```
using UnityEngine;
using System.Collections;
using UnityEngine.UIElements;

public class CatmullRomCurveInterpolation : MonoBehaviour {

    const int NumberOfPoints = 8;
    Vector3[] controlPoints;

    const int MinX = -5;
    const int MinY = -5;
    const int MinZ = 0;

    const int MaxX = 5;
    const int MaxY = 5;
    const int MaxZ = 5;

    const float TotalAnimationTime = 5.0f;

    float time = 0;

    float u = 0;
    int segNum = 0;

    ArrayList supersampleTable = new ArrayList();

    int PositiveMod(int a, int b)
    {
        return (a % b + b) % b;
    }

    public struct TableEntry
    {
        public TableEntry(float inputU, float inputS, int inputSegNum)
        {
            u = inputU;
            s = inputS;
            segNum = inputSegNum;
        }
        public float u;
        public float s;
        public int segNum;
    }

    /* Returns a point on a cubic Catmull-Rom/Blended Parabolas curve
     * u is a scalar value from 0 to 1
     * segment_number indicates which 4 points to use for interpolation
     */
    Vector3 ComputePointOnCatmullRomCurve(double u, int segmentNumber)
```

```
  {
    int index0 = PositiveMod(segmentNumber - 2, NumberOfPoints);
    int index1 = PositiveMod(segmentNumber - 1, NumberOfPoints);
    int index2 = PositiveMod(segmentNumber, NumberOfPoints);
    int index3 = PositiveMod(segmentNumber + 1, NumberOfPoints);
    float tau = .5f;

    Vector3 c3 = -tau * controlPoints[index0] + (2 - tau) * controlPoints
     [index1] + (tau - 2) * controlPoints[index2] + tau * controlPoints
     [index3];
    Vector3 c2 = 2 * tau * controlPoints[index0] + (tau - 3) *
     controlPoints[index1] + (3 - 2 * tau) * controlPoints[index2] + -tau
     * controlPoints[index3];
    Vector3 c1 = -tau * controlPoints[index0] + tau * controlPoints
     [segmentNumber];
    Vector3 c0 = controlPoints[index1];

    return Mathf.Pow((float)u, 3) * c3 + Mathf.Pow((float)u, 2) * c2 +
     (float)u * c1 + c0; ;
  }

  void GenerateControlPointGeometry()
  {
    for(int i = 0; i < NumberOfPoints; i++)
    {
      GameObject tempcube = GameObject.CreatePrimitive
       (PrimitiveType.Cube);
      tempcube.transform.localScale -= new Vector3(0.8f,0.8f,0.8f);
      tempcube.transform.position = controlPoints[i];
    }
  }

  Vector3 CalculateTangent(float u, int segmentNumber)
  {
    float delta = 0.01f; //Small step size
    Vector3 tangentA = ComputePointOnCatmullRomCurve(u - delta,
     segmentNumber);
    Vector3 tangentB = ComputePointOnCatmullRomCurve(u + delta,
     segmentNumber);
    return (tangentB - tangentA).normalized;
  }

  float Ease(float time)
  {
    return -2 * Mathf.Pow(time, 3) + 3 * Mathf.Pow(time, 2);
  }

  void CreateSupersampleTable()
  {
    float deltaU = .01f;
    float lengthOfCurve = 0;

    //Calculate length of total curve
    for (int segNum = 0; segNum < NumberOfPoints; segNum++)
    {
      for (float u = 0; u < 1; u += deltaU)
      {
          Vector3 diff = ComputePointOnCatmullRomCurve(u + deltaU, segNum) - ComputePointOnCatmullRomCurve(u, segNum);
          lengthOfCurve += Mathf.Abs(diff.magnitude);
      }
    }

    //Use length of total curve to calculate accurate s values for each table entry
    float currentDistance = 0;
    for (int segNum = 0; segNum < NumberOfPoints; segNum++)
    {
      for (float u = 0; u < 1; u += deltaU)
      {
          Vector3 diff = ComputePointOnCatmullRomCurve(u + deltaU, segNum) - ComputePointOnCatmullRomCurve(u, segNum);
          currentDistance += Mathf.Abs(diff.magnitude);
          TableEntry t = new TableEntry(u, currentDistance / lengthOfCurve, segNum);
          supersampleTable.Add(t);

      }
    }
  }

  // Interpolate u based on s
  TableEntry InterpolateU(float s)
  {
    float sGap = 1.0f;
    TableEntry closestEntry = new TableEntry();
    foreach (TableEntry TE in supersampleTable)
    {
      if(Mathf.Abs(s - TE.s) < sGap)
      {
          closestEntry = TE;
          sGap = Mathf.Abs(s - TE.s);
      }
    }
    return closestEntry;
  }

  // Use this for initialization
  void Start () {

    controlPoints = new Vector3[NumberOfPoints];

    //Set points randomly
    controlPoints[0] = new Vector3(0,0,0);
    for(int i = 1; i < NumberOfPoints; i++)
    {
      controlPoints[i] = new Vector3(Random.Range(MinX,MaxX),Random.Range ⤸
        (MinY,MaxY),Random.Range(MinZ,MaxZ));
    }

    GenerateControlPointGeometry();
    CreateSupersampleTable();
  }

  // Update is called once per frame
  void Update () {
    time += Time.deltaTime / TotalAnimationTime;
    time = Mathf.Clamp01(time);
    float s = Ease(time);

    // Interpolate u based on s
    TableEntry selectedEntry = InterpolateU(s);

    u = selectedEntry.u;
    segNum = selectedEntry.segNum;

    Vector3 position = ComputePointOnCatmullRomCurve(u,segNum);
    Vector3 tangent = CalculateTangent(u, segNum);

    if(tangent != Vector3.zero)
    {
      transform.rotation = Quaternion.LookRotation(tangent, Vector3.up);
    }

    transform.position = position;
  }
}
```