Számítógépes grafika

XXVIII. rész

OpenGL Visual C++-ban

Ha OpenGL programot szeretnénk létrehozni VisualC++-ban, három lehetőségünk van: *Win32 alkalmazás, Win32 konzol alkalmazás és MFC platformon történő programozás.*

Ha az első kettőt választjuk, akkor a GLUT (OpenGL Utility Toolkit) feladata az ablakozó rendszer kezelése és a grafika megjelenítése. A harmadik esetben az ablakozó rendszert a Visual C++ MFC osztályhierarchiája oldja meg és a grafika egy Windows-os kontrollban jelenik meg.

Win32 alkalmazás

Ha Win32 alkalmazásként szeretnénk létrehozni az OpenGL programot, akkor:

- Elindítjuk a *Visual C++* 6.0-ást
- *File / New... / Projects / Win 32 Application* utat járjuk be a menüből kiindulva
- Beírjuk a project nevét: *Project name: Elso*
- Beállítjuk a mentési útvonalat
- OK gomb, majd:
- *A simple Win32 application.*
- Így a következő főprogram-modul jött létre:

```
1.  // Elso.cpp : Defines the entry point for the application.
2.  //
3.
4.  #include "stdafx.h"
5.
6.  int APIENTRY WinMain(HINSTANCE hInstance,
7.                        HINSTANCE hPrevInstance,
8.                        LPSTR     lpCmdLine,
9.                        int       nCmdShow)
10. {
11.   // TODO: Place code here.
12.
13.   return 0;
14. }
```

- Ha ezzel megvagyunk (a varázsló befejeződött), előjön a *Visual C++* programozói felülete, és elkészült a projektnek megfelelő könyvtárstruktúra is.
- Ha nincs OpenGL bekonfigurálva *Visual C++* alá, akkor ezt a következőképpen tehetjük meg:
  - Például a *http://www.xmission.com/~nate/glut.html* honlapról töltsük le a *glut-3.7.6-bin.zip* állományt (vagy, ha közben frissítették, akkor az újabb verziót)
  - Kicsomagolás után öt állományt kapunk, amelyből három fontos számunkra: *glut.h*, *glut32.lib*, valamint *glut32.dll*.
  - Ha nincs írásjogunk a rendszerkönyvtárakhoz, akkor másoljuk be a *glut.h*-t és a *glut32.lib*-et a projekt könyvtárába, a *glut32.dll*-t pedig a projekt *Debug* könyvtárába.
  - Ha van írásjogunk a rendszerkönyvtárakhoz, akkor véglegesen is feltelepíthetjük az OpenGL-t (így minden projekt tudja használni a fent említett állományokat): másoljuk a *glut32.dll*-t a *Windows / System32* könyvtárba, a *glut32.lib*-et a *Visual Studio* Library könyvtárába (pl. *c:\Program Files\Microsoft Visual Studio\VC98\Lib*),

glut.h állománynak pedig hozzunk létre egy saját *GL* nevű könyvtárat a *Visual Studio* Include könyvtárában (pl. *c:\Program Files\Microsoft Visual Studio\VC98\Include\GL*).

- A *Visual C++* menüjéből kiindulva, a *Project / Settings* beállításoknál, a *Link* fülnél írjuk hozzá a már meglévő *Object/libary modules* sorhoz a következőket: glut32.lib glu32.lib opengl32.lib glaux.lib.
- A fenti főprogram-modul *include* sorába írjuk be az OpenGL headerállományát is: `#include <GL\glut.h>`, vagy `#include "glut.h"`, ha a *glut.h* a projekt könyvtárában van.

Win32 konzol alkalmazás

Ha konzol alkalmazásként hozzuk létre az OpenGL programot, akkor a grafikus ablakunk mellett egy szöveges üzemmódú ablakunk is lesz, ahová adatokat tudunk kiírni egy egyszerű `printf` utasítással. A parancssor paramétereit is át tudjuk adni az OpenGL-nek.

Az alkalmazás létrehozása annyiban különbözik az előzőtől, hogy ebben az esetben a *Visual C++* elindítása után a *File / New... / Projects / Win 32 Console Application* utat járjuk be a menüből kiindulva.

Az alkalmazás főfüggvénye egyszerűen a következő lesz:

```
1. #include "stdafx.h"
2.
3. int main(int argc, char* argv[])
4. {
5.   return 0;
6. }
```

Az alkalmazás csak annyiban különbözik az előzőtől, hogy a `main` függvény első sorában meghívjuk a `glutInit(&argc, argv);` parancsot, amely inicializalja a glut libet a parancssor paramétereinek megfelelően.

