Część I
Oświetlenie na podstawie map środowiska

Wiadomo jak, korzystając z modelu oświetlenia Phonga, policzyć kolor powierzchni dla danej listy pojedynczych źródeł światła. Przyjmijmy teraz, że zamiast listy pojedynczych źródeł światła dane jest natężenie i kolor światła dochodzącego do obiektów w scenie z dowolnego kierunku. Aby policzyć dla takiego ogólnego przypadku oświetlenie rozproszone w punkcie P o normalnej N i kolorze K_d K musielibyśmy policzyć całkę:

(1) $$\int_{l \in S(0,1)} K_d * L(l) * \max(< l, n >, 0) dS$$

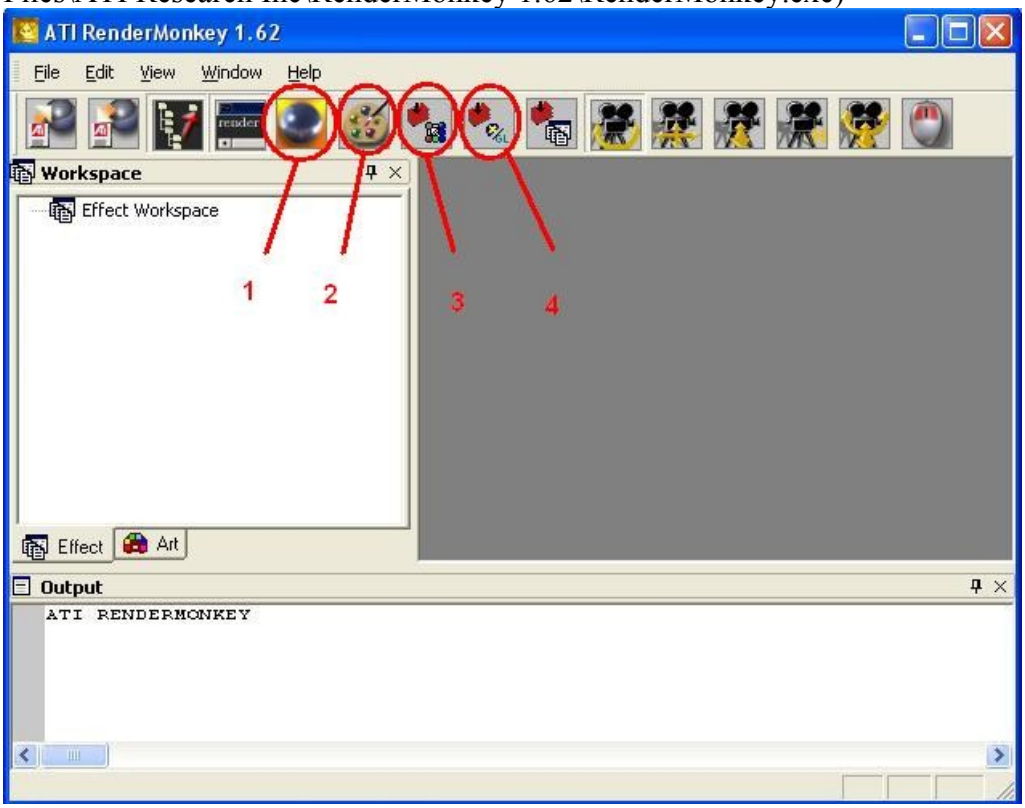
, gdzie $S(0,1)$ to sfera jednostkowa w punkcie 0, l jest dowolnym kierunkiem (punktem należącym do S), $L(l)$ to kolor światła. Załóżmy, że wartości $L(l)$ są dane w postaci mapy środowiska, w szczególności mapy sześciennej. W powyższym wzorze, podobnie jak przy mapowaniu środowiska, pominięty został wpływ pozycji wierzchołka P, a więc przyjęto, że w przybliżeniu dla każdego punktu cieniowanego obiektu kolor światła dla każdego kierunku jest taki sam. Teraz widać, że powyższy wzór jest tylko funkcją normalnej n, a zatem można w preprocessingu przygotować odpowiednią mapę sześcienną, której teksele będą przechowywać policzoną wartość całki dla kierunków z nimi związanych. Dla oświetlenia zwierciadlanego i ustalonej wartości wykładnika rozbłysku w odpowiednia całka również będzie z kolei funkcją wektora wzroku odbitego od powierzchni r i również będzie ją można policzyć w preprocessingu i zapamiętać wynik w postaci mapy sześciennej:

(2) $$\int_{l \in S(0,1)} K_s * L(l) * \max(< l, r >, 0)^w dS$$

Badaniem oświetlenia na podstawie fotografii odbić scenerii w kuli zajmował się między innymi Paul Debevec. Na stronie http://www.debevec.org/probes/ znaleźć można gotowe próbki oświetlenia zapisane w formatach o wysokim zakresie dynamiki intensywności (tzn. intensywność oświetlenia dla każdej składowej koloru jest reprezentowana za pomocą liczby zmiennoprzecinkowej lub stałoprzecinkowej o dużej liczbie bitów i można reprezentować dwie wartości intensywności, których stosunek może być znacznie większy niż 255:1, jaki można maksymalnie uzyskać przy standardowych 8 bitach na składową koloru). Na stronie jest również do pobrania aplikacja HDR Shop 1.0 pozwalająca konwertować sfotografowane odbicia w kuli do mapy sześciennej, a następnie liczyć mapy sześcienne reprezentujące całki (1) oraz (2). Ponieważ jednak proces liczenia tych całek zajmuje dużo czasu, to wykorzystamy gotowe mapy sześcienne. Trzy mapy sześcienne znajdują się w plikach: CubeMap.dds, DiffuseCubeMap.dds oraz Specular50CubeMap.dds i reprezentują odpowiednio oryginalną mapę środowiska, mapę sześcienną całki (1) oraz mapę sześcienną całki (2) dla wykładnika rozbłysku 50. Można zauważyć, że mapy reprezentujące (1) oraz (2) są mocno rozmyte – dlatego mogą być wykonane w dość niskiej rozdzielczości.

Takie mapy sześcienne można efektywnie kompresować przez przekształcenie do bazy funkcji harmonicznych (takich, których laplasjan $\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = 0$) określonych na sferze. Takie przekształcenie przebiega dość podobnie do zamiany funkcji rzeczywistej zmiennej rzeczywistej w szereg Fouriera, kompresja (stratna) polega natomiast na odrzuceniu wszystkich wyrazów od pewnego miejsca czyli takich, dla których funkcje bazowe odpowiadają wysokim częstotliwościom. W praktyce używa się często reprezentacji map oświetlenia rozproszonego w postaci 9 współczynników przy 9 pierwszych funkcjach bazowych sferycznych harmonicznych, które odpowiadają najmniejszym częstotliwościom, dla każdej składowej koloru (w sumie 27 liczb rzeczywistych).

1. Uruchom ATI RenderMonkey (standardowo instaluje się w katalogu C:\Program Files\ATI Research Inc\RenderMonkey 1.62\RenderMonkey.exe)



Najbardziej interesujące ikony: 1 – otwarcie okna podglądu (bez niego nie można kompilować shaderów); 2 – otwarcie edytora wybranych parametrów shaderów (edytor artysty; pozwala przy pomocy graficznego interfejsu zmieniać wybrane przez programistę parametry shaderów); 3 – kompilacja aktywnego shadera; 4 – kompilacja aktywnego efektu (wszystkich shaderów we wszystkich przebiegach renderowania dla tego efektu)

