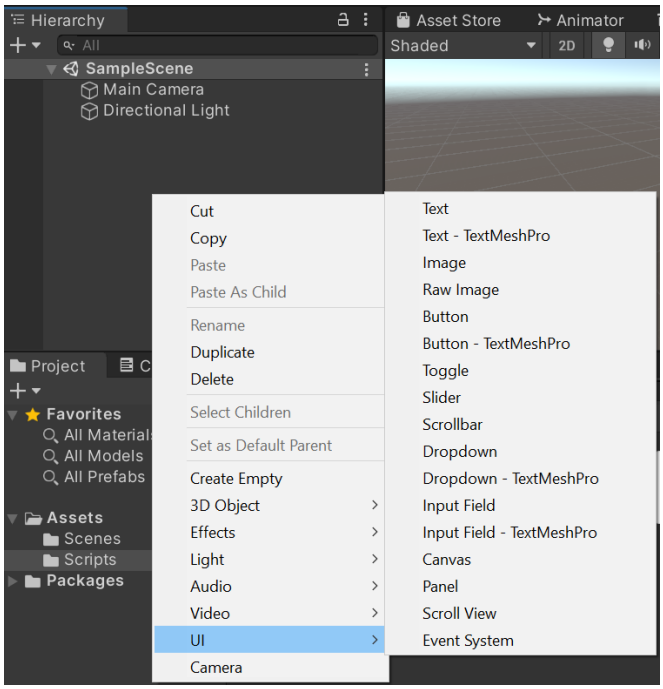
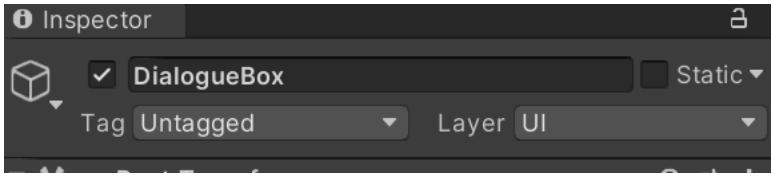
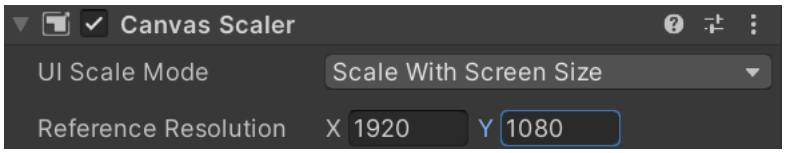
In this tutorial, I will be showing you a quick and efficient way of making a dialogue system.

Right click in the hierarchy, go to UI and click panel. You're then going to rename this panel to ‘DialogueBox’

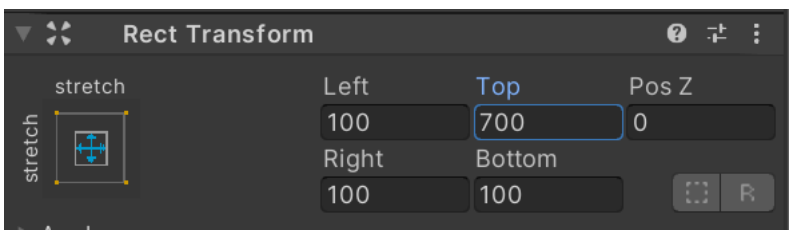




Next, we are going to start configuring the canvas. To do this, click on your canvas, scroll to ‘Canvas Scaler’, and change the ‘UI Scale Mode from ‘Constant Pixel Size’ to ‘Scale with Screen Size’. In this case, I'm going to set the X & Y value to ‘1920 by 1080’.

