Geometrie computațională

Laborator 2

1. Ecran virtual si ecran grafic

La „ecranul grafic" originea este definită în colțul stânga sus, cu ordonata pozitivă orientată în jos. Pentru a transforma originea în mijlocul „ecranului virtual grafic" avem nevoie de următoarele formule

$$x \to x_e(x) = \frac{x - x_1}{x_2 - x1}a$$
$$y \to y_e(y) = -\frac{y - y_1}{y_2 - y1}b$$

unde a = *getmaxx*() iar b = *getmaxy*(), adică numărul maxim de pixeli pe coordonatele x și y.

```
// Template project.
#include "graphics.h"
#include <math.h>
#include <stdlib.h>
#include <dos.h>
#include <windows.h>
#include <iostream>

using namespace std;

#define pi 3.14159265359

float x_1, x_2, y_1, y_2;
int a, b;
int xe(float x)// normalizarea coordonatei x
{
	return((int)((x - x_1) / (x_2 - x_1)*a));
}
int ye(float y)// normalizarea coordonatei y
{
	return((int)((y_2 - y) / (y_2 - y_1)*b));
}
void axe()
{
	setcolor(0);
	outtextxy(xe(x_2) - 20, ye(0) - 20, "x");
	outtextxy(xe(x_2) - 18, ye(0) - 7, ">");
	outtextxy(xe(0) - 15, ye(y_2) + 15, "y");
	outtextxy(xe(0) - 15, ye(0) - 15, "O");
	outtextxy(xe(0) - 1, ye(y_2) , "^");
	line(xe(x_1), ye(0), xe(x_2), ye(0));
	line(xe(0), ye(y_1), xe(0), ye(y_2));
}
int main()
{
	printf("Limitele domeniului orizontal:\n");
	printf("Atentie, x_1<0<x_2  si y_1<0<y_2 \n");
```

```
        printf("x_1="); scanf("%f", &x_1); //x_1<0<x_2
        printf("x_2="); scanf("%f", &x_2);
        printf("Limitele domeniului vertical:\n");
        printf("y_1="); scanf("%f", &y_1); //y_1<0<y_2
        printf("y_2="); scanf("%f", &y_2);
        initwindow(800, 600, "AXE", 200, 200);
        setbkcolor(10);
        cleardevice();
        a = getmaxx(); //nr. maxim de pixeli pe coord. x
        b = getmaxy();
        axe();
        getch();
        closegraph();
        return 0;
}
```

2. Reprezentări 2D

În cele ce urmează se utilizează aşa-numitele reprezentări „parametrice", adică y nu este funcţie de x ci x şi y sunt funcţie de un parametru independent, de exemplu notat cu θ.
Vom defini o funcţie de timp $sin(\theta)$

```
void grafic()
{ float theta ;
float x,y;
float pie=3.1415;
float h=2*pie/4000; //,,pasul" de reprezentare
//float r=1.f;//raza cercului
     theta=0;
while (theta<=2*pie)
      {
      x=theta;
      y=sin(theta);
      putpixel(xe(x),ye(y),14);
      theta=theta+h;
      }
}
```

Temă: Construiţi $sin(t) * cos(t)$.

Cercul admite o reprezentare „parametrică" de forma:
$(x(\theta); y(\theta)) = (r\cos(\theta); r\sin(\theta))$, unde θ este în intervalul $[0; 2\pi)$

Pentru a reprezenta un cerc de rază 1, cu centrul în origine adăugăm

```
void grafic()
{ float theta ;
float x,y;
float pie=3.1415;
float h=2*pie/4000; //,,pasul" de reprezentare
float r=1.f;//raza cercului
```

```
theta=0;
while (theta<=2*pie)
      {
      x=r*cos(theta);
      y=r*sin(theta);
      putpixel(xe(x),ye(y),14);
      theta=theta+h;
      }
}
```

Lemniscata lui Bernoulli a fost menţionată de Bernoulli în 1694 şi studiată de Fagnano în 1750.

$$(x(\theta), y(\theta)) = (\frac{\alpha}{1+\sin^2(\theta)}\cos(\theta), \frac{\alpha\cos(\theta)}{1+\sin^2(\theta)}\sin(\theta)), \text{ unde } \theta \in [0, 2\pi),\ \alpha > 0$$

