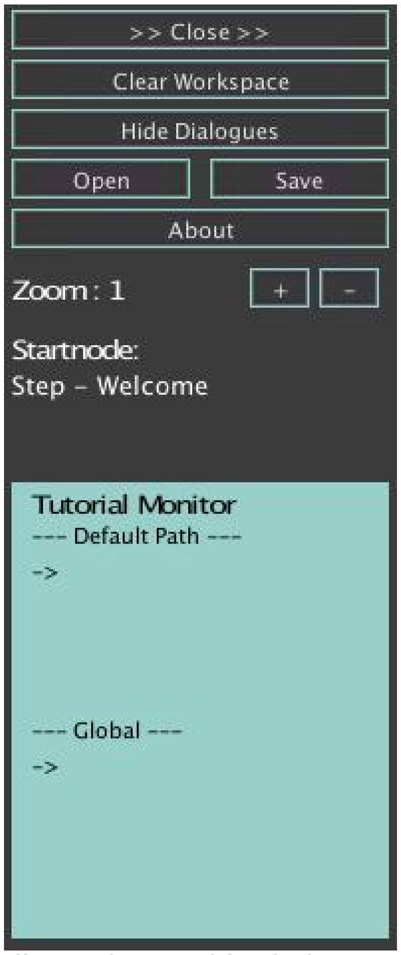


Illustration 4: Sidewindow

"*Close*" → Closes / Opens the Sidewindow

"*Clear Workspace*" → Deletes everything in the Editor Window. Don't worry, this action can be undone

"*Hide Dialogues*" → Hides all Dialogue Gameobjects in the Scene

"Open / *Save*" - Current Nodes in the scene can be saved to an asset file, and imported additionally to the existing Tutorial.

"About" - Show About Window for info and support

"*Zoom*" → Zooms the Editor window. Goes from factor 0.2 to 1.0

"*Startnode*" → Displays the Node from where the Tutorial is started after starting the Game

"*Tutorial Monitor*" → Active when the Game is running. Shows the current active Nodes on default- and Global Paths. See Working Paths

3 Node System

The Editor window opens the door to a graphical workflow design on the basis of your scripts. This is achieved by using Nodes for different purposes. By connecting them, you're able to define a processing order of your Tutorial segments.

In a connected group of Nodes, only one can be active at a time. If a Node is done, the next one gets active. Nodes can have multiple Inputs, but only one Output! Because multiple Nodes can lead to one Node, but every Node has a certain way how it continues, depenging on its settings.

This simple picture shows the basic technique behind the system. If a Node gets active, it processes its jobs and when finished, passes to the next Node in line. It's up to you making those connections. That group of connected Nodes is called **Working Path**. Only one Node can be active at a time in such a Working Path.

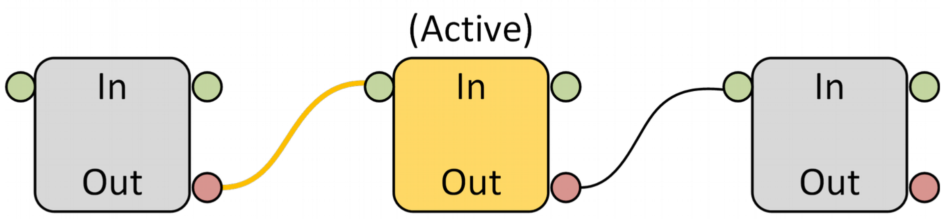


Illustration 5: Simple Node group (Working Path)

There can be more than one Nodes connected at the **Inputs**, but only **one at an Output**! Instead Nodes can allow multiple Output Connectors, on the basis of its purpose.

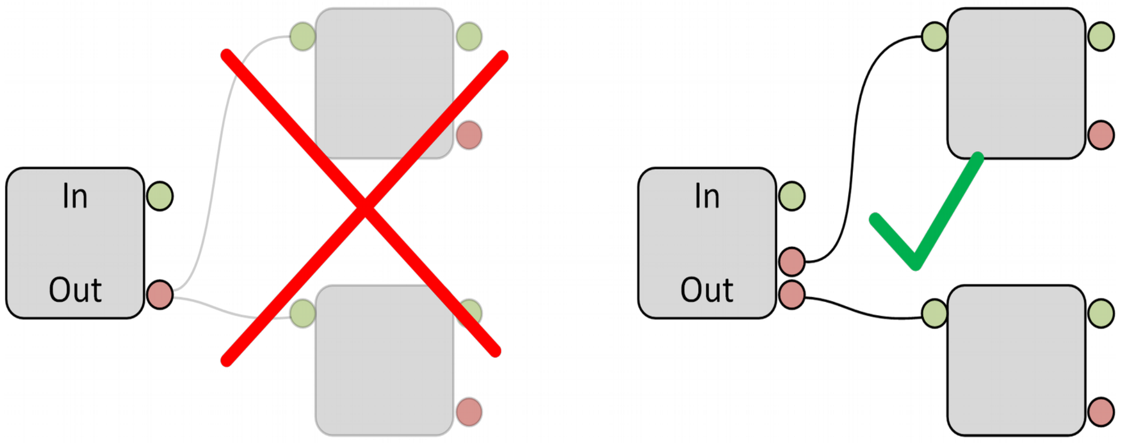


Illustration 6: Input- / Output Connections

3.1 Step Node

Step Nodes are intended to invoke custom script actions, display Dialogues and give the player up to three choices on how to continue.

