Programowanie Grafiki Komputerowej

Microsoft DirectX cz. 1 - wprowadzenie, Windows API, podstawy Direct3D

Grzegorz Łukawski

Katedra Informatyki, Politechnika Świętokrzyska w Kielcach



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



Politechnika Świętokrzyska

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

" Program Rozwojowy Potencjału Dydaktycznego Politechniki Świętokrzyskiej w Kielcach: *kształcenie w nowoczesnych obszarach techniki* "

Program Operacyjny Kapitał Ludzki, Priorytet IV Szkolnictwo wyższe i nauka, Działanie 4.1. Wzmocnienie i rozwój potencjału dydaktycznego uczelni oraz zwiększenie liczby absolwentów kierunków o kluczowym znaczeniu dla gospodarki opartej na wiedzy, Poddziałanie 4.1.1. Wzmocnienie potencjału dydaktycznego uczelni.

Umowa UDA–POKL.04.01.01-00395/09-00

Spis treści

1. Bibliografia
2. Wstęp
   - Komponenty DirectX
   - Programowanie z użyciem DirectX
3. Windows API
   - Szablon programu
   - Funkcja WinMain
   - Funkcja WindowProc
   - Przykładowy program (dx-p1)
4. Programowanie z użyciem Direct3D
   - Inicjalizacja Direct3D
   - Zamknięcie Direct3D
   - Funkcja renderująca
   - Przykładowy program (dx-p2)
5. Koniec

Uwagi

- Animacje są uruchamiane, gdy plik PDF jest wyświetlany na pełnym ekranie.
- Animacja jest uruchamiana dopiero po naciśnięciu spacji (program Adobe Reader).
- Klatki animacji wyświetlane są co 2 sekundy.
- Symulacje są uruchamiane po kliknięciu w ikonę z odpowiednim systemem operacyjnym. Przeglądarka wyświetli okno z ostrzeżeniem, należy go zaakceptować. W systemie Linux należy dodatkowo kliknąć dwukrotnie w nazwę programu.

Bibliografia

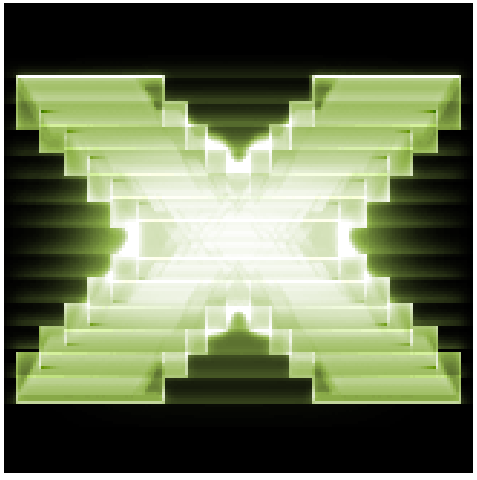

Foley, van Dam, Feiner, Hughes, Philips:
Wprowadzenie do grafiki komputerowej.
WNT 2001


Microsoft DirectX SDK:
Tutorials.


Wojciech Jawor:
Principia Silnika.


