Zadanie 4: Mapy światła i cienia

Zadanie

W tym zadaniu dany jest projekt renderujący scenę przedstawiającą pokój z bujającą się lampą zawieszoną pod sufitem. Pierwszym zadaniem będzie rzutowanie perspektywiczne tekstury światła (białe koło z rozmytym brzegiem na czarnym tle) na całą scenę, zgodnie z aktualnym położeniem lampy. Drugim zadaniem będzie dodanie cieni przy pomocy map cienia. Wykorzystanie mapy cienia w scenie przebiega analogicznie do wykorzystania mapy światła, dlatego ta część sprowadza się głównie do odpowiedniego zainicjalizowania parametrów tekstury oraz wyrenderowania jej zawartości (bufor głębokości sceny widzianej z pozycji lampy).

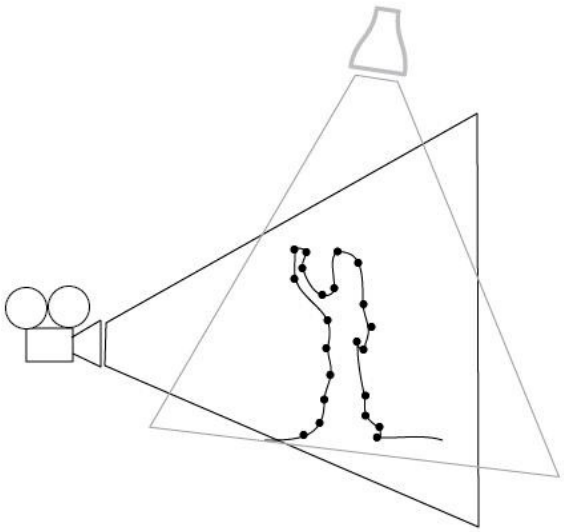
Bez map światła i cienia

Z mapami światła i cienia.

Uzupełnić należy funkcje klasy LightShadowEffect w pliku gk2_lightShadowEffect.cpp oraz powiązany z nią kod shaderów w pliku LightShadow.hlsl. Drobnych poprawek wymagać będzie też funkcja Render klasy Room (plik gk2_room.cpp).

1. Wyznaczanie współrzędnych tekstury

(funkcja UpdateLight oraz shader pikseli)



Perspektywiczne rzutowanie tekstury bardzo przypomina realizowanie rzutowania wierzchołków na ekran. Mamy daną macierz przekształcenia do układu kamery (w tym wypadku lampy) i macierz rzutowania perspektywicznego. Przekształcamy punkty z układu sceny (świata) przez obie macierze. W przypadku tekstur musimy jeszcze przeskalować i przesunąć wynik, ponieważ zakres współrzędnych tekstury jest inny niż zakres współrzędnych okna widoku (dla okna widoku współrzędne x oraz y są z zakresu [-1, 1]; dla tekstury współrzędne te są z zakresu [0, 1] – dodatkowo w przypadku współrzędnych tekstury oś Y jest odwrócona względem współrzędnych okna widoku).

Najpierw należy ustalić przekształcenie dla współrzędnych tekstury z układu sceny do układu lampy (w funkcji UpdateLight()). Będzie to iloczyn kolejno: macierzy widoku lampy, macierzy perspektywy dla lampy, macierzy skalowania XMMatrixScaling(0.5f, -0.5f, 1.0f) oraz macierzy przesunięcia XMMatrixTranslation(0.5f, 0.5f, 0.0f). Te dwie ostatnie służą przekształceniu współrzędnych okna widoku do współrzędnych tekstury.

Drugim krokiem jest wyznaczenie w shaderze pikseli współrzędnych tekstury dla danego piksela. Otrzymujemy je poprzez przekształcenie współrzędnych piksela w układzie świata (pole worldPos w strukturze PSInput) przez opisaną powyżej macierz (dostępną w zmiennej mapMatrix). Aby poprawnie wykonać dzielenie perspektywiczne należy jeszcze podzielić trzy pierwsze współrzędne wynikowego wektora (x, y oraz z) podzielić przez wartość czwartej współrzędnej w (Uwaga: Warto zauważyć, że w shaderze wierzchołków po przekształceniu pozycji wierzchołka przez macierz perspektywy, nie wykonujemy dzielenia współrzędnych otrzymanego wyniku przez współrzędną w. Tam nie musimy tego robić, gdyż operację tą automatycznie wykonuje za nas karta graficzna w fazie rasteryzacji. W przypadku perspektywicznego rzutowania tekstury nie mamy jednak tego mechanizmu i dzielenie to musimy wykonać ręcznie). Mimo, że do adresowania tekstury używać będziemy współrzędnych x i y otrzymanego wektora, należy również wykonać dzielenie dla współrzędnej z – przyda nam się ona później przy mapie cieni.

