```c
// Superfície de Bézier
#include <stdio.h>
#include <math.h>
#include <GL/glut.h>

#define LARGURA 500
#define ALTURA 500
// Uma granularidade muito pequena pode tornar a geração da Grade lenta(min 0.001)
#define GRANULARIDADE 0.05f

double pt[1001][1001][3]; // 3M*sizeof(double)
double p[4][4][3];
int indice;
double ax,ay,az;

GLint cx,cy;

void reseta()
{
    // define a rotação inicial dos objetos
    ax = 257.0f; // valor arbitrário
    ay = 0.0f;
    az = 25.0f; // valor arbitrários
    // retorna os pontos centrais a posição de origem
    p[1][1][2] = 1.0f;
    p[1][2][2] = 1.0f;
    p[2][1][2] = 1.0f;
    p[2][2][2] = 1.0f;
}

double fatorial(int a)
{
    int i;
    double produto;

    produto = 1.0f;
    for (i=a;i>0;i--)
    produto *= (double) i;
    return produto;
}

double combinacao(int a, int b)
{
    return fatorial(b)/(fatorial(a)*fatorial(b-a));
}

void geraGrade()
{
    GLint i,j;
    GLdouble u,v,x,y,z,vx,vy,vz,acx,acy,acz;

    cx=0;
    cy=0;
    for (u=0.0f;u<=1.01f;u+=GRANULARIDADE)
    {
        cy = 0;
        for (v=0.0f;v<=1.01f;v+=GRANULARIDADE)
        {
            acx = 0;
            acy = 0;
            acz = 0;

            for (i=0;i<4;i++)
            {
                vx = 0;
                vy = 0;
```

```
                vz = 0;
                x = combinacao(i,3)*pow(u,i)*pow(1.0f-u,3-i);
                y = combinacao(i,3)*pow(u,i)*pow(1.0f-u,3-i);
                z = combinacao(i,3)*pow(u,i)*pow(1.0f-u,3-i);

                for (j=0;j<4;j++)
                {
                    vx += combinacao(j,3)*pow(v,3-j)*pow(1.0f-v,j)*p[i][j][0];
                    vy += combinacao(j,3)*pow(v,3-j)*pow(1.0f-v,j)*p[i][j][1];
                    vz += combinacao(j,3)*pow(v,3-j)*pow(1.0f-v,j)*p[i][j][2];
                }

                x *= vx;
                y *= vy;
                z *= vz;

                acx += x;
                acy += y;
                acz += z;
            }
            pt[cx][cy][0] = acx;
            pt[cx][cy][1] = acy;
            pt[cx][cy][2] = acz;
            cy++;
        }
        cx++;
    }
}

void Inicia()
{

    p[0][0][0] = -0.75f; // ponto das extremidades 1
    p[0][0][1] = 0.75f;
    p[0][0][2] = 0.0f;
    p[0][1][0] = -0.25f;
    p[0][1][1] = 0.75f;
    p[0][1][2] = 0.0f;
    p[0][2][0] = 0.25f;
    p[0][2][1] = 0.75f;
    p[0][2][2] = 0.0;
    p[0][3][0] = 0.75f; // ponto das extremidades 2
    p[0][3][1] = 0.75f;
    p[0][3][2] = 0.0f;

    p[1][0][0] = -0.75f;
    p[1][0][1] = 0.25f;
    p[1][0][2] = 0.0f;
    p[1][1][0] = -0.25f; // ponto do meio 1
    p[1][1][1] = 0.25f;
    p[1][1][2] = 1.0f;
    p[1][2][0] = 0.25f;  // ponto do meio 2
    p[1][2][1] = 0.25f;
    p[1][2][2] = 1.0f;
    p[1][3][0] = 0.75f;
    p[1][3][1] = 0.25f;
    p[1][3][2] = 0.0f;

    p[2][0][0] = -0.75f;
    p[2][0][1] = -0.25f;
    p[2][0][2] = 0.0f;
    p[2][1][0] = -0.25f;  // ponto do meio 3
    p[2][1][1] = -0.25f;
    p[2][1][2] = 1.0f;
    p[2][2][0] = 0.25f;  // ponto do meio 4
    p[2][2][1] = -0.25f;
```

```
    p[2][2][2] = 1.0f;
    p[2][3][0] = 0.75f;
