CS4102 Computer Graphics – P1: Bézier Curves

170009629 - pmh20@st-andrews.ac.uk
School of Computer Science, University of St Andrews

1. INTRODUCTION

This practical was based around the concept of joining points in a 2-D space with a smooth transitioning line. Specifically, this was to be done by means of Bézier Curves. It was imperative that the user has as much control over these curves as possible, including control points (anchors and handles), sample frequency, and light positioning. The result should be a user-friendly GUI, which models accurately the mathematical processes that are happening behind the scenes.

3. SOFTWARE DESCRIPTION

The software follows the structure of a standard model – delegate GUI. This utilised OOP encapsulation and observers to ensure a robust piece of software was made. This however was not the focus, as the main attention was given to the mathematical models that were to be used to manipulate the geometry in any way specified by the user.

Along with the underlying methods that carried out the first principle maths, a lot of thought had to go in to how this data was stored, passed around, manipulated, and ultimately drawn to the screen for the visual pleasure of the user. The aim was to provide a fluid and enjoyable experience that was not restricted by fixed points, and the need to restart every time a change is desired.

In terms of how this application was designed, it was attempted to keep simplicity at the forefront, both for the user, and the marker (or anyone who must view the code). On startup, the user is presented with a blank canvas, with some initial parameters set to default. They have the option of selection the order of the Bézier curve (in simpler terms, the number of control points), and can immediately begin to draw them. There are no restrictions in the way these curves are drawn, other than the order they were programmed to go up to. These include linear, quadratic, and as an enhancement, cubic. One other restriction (in a sense) is the inability to click on the precise point of another control point that has already been placed, as unlikely as that would be anyway.

Once the user has finished placing their points, the curve draws itself on to the screen. Notice here that the number of points that determine the smoothness of the curve are drawn as small ovals. At any point, the user can update this parameter, specifying a desired sampling rate. This is represented as a percentage frequency (i.e. 0.02 means sampling a point every 2% along the curve).

Once happy with the shape, order, and smoothness of the drawn curve, the user has the ability to press the light button, and place it anywhere on the canvas. This will immediately illuminate all the legal sampling points along the curve (at the center point between 2 vertexes, i.e. the differentiable part where the surface normal can be calculated). This gives the semi-realistic effect of how real-life interaction of light and surfaces result in illuminance determined by angles. The user is then aloud to continue to move the light to any desired position, or return to drawing the curve (or reset it by selecting a new number of control points).

4. LIST OF PARTS

Basics:

- The user can select 3 control points.
- The application draws a Bézier Curve, defined by said control points.
- The user can specify a point light source.
- The curve is sampled uniformly (the number of samples specified by the user).
- Unit vectors of the normals and also the light components are multiplied using the dot product, and the illuminance is calculated.
- If the light hits a point on the right side (relative to the curve), then the sample brightness is simply zero. Otherwise the brightness is determined by how much of the light vector aligns with the normal (90° = none, 0° = full-illumination).
- Self-occlusion was accounted for, in all possible scenarios.

Extensions:

- Bezier curves were handled for different orders, i.e. 2 control points, 3 control points, and 4 control points.
- Additional functionality was focused on, such as the ability for the user to continually update the

light's position, with it updating the sampling points fluidly and quickly. This allows the user to essentially simulate what an animated light would do, by viewing the changes in illumination as they click it from point to point.

5. DESIGN DECISIONS AND JUSTIFICATIONS

One particular design decision that I feel was different to most people, was the way that self-occlusion was handled by the application. The usual way would probably be to use the 2 vectors to calculate the points at which the 2 lines would intersect. This could be used but would need additional constraints added to it, such as checking directionality (if intersections come from opposite directions they may both be allowed to be illuminated). Also there may be many more than 2 intersections, and additional checks must be made for distances and checking which lines fall behind others.

The approach I took was instead to use the four co-ordinates from the two lines to find if they cross each other. One of my lines would be from the sample to the light source, and the other would be an individual line than contributes to the overall curve (this is obviously looped through for each small line of the curve). Intersection is checked by comparing different combos of three points to essentially discover their orientation in relation to each other (CW/CCW/in line). If it ever was found to intersect, the point was drawn black, with no other need for additional checks.

6. CONCLUSION

The practical went successfully, with the only disappointment being running out of time when desiring to finish all enhancements (all of which had been thought about and planned for, with only other deadlines causing a hindrance). However, the software is successful in completing all the aims that were set out at the beginning, and it seems to be a robust and well organized program, that will be simple for me to go back to and continue at a later date. The assignment really helped reinforce the concepts of how the underlying mathematical equations are represented in geometry, which has been profoundly useful.

7. IMAGES