MFC alkalmazás

Ha ezt a lehetőséget választjuk, a programunk nem lesz többé átvihető más platformra, csak Windows alatt fog futni, viszont kényelmesen, Windows kontrollok segítségével vezényelhetjük a rajzolást.

Az alkalmazás létrehozásakor a *Visual C++* elindítása után a *File / New... / Projects / MFC AppWizard (exe)* utat járjuk be a menüből kiindulva, és a varázsló segítségével létrehozzuk a dialógus alapú *(dialog based)* MFC alkalmazásoknál megszokott pl. `COglMFCDialogApp` és `CoglMFCDialogDlg` nevű osztályokat, valamint az ezeket tartalmazó header és cpp forrásállományokat.

Moduláris programozás szempontjából jó, ha az OpenGL-t kezelő programrészeket egy külön osztályba írjuk, ezért hozzunk létre egy osztályt a következő definíciókkal (*OpenGLControl.h* headerállomány):

```
1. #if !defined(AFX_OPENGLCONTROL_H__71C34264_3EB3_
2. 41CF_AD97_0A6020768E03__INCLUDED_)
3. #define AFX_OPENGLCONTROL_H__71C34264_3EB3_
4. 41CF_AD97_0A6020768E03__INCLUDED_
5.
6. #if _MSC_VER > 1000
7. #pragma once
8. #endif // _MSC_VER > 1000
1. // OpenGLControl.h : header file
9.
10.#include <GL\glut.h>
```

```
11.
2.  // COpenGLControl window
12.
13. class COpenGLControl : public CWnd
14. {
15. public:
16.   COpenGLControl();
17. public:
18.   UINT_PTR m_unpTimer;
19.   bool m_bIsMaximized;
20. private:
21.   CWnd    *hWnd;
22.   HDC     hdc;
23.   HGLRC   hrc;
24.   int     m_nPixelFormat;
25.   CRect   m_rect;
26.   CRect   m_oldWindow;
27.   CRect   m_originalRect;
28.   float   m_fPosX;
29.   float   m_fPosY;
30.   float   m_fZoom;
31.   float   m_fRotX;
32.   float   m_fRotY;
33.   float   m_fLastX;
34.   float   m_fLastY;
35. public:
36.   void oglCreate(CRect rect, CWnd *parent);
37.   void oglInitialize(void);
38.   void oglDrawScene(void);
39.   //{{AFX_VIRTUAL(COpenGLControl)
40.   //}}AFX_VIRTUAL
41. public:
42.   virtual ~COpenGLControl();
43. public:
44.   afx_msg void OnDraw(CDC *pDC);
45.   //{{AFX_MSG(COpenGLControl)
46.   afx_msg void OnPaint();
47.   afx_msg int OnCreate(LPCREATESTRUCT
48.                        lpCreateStruct);
49.   afx_msg void OnTimer(UINT nIDEvent);
50.   afx_msg void OnSize(UINT nType, int cx, int cy);
51.   afx_msg void OnMouseMove(UINT nFlags,
52.                            CPoint point);
53.   //}}AFX_MSG
54.   DECLARE_MESSAGE_MAP()
55. };
56.
57. #endif // !defined(AFX_OPENGLCONTROL_H__71C34264_
58. 3EB3_41CF_AD97_0A6020768E03__INCLUDED_)
```

Az OpenGL beállításait és a rajzoló kódrészletet az *OpenGLControl.cpp* forrásállományba írjuk be. Mivel itt az inicializálást már nem a GLUT végzi el, és ez erősen rendszerfüggő, egy pixelformátum leíróval be kell állítanunk a használatos rendszerelemeket és meg kell feleltetnünk ezeket a Windows GDI rendszerének. Az ablak frissítését a legegyszerűbb, ha egy időzítő (*timer*) segítségével végezzük. Külön kell ügyelnünk arra, hogy az ablak átméretezése hogyan érinti az OpenGL felület átméretezését. Itt teremtjük meg az OpenGL felület és a Windows kontrollok kapcsolatát is, valamint itt végezzük el az eseményvezérlést (az eseménykezelést sem a GLUT végzi).

```
1.   #include "stdafx.h"
2.   #include "oglMFCDialog.h"
3.   #include "oglMFCDialogDlg.h"
4.   #include "OpenGLControl.h"
```