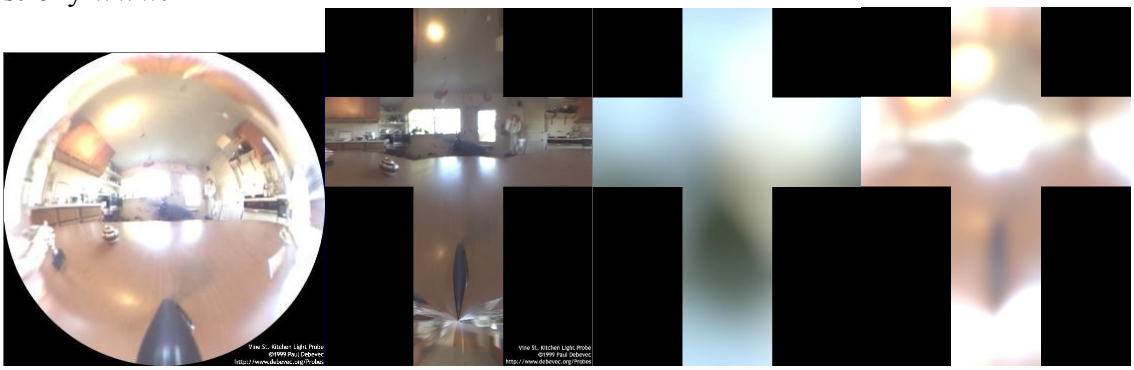
2. Utwórz nowy efekt na bazie efektu Textured Phong: kliknij prawym przyciskiem myszki na **Effect Workspace**, wybierz **Add Default Effect\DirectX\Textured Phong**

Utworzymy nowy przebieg renderowania, w którym wyrenderujemy sześcian otaczający scenę, oteksturowany od wewnątrz mapą sześcienną, która na późniejszym etapie posłuży za mapę środowiska. Skorzystamy z siatki ***Cube.3ds***, która zawiera sześcian o przeciwległych rogach (-1,-1,-1) i (1,1,1), ale powiększymy go 400-krotnie. Wykorzystamy oryginalne pozycje jego wierzchołków jako współrzędne tekstury mapy sześciennej ***CubeMap.dds***.

3. Utwórz nowy przebieg renderowania: prawy przycisk myszki na **Textured Phong**, **Add Pass**
4. Dodaj nowy model o nazwie Cube wczytany z pliku Cube.3ds: prawy przycisk myszki na **Textured Phong**, **Add Model**, **Model**; prawy przycisk myszki na **myModel1**, **Edit**, wskaż plik **Cube.3ds** (jest dołączony do materiałów do zajęć umieszczonych na stronie www); prawy przycisk myszki na **myModel1**, **Rename**, **Cube**
5. Ustaw model wykorzystywany w nowo utworzonym przebiegu na Cube: rozwiń gałąź **Pass 1**, prawy przycisk myszki na **Model**, **Reference Node**, **Cube**

