zel 6 milliárd dollárra rúgott. Az áramszünet igen komplex események végeredménye volt, de egy szoftverhiba is szerepet kapott benne, amely miatt a vészjelző rendszer felmondta a szolgálatot. Ezt követően olyan folyamatok indultak el, amelyek miatt a fő és másodlagos irányítórendszer szervere is lelassult, majd leállt, az áramellátó rendszer pedig túlterhelődött.

Érdekes informatika feladatok

XXXIV. rész

A Lorentz-attraktor

1963-ban Edward Norton Lorenz (1917–2008) meteorológus egy egyszerű időjárási modell felállításával próbálkozott. Amikor a rendszer viselkedését fázistérben ábrázolta, egy igen furcsa attraktor képe bontakozott ki a szemei előtt: megszületett a *Lorentz-attraktor*.

Az alábbi nem lineáris dinamikus rendszert vizsgálta:

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = rx - y - xz \\ \dot{z} = -bz + xy \end{cases}$$

A dinamikus rendszerek egy állapottérrel leírt rendszerek, és a dinamikus rendszerek elmélete a rendszer valamely állapotainak rögzített szabályok szerinti időbeli változásával foglalkozik. Az inga lengésének, a csövekben áramló víznek, vagy egy tóban élő halak számának a matematikai leírása mind egy-egy példa dinamikus rendszerre.

Lorentz a következő jelenségre adta meg ezt az egyszerű modellt:

Melegítsünk egy vízszintes folyadékréteget alulról. Ha a hőmérséklet gradiense egy küszöbértéket meghalad, a folyadék mozgásba jön, és egy idő múlva stacionárius áramlás alakul ki. A felfelé és lefelé áramlás váltakozása révén a folyadékrétegben sajátos struktúrák jönnek létre. Ennek, az úgynevezett szabad konvekciónak a beindulása a hidrodinamikai instabilitások egyik legegyszerűbb példája. Az első kísérleti megfigyeléseket Bénard végezte 1900-ban, míg a konvekciómentes állapot stabilitásának feltételét Rayleigh vezette le elsőként, 1916-ban. Innen a jelenség neve: *Rayleigh–Bénard instabilitás*.

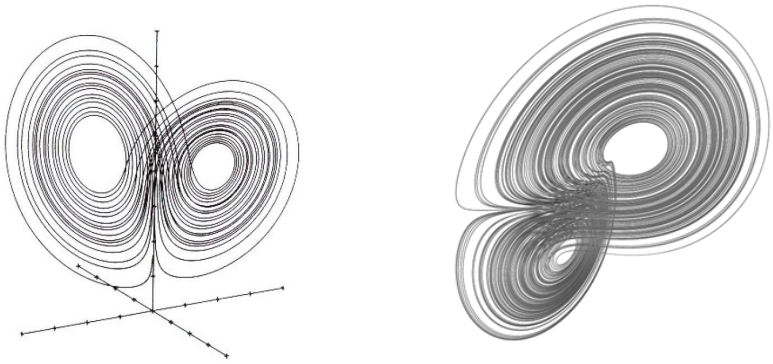
Amíg az edény alja és teteje között a hőmérsékletkülönbség kicsi, az energiaáramlás hődiffúzió révén valósul meg a folyadékban. Amikor azonban túllépünk egy kritikus hőmérsékletkülönbséget, makroszkópikus mozgás kezdődik. A hőmérséklet különbség további növelésekor a szabályos áramlási kép elromlik, a folyadék mozgása egyre bonyolultabbá, majd kaotikussá válik.

Lorentz a rendszer nagyszámú módusai közül hármat tartott meg. E három módus $X(t)$, $Y(t)$ és $Z(t)$ amplitúdója a hőmérséklet eloszlással és az áramlási teret jellemző áramlási függvénnyel kapcsolatos. Az $r > 0$ mennyiséget tekintjük kontrollparaméternek, a további paraméterek standard értékei: $\sigma = 10$, $b = 8/3$. A σ az úgynevezett Prandtl-szám, az r pedig a Rayleigh-szám.

Észrevette, hogy $r = 28$, $\sigma = 10$, $b = 8/3$ paraméterek mellett kis kezdeti feltételekbeli különbség esetén is igen eltérő időfejlődés tapasztalható. Amikor a rend-

szer viselkedését fázistérben ábrázolta, egy igen furcsa attraktor képe bontakozott ki a szemei előtt. Ez a róla Lorenz-attraktornak elnevezett különös ábra azóta a káosz egyik jelképévé vált.

