AUTHOR	**Cardboard Buddies**
CONTACT	**michael.soler.beatty@gmail.com**
Unity Ver.	**2019.1.f1**

Index

1.Dependencies

This package does not need any other packages.

2.Description of the package.

This package generates Bezier curves in linerenderes and is able to deform a mesh accordingly to these lines in 3d space. A cubic Bezier curve is used with two control points per fragment. A fragment is the curve considered between to different points. The mesh is deformed using and editor script that runs in the Update() routine. Once the fragments are created, the user can select a mesh to apply to them. Finally, the user must select an object to follow the trajectory.

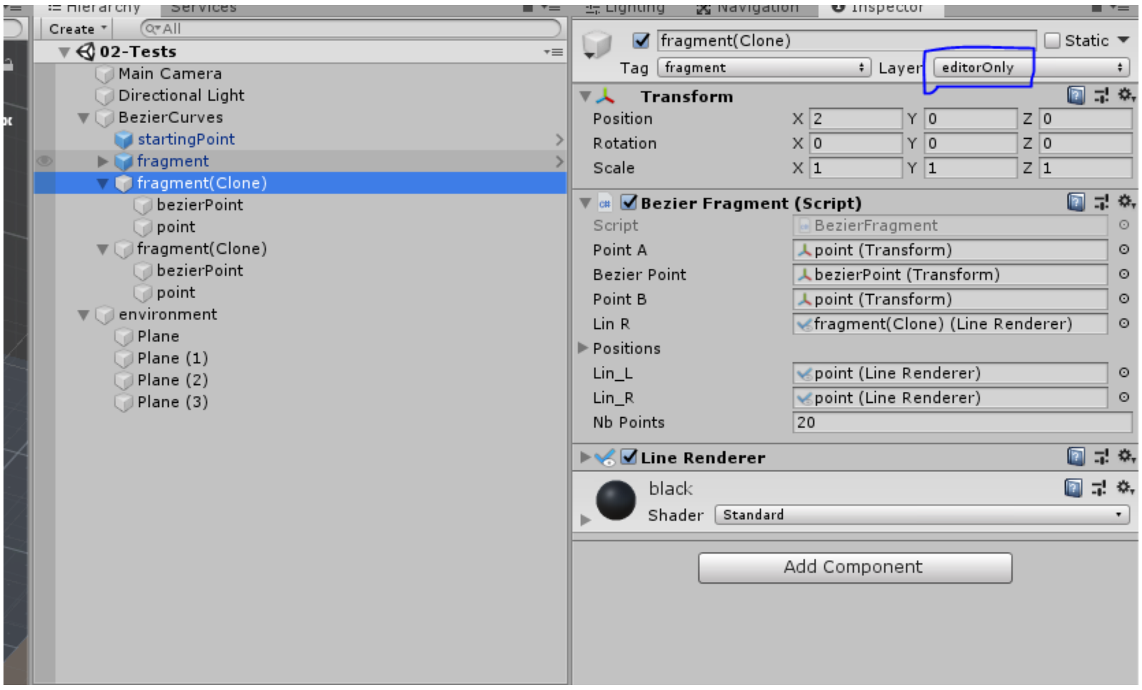
This package includes:

- The scripts that control the Bezier fragment generation.
- The scripts that control the mesh deformation.
- Scripts that control the movement of the gameobject following the trajectory.
- The textures, models, materials and scenes showed in the video.

For further pre-post sales customer service please contact michael.soler.beatty@gmail.com

3.Tags, layers and other considerations

The gameobjects that won't appear on the gameView are tagged as "onyEditor".