6. Edytuj shader wierzchołków dla tego przebiegu:
    a. podwójne kliknięcie na **Vertex Shader** w gałęzi **Pass 1**
    b. dodaj do struktury `VS_OUTPUT` współrzędne tekstury dla wierzchołka
    ```
    float3 TexCoord : TEXCOORD0;
    ```
    c. dodaj w funkcji `vs_main` liczenie współrzędnych tekstury
    ```
    Output.TexCoord = Input.Position.xyz;
    ```
    d. przeskaluj pozycję (tylko składowe xyz, bez składowej w) przed przekształceniem do układu kamery i rzutowaniem perspektywicznym:
    ```
    Input.Position.xyz *= 400.0;
    ```
    po:
    ```
    Output.TexCoord = Input.Position.xyz;
    ```
    przed:
    ```
    Output.Position = mul( Input.Position,
      matViewProjection );
    ```
7. Dodaj nową mapę sześcienną o nazwie envmap wczytaną z pliku CubeMap.dds: prawy przycisk myszki na **Textured Phong**, **Add Texture**, **Add Cubemap**, **Cubemap**; prawy przycisk myszki na **myCubemap**, **Edit**, wskaż plik **CubeMap.dds** (jest dołączony do materiałów do zajęć umieszczonych na stronie www); prawy przycisk myszki na **myCubemap**, **Rename**, **envmap;** prawy przycisk myszy na **envmap, Origin, Top Left Origin**
8. Dodaj sampler (zwany w RenderMonkey Texture Object) powiązany z teksturą envmap w przebiegu Pass 1: prawy przycisk myszki na **Pass 1**, **Add Texture Object**; rozwiń **Texture0**, prawy przycisk myszki na **baseMap**, **Reference Node**, **envmap**; prawy przycisk myszki na **Texture0**, **Rename**, **envmap;** podwójne kliknięcie na **envmap, D3DSAMP_MAGFILTER Value** ustaw na **D3DTEXF_LINEAR**, podobnie **D3DSAMP_MINFILTER** i **D3DSAMP_MIPFILTER**
9. Edytuj shader pikseli (fragmentów) dla przebiegu Pass 1:
    a. podwójne kliknięcie na **Pixel Shader** w gałęzi **Pass 1**
    b. zadeklaruj zmienną globalną powiązaną z samplerem, który właśnie utworzyliśmy:
    ```
    samplerCUBE envmap;
    ```
    c. dodaj do parametrów funkcji `ps_main` współrzędną tekstury, którą wcześniej dodaliśmy do wyjścia shadera wierzchołków:
    ```
    float4 ps_main(float3 TexCoord : TEXCOORD0) : COLOR0
    ```
    d. w funkcji `ps_main` pobierz na kolor z mapy sześciennej i zwróć jako wynik:
    ```
    float4 envcol = texCUBE(envmap, TexCoord);
    return envcol;
    ```
10. Dodaj zmianę ustawień stanu renderowania w przebiegu Pass 1 i wyłącz odrzucanie trójkątów zwróconych tyłem: prawy przycisk myszy na **Pass 1**, **Add Render State Block**; podwójne kliknięcie na **Render State**, w wierszu **D3DRS_CULLMODE**, prawy przycisk myszy w kolumnie **Value**, **D3DCULL_NONE**
11. Zmień model o nazwie Model na czajnik załadowany z pliku Teapot.3ds: prawy przycisk myszki na **Model**, **Edit**, **Teapot.3ds**
12. Skompiluj wszystkie shadery: kliknij na ikonie **Compile Active Effect** (pasek ikon poniżej menu programu)

Utworzymy teraz nowy przebieg renderowania, w którym będziemy renderować czajnik z oświetleniem Phonga, ale zamiast pojedynczego światła punktowego wykorzystamy przygotowane wcześniej programem HDR Shop mapy sześcienne przechowujące wyniki obliczeń oświetlenia rozproszonego i zwierciadlanego przy założeniu, że światło dociera do sceny ze wszystkich kierunków. Mapa sześcienna środowiska, którą wykorzystaliśmy powyżej do teksturowania sześcianu, oraz mapy sześcienne przechowujące odpowiednie całki, zostały utworzone na podstawie pojedynczego zdjęcia odbicia świata w chromowej kuli. Źródłowe zdjęcie i program HDR Shop są autorstwa Paula Debeveca i pochodzą z jego strony www.



Wszystkie te dane zapisane są w formacie o wysokiej dynamice jasności, co oznacza, że dla każdego kanału koloru stosunek najjaśniejszego do najciemniejszego reprezentowanego punktu może być znacznie większy niż 255:1 jaki można uzyskać na monitorze. Wartości odczytane z map o wysokiej dynamice mogą wykraczać poza przedział [0,1], do którego należą wartości koloru odczytywane w shaderach ze zwykłych tekstur. Przed zapisem na ekran obliczona wartość koloru jest obcinana do przedziału [0,1], ale na etapie obliczeń w shaderach dysponujemy znacznie większą precyzją. Główne zalety są takie, że można uzyskiwać jasne odbicia nawet w bardzo ciemnych materiałach, a także dynamicznie regulować naświetlenie obrazu.