A Lorentz-rendszer volt az első példa az előrejelezhetetlenség megjelenésére kis szabadságfokú autonóm rendszerben. A modell azóta a folytonos idejű kaotikus rendszerek alappéldája lett.



1 ábra
A Lorenz-attraktor

A következő OpenGL programmal kirajzoljuk a Lorentz-attraktort. Érdemes kipróbálni a Lorentz() függvényt más-más *r* értékekre is (pl. 14, 13, 15, 28 stb.).

```
#include "glut.h"
#include <math.h>

float xRot = 0.0f;
float yRot = 0.0f;

void Lorentz()
{
  int i, n=10000;
  double x0=0.1, y0=0, z0=0, x1, y1, z1;
  double h=0.005;
  double r=28.0, sigma=10.0, b=8.0/3.0;
  glColor3f(1.0,0.0,0.0);
  glBegin(GL_POINTS);
  for (i=0;i<=n;i++)
  {
       x1 = x0 + h * sigma * (y0 - x0);
       y1 = y0 + h * (x0 * (r - z0) - y0);
       z1 = z0 + h * (x0 * y0 - b * z0);
       x0 = x1;
       y0 = y1;
       z0 = z1;
       glVertex3f((x0-0.95)/5,(y0-1.78)/5,(z0-26.7)/5);
  }
  glEnd();
```

```
}

void RenderScene()
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glPushMatrix();
  glEnable(GL_LIGHTING);
  glRotatef(yRot,0,1,0);
  glRotatef(xRot,1,0,0);
  Lorentz();
  glPopMatrix();
  glutSwapBuffers();
}

void SpecialKeys(int key, int x, int y)
{
  if(key == GLUT_KEY_UP)
       xRot-= 5.0f;
  if(key == GLUT_KEY_DOWN)
       xRot += 5.0f;
  if(key == GLUT_KEY_LEFT)
       yRot -= 5.0f;
  if(key == GLUT_KEY_RIGHT)
       yRot += 5.0f;
  if(xRot > 356.0f) xRot = 0.0f;
  if(xRot < -1.0f) xRot = 355.0f;
  if(yRot > 356.0f) yRot = 0.0f;
  if(yRot < -1.0f) yRot = 355.0f;
  glutPostRedisplay();
}

void spinDisplay1()
{
  xRot += 5.0f;
  if(xRot > 356.0f) xRot = 0.0f;
  if(xRot < -1.0f) xRot = 355.0f;
  glutPostRedisplay();
}

void spinDisplay2()
{
  yRot += 5.0f;
  if(yRot > 356.0f) yRot = 0.0f;
  if(yRot < -1.0f) yRot = 355.0f;
  glutPostRedisplay();
}

void mouse(int button, int state, int x, int y)
{
  switch (button) {
  case GLUT_LEFT_BUTTON:
       if (state == GLUT_DOWN)
            spinDisplay1();
       if (state == GLUT_UP)
            glutIdleFunc(NULL);
       break;
  case GLUT_RIGHT_BUTTON:
       if (state == GLUT_DOWN)
```

```
            glutIdleFunc(spinDisplay2);
        if (state == GLUT_UP)
            glutIdleFunc(NULL);
        break;
    default:
        break;
    }
}

void ChangeSize(GLsizei w, GLsizei h)
{
    GLfloat lightPos[] = { -50.f, 50.0f, 100.0f, 1.0f };
    if(h == 0) h = 1;
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h) glOrtho (-9, 9, -9*h/w, 9*h/w, -10.0, 10.0);
    else glOrtho (-9*w/h, 9*w/h, -9, 9, -10.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void SetupRC()
{
    GLfloat  ambientLight[] = { 0.3f, 0.3f, 0.3f, 1.0f };
    GLfloat  diffuseLight[] = { 0.7f, 0.7f, 0.7f, 1.0f };
    glEnable(GL_LIGHTING);
    glLightfv(GL_LIGHT0,GL_AMBIENT,ambientLight);
    glLightfv(GL_LIGHT0,GL_DIFFUSE,diffuseLight);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
    glShadeModel(GL_SMOOTH);
    glDisable(GL_CULL_FACE);
    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f );
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutInitWindowSize(800,800);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutCreateWindow("Lorentz");
    glutReshapeFunc(ChangeSize);
    glutSpecialFunc(SpecialKeys);
    glutDisplayFunc(RenderScene);
    glutMouseFunc(mouse);
    SetupRC();
    glutMainLoop();
    return 0;
}
```

Kovács Lehel István