Mason McCuskey:
Programowanie gier w DirectX.
MIKOM 2003

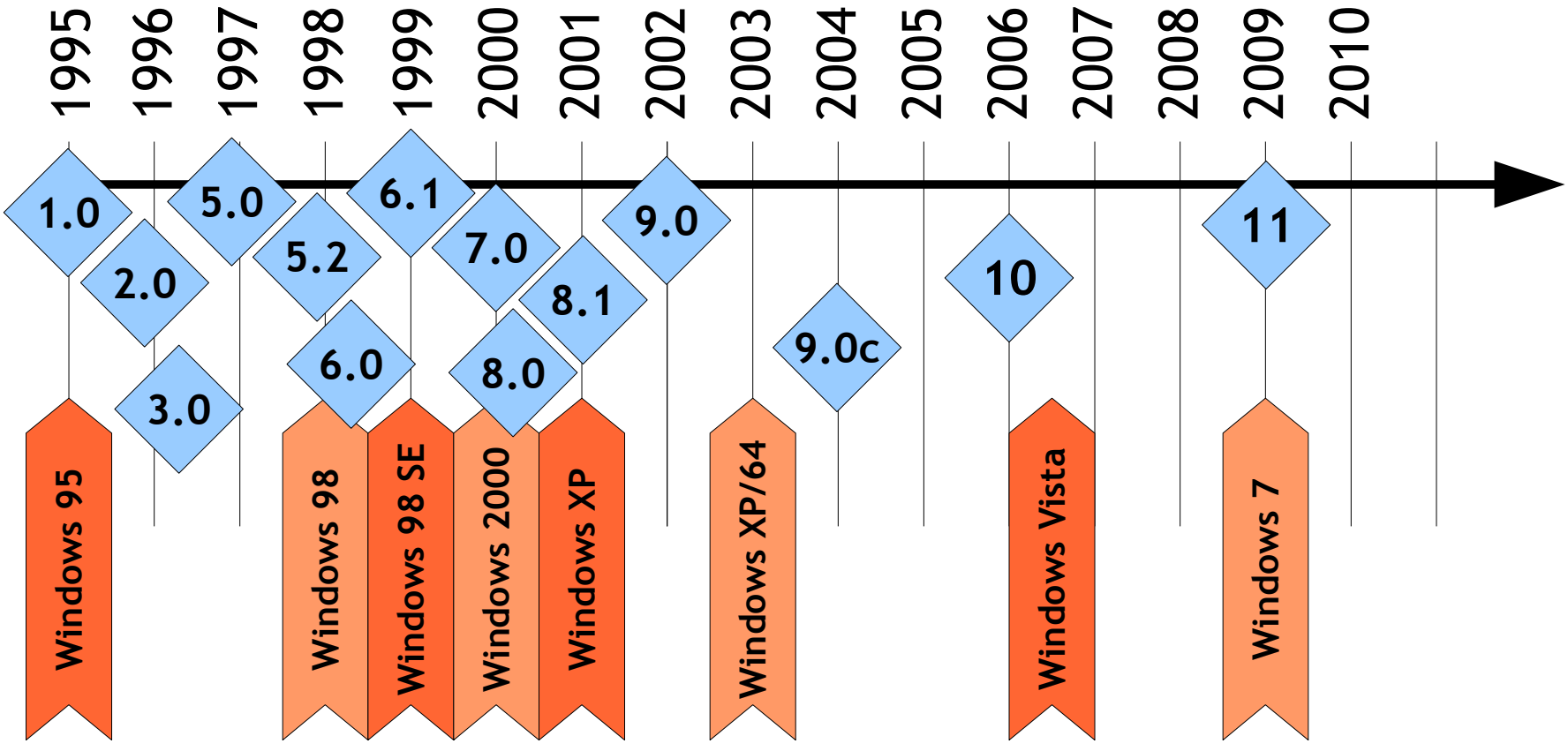
DirectX



Microsoft DirectX to zestaw bibliotek wspomagający generowanie grafiki 2D i 3D. Obejmuje także generowanie dźwięku, obsługę urządzeń we/wy (np. joysticka) i inne.
Ponieważ jest to produkt firmy Microsoft, DirectX jest dostępny wyłącznie na platformie Windows oraz konsoli XBOX.

Historia

30 września 1995 wydana została pierwsza wersja DirectX, udostępniona jako *Windows Games SDK* dla systemu Windows 95.



Komponenty (1/2)

Biblioteka DirectX składa się z szeregu komponentów odpowiedzialnych za realizację różnych zadań biblioteki:

- **DirectX Graphics** składające się z:
  - **DirectDraw** – obsługuje grafikę 2D;
  - **Direct3D** – obsługuje grafikę 3D;
  - **DirectGI** – umożliwia bezpośrednią obsługę sprzętu graficznego;
- **DirectInput** – obsługa urządzeń wejścia/wyjścia, jak klawiatura, mysz, joystick;
- **DirectPlay** – komponent używany w grach sieciowych;
- **DirectSound** – odtwarzanie i nagrywanie dźwięku;
- **DirectSound3D** – jak wyżej, ale dla dźwięku 3D;
- **DirectMusic** – odtwarzanie muzyki;

...

Komponenty (2/2)

...

- **DirectShow** – odtwarzanie multimediów (plików audio i wideo);
- **DirectX Media Objects** – podobnie do DirectShow;
- **DirectSetup** – obsługa instalacji i konfiguracji komponentów DirectX;
- **DirectWrite** – wspomaganie renderowania tekstu;
- **DirectCompute** – realizacja obliczeń (nie-graficznych) z pomocą GPU.

