

## Programming tutorial – Making a dialogue system in Unity

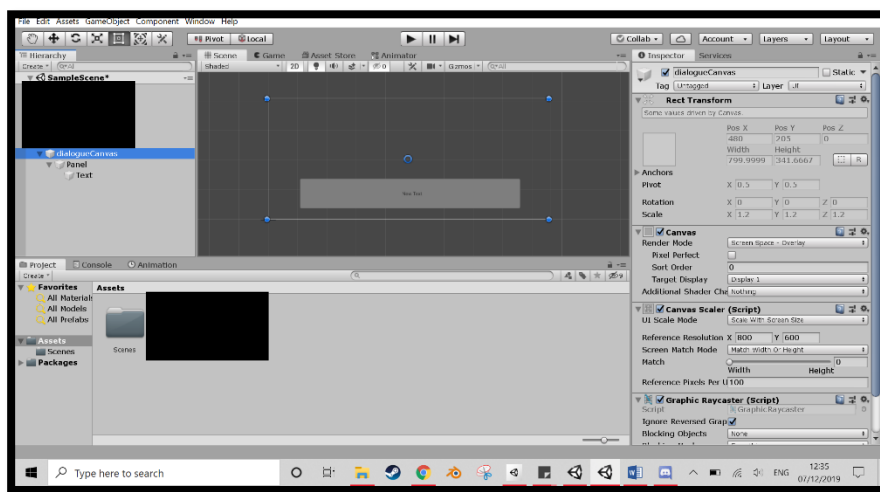
How to make a dialogue system in Unity:

### 1. Set up the scene

Create a new canvas in the scene and name it dialogueCanvas, ensure that the “Canvas Scaler” of the canvas is set to “Scale with screen size” so that the dialogue box doesn’t change on different screen sizes.

After this, add a panel to the canvas and scale it to a suitable size for a text box in your game. Add text to this panel and alter it according to how you want the text to look, untick the ‘Best Fit’ option so that the scale of the text remains the same size no matter how much text there is.

The scene should look something like this:



### 2. The public variables and the dialogue script

Create a new script and name it “Dialogue”, to ensure that the script is able to link with text components (and other UI components), type “using UnityEngine.UI;” beneath “using UnityEngine;”.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
```

Once this is done create a public variable – “public Text dialogueText;” (under the public class function) which will allow the script to link with the text created in step 1. Beneath the newly created public variable, create a public string array named ‘sentences’ (“public string[] sentences;”), this will hold all the sentences within the scene. Finally, create a private int variable called ‘index’ (“private int index;”) which refers to the number of sentences in the array.

These should look like:

```
public class Dialogue : MonoBehaviour
{
    public Text dialogueText;
    public string[] sentences;
    private int index;
```

### 3. Sentence typing effect

Create a coroutine called 'Type' by adding (Below the Start() and Update() functions):

```
"IEnumerator Type ()
```

```
{
```

```
}
```

```
"
```

What an IEnumerator/coroutine function does is that it does not immediately call the function within it when the game starts, the code inside waits for it to be called then after a few seconds, or after a frame/few frames, it does its work.

Between the brackets, create a 'foreach' loop and make it call the same amount of times as the amount of letters in the sentence that is currently playing:

```
"foreach (char letter in sentences[index].ToCharArray())
```

```
{
```

```
}
```

```
"
```

And within this add:

```
"
```

```
dialogueText.text += letter;
```

```
"
```

This will add a single letter to the dialogue text on display when the function runs.

After this, to make it so that the next letter is added after a short delay, add a 'WaitForSeconds(xx)' rule:

```
"yield return new WaitForSeconds(0.02f);"
```

So far, the script should look like:

```
0 references
void Update()
{
    ...
}
1 reference
IEnumerator Type()
{
    foreach (char letter in sentences[index].ToCharArray())
    {
        dialogueText.text += letter;
        yield return new WaitForSeconds(0.02f);
    }
}
```

Finally, call the coroutine in the Start() function by typing “StartCoroutine(Type());”

