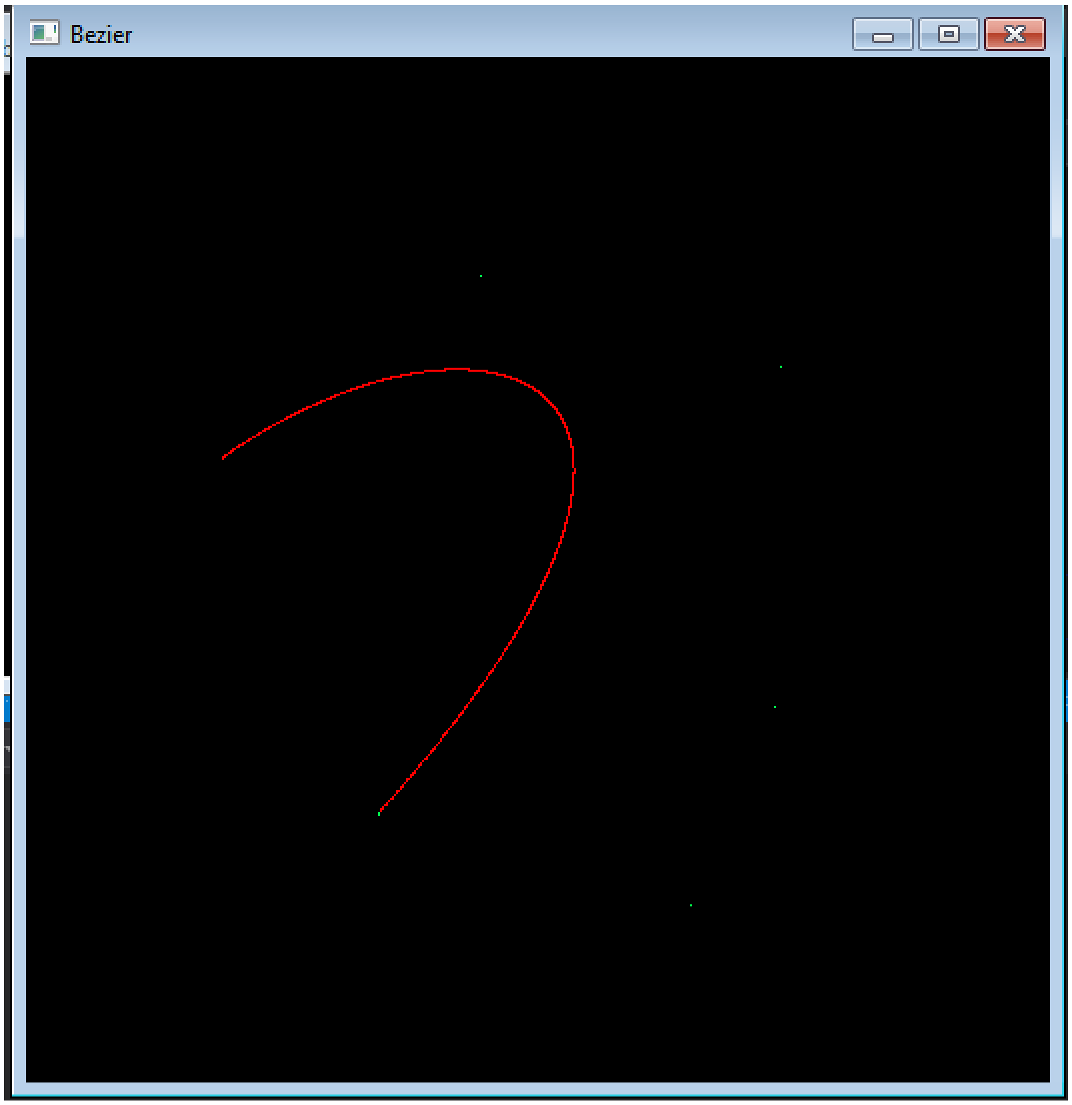
Report

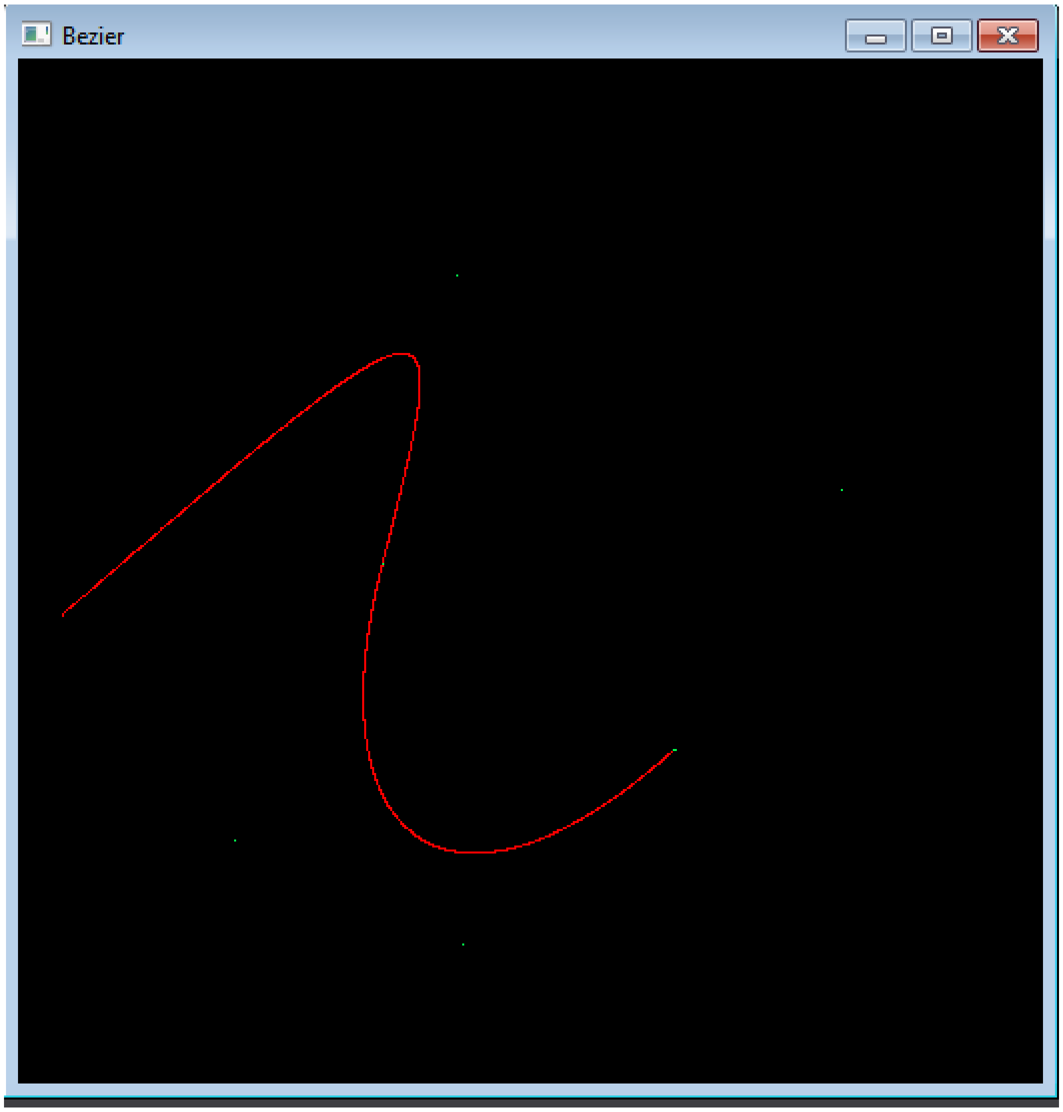
Garima Singh

UFID: 5197-5877

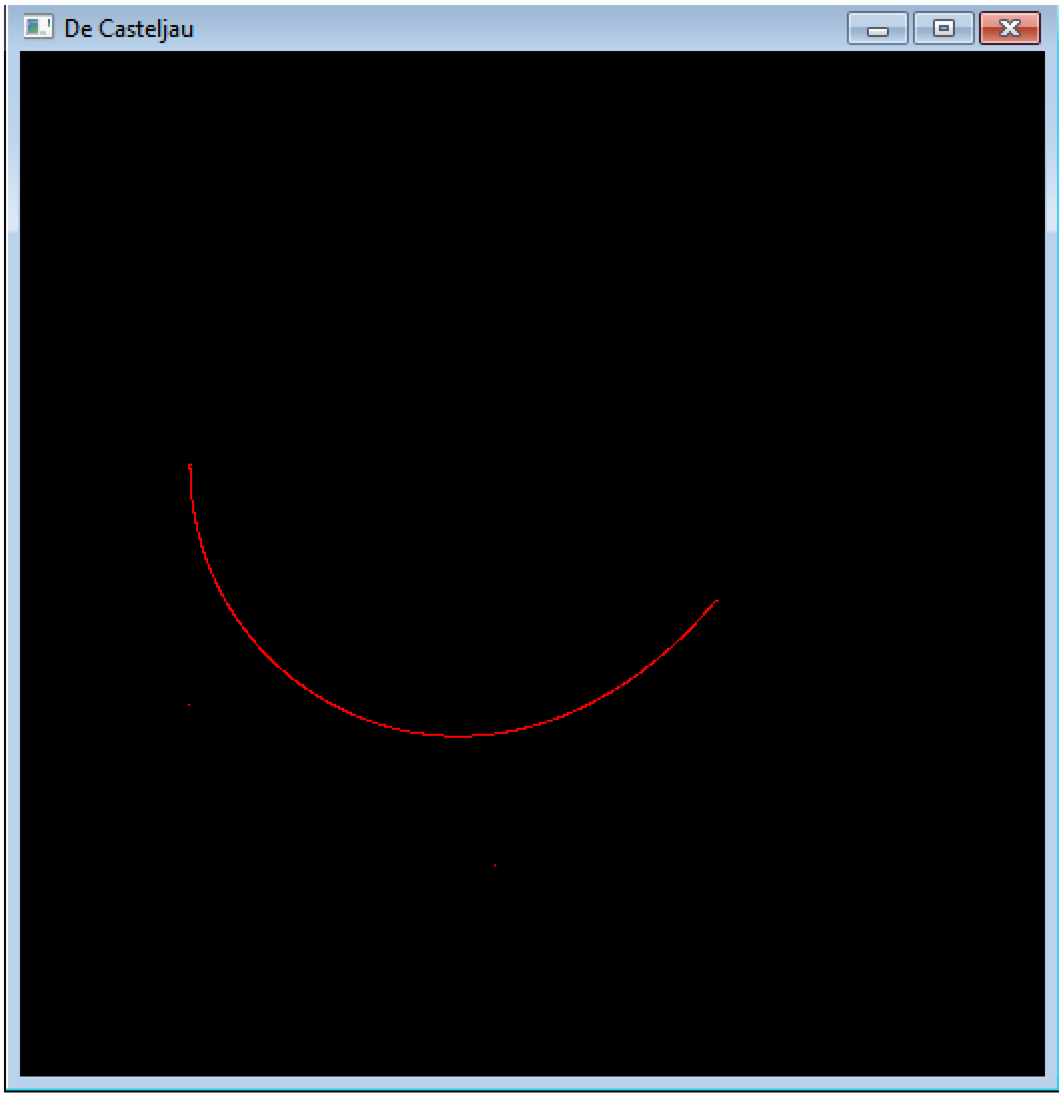
Create a (PDF) report on the results of each method discussed above. Explore and discuss what happens when a control point is repeated (e.g., P0 = P1). Also comment on the advantages and disadvantages each approach. Include images of at least two different configurations of control points and show how each method renders the curve.

