Część II
Mapowanie wypukłości

Mapowanie wypukłości polega na wyznaczeniu normalnej dla każdego fragmentu (piksela) renderowanej powierzchni na podstawie informacji pobranej z tekstury. Tak wyznaczona normalna jest następnie wykorzystywana do obliczeń oświetlenia w danym punkcie powierzchni.

Przy mapowaniu wypukłości wykorzystywane są wektory określone w **przestrzeni stycznej do powierzchni**. Przestrzeń ta wyznaczona jest przez trzy wersory: styczną, normalną i binormalną. Wersor normalny w danym punkcie powierzchni pojawił się już na przykład w opisie modelu oświetlenia Phonga. Dwa pozostałe wersory leżą w płaszczyźnie stycznej do powierzchni. Są one znormalizowanym gradientem współrzędnych tekstury (gradient liczony jest w płaszczyźnie stycznej do powierzchni). Przy mapowaniu wypukłości dopuszcza się, żeby styczna i binormalna nie były dokładnie prostopadłe. Wersory wyznaczające przestrzeń styczną do powierzchni są liczone w każdym wierzchołku siatki trójkątów (w DirectX istnieje gotowa funkcja, która wykonuje takie obliczenia dla obiektu siatki; Rendermonkey wywołuje tą funkcję automatycznie jeśli tylko w deklaracji strumienia wierzchołków pojawią się pozycje TANGENT lub BINORMAL).

Z reguły mapa wypukłości (mapa normalnych) przechowuje współrzędne nowej, zaburzonej, wartości normalnej, w układzie utworzonym przez styczną, oryginalną normalną i binormalną. Współrzędne wersorów mogą być z zakresu [-1,1], więc żeby zakodować je w teksturze reprezentującej wartości z przedziału [0,1] trzeba zastosować odpowiednie przekształcenie liniowe.

W innej odmianie mapowania wypukłości, którą nie będziemy się zajmować, **mapowaniu wypukłości Blinna**, w mapie wypukłości przechowywane są gradienty **mapy wysokości** w kierunkach wersora stycznego i binormalnego. Gradienty te są dodawane do oryginalnej normalnej, a wynik jest następnie normalizowany. W odmianie, którą będziemy się zajmowali, zakładamy, że tak obliczony wynik, wyrażony w przestrzeni stycznej, jest już zakodowany w mapie wypukłości. Zdarza się również, chociaż rzadko, że mapa wypukłości zawiera normalne wyrażone już w układzie modelu (wtedy ta sama mapa nie może być wykorzystana z innym modelem).

Nadal korzystając z map sześciennych oświetlenia rozproszonego i zwierciadlanego zmienimy sposób liczenia normalnych na powierzchni czajnika. Dodamy do strumienia danych, będących wejściem shadera wierzchołków, nowe elementy: styczną i binormalną. Dodamy dwie nowe tekstury: mapę normalnych i mapę połysku. Będziemy liczyli dla każdego piksela układ współrzędnych przestrzeni stycznej do powierzchni, pobierali z mapy normalnych zaburzoną normalną w tym układzie współrzędnych, oraz przekształcali ją do układu sceny przed wykonaniem tych samych co poprzednio obliczeń oświetlenia. Dodatkowo będziemy mnożyć wynik oświetlenia zwierciadlanego przez odcień szarości pobrany z mapy połysku co sprawi, że pewne fragmenty obiektu nie będą odbijać światła tak dobrze jak inne.

