Departamento de Matemática

FCTUC

Relatório

Visualização Computacional

Professor: José Carlos Teixeira (teixeira@mat.uc.pt)

Trabalho 1 – Editor de curvas

Autores:
Ana Catarina Quitério Lourenço
catarinaql@gmail.com
Israel Campiotti
israelcampiotti@gmail.com

Data:
06-11-2016

ÍNDICE

1. SUMÁRIO

O presente trabalho tem por objectivo a utilização dos recursos disponibilizados pela linguagem C e pelo OpenGL no desenvolvimento de um editor de curvas, considerando, pelo menos, as seguintes curvas: astroide, cardioide, epicicloide e lemniscata de Bernoulli.

O editor deve permitir definir e visualizar as curvas e alterar os seus atributos de visualização (cor das curvas, espessura da linha, raio, centro, etc.). Deve, ainda, ser original e ergonómico, disponibilizando instruções que facilitem a sua utilização.

O trabalho foi realizado recorrendo às várias bibliotecas da linguagem C e, especialmente, à biblioteca glut.h.

O presente relatório explica pormenorizadamente a construção do programa criado e o modo de utilização do mesmo.

2. INTRODUÇÃO

Este trabalho tem por objectivo utilizar os recursos disponibilizados pela linguagem C e pelo OpenGL na implementação de um programa modular e eficiente que funcione como um editor de curvas, considerando, pelo menos, as seguintes curvas:

a. Astroide, com equações paramétricas

$$x = \frac{r}{4}\,(3\cos t + \cos 3t)$$

$$y = \frac{r}{4}\,(3\sin t - \sin 3t)$$

b. Cardioide, com equações paramétricas

$$x = r\,(2\cos t - \cos 2t)$$

$$y = r\,(2\sin t - \sin 2t$$

c. Epicicloide, com equações paramétricas

$$x = r\,(k+1)\cos t\; - r\cos\,((k+1)t)$$

$$y = r\,(k+1)\sin t\; - r\sin\,((k+1)t)$$

d. Lemniscata de Bernoulli, com equações paramétricas

$$x = \frac{r\sqrt{2}\cos t}{sin^2(t)\; + 1}$$

$$y = \frac{r\sqrt{2}\cos t\sin t}{sin^2(t)\; + 1}$$

O editor deve permitir definir e visualizar as curvas e alterar os seus atributos de visualização (cor das curvas, espessura da linha, raio, centro, etc.) e deve ser ergonómico, tendo a ajuda necessária para uma fácil utilização, e original.

3. PROGRAMA

O editor de curvas foi construído através da criação das bibliotecas “axes.h”, “auxf.h” e “curves.h” e do programa “newmain.c”.

3.1 “axes.h”

Declaração de variáveis globais

```
//
//  Variaveis Globais ////////////////////////////////////////////////////////
//
double mult = 0.35;
```

Figura 1 - Variáveis globais de axes.h

- **mult** – variável do tipo *double* usada para definir o comprimento dos eixos em relação ao valor da variável normalizer.

Funções

- **void displayAxes(const double normalizer, const int n)** – imprime os eixos coordenados, com x ϵ [-mult*normalizer, mult*normalizer], y ϵ [-mult*normalizer, mult*normalizer], e cada semieixo dividido em n partes. Nesta função desenha-se as rectas dos eixos, as setas de cada eixo e as linhas que os dividem.

3.2 “curves.h”

Inclusão de bibliotecas necessárias

```
//
//  Bibliotecas a incluir ////////////////////////////////////////////////////
//
#include <GL\glut.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include "axes.h"
```

Figura 2 - Bibliotecas de curves.h

Definição de constantes

```
//
//  Definicoes ///////////////////////////////////////////////////////////////
//define pi e raiz quadrada de dois
#define PI   3.14159265359
#define sqr2 1.41421356237
```

Figura 3 - Constantes de curves.h

Declaração de variáveis globais

```
//
//  Variaveis Globais ////////////////////////////////////////////////////////
//variaveis globais para auxilio no codigo
double radius = 100.0, k = 4.0, nWidht = 1.2, nHeight = 0.7;
int normalizer, lineSize = 1;
char dCurv;
double cColors[4][3];
```

