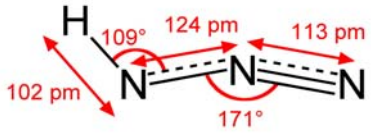
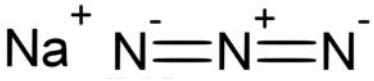


hidrogén-azid szerkezete



nátrium-azid

Ahogy öregednek a légzsákkal felszerelt autók, úgy válik egyre sürgetőbb problémává a méreggel töltött robbanópatronok megfelelő megsemmisítése. A környezetvédők a roncsautók megsemmisítését szabályozó rendeletek megváltoztatásáért kampányolnak. Szeretnék elérni, hogy a szeméttelepre való kerülésük előtt a roncsautókban a légzsákokat aktiválják, azért, hogy a nátrium-azid bomoljon le a kevésbé veszélyes összetevőkre. Az autógyárak tervezői is felismerték a nátrium-azidos légzsákok alkalmazásának hátrányait, és új megoldásokkal próbálkoznak. Azidmentes töltetként trinitrát-cellulózt (lőgyapot, vagy piroxilin néven is ismert) próbáltak használni, de nem vált be. Hibrid-töltelékes megoldásoknál a légzsákban lőgyapotba ágyazott kevés nátrium-azidot használnak. Ezzel mind a két komponensnek a káros hatását, környezetszennyező mértékét részben csökkenteni lehet. A mostanában gyártott autókban már sűrített nitrogén fújja fel a légzsákokat mielőtt a kismennyiségű pirotechnikai töltet megemeli a hőmérsékletét. A tökéletes természetbarát légzsák megvalósítására nagynyomású sűrített levegő és hidrogén elegyet használnak, amely gyújtószikra hatására berobban, és vízgőz, valamint nitrogén gáz hő okozta kiterjedésével fejti ki védő szerepét, miközben semmilyen környezetre káros termék nem keletkezik. E rendszer elterjedésének még anyagtechnikai és gazdasági akadályai vannak.

M. E.

Érdekes informatika feladatok

XXXVII. rész

Részecskerendszerek

A generatív számítógépes grafika számtalan effektus megvalósítására (pl. tűz, robbanás, tűzijáték, szökőkút stb.) használja a *részecskerendszereket*. Általánosságban, részecskerendszernek tekintjük a mozgási szabályokkal felruházott pontok halmazát.

1962-ben, Steve Russel, Martin Graetz és Wayne Wiitanen a massachusettsi egyetem PDP-1-es számítógépére írták meg az egyik legelső számítógépes játékot, a SpaceWar!-t.

Egy közös pontból véletlenszerű irányokba kiinduló pöttyöket alkalmaztak a robbanás látványának megjelenítésére. Így született meg az első részecskerendszer.

1983. volt a részecskerendszerek történetének fontos pillanata, amikor a Star Trek II. – *Khan haragja* című filmnek a látványelemeként elkészített részecskerendszerből megszületett William T. Reeves mai napig alapvető cikke.

A filmhez szükséges volt ugyanis egy bolygó felszínén végigfutó tűz-animációra, amelyet részecskerendszerrel valósítottak meg.

Ha osztályozni szeretnénk a részecskerendszereket, két nagy csoportot tudunk elkülöníteni:

- állapotmentes részecskerendszer
- állapotot megőrző részecskerendszer

Állapotmentes részecskerendszerek azok a rendszerek, amelyeknél a részecske pozíciója egyértelműen meghatározható a kiindulási adatokból, tudva azt, hogy mennyi ideje is él a rendszerben. Ezek a rendszerek egyszerű hatások leírására használatosak, könnyen lehet velük látványos, a környezet által nem befolyásolt effektusokat készíteni.

Az akciójátékokban ezek lehetnek például a fegyverek tüzelésekor létrejövő szikrák, a lövedék becsapódásakor keletkező foltok.

Az állapotot megőrző részecskerendszereket nagyobb látványelemek elkészítéséhez szokták használni, amikor szükség van arra, hogy a rendszer reagáljon a környezetre.

Minden egyes lépésben frissülnek a részecskék tulajdonságai annak függvényében, hogy hol voltak előzőleg, s milyen kölcsönhatásokba kerültek más részecskékkel vagy a környezettel.

Ezt követően kell eldönteni, hogy mely részecskéket kell eltávolítani a rendszerből, s mennyi új részecskét kell a rendszerbe behelyezni.

Az előbbiek alapján megfigyelhetjük, hogy a részecskék bizonyos tulajdonságokkal rendelkeznek, mint például:

- pozíció
- sebesség
- méret
- szín
- átlátszóság
- alak
- élettartam stb.

Egyes részecskéken textúra is lehet.