```
// Start is called before the first frame update
0 references
void Start()
{
    StartCoroutine(Type());
}
```

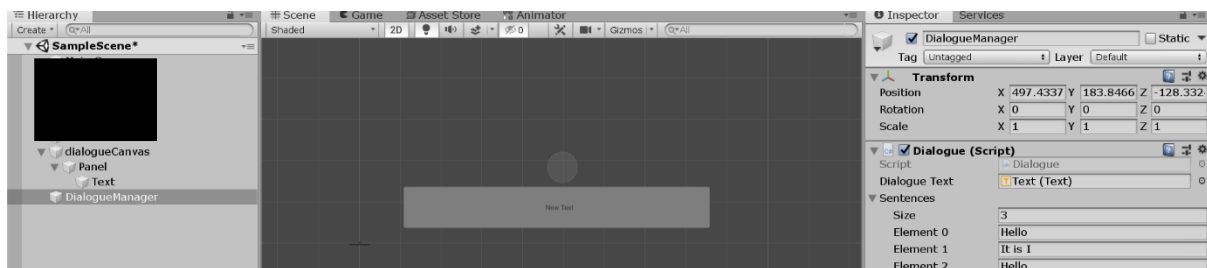
Save this and go back to Unity.

#### 4. Script test

Create an empty game object within Unity called ‘Dialogue Manager’ and drag and drop the ‘Dialogue’ script into it. Also drag the text box from the dialogueCanvas canvas into the text box section.

Write a few random sentences into the array and change the ‘size’ to any number you want – the size is the amount of sentences/indexes.

It should look like this:



Once you do this, remove the sentence ‘New Text’ from the text object and play the scene to test it, the first sentence should be ‘typed’ out.

To change the typing speed of the text in the inspector, go back to the ‘Dialogue’ script and create a public variable called “public float typingSpeed;”. Replace the “0.02f” within the ‘WaitForSeconds()’ section with ‘typingSpeed’ and you should be able to change it back in the unity project.

This should look like:

```
public float typingSpeed = 0.02f;

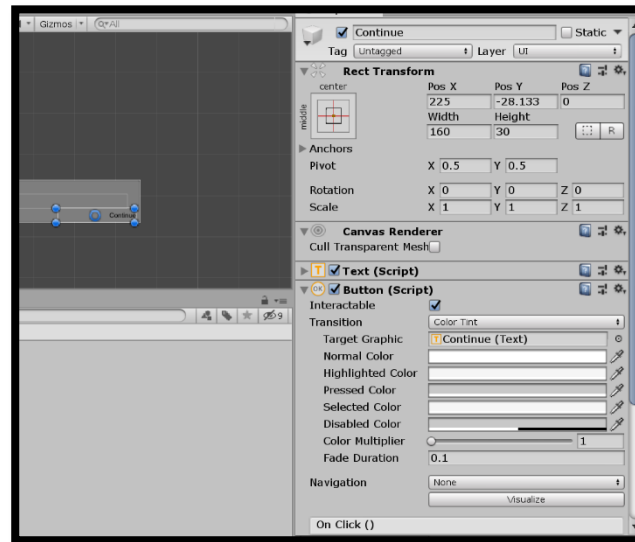
// Start is called before the first frame update
0 references
void Start()
{
    StartCoroutine(Type());
}

// Update is called once per frame
0 references
void Update()
{
}

1 reference
IEnumerator Type()
{
    foreach (char letter in sentences[index].ToCharArray())
    {
        dialogueText.text += letter;
        yield return new WaitForSeconds(typingSpeed);
    }
}
```

## 5. The continue button

Go back to the Unity project and add a new piece of text to the 'dialogueCanvas' canvas, name it "Continue" and also change its text to "Continue". Add a button component to this text and place it wherever you want.



Go back to the 'Dialogue' script and create a public void function called 'NextSentence' below the IEnumerator function.

