```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

//[ExecuteInEditMode]

struct Dvec3
{
    public double x;
    public double y;
    public double z;
}


public class Line : MonoBehaviour
{
    public Color baseColor;
    public GameObject target;
    public Material material;

    [SerializeField] private Button Resetbutton;
    [SerializeField] private Slider slider;

    //[SerializeField] private Button Resetbutton;

    public List<Vector3> points = new List<Vector3>();
    public List<float> coeffAs = new List<float>();
    public List<float> coeffBs = new List<float>();
    public List<float> KnotSequence = new List<float>();
    int Degree = 0;
    int N = 0;
    float t_value = 0;
    float J = 0;
    Vector3[,] res;

    [SerializeField] TemplatePointPlacer pointplaceController;
    public Vector3 mousePos;

    private void Start()
    {
        KnotSequence.Add(0);
        KnotSequence.Add(1);
        KnotSequence.Add(2);
        KnotSequence.Add(3);
        Degree = 3;

    }


    void Update()
    {
        N = KnotSequence.Count;
```

```csharp

        Resetbutton.onClick.AddListener(ResetActions);


        if (pointplaceController != null)
        {
            if (pointplaceController.degree < 31)
            {
                res = new Vector3[pointplaceController.degree + 1, pointplaceController.degree + 1];
                for (int a = 0; a <= pointplaceController.degree; a++)
                {
                    res[0, a] = pointplaceController.pointPool[a].transform.position;
                }
            }
        }
        if (points.Count < 31)
            if (Input.GetMouseButtonDown(1))
            {
                mousePos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
                mousePos.z = 0;

                if (pointplaceController != null)
                {
                    if (pointplaceController.degree < 31)
                    {
                        points.Add(pointplaceController.pointPool[pointplaceController.degree].transform.position);
                        KnotSequence.Add(pointplaceController.degree + 4);//TODO
                        t_value = KnotSequence.Count / 2f;//TODO
                    }
                }

            }
        if (pointplaceController != null)
        {
            if (pointplaceController.degree < 31)
            {
                if (pointplaceController.degree > -1)
                {
                    for (int i = 0; i <= pointplaceController.degree; i++)
                    {
                        points[i] = pointplaceController.pointPool[i].transform.position;
                    }
                }

            }
        }

```

```

    }

    void OnRenderObject()
    {
        if (points.Count > 0)
        {
            RenderLines(points, baseColor);
            RenderLines1(points, baseColor);
        }
    }

    public float FindJ(List<float> Knot, float tVal)
    {
        for (int i = 0; i < Knot.Count - 1; i++)
        {
            if (Knot[i] <= tVal && tVal < Knot[i + 1])
                return i;
        }
        return 0;
    }
    virtual public void RenderLines(List<Vector3> points, Color color)
    {
        GL.Begin(GL.LINES);
        material.SetPass(0);
        for (int i = 0; i < points.Count - 1; i++)
        {
            GL.Color(Color.white);
            GL.Vertex(points[i]);

            //GL.Color(material.color);
            GL.Vertex(points[i + 1]);
        }
        GL.End();

    }
    virtual public void RenderLines1(List<Vector3> points, Color color)
    {
        int size_ = points.Count;
        List<Vector3> result = new List<Vector3>();

        float a = KnotSequence[Degree];
        float b = KnotSequence[N - Degree - 1];

        if (Degree < size_ && size_ < (N - Degree))
            for (float i = a; i < b; i += 0.1f)
            {
                J = FindJ(KnotSequence, i);
                result.Add(P_d_J(i, KnotSequence, (int)J, Degree,
                  Degree));
            }

        GL.Begin(GL.LINES);
        material.SetPass(0);

        for (int i = 0; i < result.Count - 1; i++)
        {
            GL.Color(Color.red);
            GL.Vertex(result[i]);
            GL.Vertex(result[i + 1]);
        }
        GL.End();
    }

    public Vector3 P_d_J(float t, List<float> t_i, int index_, int d, int k)
    {
        if (k == 0)
        {
            if (index_ < 0 || index_ >= points.Count)
                return new Vector3(0, 0, 0);
            return points[index_];
        }

        Vector3 left = ((t - t_i[index_]) / (t_i[index_ + d - (k - 1)] - t_i[index_])) * P_d_J(t, t_i, index_, d, k - 1);
        Vector3 right = ((t_i[index_ + d - (k - 1)] - t) / (t_i[index_ + d - (k - 1)] - t_i[index_])) * P_d_J(t, t_i, index_ - 1, d, k - 1);

        return left + right;
    }

    public void ResetActions()
    {
        points.Clear();



        if (pointplaceController != null)
            for (int i = 0; i < 30; i++)
            {
                pointplaceController.pointPool[i].gameObject.SetActive(false);
                pointplaceController.pointPool[i].transform.position = new Vector3(0, 0, 0);
            }
        if (pointplaceController != null)
            pointplaceController.degree = -1;
    }


}
```