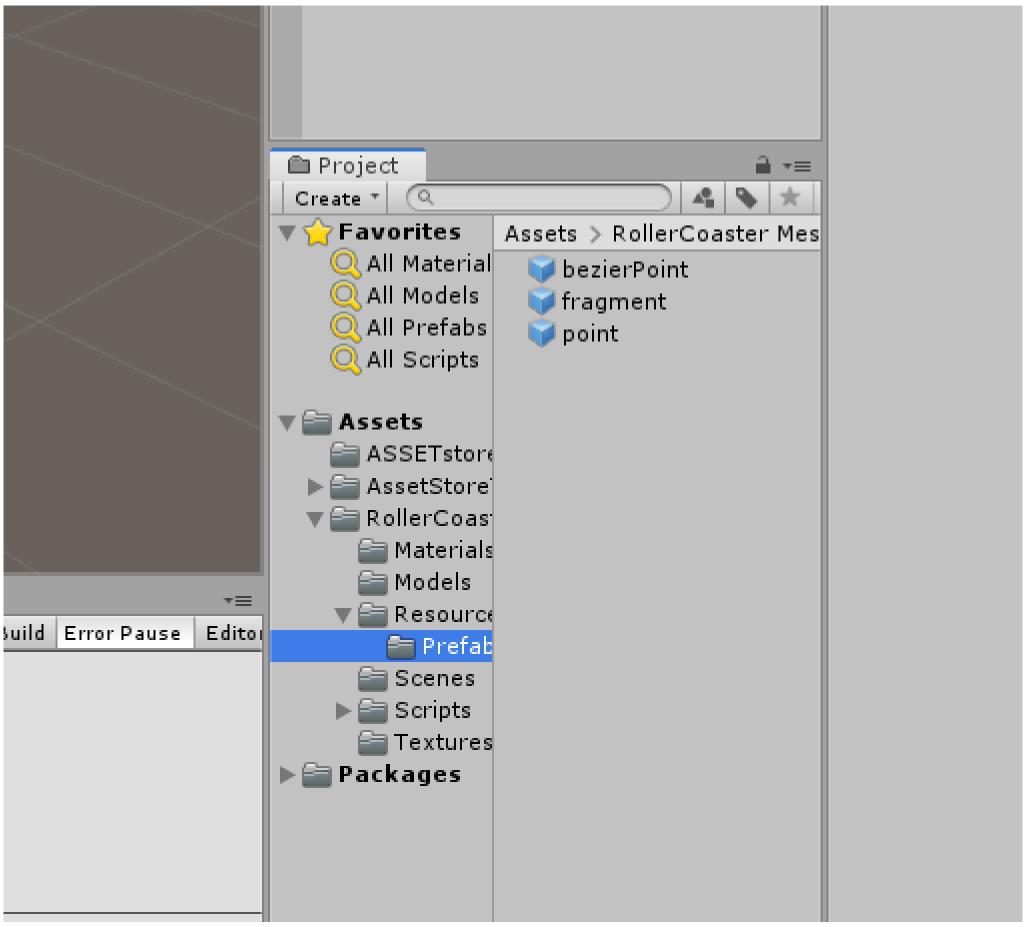


```
02-Tests
  Main Camera
  Directional Light
  BezierCurves
    startingPoint
    fragment
    fragment(Clone)
      bezierPoint
      point
    fragment(Clone)
      bezierPoint
      point
  environment
    Plane
    Plane (1)
    Plane (2)
    Plane (3)
```