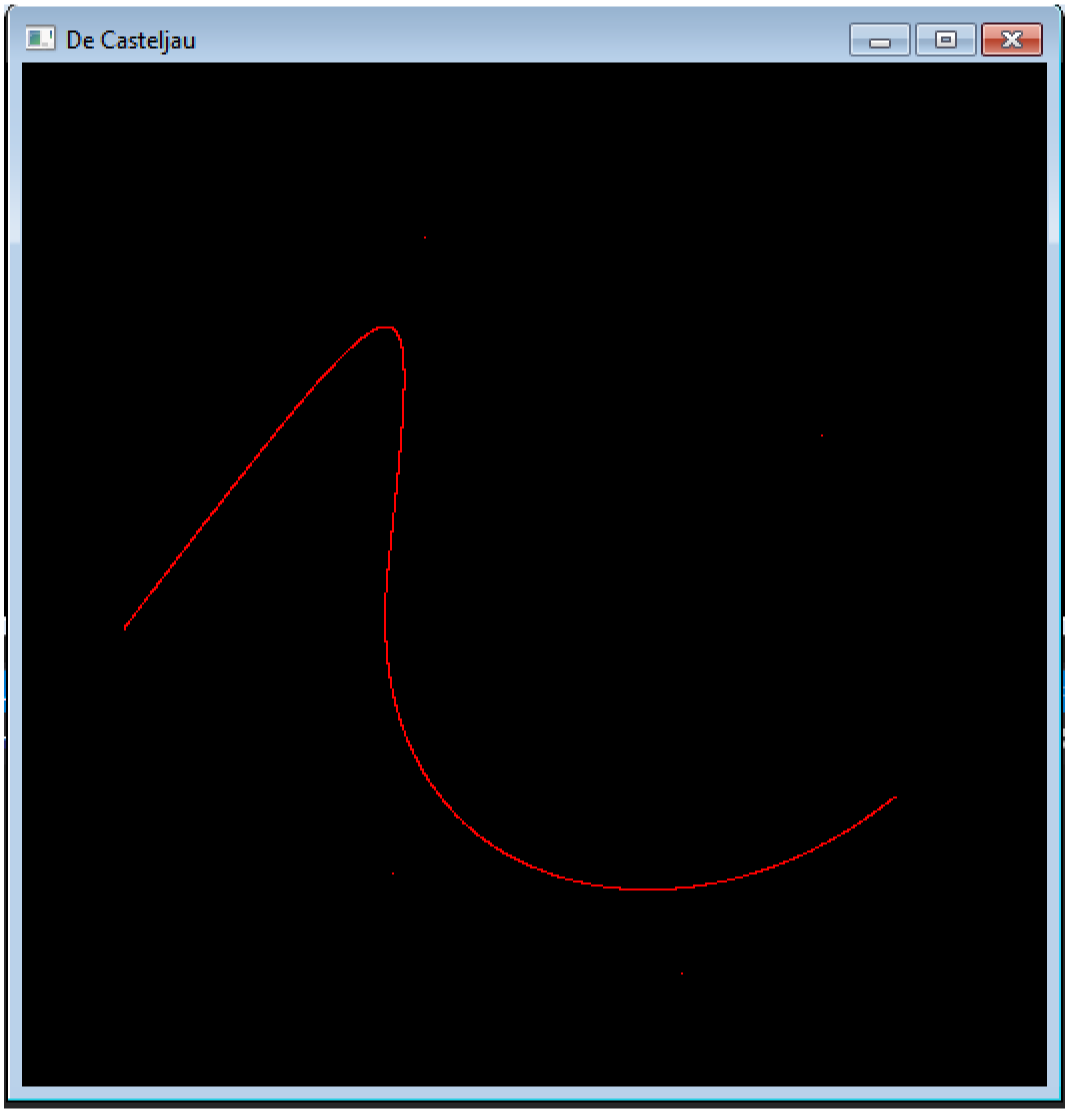
- The consecutive segments of the curve drawn are c1 continuous. The second control point of the segments after the first one is chosen such that the common point between the 2 segments is the midpoint of the segment joining this point and the second to last point of the previous curve. Rest of the points are those provided by the user. The curve is drawn for the first n points out of user given points where n= (k)%3 +1 for some k. this is done for each of the methods : the one using blending functions, the one using De Casteljau's algorithm and the one using OpenGL API.
- Bezier curve using blending function: Here, I use the Bernstein polynomials (blending functions) to calculate points on the curve for various values of the parameter t [0, 1]. It gives a pretty good result for step size of 0.001 (increment t by step size for each iteration). Here are the screenshots for 2 configurations. The dots are control points.