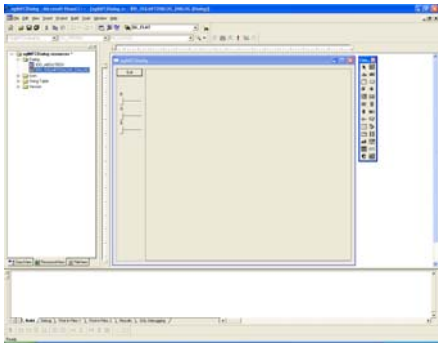
```
5.
6.    #ifdef _DEBUG
7.    #define new DEBUG_NEW
8.    #undef THIS_FILE
9.    static char THIS_FILE[] = __FILE__;
10.   #endif
11.
12.   COpenGLControl::COpenGLControl()
13.   {
14.     m_fPosX = 0.0f;
15.     m_fPosY = 0.0f;
16.     m_fZoom = 10.0f;
17.     m_fRotX = 0.0f;
18.     m_fRotY = 0.0f;
19.     m_fLastX = 0.0f;
20.     m_fLastY = 0.0f;
21.     m_bIsMaximized = false;
22.   }
23.
24.   COpenGLControl::~COpenGLControl()
25.   {
26.   }
27.
28.   void COpenGLControl::oglCreate(CRect rect,
29.                                  CWnd *parent)
30.   {
31.     CString className =
32.       AfxRegisterWndClass(CS_HREDRAW |
33.       CS_VREDRAW | CS_OWNDC, NULL,
34.       HBRUSH)GetStockObject(BLACK_BRUSH), NULL);
35.     CreateEx(0, className, "OpenGL", WS_CHILD |
36.       WS_VISIBLE | WS_CLIPSIBLINGS |
37.       WS_CLIPCHILDREN, rect, parent, 0);
38.     m_oldWindow = rect;
39.     m_originalRect = rect;
40.     hWnd = parent;
41.   }
42.
43.   BEGIN_MESSAGE_MAP(COpenGLControl, CWnd)
44.     //{{AFX_MSG_MAP(COpenGLControl)
45.     ON_WM_PAINT()
46.     ON_WM_CREATE()
47.     ON_WM_TIMER()
48.     ON_WM_SIZE()
49.     ON_WM_MOUSEMOVE()
50.     //}}AFX_MSG_MAP
51.   END_MESSAGE_MAP()
52.
53.   void COpenGLControl::OnPaint()
54.   {
55.     ValidateRect(NULL);
56.   }
57.
58.   int COpenGLControl::OnCreate(LPCREATESTRUCT
59.       lpCreateStruct)
60.   {
61.     if (CWnd::OnCreate(lpCreateStruct) == -1)
62.       return -1;
63.     oglInitialize();
64.     return 0;
65.   }
66.
67.   void COpenGLControl::oglInitialize(void)
68.   {
69.     static PIXELFORMATDESCRIPTOR pfd =
70.     {
```

```
71.       sizeof(PIXELFORMATDESCRIPTOR),
72.       1,
73.       PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL |
74.       PFD_DOUBLEBUFFER,
75.       PFD_TYPE_RGBA,
76.       32,    // bit depth
77.       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
78.       16,    // z-buffer depth
79.       0, 0, 0, 0, 0, 0, 0,
80.     };
81.     hdc = GetDC()->m_hDC;
82.     m_nPixelFormat = ChoosePixelFormat(hdc, &pfd);
83.     SetPixelFormat(hdc, m_nPixelFormat, &pfd);
84.     hrc = wglCreateContext(hdc);
85.     wglMakeCurrent(hdc, hrc);
86.     glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
87.     glClearDepth(1.0f);
88.     glFrontFace(GL_CCW);
89.     glCullFace(GL_BACK);
90.     glEnable(GL_DEPTH_TEST);
91.     glDepthFunc(GL_LEQUAL);
92.     GLfloat ambientLight[] = {0.3f, 0.3f,
93.                               0.3f, 1.0f};
94.     GLfloat diffuseLight[] = {0.7f, 0.7f,
95.                               0.7f, 1.0f};
96.     glEnable(GL_LIGHTING);
97.     glLightfv(GL_LIGHT0, GL_AMBIENT, ambientLight);
98.     glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuseLight);
99.     glEnable(GL_LIGHT0);
100.    glEnable(GL_DEPTH_TEST);
101.    glShadeModel(GL_SMOOTH);
102.    glDisable(GL_CULL_FACE);
103.    glEnable(GL_COLOR_MATERIAL);
104.    glColorMaterial(GL_FRONT,
105.                    GL_AMBIENT_AND_DIFFUSE);
106.    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
107.    OnDraw(NULL);
108. }
109.
110. void COpenGLControl::OnDraw(CDC *pDC)
111. {
112.    glLoadIdentity();
113.    glTranslatef(0.0f, 0.0f, -m_fZoom);
114.    glTranslatef(m_fPosX, m_fPosY, 0.0f);
115.    glRotatef(m_fRotX, 1.0f, 0.0f, 0.0f);
116.    glRotatef(m_fRotY, 0.0f, 1.0f, 0.0f);
117. }
118.
119. void COpenGLControl::OnTimer(UINT nIDEvent)
120. {
121.    switch (nIDEvent)
122.    {
123.      case 1:
124.      {
125.        glClear(GL_COLOR_BUFFER_BIT |
126.                GL_DEPTH_BUFFER_BIT);
127.         oglDrawScene();
128.         SwapBuffers(hdc);
129.         break;
130.      }
131.      default:
132.         break;
133.    }
134.    CWnd::OnTimer(nIDEvent);
135. }
136.
```