13. Skopiuj przebieg Pass 0 i umieść go po przebiegu Pass 1, zdezaktywuj przebieg Pass 0: prawy przycisk myszki na **Pass 0, Copy**; prawy przycisk myszki na **Pass 1, Paste**; prawy przycisk myszki na **Pass 0, Enable / Disable Pass**
14. Dodaj do projektu mapę sześcienną oświetlenia rozproszonego o nazwie diffmap z pliku DiffuseCubeMap.dds oraz mapę sześcienną oświetlenia zwierciadlanego o nazwie specmap z pliku Specular50CubeMap.dds: prawy przycisk na **Textured Phong**, **Add Texture, Add Cubemap, Cubemap**; prawy przycisk na **myCubemap**, **Edit**, wskaż plik **DiffuseCubeMap.dds** (jest dołączony do materiałów do zajęć umieszczonych na stronie www); prawy przycisk na **myCubemap, Origin, Top Left Origin**; prawy przycisk na **myCubemap, Rename, diffmap**; powtórz powyższe kroki dla **specmap**
15. Dodaj w przebiegu Pass 2 samplery dla tekstur diffmap i specmap, nazywając je tak samo jak tekstury: prawy przycisk na **Pass 2**, **Add Texture Object**; rozwiń **Texture1**, prawy przycisk na **baseMap, Reference Node, diffmap**; prawy przycisk na **Texture1**, **Rename, diffmap**; podwójne kliknięcie na **diffmap, D3DSAMP_MAGFILTER Value** ustaw na **D3DTEXF_LINEAR**, podobnie **D3DSAMP_MINFILTER** i **D3DSAMP_MIPFILTER**; powtórz powyższe kroki dla **specmap**
16. Edytuj shader pikseli (fragmentów) dla Pass 2: podwójne kliknięcie na **Pixel Shader** w gałęzi **Pass 2**
    a. Usuń zmienną globalną `float fSpecularPower;` (wykładnik rozbłysku jest teraz zakodowany w samej mapie sześciennej, więc nie można zmieniać go dynamicznie)
    b. Usuń następujące linijki, w których pojawia się kierunek światła (teraz wykonujemy obliczenia dla wszystkich kierunków światła jednocześnie przy pomocy przygotowanych uprzednio map sześciennych):

```
float3 fvLightDirection = normalize( Input.LightDirection );
float  fNDotL           = dot( fvNormal, fvLightDirection );
float3 fvReflection     = normalize( ( ( 2.0f * fvNormal ) *
   ( fNDotL ) ) - fvLightDirection );
float  fRDotV           = max( 0.0f, dot( fvReflection,
   fvViewDirection ) );
```

    c. Dodaj obliczanie wektora wzroku po odbiciu:

```
float3 fvViewReflection = -reflect(fvViewDirection, fvNormal);
```

    (powyższa instrukcja jest równoważna dłuższemu wywołaniu:

```
float3 fvViewReflection = ( ( 2.0f * fvNormal ) * ( dot( fvNormal,
   fvViewDirection ) ) ) - fvViewDirection ;)
```

    zaraz po

```
float3 fvNormal         = normalize( Input.Normal );
float3 fvViewDirection  = normalize( Input.ViewDirection );
```

    d. Zadeklaruj zmienne globalne powiązaną z samplerami diffmap i envmap:

```
samplerCUBE diffmap, specmap;
```

    e. Zmień odpowiednie linijki dotyczące obliczeń oświetlenia rozproszonego i zwierciadlanego tak, żeby próbkowały odpowiednie mapy sześcienne:

```
float4 fvTotalDiffuse   = fvDiffuse * texCUBE(diffmap, fvNormal) *
   fvBaseColor;
   float4 fvTotalSpecular  = fvSpecular * texCUBE(specmap,
   fvViewReflection);
```

    f. Usuń ze struktury `PS_INPUT` wektor do światła (`float3 LightDirection: TEXCOORD2;`), który nie jest już nigdzie wykorzystywany.