1. Dodaj do projektu teksturę mapy normalnych: prawy przycisk na **Textured Phong, Add Texture, Add 2D Texture, 2D Texture**; prawy przycisk na **my2DTexture, Edit**, z katalogu, w którym zainstalowane jest **RenderMonkey 1.62**, wybierz **Examples\Media\Textures\Rusty Metal\rustymetalBump.tga;** prawy przycisk na **my2DTexture, Rename, bump**
2. Dodaj do projektu teksturę mapy połysku: prawy przycisk na **Textured Phong, Add Texture, Add 2D Texture, 2D Texture**; prawy przycisk na **my2DTexture, Edit**, z katalogu, w którym zainstalowane jest **RenderMonkey 1.62**, wybierz **Examples\Media\Textures\Rusty Metal\rustymetalSpec.tga**; prawy przycisk na **my2DTexture, Rename, gloss**
3. Dodaj w przebiegu Pass 2 samplery dla tekstur bump i gloss, nazywając je tak samo jak tekstury: prawy przycisk na **Pass 2**, **Add Texture Object**; rozwiń **Texture3**, prawy przycisk na **baseMap, Reference Node, bump**; prawy przycisk na **Texture3**, **Rename, bump**; podwójne kliknięcie na **bump, D3DSAMP_MAGFILTER Value** ustaw na **D3DTEXF_LINEAR**, podobnie **D3DSAMP_MINFILTER** i **D3DSAMP_MIPFILTER**; powtórz powyższe kroki dla **gloss**
4. Dodaj w shaderze pikseli w przebiegu Pass 2 zmienne globalne odpowiadające dodanym samplerom: dwuklik na Pixel Shader w gałęzi Pass 2, dodaj linijkę:
   `sampler2D bump, gloss;`
5. Dodaj styczną i binormalną do strumienia wierzchołków: dwuklik na **Stream Mapping** (w gałęzi Textured Phong), kliknij przycisk **Add**, dla nowego wpisu ustaw Usage na **TANGENT**, Index na 0, Data Type na **FLOAT3**; kliknij przycisk **Add**, dla nowego wpisu ustaw Usage na **BINORMAL**, Index na 0, Data Type na **FLOAT3;** kliknij przycisk **OK**
6. Edytuj shader wierzchołków w przebiegu Pass 2: dwuklik na **Vertex Shader** w gałęzi **Pass 2**
   a. Dodaj do struktury `VS_INPUT`:
      ```
      float3 Tangent  : TANGENT0;
      float3 Binormal : BINORMAL0;
      ```
   b. Dodaj do struktury `VS_OUTPUT`:
      ```
      float3 Tangent  : TEXCOORD2;
      ```
   c. Dodaj do `vs_main` linijkę:
      ```
      Output.Tangent = Input.Tangent;
      ```
7. Edytuj shader pikseli w przebiegu Pass 2: dwuklik na **Pixel Shader** w gałęzi **Pass 2**
   a. Dodaj do struktury `PS_INPUT`:
      ```
      float3 Tangent : TEXCOORD2;
      ```
   b. W funkcji `ps_main` usuń linijkę:
      ```
      float3 fvNormal = normalize( Input.Normal );
      ```
   (będziemy liczyć wektor fvNormal w inny sposób)
   c. W funkcji `ps_main` na samym początku pobierz i zdekoduj wartość normalnej z mapy wypukłości bump:
      ```
      float3 fvNormalTS = normalize( tex2D( bump, Input.Texcoord )
         .xyz*2.0-1.0 );
      ```
   d. Zaraz po powyższej instrukcji umieść obliczenia układu współrzędnych przestrzeni stycznej:
      ```
      float3 Normal   = normalize(Input.Normal);
      float3 Tangent  = normalize(Input.Tangent);
      float3 Binormal = cross(Normal, Tangent);
      ```
   e. Następnie przekształć wektor `Normal` z układu współrzędnych przestrzeni stycznej do układu współrzędnych sceny:
      ```
      float3 fvNormal = Tangent * fvNormalTS.x + Binormal *
         fvNormalTS.y + Normal * fvNormalTS.z;
      ```

f. Zmień obliczenia `fvTotalSpecular` tak, żeby wykorzystać odcień szarości pobrany z mapy połysku:

```
float4 fvTotalSpecular   = fvSpecular * tex2D(gloss,
   Input.Texcoord) * texCUBE(specmap, fvViewReflection);
```

8. Skompiluj shadery, żeby zobaczyć gotowy efekt mapowania wypukłości.

Dodatkowe informacje na temat mapowania wypukłości i innych zbliżonych technik.

