CSE520
Samuel Marrujo
Professor Yu
Lab 15

Use the program and to find the knot vector

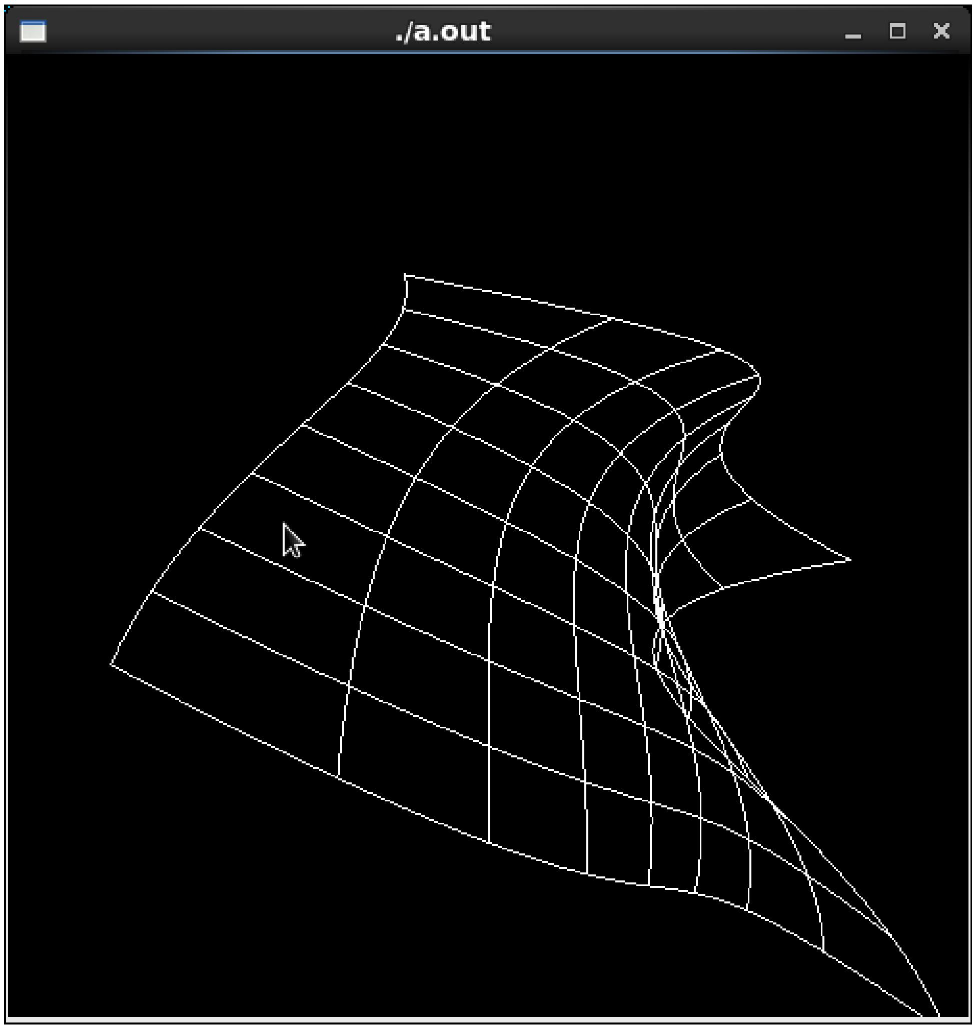
In this lab, we are to use the program given and find the knot vector of a given 3-order B-spline with 7 control points. I have completed the task:

A knot vector of 7 control points, order 3 is the following:

u0 = u1 = u2 = 0

```
u3 = 1
u4 = 2
u5 = 3
u6 = 4

u7 = u8 = u9 = 5
```



Code:

```
/*  bezsurf.c
 *  This program renders a wireframe Bezier surface,
 *  using two-dimensional evaluators.
 */
#include <GL/glut.h>
#include <stdlib.h>

GLfloat ctrlpoints[4][4][3] = {
   {{-2.5, -1.5, 5.0}, {-0.5, -1.5, 3.0},
    {-0.5, -1.5, -1.0}, {1.5, -2.5, 2.0}},

   {{-1.5, -0.5, 1.0}, {-0.5, -0.5, 3.0},
    {0.5, -0.5, 2.0}, {1.5, -0.5, -1.0}},

   {{-3.5, 0.5, 5.0}, {-0.5, 0.5, 2.0},
    {0.5, 0.5, -1.0}, {2.5, 0.5, 5.0}},

   {{-2.5, 1.5, -2.0}, {-0.5, 1.5, -2.0},
    {0.5, 2.5, 0.0}, {1.5, 1.5, -1.0}}
};

void display(void)
{
   int i, j;

   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glColor3f(1.0, 1.0, 1.0);
   glPushMatrix ();
   glRotatef(85.0, 1.0, 1.0, 1.0);
   for (j = 0; j <= 8; j++) {
      glBegin(GL_LINE_STRIP);
      for (i = 0; i <= 30; i++)
         glEvalCoord2f((GLfloat)i/30.0, (GLfloat)j/8.0);
      glEnd();
      glBegin(GL_LINE_STRIP);
      for (i = 0; i <= 30; i++)
         glEvalCoord2f((GLfloat)j/8.0, (GLfloat)i/30.0);
      glEnd();
   }
   //glEvalMesh2( GL_LINE, 0, 20, 0, 20 );
   glPopMatrix ();
   glFlush();
}

void init(void)
{
   glClearColor (0.0, 0.0, 0.0, 0.0);
   /*
    GL_MAP1_VERTEX_3 -- specifies that 3-dimensional control points are
                  provided and 3-D vertices should be produced
    0   -- min u
    1   -- max u
    3   -- ustride, number of values to advance in the data between two
         consecutive control points ( see diagram below )
    4   -- order of u ( = degree + 1 )
    0   -- min v
    1   -- max v
    12  -- vstride ( see diagram below )
    4   -- order of v ( = degree + 1 )
   */
   glMap2f(GL_MAP2_VERTEX_3, 0, 1, 3, 4,
           0, 1, 12, 4, &ctrlpoints[0][0][0]);
   glEnable(GL_MAP2_VERTEX_3);
   /*
    glMapGrid2f() -- defines a grid going from u1 to u2; v1 to v2 in
                 evenly-spaced steps
    20  --  number of u steps
    0.0 --  u1 ( starting u )
    1.0 --  u2 ( ending u )
    20  --  number of v steps
    0.0 --  v1 ( starting v )
    1.0 --  v2 ( ending v )

    use with glEvalMesh2()
   */
   //glMapGrid2f(20, 0.0, 1.0, 20, 0.0, 1.0);
   glEnable(GL_DEPTH_TEST);
   glShadeModel(GL_FLAT);
}

void reshape(int w, int h)
{
   glViewport(0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   if (w <= h)
      glOrtho(-4.0, 4.0, -4.0*(GLfloat)h/(GLfloat)w,
              4.0*(GLfloat)h/(GLfloat)w, -4.0, 4.0);
   else
      glOrtho(-4.0*(GLfloat)w/(GLfloat)h,
              4.0*(GLfloat)w/(GLfloat)h, -4.0, 4.0, -4.0, 4.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)
{
   switch (key) {
      case 27:
         exit(0);
         break;
   }
}

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow (argv[0]);
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutKeyboardFunc(keyboard);
   glutMainLoop();
   return 0;
}
```