Számítógépes grafika

XXIX. rész

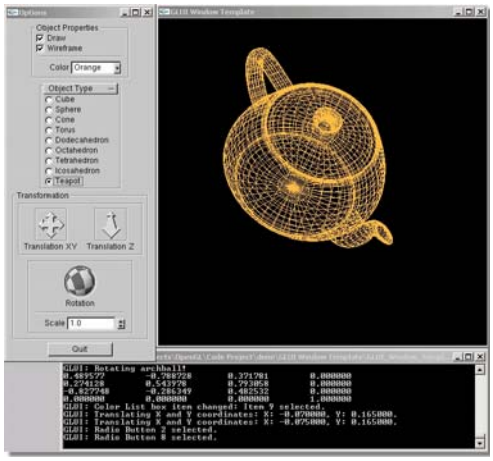
Más OpenGL lehetőségek

A GLUI

A GLUI egy Paul Rademacher által fejlesztett GLUT alapú C++-ban felhasználói felületet megvalósító függvénykönyvtár, amely letölthető a *http://www.cs.unc.edu/~rademach/glui/* honlapról.

A GLUI a GLUT teljes integrálása mellett grafikus felhasználói felületet (ablakokat és kontrollokat) biztosít az OpenGL alkalmazások számára. Segítségével könnyen és egyszerűen fejleszthetünk olyan felületeket az OpenGL alkalmazásunk számára, amelyek gombokat (button), jelölődobozokat (checkbox), rádió-gomb (radio button), szövegdobozokat (text box), listákat (listbox), címkéket (static text), panelleket (panel) és csoportosító dobozokat (groupbox) és más grafikus kontrollokat tartalmaznak, amelyek *callback* függvények segítségével megvalósítják az eseményvezérlést. A kontrollok változókkal szinkronizálhatók, így az értékük valós időben változik, ha az OpenGL kód megváltoztatja őket. Hasonlóan a kontrollok kiválthatják az OpenGL kép frissítését.

A hagyományos (Windows alatt is jól ismert) kontrollokon kívül a GLUI rendszer egy pár saját kontrollt is bevezet, például a grafikus objektumok eltolására, skálázására, forgatására vonatkozó egyszerű, de annál szemléletesebb kontrollokat.



Egyszerű GLUI példa

OpenGL Delphiben

Mivel a *Delphi* már eleve felkínálja a *form*ot, azt az űrlapot, amelyet megtervezve megkapjuk az alkalmazás ablakát, az OpenGL használata *Delphi*-ből nagyon hasonlít a előbbi MFC példához, csak sokkal egyszerűbb. Több unitot is használhatunk, több OpenGL implementáció létezik *Delphi* alá.

Például, ha az *OpenGL12* unitot használjuk (*http://delphi-jedi.org*), a rajzoló felületünk (a form unitja) így fog kinézni:

```
1.  unit PontokMain;
2.
3.  interface
4.
5.  uses Windows, Messages, SysUtils, Variants,
6.    Classes, Graphics, Controls, Forms, Dialogs,
7.    ExtCtrls;
8.
9.  type
10.   TForm1 = class(TForm)
11.     procedure FormPaint(Sender: TObject);
12.     procedure FormDestroy(Sender: TObject);
13.     procedure FormCreate(Sender: TObject);
14.   end;
15.
16. var
17.   Form1: TForm1;
18.   dc: HDC; // az OpenGL inicializálásához
19.   gc: HGLRC; // az OpenGL inicializálásához
20.
21. implementation
22.
23. uses OpenGL12;
24.
25. {$R *.dfm}
26.
27. procedure TForm1.FormCreate(Sender: TObject);
28. var h: HPALETTE;
29. begin
30.   // a felület inicializálása
31.   dc := GetDC(Handle);
32.   gc := CreateRenderingContext(dc, [], 32, 0, 0,
33.                                0, 0, h);
34.   ActivateRenderingContext(dc, gc);
35.   // az OpenGL programunk inicializálása
36.   glViewport(0, 0, ClientWidth, ClientHeight);
37.   glMatrixMode(GL_PROJECTION);
38.   glLoadIdentity();
39.   gluPerspective(60, ClientWidth / ClientHeight,
40.                  0.1, 30.0);
41.   glClearColor(0, 0, 0, 0);
42.   glMatrixMode(GL_MODELVIEW);
43.   glLoadIdentity;
44. end;
45.
46. procedure TForm1.FormDestroy(Sender: TObject);
47. begin
48.   DeactivateRenderingContext;
49.   DestroyRenderingContext(gc);
50. end;
51.
52. procedure TForm1.FormPaint(Sender: TObject);
53. begin
54.   glClear(GL_COLOR_BUFFER_BIT);
```

```
55.   glPointSize(10);
56.   glPushMatrix();
57.   glTranslatef(-3.0, 2.5, -5.0 );
58.   glBegin(GL_POINTS);
59.     glColor3f(1.0, 1.0, 0.0);
60.     glVertex3f(0.0, 0.0, 0.0);
61.     glColor3f(1.0, 0.0, 0.0);
62.     glVertex3f(2.0, -2.0, 0.0);
63.   glEnd();
64.   glPopMatrix();
65.   glFlush;
66. end;
67.
68. end.
```