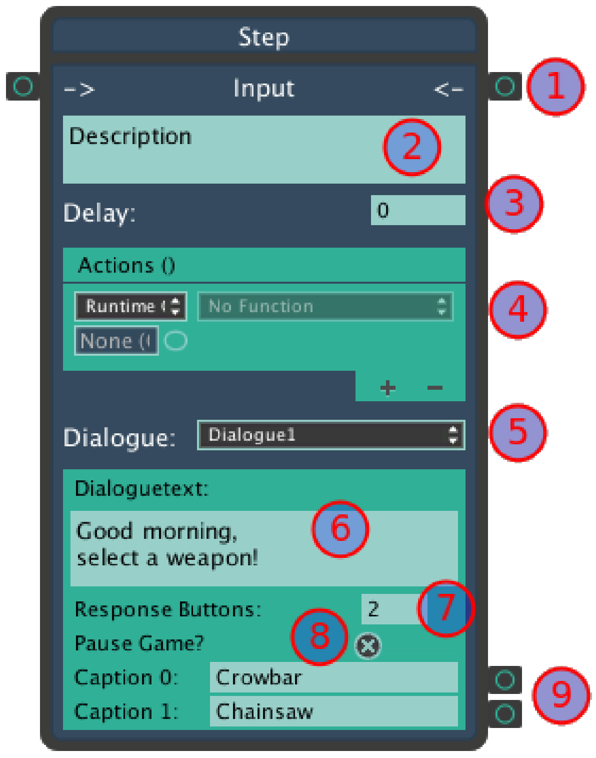


Illustration 7: Step Node

1) Input Connectors. One or more Nodes can be connected to them.

2) Description can be set by User. Won't affect the Game

3) Seconds before this Node starts working after getting active

4) Custom script actions, this module is based on UnityEvent. **+** = Add action, **-** = Remove action

5) This Popup menu chooses Dialogue templates (details at Included templates)

6) Dialogue Text, displayed in the Game

7) Max 3 Buttons can be added to the Dialogue, for chosing an ongoing way of the Tutorial

8) Sets the Game timer to 0 while the Dialogue is displayed, if the box is checked. This simulates a Game pause

9) If Response Buttons exist, every one has a specific Output. If there are no Buttons, Step Node has a default Output Connector

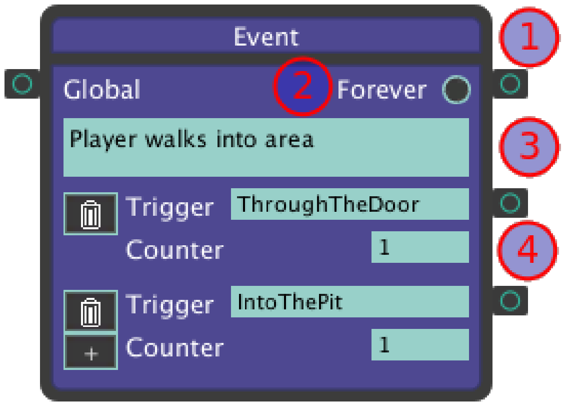
For keeping the display tied up if there are a few Nodes and connected curves, you can decide on which Input Connector you connect another Node, left or right, doesn't matter.
The Output Connectors however, swap sides automatically between left and right, trying to keep the view clear.

3.2 Event Node

Event Nodes listen for one or more Event Triggers that you're free to define. You can also set counters for them, in case one Trigger should be called more than once until the Event completes.

There are two types of Event Nodes: Node dependent and Global. Node dependents are only active after the previous connected Node passes over. Global Nodes become active as the Game starts.

For triggering Events, read the chapter: Eventmanager.



Player walks into area

Trigger	ThroughTheDoor
Counter	1
Trigger	IntoThePit
Counter	1

Illustration 8: Event Node

1) Input Connectors. Same as Step Node

2) Forever: Only available on Global Events. If checked, this Event Node stays active forever. After it completes, the values are reseted

3) Description of this Node

4) Event Triggers. One or more allowed. Every one has its own Output. A counter defines how often the Event must be triggered, in order to complete. Create new Trigger with **+**, delete one with the trashcan icon.

Same as at Step Nodes, Input Connectors are fix, Outputs swap left and right, depending on their connections.

4 Working Paths

Global Nodes are active since Game start, so is the Startnode, shown by Sidewindow. Therefore it is possible that multiple Nodes are active simultaneously. But on different Paths! A working Path is a group of Nodes that are connected together:

Example: The defined Start-Node is always on the first Working Path. Every other Global Node that runs simultaneously, begins a new Path. These Paths run independent from each other.

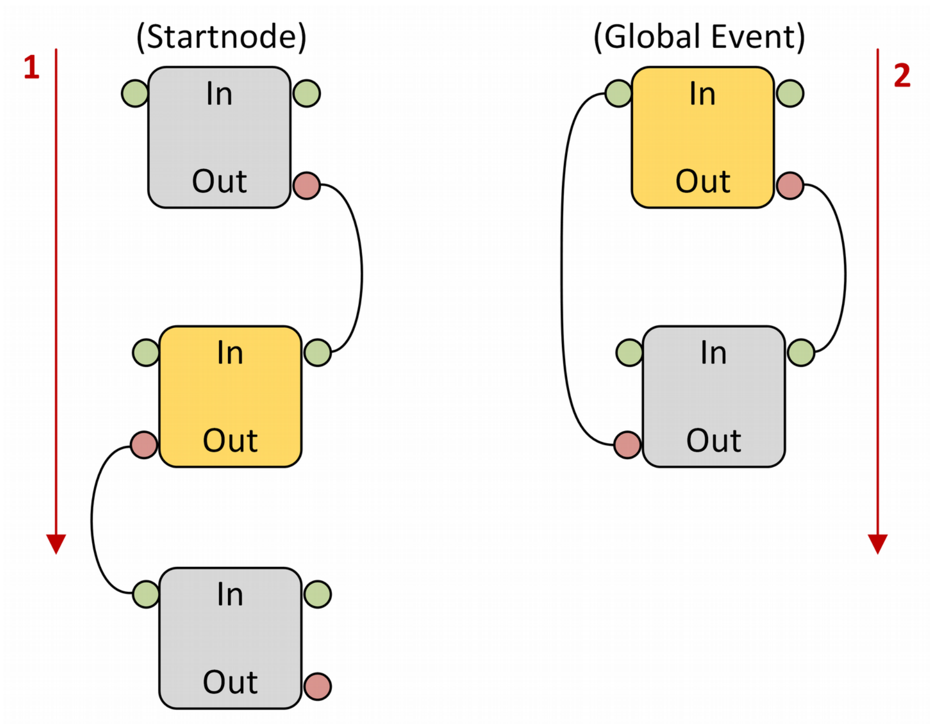


Illustration 9: Multiple Working Paths

In this example two Paths are working at the same time because Global Events don’t have to wait for other Nodes to become active. They start working when the Game begins.

