##### AUGMENTED REALITY SETUP WITH MOBILE PHONES FOR REALTIME OBJECT RECOGNITION

##### SUMMER INTERNSHIP PROJECT CODE

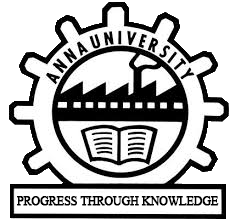
###### ***Submitted by***

**S SASHANK (2016103060)**

**APARAJIT K R (2016103505)**

**S HARINAARAAYAN (2016103029)**

**COLLEGE OF ENGINEERING GUINDY**



**ANNA UNIVERSITY: CHENNAI 600 025**

**MAY-JUNE 2018**

using UnityEngine;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using Vuforia;

using System.IO;

using System.Text.RegularExpressions;

using Newtonsoft.Json.Linq;

using IBM.Watson.DeveloperCloud.Utilities;

using IBM.Watson.DeveloperCloud.Services.Assistant.v1;

using IBM.Watson.DeveloperCloud.Services.TextToSpeech.v1;

using IBM.Watson.DeveloperCloud.Logging;

using IBM.Watson.DeveloperCloud.Connection;

using UnityEngine.UI;

public class UDTEventHandler : MonoBehaviour, IUserDefinedTargetEventHandler

{

#region PUBLIC\_MEMBERS

//To Define the image target

public ImageTargetBehaviour ImageTargetTemplate;

// To Define the 3D word Model

public GameObject textobj;

// To define the Sound Player

public GameObject musicobj;

public int LastTargetIndex

{

get { return (m\_TargetCounter - 1) % MAX\_TARGETS; }

}

#endregion PUBLIC\_MEMBERS

//To define the byte array of the image captured

public byte[] imageByteArray;

// Google Reverse Search Url

private const string BASE\_URL = "http://www.google.com/searchbyimage?hl=ru&image\_url=";

// To save the url returned by Cloudinary

private string imageURl;

// To save the phrase that needs to be searched for

private string wordsToSearch;

// Cloudinary Credentials

private const string CLOUD\_NAME = "dylrioik3";

private const string UPLOAD\_PRESET\_NAME = "qylb43yg";

private const string CLOUDINARY\_API\_KEY = "284551392584861";

private const string CLOUDINARY\_SIGNATURE = "K5IE-CzFZS5HCZw\_DHz5G6MVQVo";

// Google Custom Search API Key

private const string GOOGLE\_API\_KEY = "AIzaSyD7SHc\_jTdUmsYteNArB2f7ME9LXzoTM-g";

// Google Custom Search Engine Id

private const string GOOGLE\_CUSTOM\_ENGINE\_ID = "002966606582515264909:wuqx1wxqewe";

// Google Custom search

private const string GOOGLE\_SEARCH\_URL = "https://www.googleapis.com/customsearch/v1?cx=" +

        GOOGLE\_CUSTOM\_ENGINE\_ID+"&key="+GOOGLE\_API\_KEY+"&cref&q=";

// Oxford Dictionary API key and Credentials

private const string OXFORD\_API\_KEY = "63c0e519bf3923479a4f666f1d27e947";

     private const string OXFORD\_APP\_ID = "d866d9df";

private const string OXFORD\_SEACRH\_URL = "https://od-api.oxforddictionaries.com/api/v1/entries/en/{0}/definitions";

// To display the status

public GameObject status;

public string txt;

// To add text to the 3D word Model

TextMesh tm;

#region PRIVATE\_MEMBERS

const int MAX\_TARGETS = 5;

UserDefinedTargetBuildingBehaviour m\_TargetBuildingBehaviour;

QualityDialog m\_QualityDialog;

ObjectTracker m\_ObjectTracker;

TrackableSettings m\_TrackableSettings;

FrameQualityMeter m\_FrameQualityMeter;

// DataSet that newly defined targets are added to

DataSet m\_UDT\_DataSet;

// Currently observed frame quality

ImageTargetBuilder.FrameQuality m\_FrameQuality = ImageTargetBuilder.FrameQuality.FRAME\_QUALITY\_NONE;

// Counter used to name newly created targets

int m\_TargetCounter;