Bezier

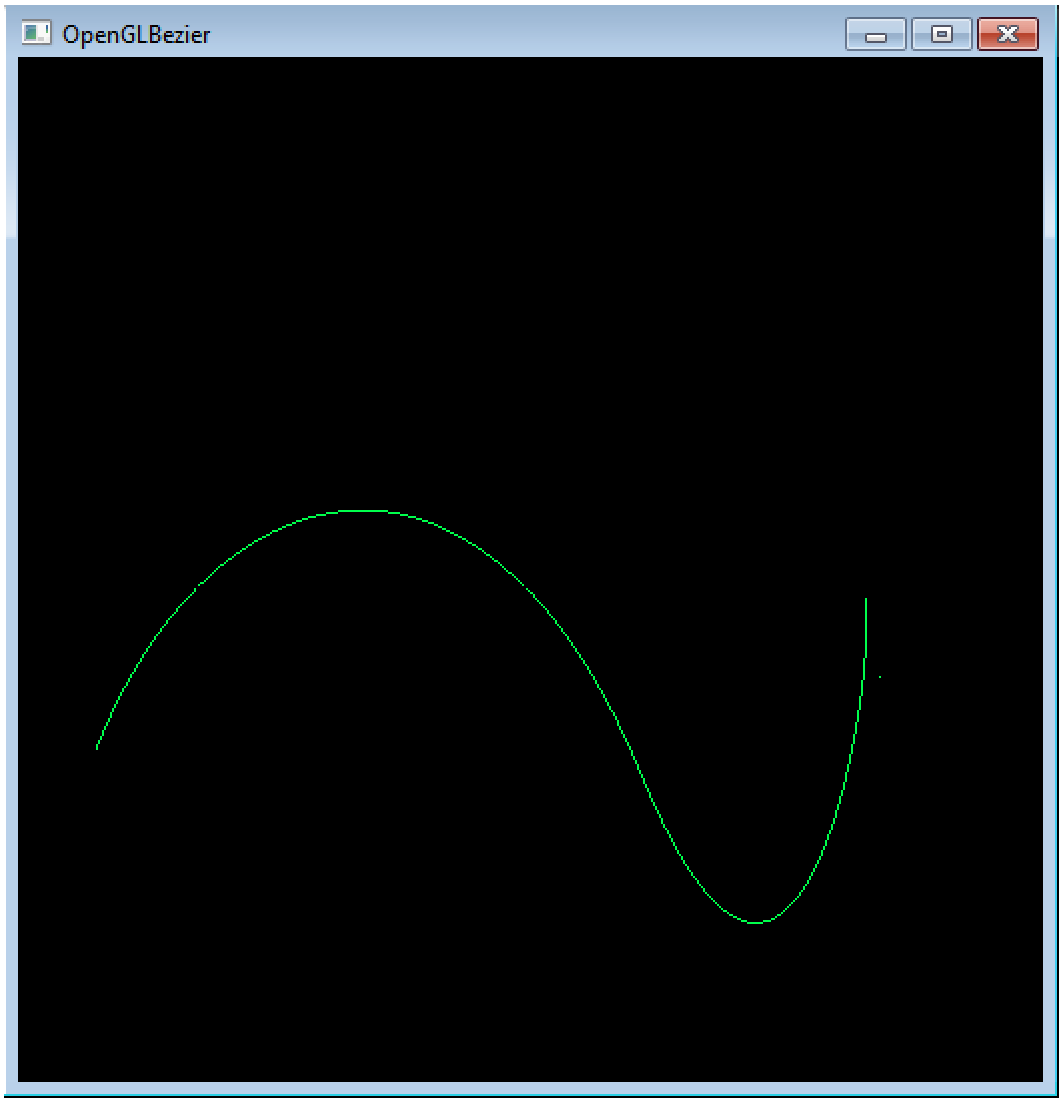


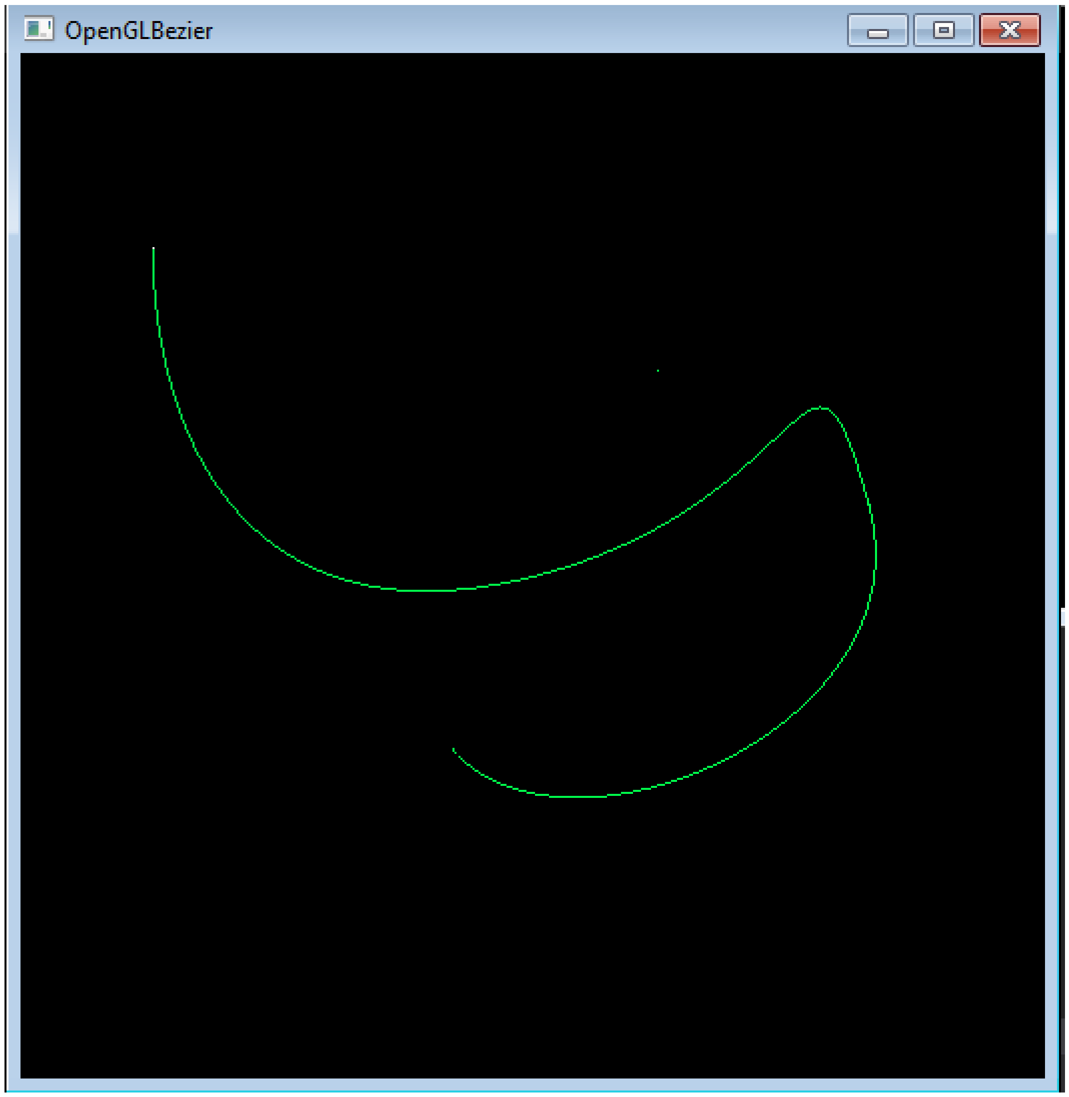
- Bezier curve using De Casteljau's algorithm: Here, I use the De Casteljau's algorithm to calculate points on the curve for various values of the parameter t [0,1]. It gives a pretty good result for step size of 0.001 (increment t by step size for each iteration). Here are the screenshots for 2 configurations. The dots are control points.