Figura 4 - Variáveis globais de curves.h

- **radius** – variável do tipo *double* que corresponde ao raio de uma curva de qualquer tipo. É inicializada atribuindo o valor 100;
- **k** – variável do tipo *double* que correnponde a uma das constantes das equações paramétricas do epicicloide. É inicializada atribuindo o valor 4;
- **nWidth** – variável do tipo *double* usada para definir a largura da janela em relação ao valor da variável normalizer. É inicializada atribuindo o valor 1.2;
- **nHeight** – variável do tipo *double* usada para definir a altura da janela em relação ao valor da variável normalizer. É inicializada atribuindo o valor 0.7;
- **normalizer** – número inteiro utilizado para definir a largura e altura da janela;
- **lineSize** – número inteiro que corresponde à espessura da linha das curvas. É inicializado atribuindo o valor 1;
- **dCurv** – variável do tipo *char* que corresponde ao tipo de curva a desenhar;
- **cColors[4][3]** – matriz do tipo *double* que define as cores utilizadas para desenhar os quatro tipos de curvas.

Funções

- **void astroide(const double radius, int Ox, int Oy, const int region)** – desenha um astroide de raio radius e centro (Ox, Oy) com a cor definida pela linha 0 de cColors. Apenas são desenhados os pontos que estão entre os limites do gráfico;
- **void cardioide(const double radius, int Ox, int Oy, const int region)** – desenha um cardioide de raio radius e centro (Ox, Oy) com a cor definida pela linha 1 de cColors. Apenas são desenhados os pontos que estão entre os limites do gráfico;
- **void epicloide(const double radius, const double k, int Ox, int Oy, const int region)** – desenha um epicicloide de raio radius e e constante k *double* k e centro (Ox, Oy) com a cor definida pela linha 2 de cColors. Apenas são desenhados os pontos que estão entre os limites do gráfico;
- **void bernoulli(const double radius, int Ox, int Oy, const int region)** – desenha uma lemniscata de Bernoulli de raio radius e centro (Ox, Oy) com a cor definida pela linha 3 de cColors. Apenas são desenhados os pontos que estão entre os limites do gráfico.

3.3 "auxf.h"

Protecção dos ficheiros de interface

```
//
//  Proteção do ficheiro de interface ////////////////////////////////////////
//
#pragma once
```

Figura 5 - Protecção dos ficheiros de interface em auxf.h

Funções

- **int clean_stdin** – lê caracteres até encontrar um parágrafo. Esta função é usada para ler e eliminar o excesso de caracteres introduzido pelo utilizador ou caracteres do tipo errado;
- **void displayText(const double x, const double y, const char *string, double r, double g, double b)** – imprime o texto *string na posição (x,y) com a cor [r g b];
- **void displayBlackRegion()** – imprime um quadrado preto do lado esquerdo da imagem, que corresponde ao gráfico

  {(x, y) : x ϵ [-mult*normalizer, mult*normalizer], y ϵ [-mult*normalizer, mult*normalizer]} **;**

- **void zeraVetor(int n, int*v)** – inicializa o vector *v, atribuindo zero a cada elemento;
- **void resetColor(double cColors[4][3])** – redefine as cores a usar em cada tipo de curva para as cores padrão;
- **void txtAstroide(double xi, double yi)** – imprime o texto "Astroide - a" na posição (xi*normalizer, yi*normalizer) e desenha um astroide de raio 60 com centro

((xi + 0.10)*normalizer, (yi-0.07)*normalizer) **,** usando a cor definida pela linha 0 de cColors;

- **void txtCardioide(double xi, double yi)** – imprime o texto "Cardioide - c" na posição (xi*normalizer, yi*normalizer) e desenha um cardioide de raio 40 com centro

((xi + 0.10)*normalizer, (yi-0.09)*normalizer) **,** usando a cor definida pela linha 1 de cColors;

- **void txtEpicloide(double xi, double yi)** – imprime o texto "Epicicloide - a" na posição (xi*normalizer, yi*normalizer) e desenha um epicicloide de raio 20 e k 4 com centro
- ((xi + 0.10)*normalizer, (yi-0.09)*normalizer), e um epicicloide de raio 20 e k 3.8 com centro

((xi + 0.28)*normalizer, (yi-0.09)*normalizer)**,** usando a cor definida pela linha 2 de cColors;