Ha a *Delphi*-vel forgalmazott *OpenGL* unitot használjuk (`uses OpenGL;`), akkor az MFC-hez hasonlóan ki kell töltenünk egy pixelformátum-leírót. Ekkor az inicializáló függvényünk így néz ki:

```
1.  function InitOpenGL(aDC: HDC; ColorBits: integer;
2.    DoubleBuffer: boolean ): boolean;
3.  // aDC a Device Context, ahova rajzolni fogunk
4.  // ColorBits a rajz színskálája: 16/24/32 bit
5.  // DoubleBuffer a dupla buffer használata
6.  var
7.    PixelFormat: TPixelFormatDescriptor;
8.    cPixelFormat: integer;
9.    gc: HGLRC;
10. begin
11.   FillChar(PixelFormat, SizeOf(PixelFormat), 0);
12.   with PixelFormat do
13.     begin
14.      nSize := Sizeof(TPixelFormatDescriptor);
15.      if DoubleBuffer then
16.        dwFlags := PFD_DOUBLEBUFFER or
17.                   PFD_DRAW_TO_WINDOW or
18.                   PFD_SUPPORT_OPENGL
19.      else
20.        dwFlags := PFD_DRAW_TO_WINDOW or
21.                   PFD_SUPPORT_OPENGL;
22.      iLayerType := PFD_MAIN_PLANE;
23.      iPixelType := PFD_TYPE_RGBA;
24.      nVersion := 1;
25.      cColorBits := ColorBits;
26.      CdepthBits := 16;
27.     end;
28.   cPixelFormat := ChoosePixelFormat(aDC,
29.                   @PixelFormat);
30.   Result := cPixelFormat <> 0;
31.   if not Result then
32.     begin
33.       MessageBox(0,
34.            pChar(SysErrorMessage(GetLastError)),
```

```
35.           'Init OpenGL', mb_OK);
36.       exit;
37.     end;
38.   Result := SetPixelFormat(aDC, cPixelFormat,
39.             @PixelFormat);
40.   if not Result then
41.     begin
42.       MessageBox(0,
43.            pChar(SysErrorMessage(GetLastError)),
44.            'Init OpenGL', mb_OK);
45.       exit;
46.     end;
47.   gc := wglCreateContext(aDC);
48.   Result := gc <> 0;
49.   if not Result then
50.     begin
51.       MessageBox(0,
52.            pChar(SysErrorMessage(GetLastError)),
53.            'Init OpenGL', mb_OK);
54.       exit;
55.     end;
56.   Result := wglMakeCurrent(aDC, gc);
57.   if not Result then
58.     begin
59.       MessageBox(0,
60.            pChar(SysErrorMessage(GetLastError)),
61.            'Init OpenGL', mb_OK);
62.       exit;
63.     end;
64. end;
```

A fenti inicializáló függvény meghívása után az elkészített rajzaink meg fognak jelenni a DC által mutatott objektum területén. Windows alatt ügyelnünk kell arra, hogy a használt színmélység egyezzen meg a Windows által beállítottal (Start menü\Control Panel\Display – Settings – Color Quality), ellenkező esetben alkalmazásunk jóval lassúbb lesz, mert az operációs rendszer színkonverziót hajt végre.