Wymagania

Narzędzia niezbędne do programowania z wykorzystaniem DirectX:

- **Komputer PC** z dowolną kartą graficzną i systemem Windows.
- **Microsoft DirectX SDK (Software Development Kit)**
  `http://msdn.microsoft.com/en-us/directx`
- **Microsoft Visual C++ Express**[1]
  `http://www.microsoft.com/express/Windows/`

[1] Można wykorzystać inny kompilator języka C++ dla systemu Windows. Podany kompilator zalecany jest ze względu na bezproblemową współpracę z DirectX SDK.

WinAPI

Windows API (Windows Application Programming Interface), nazywane też **WinAPI** to bezpośredni interfejs programistyczny systemu Microsoft Windows.

WinAPI obejmuje zestaw funkcji, stałych, typów zmiennych i struktur (między innymi) dla języka C.

WinAPI pozwala na obsługę systemu plików, urządzeń, procesów, sytuacji błędnych, sieci i innych. Pozwala na budowę i obsługę graficznego interfejsu użytkownika, czyli **GUI (Graphical User Interface)**. Interfejs **GDI (Graphics Device Interface)** bedący częścią składową WinAPI obsługuje grafikę 2D.

Typowy program w języku C korzystający z WinAPI składa się z kilku podstawowych elementów:

- Dyrektywy dołączającej funkcje i pozostałe elementy niezbędne do użycia WinAPI **#include** <**windows.h**>.
- Funkcji **WinMain()**, czyli głównej funkcji od której zaczyna się wykonanie programu.
- Funkcji **WindowProc()** obsługującej komunikaty programu i systemu Windows.

(1/4)

```
// Prototyp funkcji obsługującej komunikaty:
LRESULT CALLBACK WindowProc(HWND hWnd, UINT message,
                            WPARAM wParam, LPARAM lParam);

// Funkcja główna:
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                            LPSTR lpCmdLine, int nCmdShow)
{
// Deskryptor (uchwyt) okna:
HWND hWnd;

// Struktura przechowująca tzw. "klasę" okna:
WNDCLASSEX wc;
ZeroMemory(&wc, sizeof(WNDCLASSEX));

...
```

(2/4)

```
...
// Wypełnienie struktury "klasy" okna:
wc.cbSize = sizeof(WNDCLASSEX);
wc.style = CS_HREDRAW | CS_VREDRAW; // Odświeżanie okna pion i poziom
wc.lpfnWndProc = WindowProc; // Funkcja obsługi komunikatów
wc.hInstance = hInstance; // Instancja aplikacji (parametr)
wc.hCursor = LoadCursor(NULL, IDC_ARROW); // Standardowy kursor
wc.hbrBackground = (HBRUSH)COLOR_WINDOW; // Standardowy kolor tła okna
wc.lpszClassName = "WindowClass1"; // Nazwa "klasy" okna

// Zarejestrowanie "klasy" okna:
RegisterClassEx(&wc);

...
```

(3/4)

```
...
// Utworzenie okna i zapisanie jego deskryptora:
hWnd = CreateWindowEx(NULL,
    "WindowClass1", // Nazwa "klasy" okna
    "Hello DirectX", // Tytuł okna na pasku tytułowym,
    WS_OVERLAPPEDWINDOW, // Styl okna (domyślnie)
    100, // Pozycja X na ekranie
    100, // Pozycja Y na ekranie
    640, // Szerokość okna
    480, // Wysokość okna
    NULL, // Okno macierzyste (brak)
    NULL, // Menu okna (brak)
    hInstance, // Uchwyt aplikacji (parametr)
    NULL); // Wielokrotne okna (brak)

ShowWindow(hWnd, nCmdShow); // Wyświetlenie okna na ekranie

dx_init(hWnd); // Inicjalizacja DirectX!

...
```