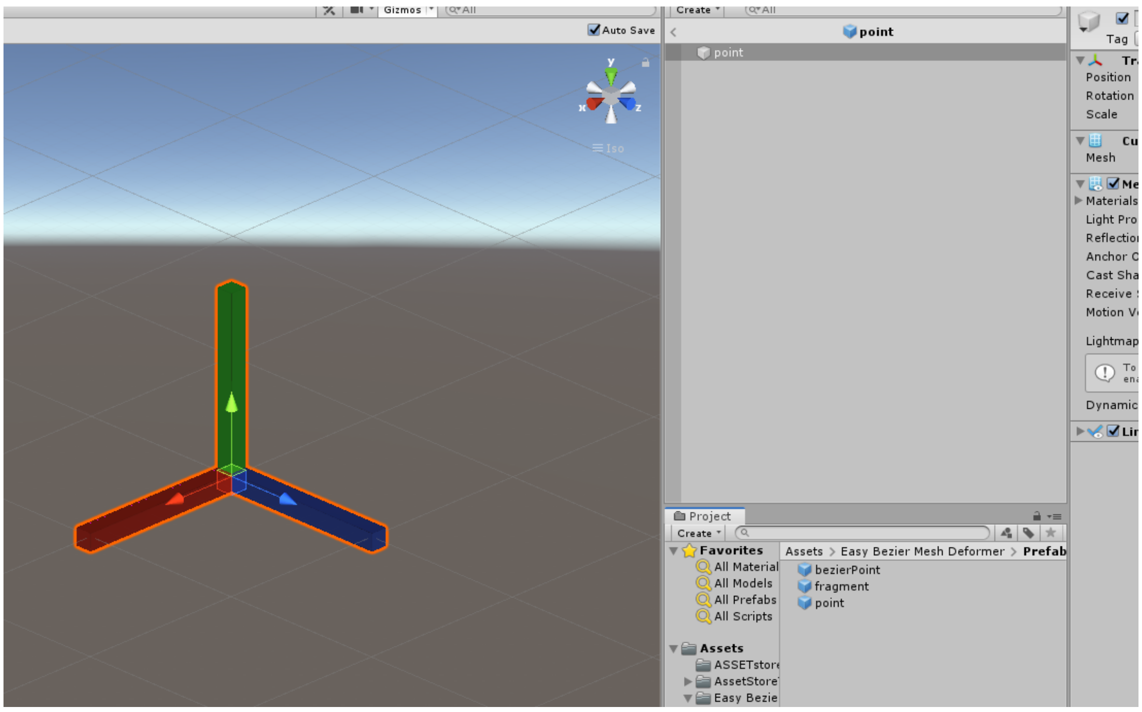
```
fragment(Clone)                         Static
Tag  fragment            Layer  editorOnly
Transform
  Position   X 2   Y 0   Z 0
  Rotation   X 0   Y 0   Z 0
  Scale      X 1   Y 1   Z 1
Bezier Fragment (Script)
  Script        BezierFragment
  Point A       point (Transform)
  Bezier Point  bezierPoint (Transform)
  Point B       point (Transform)
  Lin R         fragment(Clone) (Line Renderer)
  Positions
  Lin_L         point (Line Renderer)
  Lin_R         point (Line Renderer)
  Nb Points     20
Line Renderer
  black
  Shader  Standard
Add Component
```

3.Prefabs

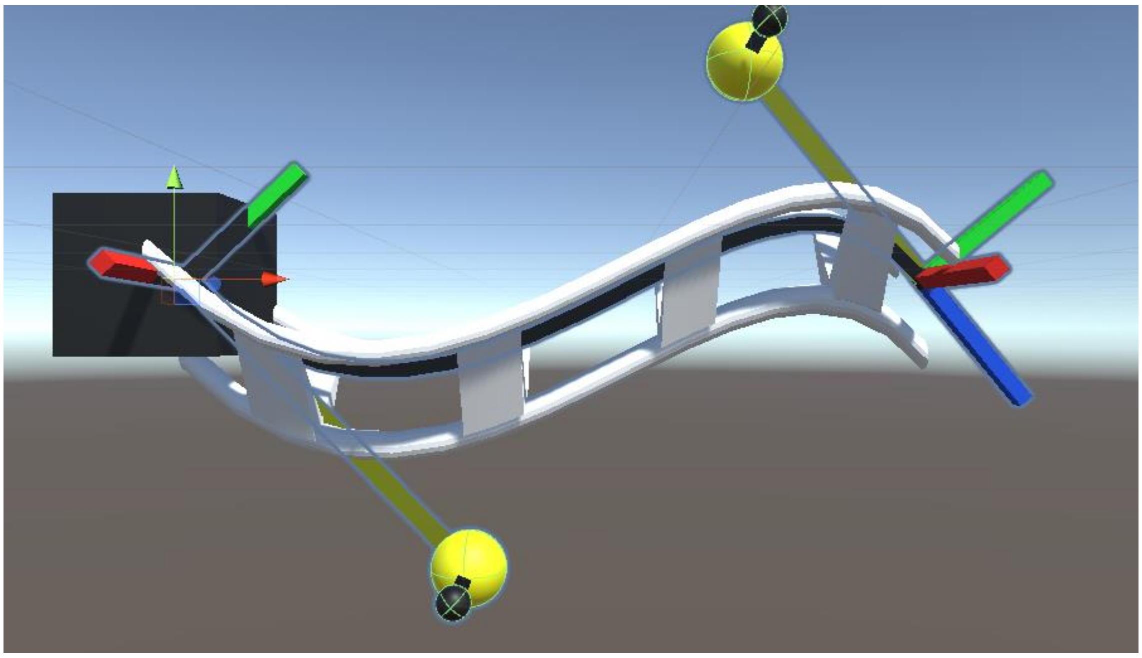
The prefabs consist on the following gameobjects:



point: displays the 3D point in the editor

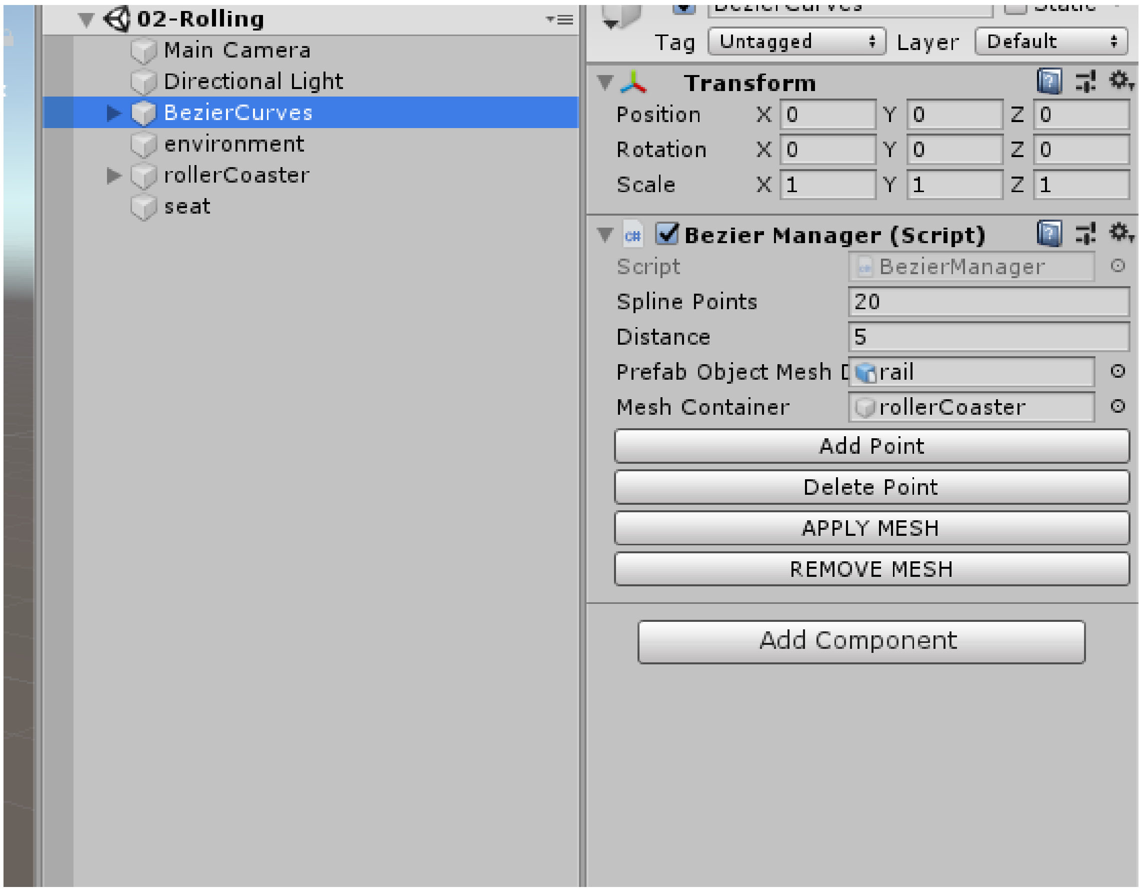


fragment: it is two bezierpoint + two regular point +linerenderer.



