DIRECTED RESEARCH REPORT

Submitted by:
Akash Mohan Das
Graduate Student
Department of Electrical Engineering
Viterbi School of Engineering
USC

Under the Guidance of :
Dr. David Krum
Associate Director
ICT Mixed Reality Lab

Abstract :

The research was aimed at understanding the basic concepts of Unity, Steam VR using HTC Vive. The work done so far was to get acquainted with the VR environment in Unity. The work so far can be divided as :

(i) Playing around with the Unity GameObject using HTC controller to pick up the objects using Raycast.
(ii) Scatter plot of iris Data using Unity scripts.

WORK :

(i) I got started with Unity and C# scripting. The first task was to cast a ray from the HTC vive controller and pick up the object that is closest using the trigger button. The object is brought close to the controller for inspection. When, trigger button was released, the object goes back to its original location.

For this task, several objects were placed in the Scene. SteamVR package was downloaded and imported from the asset store. The following code was added to the left and right controllers :

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Controller : MonoBehaviour {
public SteamVR_Controller.Device controller
    {
        get
        {
            return SteamVR_Controller.Input((int)GetComponent<SteamVR_TrackedObject>().index);
        }
    }
}
```

Another script named HeldObject was scripted and added to all the game objects in the scene:

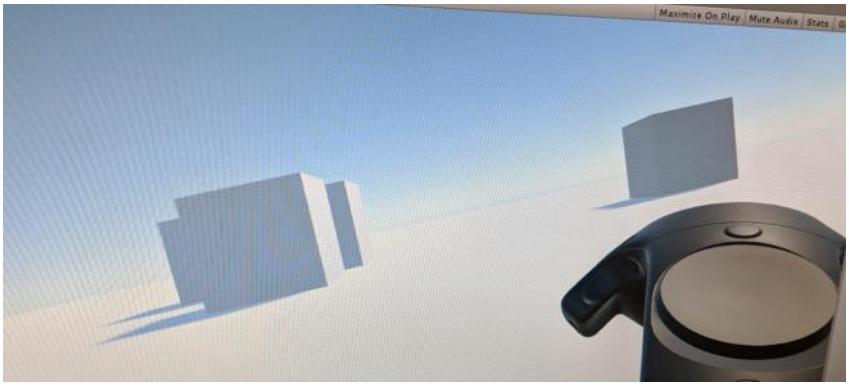
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
[RequireComponent(typeof(Controller))]
public class Raycast : MonoBehaviour
{
    GameObject heldObject;
    Controller controller;
Rigidbody simulator;
    public double distance;
    private LineRenderer laserline;
    public Transform laserend;

//public Valve.VR.EVRButtonId pickUpButton;
    //public Valve.VR.EVRButtonId dropButton;
    void Start()
    {
        simulator = new GameObject().AddComponent<Rigidbody>();
        simulator.name = "simulator";
        simulator.transform.parent = transform.parent;
        controller = GetComponent<Controller>();
        laserline = GetComponent<LineRenderer>();
```

```
    }
void Update()
    {

        if (heldObject)
        {
            simulator.velocity = (transform.position - simulator.position) * 50f;
            if
(controller.controller.GetPressUp(Valve.VR.EVRButtonId.k_EButton_SteamVR_Trigger))
            {
                heldObject.transform.parent = null;
                heldObject.GetComponent<Rigidbody>().isKinematic = false;
                heldObject.GetComponent<Rigidbody>().velocity = simulator.velocity;
                heldObject.GetComponent<HeldObject>().parent = null;
                //heldObject.GetComponent < HeldObject>().hideLines();
                heldObject = null;
            }
        }
        else
        {
            if
(controller.controller.GetPressDown(Valve.VR.EVRButtonId.k_EButton_SteamVR_Trigger))
            {
                //laserline.enabled = true;
                laserline.SetPosition(0, controller.transform.position);
                RaycastHit[] cols =
Physics.RaycastAll(controller.transform.position,controller.transform.forward, 20.0f);
                Debug.Log(cols.Length);
if (cols.Length > 0)
                {
                    Debug.Log("collider hit");
                    laserline.SetPosition(1, cols[0].point);
                    Collider col = cols[0].collider;

                    distance = cols[0].distance;

for (int i = 0; i < cols.Length; i++)
                    {
                        Debug.Log(cols[i].distance);
if (cols[i].distance < distance)
                        {
                            col = cols[i].collider;
                            distance = cols[i].distance;

                        }
                    }
if (heldObject == null && col.GetComponent<HeldObject>() && col.GetComponent<HeldObject>
().parent == null)
                    {
                        heldObject = col.gameObject;
                        heldObject.transform.parent = transform;
                        heldObject.transform.localPosition = Vector3.zero;
                        heldObject.transform.localRotation = Quaternion.identity;
                        heldObject.GetComponent<Rigidbody>().isKinematic = true;
                        heldObject.GetComponent<HeldObject>().parent = controller;
                        //heldObject.GetComponent<HeldObject>().showLines();
                    }
}
                else
                {
                    laserline.SetPosition(1, controller.transform.forward * 20.0f);
                }
            }
        }
    }
}
```

