160010723
CS4102

How to Run

The application has been packaged into a jar file, please type the command into the terminal to run:

```
java -jar P1.jar
```

Use the left mouse button to select points and the right button to draw all the lines. New points can be added to the lines in this state. Click the right mouse button again to clear the window and start again. Click the scroll wheel to show the intersection points.

Introduction

We were asked to produce an application that took a series of points and formed 3 separate curves. The Bezier Curve, the Bezier Spline and an Interpolating Cubic Polynomial Spline with C^1 continuity. A random point of intersection between the bezier curve and the interpolating spline should also be found and shown.

Implementation

Each line has a colour associated with it. Bezier curve: black. Bezier spline: blue. Interpolating Cubic Polynomial Spline: red. Interception: green.

Bezier Curve

This curve simply produces a sum of a polynomial equation based on Pascal's triangle over "time". Values from the control points of the curve were inserted to calculate x and y values. The degree to which Pascal's triangle is calculated is based on how many points are involved in forming the bezier curve.

Bezier Spline

A midpoint was selected between the second and third point of every four points. This was used as a control point that pulled the cubic bezier curves in such a way that it provided a smooth continuous transition between two segments, and thus, all segments.

Interpolating Cubic Polynomial Spline

It was required that a parametric spline be produced, as such a constant matrix was produced by inserting the values 0 and 1 into a cubic polynomial, and into the first and second derivatives such that the following rules were met:

$$p_n'(x_{n+1}) = p_{n+1}'(x_{n+1})$$

$$p_n''(x_{n+1}) = p_{n+1}''(x_{n+1})$$

C^1 and C^2 continuities were applied by the use of the first and second derivatives. The values resulting from the insertion of 0 and 1 into the polynomials and derivatives were then added to a square matrix of size (4 x number of segments).

This massive matrix was then inverted and multiplied with the series of known points to find the coefficients that would affect position of points. These coefficients were applied to a polynomial equation of the form:

$$x = a_0 + a_1t + a_2t^2 + a_3t^3$$

Where t represents the time at the point on the graph. And similarly for y with b_i coefficients.

The library Apache Commons Maths was used for matrix multiplication.

Intersection

The derivative was taken at a random point to produce a tangent at that point on the line. The perpendicular line to that point was formed using the gradient and points from the tangent. The roots intersections were found using the values from taking the 4 y coefficients from the 4 x coefficients of the spline segments and adding the y-intercept to the final coefficient. These roots were transformed into points using the same method as for the ICPS spline but replaced the t value for time with the root values.

Conclusion

In conclusion, the application implemented meets the requirements of the specification by creating all lines requested and finding the interceptions as required.

References

1. Java-programs.thiyagaraaj.com. n.d. Pascal Triangle Example Java Program - Java Programs. [online] Available at: <https://www.java-programs.thiyagaraaj.com/pascal_triangle_java_example_program.html> [Accessed 5 March 2020].
2. Kamermans, M., 2011. A Primer On Bézier Curves. [online] Pomax.github.io. Available at: <https://pomax.github.io/bezierinfo/> [Accessed 8 March 2020].
3. Christersson, M., 2014. How To Make A Cubic Bézier Spline. [online] Malin Christersson. Available at: <http://www.malinc.se/m/MakingABezierSpline.php> [Accessed 8 March 2020].
4. Holton, G., n.d. Cubic Spline Interpolation - Value-At-Risk: Theory And Practice. [online] Value-at-Risk. Available at: <https://www.value-at-risk.net/cubic-spline-interpolation/> [Accessed 9 March 2020].
5. GitHub. 2008. Davidzof/Wattzap. [online] Available at: <https://github.com/davidzof/wattzap/blob/master/src/com/wattzap/model/power/Cubic.java?fbclid=IwAR10V2Gxihz__DsMaPtVMcOhRW_cqJ8SuKRC2n5RBNjXgu2ark_Pjz4WfYA> [Accessed 13 March 2020].

Examples