#endregion //PRIVATE\_MEMBERS

#region MONOBEHAVIOUR\_METHODS

void Start()

{

// Set the Target Frame

m\_TargetBuildingBehaviour = GetComponent<UserDefinedTargetBuildingBehaviour>();

if (m\_TargetBuildingBehaviour)

{

m\_TargetBuildingBehaviour.RegisterEventHandler(this);

Debug.Log("Registering User Defined Target event handler.");

}

// Initialise Frame Quality And Tracking Settings

m\_FrameQualityMeter = FindObjectOfType<FrameQualityMeter>();

m\_TrackableSettings = FindObjectOfType<TrackableSettings>();

m\_QualityDialog = FindObjectOfType<QualityDialog>();

if (m\_QualityDialog)

{

m\_QualityDialog.GetComponent<CanvasGroup>().alpha = 0;

}

// Find 3D Word Model

textobj = GameObject.Find("wordobj");

// Find Audio Player

musicobj = GameObject.Find("music");

// Find the Status Displayer

status = GameObject.Find("status");

}

#endregion //MONOBEHAVIOUR\_METHODS

#region IUserDefinedTargetEventHandler Implementation

/// Called when UserDefinedTargetBuildingBehaviour has been initialized successfully

public void OnInitialized()

{

// Initialise the Tracking of Frames

m\_ObjectTracker = TrackerManager.Instance.GetTracker<ObjectTracker>();

if (m\_ObjectTracker != null)

{

// Create a new dataset

m\_UDT\_DataSet = m\_ObjectTracker.CreateDataSet();

m\_ObjectTracker.ActivateDataSet(m\_UDT\_DataSet);

}

}

/// Updates the current frame quality

public void OnFrameQualityChanged(ImageTargetBuilder.FrameQuality frameQuality)

{

Debug.Log("Frame quality changed: " + frameQuality.ToString());

m\_FrameQuality = frameQuality;

if (m\_FrameQuality == ImageTargetBuilder.FrameQuality.FRAME\_QUALITY\_LOW)

{

Debug.Log("Low camera image quality");

}

m\_FrameQualityMeter.SetQuality(frameQuality);

}

/// Takes a new trackable source and adds it to the dataset

/// This gets called automatically as soon as you 'BuildNewTarget with UserDefinedTargetBuildingBehaviour

public void OnNewTrackableSource(TrackableSource trackableSource)

{

m\_TargetCounter++;

// Deactivates the dataset first

m\_ObjectTracker.DeactivateDataSet(m\_UDT\_DataSet);

// Destroy the oldest target if the dataset is full or the dataset

// already contains five user-defined targets.

if (m\_UDT\_DataSet.HasReachedTrackableLimit() || m\_UDT\_DataSet.GetTrackables().Count() >= MAX\_TARGETS)

{

IEnumerable<Trackable> trackables = m\_UDT\_DataSet.GetTrackables();

Trackable oldest = null;

foreach (Trackable trackable in trackables)

{

if (oldest == null || trackable.ID < oldest.ID)

oldest = trackable;

}

if (oldest != null)

{

Debug.Log("Destroying oldest trackable in UDT dataset: " + oldest.Name);

m\_UDT\_DataSet.Destroy(oldest, true);

}

}

// Get predefined trackable and instantiate it

ImageTargetBehaviour imageTargetCopy = Instantiate(ImageTargetTemplate);

imageTargetCopy.gameObject.name = "UserDefinedTarget-" + m\_TargetCounter;

// Add the duplicated trackable to the data set and activate it

m\_UDT\_DataSet.CreateTrackable(trackableSource, imageTargetCopy.gameObject);

// Activate the dataset again

m\_ObjectTracker.ActivateDataSet(m\_UDT\_DataSet);

// Extended Tracking with user defined targets only works with the most recently defined target.

// If tracking is enabled on previous target, it will not work on newly defined target.

// Don't need to call this if you don't care about extended tracking.

