**How to Set Up an NPC Chatbot with Voice Using Unity and Google Cloud**

**Step 1: Create an NPC Model**

* Use **Ready Player Me** to create your NPC model. You can do this at: <https://readyplayer.me/avatar>
* Once created, download or copy the link to your model (make sure it ends with .glb), for example:  
  https://models.readyplayer.me/67b81eb6b16da2e12ef14c62.glb
* Import the .glb model into Unity by dragging it into your Assets folder.

**Step 2: Add Face Animation and Lip Sync**

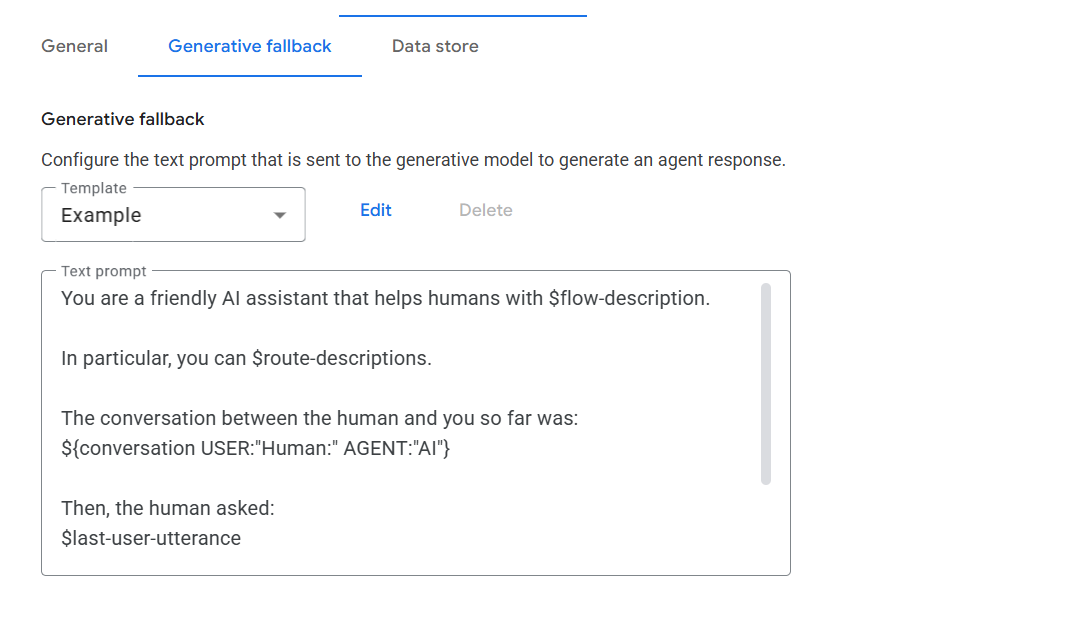
* Follow this tutorial video to add face animation and lip sync to your model in Unity:  
  <https://youtu.be/Q4sPGTVylnY>

**Step 3: Set Up Google Cloud Account**

* Create a Google Cloud account at: <https://cloud.google.com/>
* After creating the account, go to the **Google Cloud Console**:  
  https://console.cloud.google.com/

**Step 4: Create a Dialogflow CX Agent**

1. In the search bar, type and select **Agent Builder**.
2. Choose **Conversational Chatbot**.
3. Enter your agent details and create it.
4. Add your database and upload any necessary training files (intents, entities, etc.).
5. In the agent settings:
   * Go to **Generative AI** > **Generative Fallback**.
   * Add a prompt that reflects your NPC's personality and behavior.



**Step 5: Enable Required APIs**

In the Google Cloud Console:

1. Go to **Home**, then open the navigation menu and go to **APIs & Services** > **Enable APIs and Services**.
2. Enable the following:
   * **Speech-to-Text API**
   * **Text-to-Speech API**
   * **Dialogflow**

**Step 6: Set Up Service Account**

1. Go to **IAM & Admin** > **Service Accounts**.
2. Create a new service account.
3. After creating it, go to the **Keys** section and generate a new key.
4. Choose JSON format and download the file to your computer.

**Step 7: Get Access Token Using Google Cloud SDK**

1. Create a new folder on your computer.
2. Move the downloaded JSON file into this folder.
3. Open a terminal in this folder and set the environment variable:

$env:GOOGLE\_APPLICATION\_CREDENTIALS="path-to-your-json-file.json"

1. Download and extract the **Google Cloud SDK** into the same folder:  
   https://cloud.google.com/sdk/docs/install
2. Initialize the SDK:

./google-cloud-sdk/bin/gcloud init

1. Authenticate and print the access token:

./google-cloud-sdk/bin/gcloud auth application-default print-access-token

1. If you encounter metadata errors, use this command instead:

gcloud auth application-default login

1. Copy the access token and use it in your Unity scripts to authenticate requests to Dialogflow, Speech-to-Text, and Text-to-Speech APIs.