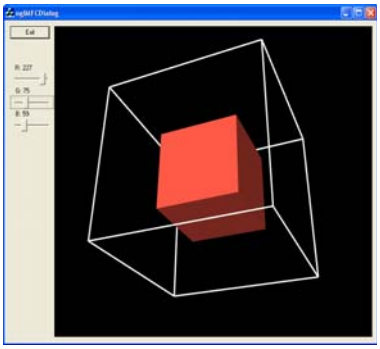
```
137. void COpenGLControl::OnSize(UINT nType,
138.                            int cx, int cy)
139. {
140.   CWnd::OnSize(nType, cx, cy);
141.   if (0 >= cx || 0 >= cy ||
142.       nType == SIZE_MINIMIZED) return;
143.   glViewport(0, 0, cx, cy);
144.   glMatrixMode(GL_PROJECTION);
145.   glLoadIdentity();
146.   gluPerspective(35.0f, (float)cx / (float)cy,
147.                  0.01f, 2000.0f);
148.   glMatrixMode(GL_MODELVIEW);
149.   switch (nType)
150.   {
151.     case SIZE_MAXIMIZED:
152.     {
153.       GetWindowRect(m_rect);
154.       MoveWindow(100, 6, cx - 110, cy – 14);
155.       GetWindowRect(m_rect);
156.       m_oldWindow = m_rect;
157.       break;
158.     }
159.     case SIZE_RESTORED:
160.     {
161.       if (m_bIsMaximized)
162.       {
163.          GetWindowRect(m_rect);
164.          MoveWindow(m_oldWindow.left,
165.                     m_oldWindow.top – 18,
166.                     m_originalRect.Width() – 4,
167.                     m_originalRect.Height() – 4);
168.          GetWindowRect(m_rect);
169.          m_oldWindow = m_rect;
170.       }
171.       break;
172.    }
173. }
174.   GLfloat lightPos[] = {-50.f, 50.0f,
175.                         100.0f, 1.0f};
176.   glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
177. }
178.
179. void COpenGLControl::oglDrawScene()
180. {
181.   COglMFCDialogDlg* d =
182.     (COglMFCDialogDlg*)theApp.GetMainWnd();
183.   glColor3ub(d->m_SliderR.GetPos(),
184.     d->m_SliderG.GetPos(),d->m_SliderB.GetPos());
185.   d->m_StaticR.Format("R: %i",
186.     d->m_SliderR.GetPos());
187.   d->m_StaticG.Format("G: %i",
188.     d->m_SliderG.GetPos());
189.   d->m_StaticB.Format("B: %i",
190.     d->m_SliderB.GetPos());
191.   d->UpdateData(false);
192.   glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
193.   glutSolidCube(1);
194.   glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
195.   glLineWidth(3);
196.   glBegin(GL_QUADS);
197.     glColor3f(1.0f, 1.0f, 1.0f);
198.     glVertex3f(1.0f, 1.0f, 1.0f);
199.     glVertex3f(1.0f, 1.0f, –1.0f);
200.     glVertex3f(–1.0f, 1.0f, –1.0f);
201.     glVertex3f(–1.0f, 1.0f, 1.0f);
202.     glVertex3f(–1.0f, –1.0f, –1.0f);
```

```
203.      glVertex3f(1.0f, -1.0f, -1.0f);
204.      glVertex3f(1.0f, -1.0f, 1.0f);
205.      glVertex3f(-1.0f, -1.0f, 1.0f);
206.      glVertex3f(1.0f, 1.0f, 1.0f);
207.      glVertex3f(-1.0f, 1.0f, 1.0f);
208.      glVertex3f(-1.0f, -1.0f, 1.0f);
209.      glVertex3f(1.0f, -1.0f, 1.0f);
210.      glVertex3f(-1.0f, -1.0f, -1.0f);
211.      glVertex3f(-1.0f, 1.0f, -1.0f);
212.      glVertex3f(1.0f, 1.0f, -1.0f);
213.      glVertex3f(1.0f, -1.0f, -1.0f);
214.      glVertex3f(-1.0f, -1.0f, -1.0f);
215.      glVertex3f(-1.0f, -1.0f, 1.0f);
216.      glVertex3f(-1.0f, 1.0f, 1.0f);
217.      glVertex3f(-1.0f, 1.0f, -1.0f);
218.      glVertex3f(1.0f, 1.0f, 1.0f);
219.      glVertex3f(1.0f, -1.0f, 1.0f);
220.      glVertex3f(1.0f, -1.0f, -1.0f);
221.      glVertex3f(1.0f, 1.0f, -1.0f);
222.    glEnd();
223. }
224.
225. void COpenGLControl::OnMouseMove(UINT nFlags,
226.                                   CPoint point)
227. {
228.    int diffX = (int)(point.x - m_fLastX);
229.    int diffY = (int)(point.y - m_fLastY);
230.    m_fLastX  = (float)point.x;
231.    m_fLastY  = (float)point.y;
232.    if (nFlags & MK_LBUTTON)
233.    {
234.      m_fRotX += (float)0.5f * diffY;
235.      if ((m_fRotX > 360.0f) ||
236.          (m_fRotX < -360.0f))
237.        m_fRotX = 0.0f;
238.      m_fRotY += (float)0.5f * diffX;
239.      if ((m_fRotY > 360.0f) ||
240.          (m_fRotY < -360.0f))
241.        m_fRotY = 0.0f;
242.    }
243.    else if (nFlags & MK_RBUTTON)
244.      m_fZoom -= (float)0.1f * diffY;
245.    else if (nFlags & MK_MBUTTON)
246.    {
247.      m_fPosX += (float)0.05f * diffX;
248.      m_fPosY -= (float)0.05f * diffY;
249.    }
250.    OnDraw(NULL);
251.    CWnd::OnMouseMove(nFlags, point);
252. }
```

