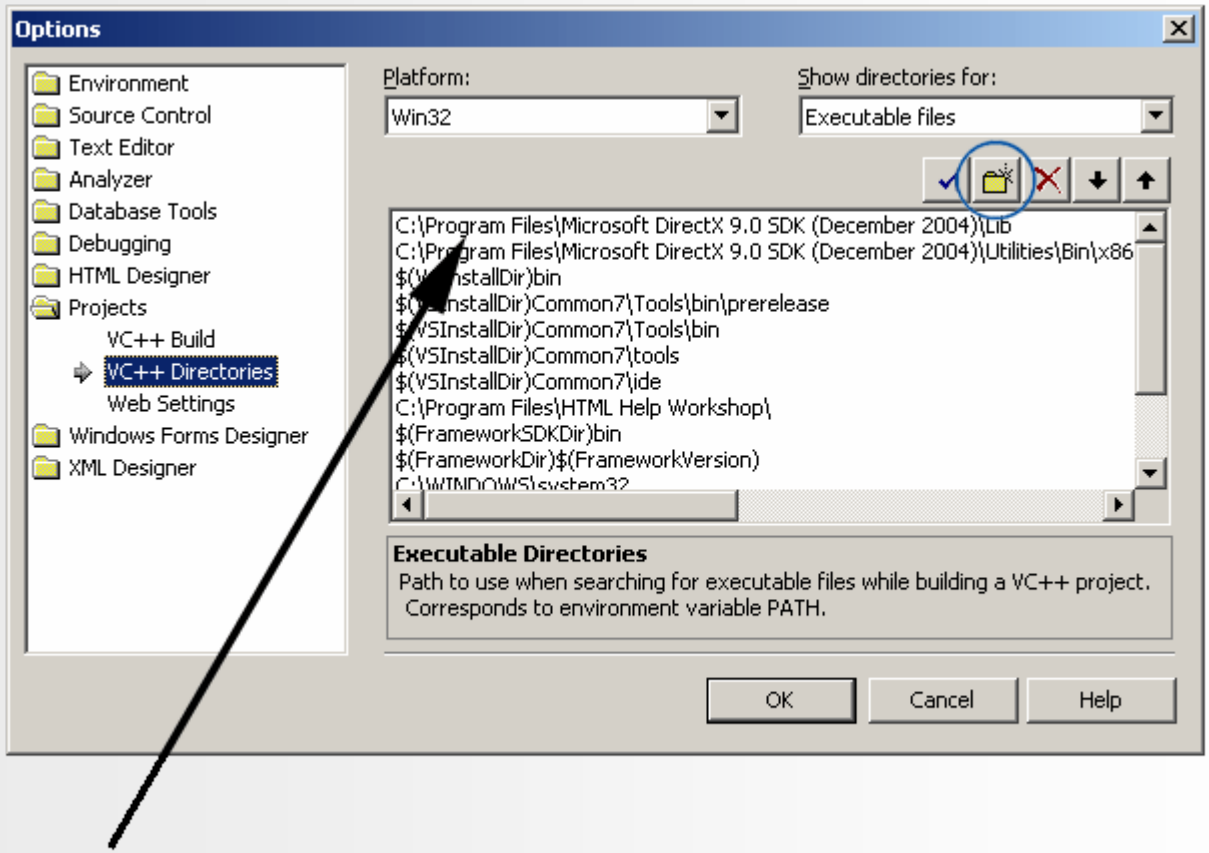
DirectX9 felhasználása számítógépes grafikában (bevezető – egy primitív keretrendszer)

2006. április 26.