De exemplu, pentru $\alpha = 1$, funcţia *graphic()* se modifică astfel, si obtinem următorul program:

```
// Template project.
#include "graphics.h"
#include <math.h>
#include <stdlib.h>
#include <dos.h>
#include <windows.h>
#include <iostream>

using namespace std;

#define pi 3.14159265359

/*int gd, gm;
int n, i, j;
double r, x, y, xp, yp, fi;
*/

float x_1, x_2, y_1, y_2;
int a, b;
int xe(float x)// normalizarea coordonatei x
{
      return((int)((x - x_1) / (x_2 - x_1)*a));
}
int ye(float y)// normalizarea coordonatei y
{
      return((int)((y_2 - y) / (y_2 - y_1)*b));
}
void axe()
{
      setcolor(0);
      outtextxy(xe(x_2) - 20, ye(0) - 20, "x");
      outtextxy(xe(x_2) - 18, ye(0) - 7, ">");
      outtextxy(xe(0) - 15, ye(y_2) + 15, "y");
      outtextxy(xe(0) - 15, ye(0) - 15, "O");
      outtextxy(xe(0) - 1, ye(y_2) , "^");
```

```
	line(xe(x_1), ye(0), xe(x_2), ye(0));
	line(xe(0), ye(y_1), xe(0), ye(y_2));
}

/*
void grafic()
{ float theta ;
float x,y;
float pie=3.1415;
float h=2*pie/4000; //,,pasul" de reprezentare
float r=1.f;//raza cercului
theta=0;
while (theta<=2*pie)
{
x=r*cos(theta);
y=r*sin(theta);
putpixel(xe(x),ye(y),14);
theta=theta+h;
}
}
*/

float l_x(float theta)
{
	float sin_t = sin(theta);
	return 1 / (1 + sin_t * sin_t);
}
float l_y(float theta)
{
	float sin_t = sin(theta);
	return 1 * cos(theta) / (1 + sin_t * sin_t);
}
void grafic()
{
	float theta;
	float x, y;
	float pie = 3.1415;
	float h = 2 * pie / 4000;
	theta = 0;
	while (theta <= 2 * pie)
	{
		x = l_x(theta)*cos(theta);
		y = l_y(theta)*sin(theta);
		putpixel(xe(x), ye(y), 14);
		theta = theta + h;
	}
}

int main()
{
	printf("Limitele domeniului orizontal:\n");
	printf("Atentie, x_1<0<x_2  si y_1<0<y_2 \n");
	printf("x_1="); scanf("%f", &x_1); //x_1<0<x_2
	printf("x_2="); scanf("%f", &x_2);
	printf("Limitele domeniului vertical:\n");
	printf("y_1="); scanf("%f", &y_1); //y_1<0<y_2
	printf("y_2="); scanf("%f", &y_2);
```

```
        initwindow(800, 600, "AXE", 200, 200);
        setbkcolor(10);
        cleardevice();
        a = getmaxx(); //nr. maxim de pixeli pe coord. x
        b = getmaxy();
        axe();
        grafic();
        getch();
        closegraph();
        return 0;
}
```

Temă:

Studiaţi **Cisoida lui Diocles**(251-100 i.e.n), definită astfel

$$(x(\theta), y(\theta)) = (r(\theta)\cos(\theta), r(\theta)\sin(\theta)), \text{ unde } r(\theta) = 2a(\frac{1}{\cos(\theta)} - \cos(\theta)),\ \theta \in [0, 2\pi),\ a > 0$$

Reprezentaţi parametric funcţiile https://en.wikipedia.org/wiki/Parametric_equation