```
public void NextSentence()
{
}
```

This makes it so that the code within it only starts when the function is called.

Within it, to ensure that the player can't press continue without the dialogue finishing and also to go to the next sentence without the sentences stacking, type:

```
if(index < sentences.Length - 1)
{
    index++;
    dialogueText.text = "";
}
"
```

This code states that if the current index/sentence number is less than the total number of sentences/indexes, the code within the brackets starts. 'index ++' means that the next index/sentence will be played and 'dialogueText.text = "";' ensures that the text is reset after each sentence. 'sentences.Length -1' is used rather than just 'sentences.Length' because the index starts from 0.

Below 'dialogueText.text = "";', start the coroutine 'Type' – "StartCoroutine(Type());"

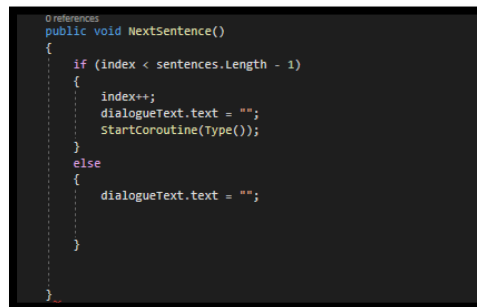
This will start the IEnumerator/coroutine created earlier.

Then below the closed bracket, type:

```
"else {
dialogueText.text = "";
}
```

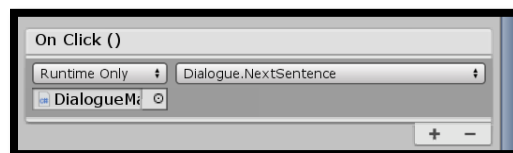
This will reset the dialogue once it is complete.

So far, the script should look like:



```
0 references
public void NextSentence()
{
    if (index < sentences.Length - 1)
    {
        index++;
        dialogueText.text = "";
        StartCoroutine(Type());
    }
    else
    {
        dialogueText.text = "";
    }
}
```

Once this is done, go back to the Unity project and add an OnClick() event to the 'Continue' button. Drag the Dialogue Manager into the gap and select the new 'NextSentence' function from the dropdown list.



Test the continue button by playing the scene.

You will notice that if you spam click the continue button, the words will start to jumble together, to stop this from happening, you will need to enable and disable the continue button at the right time.

To do this, go back to the Dialogue script and create a public gameobject called 'continueButton' – "public GameObject continueButton;". Within the Update() function create an 'if' statement to check whether the text that is currently displaying is equal to the current sentence:

```
"if(dialogueText.text == sentences[index])
{
}
"
```

And within the brackets write:

```
"continueButton.SetActive(true);"
```

Which will enable the button when each sentence is finished.

To hide the button once it has been hit, write “continueButton.SetActive(false)” within the NextSentence() function.

Also, to hide the button once the dialogue has ended, type the same thing within the ‘else’ brackets in the NextSentence() function.

This whole script should look like:

```
using UnityEngine;
using UnityEngine.UI;

public class Dialogue : MonoBehaviour
{
    public Text dialogueText;
    public string[] sentences;
    private int index;

    public float typingSpeed = 0.02f;
    public GameObject continueButton;

    // Start is called before the first frame update
    void Start()
    {
        StartCoroutine(Type());
    }

    // Update is called once per frame
    void Update()
    {
        if (dialogueText.text == sentences[index])
        {
            continueButton.SetActive(true);
        }
    }

    IEnumerator Type()
    {
        foreach (char letter in sentences[index].ToCharArray())
        {
            dialogueText.text += letter;
            yield return new WaitForSeconds(typingSpeed);
        }
    }

    public void NextSentence()
    {
        continueButton.SetActive(false);
        if (index < sentences.Length - 1)
        {
            index++;
            dialogueText.text = "";
            StartCoroutine(Type());
        }
        else
        {
            dialogueText.text = "";
            continueButton.SetActive(false);
        }
    }
}
```

After this, go back into Unity, drag the ‘Continue’ button into the new field, disable the button, and test the script one last time by playing the scene, you should have a working dialogue system.