(4/4)

```
...
// "Pompa" obsługi komunikatów:
MSG msg;
while(TRUE) {
    // Pobranie komunikatu bez blokowania:
    while(PeekMessage(&msg, NULL, 0, 0, PM_REMOVE)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    if(msg.message == WM_QUIT) break;

    dx_render(); // Funkcja renderująca DirectX!

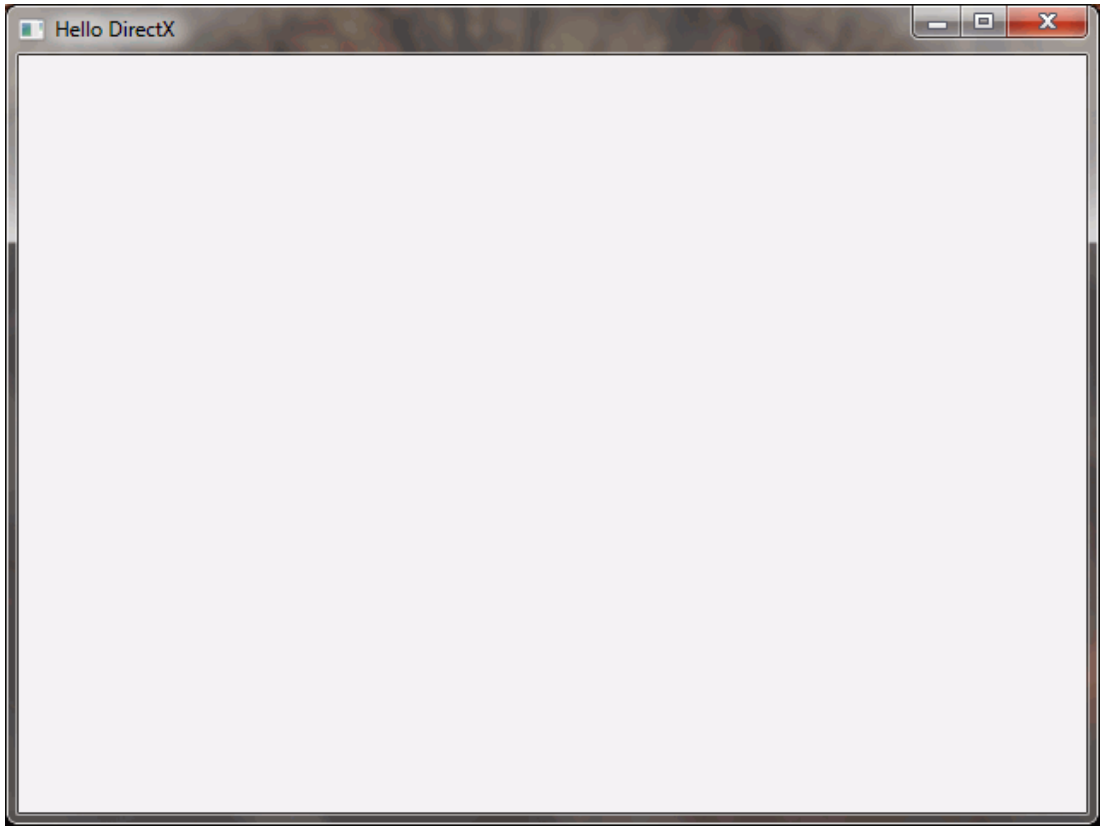
    // Zakończenie programu jeżeli naciśnięto klawisz ESC:
    if(KEY_DOWN(VK_ESCAPE)) PostMessage(hWnd, WM_DESTROY, 0, 0);
}

dx_kill(); // Zamknięcie DirectX!

// Zwrócenie kodu WM_QUIT do systemu Windows:
return msg.wParam;
}
```

```
LRESULT CALLBACK WindowProc(HWND hWnd, UINT message,
                            WPARAM wParam, LPARAM lParam)
{
    // Analiza kodu komunikatu:
    switch(message) {
        // Zamknięcie okna:
        case WM_DESTROY:
            PostQuitMessage(0); // Zakończenie programu
            return 0;
    }

    // Domyślna obsługa wszystkich pozostałych komunikatów:
    return DefWindowProc(hWnd, message, wParam, lParam);
}
```



Przykładowy program


Windows


Direct3D

Do programowania grafiki 3D wykorzystany zostanie interfejs **Direct3D**. Udostępnia od funkcjonalność i możliwości generowania grafiki 3D podobne do tych które oferuje biblioteka **OpenGL**.

