szönhetően csak az, hogy fogyasztásukkor a szájégetés és fogfájás sorrendje különböző). A modern konyhaművészeti technikák, ételreceptek Alain Ducasse művében, a Grand Livre de Cuisine (Kulináris Enciklopédia)-ban találhatók meg. Ez a könyv a hivatásos mesterszakácsok felkészülésekor kötelező olvasmány.

Forrásanyag:

[3] Kroó Gy.: Kürti Miklós köszöntése, Fizikai Szemle, 1998/5.
[4] H. This-Benckhard: Kürti Miklós, a molekuláris gasztronómia megalapítója, Fizikai Szemle, 1999/9.
[5] M. Kürti. H.This: Chemistry and Physics in the Kitchen, Scientific American, 1994., apr.
[6] Braun T. Empíriától a tudományig, Magyar Kémikusok Lapja, 2011. ápr.
[7] Boros N.: Főzzünk tudományosan! www.deol.hu

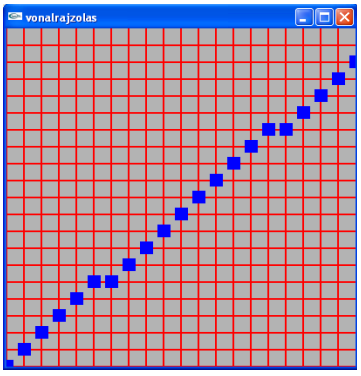
M. E.

Érdekes informatika feladatok

XXXVIII. rész

Vonalak és ellipszisek rajzolása

A számítógépes grafika alapalgoritmusai közé tartozik a vonalak és az ellipszisek kirajzolása. Azon túl, hogy ezek az algoritmusok nagyon gyorsak kell hogy legyenek (hisz nagyon sokszor hívódnak meg), az eredményük esztétikussága sem elhanyagolandó, hisz a vonalak és az ellipszisek szimmetrikusan szépek. Természetesen a fő probléma az, hogy ha a vonal nem függőleges, vagy vízszintes, hanem valamilyen szöget zár be a koordináta tengelyekkel, a pixelek feldarabolhatatlansága (csak egész pixelekkel tudunk dolgozni) oda vezet, hogy meg kell törni a vonalakat (például az 1. ábra szerint.)



1. ábra
Vonalak rajzolása

Hogy a fenti két követelményt kielégítsék, a történelem során számos módszert dolgoztak ki vonalak rajzolására. Ilyen volt a *Gyalog-módszer*, a *rekurzív rajzolás*, a *szimmetrikus DDA (Digital Differential Analyzer)*, az *egyszerű DDA*, a *midpoint algoritmus* stb.

Végül Bresenham közölt egy nagyon gyors, egész aritmetikát használó algoritmust vonalrajzolásra. Az algoritmus lényege, hogy a raszteres képen „oszlopirányban" haladva minden egész értékű *x*-koordinátában a matematikai egyeneshez függőlegesen legközelebbi pontot válasszuk.

Jack Elton Bresenham (1937–) 1965-ben publikálta a híres vonalrajzoló algoritmusát (BRESENHAM, J. E.: *Algorithm for Computer Control of a Digital Plotter*, IBM Systems Journal 4(1), p. 25-30., 1965.), és ez azóta is a számítógépes grafika alapalgoritmusa.

A Bresenham-algoritmus pszeudokódban:

```
1.  Eljárás Vonal(x1, y1, x2, y2, szín: egész);
2.    változók:
3.     du, dv, dd,
4.     p1, p2,
5.     u, v, uf, vf,
6.     S1, S2: hosszú egész;
7.
8.    du := x2 - x1;
9.    dv := y2 - y1;
10.   ha abs(dv) <= abs(du) akkor
11.       ha x1 <= x2 akkor
12.          u := x1;
13.          v := y1;
14.          uf := x2;
15.       különben
16.          u := x2;
17.          v := y2;
18.          uf := x1;
19.          du := -du;
20.          dv := -dv;
21.       (ha) vége
22.       ha dv >= 0 akkor
23.          dd := 2 * dv - du;
24.          p1 := 2 * (dv - du);
25.          p2 := 2 * dv;
26.          S1 := 1; S2 := 0;
27.       különben
28.          dd := 2 * dv + du;
29.          p1 := 2 * dv;
30.          p2 := (dv + du);
31.          S1 := 0;
32.          S2 := -1;
33.       (ha) vége
34.       TegyélPontot(u, v, szín);
35.       ameddig u < uf végezd el
36.          u := u + 1;
37.          ha dd >= 0 akkor
38.            v := v + S1;
39.            dd := dd + p1;
40.          különben
```

```
41.            v := v + S2;
42.            dd := dd + p2;
43.          (ha) vége
44.          TegyélPontot(u, v, szín);
45.       (ameddig) vége
46.    különben
47.       ha y1 <= y2 akkor
48.          v := y1;
49.          u := x1;
50.          vf := y2;
51.       különben
52.          v := y2;
53.          u := x2;
54.          vf := y1;
55.          du := -du;
56.          dv := -dv;
57.       (ha) vége
58.       ha du > 0 akkor
59.         dd := 2 * du - dv;
60.         p1 := 2 * (du - dv);
61.         p2 := 2 * du;
62.         S1 := 1;
63.         S2 := 0;
64.       különben
65.         dd := -2 * du - dv;
66.         p1 := -2 * (du + dv);
67.         p2 := -2 * du;
68.         S1 := -1;
69.         S2 := 0;
70.       (ha) vége
71.       TegyélPontot(u, v, szín);
72.       ameddig v < vf végezd el
73.         v := v + 1;
74.         ha dd >= 0 akkor
75.           u := u + S1;
76.           dd := dd + p1;
77.         különben
78.           u := u + S2;
79.           dd := dd + p2
80.         (ha) vége
81.         TegyélPontot(u, v, szín);
82.       (ameddig) vége
83.     (ha) vége
84. (Eljárás) vége;
```

