Implementation of Bezier Curve using OpenGL

Tarun Kumar
UE143103
Computer Science Engineering
University Institute of Engineering and Technology
Panjab University
Chandigarh, 160030
Email: tarunk.1996@gmail.com

***Abstract*—Building an OpenGL application to draw Bezier Curve from various control points provided by users. FreeGlut library is used in development of this application.**

I. Bezier Curve

The use of curved surfaces allows for a higher level of modeling, especially for the construction of highly realistic models. The mathematical method for drawing curves was created by Pierre Bezier in the late 1960's for the manufacturing of automobiles at Renault. To make a curved surface implementation of curves is necessary. A Bezier curve is a parametric curve frequently used in computer graphics and related fields. Bezier curves are used to model smooth curves. As the curve is completely contained in the convex hull of its control points, the points can be graphically displayed and used to manipulate the curve intuitively. Quadratic and cubic Bezier curves are most common. Higher degree curves are more computationally expensive to evaluate. When more complex shapes are needed, low order Bezier curves are patched together, producing a composite Bezier curve. A Bezier curve has an initial and a terminal point called anchors and middle points called handles.

Bezier curve is represented mathematically using the Bernstein polynomials as.

$$P(t) = \sum_{k=0}^{n} P_k BEZ_{k,n}(t) \tag{1}$$

where n is degree of the Bezier curve and $0 \leq t \leq 1$

$$BEZ_{k,n}(t) = C(n,k)(1-t)^{n-k}t^k \tag{2}$$

Here $BEZ_{k,n}(t)$ is the Bernstein polynomial and helps in rendering the curve.

II. Implementation

It is implemented using graphics library. First click on Bezier Curve button then, points are made using left clicks, first point is always the initial point of curve and the last point is the terminal point of curve and the rest are the handles of the curve. It is terminated by pressing 'B' key from the keyboard. Then lines are made between each adjacent points and the curve is also generated at the same time. It is implemented for n points but for higher number of points it will get computationally costlier which lead to slow rendering of the curve. So, it can be used for only few number of handles. Increment in the value of t is set to 0.02. Lower the increment value higher will be the smoothness in curve. The coefficients of the equation are generated using general formula. For computing factorial a separate function is made to compute. The curve and the lines are shown in blue colour.

III. About FreeGlut

FreeGLUT is a free-software/open-source alternative to the OpenGL Utility Toolkit (GLUT) library. GLUT was originally written by Mark Kilgard to support the sample programs in the second edition OpenGL 'RedBook'. Since then, GLUT has been used in a wide variety of practical applications because it is simple, widely available and highly portable.

GLUT (and hence FreeGLUT) takes care of all the system-specific chores required for creating windows, initializing OpenGL contexts, and handling input events, to allow for trully portable OpenGL programs.

FreeGLUT is released under the X-Consortium license.

IV. Some FreeGlut Functions

Following are functions used in developing the application with brief intro about it.

- **Creating A Window**
  The glutInitWindowPosition and glutInitWindowSize functions specify a desired position and size for windows that freeglut will create in the system. This glutCreateWindow("Line Drawing Algorithm") will create a Glut Window with a Title Line Drawing Algorithm on it.
- **Setting Background Color**
  This will set the background color of the screen to white.
  ```
  glClearColor(1, 1, 1, 0);
  glClear(GL_COLOR_BUFFER_BIT);
  glFlush();
  ```
- Plotting X and Y axis in the window. Set all the pixel along X-axis to black color and similar to that in Y-axis. (Remember according to FreeGlut co-ordinate system (0,0) will lie at the top-left corner of the window, So it should be shifted to the center of the screen first). In order to create X and Y - axis, you can either use inbuilt

line drawing function or can set each pixel along that line to color black. Both will gives the same result.

- **Showing Text**
  To display some text at any co-ordinate in the OpenGL window. It can be done by using the following code:

```
drawStrokeText(*string, x, y, z)
{
 char *c;
 glPushMatrix();
 glTranslatef(x, y+8,z);
 glScalef(0.09f,0.08f,z);
 for (c=string; *c != '\0'; c++)
 {
  glutStrokeCharacter(GLUT_STROKE_ROMAN,*c);
 }
 glPopMatrix();
}
```

  In the above function drawStrokeText accepts the string, x, y and z co-ordinates. It will print the string as in character by character format. $GLUT_STROKE_ROMAN$ is the text format that will be displayed on the screen.

- **Creating a Button**
  FreeGlut library as such doesnt include any provision for creating a clickable buttons and associating an action when a button is clicked. So in order to create a button, Create a button look alike box which will perform some action whenever there will a left mouse click on it. Following code will create a box in the glut screen of 80x25 size with a background color of red. GL_QUADS is used to create a quadrilateral in the OpenGL window.

```
glBegin(GL_QUADS);
glColor3f(1,0, 0);
glVertex2i(0,0);
glVertex2i(80,0);
glVertex2i(0,25);
glVertex2i(80,25);
glEnd();
drawStrokeText("Button",5,0,0);
```

  Now, to make this box a fully functional button. Mouse Left click event must be introduced in it.

```
mouse(int btn, int state, int x, int y)
  if(btn == GLUT_LEFT_BUTTON
&& state == GLUT_DOWN)
//Do some event
```

- **OpenGL Color Functions** To set the color display mode glut initialize display mode is used :

```
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB)
```

  Setting the color of a pixel can be done by 3 methods: Floating point values

```
glColor3f(0.0,1.0,0.0)
```

  An array specification

```
glColor3fv(colorArray)
```

  Integer Values

```
glColor3i(0,255,0)
```

- **Get properties of certain pixel** To read the seed pixel and extract the properties out of it is can be done by using $glReadPixels()$ function. Syntax of $glReadPixels()$

```
glReadPixels(X,Y,W,H,format,type,data);
```

  Here, X, Y specifies the window coordinates of the first pixel that is read from the frame buffer. W, H specifies the dimensions of the pixel rectangle, width and height of one correspond to a single pixel. $format$ specifies the format of the pixel data. $type$ specifies the data type of the pixel data. $data$ returns the pixel data(can be used to return the color of the seed pixel).

V. Outputs

Fig. 1. Bezier Curve from 2 control points

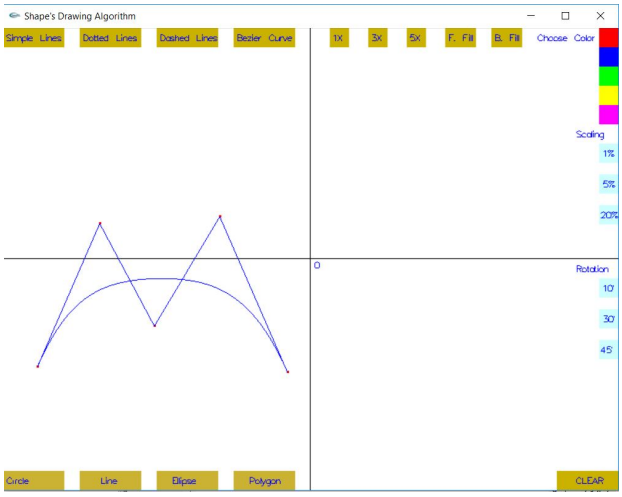

Fig. 2. Bezier Curve from 3 control points