5 Dialogues

On a Step Node, a Dialogue can be created. It will be visible du until:

1.) a Response Button in the Dialogue is clicked

2.) an Event Node completes and activates another Step Node with Dialogue

There cannot be 2 Dialogues on the same Working Path visible at the same time.

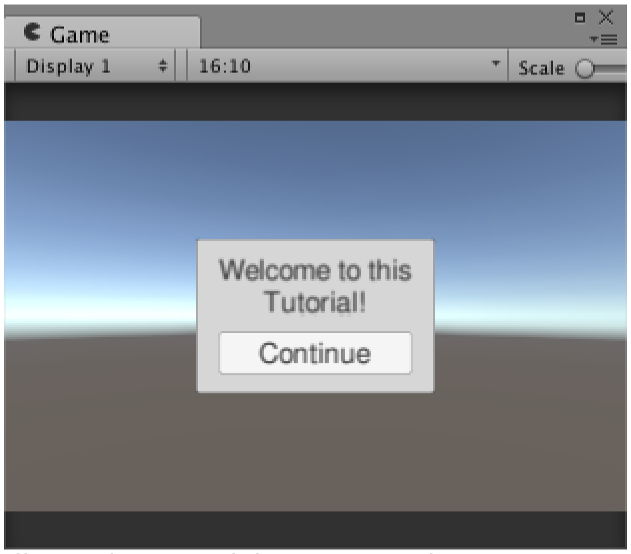
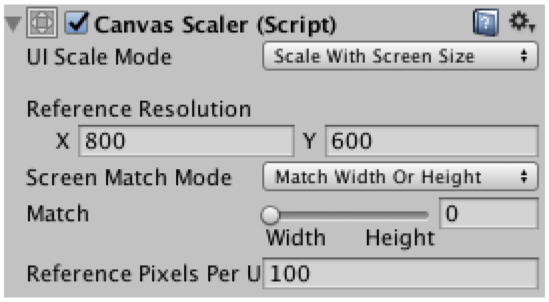


Illustration 10: Dialogue Example



Canvas Scaler (Script)
UI Scale Mode: Scale With Screen Size
Reference Resolution
X 800 Y 600
Screen Match Mode: Match Width Or Height
Match: 0 (Width – Height)
Reference Pixels Per U: 100

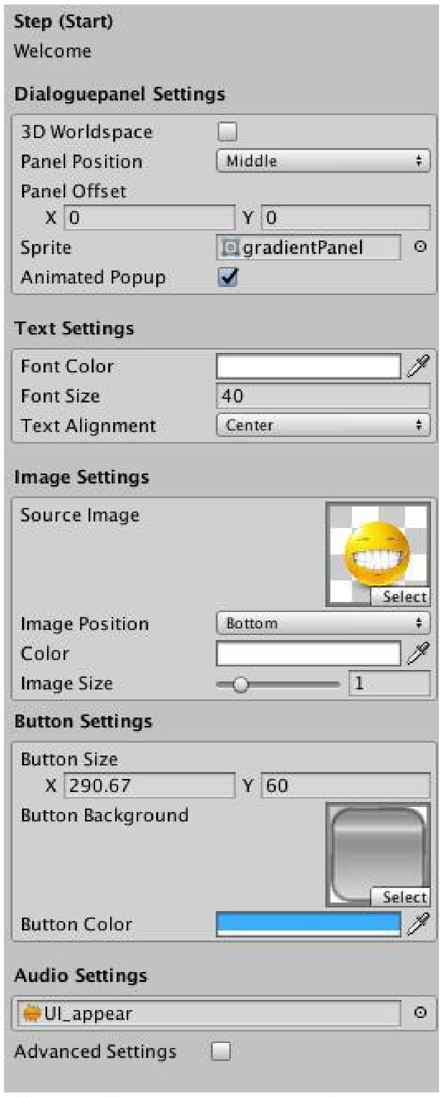
Illustration 11: Canvas Component

Preparations:

- Set "UI Scale Mode" of the Canvas Scaler Component (which you can find attached to Canvas), and adjust the resolution to your Game if necessary (Illustration 11: Canvas Component), as you would normally do for using the Unity UI. Further information: https://docs.unity3d.com/Manual/script-CanvasScaler.html
- For using custom Sprites as background for panel and Buttons, prepare their borders in the Unity Sprite Editor. Default settings for background sprites is "sliced": https://docs.unity3d.com/ScriptReference/UI.Image.Type.Sliced.html. This can be changed in Advanced Settings

5.1 Inspector

By clicking a Step Node with Dialogue in the Tutorial Editor, its parameters are shown in the Inspector. This way its appearance can can be changed, shown in realtime in the Scene- / Game Window. In reality the Dialogue's Dialogue Gameobject is adjusted by user entered values here. Many values are shown that are owned by the Gameobject's components, and directly forwarded to them.



