AUTHOR	Michael Soler
CONTACT	michael.soler.beatty@gmail.com
Unity Ver.	2018.2

Index

1.Description of the package.

From cardboard buddies we pretend to give the best VR packages to our customers with simplicity and transparency. This package allows the user to create a simple menu (buttons) that are completely curved and whose parameters can be changed on the editor mode. You will also find the following scene:

Scene based on googleVR helloVR

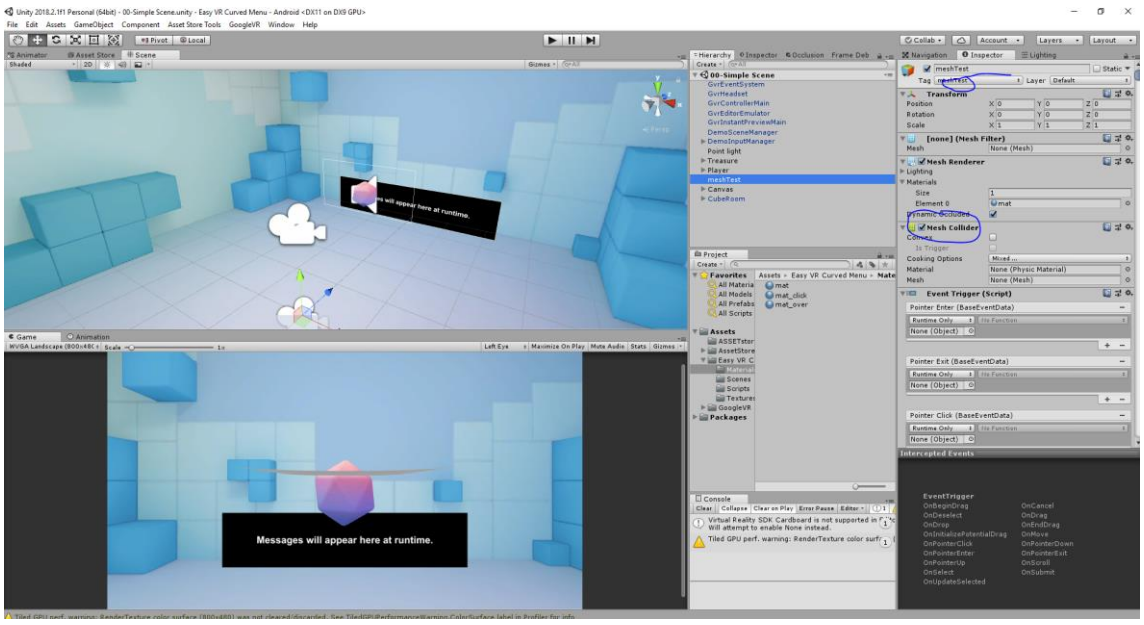
This scene was obtained attaching the radar to the “helloVR” scene in the /googleVR/demos/scenes.

The asset contains the necessary models, textures and prefabs shown in the video.

For further information please contact michael.soler.beatty@gmail.com.

2.Colliders, tags and physics

A mesh collider is added to the “meshTest”, which contains the generated mesh from script. This is also set to tag “meshTest”.



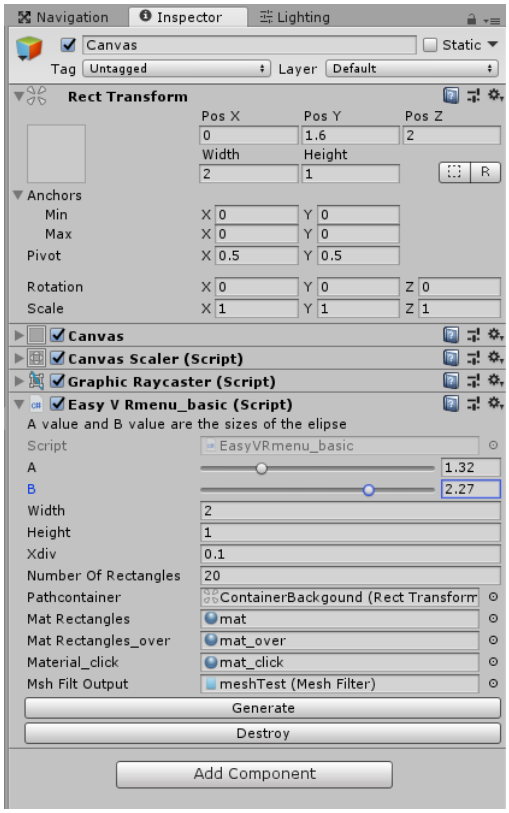
3.Scripting

The main script used in this asset is: “EasyVRmenu_basic.cs” which is described here:

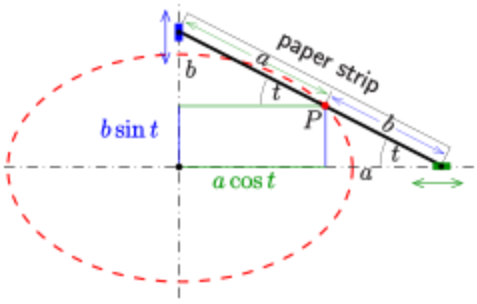
```
public void drawGeometry()
  {
      // scale the canvas to the size
      width = gameObject.GetComponent<RectTransform>().sizeDelta[0];
      height = gameObject.GetComponent<RectTransform>().sizeDelta[1];
```

```
        GameObject child;

        MeshFilter tempMeshF;
        MeshRenderer meshRender;

        Xdiv = (width / numberOfRectangles);

        Vector3[] points = new Vector3[numberOfRectangles];

        meshes = new Mesh[numberOfRectangles];
        CombineInstance[] TBC = new CombineInstance[meshes.Length];

        // top part of the geometry
        for (int ii = 0; ii < numberOfRectangles; ii++)
        {
            //creting the different rectangles
            child = new GameObject("DiffRect"+ii);
            child.transform.parent = pathcontainer;
            tempMeshF = child.gameObject.AddComponent<MeshFilter>();
            meshRender = child.gameObject.AddComponent<MeshRenderer>();
            meshRender.material = matRectangles;
            meshRender.enabled = false;

            // four points of the mesh
                float z1 = b / a * Mathf.Sqrt(Mathf.Pow(a, 2) - Mathf.Pow(ii *
Xdiv - width / 2, 2)) - b;
                float z2 = b / a * Mathf.Sqrt(Mathf.Pow(a, 2) - Mathf.Pow((ii +
1) * Xdiv - width / 2, 2)) - b;

            points[0] = transform.position+new Vector3(-width/2,-
height/2,0)+new Vector3(ii * Xdiv, 0,z1);
            points[1] = transform.position + new Vector3(-width / 2, -height /
2, 0) + new Vector3((ii+1) * Xdiv, 0, z2);
            points[3] = transform.position + new Vector3(-width / 2, height /
2, 0) + new Vector3((ii+1) * Xdiv, 0, z2);
            points[2] = transform.position + new Vector3(-width / 2, height /
2, 0) + new Vector3(ii * Xdiv, 0, z1);



            // create quads
            meshes[ii]= createGeometry(points[0], points[1], points[2],
points[3], tempMeshF, ii*Xdiv/2, (ii+1)*Xdiv/2);

            TBC[ii].mesh=meshes[ii];
            TBC[ii].transform = child.transform.localToWorldMatrix;

        }

        // combine quads
        Mesh meshOut = new Mesh();
        meshOut.CombineMeshes(TBC,true);

        mshFiltOutput.mesh = meshOut;
        mshFiltOutput.transform.GetComponent<MeshCollider>().sharedMesh =
meshOut;



        // the new object has the same events that the ones set on the initial
object
```

```
        //click event
        EventTrigger.Entry entry = new EventTrigger.Entry();
        entry.eventID = EventTriggerType.PointerClick;
        entry.callback.AddListener((eventData) => { MyFunction_CLICK
(mshFiltOutput.gameObject); });
        mshFiltOutput.GetComponent<EventTrigger>().triggers.Add(entry);

        //enter event
        entry = new EventTrigger.Entry();
        entry.eventID = EventTriggerType.PointerEnter;
        entry.callback.AddListener((eventData) => {
MyFunction_ENTER(mshFiltOutput.gameObject); });
        mshFiltOutput.GetComponent<EventTrigger>().triggers.Add(entry);

        //quit event
        entry = new EventTrigger.Entry();
        entry.eventID = EventTriggerType.PointerExit;
        entry.callback.AddListener((eventData) => {
MyFunction_EXIT(mshFiltOutput.gameObject); });
        mshFiltOutput.GetComponent<EventTrigger>().triggers.Add(entry);

    }
```

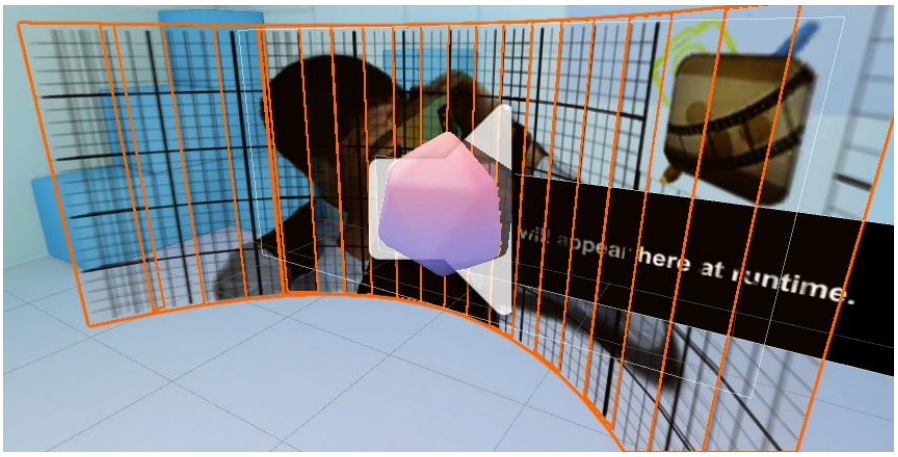


The script generates a curved surface that follows the expression of an ellipse with parameters a and b (main axis).



$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1,$$

The curved surface is generated by infinitesimal quads that are joined into a single mesh.



4. Video tutorial

We have a video tutorial explaining how the scripts and game mechanics works.

https://www.youtube.com/watch?v=ZBdrK2EA844