Mapy wypukłości można przechowywać w teksturach o takim samym formacie jak zwykłe, nieskompresowane tekstury koloru: R8G8B8. Gdy oszczędność pamięci jest ważna, przechowuje się je również w formatach o dwóch składowych: A8L8 (liczby bez znaku) lub U8V8 (liczby ze znakiem), wyliczając trzecią współrzędną już po odczycie z tekstury (cały wektor ma mieć długość jednostkową), oraz w formatach tekstur skompresowanych (stratnie) DXT1 do DXT5, 3Dc (ten ostatni tylko na kartach ATI). Formaty tekstur skompresowanych zapewniają współczynnik kompresji około 4:1, dekompresja odbywa się sprzętowo, w przypadku map normalnych (formaty skompresowanych tekstur były projektowane z myślą o zwykłych danych koloru) warto po odczycie zrenormalizować wektor.

W przypadku map wypukłości inaczej niż w przypadku map koloru, zwykłe sposoby radzenia sobie z aliasingiem takie jak filtrowanie trzyliniowe mipmap nie dają dobrych rezultatów. Michael Toksvig porusza ten temat w artykule
http://http.download.nvidia.com/developer/SDK/Individual_Samples/MEDIA/docPix/docs/Mipmapping_Normal_Maps.pdf

Można wyróżnić cztery warianty mapowania wypukłości różniące się złożonością i dokładnością obliczeń: mapowanie normalnych, mapowanie paralaksy, mapowanie reliefowe, mapowanie reliefowe z cieniami. W mapowaniu paralaksy odczytujemy oprócz wartości normalnej również wartość wysokości względem powierzchni. Modyfikujemy współrzędne tekstury mapy normalnych na podstawie przecięcia wektora do obserwatora z płaszczyzną oddaloną o wartość wysokości od płaszczyzny stycznej do powierzchni. Mapowanie reliefowe wykorzystuje wyszukiwanie liniowe ze stałym krokiem i binarne pierwszego punktu kontaktu promienia z powierzchnią na podstawie mapy wysokości. Dla cienia podobne wyszukiwanie odbywa się wzdłuż promienia światła.

Mapowanie przemieszczeń (Displacement mapping) z kolei to przemieszczenie wierzchołka siatki trójkątów wzdłuż normalnej o wartość odczytaną z mapy przemieszczeń. W HLSL instrukcją pozwalającą próbkować tekstury w programie wierzchołków jest tex2Dlod. Dozwolony format tekstury próbkowanej w programie wierzchołków w DirectX to A32B32G32R32F. Mapowanie przemieszczeń często łączy się z mapowaniem reliefowym (śledzeniem promieni).

Przykłady:

1. Bump Mapping
   a) OpenGL Shading Language 2nd Edition
   b) NVIDIA SDK: Samples: Bump Mapping
   c) NVIDIA SDK: Effects: chain
2. Normalizacja wektorów
   a) NVIDIA SDK: Samples: Normalization Heuristics
3. Kompresja map normalnych
   a) http://www.gamasutra.com/features/20051228/sherrod_01.shtml
4. Parallax/Relief Mapping
   a) ATI SDK: Dynamic Parallax Occlusion Mapping with Approximate Soft Shadows, Natalya Tatarchuk
   b) ATI SDK: Samples: ParallaxMapping
   c) ATI SDK: Parallax Occlusion Mapping: Self-Shadowing, Perspective-Correct Bump Mapping Using Reverse Height Map Tracing, ShaderX3, Zoe Brawley, Natalya Tatarchuk
   d) NVIDIA SDK: Effects: relief_maps
   e) DirectX SDK (October 2006): Samples: ParallaxOcclusionMapping
5. Displacement Mapping
   a) NVIDIA SDK: Effects: displace_maps
   b) DirectX SDK (October 2006): Samples: DisplacementMapping10

NVIDIA SDK 9.51 jest dostępne pod adresem www:
http://developer.nvidia.com/object/sdk-9.html
NVIDIA SDK 10.0 jest dostępne pod adresem www:
http://developer.nvidia.com/object/sdk_home.html
ATI SDK jest dostępne pod adresem www:
http://ati.amd.com/developer/radeonSDK.html
DirectX SDK jest dostępne pod adresem www:
http://msdn.microsoft.com/directx/sdk/