Programowanie Grafiki Komputerowej

Microsoft DirectX cz. 4 - złożone obiekty 3D, struktura mesh i format pliku ".x"

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

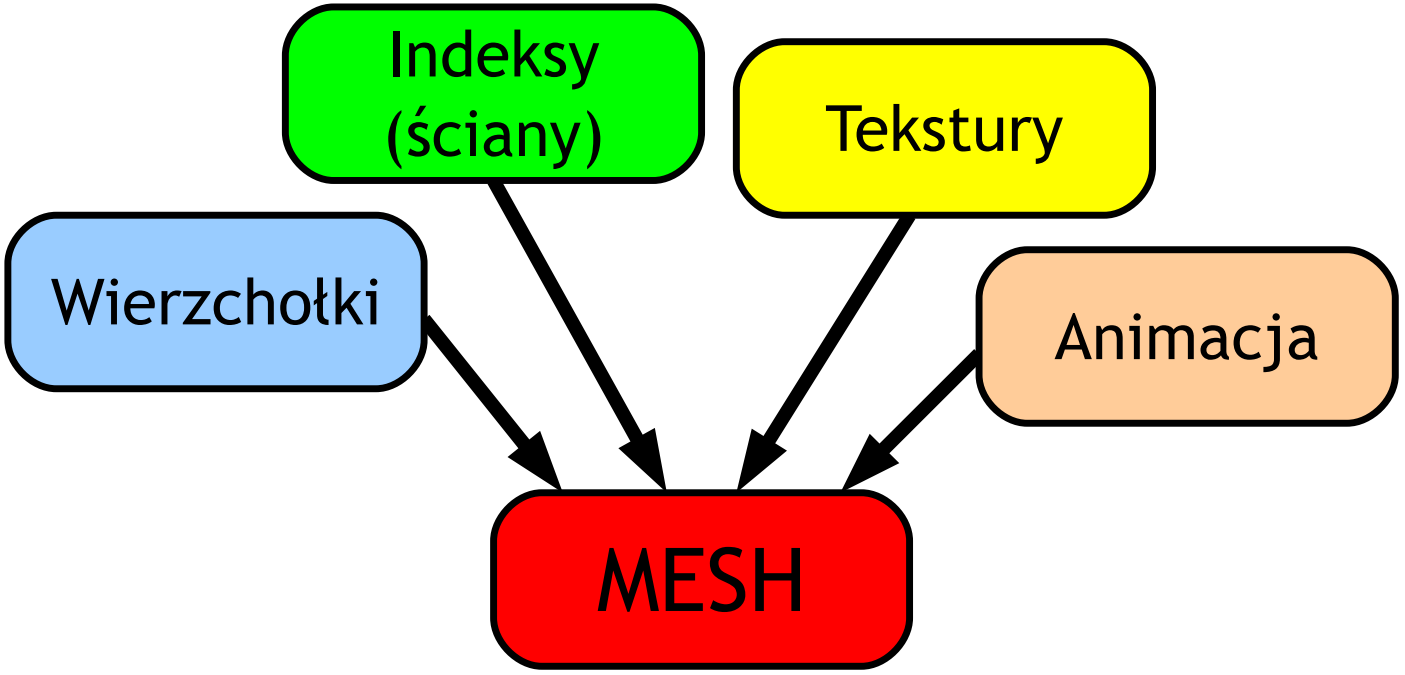
Spis treści

1. Format mesh
   - Obiekty predefiniowane
   - Rendering obiektów typu mesh
   - Przykładowy program (dx-p7)
2. Pliki w formacie ".x"
   - Odczyt plików ".x"
   - Przykładowy program (dx-p8)
3. Obiekty ".x" z teksturami
   - Przykładowy program (dx-p9)
4. 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.

Definicja złożonych obiektów 3D z pomocą przedstawionych metod, tzn. tabeli wierzchołków i indeksów, jest niewygodna. DirectX posiada swój własny format przechowywania obiektów pod nazwą **mesh (pol. siatka)**, oraz odpowiadający mu **format pliku ".x"**.



Obiekty predefiniowane

Istnieją predefiniowane funkcje Direct3D generujące podstawowe figury geometryczne jako mesh, między innymi:

- `D3DXCreateBox()` – prostopadłościan,
- `D3DXCreateSphere()` – sfera,
- `D3DXCreateTorus()` – torus,
- `D3DXCreateTeapot()` – czajnik.