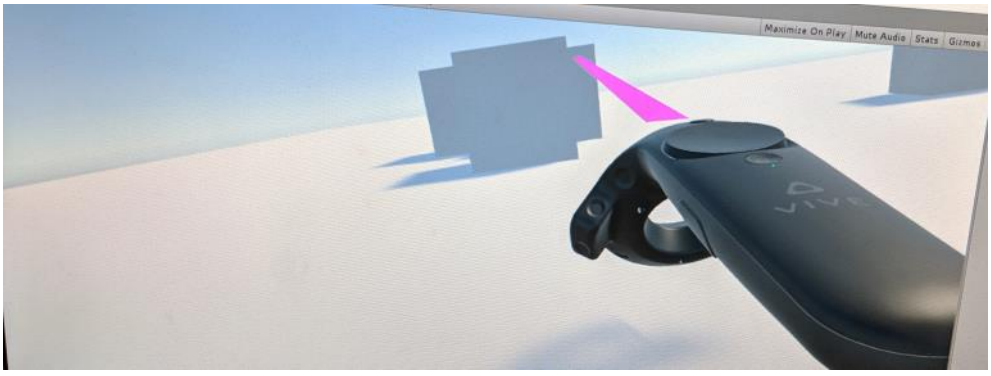
This coded includes the raycast and pick up on trigger button as well as release.

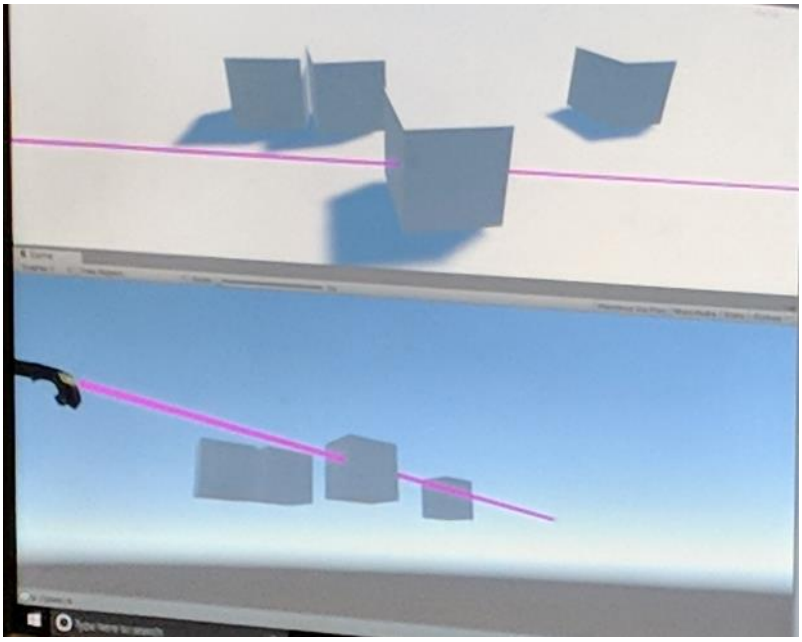
Outputs:

Initialization :

