

```

#include <GL/glut.h>
#include <stdlib.h>
#include <math.h>

typedef struct{
    float x;
    float y;
    float z;
}Punto;

Punto ctrlpoints[100];
GLint numPtos;
GLint densidad;
GLfloat xMax, yMax;

void init(void);
void display(void);
void reshape(int w, int h);
void keyboard(unsigned char key, int x, int y);
void mouse (int button, int state, int cx, int cy);

Punto evalua (float t);
float Bernstein (int n, int i, float t);
float factorial (int n);

void init(void) {
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glShadeModel(GL_FLAT);
    numPtos = 0;
    densidad = 1000;
}

void display(void) {
    int i;
    Punto aux;

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

    if (numPtos >= 2) {
        glBegin(GL_LINE_STRIP);
        for (i = 0; i <= densidad; i++) {
            aux = evalua((GLfloat) i/(GLfloat) densidad);
            glVertex3f(aux.x, aux.y, aux.z);
        }
        glEnd();
    }

    /* El siguiente codigo dibuja los puntos de control */
    glPointSize(5.0);
    glColor3f(1.0, 1.0, 0.0);
    glBegin(GL_POINTS);
    for (i = 0; i < numPtos; i++)
        glVertex3f(ctrlpoints[i].x, ctrlpoints[i].y, ctrlpoints[i].z);
    glEnd();

    glFlush();
}

void reshape(int w, int h) {
    glViewport(0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h) {
        glOrtho(-5.0, 5.0, -5.0*(GLfloat)h/(GLfloat)w,
            5.0*(GLfloat)h/(GLfloat)w, -5.0, 5.0);
        xMax = 5.0;
        yMax = 5.0*(GLfloat)h/(GLfloat)w;
    } else {
        glOrtho(-5.0*(GLfloat)w/(GLfloat)h,
            5.0*(GLfloat)w/(GLfloat)h, -5.0, 5.0, -5.0, 5.0);
        xMax = 5.0*(GLfloat)w/(GLfloat)h;
    }
}

```

```

        yMax = 5.0;
    }
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y) {
    switch (key) {
        case 27:
            exit(0);
            break;
        case 8:
            if (numPtos > 0)--numPtos;
            break;
    }

    glutPostRedisplay();
}

void mouse (int button, int state, int cx, int cy) {

    float x, y;

    /* Calculamos las coordenadas de mundo (relativas) apartir de
     * las de ventana (absolutas) */
    x = ((GLfloat)cx * 2.0 * xMax) / glutGet(GLUT_WINDOW_WIDTH) - xMax;
    y = yMax - ((GLfloat)cy * 2.0 * yMax) / glutGet(GLUT_WINDOW_HEIGHT);

    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        ctrlpoints[numPtos].x = x;
        ctrlpoints[numPtos].y = y;
        ctrlpoints[numPtos].z = 0.0;
        ++numPtos;
    }

    glutPostRedisplay();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (500, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow (argv[0]);
    init();

    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutMouseFunc(mouse);
    glutMainLoop();

    return 0;
}

//===== Las Matematicas de Bezier =====
Punto evalua (float t) {
    int i;
    float coeficiente;
    Punto vertice;

    vertice.x = 0.0f;
    vertice.y = 0.0f;
    vertice.z = 0.0f;

    for (i = 0; i < numPtos; i++) {
        coeficiente = Bernstein(numPtos - 1, i, t);
        vertice.x += ctrlpoints[i].x * coeficiente;
        vertice.y += ctrlpoints[i].y * coeficiente;
    }
}

```

```
        vertice.z += ctrlpoints[i].z * coeficiente;
    }
    return vertice;
}

float Bernstein (int n, int i, float t) {
    int coeficiente;
    coeficiente = factorial(n) / (factorial(i) * factorial(n-i));
    return coeficiente * pow(t, (float)i) * pow((1.0 - t), (float)(n-i));
}

float factorial (int n) {
    int i;
    float resultado = 1.0;

    for (i = 1; i <= n; i++) {
        resultado *= i;
    }

    return resultado;
}
```