Róth Ágoston

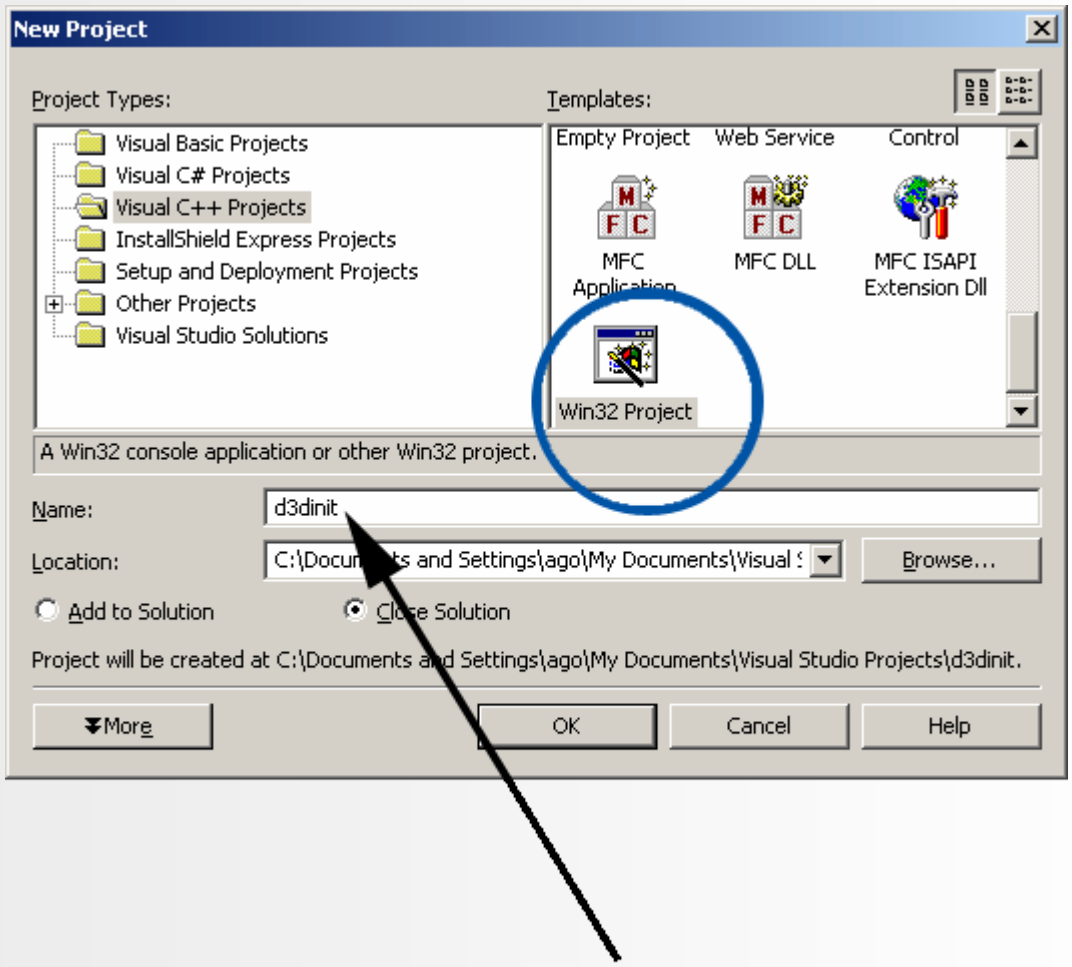
DirectX 9.0 SDK telepítése után

A fejlesztői környezeten (VC++ 7.0) belül: Tools/Options/Projects Folder/VC++ Directories



