**Coding Documentation**

**Backend code**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System;

using UnityEngine.UI;

using UnityEngine.Android;

**Declaring the various output area needed in the unity for showing the results on the screen.**

public class Display\_data : MonoBehaviour

{

public static int waterLevel = 20;

public Text outputArea;

public Text gpsOut;

int tableSize;

public bool isUpdating;

public TextAsset textAssetData;

public static int getWaterLevel()

{

return waterLevel;

}

**Creating the various columns according to the csv so the code can easily read the csv file.**

public class player

{

public string pointid;

public float grid\_code\_1;

public float X;

public float Y;

public float grid\_code\_5;

}

[System.Serializable]

public class PlayerList

{

public player[] player;

}

public PlayerList myPlayerList = new PlayerList();

**Introducing precision in all the values of coordinates as well as of altitude.**

public string compareData(string lat, string lon, float alt)

{

int precise, precise2;

if (lat.Substring(0, 1) == "-")

{

precise = 6;

}

else if (lat.Substring(0, 1) == "0")

{

precise = 1;

}

else

{

precise = 5;

}

if (lon.Substring(0, 1) == "-")

{

precise2 = 6;

}

else if (lon.Substring(0, 1) == "0")

{

precise2 = 1;

}

else

{

precise2 = 5;

}

for (int i = 0; i < tableSize; i++)

{

string datalat = myPlayerList.player[i].Y.ToString();

string datalon = myPlayerList.player[i].X.ToString();

**Comparing the water level output with two reference one with 1.5 degree and another one with 5 degree so if else loops are used here to switch between these two cases.**

if (string.Equals(datalat, lat.Substring(0, precise)) && string.Equals(datalon, lon.Substring(0, precise2)))

{

int conv1 = watersc.conv1;

string level = "";

if (conv1 == 1)

{

level = "Water Level : " + myPlayerList.player[i].grid\_code\_1.ToString();

if (alt < 40)

{

waterLevel = (int)myPlayerList.player[i].grid\_code\_1;

}

else

{

waterLevel = -10;

}

}

else

{

level = "Water Level : " + myPlayerList.player[i].grid\_code\_5.ToString();

if (alt < 40)

{

waterLevel = (int)myPlayerList.player[i].grid\_code\_5;

}

else

{

waterLevel = -10;

}

}

return level.Substring(0, 19);

}

}

return "Water Level : 0 ";

}

void Start()

{

ReadCSV();

}

void ReadCSV()

{

string[] data = textAssetData.text.Split(new string[] { ",", "\n" }, StringSplitOptions.None);

tableSize = data.Length / 5 - 1;

string myString = Convert.ToString(tableSize);

myPlayerList.player = new player[tableSize];

for (int i = 0; i < tableSize; i++)

{

myPlayerList.player[i] = new player();

myPlayerList.player[i].pointid = (data[5 \* (i + 1)]);

myPlayerList.player[i].grid\_code\_1 = float.Parse(data[5 \* (i + 1) + 1]);

myPlayerList.player[i].X = float.Parse(data[5 \* (i + 1) + 2]);

myPlayerList.player[i].Y = float.Parse(data[5 \* (i + 1) + 3]);

myPlayerList.player[i].grid\_code\_5 = float.Parse(data[5 \* (i + 1) + 4]);

}

}

**This IEnumerator GetLocation() function is used for accessing the particular coordinates of the device along with the altitude of that place.**

private void Update()

{

if (!isUpdating)

{

StartCoroutine(GetLocation());

isUpdating = !isUpdating;

}

}

IEnumerator GetLocation()

{

if (!Permission.HasUserAuthorizedPermission(Permission.FineLocation))

{

Permission.RequestUserPermission(Permission.FineLocation);

Permission.RequestUserPermission(Permission.CoarseLocation);

}

**// First, check if user has location service enabled**

if (!Input.location.isEnabledByUser)

yield return new WaitForSeconds(10);

**// Start service before querying location**

Input.location.Start();

**// Wait until service initializes**

int maxWait = 10;

while (Input.location.status == LocationServiceStatus.Initializing && maxWait > 0)

{

yield return new WaitForSeconds(1);

maxWait--;

}

**// Service didn't initialize in 20 seconds**

if (maxWait < 1)

{

outputArea.text = "Timed out";

print("Timed out");

yield break;

}

**// Connection has failed**

if (Input.location.status == LocationServiceStatus.Failed)

**The information is stored in the various output are created in order to display it in the Unity interface.**

{

outputArea.text = "Unable to determine device location";

print("Unable to determine device location");

yield break;

}

else

{

outputArea.text = compareData(Input.location.lastData.latitude.ToString(), Input.location.lastData.longitude.ToString(), Input.location.lastData.altitude);

gpsOut.text = "Location: " + Input.location.lastData.latitude + " " + Input.location.lastData.longitude + " " + Input.location.lastData.altitude;

}

**// Stop service if there is no need to query location updates continuously**

isUpdating = !isUpdating;

Input.location.Stop();

}

}

**Water level animation adjustment according to the given water level on the screen.**

using System.Collections;

using System.Collections.Generic;

using System.Numerics;

using UnityEngine;

using UnityEngine.UI;

public class sea : MonoBehaviour

{

public int level = Display\_data.getWaterLevel();

void Start()

{

transform.position = new Vector3(0, Display\_data.getWaterLevel(), 0);

}

void Update()

{

transform.position = new Vector3(0, Display\_data.getWaterLevel(), 0);

