CS4102 Curves in Computer Graphics 160026526 10 March 2021

• Overview:
The aim of the practical was to implement and manipulate Bézier curves.

• Design:
<u>Basis of application</u>
The application will be implemented using Processing which allows for the creation of PVectors that stores points in a plane. It implements the draw method which continuously executes which allows for constant updates to the display. Processing also includes methods for creating shapes, lines and reading key and mouse inputs. Processing coordinates start from the top left of the screen towards the bottom right. The program runs in fullscreen with a recommended screen resolution of 1920x1080. A list will be used to hold all the PVectors which are used to represent the list of control points. Each control point is added through left clicks.by adding PVectors at the cursor position. Right clicking removes the most recent point added. The points are displayed in the draw method by plotting them as circles with 10 pixels for radius and labelled accordingly. The keys, l and u, are used to lock and unlock adding/removing points and the key s is used to display the point number and position.

<u>Convex Hull design</u>
The algorithm used for the creating the convex hull is the gift wrapping algorithm (Jarvis march algorithm) which takes the left most point and wraps points counterclockwise to form the convex hull. To find the next point after the left most point, a method to find the orientation of three points is used which checks if the orientation of the three are collinear, clockwise or counterclockwise. For three points (a, b, c), the orientation is checked by comparing the slope of line segments (a, b) and (b, c) which are represented as (b.y – a.y)/(b.x – a.x) and (c.y – b.y)/(c.x – b.x) respectively. If the first slope is larger than the second, the orientation is more clockwise so by equating both slopes, (b.y – a.y)/(b.x – a.x) = (c.y – b.y)/(c.x – b.x) becomes,

$$(b.y - a.y)*(c.x - b.x) - (c.y - b.y)*(b.x - a.x) = 0$$

And so, collinearity makes the equation equal 0, clockwise equal 1 and counterclockwise equal 2. Returning to the algorithm, the next point, l, after the left most point, k, is the point which is counterclockwise for any other point, i, in other words orientation(k, l, i) = 2 for all i. Point l is then added to the Hull and set as point k and the next point is set as l. This is repeated until all points in the convex hull are found. Since the inner loop visits every control point, n, and the outer loop repeats for every hull point, h, the time complexity is O(nh). The worst case time complexity will be O(n^2) when every control point is a hull point and the best case time complexity is O(n) when there are only 3 control points so the algorithm is output sensitive. Its space complexity is O(n).

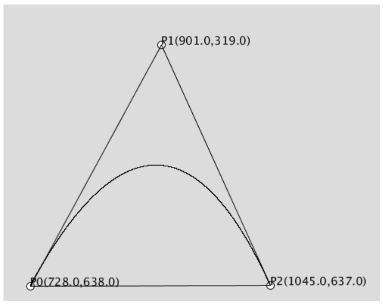
The output of the convexHull method produces the list of control points that form the convex hull in order. This list will only be updated when a new point is added or removed via clicks which is done using a Boolean pointUpdated as a pseudo switch. To visualise the hull, lines are drawn connecting each point of the hull with the next point.

Bézier curve
Using the formula for Bézier curves,

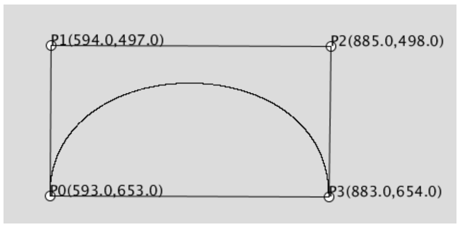
$$\mathbf{p}(u) = \sum_{i=0}^{n} \binom{n}{i} u^{i}(1-u)^{n-i}\,\mathbf{p}_{i+1}$$

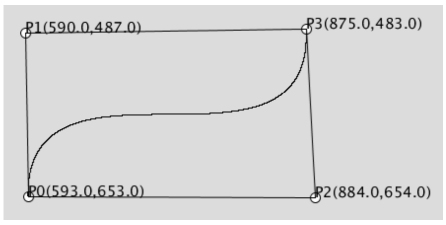
This is used to make the Bezier method that returns a PVector of a point in the Bézier curve based on the value of u from 0 to 1. The value of u is providing the position in the Bézier curve. To find the binomial coefficient, IntMath from google Guava was used. When constructing the Bézier curve, the u provided starts from 0 to 1 with increments of 0.0001 to simulate a smooth curve. The point in the Bézier curve is calculated using the Bezier method, which takes the curves order, u coefficient and an array of points, which plots the point as a dot. All the dots visually form the Bézier curve which is bounded by the convex hull with all the properties of a Bézier curve. When testing the Bézier curve, it is noticed that when more than 35 control points are used, the curve gets unstable and escapes from the convex hull. This may be due to the limitations of the binomial coefficient provided by Guava and is returning its MAXINT value.