Dołączenie biblioteki Direct3D 9

```
#include <d3d9.h>
#pragma comment (lib, "d3d9.lib")
```

Globalne dane Direct3D

```
LPDIRECT3D9 d3d; // Wskaźnik do interfejsu Direct3D
LPDIRECT3DDEVICE9 d3ddev; // Wskaźnik do urządzenia
```

Utworzenie interfejsu Direct3D:

```
// Domyślny parametr, zalecany dla zachowania kompatybilności:
d3d = Direct3DCreate9(D3D_SDK_VERSION);
```

Utworzenie struktury dla parametrów wyświetlania:

```
D3DPRESENT_PARAMETERS d3dpp;
ZeroMemory(&d3dpp, sizeof(d3dpp));

// Tryb w oknie (TRUE) czy pełnoekranowy (FALSE)?
d3dpp.Windowed = TRUE;
// Poprzednią ramkę usunąć:
d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
// Uchwyt okna Windows przeznaczonego do rysowania:
d3dpp.hDeviceWindow = hWnd;
```

Urządzenie Direct3D

Utworzenie urządzenia:

```
d3d->CreateDevice(
    // Domyślna karta graficzna:
    D3DADAPTER_DEFAULT,
    // HAL (Hardware Abstraction Layer) = sprzętowa akceleracja:
    D3DDEVTYPE_HAL,
    // Uchwyt okna Windows:
    hWnd,
    // Wszystkie kalkulacje 3D realizowane przez SOFTWARE
    // (najlepszy wybór dla zachowania kompatybilności):
    D3DCREATE_SOFTWARE_VERTEXPROCESSING,
    // Parametry wyświetlania:
    &d3dpp,
    // Wynik działania funkcji - "graphics device interface":
    &d3ddev);
```

Kalkulacje 3D

Możliwe ustawienia kalkulacji 3D realizowanych przez Direct3D (jeden z parametrów funkcji CreateDevice):

- `D3DCREATE_SOFTWARE_VERTEXPROCESSING` – realizacja programowa (najlepsza kompatybilność);
- `D3DCREATE_MIXED_VERTEXPROCESSING` – kombinacja sprzętu i oprogramowania, DirectX automatycznie dostosuje parametry;
- `D3DCREATE_HARDWARE_VERTEXPROCESSING` – realizacja sprzętowa (największa wydajność).

Zamknięcie biblioteki DirectX (Direct3D):

```
void dx_kill() {
    d3ddev->Release(); // Zamknięcie urządzenia
    d3d->Release(); // Zamknięcie Direct3D
}
```

Kodowanie kolorów i czyszczenie okna

Kodowanie przykładowych kolorów:

```
DWORD c1 = D3DCOLOR_XRGB(0, 255, 0); // Model RGB
DWORD c2 = D3DCOLOR_ARGB(136, 255, 0, 204); // Model RGBA
```

Czyszczenie okna:

```
d3ddev->Clear(0, NULL, D3DCLEAR_TARGET,
                       D3DCOLOR_XRGB(0, 0, 80), 1.0f, 0);
```

Rysowanie sceny

Wywołanie metody **BeginScene()** oznacza początek rysowania – jest to sygnał dla biblioteki DirectX, że przejmujemy kontrolę nad pamięcią. Bufor ramki jest blokowany (locked) dla naszego wyłącznego użytku.

Rysowanie sceny:

```
d3ddev->BeginScene();
    ...rysowanie...
d3ddev->EndScene();
```

Metod **BeginScene()** i **EndScene()**, dla zachowania optymalnej wydajności, należy użyć tylko raz dla każdej ramki animacji.

Wyświetlenie ramki na ekranie

Wyświetlenie ramki:

```
d3ddev->Present(NULL, NULL, NULL, NULL);
```

Pierwsze trzy parametry odnoszą się do najwolniejszej, niepolecanej metody buforowania obrazu (nie używane). Czwarty parametr dotyczy możliwości pokazania zawartości bufora w innym oknie Windows (nie używane).



Przykładowy program


Windows




KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



Politechnika Świętokrzyska

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

" Program Rozwojowy Potencjału Dydaktycznego Politechniki Świętokrzyskiej w Kielcach: *kształcenie w nowoczesnych obszarach techniki* "

Program Operacyjny Kapitał Ludzki, Priorytet IV Szkolnictwo wyższe i nauka, Działanie 4.1. Wzmocnienie i rozwój potencjału dydaktycznego uczelni oraz zwiększenie liczby absolwentów kierunków o kluczowym znaczeniu dla gospodarki opartej na wiedzy, Poddziałanie 4.1.1. Wzmocnienie potencjału dydaktycznego uczelni.

Umowa UDA–POKL.04.01.01-00395/09-00