StopExtendedTracking();

m\_ObjectTracker.Stop();

m\_ObjectTracker.ResetExtendedTracking();

m\_ObjectTracker.Start();

// Make sure TargetBuildingBehaviour keeps scanning...

m\_TargetBuildingBehaviour.StartScanning();

}

#endregion IUserDefinedTargetEventHandler implementation

#region PUBLIC\_METHODS

/// Instantiates a new user-defined target and is also responsible for dispatching callback to

/// IUserDefinedTargetEventHandler::OnNewTrackableSource

public void BuildNewTarget()

{

if (m\_FrameQuality == ImageTargetBuilder.FrameQuality.FRAME\_QUALITY\_MEDIUM ||

m\_FrameQuality == ImageTargetBuilder.FrameQuality.FRAME\_QUALITY\_HIGH)

{

// create the name of the next target.

// the TrackableName of the original, linked ImageTargetBehaviour is extended with a continuous number to ensure unique names

string targetName = string.Format("{0}-{1}", ImageTargetTemplate.TrackableName, m\_TargetCounter);

// generate a new target:

m\_TargetBuildingBehaviour.BuildNewTarget(targetName, ImageTargetTemplate.GetSize().x);

}

else

{

Debug.Log("Cannot build new target, due to poor camera image quality");

if (m\_QualityDialog)

{

StopAllCoroutines();

m\_QualityDialog.GetComponent<CanvasGroup>().alpha = 1;

StartCoroutine(FadeOutQualityDialog());

}

}

StartCoroutine("TakePic");

}

///Capture Image for Reverse Image Search

IEnumerator TakePic()

{

string filePath;

GameObject button = GameObject.Find("BuildButton");

button.SetActive(false);

GameObject Meter = GameObject.Find("QualityMeter");

Meter.SetActive(false);

if (Application.isMobilePlatform) {

                filePath = Application.persistentDataPath + "/image.png";

                ScreenCapture.CaptureScreenshot ("/image.png");

                //must delay here so picture has time to save unfortunatly

                yield return new WaitForSeconds(1.5f);

                //Encode to a PNG

                imageByteArray = File.ReadAllBytes(filePath);

print("\*\*\*\*\*\*\*\*\*\*Photo Done\*\*\*\*\*\*\*\*\*\*\*");

status.GetComponent<Text>().text +="photo taken\n";

} else {

                filePath = Application.dataPath + "/Images/" + "image.png";

                ScreenCapture.CaptureScreenshot (filePath);

                //must delay here so picture has time to save unfortunatly

                yield return new WaitForSeconds(1.5f);

                //Encode to a PNG

                imageByteArray = File.ReadAllBytes(filePath);

print("\*\*\*\*\*\*\*\*\*\*Photo Done\*\*\*\*\*\*\*\*\*\*\*");

status.GetComponent<Text>().text +="photo taken\n";

            }

button.SetActive(true);

Meter.SetActive(true);

StartCoroutine("UploadImage");

}

///Upload Image to Cloudinary

IEnumerator UploadImage(){

print ("uploading image...");

status.GetComponent<Text>().text +="uploading to cloud\n";

         string url = "https://api.cloudinary.com/v1\_1/" + CLOUD\_NAME + "/auto/upload/";

         WWWForm myForm = new WWWForm ();

         myForm.AddBinaryData ("file",imageByteArray);

        myForm.AddField ("upload\_preset", UPLOAD\_PRESET\_NAME);

        WWW www = new WWW(url,myForm);

        yield return www;

        print ([www.text](http://www.text));

// Parse Text to get the image URL

imageURl = www.text.Split('"', '"')[43];

        print ("IMAGE URL: " + imageURl);

status.GetComponent<Text>().text +="uploaded to cloud\n";

StartCoroutine ("reverseImageSearch");

}

/// Google Reverse Image Search

IEnumerator reverseImageSearch(){

status.GetComponent<Text>().text +="reverse image search\n";

string fullSearchURL = BASE\_URL + WWW.EscapeURL(imageURl);

       print (fullSearchURL);

WWWForm form = new WWWForm ();

var headers = form.headers;

// To access the URL like a browser

      headers ["User-Agent"] = "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36";

      WWW www = new WWW (fullSearchURL, null, headers);

      //create a new www object and pass in this search url

      yield return www;

      string response = www.text;

      print(response);

      Match m = Regex.Match (response, "style=\"font-style:italic\">(.\*?(?=<))");

      wordsToSearch = m.Groups [1].Value;

      print (wordsToSearch);

status.SetActive(false);

textobj.GetComponent<TextMesh>().text=wordsToSearch;

textobj.GetComponent<MeshRenderer>().enabled = true;

