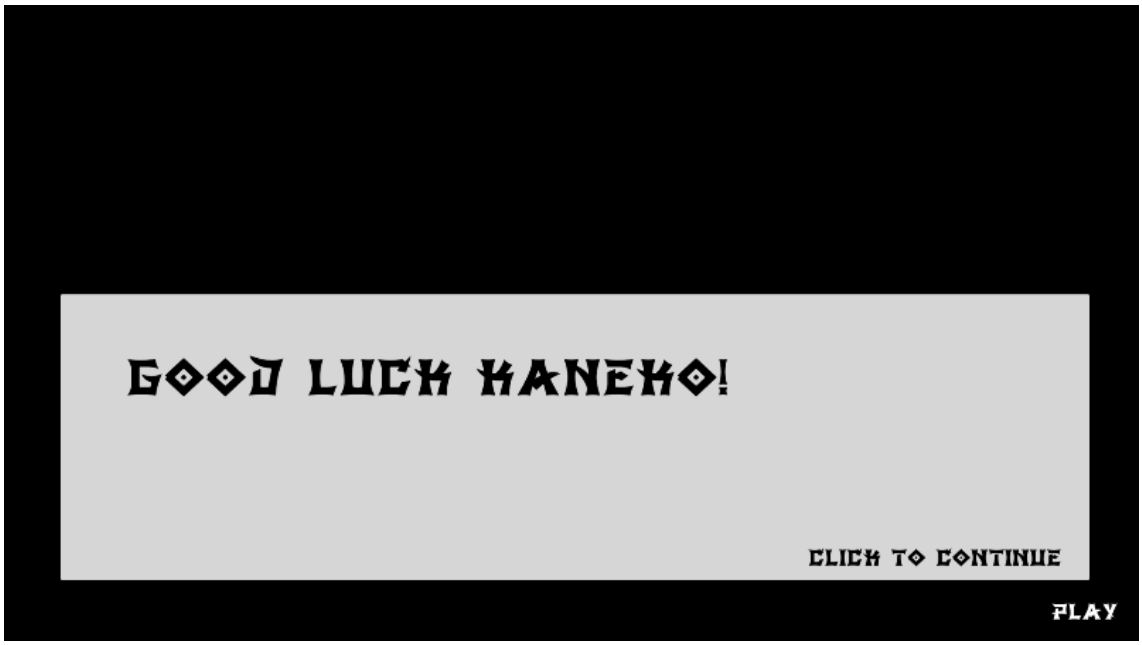
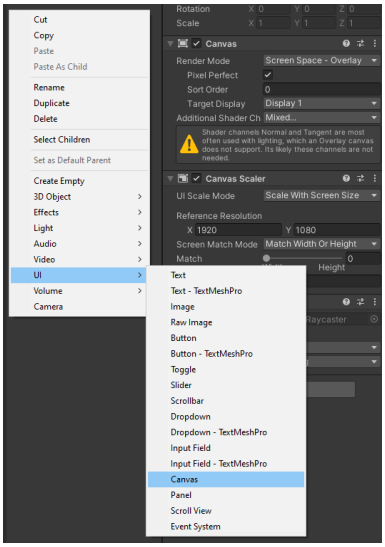
Dialogue System

By Nathan Stalley

In this tutorial, we will be making a dialogue system that can be used to display text on screen, the text will be typed out one character at a time and we will have an option that allows for multiple lines to be typed out and a way to move between these.



To start off with, we need to make a canvas for the dialogue to be displayed on. To do this, right click on the hierarchy, hover over UI, and then select Canvas.

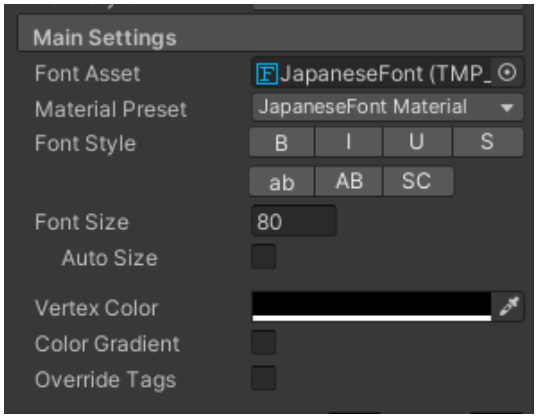


Before we move on, we need to set up the canvas. In this case, we have selected the UI Scale Mode to be 'Scale with Screen Size' and we have set our Reference Resolution to be 1920 x 1080 as we are targeting HD.

Next, we will need to create a panel for the dialogue to be displayed on. To do this, right click on the Canvas object in the hierarchy (to make it a child object) and scroll down to UI and this time, select Panel.

