```
 /* curvas de Bézier Definidas pela posição dos pontos extremos e
utilizando dois pontos adicionais para definir
indirectamente as tangentes à curva nas suas extremidades;*/
#include<windows.h>
#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

#include <stdlib.h>
//  define the control points of a Bezier curve of degree 3
   /* GLfloat ctrlpoints[5][3] = {
   { 2.0, 1.0, 0.0}, { 1.5, 4.0, 0.0},
   {4.0, 5.0, 0.0}, {4.5, 2.0, 0.0},
   {3.5, 2.0, 0.0}};*/
/*CRIA OS PONTOS DE CONTROLE*/
    GLfloat ctrlpoints[6][3] = {
    { 2.0, 2.0, 0.0}, { 2.0, 3.0, 0.0},
    {4.0, 3.0, 0.0}, {4.0, 1.0, 0.0},
    {2.0, 1.0, 0.0},{2.0, 2.0, 0.0}};

  int showPoints = 1;
  void init(void)
  {
  glClearColor(0.0, 0.0, 0.0, 0.0);
  glShadeModel(GL_FLAT);

  //define the evaluator for the Bezier curve and enable the evaluator
  //The points are 3D points, the mapping generates a 2D curve,
//the range of the parameter is from 0 to 1, each array position
//has three values (x, y, z), there are 4 points, and the 4 points
//are in the ctrlpoints array.
/*CRIA UMA MATRIZ DE CÁLCULO(FUNÇÃO PARA A CURVA).*/
glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 6, &ctrlpoints[0][0]);
/*HABILITA O EVALUATOR*/
glEnable(GL_MAP1_VERTEX_3);
}

void desenha(void)
{
int i;

glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0, 1.0, 1.0);

//plot the Bezier curve using the evaluator set up in
//the init method. Evaluate the curve at t=0, t=1/30,
//t = 2/30 . . .
/*FAZ O DESENHO E AVALIA OS PONTOS A CADA INSTANTE*/
glBegin(GL_LINE_STRIP);
for (i = 0; i <= 30; i++)
glEvalCoord1f((GLfloat) i/30.0);
glEnd();

//plot the same points from above in red but use a different method
glPointSize(5.0);
// The following code displays the control points as yellow dots.
glColor3f(1.0, 1.0, 0.0);
glBegin(GL_POINTS);
for (i = 0; i < 6; i++)
```

```
 glVertex3fv(&ctrlpoints[i][0]);
 glEnd();

 if (showPoints) {
 glPointSize(5.0);
 glColor3f(1.0, 1.0, 0.0);
 glBegin(GL_POINTS);
 for (i = 0; i < 6; i++) {
 glVertex3f(ctrlpoints[i][0],
 ctrlpoints[i][1], ctrlpoints[i][2]);
}
 glEnd();

 glLineWidth(1.0);
 glColor3f(1.0, 1.0, 1.0);
 glBegin(GL_LINE_STRIP);
 for (i = 0; i < 6; i++) {
 glVertex3f(ctrlpoints[i][0],
 ctrlpoints[i][1], ctrlpoints[i][2]);
 }
 glEnd();
}

 glFlush();
 }

 void keyboard(unsigned char key, int x, int y)
 {
 switch (key) {
 case 'c':
 case 'C':
```

```
showPoints = !showPoints;
glutPostRedisplay();
break;
case 27:
exit(0);
break;
default:
break;
}
}

 void resize(int w, int h)
{
glViewport(0, 0, (GLsizei) w, (GLsizei) h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0, 6, 0, 6);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}

int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(100, 100);
glutCreateWindow("Exemplo de curvas - CG 2016");
init();
glutDisplayFunc(desenha);
glutReshapeFunc(resize);
```

```
glutKeyboardFunc (keyboard);
glutMainLoop();
return 0;
}
```