De Casteljau

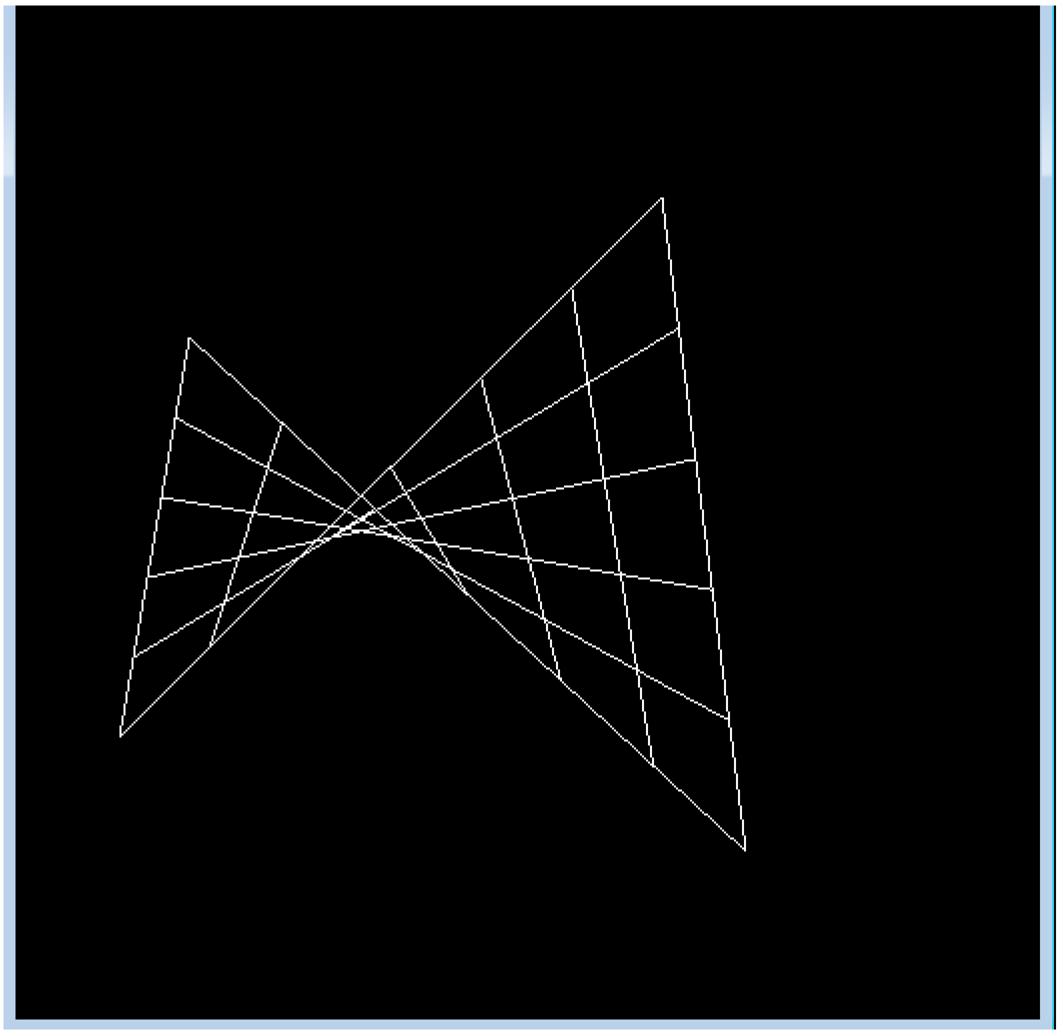


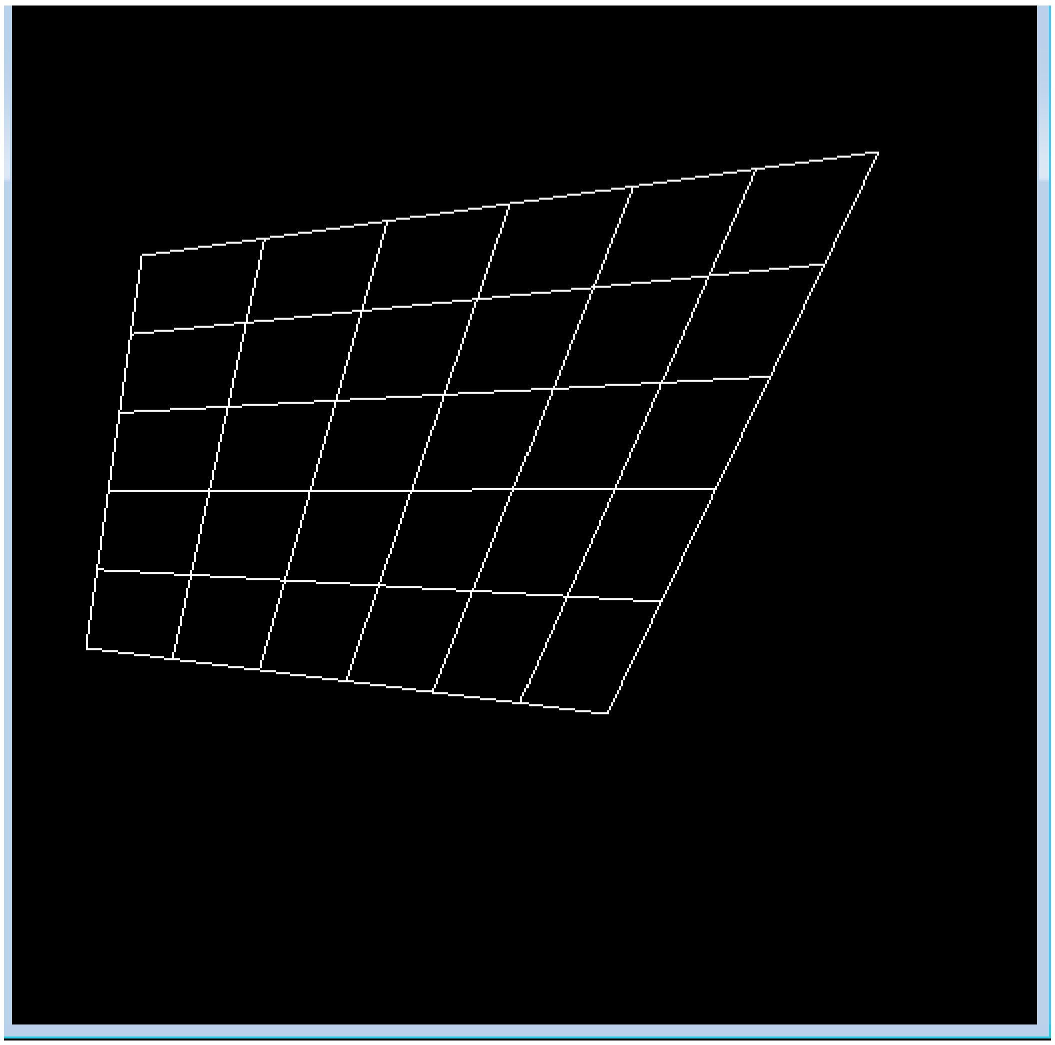
- Bezier curves using OpenGL API : Here , I use the OpenGL API glMap1f (give control points and order of the curve and other parameters) and glEvalCoord1f (evaluates Bezier curve for various parameter values)to draw the Bezier curve. Here are the screenshots for 2 configurations. The dots are control points.


OpenGLBezier



- Bezier curve approximation using subdivision: Here I call the De Casteljau's algorithm recursively with t=0.5 to approximate the curve. User can walk through various values. (Couldn't paste screenshots because the screen was flickering on my laptop, doesn't happen on the lab computers)
- Bonus part: This code draws a Bezier patch (a grid) using these 4 control points (2x2). It uses glMap2f, glMapGrid2f to calculate number of grid rows and columns and glEvalMesh2 to evaluate the mesh. Source: https://www.opengl.org/archives/resources/code/samples/mjktips/grid/index.html
  Here are the screenshots for 2 configurations. The dots are control points.





- When a control point is repeated: The effects of a point being repeated depends on which point is repeated. If the points in the beginning are repeated like p0 and p1,

  B(t) = (1-t)*(1-t)*(1-t)*p0 + 3*(1-t)*(1-t)*t*p1 + 3*(1-t)*t*t*p2 + t*t*t*p3

  = (1-t)*(1-t)*(1+2t)p0 + 3*(1-t)*t*t*p2 + t*t*t*p3

  Thus the curve will be more influenced by the repeated point and will be bent towards it for initial values of t. similarly when last (and second to last)point is repeated , the curve will be more influenced by the last points and will be bent towards it for values of t closer to 1.

- Advantages and disadvantages of the various methods: Comparing the blending function method and the De Casteljau's algorithm, the De Casteljau's algorithm uses linear interpolation which is faster to calculate as compared to the blending functions which are of degree 3. Both the methods give similar results and there were no noticeable differences as such. The method using OpenGL API might be faster than my implementation of the above 2 methods because it might be internally optimized and may perform calculations in parallel (finding 2 points on a Bezier curve are independent tasks).