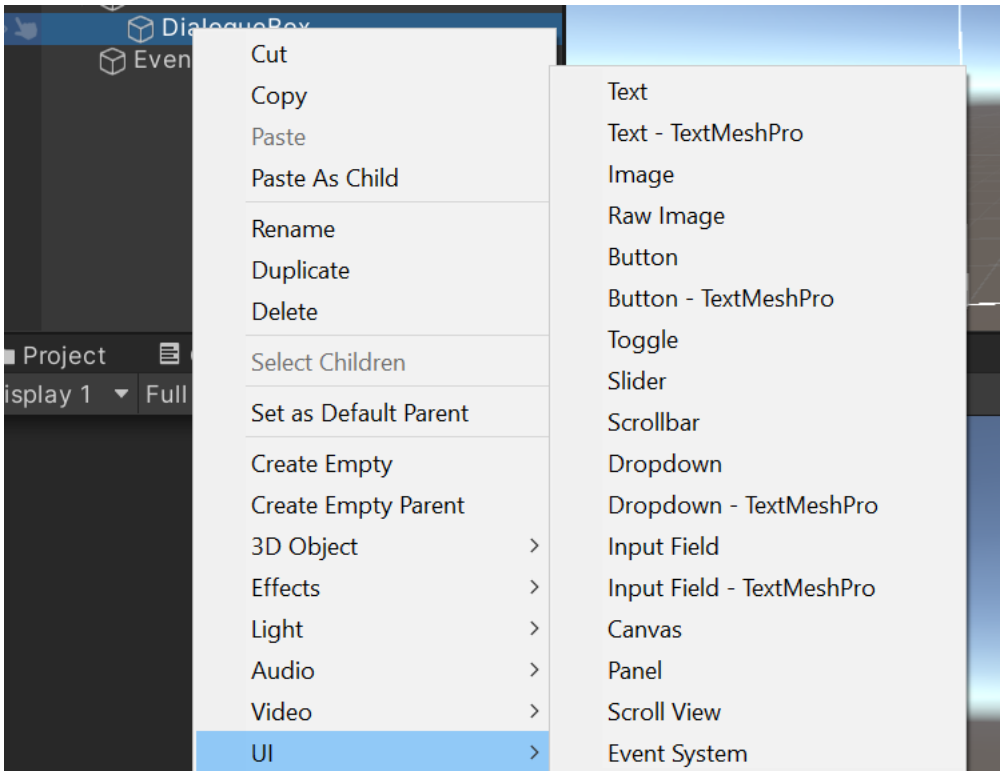


We are now going to go back to our dialogue books and start editing all wrecked values in the inspector. For the left value I am going to change it to 100 I am going to change the top value to 700 the right value to 100 and the bottom value to 100.

