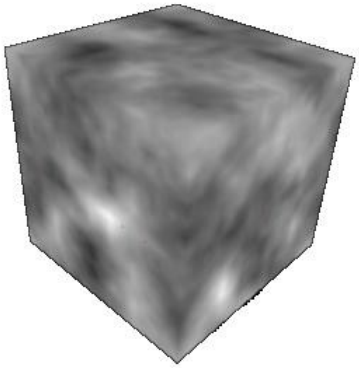
Animacja wody z użyciem szumu Perlina



Użyjemy trójwymiarowej tekstury szumu Perlina do symulacji zaburzeń powierzchni wody. Tekstura jest wygenerowana w taki sposób, że zachodzi zgodność wartości na przeciwległych ścianach. Jedna z trzech osi współrzędnych tekstury będzie reprezentować czas symulacji. Dwie pozostałe współrzędne będą reprezentowały punkt w obrębie kwadratu reprezentującego powierzchnię wody w scenie. Po zaburzeniu normalnej będziemy liczyć odbity i załamany promień od obserwatora. Dla tych promieni oraz współrzędnych w układzie sceny przeciwobrazu renderowanego piksela liczone będzie przecięcie biegu obu promieni z sześcianem otaczającym scenę. To przecięcie posłuży za współrzędne tekstury przy pobieranie koloru z mapy sześciennej. Oba kolory (dla promienia odbitego i załamanego) zostaną zmieszane w proporcji ustalonej na podstawie przybliżenia współczynnika Fresnela. To przybliżenie będzie funkcją sinusa kąta pomiędzy wektorem wzroku a powierzchnią.

1. Dodaj nowy parametr – poziom wody, gdzie -1 będzie odpowiadać dolnej płaszczyźnie sześcianu otaczającego scenę, a 1 górnej.

prawy przycisk na Textured Phong, Add Variable, Float, Float, prawy przycisk na f1MyFloat, Rename, waterlevel

2. Wstaw nowy przebieg renderowania, odpowiedzialny za powierzchnię wody, pomiędzy Pass Spring, a Pass Downsample.

prawy przycisk na Textured Phong, Add Pass, prawy przycisk na Pass 7, Rename, Pass Water, przeciągnij Pass Water na Pass Spring (aby umieścić ten przebieg zaraz po Pass Spring)

3. Nowy przebieg tak samo jak wszystkie poprzednie przebiegi renderujące obiekty sceny powinien zapisywać dane do tekstury screen, ale bez czyszczenia jej dotychczasowej zawartości.

prawy przycisk na Pass Water, Add Render Target, screen,

dwuklik na screen w gałęzi Pass Water, odznacz Enable color clear oraz Enable depth clear, OK

4. Jako bazowy model dla powierzchni wody wykorzystamy tą samą siatkę, co dla sprężyny – kwadratu podzielonego na siatkę małych trójkątów. Siatka podziałów kwadratu przyda się do tego, aby później animować w shaderze wierzchołków wysokości punktów siatki.

prawy przycisk na Model w gałęzi Pass Water, Reference Node, Plane

5. W przebiegu renderowania wody będziemy korzystać z dwóch tekstur: trójwymiarowej tekstury szumu Perlina, oraz sześciennej tekstury środowiska. Tą pierwszą trzeba dopiero dodać do projektu.

prawy przycisk na Textured Phong, Add Texture, Add 3D Texture, 3D Texture, prawy przycisk na my3DTexture, Rename, perlin, prawy przycisk na perlin, Edit, z katalogu RenderMonkey 1.62/Examples/Media/Textures/ wybierz plik NoiseVolume.dds

6. Dodajmy obiekty obu wspomnianych tekstur do przebiegu Pass Water.

prawy przycisk na Pass Water, Add Texture Object, prawy przycisk na Texture0, Rename, envmap, rozwiń envmap, prawy przycisk na baseMap, Reference Node, envmap,

prawy przycisk na Pass Water, Add Texture Object, prawy przycisk na Texture1, Rename, perlin, rozwiń perlin, prawy przycisk na baseMap, Reference Node, perlin

7. Przejdź do edycji shadera wierzchołków w przebiegu Pass Water. Sześcian otaczający scenę, z nałożonym obrazem kuchni ma współrzędne przeciwległych rogów (-400,-400,-400) i (400,400,400). Kwadrat w siatce tesselated_plane.3ds ma współrzędne przeciwległych rogów (-1,0,0) i (0,1,0). Chcemy ten kwadrat umieścić tak, by był przekrojem sześcianu na wysokości y = 400 * waterlevel.

w Vertex Shaderze przebiegu Pass Water:

a) Dodaj zmienną globalną `float waterlevel;`
b) Przed linijką `Output.Position = ...` dodaj obliczenia współrzędnych wierzchołków w układzie świata:

```
Input.Position.xz = Input.Position.xy*2.0+float2(1.0,-1.0);
Input.Position.y  = waterlevel;
Input.Position.xyz *= 400.0;
```

8. W shaderze pikseli będziemy liczyć promień odbity i załamany biegnący od obserwatora, a także normalną na podstawie szumu Perlina. Trzeba do shadera pikseli dostarczyć z shadera wierzchołków jeszcze następujących danych: współrzędnych tekstury dla szumu Perlina, wektora do obserwatora, pozycji wierzchołka w układzie sceny. Z tego ostatniego wektora, po interpolacji na powierzchni każdego trójkąta, w shaderze pikseli otrzymamy współrzędne punktu w scenie, z którego należy wystrzelić odbity i załamany promień. Dla wygody przeskalujemy pozycję wierzchołka tak, aby wnętrze sześcianu otaczającego scenę reprezentowało obszar [-1,1]x[-1,1]x[-1,1]. Współrzędne *xy* tekstury dla szumu Perlina powstaną przez pomnożenie przez stałą (w celu zwiększenia częstotliwości szumu) oryginalnych współrzędnych *xy* wierzchołka. Współrzędna *z* to czas. Wektor do obserwatora liczymy analogicznie jak w przebiegach Pass Teapot i Pass Spring.