- **void txtAstroide(double xi, double yi)** – imprime o texto "Leminiscata de Bernoulli - b" na posição (xi*normalizer, (yi-0.05)*normalizer) e desenha uma lemniscata de Bernoulli de raio 60 com centro

((xi + 0.10)*normalizer, (yi-0.1)*normalizer) **,** usando a cor definida pela linha 3 de cColors;

- **void printaCurva(char cC, double radius)** – o utilizador indica o tipo de curva que pretende desenhar (introduzindo o caracter correspondente), as coordenadas do centro, o raio e a constante k (no caso do epicicloide), sendo desenhada a curva correspondente, caso a informação introduzida seja válida**.**

3.4 "newmain.c"

Inclusão das bibliotecas necessárias

```
//
//  Bibliotecas a incluir ////////////////////////////////////////////////////
//
#include "curves.h"
#include "auxf.h"
```

Figura 6 - Bibliotecas em newmain.c

Declaração de variáveis globais

```
//
//  Variaveis Globais ////////////////////////////////////////////////////////
//
int iColor = -1;
double xTxt = 0.4, yTxt = 0.3;
```

Figura 7 - Variáveis globais em newmain.c

- **iColor** – número inteiro que corresponde à linha da matriz cColor. É inicializada atribuindo o valor -1;
- **xTxt** – coordenada x da posição para imprimir informação sobre os tipos de curva;
- **yTxt** – coordeneda y da posição para imprimir informação sobre os tipos de curva.

Funções

- **void display()** – função vazia;
- **void init()** – imprime as condições iniciais da janela. Define o fundo branco, define as coordenadas globais como

{(x, y) : x ϵ [-mult*normalizer, normalizer*(nWidht - mult)], y ϵ [-mult*normalizer, mult*normalizer]} ,

usa as funções displayBlackRegion e displayAxes para imprimir o fundo preto do lado esquerdo da janela e desenhar os eixos, respectivamente, e usa as funções txtAstroide, txtCardioide, txtEpicloide e txtBernoulli para imprimir as informações sobre as curvas que o utilizador pode desenhar;

- **void controlaMouse(GLint button, GLint state, GLint x, GLint y)** – caso o utilizador carregue numa das teclas a que corresponde uma curva (a, c, e, b), é desenhada uma curva desse tipo

com centro na posição do gráfico em que o cursor se encontra quando se carrega no botão esquerdo do rato. Quando se carrega no botão direito do rato, todas as curvas desenhadas são apagadas;

- **void changeColor()** – permite que o utilizador introduza novos valores r, g, b para definir a cor de determinado tipo de curva. Esses valores são guardados na linha adequada para esse tipo de curva na matriz cColor;
- **void controlaKeyboard(unsigned char key, GLdouble x, GLdouble y)** – define a acção a ocorrer quando se pressiona uma tecla key correspondente a uma letra;
- **void controlaEspessura(GLint key)** – define a alteração relativa à espessura da linha da curva quando se pressiona uma tecla do tipo Fi, I = 1, 2, 3, 4;
- **int main(int argc, char**argv)** – imprime informação sobre o funcionamento do programa. Recorre às funções anteriores para permitir que o utilizador altere as cores, espessura da linha e o raio das curvas (e k, no caso do epicicloide), desenhe curvas indicando o tipo de curva e carregando no ponto do gráfico que quer usar como centro, desenhe curvas indicando o tipo de curva, as coordenadas do centro e o raio (e k no caso do epicicloide) e apague todas as curvas já desenhadas.