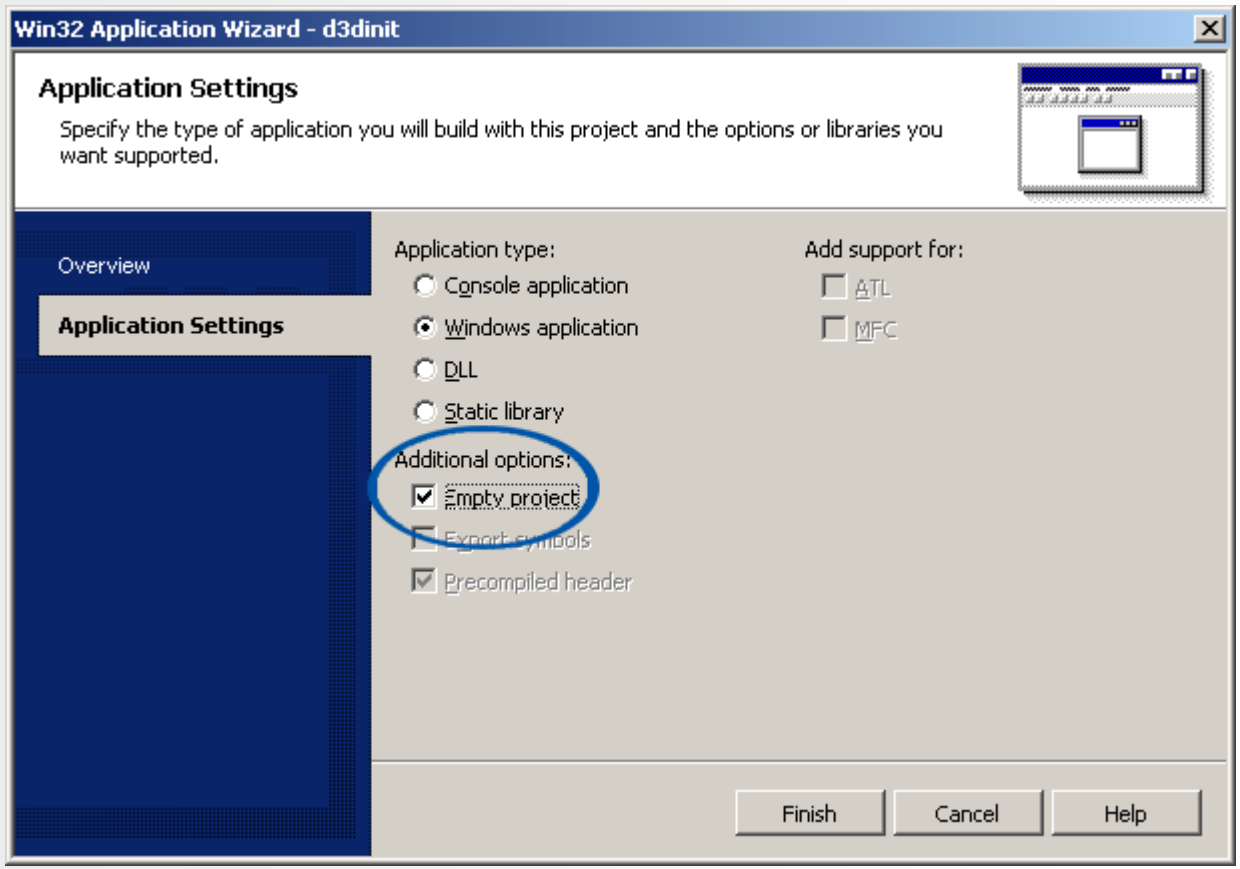
DirectX9 fejléc (*.h) és könyvtár (*.lib) állományainak elérési útvonala