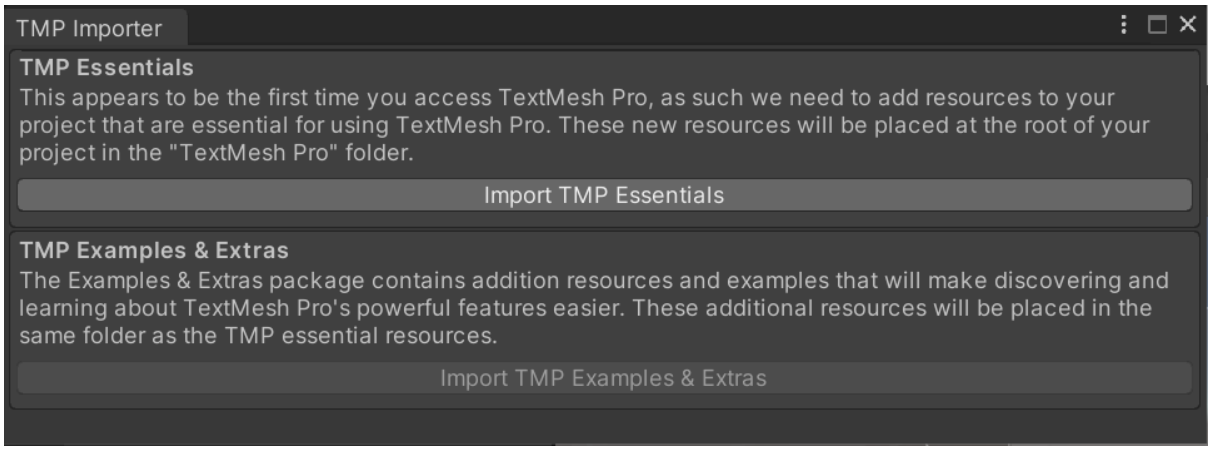


Katurah Cephas

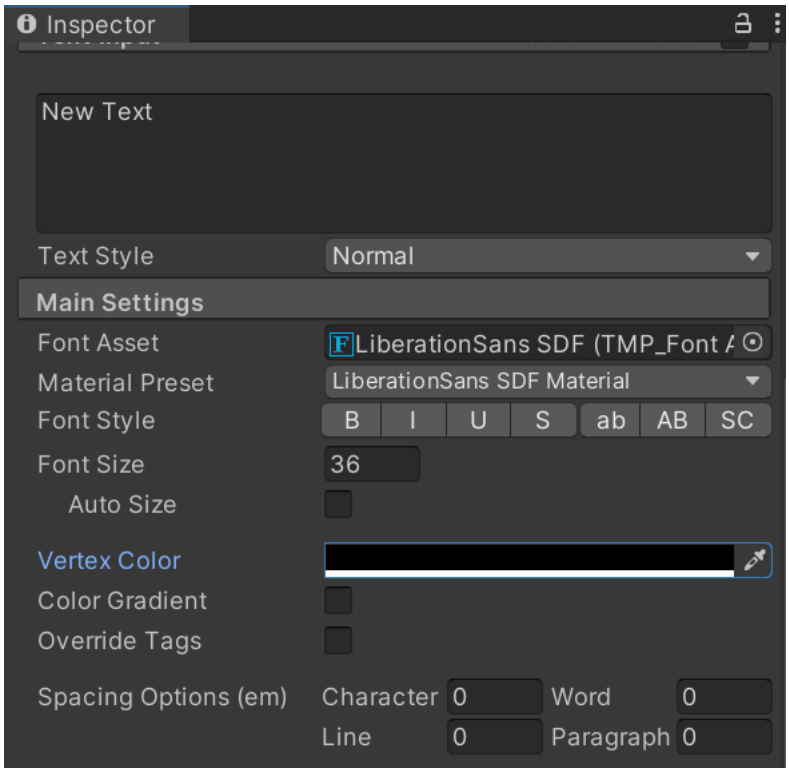
Right click on the dialogue box in your hierarchy, head to UI, and select ‘Text- TextMeshPro.



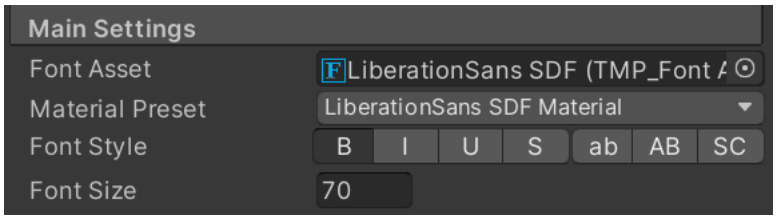
Once you click this, a text box should appear. Click on ‘Import TMP Essentials’ to continue.



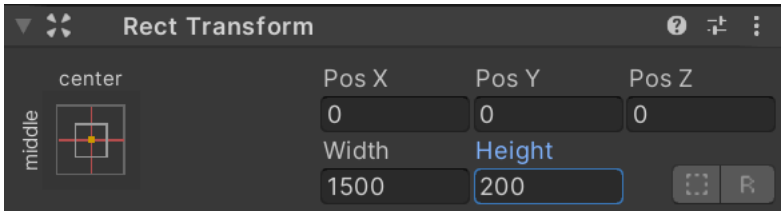
Once the importing stage has finished, click on the new text object that has been created and change the vertex colour to black in the inspector.