    p[2][3][1] = -0.25f;
    p[2][3][2] = 0.0f;

    p[3][0][0] = -0.75f; // ponto das extremidades 3
    p[3][0][1] = -0.75f;
    p[3][0][2] = 0.0f;
    p[3][1][0] = -0.25f;
    p[3][1][1] = -0.75f;
    p[3][1][2] = 0.0f;
    p[3][2][0] = 0.25f;
    p[3][2][1] = -0.75f;
    p[3][2][2] = 0.0f;
    p[3][3][0] = 0.75f; // ponto das extremidades 4
    p[3][3][1] = -0.75f;
    p[3][3][2] = 0.0f;

    reseta();
    geraGrade();
}

void exibe(void)
{
    GLint i,j;
    GLint u,v;

    glEnable(GL_DEPTH_TEST);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    glOrtho(-1.4f,1.4f,-1.4f,1.4f,-10.0f,10.0f);

    glMatrixMode(GL_MODELVIEW);

    glLoadIdentity();
    glRotatef(ax,1,0,0);
    glRotatef(ay,0,1,0);
    glRotatef(az,0,0,1);

    glColor3f(1.0f,1.0f,0.0f);
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);

    for (i=0;i<3;i++)
    {
        glBegin(GL_QUAD_STRIP);
        for (j=0;j<4;j++)
        {
            glVertex3f(p[i][j][0],p[i][j][1],p[i][j][2]);
            glVertex3f(p[i+1][j][0],p[i+1][j][1],p[i+1][j][2]);
        }
        glEnd();
    }
    glColor3f(0.0f,1.0f,0.0f);
    for (u=0;u<cx-1;u++)
    {
        glBegin(GL_QUAD_STRIP);
        for (v=0;v<cy;v++)
        {
            glVertex3f(pt[u][v][0],pt[u][v][1],pt[u][v][2]);
            glVertex3f(pt[u+1][v][0],pt[u+1][v][1],pt[u+1][v][2]);
        }
        glEnd();
    }
    glFlush();
```

```
199     glutSwapBuffers();
200
201 }
202
203 void teclado(unsigned char tecla,int x,int y)
204 {
205     switch (tecla){
206     case 'x':
207         ax++;
208         break;
209     case 'y':
210         ay++;
211         break;
212     case 'z':
213         az++;
214         break;
215     case 'X':
216         ax--;
217         break;
218     case 'Y':
219         ay--;
220         break;
221     case 'Z':
222         az--;
223         break;
224     case 'r':
225         reseta();
226         geraGrade();
227         break;
228     case '1':
229         p[1][1][2]+=0.15;
230         geraGrade();
231         break;
232     case '2':
233         p[1][2][2]+=0.15;
234         geraGrade();
235         break;
236     case '3':
237         p[2][1][2]+=0.15;
238         geraGrade();
239         break;
240     case '4':
241         p[2][2][2]+=0.15;
242         geraGrade();
243         break;
244     case '5':
245         p[1][1][2]-=0.15;
246         geraGrade();
247         break;
248     case '6':
249         p[1][2][2]-=0.15;
250         geraGrade();
251         break;
252     case '7':
253         p[2][1][2]-=0.15;
254         geraGrade();
255         break;
256     case '8':
257         p[2][2][2]-=0.15;
258         geraGrade();
259         break;
260     }
261     glutPostRedisplay();
262 }
263
264 int main(int argc, char** argv)
```

```
{

    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGB);
    glutInitWindowSize(ALTURA,LARGURA);
    glutInitWindowPosition(20,20);
    glutCreateWindow("Desenhando uma superfície de Bézier");
    Inicia();
    glutDisplayFunc(exibe);
    glutKeyboardFunc(teclado);
    glutMainLoop();

    return 0;
}
```