New Win32 Project – d3dinit



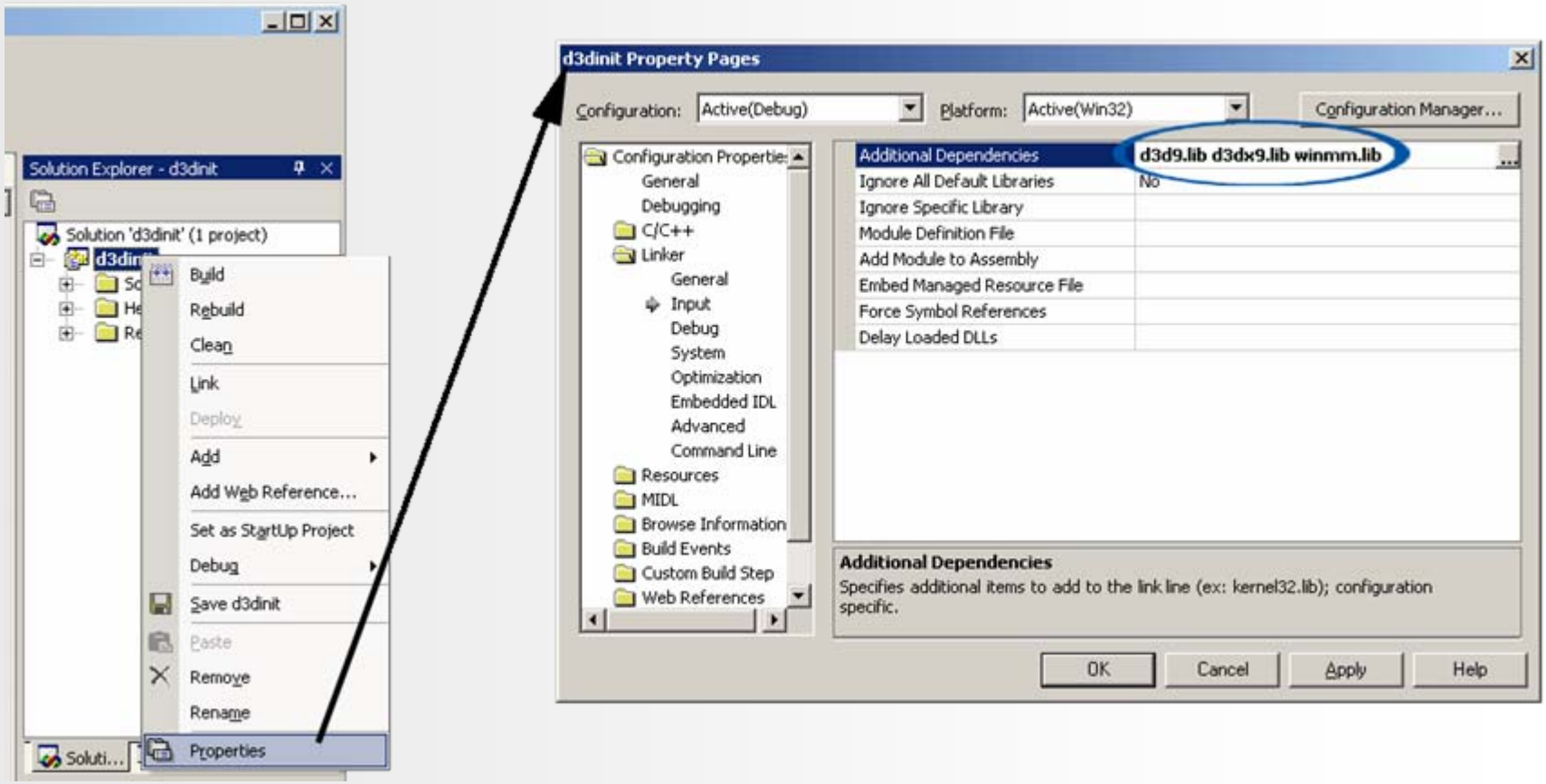
az új projektünk neve

d3dinit – üres projekt (az egyszerűség kedvéért)



d3dinit – függőségek beállítása

A fejlesztői környezet Project/Properties/Linker Folder/Input menüpontját kiválasztva...



d3dUtility.h

```
#include <d3dx9.h>
#include <string>

namespace d3d
{
            bool InitD3D(
                        HINSTANCE hInstance,       // [be] alkalmazás példánya
                        int width, int height,     // [be] az ablak mérete/backbuffer mérete
                        bool windowed,             // [be] ablakozott (true) vagy teljes képernyős (false)
                        D3DDEVTYPE deviceType,     // [be] HAL vagy REF
                        IDirect3DDevice9** device);// [ki] a létrehozott eszköz

            int EnterMsgLoop(
                        bool (*ptr_display)(float timeDelta));

            LRESULT CALLBACK WndProc(
                        HWND hwnd,
                        UINT msg,
                        WPARAM wParam,
                        LPARAM lParam);

            template<class T> void Release(T t)
            {
                        if( t )
                        {
                                    t->Release();
                                    t = 0;
                        }
            }

            template<class T> void Delete(T t)
            {
                        if( t )
                        {
                                    delete t;
                                    t = 0;
                        }
            }
}
```

d3dUtility.cpp, d3d::InitD3D(...) – 0. lépés

```
bool d3d::InitD3D( HINSTANCE hInstance, int width, int height, bool windowed,
                   D3DDEVTYPE deviceType, IDirect3DDevice9** device)
{
            // 0. lépés: létrehozzuk az alkalmazás fő ablakát (ez a rész független DirectX-től)
            WNDCLASS wc;

            wc.style         = CS_HREDRAW | CS_VREDRAW;
            wc.lpfnWndProc   = (WNDPROC) d3d::WndProc;
            wc.cbClsExtra    = 0;
            wc.cbWndExtra    = 0;
            wc.hInstance     = hInstance;
            wc.hIcon         = LoadIcon(0, IDI_APPLICATION);
            wc.hCursor       = LoadCursor(0, IDC_ARROW);
            wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
            wc.lpszMenuName  = 0;
            wc.lpszClassName = "Direct3D9App";

            if( !RegisterClass(&wc) )
            {
                        ::MessageBox(0, "RegisterClass() - FAILED", 0, 0);
                        return false;
            }

            HWND hwnd = 0;
            hwnd = ::CreateWindow("Direct3D9App", "Direct3D9App",
                        WS_EX_TOPMOST,
                        0, 0, width, height,
                        0 /*parent hwnd*/, 0 /* menu */, hInstance, 0 /*extra*/);

            if( !hwnd )
            {
                        ::MessageBox(0, "CreateWindow() - FAILED", 0, 0);
                        return false;
            }

            ::ShowWindow(hwnd, SW_SHOW);
            ::UpdateWindow(hwnd);
```