Step (Start)
Welcome

Dialoguepanel Settings

3D Worldspace	☐
Panel Position	Middle
Panel Offset	X 0 Y 0
Sprite	gradientPanel
Animated Popup	✓

Text Settings

Font Color	
Font Size	40
Text Alignment	Center

Image Settings

Source Image	Select
Image Position	Bottom
Color	
Image Size	1

Button Settings

Button Size	X 290.67 Y 60
Button Background	Select
Button Color	🟦

Audio Settings

UI_appear

Advanced Settings ☐

Illustration 12: Step Node Inspector

Description: Step Node Description from Tutorial Editor

3D Worldspace: Toggles between 3D and 2D canvas, where this Dialogue is on.

Dialoguepanel Settings: This is the background panel of the Dialogue, containing everything.

Panel Position: Panel can be aligned to a border of the Game window: Top, Bottom, Left, Right, Middle

Panel Offset: Pixel Offset from its aligned Panel Position

Sprite: Background of the Panel.

Animated Popup: Dialogue will have a quick Popup-style appearance

Text Settings: Includes Dialogue Text and Button captions

Color / Size: ←

Aligment: Same parameters as in RectTransform

*Image Settings: P*icture included in the Dialogue

Source: Selectable Sprite Image

Image Position: Below or above the Dialogue Text

Color / Size: ←

Button Settings: These settings affect ALL Response Buttons in the Dialogue

Button Size: Sets the size of all Buttons. If the Font Size is too big, this must be enlarged, too. Or the text won't be visible.

Button Background: Separate background sprite for Buttons

Button Color: Individual Color of Button Background

Audio Settings: Sound that plays on appearance

5.2 Advanced Settings

In addition to the parameters above, checking this box will show more settings under Dialogue Settings and Text Settings. They are all taken over from 3 components of the Dialogue Gameobject. You'll find their Descriptions here:

- UI.Image: https://docs.unity3d.com/ScriptReference/UI.Image.html
- UI.Text: https://docs.unity3d.com/ScriptReference/UI.Text.html
- TextMesh Pro: http://digitalnativestudios.com/textmeshpro/docs/textmeshpro-component/

Panel Padding lets you set the space between the contents to the borders. It makes sense if the Panel Background sprite is not centered, f. I. a shape with a dropped shadow to the side.

5.3 Multi-Editing Dialogues

Multiple Nodes can be selected by holding down the Shift-Button and clicking some nodes. Moving and deleting is possible. If they contain dialogues, this Inspector appears:

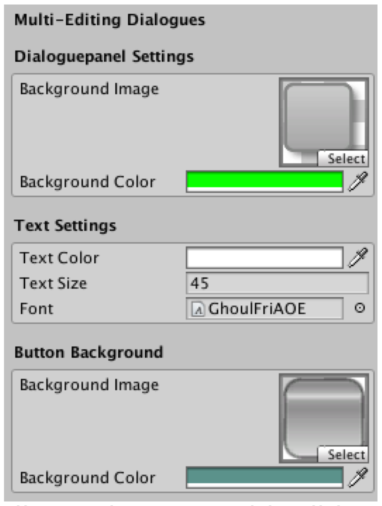

Multi-Editing Dialogues
Dialoguepanel Settings
Background Image
Background Color
Text Settings
Text Color
Text Size 45
Font GhoulFriAOE
Button Background
Background Image
Background Color

Illustration 13: Multi-Editing Dialogues

If you selected multiple Step Nodes with dialogues attached, you'll see "Multi-Editing Dialogues" in the Inspector.

The most important settings for dialogues can be edited at once. Background-, Text- and Button-Settings.

For deselecting the Nodes, release the Shift-Button, and click on a single Node. Multiselection mode will be canceled.

5.4 3D Dialogues

If you toggle the checkbox “3D Worldspace” (see Illustration 12: Step Node Inspector), the Dialogue will be moved from the 2D- to the 3D canvas. If no 3D canvas exists yet, one will be created and parented to TutorialSystem GameObject.
After that this Dialogue can be freely positioned in Worldspace by the Scene Editor:

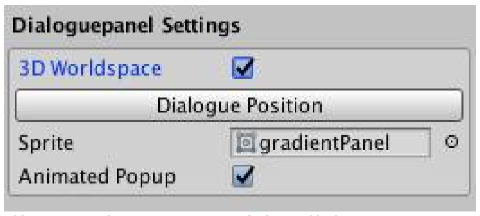

Dialoguepanel Settings
3D Worldspace ✓
Dialogue Position
Sprite gradientPanel
Animated Popup ✓

Illustration 14: Multi-Editing Dialogues

Just click the “Dialogue Position” button and you will be redirected to the Scene View with your Dialogue selected. Rotate, position or scale it like any other object.

5.5 Dialogue Gameobject

Every Dialogue is a separate Gameobject, and a child of TutorialSystem’s Canvas (TutorialSystem Gameobject). It consists of native Unity UI objects. The user is not supposed to alter any values at the Gameobject directly, but by the Inspector. If the need exists to go beyond the Inspectors capabilities, its components can be changed by clicking them in the Hierarchy, and accessing its values in the Standard Inspector. Although you have to be sure what you’re doing there.

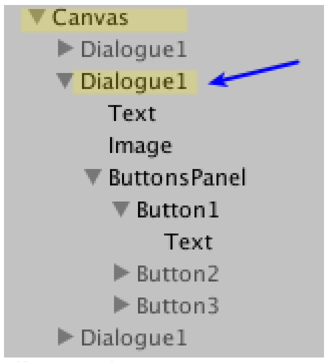

```
▼ Canvas
    ▶ Dialogue1
    ▼ Dialogue1
        Text
        Image
      ▼ ButtonsPanel
        ▼ Button1
            Text
        ▶ Button2
        ▶ Button3
    ▶ Dialogue1
```

Illustration 15: Dialogue Gameobject

Every Step Node with Dialogue creates an Instance of the chosen Dialogue. In this picture you see 3 Instances of “Dialogue1”, located as children of Canvas. Basic elements are Text, Dialogue image and a panel with 3 Buttons, while the number of visible Buttons is set by Step Node’s “Response Buttons” field.