Ezen tulajdonságok összességét egy adott időpontban nevezzük a *részecske pillanatnyi állapotá*nak.

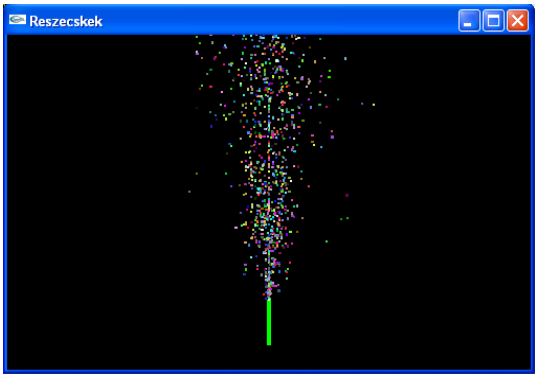
A részecskéket könnyen szimulálhatjuk struktúrák, objektumok, de egyszerű tömbök segítségével is. Például az alábbi C++ struktúra egy részecskét valósít meg:

```
struct reszecske
{
  float x;
  float y;
  float z;
  int sebesseg;
  float irany_x;
  float irany_y;
  float irany_z;
  float c_tex_x;
  float c_tex_y;
  float elet;
  int allapot;
  int ido;
  int ossz_ido;
};
```

A rendszerben mindig egy adott számú részecske él, a részecskék létrehozásáért a *részecskeforrás* felel.

Ez az az objektum, amit elhelyezünk háromdimenziós világunkban, hogy létrejöjjön a kívánt hatás. A forrás felelős azért, hogy mennyi részecskét hozunk létre egyszerre, s hogy a létrehozott részecskék milyen kezdeti értékeket vesznek fel.

A következő egyszerű programmal egy tűzijáték petárdát vagy konfettit szimulálunk részecskerendszer segítségével. A részecskéket egy tömbben tároljuk (*X* és *Y* koordináták). A mozgásukat egy külön függvény számítja ki. A részecskék pályájának és színének meghatározásában a véletlen is szerepet játszik.



A program (OpenGL és Visual C++) így néz ki:

```
#include <time.h>
#include <cmath>
#include <windows.h>
#include "glut.h"

float hatar = 1.2;
float X=0, Y=0;
const int MAX_RESZECSKEK = 1000;
const int MIN_RESZECSKEK = 10;
int AktReszecske = 1;
float pozX[MAX_RESZECSKEK], pozY[MAX_RESZECSKEK];

void ReszecskeMozgatas(int reszecskek)
{
  srand(time (NULL));
  float ujX;
  Sleep(1);
  glColor3d(2, .5, 0);
  for(int i=0; i<reszecskek; ++i)
  {
    ujX = rand() % 3 + 1;
    if(ujX==1 && pozX[i]<=hatar)
    {
      int temp1 = rand() % 100 + 1;
      int temp2 = rand() % 5 + 1;
```

```
      pozX[i]+=temp2*.001;
      pozY[i]+=temp1*.0004;
    }
    if(ujX==2)
    {
      pozX[i]+=.00;
      pozY[i]+=.01;
    }
    if(ujX==3 && pozX[i]>=-hatar)
    {
      int temp2 = rand() % 5 + 1;
      int temp1 = rand() % 100 + 1;
      pozX[i]-=temp2*.001;
      pozY[i]+=temp1*.0004;
    }
    if(pozY[i]>=hatar)
    {
      pozY[i]=0;
      pozX[i]=0;
    }
  }
}

void Reshape(int height, int width)
{
  glViewport(0, 0, width, height);
  glClearColor(0, 0, 0, 1);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(60, (float)height/(float)width, 1, 100);
  glMatrixMode(GL_MODELVIEW);
}

void Draw(void)
{
  bool volt = false;
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glColor3f(0, 1, 0);
  glBegin(GL_QUADS);
    glVertex3f(-.01, -0.2, -2);
    glVertex3f(.01, -0.2, -2);
    glVertex3f(.01, 0, -2);
    glVertex3f(-.01, 0, -2);
  glEnd();
  if(!volt)
  {
    float R, G, B;
    glPushMatrix();
    glBegin(GL_QUADS);
    for(int i=0; i<MAX_RESZECSKEK; ++i)
    {
      R = rand() % 100 + 1;
      G = rand() % 100 + 1;
      B = rand() % 100 + 1;
      glColor3d(R*.01, G*.01, B*.01);
      glVertex3f(X-.005, Y-.005, -2);
      glVertex3f(X+.005, Y-.005, -2);
      glVertex3f(X+.005, Y+.005, -2);
```

```
      glVertex3f(X-.005, Y+.005, -2);
      X = pozX[i];
      Y = pozY[i];
    }
    glEnd();
    glPopMatrix();
    volt = true;
  }
  Sleep(.5);
  ReszecskeMozgatas(AktReszecske);
  if(AktReszecske!=MAX_RESZECSKEK) ++AktReszecske;
  glutPostRedisplay();
  glutSwapBuffers();
}

void Keyboard(unsigned char key, int x, int y)
{
  switch(key)
  {
    case 27:
      exit(0);
      break;
  }
}

void init()
{
  const GLfloat light_ambient[]  = {0.0f, 0.0f, 0.0f, 1.0f};
  const GLfloat light_diffuse[]  = {1.0f, 1.0f, 1.0f, 1.0f};
  const GLfloat light_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
  const GLfloat light_position[] = {2.0f, 5.0f, 5.0f, 0.0f};
  const GLfloat mat_ambient[]    = {0.7f, 0.7f, 0.7f, 1.0f};
  const GLfloat mat_diffuse[]    = {0.8f, 0.8f, 0.8f, 1.0f};
  const GLfloat mat_specular[]   = {1.0f, 1.0f, 1.0f, 1.0f};
  const GLfloat high_shininess[] = {100.0f};
  glEnable(GL_CULL_FACE);
  glCullFace(GL_BACK);
  glEnable(GL_DEPTH_TEST);
  glDepthFunc(GL_LESS);
  glEnable(GL_LIGHT0);
  glEnable(GL_NORMALIZE);
  glEnable(GL_COLOR_MATERIAL);
  glEnable(GL_LIGHTING);
  glLightfv(GL_LIGHT0, GL_AMBIENT,  light_ambient);
  glLightfv(GL_LIGHT0, GL_DIFFUSE,  light_diffuse);
  glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
  glLightfv(GL_LIGHT0, GL_POSITION, light_position);
  glMaterialfv(GL_FRONT, GL_AMBIENT,   mat_ambient);
  glMaterialfv(GL_FRONT, GL_DIFFUSE,   mat_diffuse);
  glMaterialfv(GL_FRONT, GL_SPECULAR,  mat_specular);
  glMaterialfv(GL_FRONT, GL_SHININESS, high_shininess);
}

int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowSize(500, 500);
```

```
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Reszecskek");
    init();
    glutReshapeFunc(Reshape);
    glutDisplayFunc(Draw);
    glutKeyboardFunc(Keyboard);
    glutMainLoop();
    return 0;
}
```

