GRAFIKA KOMPUTEROWA

Podstawy programowania w środowisku Direct 3D

1. Definiowanie kształtu obiektu

Podstawową figurą geometryczną, z której buduje się obiekty trójwymiarowe w środowisku Direct 3D, jest trójkąt. W programie przykładowym, w funkcji **InitDevice()**, znajduje się tablica **vertices[]**, w której zdefiniowane są poszczególne wierzchołki kolejnych trójkątów. Typ wierzchołka można określać w zależności od potrzeb, w naszym przypadku jest za to odpowiedzialna struktura **SimpleVertex**, zawierająca dwa wektory trójelementowe (**XMFLOAT3**): pierwszy z nich zawiera współrzędne (*x*, *y*, *z*) punktu w przestrzeni, drugi definiuje tzw. *normalną* (wektor uwzględniany przy wyznaczaniu kierunku odbijania światła przez dany wierzchołek). Należy pamiętać o odpowiednim zmodyfikowaniu zmiennej **g_nVertices**, która przechowuje liczbę wierzchołków. Wiele wierzchołków może być wspólnych dla różnych trójkątów, dlatego trójkąty definiuje się podając kolejne indeksy wierzchołków zawartych w tablicy **vertices[]** – indeksy te są umieszczone w tablicy **indices[]**. W tym przypadku należy pamiętać o uaktualnieniu liczby trójkątów **g_nTriangles**.

W funkcji **InitGeometry()** tworzony jest także globalny bufor **g_pVertexBuffer** zawierający dane o wierzchołkach:

```
g_pd3dDevice->CreateBuffer(&bd, &InitData, &g_pVertexBuffer);
```

przy czym zmienna **bd** to tzw. *buffer description*, czyli struktura zawierająca m.in. liczbę wierzchołków, natomiast zmienna **InitData** to struktura zawierająca m.in. wskaźnik na tablicę wierzchołków. Analogicznie tworzony jest także bufor indeksów (**g_pIndexBuffer**).

Obiekt **g_pd3dDevice** reprezentuje urządzenie wyjściowe, jednak nie odpowiada za renderowanie sceny, które wykonywane jest przez tzw. kontekst natychmiastowy (*immediate context*), komunikujący się bezpośrednio ze sterownikiem (zmienna **g_pImmediateContext**).

2. Zmiana położenia elementów sceny

Ustawienie wszystkich elementów sceny względem płaszczyzny ekranu opisuje tzw. ***macierz świata***. W programie przykładowym macierze konstruowane są w funkcji **Render()**, a macierz świata jest reprezentowana przez zmienną **g_World**. Początkowo jest to macierz jednostkowa, utworzona w wyniku zastosowania funkcji **XMMatrixIdentity()**, można ją jednak modyfikować wykonując następującą operację:

```
nowa_macierz_swiata = macierz_transformacji * macierz_swiata
```

Macierz transformacji może określać translację lub obrót. Na przykład, macierz definiującą obrót wokół osi X uzyskuje się przez wywołanie funkcji **XMMatrixRotationX**:

```
XMMATRIX        XMMatrixRotationX (float Angle  /* kąt obrotu (w radianach) */);
```

W programie przykładowym macierz świata stanowi część struktury typu **ConstantBuffer**, która umieszczana jest w specjalnym buforze (**g_pConstantBuffer**), przekazywanym z kolei

do kontekstu natychmiastowego metodą **VSSetConstantBuffers**, dzięki czemu mogą z niej korzystać shadery (zdefiniowane w pliku Shaders.fx).

3. Poruszanie się obserwatora w układzie współrzędnych

Położenie obserwatora (kamery) w układzie współrzędnych określa tzw. ***macierz widoku***. Macierz widoku jest reprezentowana przez zmienną **g_View**. Konstruuje się ją przy użyciu funkcji **D3DXMatrixLookAtRH**:

```
XMMATRIX  XMMatrixLookAtRH (
   XMVECTOR EyePosition,        // wektor określający położenie oka
   XMVECTOR FocusPosition,      // wektor określający obserwowany punkt
   XMVECTOR UpDirection         // definicja kierunku pionowego, najczęściej [0, 1, 0, 0]
);
```

Macierz widoku również trafia do struktury typu **ConstantBuffer**, podobnie jak macierz świata.

4. Działanie programu przykładowego

W programie przykładowym funkcja **InitDevice()**, która tworzy i wypełnia bufor wierzchołków, uruchamiana jest tylko raz, na początku jego działania. Z kolei funkcja **Render()**, odpowiadająca za rysowanie sceny, wywoływana jest w pętli przez cały czas wykonywania programu, chyba że są jakieś komunikaty systemowe, wymagające obsłużenia:

```
while( msg.message! = WM_QUIT )
{
        if ( PeekMessage (&msg, NULL, 0U, 0U, PM_REMOVE) )
        {
                TranslateMessage (&msg);     //obsługa komunikatów systemowych
                DispatchMessage (&msg);
         }
         else
                Render();                    // ← RYSOWANIE SCENY
}
```

Przed każdym narysowaniem sceny funkcja **Render()** wykonuje operacje na macierzach. Umożliwia to animowanie narysowanego obiektu.

Właściwe renderowanie wykonywane jest poprzez uruchomienie metody **DrawIndexed**:

```
g_pImmediateContext->DrawIndexed(g_nTriangles*3, 0, 0 );
```

Przykładowe zadanie

Zadanie przeznaczone do wykonania w trakcie zajęć będzie obejmować następujące zagadnienia:

1. rysowanie trójwymiarowego kształtu,
2. animacja sceny – modyfikowanie macierzy świata,
3. poruszanie się obserwatora – modyfikowanie macierzy widoku.