By altering parameters of a “Step Node” Inspector, in reality the component values of these objects are changed.
Normally you should not need to change the GameObject directly.

5.6 Custom Dialogues

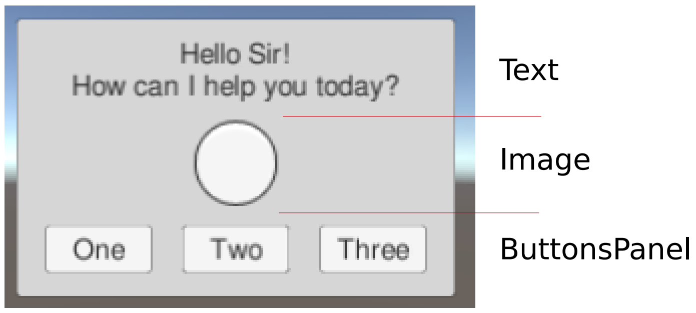
You can customize a Standard Dialogue a lot. If you still need special features, it is possible to create your very own Dialogue. But this is experimentally. Best would be to copy one of the existing Dialogues under "TutorialDesigner/Resources/ Dialogues" in your Assets folder. However, there are a few things to keep in mind:

- New Dialogue has to be in TutorialDesigner/Resources/Dialogues
- All objects of Illustration 15: Dialogue Gameobject have to be unchanged!
- Don't create objects with same names as the ones already existing
- ButtonsPanel must contain exactly 3 Buttons,

In this Asset there are 2 Dialogues included. Have a look at the next chapter to get a basic idea.

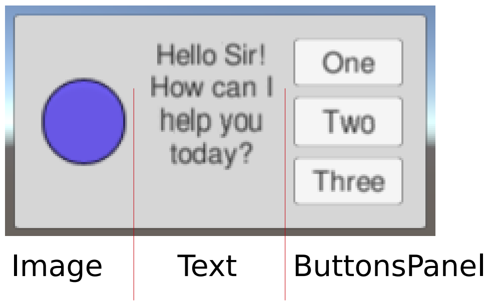
5.7 Included templates

1.)



Name: "dialogue_vertikal": Text and Image can swap positions. The Buttons are equally spreaded on the ButtonsPanel, which is at the bottom.

2.)



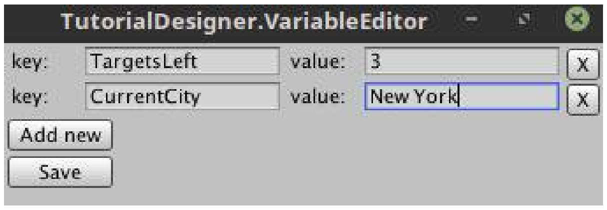
Name: "dialogue_horizontal": Text and Image can swap positions as well. The Buttons are equally spreaded from top to bottom.

6 Variables

Variables can be used inside Dialogue Texts for names, numbers or similar. Values that are random or change during the game. An example of how to use them is in the 2nd Demo Scene. There's a Player name to be called after the Player typed it into a box, and a countdown number which changes within the same Dialogue. See Demos

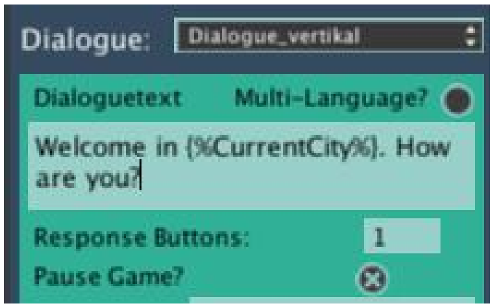
Variables consist of a key and a value, which are stored in a list in order to be serialized by Unity. There are two ways to define variables:

6.1 Variable Editor


TutorialDesigner.VariableEditor

key:	TargetsLeft	value:	3	X
key:	CurrentCity	value:	New York	X

By opening the editor for variables is the GUI-way of defining variables. You could define some keys with start-values here. They can then be called from within a StepNode:


Dialogue: Dialogue_vertikal

Dialoguetext Multi-Language?

Welcome in {%CurrentCity%}. How are you?

Response Buttons: 1

Pause Game?

The syntax to call variables inside Step Nodes (Dialogues) is: **{%VariableName%}**

So the variable name has to be inside "**{%**" and "**%}**"

It will be visible in the Scene- and Gameview immediately.

6.2 Scripting

There are 2 functions that can be used for setting and getting variables during runtime: They are included in SavePoint and can be called by using the active TutorialSystem's SavePoint component.

#SavePoint.cs

```
// Setting variables. If the key doesn't exist yet, it will be created.
// Returns true if a new key was created
public bool SetVariable(string key, object value)
```

Example:

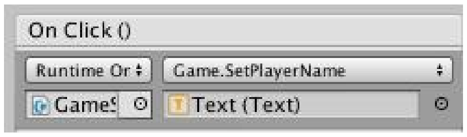
```
savePoint.SetVariable("CollectedTargets", 10);
```

```
// Get Variables
public T GetVariable<T>(string key)
```

Example:

```
int collectedTargets = savePoint.GetVariable<int>("CollectedTargets");
```

In the 2nd Demo Scene, the Player has to enter its name into a text box, which will save the entered text as variable by a custom script:a



When the "OK" button was clicked, Game.SetPlayerName will be executed – storing the entered text as variable.

