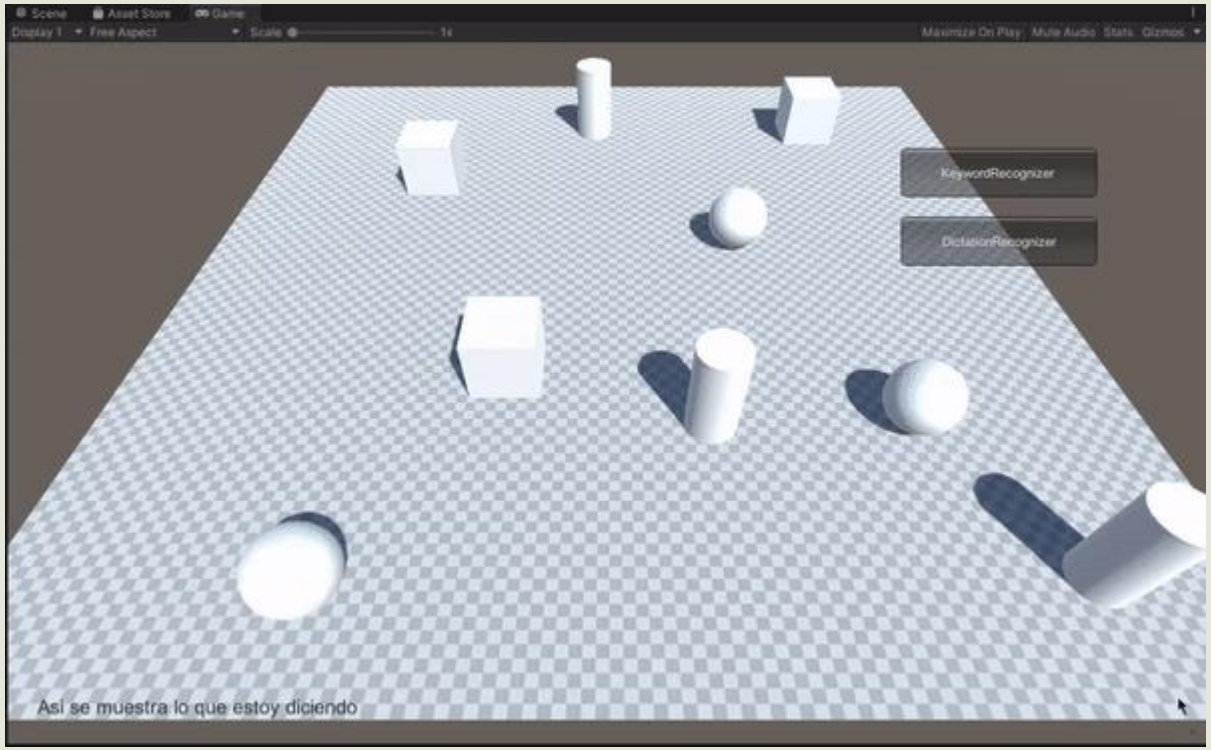
Reconocimiento de Voz

Entrega práctica

Resumen práctica

Se pide mostrar el funcionamiento de KeywordRecognizer, indicando qué palabras se espera capturar, y de DictationRecognizer, mostrando las frases devueltas por el objeto. Ambas clases no deben estar activas al mismo tiempo, sino que el usuario deberá poder escoger cuál quiere utilizar.