D3DXCreateBox()

```
HRESULT D3DXCreateBox(LPDIRECT3DDEVICE9 pDevice,
            FLOAT Width, FLOAT Height, FLOAT Depth,
            LPD3DXMESH *ppMesh, LPD3DXBUFFER *ppAdjacency);
```

- **pDevice** – urządzenie Direct3D.
- **Width, Height, Depth** – wymiary prostopadłościanu.
- **ppMesh** – wskaźnik na zmienną docelową.
- **ppAdjacency** – wskaźnik na tabelę wartości typu DWORD zawierającą informacje o sąsiedztwie trójkątów (może być NULL).

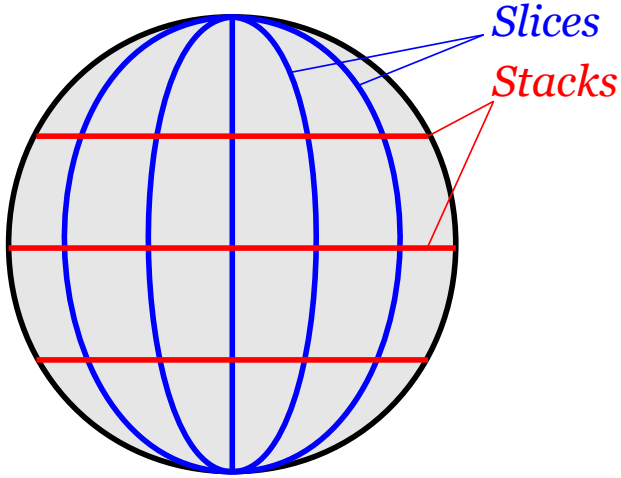
Sześcian:

```
LPD3DXMESH meshBox;
D3DXCreateBox(d3ddev, 5, 5, 5, &meshBox, NULL);
```

D3DXCreateSphere()

```
HRESULT D3DXCreateSphere(LPDIRECT3DDEVICE9 pDevice,
                         FLOAT Radius, UINT Slices, UINT Stacks,
                         LPD3DXMESH *ppMesh, LPD3DXBUFFER *ppAdjacency);
```

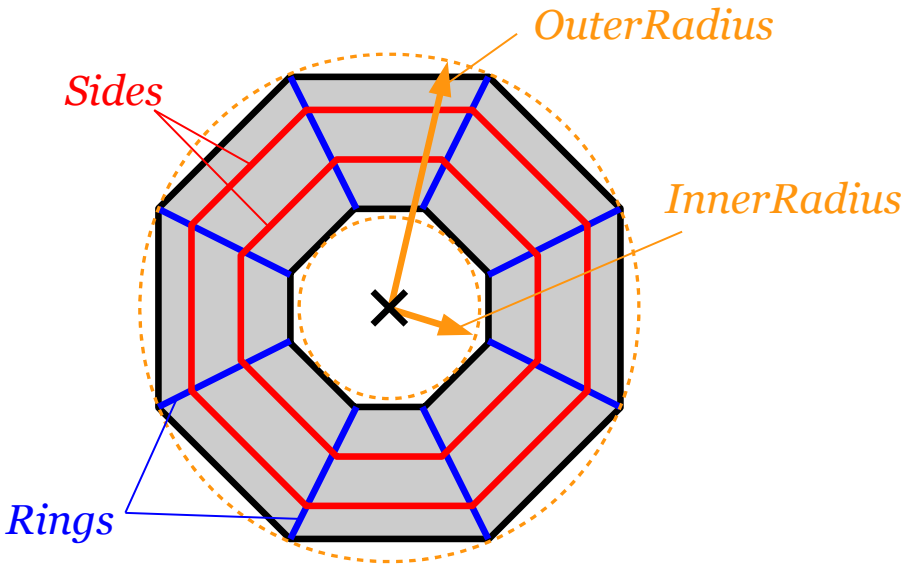
- **Radius** – promień sfery.
- **Slices, Stacks** – liczba "równoleżników" i "południków", odpowiadająca za liczbę wierzchołków i dokładność sfery.