17. Dodaj do projektu macierz odwrotną do macierzy widoku, która przyda się do poprawienia shadera wierzchołków: prawy przycisk na **Textured Phong, Add Variable, Matrix, Predefined, matViewInverse**
18. W tej chwili shader wierzchołków skopiowany ze standardowego efektu Textured Phong, liczy wektor wzroku i normalną w układzie kamery. Przy korzystaniu z map sześciennych chcemy, żeby wektory były w układzie sceny. Wykorzystamy macierz matViewInverse, żeby zmodyfikować te obliczenia w shaderze wierzchołków: podwójne klinkięcie na **Vertex Shader** w gałęzi **Pass 2**
    a. Dodaj zmienną globalną powiązaną z macierzą odwrotną do macierzy widoku:
    ```
    float4x4 matViewInverse;
    ```
    b. Zamień następujące linijki:
    ```
    float3 fvObjectPosition = mul( Input.Position, matView );
    Output.ViewDirection    = fvEyePosition - fvObjectPosition;
    Output.LightDirection   = fvLightPosition - fvObjectPosition;
    Output.Normal           = mul( Input.Normal, matView );
    ```
    na:
    ```
    float3 fvObjectPosition = Input.Position;
    Output.ViewDirection    = mul( float4(0,0,0,1), matViewInverse ) -
      fvObjectPosition;
    Output.Normal           = Input.Normal;
    ```
    c. Usuń ze struktury `VS_OUTPUT`:
    ```
    float3 LightDirection :  TEXCOORD2;
    ```
    d. Usuń niewykorzystywane już zmienne globalne:
    ```
    float3 fvLightPosition;
    float3 fvEyePosition;
    float4x4 matView;
    ```
19. Skompiluj wszystkie shadery: kliknij na ikonie **Compile Active Effect** (pasek ikon poniżej menu programu)
20. W tej chwili czajnik otrzymuje stanowczo za dużo światła. Za chwilę dodamy parametr, którym będziemy mogli regulować naświetlenie, ale wcześniej zmieńmy materiał czajnika: kliknij ikonę **Artist Editor** (pasek ikon poniżej menu programu), zmień **fvDiffuse, fvSpecular, fvAmbient** odpowiednio na (79,79,79), (34,34,34), (59,59,59); zamknij **Artist Editor**
21. Zmień teksturę czajnika na teksturę zardzewiałego metalu: prawy przycisk myszy na **base, Edit**, z katalogu, w którym zainstalowane jest **RenderMonkey 1.62**, wybierz **Examples\Media\Textures\Rusty Metal\rustymetal.tga**
22. W shaderze wierzchołków w Pass 2, zmień linijkę:
    ```
    Output.Texcoord         = Input.Texcoord;
    ```
    na
    ```
    Output.Texcoord         = Input.Texcoord * 0.4;
    ```

Dodamy teraz parametr, którym będziemy mogli regulować naświetlenie renderowanego obrazu tzn. wynikowy kolor w shaderach pikseli w przebiegach Pass 1 i Pass 2 będzie mnożony przez wartość tego parametru. Zobaczymy, że dla wartości parametru między 0 a 1 nadal będziemy otrzymywać białe rozbłyski na powierzchni czajnika, ponieważ mapy sześcienne, które są próbkowane zapisane są w formatach zmiennoprzecinkowych i mogą zwracać liczby dużo większe od 1.

23. Dodaj parametr exposure do projektu: prawy przycisk na **Textured Phong, Add Variable, Float, Float**; prawy przycisk na **f1MyFloat, Rename, exposure**; prawy przycisk na **exposure, Artist Variable**; dwukrotne kliknięcie na **exposure, 1.00, OK**.
24. W shaderach pikseli w Pass 1 oraz Pass 2 dodaj zmienną globalną:

```
float exposure;
```

Zamień w **Pass 1** `return envcol;` na return `exposure * envcol;`

Zamień w **Pass 2**:

```
   return( saturate( fvTotalAmbient + fvTotalDiffuse + fvTotalSpecular )
);
```

na

```
   return( exposure * ( fvTotalAmbient + fvTotalDiffuse +
fvTotalSpecular ) );
```

Pozostaje skompilować shadery (ikona Compile Active Effect) aby zobaczyć gotowy efekt oświetlenia czajnika przy pomocy modelu Phonga i kierunkowych źródeł światła odpowiadających wszystkim tekselom mapy środowiska. Po kliknięciu ikony Artist Editor możemy interaktywnie zmieniać wartość parametru exposure i obserwować efekty w oknie podglądu.

exposure = 0.34 exposure = 1.4