#Game.cs

```
public void SetPlayerName(UnityEngine.UI.Text text) {
        savePoint.SetVariable("PlayerName", text.text);
}
```

7 TextMesh Pro

If you use TMP instead the native UI Text objects, every Dialogue contains a "TextMesh" object instead a "Text" object. All text related Stepnode settings will alter the TextMeshPro object now. If you want to use TextMesh Pro for your tutorial, make sure that TMP is installed in your project, and load the TMP module for Tutorial Designer.

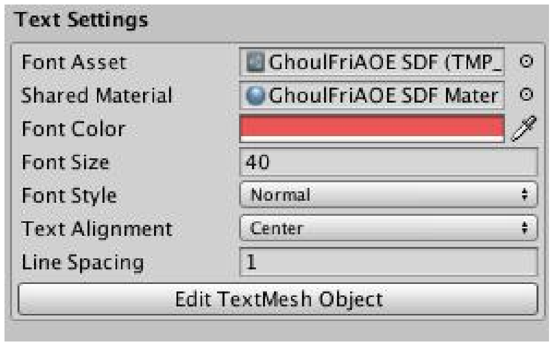
7.1 Load TMP Module

Window → Tutorial Designer → Load TextMeshPro Module

This will add the preprocessor directive "TD_MOD_TMPro" to your current Player Settings. More: https://docs.unity3d.com/Manual/PlatformDependentCompilation.html

If you switch to another player, it will automatically added as well (as long as the module is loaded). If you unload it however, you have to remove it from PlayerSettings → Custom Defined Symbols manually.

7.2 Stepnode Text Settings



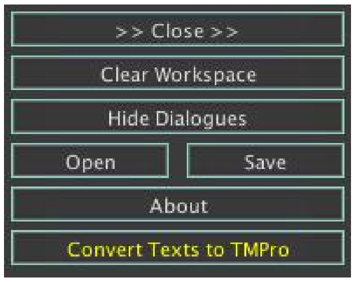
Text Settings

Font Asset	GhoulFriAOE SDF (TMP_
Shared Material	GhoulFriAOE SDF Mater
Font Color	🟥
Font Size	40
Font Style	Normal
Text Alignment	Center
Line Spacing	1

Edit TextMesh Object

If the box "Advanced Settings" is checked on your Stepnode , you will notice different Text settings in the Inspector. By clicking "Edit TextMesh Object", the Inspector jumps to the GameObject directly, that contais the TextMeshPro object.

7.3 Convert old Text objects to TextMeshPro



If you have made a scene without the TMP module (UI Text objects on dialogues), and then load the TMP module – the SideWindow will show an extra option: "Convert Texts to TMPro".

This will replace the Text objects of all dialogues in the current scene by TextMesh objects, preserving it's settings like color, alignment and so on.

If you want to keep event the font, you can use the "Font Asset Creator" to convert your font to a TMP Font Asset. Just put it in some Resource folder, and Tutorial Designer will find it during convertion.

More about Font Assets: http://digitalnativestudios.com/textmeshpro/docs/font/

8 Eventmanager

Events are triggered by one command:

```
TutorialDesigner.Eventmanager.TriggerEvent(“eventname”);
```

It can be called from everywhere in your scripts. If you want to call more than one Triggers from a single script, it makes sense to include the namespace “TutorialDesigner”. Like so:

```
using TutorialDesigner;
```

After this the Trigger command shortens to this:

```
Eventmanager.TriggerEvent(“eventname”);
```

<u>Example:</u>

Lest say you made an Event Node like this: Illustration 8: Event Node. There are 2 Events, we want to trigger the first one “ThroughTheDoor” right after the Game started. So we would follow these steps:

- Create a new script: “LevelActions.cs”
- Attach it to MainCamera, so it is executed at Game start
- Open the script and trigger the Event:

```
using UnityEngine;
using System.Collections;
using TutorialDesigner;

public class LevelActions : MonoBehaviour {
	void Start() {
		Eventmanager.TriggerEvent(“ThroughTheDoor”);
	}

	void Update() {

	}
}
```

Extend the script by the 2 yellow lines. Thats it! After starting the Game, this one is triggered. Since the counter in the Event Node picture is set to 1, the Event will be over after this call and passes over to the next connected Node.

9 One-Time Tutorial

If you check the box at Illustration 3: Tutorial System GameObject, you can define if this Tutorial appears only once for the Player, and also set the Tutorial Name for this purpose. It can be set for the current scene, or the whole Project - across all scenes.
Following keys will be written into PlayerPrefs if the TutorialName would be "Soccer":

Key for a Scene:
`TDesigner.[TutorialName].[SceneName]`
Example: `TDesigner.Soccer.PlayScene1`

Key for the whole Project:
`TDesigner.[TutorialName].Global`
Example: `TDesigner.Soccer.Global`

If one of these keys is set, the TutorialSystem GameObject will NOT start the Tutorial. Instead its SavePoint component will be disabled, which is responsible for all Gamelogic.

9.1 Functions for writing / reading / deleting the keys

If the player has finished the Tutorial, is of course defined during the Game. Therefore three simple functions are available to set the corresponding keys.
There are two ways to set the keys during Gameplay:
1.) By a Step Node (Recommended)

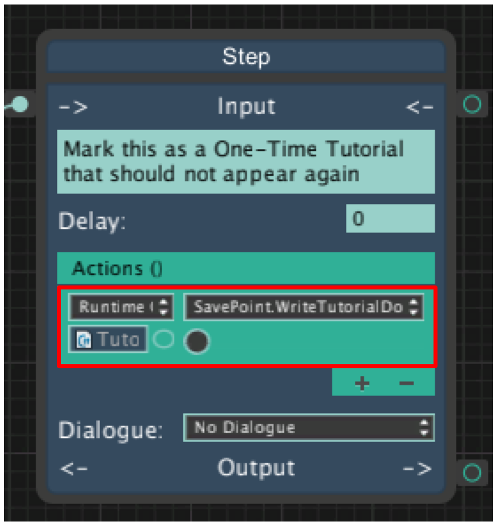

Step
->
Input
<-
Mark this as a One-Time Tutorial that should not appear again
Delay:
0
Actions ()
Runtime
SavePoint.WriteTutorialDo
Tuto
+
−
Dialogue:
No Dialogue
<-
Output
->