4. APRESENTAÇÃO DE UMA SESSÃO DE UTILIZAÇÃO

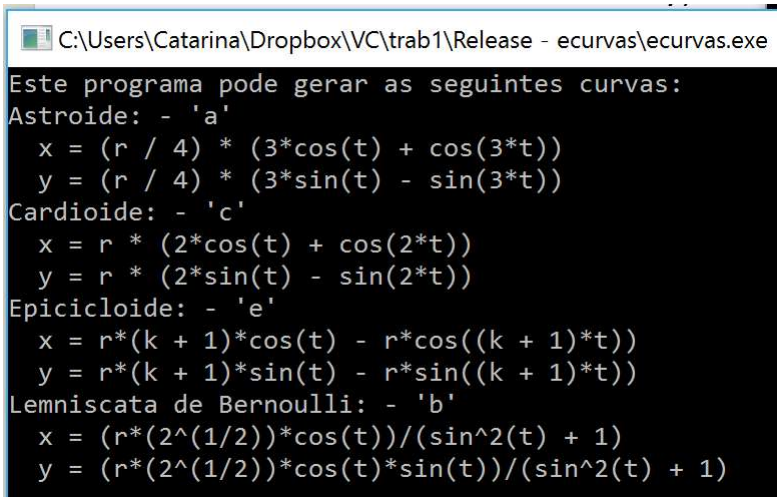

Figura 8 - Informação inicial sobre as curvas

Figura 9 - Informação sobre os comandos para alterar o raio e k

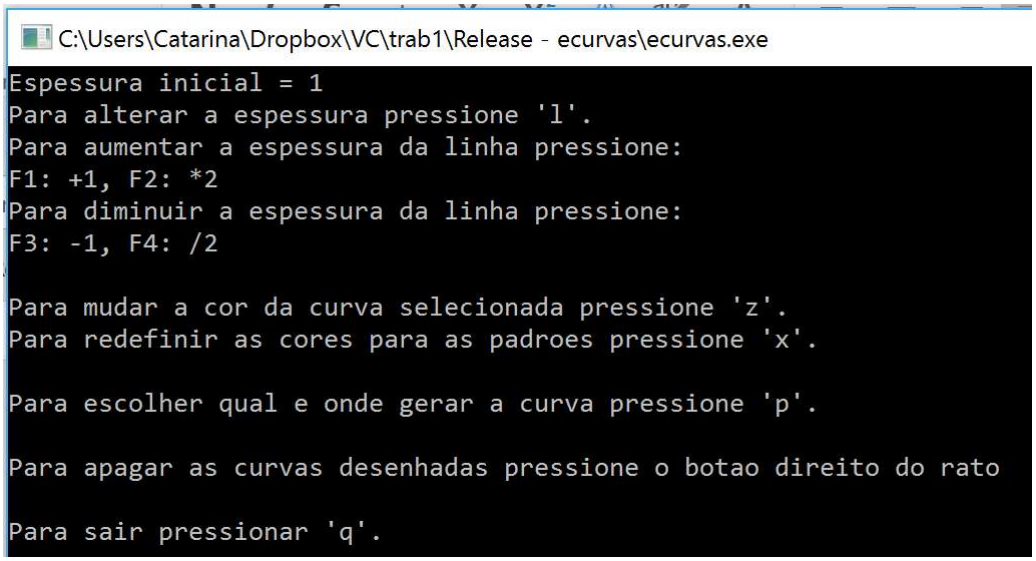

Figura 10 - Informação sobre os comandos para alterar a cor e a espessura da linha, apagar curvas desenhadas e sair do programa