d3dUtility.cpp, d3d::InitD3D(...) – 1., 2., 3. lépés

```
// a D3D tényleges inicializálása
HRESULT hr = 0;

// 1. lépés: létrehozzuk az IDirect3D9 objektumot
IDirect3D9* d3d9 = 0;
d3d9 = Direct3DCreate9(D3D_SDK_VERSION);

if( !d3d9 )
{
        ::MessageBox(0, "Direct3DCreate9() - FAILED", 0, 0);
        return false;
}

// 2. lépés: ellenőrizzük, hogy a videokártya képes-e a vertexek hardwares feldolgozására
D3DCAPS9 caps; // egy képességek/lehetőségek-struktúra
d3d9->GetDeviceCaps(D3DADAPTER_DEFAULT, deviceType, &caps);

int vp = 0;
if( caps.DevCaps & D3DDEVCAPS_HWTRANSFORMANDLIGHT )
        vp = D3DCREATE_HARDWARE_VERTEXPROCESSING;
else
        vp = D3DCREATE_SOFTWARE_VERTEXPROCESSING;

// 3. lépés: kitöltünk egy D3DPRESENT_PARAMETERS struktúrát
D3DPRESENT_PARAMETERS d3dpp;     // megjelenítő paraméterek
d3dpp.BackBufferWidth            = width;
d3dpp.BackBufferHeight           = height;
d3dpp.BackBufferFormat           = D3DFMT_A8R8G8B8; //pixelek szerkezete (4 x 8 bit → alpha, red, green, blue)
d3dpp.BackBufferCount            = 1;
d3dpp.MultiSampleType            = D3DMULTISAMPLE_NONE;
d3dpp.MultiSampleQuality         = 0;
d3dpp.SwapEffect                 = D3DSWAPEFFECT_DISCARD;
d3dpp.hDeviceWindow              = hwnd;
d3dpp.Windowed                   = windowed;
d3dpp.EnableAutoDepthStencil     = true;
d3dpp.AutoDepthStencilFormat     = D3DFMT_D24S8; //mélység-buffer szerkezete (24 bit → depth, 8 bit → stencil)
d3dpp.Flags                      = 0;
d3dpp.FullScreen_RefreshRateInHz = D3DPRESENT_RATE_DEFAULT;
d3dpp.PresentationInterval       = D3DPRESENT_INTERVAL_IMMEDIATE;
```

d3dUtility.cpp, d3d::InitD3D(...) – 4. lépés

```
        // 4. lépés: létrehozzuk az eszközt

        hr = d3d9->CreateDevice(
                    D3DADAPTER_DEFAULT, // elsődleges adapter (videokártya)
                    deviceType,         // eszköz típusa
                    hwnd,               // az eszközhöz társított ablak
                    vp,                 // vertexek feldolgozása
                    &d3dpp,             // megjelenítő paraméterek
                    device);            // a létrehozandó eszköz

        if( FAILED(hr) )
        {
                    // újból próbálkozunk, de most csak egy 16 bites mélység-bufferrel
                    d3dpp.AutoDepthStencilFormat = D3DFMT_D16;

                    hr = d3d9->CreateDevice(
                                D3DADAPTER_DEFAULT,
                                deviceType,
                                hwnd,
                                vp,
                                &d3dpp,
                                device);

                    if( FAILED(hr) )
                    {
                                d3d9->Release(); // felszabadítjuk a d3d9 objektumot
                                ::MessageBox(0, "CreateDevice() - FAILED", 0, 0);
                                return false;
                    }
        }

        d3d9->Release(); // felszabadítjuk a d3d9 objektumot

        return true;
}
```