Illustration 16: Setting PlayerPrefs by Step Node

Create a new Action, drag the TutorialSystem Game-Object to the Object field, select **SavePoint.Write-TutorialDone**. Leave the bool-checkbox empty if this should count for the current scene only, or check it if it should count for the whole Project – global for all scenes.
For unsetting the key during the game, select **SavePoint.UnsetTutorialDone**

2.) Manually by Code

If you don't want to use Step Nodes for setting the keys, you can do it by code. All needed functions are in the SavePoint component of your TutorialSystem GameObject.

Since they are non-static function, you have to get a reference to the object. Then you have 3 Options for handling those keys:

```
GameObject TutorialSystem = GameObject.Find("TutorialSystem");
if (TutorialSystem != null) {
     SavePoint TDSavePoint = TutorialSystem.GetComponent<SavePoint>();
     // Write key to PlayerPrefs
     1.) TDSavePoint.WriteTutorialDone(bool forAllScenes);
     // Delete key from PlayerPrefs
     2.) TDSavePoint.UnsetTutorialDone(bool forAllScenes);
     // Check all keys from PlayerPrefs
     3.) TDSavePoint.IsTutorialDone();
}
```

1.) Write key to PlayerPrefs

Set the key either for the current scene (forAllScenes = false), or for the whole project (forAllScenes = true).

2.) Delete key from PlayerPrefs

Delete the key from PlayerPrefs. Same bool parameter as above.

3.) Check all keys from PlayerPrefs

This returns true if a global- OR scene-key was set. You can use this in your Game to determine if a Tutorial should be started or not.

9.2 Using the PlayerPrefs keys

This is a quick example to make a check in your Game, if the Tutorial was already done. Two things are important here:

- **SavePoint.oneTimeTutorial** – boolean Value. Set by TutorialSystem GameObject

- **SavePoint.IsTutorialDone()** - Function that returns a boolean if a Tutorial was done for the current scene OR the whole project. It checks the defined PlayerPrefs keys

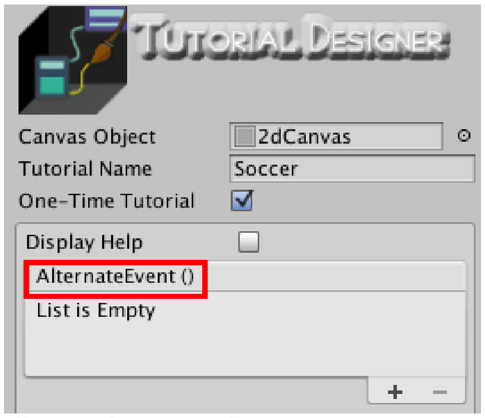
```
GameObject TutorialSystem = GameObject.Find("TutorialSystem");
if (TutorialSystem != null) {
       SavePoint TDSavePoint = TutorialSystem.GetComponent<SavePoint>();
       if (TDSavePoint.oneTimeTutorial && TDSavePoint.IsTutorialDone()) {
              // Your code here
              // If the Tutorial was done, it will be considered here
       }
}
```

Include this checks in your code to check if the player finished the Tutorial. To reactivate it by Player action, use **TDSavePoint.Reactivate()**, f.I. by an Unity-UI-Button. This re-enables the SavePoint component and starts the Tutorial at the beginning.

9.3 Alternate Action

Let’s say you want the Tutorial to appear only once (by hitting the “One-Time Tutorial”-toggle on the TutorialSystem Gameobject). Imagine the Player already completed the Tutorial and starts the Game the second time. Since the toggle is checked, at this point the Tutorial does NOT appear again.

Now you can define alternate Actions do be executed, instead of activating the Tutorial. F. I.: jumping to another Section in the Level, arming the Player or anything else. Define those actions in the Inspector, with TutorialSystem GameObject selected:



Canvas Object: 2dCanvas
Tutorial Name: Soccer
One-Time Tutorial: ✓
Display Help: ☐
AlternateEvent ()
List is Empty

Illustration 17: Alternate Event

Like at Step Nodes a UnityEvent is shown here, handled the same way.

It appears only if “One-Time Tutorial” is toggled.

10 Build your Game

Don’t worry about the overhead of all the TutorialEditor features, that are running while you test your Game. They won’t affect your Game performance. In builts, the Tutorial-Designer is minimized to only it’s processing functions, leaving all Editor features aside. In a nutshell, this is what will be included in a built:

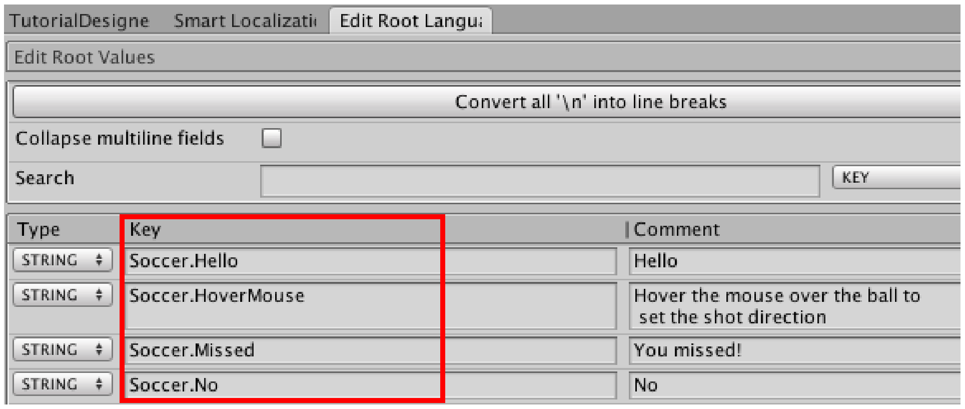
- Working routines of existing Nodes
- Information about Node’s connections, for understanding the workflow
- Dialogue Gameobjects