The Panel will take up the size of the Canvas, so let us re-size it. I have changed the Left, Right and Bottom values to 100 and the Top value to 500, this creates a nice box effect for our dialogue box, but you can of course play with these settings until you get the result that you desire. After re-sizing the panel, I also changed the colour to a solid white, to do this, scroll to Image in the Inspector, click on Colour, and change the colour to whatever you desire. You may also need to change the opacity to make the panel a solid colour.

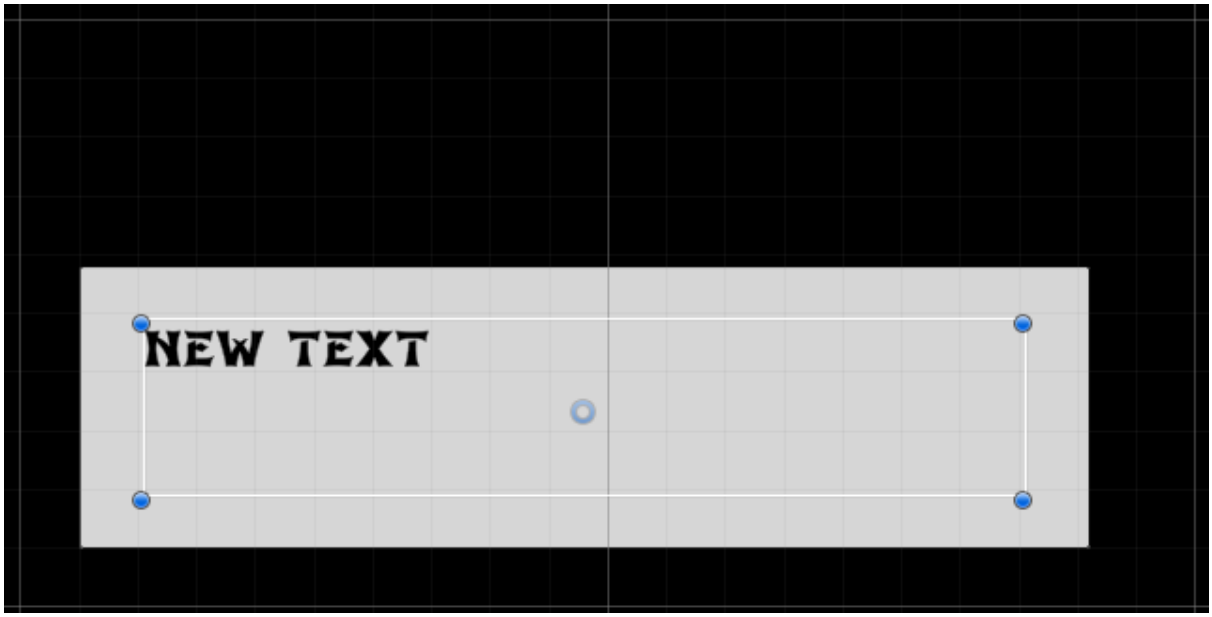
The next step is to add text to the Panel that we have made. To do this, right click on the dialogue box in the hierarchy, hover over UI and select Text – TextMeshPro. When you click on TextMeshPro (TMP), you will have a TMP Importer window pop up, to use TMP you need to click on 'Import TMP Essentials' the 'Examples & Extras' are optional.



Once the TMP object has been created, you are able to change some of the settings of the text itself.

To start with, I just changed the text colour to black and increased the font size to 80, I also added in a custom font that I found online, but I will go over that towards the end of this tutorial. You can change the settings to fit your game.

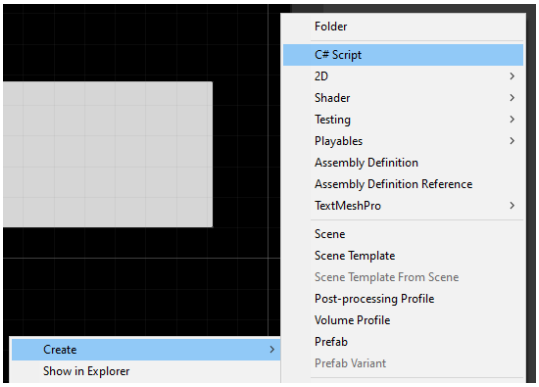
After that, I also changed the Rect Transform Width and Height, I changed these to 1500 and 300 respectively and here is the result.



As you can see, the text box and font are both of a decent size, the next step we will take is to make the text appear on screen one character at a time and we will also code the ability to skip through the text.

So, the next step is to make a C# script, to do this, right click in your Assets folder, hover over Create and then click on C# Script. From here, name the Script, I named mine Dialogue.

Before opening the script, drag the script from the Assets folder onto the Dialogue game object. From here, you can double click on the script in the Assets folder to open it up.



Once the script has opened in the editing program of your choice (Visual Studios for me), the first thing we need to add a Using statement at the top of the script. We need to add "using TMPro", this will allow us to reference the TMP component.