2. Renderowanie z użyciem mapy światła

Kolor światła używany dalej przy cieniowaniu piksela należy pobrać z mapy światła. Jest ona już zainicjalizowana i dostępna w zmiennej lightMap. Do adresowania tej tekstury należy użyć współrzędnych x i y otrzymanego w poprzednim kroku wektora. Mapa światła dla współrzędnych w pobliży i poza krawędziami tekstury zwróci nam kolor czarny. Chcemy jednak, aby piksele dla których wartość zwrócona z mapy światła jest za niska, były oświetlone pewnym domyślnym przyciemnionym światłem (jego kolor zapisany jest w zmiennej defaultLightColor). Należy więc porównać wartość zwróconą z mapy światła z wartością domyślną i wziąć większą z nich – jako że mapa cienia zawiera tylko odcienie szarości, porównywać można tylko jedną ze współrzędnych obu kolorów.

Aby zobaczyć wynik renderowania z użyciem mapy światła należy w funkcji Render w klasie Room zamienić m_phongEffect używany przy renderowaniu sceny, na m_lightShadowEffect.

3. Projekcja wsteczna
(poprawka do shadera pikseli)

Istnieje pewna różnica pomiędzy rzutowaniem geometrii na ekran i rzutowaniem geometrii w celu wygenerowania współrzędnych tekstury. W pierwszym przypadku geometria nieznajdująca się pomiędzy bliższą i dalszą płaszczyzną obcinania jest wykluczana, w drugim nie. Dlatego mapa światła na powyższym obrazie rzutowana jest zarówno na geometrię poniżej jak i powyżej lampy (jaśniejsze koło na suficie).

Najbardziej uniwersalnym rozwiązaniem byłoby przekazanie do shadera pikseli położenia lampy i wektora kierunku w jakim rzuca światło oraz odległości bliższej płaszczyzny obcinania wzdłuż tego wektora (płaszczyzna obcinania byłaby do tego wektora prostopadła). Dla danego piksela należałoby sprawdzić, czy znajduje się on przed czy za tą płaszczyzną. Na szczęście odchylenia lampy są na tyle małe, że można ten przypadek trochę uprościć i zastosować płaszczyznę obcinania prostopadłą do osi OY przechodzącą przez aktualne położenie światła. Należy więc porównać położenie piksela w układzie sceny (pole worldPos struktury PSInput) z położeniem światła (zmienna lightPosPS) i jeżeli współrzędna y położenia piksela jest większa niż współrzędna y położenia światła, należy zastosować domyślny, przyciemniony kolor światła zamiast tego pobranego z mapy światła.

4. Inicjalizacja tekstury mapy cienia
(funkcja InitializeTextures)

Zanim zaczniemy renderować zawartość mapy cieni, musimy odpowiednio zainicjalizować tą teksturę. Tekstura będzie spełniać dwa zadania: będzie służyła jako bufor głębokości przy renderowaniu sceny z punktu widzenia światła, oraz jako tekstura przy finalnym renderowaniu sceny na ekran. Dlatego będziemy musieli stworzyć dla niej dwa widoki: ID3D11DepthStencilView dla pierwszego zadania, oraz ID3D11ShaderResourceView dla drugiego.

Do poprawnej inicjalizacji samej tekstury musimy podać jej wymiary (pola Width i Height struktury D3D11_TEXTURE2D_DESC – ich wartość powinna być równa stałej TEXTURE_SIZE), format (pole Format – wartość powinna być równa DXGI_FORMAT_R32_TYPELESS; musimy użyć formatu R32_TYPELESS, gdyż w przypadku patrzenia na teksturę jako bufor głębokości chcemy interpretować jej wartości jako typ D32_FLOAT, natomiast w przypadku patrzenia na nią jako teksturę w shaderze pikseli interpretujemy dane z niej pobierane jako R32_FLOAT. Te dwa ostatnie formaty nie są ze sobą do końca kompatybilne – mimo, że reprezentacja bitowa jest taka sama – i jedynym formatem, który można konwertować na oba z nich jest R32_TYPELESS). Pole BindFlags powinno być równe bitowej operacji OR wartości D3D11_BIND_DEPTH_STENCIL oraz D3D11_BIND_SHADER_RESOURCE, natomiast pole MipLevels powinno mieć wartość 1 (używamy tylko jednego poziomu mipmap).