}

}

**UI code**

**This chunk of code is used for managing accounts for creating and signing pages.**

using System.Collections;

using System.Collections.Generic;

using PlayFab;

using PlayFab.ClientModels;

using UnityEngine;

using UnityEngine.Events;

public class UserAccountManager : MonoBehaviour

{

public static UserAccountManager Instance;

public static UnityEvent OnSignInSuccess = new UnityEvent();

public static UnityEvent<string> OnSignInFailed = new UnityEvent<string>();

public static UnityEvent<string> OnCreateAccountFailed = new UnityEvent<string>();

void Awake()

{

Instance = this;

}

**This chunk of code is used for creating a “sign-in” option in Ui interface.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class UISIngIN : MonoBehaviour

{

[SerializeField] Text errorText;

[SerializeField] Canvas canvas;

string username, password;

void OnEnable()

{

UserAccountManager.OnSignInFailed.AddListener(OnSignInFailed);

UserAccountManager.OnSignInSuccess.AddListener(OnSignInSuccess);

}

void OnDisable()

{

UserAccountManager.OnSignInFailed.RemoveListener(OnSignInFailed);

UserAccountManager.OnSignInSuccess.RemoveListener(OnSignInSuccess);

}

void OnSignInFailed(string error)

{

errorText.gameObject.SetActive(true);

errorText.text = error;

}

void OnSignInSuccess()

{

canvas.enabled = false

}

public void UpadteUsername(string \_username)

{

username = \_username;

}

public void UpadtePassword(string \_password)

{

password = \_password;

}

public void SignIn()

{

UserAccountManager.Instance.SignIn(username, password);

}

}

**This chunk of code is used for “creating an account” option for the user trying to login in the application**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class UICreateAccount : MonoBehaviour

{

[SerializeField] Text errorText;

[SerializeField] Canvas canvas;

string username, password, emailAddress;

void OnEnable()

{

UserAccountManager.OnCreateAccountFailed.AddListener(OnCreateAccountFailed);

UserAccountManager.OnSignInSuccess.AddListener(OnSignInSuccess);

}

void OnDisable()

{

UserAccountManager.OnCreateAccountFailed.RemoveListener(OnCreateAccountFailed);

UserAccountManager.OnSignInSuccess.RemoveListener(OnSignInSuccess);

}

void OnCreateAccountFailed (string error)

{

errorText.gameObject.SetActive(true);

errorText.text = error;

}

void OnSignInSuccess()

{

canvas.enabled = false;

}

public void UpadteUsername(string \_username)

{

username = \_username;

}

public void UpadtePassword(string \_password)

{

password = \_password;

}

public void UpadteEmailAdress(string \_emailAddress)

{

emailAddress = \_emailAddress;

}

public void CreateAccount()

{

UserAccountManager.Instance.CreateAccount(username, emailAddress, password);

}

}

**This is the code for taking screenshot of the live camera image along with hashtag**

using System.Collections;

using System.Collections.Generic;

//using System.Numerics;

using UnityEngine;

public class TakeScreenshot : MonoBehaviour

{

[SerializeField]

GameObject blink;

public void TakeAShot()

{

StartCoroutine("CaptureIt");

}

IEnumerator CaptureIt()

{

string timeStamp = System.DateTime.Now.ToString("dd-MM-yyyy-HH-mm-ss");

string fileName = "Screenshot" + timeStamp + ".png";

string pathToSave = fileName;

ScreenCapture.CaptureScreenshot(pathToSave);

yield return new WaitForEndOfFrame();

Instantiate(blink, new Vector2(0f, 0f), Quaternion.identity);

}

}

**The Following code is used for calculating the carbon footprint.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.Networking;

public class carbon1 : MonoBehaviour

{

public Text outputArea;

public ToggleGroup choice1;

public ToggleGroup choice2;

public ToggleGroup choice3;

private string Option1;

private string Option2;

private string Option3;

public int carbon;

public void Submit()

{

Option1 = choice1.GetFirstActiveToggle().transform.GetChild(1).GetComponent<Text>().text;

}

public void Send()

{

Option2 = choice2.GetFirstActiveToggle().transform.GetChild(1).GetComponent<Text>().text;

}

public void Result()

{

Option3 = choice3.GetFirstActiveToggle().transform.GetChild(1).GetComponent<Text>().text;

float final;

int conv1 = int.Parse(Option1);

int conv2 = int.Parse(Option2);

int conv3 = int.Parse(Option3);

carbon = (conv1 + conv2 + conv3);

final = carbon / 3;

outputArea.text = final.ToString();

}

}

**This chunk of code is used multiple times for switching between different scenes.**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class way : MonoBehaviour

{

public void LoadScene(string sceneName)

{

SceneManager.LoadScene(sceneName);

}

}

**The following code is used for transferring the survey form data to google sheets.**

using System.Collections;

using System.Collections.Generic;

//using System.Diagnostics;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.Networking;

public class Endform1 : MonoBehaviour

{

public ToggleGroup answer;

private string Answer;

private string URL = "https://docs.google.com/forms/u/0/d/e/1FAIpQLScFwOSQ6yoTNW6MDngX8284dVmGOv\_yOkZMeOQSwXx6EjajuA/formResponse";

public void Submit()

{

Answer = answer.GetFirstActiveToggle().transform.GetChild(1).GetComponent<Text>().text;

StartCoroutine(Post(Answer));

Debug.Log(Answer);

}

IEnumerator Post(string answer)

{

Debug.Log("Post Started");

WWWForm form = new WWWForm();

form.AddField("entry.1842610452", answer);

UnityWebRequest WWW = UnityWebRequest.Post(URL, form);

yield return WWW.SendWebRequest();

}

}