4.Scripting

1. Select the "BezierCurves" gameobject.



Tag Untagged Layer Default

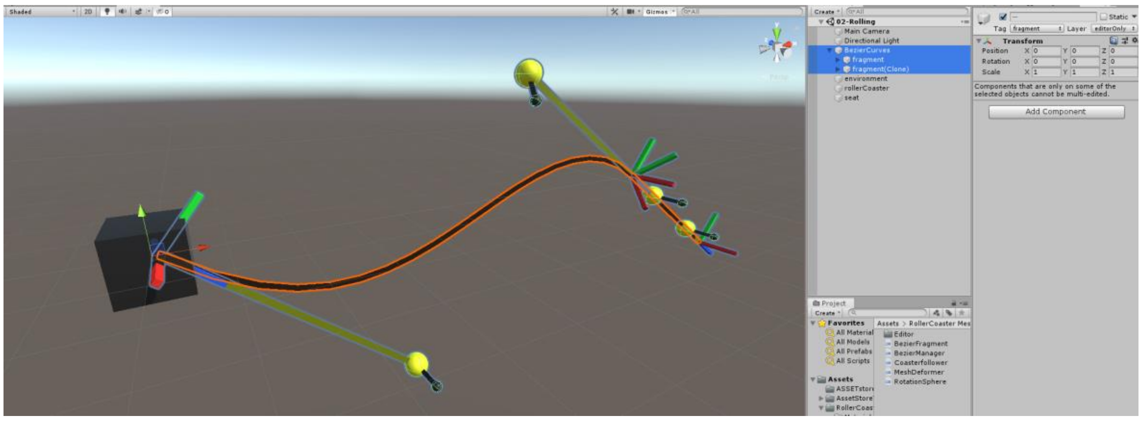
Transform

Position	X	0	Y	0	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

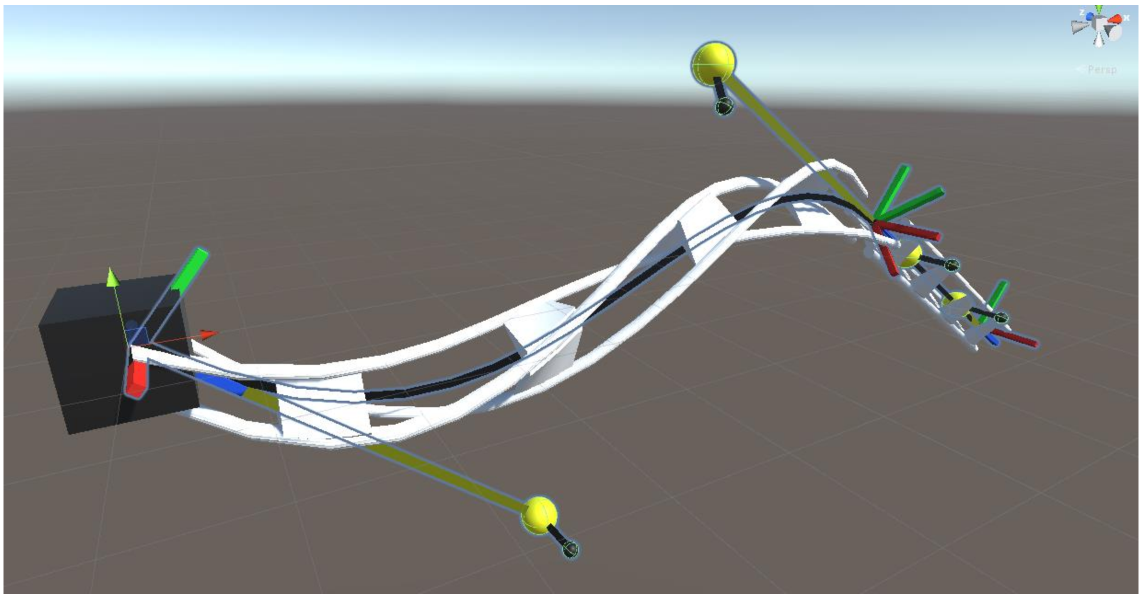
Bezier Manager (Script)

Script	BezierManager
Spline Points	20
Distance	5
Prefab Object Mesh D	rail
Mesh Container	rollerCoaster

2. To add a point (fragment) click on the add point button. This will add a new fragment.



3. Select a mesh an put it in “prefab object mesh “ variable. Select the “mesh container” as well, where the gameobjects will be instantiated. Click on apply mesh to add the deformed mesh.