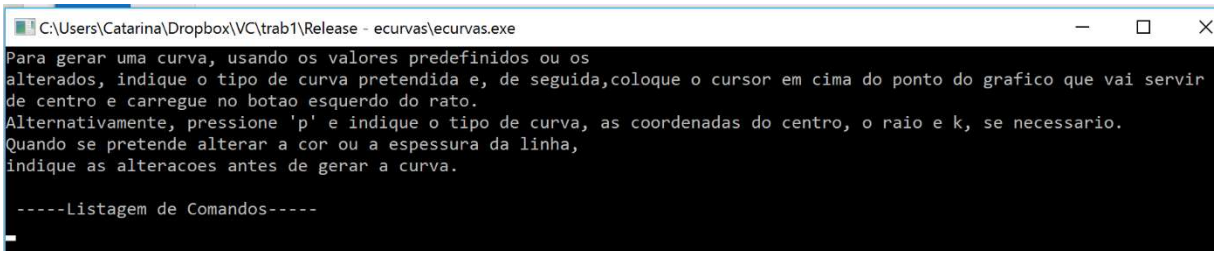

Figura 11 - Informação sobre o funcionamento do programa

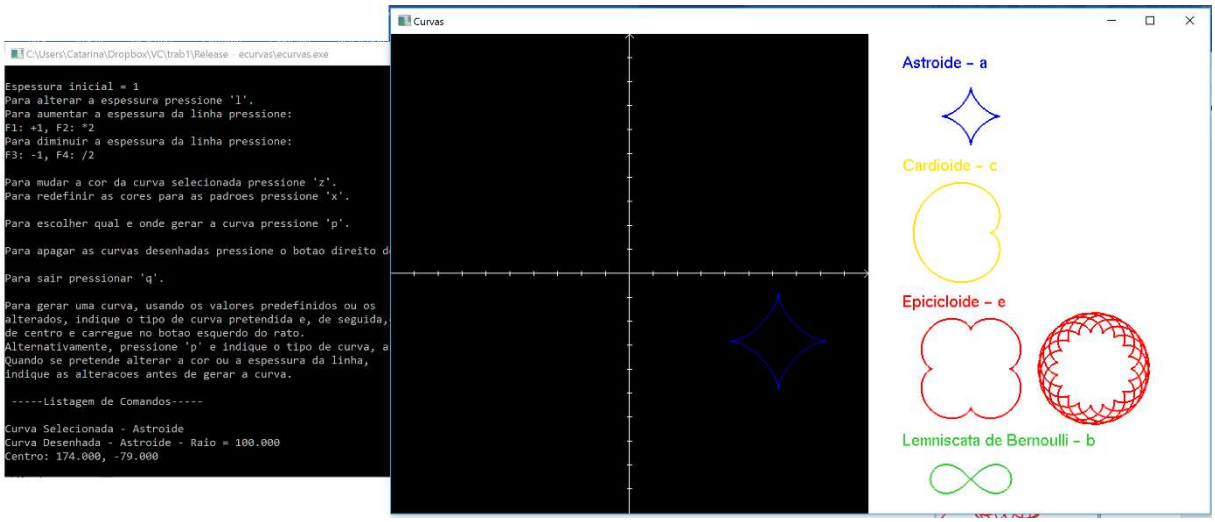

Figura 12 - Curva desenhada com os atributos predefinidos pressionando 'a' e carregando no botão esquerdo do rato em cima do centro

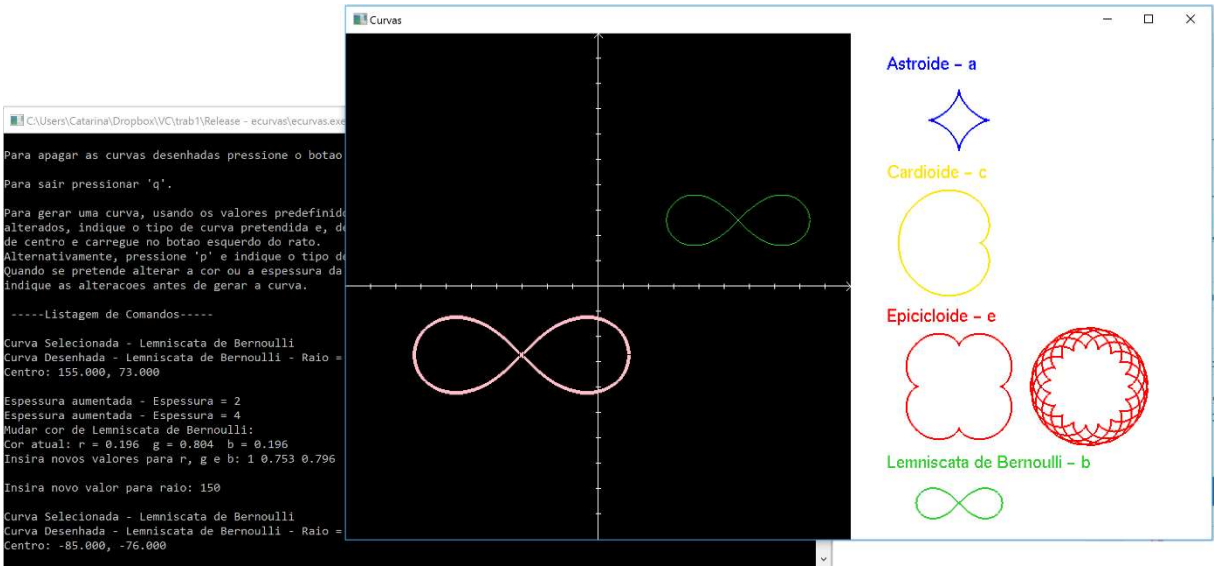

Figura 13 - Curva verde desenhada com os atributos predefinidos e curva rosa desenhada após os comandos: 'F2', 'F2', 'z', 'r', 'b'