Ejecucion


KeywordRecognizer
DictationRecognizer
Así se muestra lo que estoy diciendo

KeywordRecognizer

```
using System;
using System.Text;
using UnityEngine;
using UnityEngine.Windows.Speech;

public class KeywordScript : MonoBehaviour
{
        private bool activated_ = false;

  [SerializeField]
  public string[] m_Keywords;

  private KeywordRecognizer m_Recognizer;

  void Start()
  {
        m_Keywords = new string[] {"Cubos", "Bolas", "Cilindros"};
    m_Recognizer = new KeywordRecognizer(m_Keywords);
    m_Recognizer.OnPhraseRecognized += OnPhraseRecognized;
  }

  private void OnPhraseRecognized(PhraseRecognizedEventArgs args)
  {
    StringBuilder builder = new StringBuilder();
    builder.AppendFormat("{0} ({1}){2}", args.text, args.confidence, Environment.NewLine);
    builder.AppendFormat("\tTimestamp: {0}{1}", args.phraseStartTime, Environment.NewLine);
    builder.AppendFormat("\tDuration: {0} seconds{1}", args.phraseDuration.TotalSeconds, Environment.NewLine)
    Debug.Log(builder.ToString());
    Accion(args.text);
    }
```

```
void OnGUI()
        {

                if(!activated_) // No esta en uso
                {
                        if(GUI.Button(new Rect(Screen.width/2+300, Screen.height/2-250, 200, 50), "KeywordRec
                        {
        m_Recognizer.Start();
        activated_ = true;
                        }
                }
                else
                {
                        if(GUI.Button(new Rect(Screen.width/2+300, Screen.height/2-250, 200, 50), "Stop Keywo
                        {
                                m_Recognizer.Stop();
                                activated_ = false;
                        }

                        GUI.Label(new Rect(Screen.width/2+340, Screen.height/2-200, 200, 50), "Listening...")
                }
        }

        private void OnDestroy()
{
  m_Recognizer.Dispose();
}

private void Accion (string word)
{

        switch (word)
        {
                case "Cubos":
                {
                        GameObject[] cubos;
        cubos = GameObject.FindGameObjectsWithTag("Cubo");

        foreach(GameObject cubo in cubos) {
          cubo.GetComponent<Renderer>().enabled=false;
        }
                }
                break;

                case "Bolas":
                {
                        GameObject[] bolas;
        bolas = GameObject.FindGameObjectsWithTag("Bola");

        foreach(GameObject bola in bolas) {
          bola.GetComponent<Renderer>().enabled=false;
        }
                }
                break;

                case "Cilindros":
                {
                        GameObject[] cilindros;
        cilindros = GameObject.FindGameObjectsWithTag("Cilindro");

        foreach(GameObject cilindro in cilindros) {
          cilindro.GetComponent<Renderer>().enabled=false;
        }
                }
                break;

                default:
                break;
        }
```

DictationRecognizer

```
public class DictationScript : MonoBehaviour
{
                private bool activated_ = false;

    [SerializeField]
    private Text m_Hypotheses;

    [SerializeField]
    private Text m_Recognitions;

    private DictationRecognizer m_DictationRecognizer;

    void Start()
    {
        m_DictationRecognizer = new DictationRecognizer();

        m_DictationRecognizer.DictationResult += (text, confidence) =>
        {
            Debug.LogFormat("Dictation result: {0}", text);
            m_Recognitions.text += text + "\n";
        };

        m_DictationRecognizer.DictationHypothesis += (text) =>
        {
            Debug.LogFormat("Dictation hypothesis: {0}", text);
            m_Hypotheses.text += text;
        };

        m_DictationRecognizer.DictationComplete += (completionCause) =>
        {
            if (completionCause != DictationCompletionCause.Complete)
                Debug.LogErrorFormat("Dictation completed unsuccessfully: {0}.", completionCause);
        };

        m_DictationRecognizer.DictationError += (error, hresult) =>
        {
            Debug.LogErrorFormat("Dictation error: {0}; HResult = {1}.", error, hresult);
        };

    }

    void OnDestroy()
    {
        m_DictationRecognizer.Dispose();
    }

  void OnGUI()
        {

                if(!activated_) // No esta en uso
                {
                        if(GUI.Button(new Rect(Screen.width/2+300, Screen.height/2-180, 200, 50), "DictationR
                        {
        m_DictationRecognizer.Start();
        activated_ = true;
                        }
                }
                else
                {
                        if(GUI.Button(new Rect(Screen.width/2+300, Screen.height/2-180, 200, 50), "Stop Dicta
                        {
                                m_DictationRecognizer.Stop();
                                activated_ = false;
                        }

                        GUI.Label(new Rect(Screen.width/2+340, Screen.height/2-130, 200, 50), "Listening...")
                }
        }
}
```