StartCoroutine("SearchWeb");

}

public string definition { get; set; }

// Google Custom Search

public IEnumerator SearchWeb()

{

string searchURL = GOOGLE\_SEARCH\_URL + wordsToSearch;

WWW www = new WWW(searchURL);

yield return www;

string result = www.text;

print(result);

string definition = "";

// Using Regex find the Wikipedia link and its text

Regex regex = new Regex("Wikipedia");

Match match = regex.Match(result);

if (match.Index != 0)

{

regex = new Regex("snippet\": \"(.\*?(?=\\.))", RegexOptions.Singleline);

match = regex.Match(result, match.Index);

definition = match.Groups[1].Value;

}

print(definition);

// If there is no Wikipedia link present start Oxford Dictionary

if ("".Equals(definition) || definition == null)

{

StartCoroutine("OxfordAPI");

}

else

{

print("google");

yield return new WaitForSeconds(1.5f);

if(txt!="")

{

print("Caption" + "\n\n" + txt);

}

StartCoroutine(play(wordsToSearch));

int len = wordsToSearch.Length;

yield return new WaitForSeconds(2.0f);

len += definition.Length;

print(definition);

StartCoroutine(play(definition));

yield return new WaitForSeconds(0.1f \* len);

}

}

//Oxford Dictionary Search

public IEnumerator OxfordAPI(){

string words = wordsToSearch.Replace (" ", "\_").ToLower();

print("Searching for meaning...");

string url = String.Format(OXFORD\_SEACRH\_URL, words);

        var headers = new Dictionary<String, String>();

        headers ["app\_id"] = OXFORD\_APP\_ID;

        headers ["app\_key"] = OXFORD\_API\_KEY;

        headers ["Accept"] = "application/json";

        WWW www = new WWW (url, null, headers);

        yield return(www);

        string result = www.text;

        Match m = Regex.Match(result, "definitions\":.\*?(?=\")\"(.\*?(?=\"))", RegexOptions.Singleline);

        definition = m.Groups[1].Value;

        print(definition);

print("Definition found!!");

yield return new WaitForSeconds(1.5f);

print(wordsToSearch + "\n\n");

StartCoroutine(play(wordsToSearch));

yield return new WaitForSeconds(2.0f);

 print(definition);

StartCoroutine(play(definition));

int len = wordsToSearch.Length;

len += definition.Length;

yield return new WaitForSeconds(0.1f \* len);

}

// Play the word and its definition

IEnumerator play(string word)

{

Credentials cred = new Credentials("9b54aa5a-221a-46ad-9a20-9a26cb0f34fb","reGOv0w75InZ","https://stream.watsonplatform.net/text-to-speech/api");

Assistant assist = new Assistant(cred);

TextToSpeech tts = new TextToSpeech (cred);

tts.Voice = VoiceType.en\_US\_Lisa;

AudioClip au;

tts.ToSpeech(OnSynthesize,OnFail,word);

yield return new WaitForSeconds(0.5f\*word.Length);

}

private void OnFail(RESTConnector.Error error, Dictionary<string, object> customData)

{

Log.Error("ExampleTextToSpeech.OnFail()", "Error received: {0}", error.ToString());

}

private void OnSynthesize(AudioClip clip, Dictionary<string, object> customData)

{

PlayClip(clip);

}

private void PlayClip(AudioClip clip)

{

musicobj.GetComponent<AudioSource>().clip = clip;

musicobj.GetComponent<AudioSource>().Play();

}

#endregion //PUBLIC\_METHODS

#region PRIVATE\_METHODS

// Fade Out Effect for Messages Displayed

IEnumerator FadeOutQualityDialog()

{

yield return new WaitForSeconds(1f);

CanvasGroup canvasGroup = m\_QualityDialog.GetComponent<CanvasGroup>();

for (float f = 1f; f >= 0; f -= 0.1f)

{

f = (float)Math.Round(f, 1);

Debug.Log("FadeOut: " + f);

canvasGroup.alpha = (float)Math.Round(f, 1);

yield return null;

}

}