Kovács Lehel István

Katedra

Hogyan tanuljunk?

III. 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 tisztásásra váró okok miatt – messze elmaradnak a kor követelményeitől. Reméljük, sorozatunkkal segíteni tudunk mind a tanároknak, mind a tanulni szándékozóknak.

A szövegek olvasása

Minden tanulási folyamat az ún. értelmes „olvasással" veszi kezdetét. Az olvasás fogalmát itt a legtágabb értelemben értjük, vagyis nem csak egy bizonyos szövegnek az elolvasására gondolunk, hanem az észlelt valóság egy bizonyos részének tudatos megfigyelésére, megértésére, és végül a helyes értelmezésére is. Az iskolai tanulás kiindulópontját a legtöbb esetben magának egy szövegnek az elolvasása képezi. Ennek hiányában csupán magolásról, értelmetlen időpocsékolásról beszélhetünk. Az alábbiakban Derek Rowntree alapján az értő szövegolvasás szakaszait szeretnénk bemutatni (Rowntree 1980).

I. A szöveg rövid áttekintése. Amikor kezünkbe vesszük a tankönyvet, kikeressük az aktuális anyagrészt (a leckét), és először annak a rövid áttekintésével kezdjük. Ennek az a célja, hogy átfogó képet alkossunk az anyagról. Azt kell figyelemmel követnünk, hogy milyen előzetes ismeretelemekhez kapcsolódik ez az anyagrész, miért van szükség erre az anyagra, milyen célból kell azt megtanulni, mi ennek a résznek a központi gondolata, hogyan épül fel a tartalma? Ezt követően elolvassuk a lecke címét, az alcímeket, minden résznek a következtetését, figyelmesen megnézzük a rajzokat, grafikonokat, táblázatokat. Olvasgassunk bele véletlenszerűen a szövegbe, ízlelgessük a szöveget. Olvassuk el az első, majd az utolsó bekezdést, amelyek rendszerint a fő gondolatot, illetve következtetést szokták tartalmazni. Próbáljunk rájönni arra, hogy a vizsgált kérdések hogyan kapcsolódnak egymáshoz.

II. Kérdések megfogalmazása. Ha sikerült magunknak az anyaggal kapcsolatos kérdéseket megfogalmazni, a vizsgálódásunkat a cél irányába terelgethetjük. A legelső kérdésünk az anyag iránti érdeklődésünket jellemzi. Az anyag áttekintése, a címek elolvasása