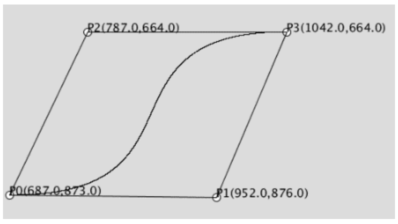
Screenshots of the application
• 3 control point Bézier curve



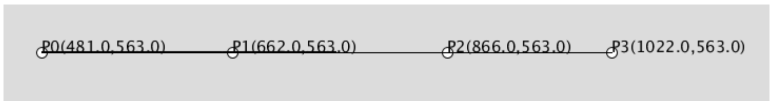
• Bézier curve drawn depending on 4 control points



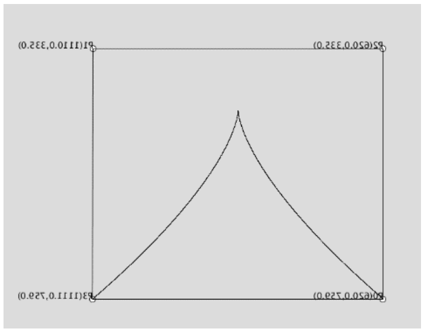




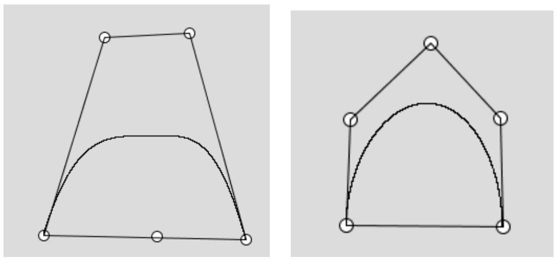
• 3 Collinear points makes the Bézier curve a straight line