D3DXCreateTorus()

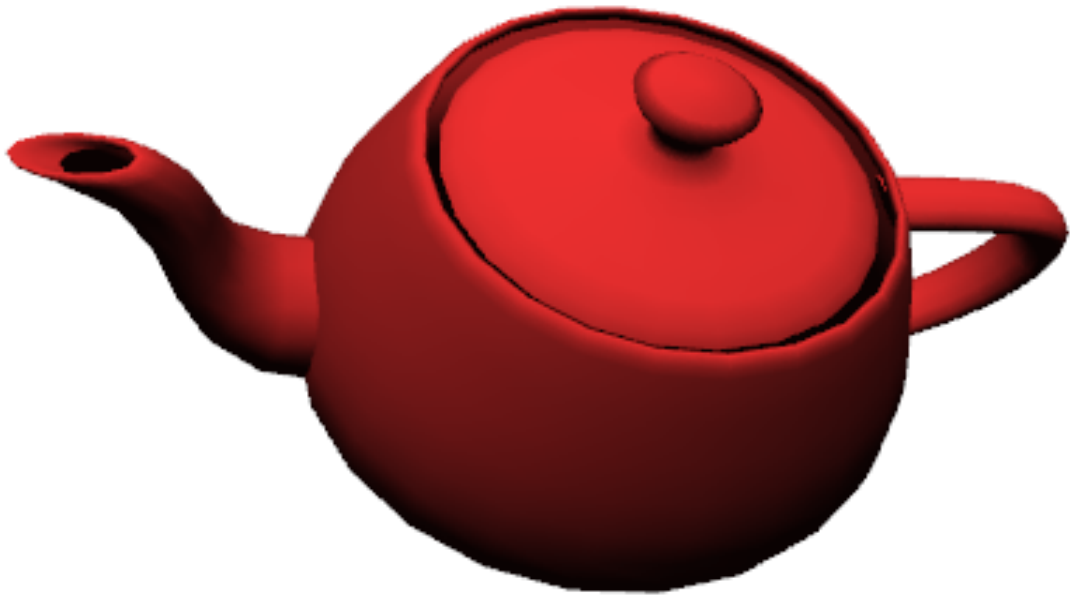
```
HRESULT D3DXCreateTorus(LPDIRECT3DDEVICE9 pDevice,
                        FLOAT InnerRadius, FLOAT OuterRadius,
                        UINT Sides, UINT Rings,
                        LPD3DXMESH *ppMesh, LPD3DXBUFFER *ppAdjacency);
```

- **InnerRadius, OuterRadius** – promień wewnętrzny i zewnętrzny.
- **Sides, Rings** – liczba segmentów i liczba wierzchołków przekroju.



D3DXCreateTeapot()

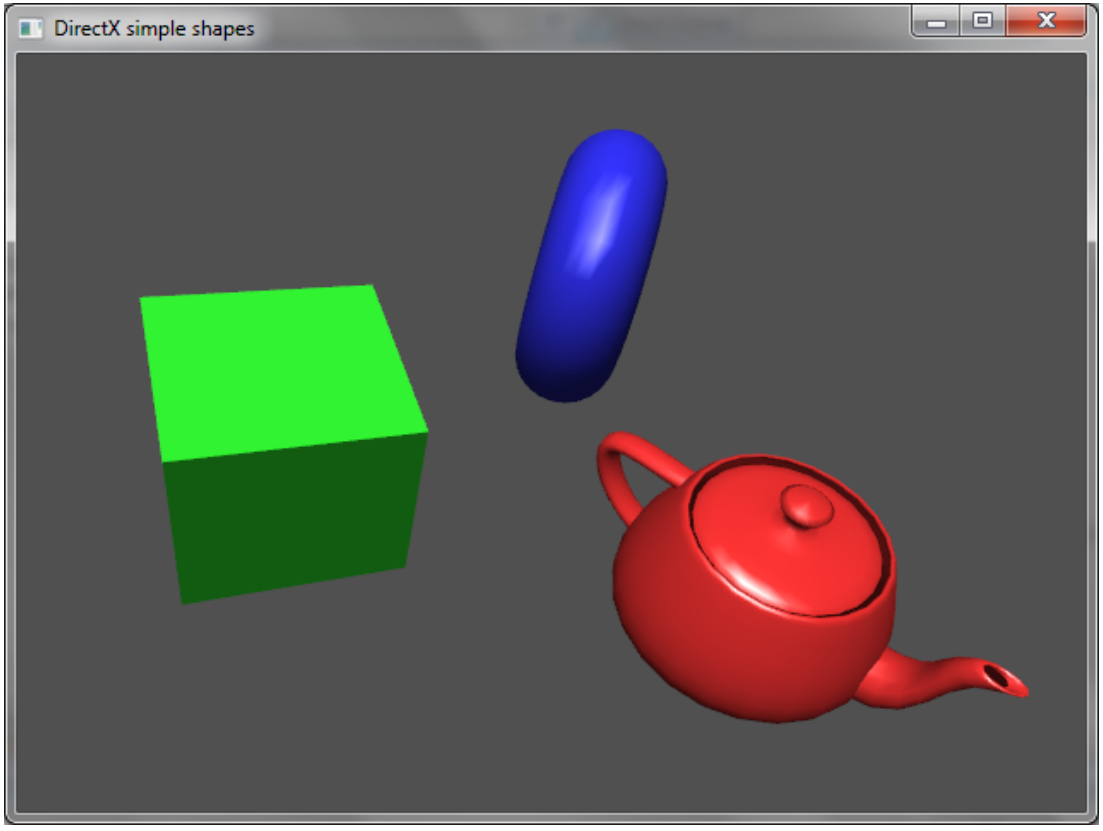
```
HRESULT D3DXCreateTeapot(LPDIRECT3DDEVICE9 pDevice,
                         LPD3DXMESH *ppMesh,
                         LPD3DXBUFFER *ppAdjacency);
```