Do inicjalizacji obiektu ID3D11DepthStencilView dla mapy cieni wystarczy zmodyfikować format (pole Format struktury D3D11_DEPTH_STENCIL_VIEW_DESC, którego wartość powinna być równa DXGI_FORMAT_D32_FLOAT).

Struktura D3D11_SHADER_RESOURCE_VIEW_DESC opisująca obiekt ID3D11ShaderResourceView dla mapy cieni powinna w polu Format mieć wartość DXGI_FORMAT_R32_FLOAT, a w polu Texture2D.MipLevels wartość 1.

5. Renderowanie z użyciem mapy cienia
(poprawki w shaderze pikseli i funkcji UpdateLight)

Do adresowania mapy cieni w shaderze pikseli używamy tych samych współrzędnych, co do adresowania mapy światła. Różnica będzie tylko w wartości zwracanej z metody Sample. W przypadku mapy światła był to kolor (typ float4), natomiast samplowanie mapy cieni zwróci tylko pojedynczą wartość (skalar) typu float. Wartość ta jest wartością z bufora głębokości w miejscu, gdzie znalazłby się aktualnie rozpatrywany piksel przy renderowaniu sceny z punktu widzenia lampy. Wartość tą należy porównać z wartością współrzędnej z wektora wyznaczanego w punkcie 1. Jeżeli wartość pobrana z mapy cienia jest mniejsza niż wartość współrzędnej z tego wektora, to znaczy, że na drodze pomiędzy tym pikselem a źródłem światła, przy renderowaniu sceny z punku widzenia lampy znalazł się jakiś inny obiekt, czyli aktualnie rozpatrywany piksel znajduje się w cieniu. Dla takiego piksela, zamiast koloru światła pobranego z mapy światła, należy użyć koloru domyślnego (defaultLightColor).

Aby zobaczyć efekt użycia mapy cieni, należy z funkcji Render w klasie Room odkomentować linie kodu odpowiedzialne za renderowanie zawartości mapy cieni.

Po uruchomieniu programu powinny pojawić się znaczne artefakty. Wynika to z faktu, że obliczanie współrzędnej z zrzutowanej perspektywicznie z punktu widzenia lampy nie są dokładne, a piksel na ekranie nigdy nie odpowiada bezpośrednio środkowi piksela w mapie cienia. Aby temu zapobiec należy tą współrzędną z zmniejszyć o jakąś małą wartość. Najłatwiej to zrobić modyfikując funkcję UpdateLamp, gdzie tworzona jest macierz przekształcenia współrzędnych tekstury (patrz punkt 1). Częścią składową tej macierzy jest macierz translacji, należy ją zmodyfikować tak, by dodatkowo następowało przesunięcie wzdłuż osi Z o wartość -0.00001f.

6. Uwagi końcowe

Mapy cienia są jedną z możliwych metod uzyskania cieni dla dynamicznej sceny. Inną popularną metodą są bryły cienia. Dla brył cienia uzyskuje się dużą dokładność wyznaczenia krawędzi cienia, podczas gdy dla map cienia może pojawić się aliasing jego krawędzi (jego wielkość zależy tutaj od rozdzielczości mapy cienia oraz kąta między kierunkiem światła i kierunkiem patrzenia) i błędy określenia czy obiekt jest w cieniu dla płaszczyzn prawie równoległych do kierunku światła. Z kolei mapy cienia można wykorzystać w połączeniu z teksturami zawierającymi maskę przezroczystości (np. prostokąty reprezentujące liście na drzewie z nałożoną teksturą posiadającą odpowiednią maskę przezroczystości), zaś bryły cienia operują wyłącznie na krawędziach geometrii sceny. Istnieje wiele odmian map cienia, które poprawiają ich precyzję (Trapezoidal Shadow Maps, Perspective Shadow Maps) oraz zapewniają lepsze filtrowanie i rozmycie(Variational Shadow Maps). Osobnym problemem, ale również dość dobrze już opracowanym, jest liczenie realistycznych cieni dla świateł powierzchniowych oraz światła otaczającego dochodzącego ze wszystkich kierunków (Ambient Occlusion).