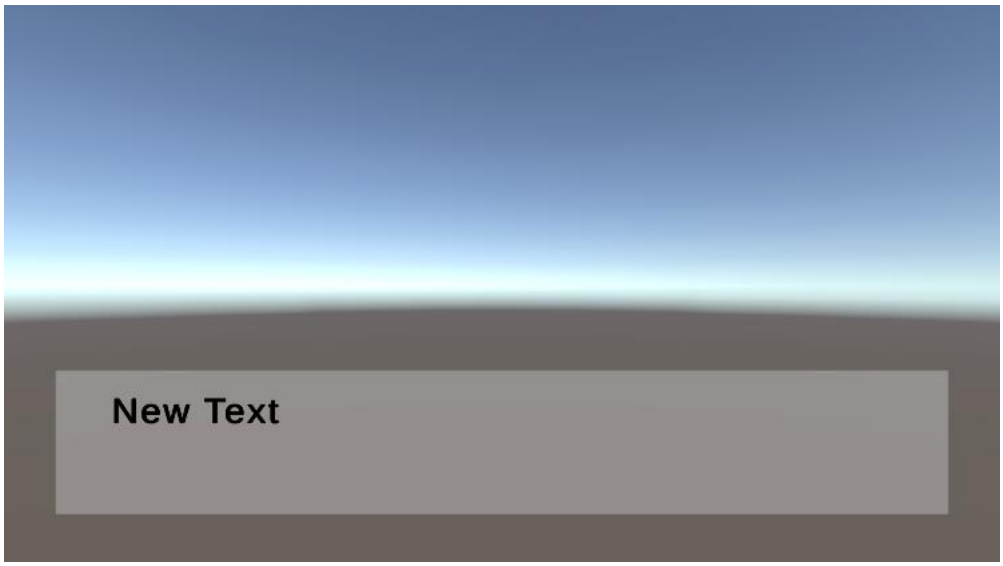
For the font settings, we can go ahead and change the font style to bold and the font size to around 70.



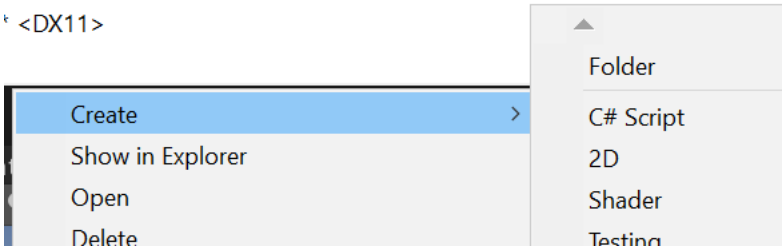
In order to change the position of our text, we can change the rect values of the text object. I am going to change the width to 1500 and the height to 200 list should make the text fit nicely into the dialogue box.



Right now, your dialogue box should look something like this.



Now, we are going to want to start working on our script. In your script folder, right click, click create and create a ‘C# script’.



Rename this script ‘Dialogue’ and double click to open it.

At the top of the page, We are going to want to ad a using statement so that we are able to reference the TextMeshPro component.

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

Now we want to start adding all of the variables that we need for the script. First, we are going to want a reference to our TextMeshPro Component. Next, we need a reference for the number of lines we need for the dialogue, in this instance I used a string array to make things simple. We need a reference for the speed of our text. And finally, we need a reference for where we are in the dialogue. It should now look something like this.

```
public class Dialogue2 : MonoBehaviour
{
    public TextMeshProUGUI textComponent;
    public string[] lines;
    public float textSpeed;

    private int index;
```

Underneath the Start and Update method, create a new method called 'StartDialogue'. Here we are going to want to set the index equal to 0.

```
0 references
void StartDialogue()
{
    index = 0;
}
```

Underneath our StartDialogue Method, we are going to want to make a coroutine. This is going to make the characters in our dialogue box be typed out letter by letter. The text speed controls how fast each letter is placed.

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

Go back to the StartDialogue method and reference the Coroutine that we have just created.