d3dUtility.cpp, d3d::EnterMsgLoop(...)

```
// ez a függvény rejti el az alkalmazás üzenetkezelő függvényét
// paraméterként egy olyan függvény pointerét várja, amely a kirajzolást végzi
// (muszáj megadnunk, mert akkor végezzük a rajzolást, amikor az alkalmazásunk
//  semmilyen más üzenetet [billentyű, egér, stb.] nem kell feldolgozzon)

int d3d::EnterMsgLoop( bool (*ptr_display)(float timeDelta) )
{
            MSG msg;
            ::ZeroMemory(&msg, sizeof(MSG));

            static float lastTime = (float)timeGetTime();

            while(msg.message != WM_QUIT)
            {
                        if(::PeekMessage(&msg, 0, 0, 0, PM_REMOVE))
                        {
                                    ::TranslateMessage(&msg);
                                    ::DispatchMessage(&msg);
                        }
                        else
        {
                                    float currTime  = (float)timeGetTime();
                                    float timeDelta = (currTime - lastTime)*0.001f;

                                    ptr_display(timeDelta);

                                    lastTime = currTime;
        }
    }
    return msg.wParam;
}
```

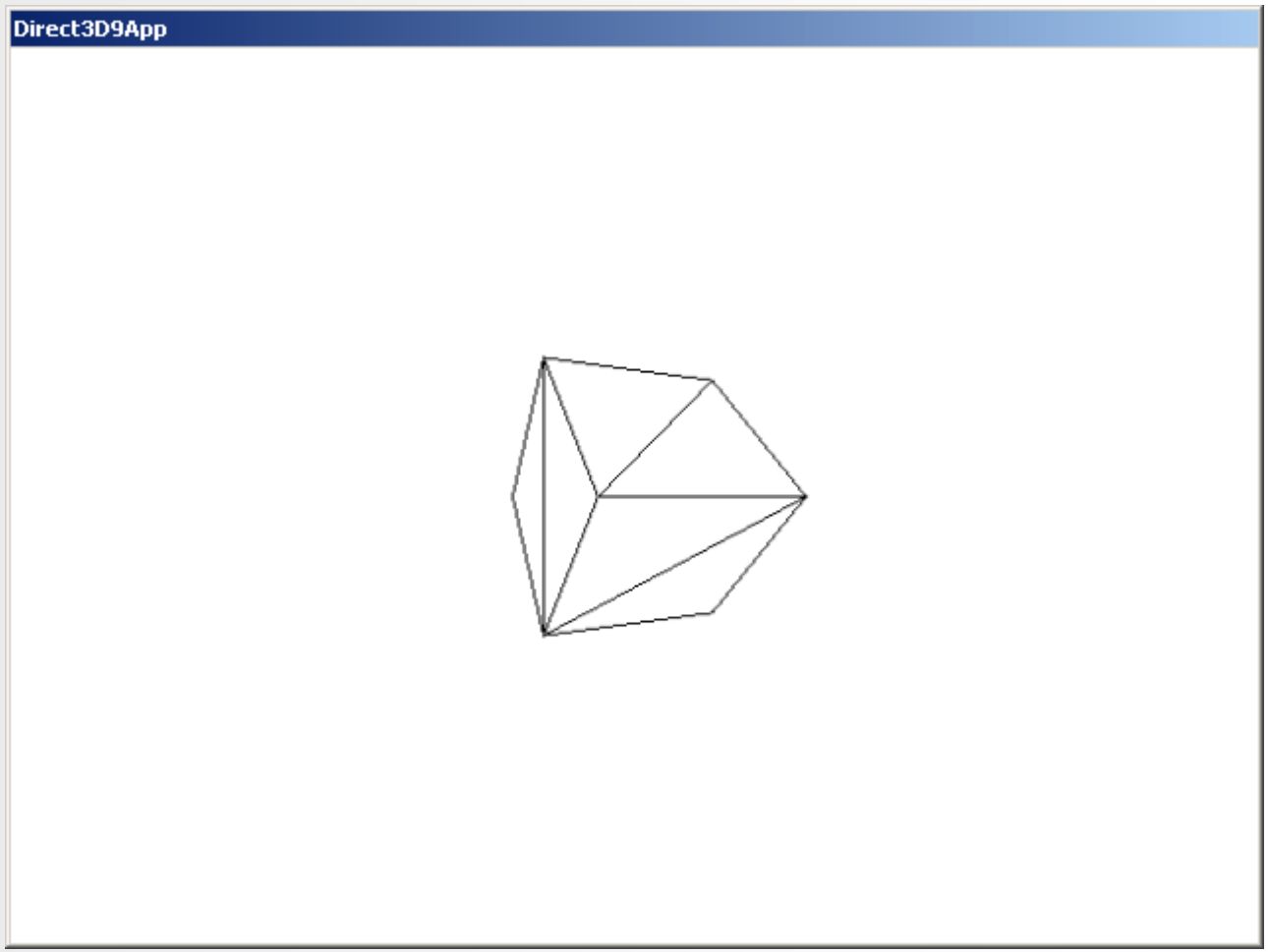
d3dInit.cpp – egy keretrendszer: Setup/Cleanup/Display

```
#include "d3dUtility.h"


// globális változók


IDirect3DDevice9* Device = 0;


// keretrendszer függvények: Setup, Cleanup, Display

bool Setup()
{
            // nincs mit beállítsunk/inicializáljunk ebben a példában

            return true;
}

void Cleanup()
{
            // nincs mit felszabadítsunk ebben a példában
}

// a rajzoló/renderelő függvényünk
bool Display(float timeDelta)
{
  if( Device ) // csak akkor használjuk a Device eszköz metódusait, ha egy érvényes/létező eszköz
  {
    // arra utasítjuk az eszközt, hogy a háttér-buffer minden pixelét fekete színűre
    // [D3DCLEAR_TARGET: 0x00000000 (fekete)],
    // míg a mélység buffer minden elemét 1.0-re állítsa
    // [D3DCLEAR_ZBUFFER: 1.0f]
    Device->Clear(0, 0, D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER, 0x00000000, 1.0f, 0);

    // megcseréljük a háttér- és előtér-buffereket
    Device->Present(0, 0, 0, 0);
  }
  return true;
}
```