**Step 8:** Create a c sharp script attach it to the npc and copy this code and give all necessary credentials like accesstoken, projectid, agentid

using System;

using System.Text;

using UnityEngine;

using UnityEngine.Networking;

using UnityEngine.Windows.Speech;

using Newtonsoft.Json.Linq;

public class DialogflowChatbotM : MonoBehaviour

{

    public string projectId = "xxxxx";

    public string agentId = "xxxxxx";

    public string location = "global"; // Location of your agent

    public string languageCode = "en"; // Default language

    public string accessToken; // Use OAuth 2.0 token for secure authentication

    private string sessionId;

    private DictationRecognizer dictationRecognizer;

    private bool isDictating = false;

    private bool isPlayerInTrigger = false;

    private AudioSource audioSource;

    // Added headers

    private const string ContentTypeHeader = "application/json";

    private const string XGoogUserProjectHeader = "x-goog-user-project";

    private const string AuthorizationHeader = "Authorization";

    void Start()

    {

        sessionId = GenerateSessionId();

        audioSource = GetComponent<AudioSource>();

        // Initialize DictationRecognizer

        dictationRecognizer = new DictationRecognizer();

        dictationRecognizer.DictationResult += OnDictationResult;

        dictationRecognizer.DictationComplete += OnDictationComplete;

        dictationRecognizer.DictationError += OnDictationError;

        dictationRecognizer.InitialSilenceTimeoutSeconds = 10;

        // Pre-initialize the recognizer to reduce delay

        dictationRecognizer.Start();

        dictationRecognizer.Stop(); // Stop it immediately to keep it ready

    }

    private string GenerateSessionId()

    {

        return Guid.NewGuid().ToString();

    }

    private void Update()

    {

        // Check if the player is in the trigger area and presses/releases the Spacebar

        if (isPlayerInTrigger)

        {

            if (Input.GetKeyDown(KeyCode.Space))

            {

                StartDictation();

            }

            else if (Input.GetKeyUp(KeyCode.Space))

            {

                StopDictation();

            }

        }

    }

    private void OnTriggerEnter(Collider other)

    {

        if (other.CompareTag("Player"))

        {

            isPlayerInTrigger = true;

        }

    }

    private void OnTriggerExit(Collider other)

    {

        if (other.CompareTag("Player"))

        {

            isPlayerInTrigger = false;

            // Stop dictation and NPC speech when the player leaves the trigger area

            if (isDictating)

            {

                StopDictation();

            }

            if (audioSource.isPlaying)

            {

                audioSource.Stop();

            }

        }

    }

    private void StartDictation()

    {

        if (isDictating)

        {

            return;

        }

        isDictating = true;

        dictationRecognizer.Start();

    }

    private void StopDictation()

    {

        if (!isDictating || dictationRecognizer.Status != SpeechSystemStatus.Running)

        {

            return;

        }

        dictationRecognizer.Stop();

        isDictating = false;

    }

    private void OnDictationResult(string text, ConfidenceLevel confidence)

    {

        // Send the user's input to Dialogflow

        StartCoroutine(SendToDialogflow(text));

    }

    private void OnDictationComplete(DictationCompletionCause cause)

    {

        if (cause != DictationCompletionCause.Complete)

        {

            Debug.LogWarning("Dictation completed unsuccessfully: " + cause);

        }

        isDictating = false; // Ensure isDictating is reset

    }

    private void OnDictationError(string error, int hresult)

    {

        Debug.LogError("Dictation error: " + error);

        isDictating = false; // Ensure isDictating is reset

    }

    private IEnumerator SendToDialogflow(string userQuery)

    {

        string url = $"https://dialogflow.googleapis.com/v3/projects/{projectId}/locations/{location}/agents/{agentId}/sessions/{sessionId}:detectIntent";

        // Create JSON payload

        JObject jsonRequest = new JObject

        {

            ["queryInput"] = new JObject

            {

                ["languageCode"] = languageCode,

                ["text"] = new JObject

                {

                    ["text"] = userQuery,

                }

            }

        };

        UnityWebRequest request = new UnityWebRequest(url, "POST");

        byte[] bodyRaw = Encoding.UTF8.GetBytes(jsonRequest.ToString());

        request.uploadHandler = new UploadHandlerRaw(bodyRaw);

        request.downloadHandler = new DownloadHandlerBuffer();