w Vertex Shaderze przebiegu Pass Water:

a) Dodaj zmienną globalną `float time;`

b) Dodaj zmienną globalną `float4x4 matViewInverse;`

c) Dodaj do struktury `VS_OUTPUT`:

```
float3 PosWorld   : TEXCOORD0;
float3 ViewDir    : TEXCOORD1;
float3 NoiseCoord : TEXCOORD2;
```

a) Liczenie NoiseCoord dodaj jako pierwszą operację w shaderze wierzchołków:

```
Output.NoiseCoord = float3(Input.Position.xy*10.0, time);
```

b) Liczenie ViewDir i PosWorld można dodać tuż przed zwróceniem wyniku:

```
Output.ViewDir  = mul( float4(0,0,0,1), matViewInverse ) - Input.Position;
Output.PosWorld = Input.Position.xyz / 400.0;
```

9. Przejdźmy do edycji shadera pikseli i dodajmy niezbędne zmienne globalne i parametry.

w Pixel Shaderze przebiegu Pass Water:

a) Dodaj zmienne globalne:

```
float exposure;
samplerCUBE envmap;
sampler3D perlin;
```

b) Dodaj parametry funkcji `ps_main()`:

```
float4 ps_main(float3 Position : TEXCOORD0, float4 ViewDir  : TEXCOORD1,
   float3 NoiseCoord : TEXCOORD2) : COLOR0
```

10. Policzmy teraz zaburzoną normalną, oraz wersor promienia odbitego i załamanego. Dwuwymiarowy wektor zaburzenia normalnej policzymy próbkując dwukrotnie (drugi raz w punkcie z pewnym przesunięciem) teksturę trójwymiarowego szumu Perlina. Tekstura ta ma format L8, co oznacza tylko jedną składową koloru, która jest kopiowana przy odczycie na wszystkie pozostałe. Dlatego nie możemy obliczyć zaburzenia przy jednym odczycie tekstury.

na początku funkcji `ps_main()` w przebiegu Pass Water:

```
float2 error = 2.0 * float2(tex3D( perlin, NoiseCoord ).r,
                tex3D( perlin, NoiseCoord+float3(.5,.5,.5) ).r) -
                float2(1.0,1.0);
 float3 Normal    = normalize( float3(error.x,20.0,error.y) );
 ViewDir = normalize(ViewDir);
 float3 ViewRefl  = reflect( -ViewDir, Normal );
 float3 ViewRefr  = normalize(-ViewDir-1.5*Normal);
```

11. Teraz potrzebna będzie funkcja pomocnicza, która mając dany punkt wewnątrz sześcianu [-1,1]x[-1,1]x[-1,1] oraz wektor kierunku promienia wychodzącego z tego punktu, policzy przecięcie tego promienia z najbliższą ścianą sześcianu. Celem tej funkcji jest policzenie współrzędnych tekstury sześciennej na podstawie pozycji renderowanego piksela w scenie i promienia odbitego lub załamanego.

w Pixel Shaderze:
Napisz funkcję:

```
float3 ray_to_texcoord(float3 point, float3 dir)
```

, która przyjmuje dany punkt i promień, a następnie liczy i zwraca przecięcie promienia wypuszczonego z tego punktu z najbliższą ścianą sześcianu (czyli współrzędne mapy sześciennej). Z funkcji HLSL wystarczy wykorzystać `abs()` oraz `min()`.

12. Dodajmy funkcję liczenia współczynnika Fresnela na podstawie kąta patrzenia względem powierzchni (współczynnik Fresnela posłuży do zmieszania światła odbitego i załamanego w odpowiedniej proporcji).

```
const float rzero = pow((1.33 - 1.0)/(1.33 + 1.0), 2.0);
float fresnel(float cosfi)
{
   return rzero + (1.0-rzero) * pow(1.0-cosfi, 5.0);
}
```

13. Mamy już gotowe wszystkie elementy potrzebne do dokończenia shadera pikseli. Pozostaje wywołać funkcję ray_to_texcoord() dwukrotnie, by uzyskać współrzędne tekstury sześciennej dla promienia odbitego i załamanego, pobrać kolory z tekstury, pomnożyć je przez współczynnik naświetlenie, dla promienia załamanego pomnożyć pobrany kolor przez kolor wody, zmieszać kolory dla promienia odbitego i załamanego na podstawie współczynnika Fresnela.

w funkcji `ps_main()` usuwamy `return ...` i dodajemy końcówkę:

```
   ViewRefl = ray_to_texcoord( Position, ViewRefl );
   ViewRefr = ray_to_texcoord( Position, ViewRefr );
   float4 refl = exposure * texCUBE( envmap, ViewRefl );
   float4 refr = float4(.7,.8,1.0,1.0)* exposure * texCUBE( envmap,
ViewRefr );
   float w = fresnel(abs(normalize(ViewDir).y));
   return( lerp(refr,refl,w) );
```

14. Gotowe. Pozostaje odblokować upływ czasu: prawy przycisk na time, Variable Semantic, Time0_X. W zależności od tego jak zaimplementowana została funkcji `ray_to_texcoord` shader pikseli mieści się w 64 instrukcjach lub nie. W tym drugim przypadku trzeba zarówno w Vertex Shaderze jak i w Pixel Shaderze w przebiegu Pass Water zmienić Target na odpowiednio vs_3_0 i ps_3_0.