d3dInit.cpp (folytatás)

```
// WndProc
LRESULT CALLBACK d3d::WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
            switch( msg )
            {
            case WM_DESTROY:
                        ::PostQuitMessage(0);
                        break;

            case WM_KEYDOWN:
                        if( wParam == VK_ESCAPE )           //ESCAPE-re kilépünk
                                    ::DestroyWindow(hwnd);
                        break;
            }
            return ::DefWindowProc(hwnd, msg, wParam, lParam);
}

// WinMain
int WINAPI WinMain(HINSTANCE hinstance, HINSTANCE prevInstance, PSTR cmdLine, int showCmd)
{
            if(!d3d::InitD3D(hinstance,640, 480, true, D3DDEVTYPE_HAL, &Device))
            {
                        ::MessageBox(0, "InitD3D() - FAILED", 0, 0);
                        return 0;
            }

            if(!Setup())
            {
                        ::MessageBox(0, "Setup() - FAILED", 0, 0);
                        return 0;
            }

            d3d::EnterMsgLoop( Display );

            Cleanup();

            Device->Release();
            return 0;                                                          exe
}
```

Alkalmazás – Drótvázas kocka



cube.cpp – a keretrendszer függvényeinek módosulása

```
//globális változók

IDirect3DDevice9* Device = 0;

IDirect3DVertexBuffer9* VB = 0; // vertex buffer
IDirect3DIndexBuffer9*  IB = 0; // index buffer

struct Vertex
{
            Vertex(){}
            Vertex(float x, float y, float z)
            {
                        _x = x;  _y = y;  _z = z;
            }
            float _x, _y, _z;
            static const DWORD FVF;
};
const DWORD Vertex::FVF = D3DFVF_XYZ;

bool Setup()
{
            // vertex és index bufferek létrehozása
            Device->CreateVertexBuffer(
                        8 * sizeof(Vertex),
                        D3DUSAGE_WRITEONLY,
                        Vertex::FVF,
                        D3DPOOL_MANAGED,
                        &VB,
                        0);

            Device->CreateIndexBuffer(
                        36 * sizeof(WORD),
                        D3DUSAGE_WRITEONLY,
                        D3DFMT_INDEX16,
                        D3DPOOL_MANAGED,
                        &IB,
                        0);
```