Obiekty opisane przez *mesh* renderuje się (wyświetla) z pomocą metody **DrawSubset()**. Argumentem jest numer podzbioru, obiekt typu *mesh* może składać się z kilku pozdbiorów (and. *subsets*) wierzchołków i ścian. Dla obiektów jednolitych należy podać argument 0 (zero).

Rendering sześcianu:

```
LPD3DXMESH meshBox;
D3DXCreateBox(d3ddev, 5, 5, 5, &meshBox, NULL);
// ...
meshBox->DrawSubset(0);
```



Przykładowy program


Windows


Niektóre programy do modelowania obiektów 3D pozwalają na zapis obiektów w formacie ".x". Dla DirectX SDK (do wersji 9 włącznie) dołączany jest konwerter popularnego formatu ".3ds" na ".x":

```
conv3ds file.3ds
```

Nowsze wersje DirectX:

Począwszy od wersji 10 DirectX wykorzystywany jest nowy format opisu obiektu **sdkmesh**. Konwersja obiektów możliwa jest z pomocą narzędzia **meshconvert.exe**. Obsługę nowego formatu zapewnia komponent **DXUT**.

Odczyt plików .x

```
HRESULT D3DXLoadMeshFromX(LPCTSTR pFilename, DWORD Options,
      LPDIRECT3DDEVICE9 pD3DDevice,
      LPD3DXBUFFER *ppAdjacency, LPD3DXBUFFER *ppMaterials,
      LPD3DXBUFFER *ppEffectInstances, DWORD *pNumMaterials,
      LPD3DXMESH *ppMesh);
```

- **pFilename** – nazwa pliku ".x".
- **Options** – flaga D3DXMESH_SYSTEMMEM wymusza użycie pamięci systemowej (zalecane).
- **pD3DDevice** – wskaźnik na urządzenie Direct3D.
- **ppAdjacency** – wskaźnik na tabelę zawierającą informacje o sąsiedztwie trójkątów (lub NULL).
- **ppMaterials** – wskaźnik na bufor z opisem materiałów.
- **ppEffectInstances** – wskaźnik związany z efektami (lub NULL).
- **pNumMaterials** – wskaźnik na zmienną DWORD w której zostanie zapisana informacja o liczbie materiałów (i tym samym liczbie podzbiorów obiektu).
- **ppMesh** – wskaźnik na wynikową zmienną przechowującą *mesh*.

Przykład

Wczytywanie obiektu z pliku .x:

```
LPD3DXMESH mesh_obj; // Struktura MESH obiektu
DWORD mesh_nmat; // Liczba materiałów (i podzbiorów)
LPD3DXBUFFER mesh_bufmat; // Bufor materiałów

HRESULT res = D3DXLoadMeshFromX(
    "spaceship 2.x", // Nazwa pliku
    D3DXMESH_SYSTEMMEM,
    d3ddev, // Wskaźnik na urządzenie
    NULL,
    &mesh_bufmat, // Bufor materiałów
    NULL,
    &mesh_nmat, // Liczba używanych materiałów
    &mesh_obj); // Wynikowy MESH
```

Materiały

Aby informacja o materiale mogła być wprost użyta, powinna się znaleźć w zmiennej typu `D3DMATERIAL9`.