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

```
public class Dialogue : MonoBehaviour
{
    public TextMeshProUGUI textComponent;
    public string[] lines;
    public float textSpeed;

    private int index;
```

Next, we will add all the variables that we will need to use within the script.

To start with, we will add a reference to the TMP component, we will make this public and call it 'textComponent'.

We then need somewhere to store all the lines that we wish to have displayed on the screen as dialogue, to do this, we will make a public string array and call this one 'lines'. The array is shown by the square brackets after string.

Next, we need a value to control the speed at which the text is typed out onto the screen, here we will make a public float and call it 'textSpeed'. We have made it public so that the value can be changed within the inspector and we have used a float so that we are able to use decimal point values. We then need to make a private index; this will be used to keep track of where we are within the text that is being typed out on screen.

The next part of the script is what is used to type the characters out one at a time.

We need to set up an Enumerator that will be used by a Coroutine, we will call this Enumerator TypeLine() and we will reference this is a minute. Inside this Enumerator, we type out 'foreach' and then inside of the brackets, type 'char c' for the characters, then type in 'lines' for our collection of the current [index] and then we will break this down into a character array by using ToCharArray().

```
IEnumerator TypeLine()
{
    foreach (char c in lines[index].ToCharArray())
    {
        textComponent.text += c;
        yield return new WaitForSeconds(textSpeed);
    }
}
```

Inside the curly brackets, type in 'textComponent.text +- c' this will add the characters in one by one, just under this, type in 'yield return new WaitForSeconds(textSpeed)' this will be used to determine how fast or slow the characters will be typed out.

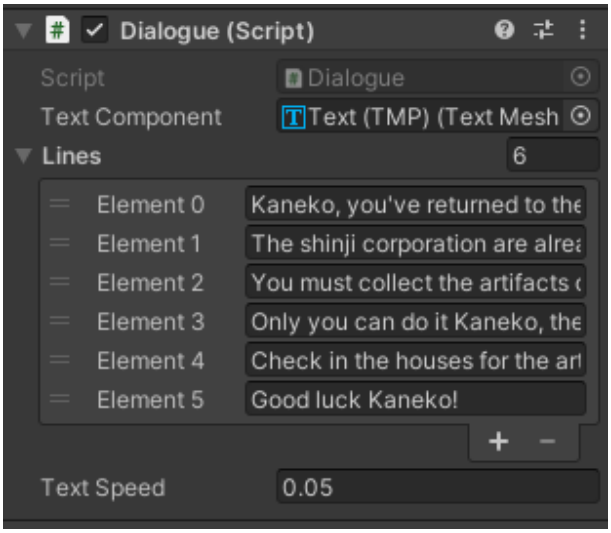
```
void StartDialogue()
{
    index = 0;
    StartCoroutine(TypeLine());
}
```

After we have created the Enumerator, we need to reference it using a Coroutine, create a new void called StartDialogue() and in here we will set the index to 0 (this will start the text typing from the beginning) and then pass the enumerator we just made though a StartCoroutine ().

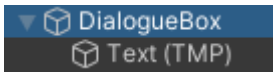
```
void Start()
{
    textComponent.text = string.Empty;
    StartDialogue();
}
```

Then, under start we need to call the textComponent string and the Start Dialogue () method, after this, we can jump back into unity.

Once you are back into Unity, you should see that our script has a text component, lines, and text speed.



To start with, drag the TMP component from



the hierarchy into the script in the inspector, then you can change the number of lines to what you desire, this will be used for your dialogue. Finally, you can set the text speed, this will determine how quickly each character is typed out.

At this stage, the text will type out one character at a time, but you will not be able to move to the next line of dialogue, so let us fix that.

```
void NextLine()
{
    if (index < lines.Length - 1)
    {
        index++;
        textComponent.text = string.Empty;
        StartCoroutine(TypeLine());
    }
    else
    {
        gameObject.SetActive(false);
    }
}
```

To make the text move to the next line, we need to set up this method.

This is used to check the length of the text and to see if anything else in the dialogue needs to be typed out. Once everything has been typed out, the game object will be set to false and will be made inactive.

After we have made this method, we need to head back to void update and create another if statement.

```
void Update()
{
    if (Input.GetMouseButtonDown(0))
    {
        if (textComponent.text == lines[index])
        {
            NextLine();
        }
        else
        {
            StopAllCoroutines();
            textComponent.text = lines[index];
        }
    }
}
```

This if statement will take mouse button 0 (left click) and will use it to move to the next line after all the text has been typed out.

It is also used to fill out the rest of the array, this will allow you to skip the individual typing of the text and progress through the dialogue quicker.

After this final piece of code, you will have completed a basic dialogue system that can be used within your game.