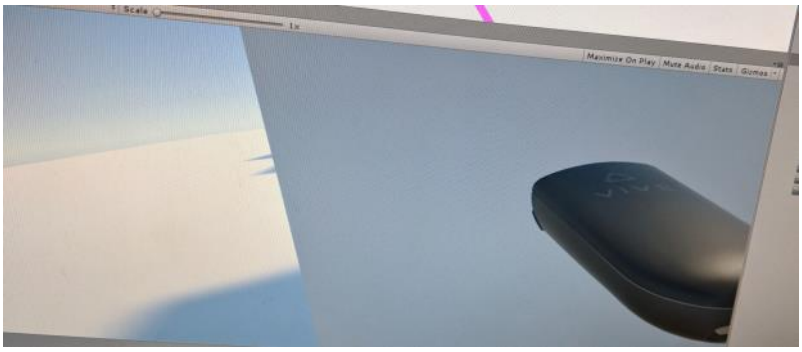


Raycast :

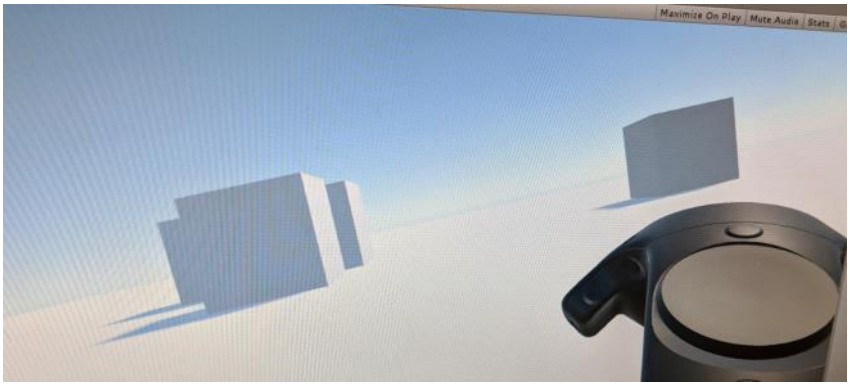




Pickup :



Release:



(ii) The second task was aimed at Data Visualization in Unity and VR. I got started with the iris data plotting in Unity with different colors for different species : Setosa, Versicolor, Verginica. Prefabs were created to instantiate a spherical databall that acts as points in three dimensions. There are two scripts. One to read the CSV files as the iris data was in the CSV format:

Code: CSVReader

```
using UnityEngine;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text.RegularExpressions;
public class CSVReader
{
    static string SPLIT_RE = @",(?=(?:[^""]*""[^""]*"")*(?![^""]*""))";
    static string LINE_SPLIT_RE = @"\r\n|\n\r|\n|\r";
    static char[] TRIM_CHARS = { '\"' };
public static List<Dictionary<string, object>> Read(string file)
    {
```

```
        var list = new List<Dictionary<string, object>>();
        TextAsset data = Resources.Load(file) as TextAsset;
var lines = Regex.Split(data.text, LINE_SPLIT_RE);
if (lines.Length <= 1) return list;
var header = Regex.Split(lines[0], SPLIT_RE);
        for (var i = 1; i < lines.Length; i++)
        {
var values = Regex.Split(lines[i], SPLIT_RE);
            if (values.Length == 0 || values[0] == "") continue;
var entry = new Dictionary<string, object>();
            for (var j = 0; j < header.Length && j < values.Length; j++)
            {
                string value = values[j];
                value = value.TrimStart(TRIM_CHARS).TrimEnd(TRIM_CHARS).Replace("\\", "");
                object finalvalue = value;
                int n;
                float f;
                if (int.TryParse(value, out n))
                {
                    finalvalue = n;
                }
                else if (float.TryParse(value, out f))
                {
                    finalvalue = f;
                }
                entry[header[j]] = finalvalue;
            }
            list.Add(entry);
        }
        return list;
    }
}
```