Uzyskanie tego typu informacji ze zmiennej typu `LPD3DXBUFFER` składa się z kilku etapów:

1. Użycie metody **GetBufferPointer()** zwracającej wskaźnik na bufor przechowujący informacje o materiałach.
2. Skopiowanie materiałów do zmiennych typu **D3DMATERIAL9**, z pomocą pola **MatD3D**.
3. Uzupełnienie współczynnika odbicia światła otoczenia (**Ambient**) finalnego materiału, ponieważ pliki ".x" nie przechowują takiej informacji.

Materiały

Analiza materiałów z pliku x.:

```
D3DMATERIAL9* mesh_mat; // Materiały obiektu

// Uzyskanie wskaźnika do bufora materiałów:
D3DXMATERIAL* tempMat =
              (D3DXMATERIAL*)mesh_bufmat->GetBufferPointer();

// Utworzenie struktury opisującej każdy materiał:
mesh_mat = new D3DMATERIAL9[mesh_nmat];

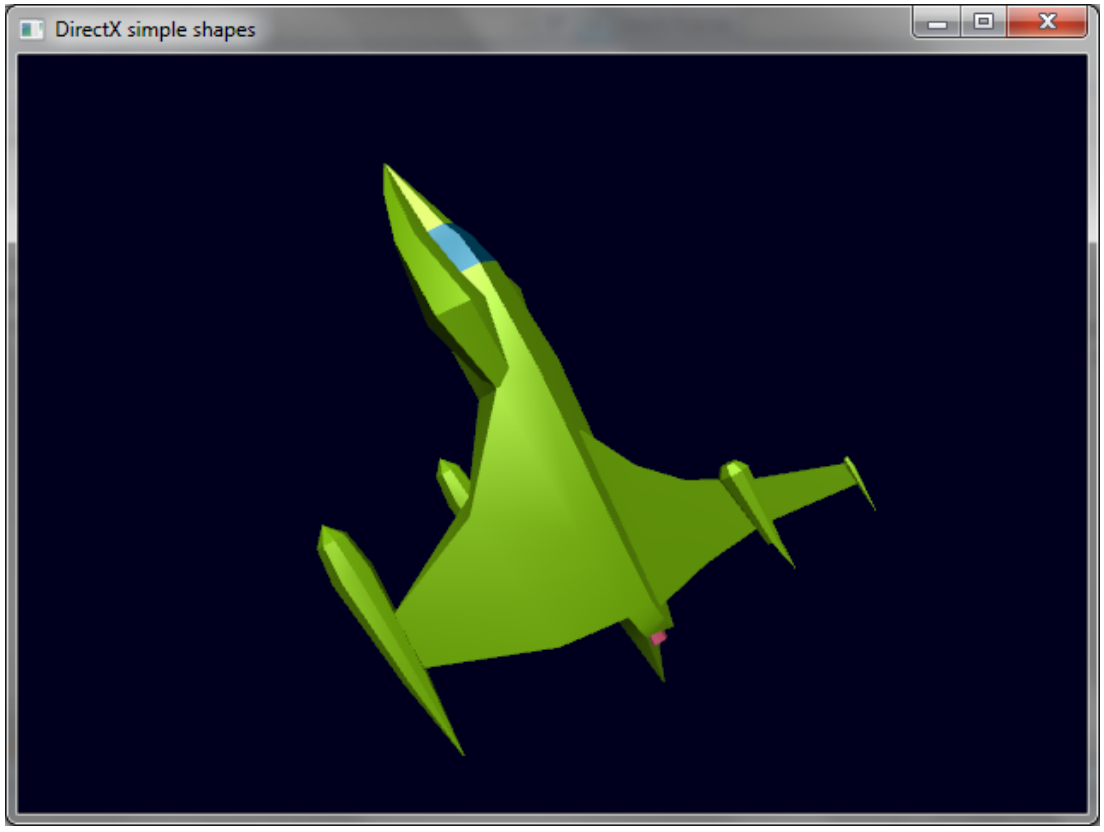
for(int i=0;i < mesh_nmat;i++) {
    // Odczyt informacji o materiale:
    mesh_mat[i] = tempMat[i].MatD3D;
    /// Składowa AMBIENT zastąpiona przez DIFFUSE:
    mesh_mat[i].Ambient = mesh_mat[i].Diffuse;
}
```

Użycie materiałów

Przed narysowanie podzbioru obiektu odpowiedni materiał należy aktywować z pomocą metody **SetMaterial()**.