        // Add headers

        request.SetRequestHeader("Content-Type", ContentTypeHeader);

        request.SetRequestHeader(AuthorizationHeader, $"Bearer {accessToken}");

        request.SetRequestHeader(XGoogUserProjectHeader, projectId);

        // Send request and yield

        yield return request.SendWebRequest();

        if (request.result == UnityWebRequest.Result.ConnectionError || request.result == UnityWebRequest.Result.ProtocolError)

        {

            Debug.LogError("Dialogflow Error: " + request.error);

        }

        else

        {

            string jsonResponse = request.downloadHandler.text;

            JObject responseObj = JObject.Parse(jsonResponse);

            string botResponse = responseObj["queryResult"]["responseMessages"][0]["text"]["text"][0]?.ToString();

            // Synthesize speech in the detected language

            StartCoroutine(SynthesizeSpeech(botResponse, languageCode));

        }

    }

    private IEnumerator SynthesizeSpeech(string text, string languageCode)

    {

        string voiceName = GetVoiceNameForLanguage(languageCode);

        // Create JSON request body for Google Text-to-Speech

        JObject jsonRequest = new JObject

        {

            ["input"] = new JObject { ["text"] = text },

            ["voice"] = new JObject { ["languageCode"] = languageCode, ["name"] = voiceName },

            ["audioConfig"] = new JObject { ["audioEncoding"] = "MP3" }

        };

        string url = "https://texttospeech.googleapis.com/v1/text:synthesize";

        UnityWebRequest request = new UnityWebRequest(url, "POST");

        byte[] bodyRaw = Encoding.UTF8.GetBytes(jsonRequest.ToString());

        request.uploadHandler = new UploadHandlerRaw(bodyRaw);

        request.downloadHandler = new DownloadHandlerBuffer();

        // Add headers

        request.SetRequestHeader("Content-Type", ContentTypeHeader);

        request.SetRequestHeader(AuthorizationHeader, $"Bearer {accessToken}");

        request.SetRequestHeader(XGoogUserProjectHeader, projectId);

        yield return request.SendWebRequest();

        if (request.result == UnityWebRequest.Result.ConnectionError || request.result == UnityWebRequest.Result.ProtocolError)

        {

            Debug.LogError("Text-to-Speech Error: " + request.error);

        }

        else

        {

            string jsonResponse = request.downloadHandler.text;

            JObject responseObj = JObject.Parse(jsonResponse);

            string audioContent = responseObj["audioContent"]?.ToString();

            byte[] audioBytes = Convert.FromBase64String(audioContent);

            PlayAudioClip(audioBytes); // Play the TTS response

        }

    }

    private string GetVoiceNameForLanguage(string languageCode)

    {

        string voiceName;

        switch (languageCode)

        {

            case "en":

                voiceName = "en-US-Standard-C"; // English (US)

                break;

            case "hi":

                voiceName = "hi-IN-Standard-C"; // Hindi (India)

                break;

            case "te":

                voiceName = "te-IN-Standard-C"; // Telugu (India)

                break;

            case "ta":

                voiceName = "ta-IN-Standard-A"; // Tamil (India)

                break;

            default:

                voiceName = "en-US-Standard-C"; // Default to English

                break;

        }

        return voiceName;

    }

    private void PlayAudioClip(byte[] audioBytes)

    {

        string path = Application.persistentDataPath + "/tts\_audio.mp3";

        System.IO.File.WriteAllBytes(path, audioBytes);

        StartCoroutine(LoadAndPlayAudio(path));

    }

    private IEnumerator LoadAndPlayAudio(string path)

    {

        using (UnityWebRequest www = UnityWebRequestMultimedia.GetAudioClip("file://" + path, AudioType.MPEG))

        {

            yield return www.SendWebRequest();

            if (www.result == UnityWebRequest.Result.ConnectionError || www.result == UnityWebRequest.Result.ProtocolError)

            {

                Debug.LogError("Audio Load Error: " + www.error);

            }

            else

            {

                AudioClip clip = DownloadHandlerAudioClip.GetContent(www);

                audioSource.clip = clip;

                audioSource.Play();

            }

        }

    }

}

Step 9: Install all necessary packages like newtonsoft, Enable unity dictation recogniser

Step 10: for npc in voice handler select audio clip, Attach the script (NPCChatbot) paste accesstoken in accesstoken token field  
Step 11: Run the Program

Cloning

Step 1: Download the zip file of Repository

Step2: Open in unity

Step3: in assets go to chatbot in that go to scenes and select NoonA

Step4: repeat step 3 to step 7 and get the access token projectid, and paste it in the fields in inspector tabb and run the program

Step5: Separate code for NPC-to-NPC communication is already attached to NPCs (paste the access code in the accesstoken fields in inspector tab)