cube.cpp (folytatás)

```
// a bufferek feltöltése

// egyedi vertexek: a kocka 8 csúcsa
Vertex* vertices;
VB->Lock(0, 0, (void**)&vertices, 0);

vertices[0] = Vertex(-1.0f, -1.0f, -1.0f);
vertices[1] = Vertex(-1.0f,  1.0f, -1.0f);
vertices[2] = Vertex( 1.0f,  1.0f, -1.0f);
vertices[3] = Vertex( 1.0f, -1.0f, -1.0f);
vertices[4] = Vertex(-1.0f, -1.0f,  1.0f);
vertices[5] = Vertex(-1.0f,  1.0f,  1.0f);
vertices[6] = Vertex( 1.0f,  1.0f,  1.0f);
vertices[7] = Vertex( 1.0f, -1.0f,  1.0f);

VB->Unlock();

// a kocka oldallapjait alkotó háromszögek meghatározása
WORD* indices = 0;
IB->Lock(0, 0, (void**)&indices, 0);

// elől
indices[0]  = 0; indices[1]  = 1; indices[2]  = 2;
indices[3]  = 0; indices[4]  = 2; indices[5]  = 3;

// hátul
indices[6]  = 4; indices[7]  = 6; indices[8]  = 5;
indices[9]  = 4; indices[10] = 7; indices[11] = 6;

// bal
indices[12] = 4; indices[13] = 5; indices[14] = 1;
indices[15] = 4; indices[16] = 1; indices[17] = 0;

// jobb
indices[18] = 3; indices[19] = 2; indices[20] = 6;
indices[21] = 3; indices[22] = 6; indices[23] = 7;
```

cube.cpp (folytatás)

```
            // felül
            indices[24] = 1; indices[25] = 5; indices[26] = 6;
            indices[27] = 1; indices[28] = 6; indices[29] = 2;

            // alul
            indices[30] = 4; indices[31] = 0; indices[32] = 3;
            indices[33] = 4; indices[34] = 3; indices[35] = 7;

            IB->Unlock();

            // elhelyezünk egy kamerát a világban
            D3DXVECTOR3 position(0.0f, 0.0f, -5.0f);
            D3DXVECTOR3 target(0.0f, 0.0f, 0.0f);
            D3DXVECTOR3 up(0.0f, 1.0f, 0.0f);

            //a nézeti mátrix beállítása
            D3DXMATRIX  V;
            D3DXMatrixLookAtLH(&V, &position, &target, &up);
            Device->SetTransform(D3DTS_VIEW, &V);

            // a vetítési mátrix beállítása
            D3DXMATRIX proj;
            D3DXMatrixPerspectiveFovLH(&proj,
                                       D3DX_PI * 0.5f, // 90 - degree
                                       (float)Width / (float)Height,
                                       1.0f,
                                       1000.0f);
            Device->SetTransform(D3DTS_PROJECTION, &proj);

            // megjelenítési mód: drótváz
            Device->SetRenderState(D3DRS_FILLMODE, D3DFILL_WIREFRAME);

            return true;
}

void Cleanup()
{
            d3d::Release<IDirect3DVertexBuffer9*>(VB);
            d3d::Release<IDirect3DIndexBuffer9*>(IB);
}
```

cube.cpp (folytatás)

```
bool Display(float timeDelta)
{
            if( Device )
            {
                        // forgatjuk a kockát az Ox, Oy tengelyek körül
                        D3DXMATRIX Rx, Ry;

                        // 45 fokkal az Ox körül
                        D3DXMatrixRotationX(&Rx, D3DX_PI / 4.0f);

                        // minden képkeretnél az Oy tengely körüli forgatási szög nagyságát növeljük
                        static float y = 0.0f;
                        D3DXMatrixRotationY(&Ry, y);
                        y += timeDelta;

                        // ha túl léptük 2*pi-t, akkor nullázzuk a szöget
                        if( y >= 6.28f )
                                    y = 0.0f;

                        // ötvözzük az Ox és Oy tengelyek körüli forgatási transzformációkat
                        D3DXMATRIX p = Rx * Ry;

                        Device->SetTransform(D3DTS_WORLD, &p);

                        // megrajzoljuk a színteret
                        Device->Clear(0, 0, D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER, 0xffffffff, 1.0f, 0);
                        Device->BeginScene();

                                    Device->SetStreamSource(0, VB, 0, sizeof(Vertex));
                                    Device->SetIndices(IB);
                                    Device->SetFVF(Vertex::FVF);

                                    // kocka kirajzolása
                                    Device->DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 0, 8, 0, 12);

                        Device->EndScene();
                        Device->Present(0, 0, 0, 0);
            }
            return true;
}
```

exe