Comandos que utilizei para compilar e executar o programa:

1: g++ -c -Wall 202010140-EP03-Q02.cpp
2: g++ 202010140-EP03-Q02.o -lglut -lGLU -lGL -o <nome-do-executavel>
3: ./<nome-do-executavel>

Programa:

```
#include <GL/glut.h>
#include <ctime>
#include <cstdlib>
#include <iostream>
#include <unistd.h>

static int x0, x1, y0, y1;

bool rasterizar = false;

void algoritmoBresenham(int x0, int x1, int y0, int y1) {

   int dX = x1 - x0,
   dY = y1 - y0,
   ix = 1,
   iy = 1,
   e, x, y, i;

   if (dX < 0) {
     ix = -ix;
  }
   if (dY < 0) {
     iy = -iy;
  }

   dX = abs(dX);
   dY = abs(dY);
   x = x0;
   y = y0;

   if (dX > dY) {
       e = (dY << 1) - dX;
       glBegin(GL_POINTS);
        for (i = 0; i < dX; i++) {
           glVertex2i(x,y);
           if (e < 0) {
              e += dY << 1;
           }
           else {
              y += iy;
              e += (dY - dX) << 1;
           }

           x += ix;
        }
       glEnd();
   } else {
       e = (dX << 1) - dY;
       glBegin(GL_POINTS);

        for (i = 0; i < dY; i++) {
           glVertex2i(x,y);

           if (e < 0) {
              e += dX << 1;
           }
           else {
              x += ix;
              e += (dX - dY) << 1;
           }
           y += iy;
        }
     glEnd();
   }
}

void display(void) {
  glColor3f(0, 1.0, 0);
  glPushMatrix();
  glPointSize(2.8);
   glBegin (GL_LINES);
   glVertex2f (-0.5, 0.5);
   glVertex2f (0.5, -0.5);
   glEnd();
   glPopMatrix();

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(0.0, 600.0, 0.0, 600.0);
   glPopMatrix();
  glFlush();

  srand(time(NULL));
   x0 = rand() % 700;
   x1 = rand() % 700;
   y0 = rand() % 700;
   y1 = rand() % 700;

  if (!rasterizar) {
      algoritmoBresenham(x0, x1, y0, y1);
  }
   glFlush();
}

void keyboard(unsigned char key, int x, int y) {
  switch (key) {
       case 'e':
           if(!rasterizar) {
               rasterizar = true;
           } else {
               rasterizar = false;
           }
           glutPostRedisplay();
       default:
           break;
  }
}

void init(void) {
  GLfloat values[2];
  glGetFloatv (GL_LINE_WIDTH_GRANULARITY, values);
  printf ("GL_LINE_WIDTH_GRANULARITY value is %3.1f\n", values[0]);

  glGetFloatv (GL_LINE_WIDTH_RANGE, values);
  printf ("GL_LINE_WIDTH_RANGE values are %3.1f %3.1f\n",
           values[0], values[1]);

  glEnable (GL_LINE_SMOOTH);
  glEnable (GL_BLEND);
  glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
  glHint (GL_LINE_SMOOTH_HINT, GL_DONT_CARE);
  glLineWidth (1.5);

  glClearColor(0.0, 0.0, 0.0, 0.0);
}

void reshape(int w, int h) {
  glViewport(0, 0, w, h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  if (w <= h)
     gluOrtho2D (-1.0, 1.0,
        -1.0*(GLfloat)h/(GLfloat)w, 1.0*(GLfloat)h/(GLfloat)w);
  else
     gluOrtho2D (-1.0*(GLfloat)w/(GLfloat)h,
        1.0*(GLfloat)w/(GLfloat)h, -1.0, 1.0);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}

int main(int argc,char** argv) {
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize(800, 600);
   glutCreateWindow("EP04 - Vitor Melo");
   init();
   glutReshapeFunc (reshape);
   glClearColor(0.0, 0.0, 0.0, 0.0);
   glClear(GL_COLOR_BUFFER_BIT);
   glutKeyboardFunc(keyboard);
   glutDisplayFunc(display);
   glutIdleFunc(display);
   glutMainLoop();
}
```