11 Multi-Language

There's a free Third Party Plugin included, for making Dialogue-Texts and -Buttons multilingual – SmartLocalization 2:
https://github.com/NiklasBorglund/Smart-Localization-2
I would recommend to have a look at its documentation for basic usage. Or you can check out the first Demo scene, which is made multilingual: Demos

11.1 SmartLocalization

The plugin is opened by Window → Tutorial Designer → Smart Localization. There is one thing to take note, keep the naming convention of your language keys clean and short. Those are going to be used in Popup Menus within Step Nodes.



Type	Key	Comment
STRING	Soccer.Hello	Hello
STRING	Soccer.HoverMouse	Hover the mouse over the ball to set the shot direction
STRING	Soccer.Missed	You missed!
STRING	Soccer.No	No

Illustration 18: SmartLocalization Language Keys

11.2 Integration in your Tutorial

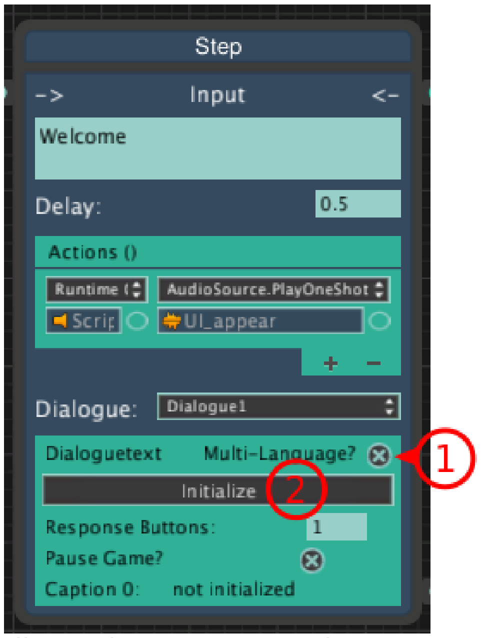


Illustration 19: Step Node + ML

1.) Check this toggle to enable Multi-Language support on this particular Node.

2.) To refresh the Tutorial Designer's language database according to SmartLocalization, click "Initialize". This must also be done after making changes to the existing database of Smart-Localization.

After initializing TutorialDesigner’s language database - which means it will take over all settings from SmartLocalization – you’ll notice Popups instead of Textfields, now. Within those Popups you can chose a given key of SmartLocalizations database. It will be displayed in the chosen language (see below).

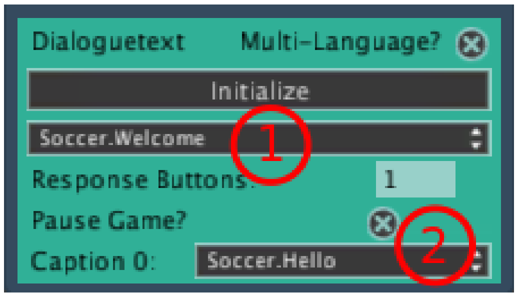

Dialoguetext Multi-Language?
Initialize
Soccer.Welcome
Response Buttons 1
Pause Game?
Caption 0: Soccer.Hello

Illustration 20: Language Key Selection

1.) Dialogue-Text Popup

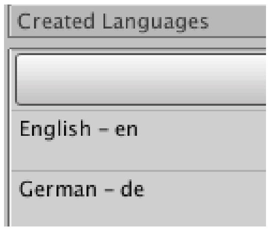
2.) A Button caption Popup for every Response Button will appear

Play around with these settings, you will see your Dialogue, updated in realtime.

11.3 Change language

Select the TutorialSystem Gameobject. In the Inspector a new Option will be available: “Dialogue Language”
Dialogue Language en

The available languages depend on SmartLocalization’s created languages:


Created Languages
English - en
German - de

See the Getting-Started guide to learn how to create languages: SmartLocalization (Page 3)

The short language codes (en, de) will appear in the Popup above. By changing that, all Dialogue-Texts and -Buttons will get the new language automatically.

12 Contact

If you have any questions, comments, suggestions or other concerns about our product, do not hesitate to contact us. You will find all important links in our „About“ window, located in the Sidewindow:

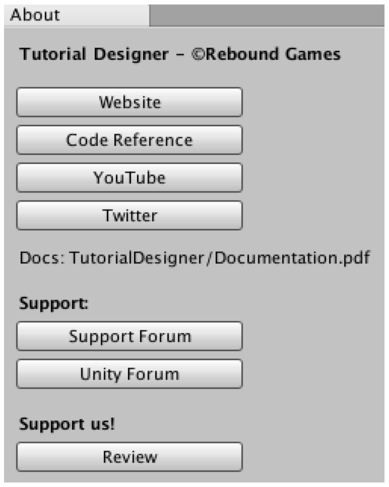


Illustration 21: About Window

There you'll find all current Links to our Website, to our Forum, as well as the Unity Forum. If you have any issues, this is the first place to look at.

Also all the Code is well documented, you get access to it by the “Code Reference” button.

If you don't find your answer there, don't hesitate to contact us at: info@rebound-games.com, and we are happy to help.

If you would like to support us on the Unity Asset Store, please write a short review there so other developers can form an opinion. Thanks for your support, and good luck with your apps!

Rebound Games