Rendering obiektu z materiałami:

```
for(int i=0;i < mesh_nmat;i++) {
    d3ddev->SetMaterial(&mesh_mat[i]);
    mesh_obj->DrawSubset(i);
}
```



Przykładowy program


Windows


Załadowanie obiektu z teksturami

Plik w formacie ".x" zawiera nazwy plików tekstur nałożonych na obiekt. W trakcie odczytu pliku należy załadować odpowiednie tekstury:

Odczyt pliku ".x" z teksturami:

```
LPDIRECT3DTEXTURE9* mesh_tex;

// Tekstury:
mesh_tex = new LPDIRECT3DTEXTURE9[mesh_nmat];

for(DWORD i=0;i < mesh_nmat;i++) {
    mesh_mat[i] = tempMaterials[i].MatD3D;
    mesh_mat[i].Ambient = mesh_mat[i].Diffuse;

    // Próba załadowania odpowiedniej tekstury:
    res = D3DXCreateTextureFromFileA(d3ddev,
          tempMaterials[i].pTextureFilename, &mesh_tex[i]);
    // Brak tekstury:
    if(res != D3D_OK) mesh_tex[i] = NULL;
}
```

Rendering obiektu z teksturami

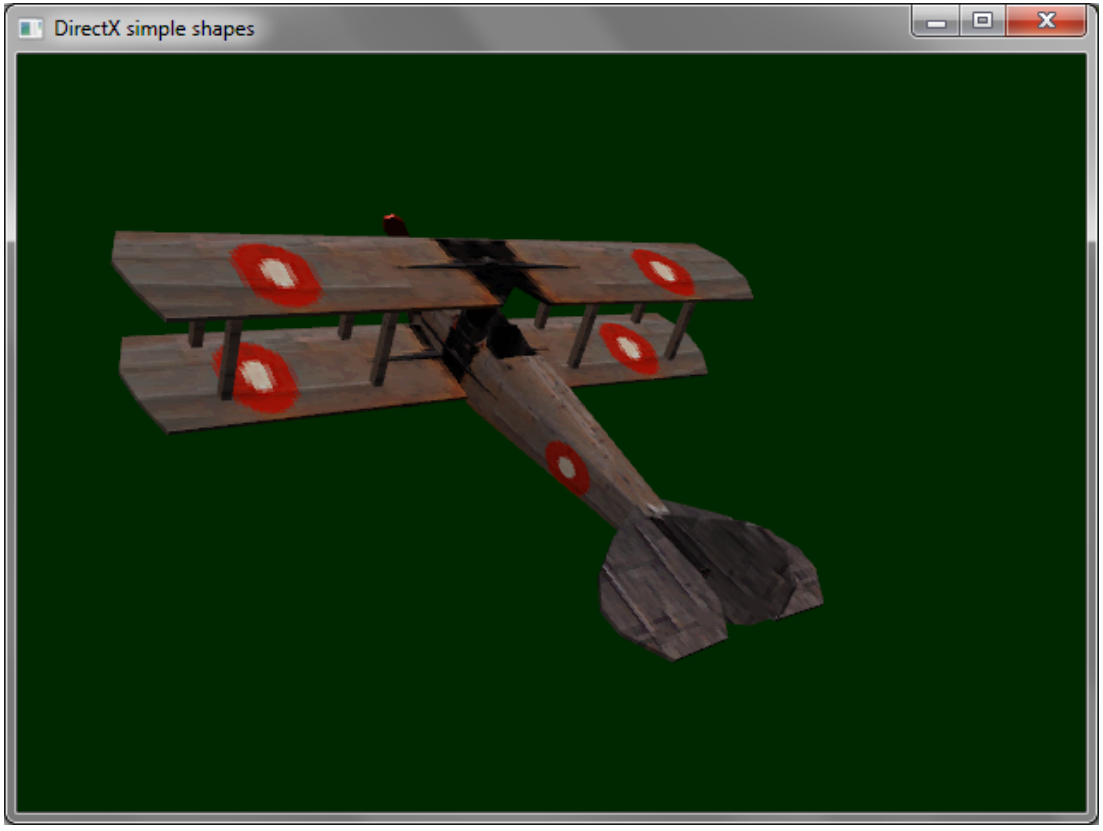
Przed narysowaniem kolejnej części obiektu, należy aktywować odpowiednią teksturę (jeżeli jest):

Rendering obiektu:

```
for(DWORD i=0;i < mesh_nmat;i++) {
    d3ddev->SetMaterial(&mesh_mat[i]);

    // Aktywacja tekstury:
    if(mesh_tex[i]) d3ddev->SetTexture(0, mesh_tex[i]);

    mesh_obj->DrawSubset(i);
}
```



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