Grafika feladatok

1. Bűvös kocka. Jelenítsünk meg egy 3×3×3-as Rubik-kockát. Szimuláljuk a kocka működését! Legyen lehetőség a kocka oldallapjainak az elforgatására, a kocka összekeverésére és kirakására!

Útmutatás a megoldáshoz

A Rubik-kocka (másként bűvös kocka) mechanikus, egyéni logikai játék. A kocka oldalai különféle színűek és elforgathatók a lap középpontja körül. A forgatás során a szomszédos oldalak színe megváltozik. A rendszertelen forgatással az oldalak színösszeállítása összekeverhető. Összesen 43 252 003 274 489 856 000-féle (kb. $4{,}3 \cdot 10^{19}$) eltérő állás hozható létre. A játék célja, hogy egy előzetesen összekevert kockából forga-

tással visszaállítsuk az eredeti, rendezett színösszeállítást, vagyis minden oldalon azonos színű lapocskák legyenek.

A kockát Rubik Ernő (1944–) magyar feltaláló, tervező alkotta meg 1975-ben és hamarosan világszinten népszerű játék lett.

A feladat megoldásához célszerű objektumorientáltan eljárni. Egy kis kocka legyen egy objektum, melyen minden oldallapot különböző színűre állíthatunk be. Az objektumokat tároljuk el egy 3×3×3-as mátrixban.

Oldjuk meg a kis kockák csoportosítását oldallapok szerint (egy kis kocka több oldallaphoz is tartozhat), oldjuk meg az oldallapok forgatását, a színek kicserélését!

Általánosítás: Próbáljuk meg 2×2×2, 4×4×4, ..., $n \times n \times n$-es Rubik-kockákat is megvalósítani!

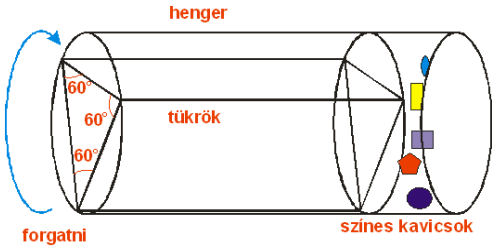
2. Kaleidoszkóp. Írjunk kaleidoszkópot szimuláló grafikus alkalmazást!

Útmutatás a megoldáshoz

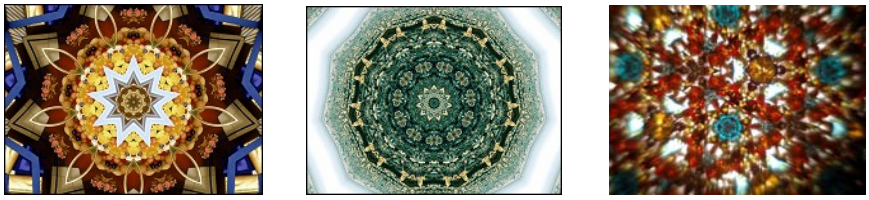
A görögök már ismerték, Sir David Brewster (1781–1868) 1816-ban találta fel ismét a *kaleidoszkóp*ot.

Egyszerűen úgy készíthetünk kaleidoszkópot, hogy három azonos méretű téglalap alakú tükörből egyenlő oldalú háromszög alapú hasábot képezünk, majd ezt egy hengerbe (csőbe) csúsztatjuk. A cső egyik végére feszítsünk átlátszó műanyag fóliát vagy zsírpapírt, majd a csövet kívülről borítsuk be fekete papírral! Szórjunk a cső lezárt végébe apró színes tárgyakat, ezüstpapírt, műanyagdarabkákat, gyöngyöt, stb.!

A cső szabad végét szemünk elé emelve, szabályos háromszögekből álló szimmetrikus mintázatban gyönyörködhetünk. A minta a cső mozgatásával (forgatásával) változik (átrendeződnek a gravitáció miatt a színes tárgyak).



A kaleidoszkóp vázlatos szerkezete



Kaleidoszkóp mintázatai

Kovács Lehel