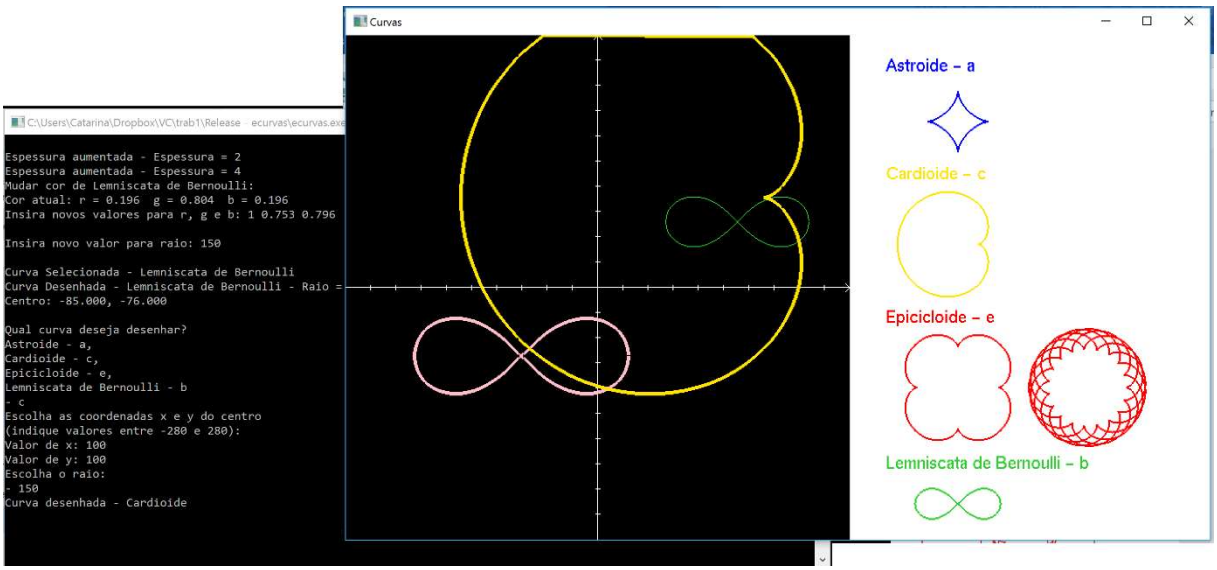

Figura 14 - Cardioide desenhado recorrendo ao comando 'p'

5. CONCLUSÕES

A realização deste trabalho permitiu a consolidação dos conhecimentos adquiridos nas aulas de Visualização Computacional, bem como o aprofundamento de algumas áreas do OpenGL, já que, até ao momento, foi o trabalho mais complexo em que tivemos que recorrer à biblioteca glut.h.

Um dos maiores desafios com que nos deparámos neste trabalho está relacionado com o facto de algumas funções da linguagem C que usamos frequentemente não terem o mesmo resultado quando utilizamos o Microsoft Visual Studio.

Consideramos que conseguimos cumprir, de forma original, todos os objectivos do trabalho, criando uma aplicação de fácil de utilização.

6. BIBLIOGRAFIA

Apontamentos das aulas

Wikipedia contributors, "Astroid", *Wikipedia, The Free Encyclopedia*
URL: https://en.wikipedia.org/w/index.php?title=Astroid&oldid=717759872 (Conferido em: 28 de Outubro de 2016)

Wikipedia contributors, "Cardioid", *Wikipedia, The Free Encyclopedia*
URL: https://en.wikipedia.org/w/index.php?title=Cardioid&oldid=744573112 (Conferido em: 28 de Outubro de 2016)

Wikipedia contributors, "Epicycloid", *Wikipedia, The Free Encyclopedia*
URL: https://en.wikipedia.org/w/index.php?title=Epicycloid&oldid=721339475 (Conferido em: 28 de Outubro de 2016)

Wikipedia contributors, "Lemniscate of Bernoulli", *Wikipedia, The Free Encyclopedia*
URL: https://en.wikipedia.org/w/index.php?title=Lemniscate_of_Bernoulli&oldid=739712834
(Conferido em: 28 de Outubro de 2016)