Dialogue Manager User Documentation

This document explains how to use the Dialogue Manager script in your Unity project. The Dialogue Manager is responsible for displaying dialogue text to the player, controlling the flow of conversation, and handling user interaction.

How it Works

The Dialogue Manager works in conjunction with NPCManager script. Here's a breakdown of the process:

1. **Triggering Dialogue:** When the player interacts with an NPC, the NPCManager calls the ShowDialogue method in the DialogueManager, passing itself as a parameter.
2. **Displaying Dialogue:**
    - The ShowDialogueC coroutine is started.
    - The dialogue title is displayed.
    - The dialogue text is displayed character by character, with delays for readability.
    - If the dialogue text exceeds a maximum number of words (maxWords), a "moveNextButt" is activated, and the dialogue pauses until the player interacts with it.
    - Once the player interacts with the "moveNextButt", the next part of the dialogue is displayed.
3. **Ending Dialogue:** After all dialogue text has been displayed, the "moveNextButt" is activated, and the dialogue pauses until the player interacts with it. After the player input, the dialogue canvas is disabled, and the NPCManager is notified to move to the next dialogue, if any.

Using the Dialogue Manager

1. Adding the Script to Your Scene

- Create a Canvas in your Unity scene to serve as the dialogue UI.
- Create two TextMeshPro Text (TMP_Text) objects as children of the Canvas: one for the title and one for the dialogue content.
- Create a Button as a child of the Canvas to serve as the "moveNextButt".
- Create an empty GameObject in your scene and name it "DialogueManager".
- Attach the DialogueManager script to the "DialogueManager" GameObject.
- In the Inspector, assign the TMP_Text objects and the Button to the corresponding fields in the DialogueManager component.
- Disable the "moveNextButt" GameObject.

- Ensure the Canvas is enabled.

2. Setting up the NPC

- Create an NPC GameObject in your scene.
- Attach the NPCManager script to the NPC GameObject.
- Create a Scriptable Object for each dialogue and assign the dialogues to the NPCManager.

3. Customizing the Dialogue Manager

The DialogueManager script has several public variables that you can adjust in the Inspector:

- **title:** Assign the TMP_Text object that will display the dialogue title.
- **content:** Assign the TMP_Text object that will display the dialogue content.
- **characterDelay:** Adjust the delay (in seconds) between the display of each character in the dialogue text. A smaller value will make the text appear faster.
- **punctuationDelay:** Adjust the delay (in seconds) after punctuation marks (., ?, !).
- **maxWords:** Set the maximum number of words to display before pausing the dialogue and showing the "moveNextButt".
- **moveNextButt:** Assign the Button GameObject that the player will use to continue the dialogue.
- **dialogueCanvas:** Assign the Canvas GameObject that contains the dialogue UI.

4. Interacting with the Dialogue Manager

The DialogueManager interacts with other scripts, specifically the NPCManager. You don't directly call methods on the DialogueManager from other scripts. Instead, you trigger dialogue through the NPCManager, which then uses the DialogueManager.

Technical Details

- The ShowDialogueC method is a coroutine, which allows for pausing and resuming the execution of the dialogue display.
- The IsPunctuation method checks if a character is a punctuation mark (., ?, !).
- The moveNext variable is used to synchronize the dialogue flow with player input. When moveNext is true, the coroutine continues.

Script Reference

Dialogue Manager Script

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

```
using UnityEngine.TextCore.Text;
using UnityEngine.UI;

public class DialogueManager : MonoBehaviour
{
    public static DialogueManager instance;

    public TMP_Text title;
    public TMP_Text content;

    public float characterDelay = 0.1f;
    public float punctuationDelay = 0.5f;

    public int maxWords = 10;

    public bool moveNext;
    public GameObject moveNextButt;

    public Canvas dialogueCanvas;

    private void Awake()
    {
        instance = this;
    }

    public void MoveNext()
    {
        moveNext = true;
    }

    public void ShowDialogue(NPCManager nPCManager)
    {
        StartCoroutine(ShowDialogueC(nPCManager));
    }

    IEnumerator ShowDialogueC(NPCManager nPCManager)
    {
        title.text = nPCManager.dialogues[nPCManager.currentDialogueIndex].title;
        content.text = "";
```

```
        dialogueCanvas.enabled = true;

        foreach (char character in
nPCManager.dialogues[nPCManager.currentDialogueIndex].lines)
        {
            content.text += character;
            yield return new WaitForSeconds(IsPunctuation(character) ? punctuationDelay
: characterDelay);

            if (content.textInfo.wordCount >= maxWords && (character == ' ' ||
IsPunctuation(character) || character == ','))
            {
                moveNextButt.SetActive(true);
                yield return new WaitUntil(() => moveNext);

                content.text = "";
                moveNextButt.SetActive(false);
                moveNext = false;
            }
        }

        moveNextButt.SetActive(true);
        yield return new WaitUntil(() => moveNext);

        content.text = "";
        moveNextButt.SetActive(false);
        moveNext = false;

        dialogueCanvas.enabled = false;

        nPCManager.MoveNext();
    }

    bool IsPunctuation(char character)
    {
        return character == '.' || character == '?' || character == '!';
    }
}
```

NPC Dialogue Scriptable Object

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

[CreateAssetMenu(fileName = "NPCDialogueSO", menuName = "Simple Dialogue
System/NPC Dialogue")]
public class NPCDialogueSO : ScriptableObject
{
   public string title;
   public string lines;
}
```

Variables

- public TMP_Text title: The TextMeshPro Text object for the dialogue title.
- public TMP_Text content: The TextMeshPro Text object for the dialogue content.
- public float characterDelay: The delay between characters.
- public float punctuationDelay: The delay after punctuation.
- public int maxWords: Maximum words before showing "moveNextButt".
- public bool moveNext: A flag indicating whether to continue the dialogue.
- public GameObject moveNextButt: The "Move Next" button.
- public Canvas dialogueCanvas: The Canvas for the dialogue UI.

Methods

- public void MoveNext(): Sets the moveNext flag to true, indicating that the player wants to continue the dialogue. This method is typically called by the "moveNextButt" Button.
- public void ShowDialogue(NPCManager nPCManager): Starts the dialogue coroutine.
- IEnumerator ShowDialogueC(NPCManager nPCManager): Coroutine that displays the dialogue text.
- bool IsPunctuation(char character): Checks if a character is punctuation.