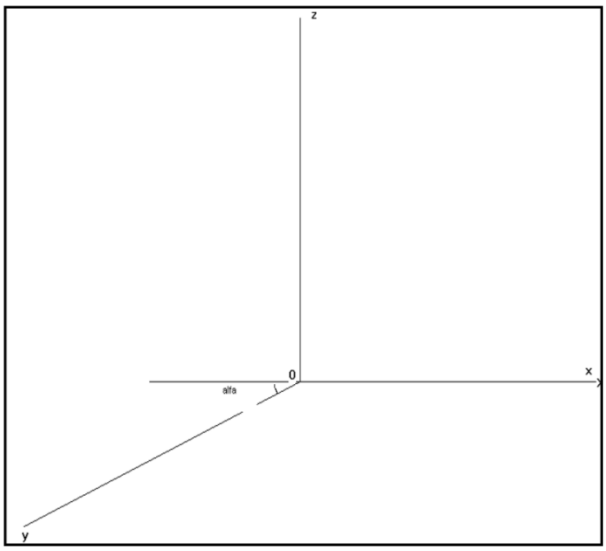
3. Reprezentări 3D

Principala problemă care se pune când vorbim despre obiecte tridimensionale în grafica pe calculator este de a reprezenta pe ecranul bi-dimensional puncte având trei coordonate spaţiale. În acest caz, se alege ca axă verticală axa *Oz* în loc de *Oy*, înlocuind funcţia de renormare a coordonatelor verticale int ye(float y) din capitolul precedent cu

```
int ze(float z)
// normalizarea coordonatei y
{return((int) floor((2*x2-y)/(2*x2-2*x1)*yemax));}
```

unde dimensiunea pe verticală a ferestrei de vedere este stabilită la aceeaşi mărime cu cea orizontală: (x1,x2).

A treia axă, *Oy*, se introduce ca o semidreaptă ce plecă din origine spre stânga-jos, cu un unghi arbitrar fixat *alfa*.



Transformarea de coordonate de la 3D la 2D se obţine prin proiecţie:

(x,z,y) → (x-cos(alfa)*y, z-sin(alfa)*y) →(**xe**(x-cos(alfa)*y), **ze**(z-sin(alfa)*y))

Reprezentarea unei elice simple *(cos(t), sin(t), kt)* ; *k*=0,3 în coordonate sferice

```
#include "graphics.h"
#include <math.h>
#include <stdlib.h>
#include <dos.h>
#include <windows.h>
#include <iostream>

using namespace std;

#define pi 3.14159265359

float x_1, x_2, y_1, y_2, alfa = 3.1415 / 8;
int xemax, yemax;
int xe(float x)
// normalizarea coordonatei x
{
	return((int)floor((x - x_1) / (x_2 - x_1)*xemax));
}
int ze(float z)
// normalizarea coordonatei y
{
	return((int)floor((y_2 - z) / (y_2 - y_1)*yemax));
}
void axe()
{
	setcolor(0);
	line(xe(x_1 / 2), ze(0), xe(x_2), ze(0));
	line(xe(0), ze(0), xe(0), ze(y_2));
	line(xe(0), ze(0), xe(0 + x_1 * cos(alfa)), ze(0 + x_1 * sin(alfa)));
	outtextxy(xe(0) - 15, ze(0) - 15, "O");
```

```
	outtextxy(xe(x_2) - 20, ze(0) - 20, "x");
	outtextxy(xe(x_2) - 6, ze(0) - 7, ">");
	outtextxy(xe(0) + 15, ze(y_2) + 5, "z");
	outtextxy(xe(0) - 1, ze(y_2) + 1, "^");
	outtextxy(xe(x_1*cos(alfa)), ze(x_1*sin(alfa)), "y");
}
float f(float theta) {
	return (2 * 0.7*sin(theta)*tan(theta));
}
void grafic() {
	float t, h;
	float x, y, z;
	float pie = 3.1415;
	t = 0; h = pie / 800;
	while (t <= +15)
	{
		x = 0.2*t*cos(3 * t); //ecuatiile elicei conice
		y = 0.2*t*sin(3 * t);
		z = 0.4*t;
		putpixel(xe(x - y * cos(alfa)), ze(z - y * sin(alfa)), 3);
		t = t + h;
	}
	t = 0;
}
int main()
{
	int gd, gm;
	printf("Limitele domeniului orizontal:\n");
	printf("x_1="); scanf("%f", &x_1);
	printf("x_2="); scanf("%f", &x_2);
	y_1 = x_1; y_2 = x_2;
	initwindow(800, 800, "Elice conica", 200, 200);
	setbkcolor(15);
	cleardevice();
	xemax = getmaxx(); yemax = getmaxy();
	axe();
	setcolor(2);
	grafic();
	getch();
	closegraph();
	return 0;
}
```

Bibliografie

[1] Anonymouse, *Dev-C++*, available at http://www.bloodshed.net/devcpp.html.
[2] Main, M., *WinBGIm*, available at http://www.cs.colorado.edu/~main/bgi/dev-c++/
[3] Anonymouse, *Borland Graphics Interface (BGI)*, http://www.cs.colorado.edu/~main/bgi/doc/