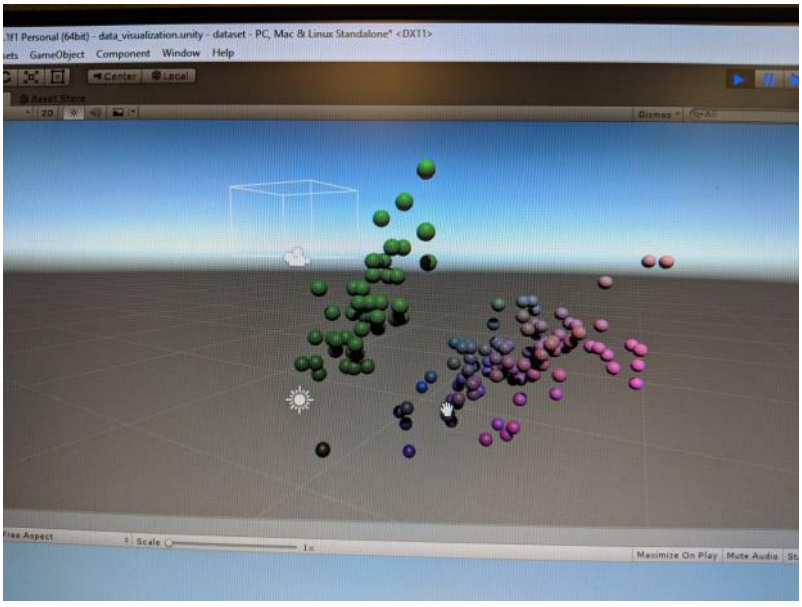
The other code is for plotting the values after normalizing it.

Code : DataPlotter

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
public class DataPlotter : MonoBehaviour {
public string inputfile;
    private List<Dictionary<string, object>> pointList;
    public int columnX = 0;
    public int columnY = 1;
    public int columnZ = 2;
    public string xName;
    public string yName;
    public string zName;
    public GameObject PointPrefab;
    public GameObject PointHolder;
    public float plotScale = 10;
    Color color;



// Use this for initialization
      void Start () {
        pointList = CSVReader.Read(inputfile);
Debug.Log(pointList);
        List<string> columnList = new List<string>(pointList[1].Keys);
        Debug.Log("There are " + columnList.Count + " columns in CSV");
foreach (string key in columnList)
            Debug.Log(" Column name is" + key);
        xName = columnList[columnX];
        yName = columnList[columnY];
        zName = columnList[columnZ];
        float xMax = findMaxVal(xName);
        float yMax = findMaxVal(yName);
```

```
        float zMax = findMaxVal(zName);
        float xMin = findMinVal(xName);
        float yMin = findMinVal(yName);
        float zMin = findMinVal(zName);
for ( var i=0; i<pointList.Count; i++)
        {
            float x= (System.Convert.ToSingle(pointList[i][xName]) - xMin)/(xMax -xMin);
            float y = (System.Convert.ToSingle(pointList[i][yName]) -yMin) / (yMax - yMin);
            float z = (System.Convert.ToSingle(pointList[i][zName]) - zMin) / (zMax - zMin);
            GameObject dataPoint = Instantiate(PointPrefab, new Vector3(x,y,z) * plotScale,
Quaternion.identity);
            dataPoint.transform.parent = PointHolder.transform;
            string dataPointName = pointList[i][xName] + " " + pointList[i][yName] + " " +
pointList[i][zName];
            dataPoint.transform.name = dataPointName;
            dataPoint.GetComponent<Renderer>().material.color = new Color(x, y, z, 1.0f);
            //if (String.Compare(System.Convert.ToString(pointList[i][5]), "setosa", true))
            //{
            //    color = new Color(1, 0, 0, 1.0f);
            //}
            //else if(String.Compare(System.Convert.ToString(pointList[i][5]), "versicolor",true))
            //{
            //    color = new Color(0, 1, 0, 1.0f);
            //}
            //else
            //{
            //    color = new Color(0, 0, 1, 1.0f);
            //}
            //dataPoint.GetComponent<Renderer>().material.color = color;

        }
      }
private float findMaxVal(string columnName)
    {
        float maxVal = Convert.ToSingle(pointList[0][columnName]);
for( var i=0; i<pointList.Count; i++)
        {
            if(maxVal < Convert.ToSingle(pointList[i][columnName]))
            {
                maxVal = Convert.ToSingle(pointList[i][columnName]);
}
        }
        return maxVal;
    }
private float findMinVal(string columnName)
    {
        float minVal = Convert.ToSingle(pointList[0][columnName]);
for (var i = 0; i < pointList.Count; i++)
        {
            if (minVal > Convert.ToSingle(pointList[i][columnName]))
            {
                minVal = Convert.ToSingle(pointList[i][columnName]);
}
        }
        return minVal;
    }
}
```

Output :



Present Work:
Currently, I am working on data visualization using raycasting to choose the parameters for x,y,z axes among population, revenue , area of 50 states of USA and plotting them.