Ellipszist rajzoló Da Silva algoritmus OpenGL-ben:

```
1.   #include "glut.h"
2.
3.   void init()
4.   {
5.     glClearColor(1.0, 1.0, 1.0, 1.0);
6.     glMatrixMode(GL_PROJECTION);
7.     glLoadIdentity();
8.     gluOrtho2D(-400, 400, -400, 400);
9.   }
```

```
10.
11.   void pont(double e, double f)
12.   {
13.     glPointSize(2.0);
14.     glBegin(GL_POINTS);
15.       glVertex2d(e, f);
16.       glVertex2d(-e, f);
17.       glVertex2d(-e, -f);
18.       glVertex2d(e, -f);
19.     glEnd();
20.   }
21.
22.   void ellipse(double a, double b)
23.   {
24.     double x, y;
25.     double d1, d2;
26.     x=0.0;
27.     y=b;
28.     d1=b*b-a*a*b+a*a/4;
29.     pont(x, y);
30.     while((a*a*(y-1/2)) > (b*b*(x+1)))
31.     {
32.       if(d1<0)
33.       {
34.         d1=d1+b*b*(2*x+3);
35.         ++x;
36.       }
37.       else
38.       {
39.         d1=d1+b*b*(2*x+3)+a*a*(-2*y+2);
40.         ++x;
41.         --y;
42.       }
43.     pont(x, y);
44.   }
45.   d2=b*b*(x*x+1/4+x)+a*a*(y*y-2*y+1)-a*a*b*b;
46.   while(y>0)
47.   {
48.     if(d2<0)
49.     {
50.       d2=d2+b*b*(2*x+2)+a*a*(-2*y+3);
51.       ++x;
52.       --y;
53.     }
54.     else
55.     {
56.       d2=d2+a*a*(-2*y+3);
57.       --y;
58.     }
59.     pont(x, y);
60.    }
61.   }
62.
63.   void display()
64.   {
65.     glClear(GL_COLOR_BUFFER_BIT);
```

```
66.     glColor3d(0.0, 0.0, 0.0);
67.     ellipse(150, 360);
68.     glFlush();
69.   }
70.
71.   void keyboard(unsigned char key, int x, int y)
72.   {
73.     switch (key)
74.     {
75.       case 27:
76.         exit(0);
77.         break;
78.     }
79.   }
80.
81.   int APIENTRY WinMain(HINSTANCE hInstance,
82.                        HINSTANCE hPrevInstance,
83.                        LPSTR     lpCmdLine,
84.                        int       nCmdShow)
85.   {
86.     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
87.     glutInitWindowSize(200, 200);
88.     glutInitWindowPosition(100, 100);
89.     glutCreateWindow("ellipszis");
90.     init();
91.     glutDisplayFunc(display);
92.     glutKeyboardFunc(keyboard);
93.     glutMainLoop();
94.     return 0;
95.   }
```

Házi feladat: Az ellipszis-rajzoló program mintájára ültessük át OpenGL-re és C-re a vonalrajzoló algoritmust is!

Kovács Lehel István

Katedra

Hogyan tanuljunk?

IV. rész

A Firka 2011-2012-es évfolyamában a Katedra rovatot a tanulásnak szenteljük, mivel Romániában a tanulóknak a 2011 júliusi érettségi vizsgáján elért nagyon gyenge eredményei (a vizsgára jelentkezetteknek több mint fele sikertelen volt) többek között arra vezethetők vissza, hogy a tanulók tanulással kapcsolatos ismeretei és szokásai – még tisztításra váró okok miatt – messze elmaradnak a kor követelményeitől. Reméljük, sorozatunkkal segíteni tudunk mind a tanároknak, mind a tanulni szándékozóknak.