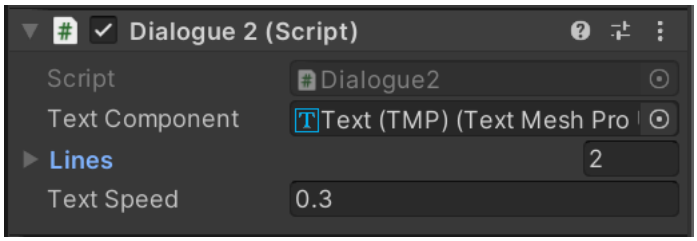
```
void StartDialogue()
{
    index = 0;
    StartCoroutine(TypeLine());
}
```

Katurah Cephas

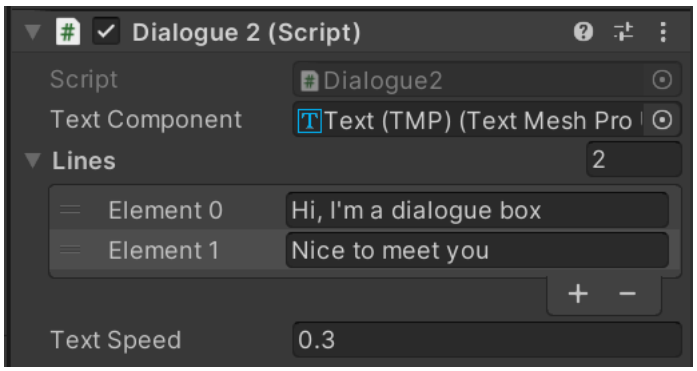
In our Start Method, we are going to reference the start dialogue method in order to get our dialogue system working.

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

Drag the Dialogue script onto your DialogueBox and proceed to drag in your text object where it says ‘Text Component’. The number of lines is the number of sentences you want to have, in this case, I have two. Finally, set the speed to 0.3.



In the Lines dropdown, write your desired sentences line by line.



We are now going to be working on the code that allows the dialogue box to move to the next line. Create a new method called ‘NextLine’ This code activates the next line, else, if there are no more sentences, the dialogue box becomes inactive.

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

Katurah Cephas

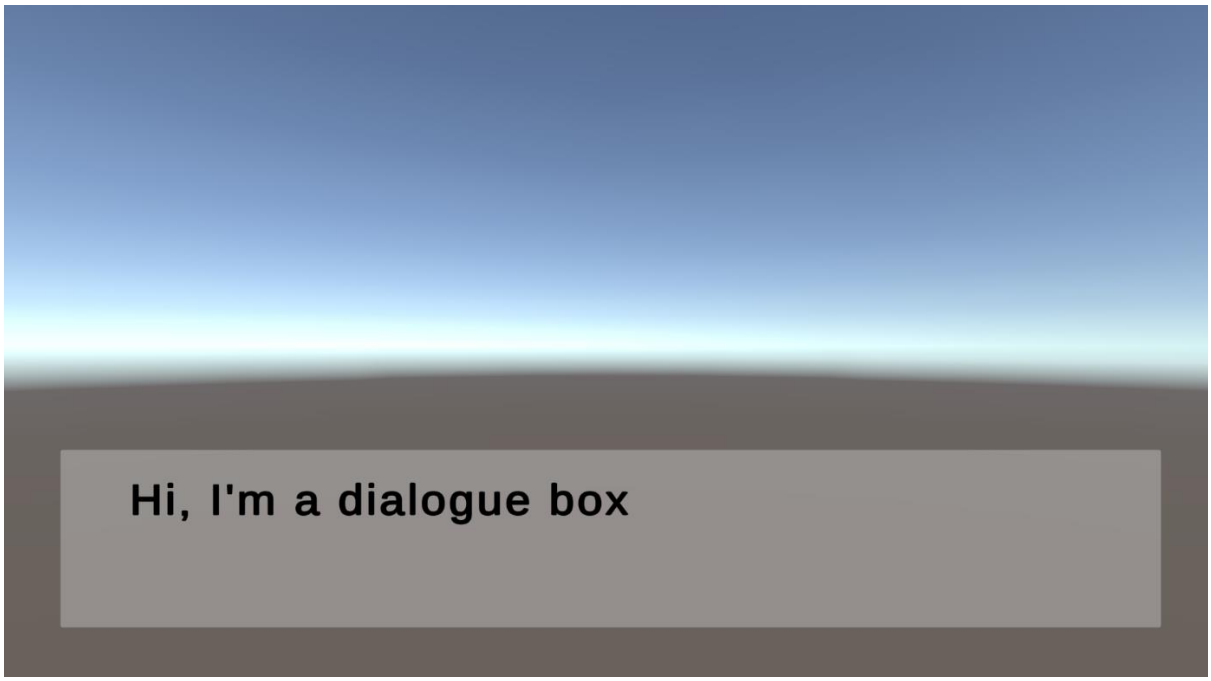
In the Update method, we are going to be controlling the activation of the next line when the mouse button is clicked. When the dialogue is finished, we can click and close the dialogue box.