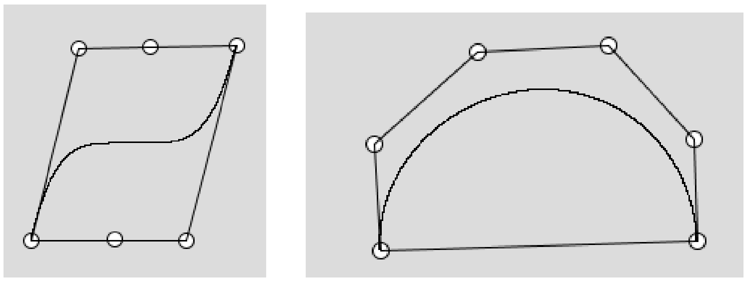
• Bézier curve with cusp



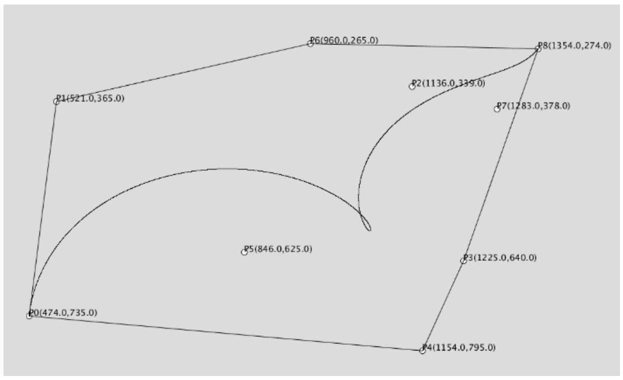
• 5 control point Bézier curve



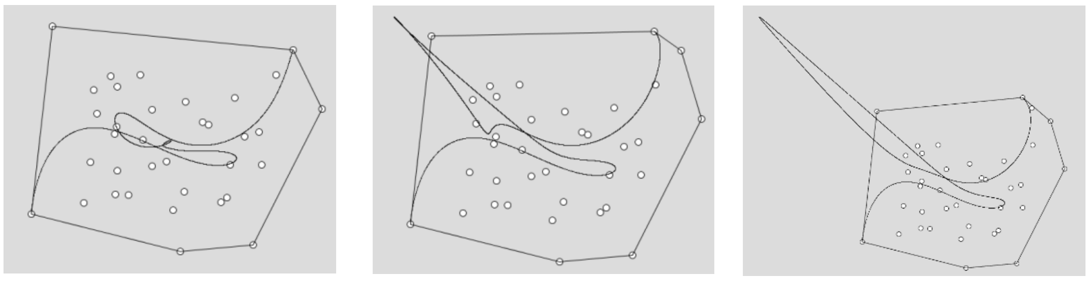
• 6 control point Bézier curve



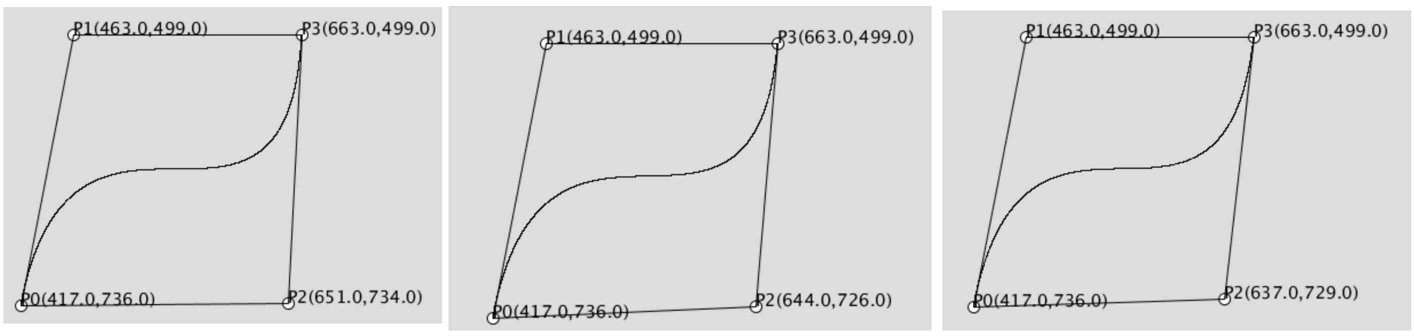
• 9 control point Bézier curve



• 35, 36, 37 control point Bézier curve



• Manual perturbation of P2 by a small amount



Perturbation attempt

Changes in control points require the whole curve to be recalculated due to the nature of the formula. It was attempted to perturb an interior point, but the implementation fell short due to the positioning of vectors in Processing. Attempting to take P2-P1 in some cases will lead to negative values that places the vectors outside of the application window. Therefore, a manual perturbation was attempted to examine the stability of the curve.

• Discussion and Evaluation:

The current convex hull algorithm used was the gift wrapping algorithm with a time complexity of O(nh) where n is the number of control points and h is the number of points in the hull. Other algorithms could be used such as the Graham scan or divide and conquer which have O(n log n) time complexities making them better than the gift wrapping algorithm. The gift wrapping algorithm is only faster if $h<\log n$. However, the gift wrapping algorithm can be extended into 3 dimensions where else the Graham scan cannot.

As for the Bézier curve algorithm, it can be optimized by simplifying the code for separate quadratic and cubic curves if curves of higher order are not needed which will speed up calculation and does not require the use of the binomials. Plotting the curve can get computationally expensive and as such another plotting method that could have been used is the de Casteljau's algorithm. Instead of calculating each x and y value for t, a recursive formula is used to plot lines based on the ratio of each line recursively until a single line is used to formulate the curve based of the ratio t from 0 to 1.

The code was written in a single pde file called Curves and exported as a windows and linux application.

• Conclusion:

In conclusion, the application was able to place and remove n number of points in the main application window which displays the points convex hull and the Bézier curve described by the control points. There was an attempt to allow for the perturbation of an interior control point but was unsuccessful, although it was attempted to examine the stability of the curve by manually perturbing an interior point.