A többi beállítást a varázsló által generált header és forrásállományokban végezzük el. Miután megterveztük a felületet (erőforrásként, resource view, dialog), a legfontosabb, hogy a `COglMFCDialogDlg` osztályba helyezzünk el egy `COpenGLControl m_oglWindow;` változót, így tudjuk megteremteni a kapcsolatot az OpenGL felület és az MFC dialógusablak között.



A dialógusablak, az OpenGL felületet egy Picture testesíti meg



Egyszerű MFC–OpenGL példa

Ezután a `COglMFCDialogDlg::OnInitDialog()` függvénybe helyezzük el a következő kódrészt:

```
1.    CRect rect;
2.    GetDlgItem(IDC_OPENGL)->GetWindowRect(rect);
3.    ScreenToClient(rect);
4.    m_oglWindow.oglCreate(rect, this);
5.    m_oglWindow.m_unpTimer = m_oglWindow.SetTimer(1,
6.        1, 0);
```

Hogy az ablak átméretezése tökéletesen működjön, a következő függvényt kell még megírnunk:

```
1.  void COglMFCDialogDlg::OnSize(UINT nType,
2.                                int cx, int cy)
3.  {
4.    CDialog::OnSize(nType, cx, cy);
5.    switch(nType)
6.    {
7.      case SIZE_RESTORED:
8.      {
9.         if (m_oglWindow.m_bIsMaximized)
10.        {
11.          m_oglWindow.OnSize(nType, cx, cy);
12.          m_oglWindow.m_bIsMaximized = false;
13.        }
14.        break;
15.     }
16.     case SIZE_MAXIMIZED:
17.     {
18.       m_oglWindow.OnSize(nType, cx, cy);
19.       m_oglWindow.m_bIsMaximized = true;
20.       break;
21.     }
22.   }
23. }
```

Kovács Lehel