The main script is the one used to generate the mesh deformation

```
[ExecuteInEditMode]
public class BezierFragment : MonoBehaviour
{
    // Start is called before the first frame update
    //this is the starting point of the bezier courve
    public Transform pointA;
    //this is the point that changes the shape
    public Transform[] bezierPoint;
    //this is the ending point of the bezier courve
    public Transform pointB;
    //line renderer for displaying info in editor
    public LineRenderer linR;

    //public readable position and rotation
    public Vector3[] positions;
    public Quaternion[] rotations;

    //left and right renderes
    public LineRenderer lin_L;
    public LineRenderer lin_R;

    //the number of points of the generated linerenderer
    public int nbPoints = 20;

    public void Update()
    {
        if (!Application.IsPlaying(gameObject))
        {

            positions = new Vector3[nbPoints];
            rotations = new Quaternion[nbPoints];

            if (pointA != null && pointB != null && bezierPoint != null)
            {

                //set the line renderer
                linR = transform.GetComponent<LineRenderer>();
                linR.positionCount = nbPoints;
                lin_L.positionCount = 2;
                lin_R.positionCount = 2;

                //generate the curve
                generateBezierFragmentCubic(pointA, bezierPoint[0],
bezierPoint[1], pointB);

            }
        }
    }


    public void generateBezierFragmentCubic(Transform A, Transform Z, Transform
Z2, Transform B)
    {

        //positions of each point
        Vector3 posA = A.position;
        Vector3 posB = B.position;
        Vector3 posZ = Z.position;
        Vector3 posZ2 = Z2.position;


        //set both points in linerender
        lin_L.SetPosition(0, posA);
        lin_L.SetPosition(1, posZ);

        lin_R.SetPosition(0, posZ2);
        lin_R.SetPosition(1, posB);


        //temp variables
        float x = 0;
        float y = 0;
        float z = 0;

        float nx = 0;
        float ny = 0;
        float nz = 0;


        //looping for the line renderer
        for (int ii = 0; ii < linR.positionCount; ii++)
        {
            float t = (float)ii / (float)(linR.positionCount - 1);

            x = (1 - t) * (1 - t)* (1 - t) * posA.x + 3 * (1 - t) * (1 - t) * t
* posZ.x+ 3 * (1 - t) * t * t * posZ2.x + t * t*t * posB.x;
            y = (1 - t) * (1 - t) * (1 - t) * posA.y + 3 * (1 - t) * (1 - t) *
t * posZ.y + 3 * (1 - t) * t * t * posZ2.y + t * t*t * posB.y;
            z = (1 - t) * (1 - t) * (1 - t) * posA.z + 3 * (1 - t) * (1 - t) *
t * posZ.z + 3 * (1 - t) * t * t * posZ2.z + t * t *t* posB.z;

            nx = (-3 * (1 - t) * (1 - t)) * posA.x + 3 * (-2 * (1 - t) * t + (1
- t) * (1 - t)) * posZ.x + 3 * (-t * t + (1 - t) * 2 * t) * posZ2.x +3* t *  t
* posB.x;
            ny = (-3 * (1 - t) * (1 - t)) * posA.y + 3 * (-2 * (1 - t) * t + (1
- t) * (1 - t)) * posZ.y + 3 * (-t * t + (1 - t) * 2 * t) * posZ2.y + 3* t * t
* posB.y;
            nz = (-3 * (1 - t) * (1 - t)) * posA.z + 3 * (-2 * (1 - t) * t + (1
- t) * (1 - t)) * posZ.z + 3 * (-t * t + (1 - t) * 2 * t) * posZ2.z + 3* t * t
* posB.z;

            //spline points positions
            positions[ii] = new Vector3(x, y, z);
            linR.SetPosition(ii, positions[ii]);

            //normal directions and quaternions
            Vector3 normal = new Vector3(nx,ny,nz);
            Quaternion quat = Quaternion.LookRotation(normal, Vector3.up);

            //assign displaced vertices inn local space
            float rotZ = 0;
            if (pointA.localEulerAngles.z >= 0 && pointB.localEulerAngles.z >=
0)
            {
                rotZ = Mathf.Lerp(pointA.localEulerAngles.z,
pointB.localEulerAngles.z, t);
            }
            else if (pointA.localEulerAngles.z < 0 && pointB.localEulerAngles.z
< 0)
            {
                rotZ = Mathf.Lerp(-pointA.localEulerAngles.z, -
pointB.localEulerAngles.z, t);
            }


            rotations[ii] = quat*Quaternion.Euler(0,0,rotZ);

        }


    }

}
```

Important: mark the model object mesh as read/write enabled

5.VIDEO TUTORIAL

We intend to give our customer the best service. To this aim, we upload tutorial videos of the asset always:

https://youtu.be/QL3Dek6XAU4