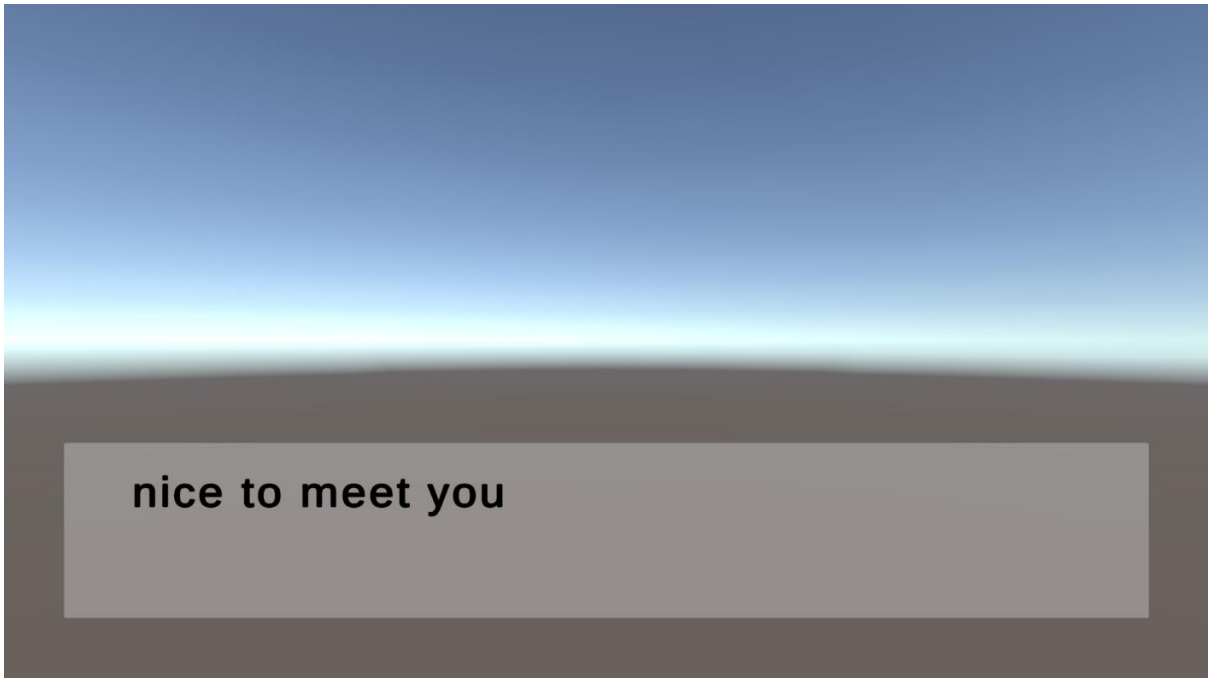
```
void Update()
{
    if (Input.GetMouseButtonDown(0))
    {
        if (textComponent.text == lines[index])
        {
            NextLine();
        }
        else
        {
            StopAllCoroutines();
            textComponent.text = lines[index];
        }

    }

}
```

Save your script, and your dialogue system should now be fully functional when you play your game.




nice to meet you