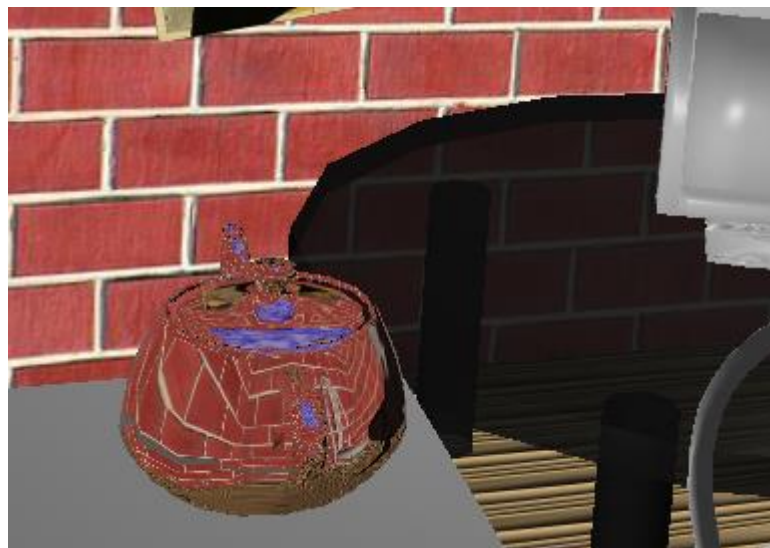


Zadanie 2. Mapowanie środowiska

Zadanie

W pliku „gk2_environmentMapper.cpp” należy uzupełnić funkcje „InitializeTextures”, „SetupFace” oraz „EndFace” i zmodyfikować shader wierzchołków zawarty w pliku „envMapVS.hlsl” tak, aby uzyskać efekt odbić sceny w powierzchni obiektu czajnika. Wykonanie efektu mapowania środowiska składa się z dwóch kroków:

- renderowanie sceny widzianej ze środka obiektu, w którym ma się odbijać scena, do każdej ze ścian mapy sześciennej („SetupFace” i „EndFace”)
- renderowania obiektu, w którym ma się odbijać scena, odpowiednio teksturowanego przy użyciu mapy sześciennej („envMapVS.hlsl” i „envMapPS.hlsl”)



1. Inicjalizacja tekstur.

W klasie EnvironmentMapper zdefiniowane są potrzebne pola:

- m_nearPlane, m_farPlane – odległość bliższej i dalszej płaszczyzny obcinania dla kamery podczas renderowania sceny do mapy sześciennej.
- m_position – położenie środka czajnika w układzie sceny.
- TEXTURE_SIZE – rozmiar pojedynczej ściany w teksturze sześciennej.

Dodatkowo zdefiniowane są pola dla tekstur i odpowiadających im widoków:

- m_envTexture – tekstura sześcienna, wykorzystywana przy mapowaniu środowiska
- m_envTextureView – widok tekstury sześciennej, wymagany do przekazania tekstury na wejściu piksel shadera
- m_envFaceRenderTexture – tymczasowa tekstura do której renderowane będą pojedyncze ściany tekstury sześciennej
- m_envFaceRenderTarget – widok tekstury tymczasowej, potrzebny do ustawienia tekstury jako wyjściowego bufora kolorów dla potoku renderowania

- `m_envFaceDepthTexture` – tymczasowa tekstura służąca za bufor głębokości przy renderowaniu poszczególnych ścian tekstury sześcienniej
- `m_envFaceDepthView` – widok tymczasowej tekstury głębokości, potrzebny do ustawienia tej tekstury jako bufora głębokości dla potoku renderowania.

Aby poprawnie zainicjalizować teksturę `m_envFaceRenderTexture`, należy zmodyfikować obiekt typu `D3D11_TEXTURE2D_DESC`, na podstawie którego jest ona tworzona, tak aby wymiary odpowiadały rozmiarowi pojedynczej ściany tekstury sześcienniej. Dodatkowo należy ustawić pole `BindFlags` na wartość `D3D11_BIND_RENDER_TARGET`, aby można było na jej podstawie stworzyć odpowiedni widok, natomiast pole `MipLevels` ustawić na wartość 1.

Przy tworzeniu tekstury sześcienniej `m_envTexture`, obiekt `D3D11_TEXTURE2D_DESC` musi określać poprawne wymiary pojedynczej ściany (pola `Width` i `Height`), pole `BindFlags` powinno zawierać wartość `D3D11_BIND_SHADER_RESOURCE`, pole `MipLevels` wartość 1, pole `MiscFlags` wartość `D3D11_RESOURCE_MISC_TEXTURECUBE`, natomiast pole `ArraySize` wartość 6.

Niestandardowych parametrów wymaga również tworzenie widoku tekstury sześcienniej `m_envTextureView`. W obiekcie `D3D11_SHADER_RESOURCE_VIEW_DESC` musimy określić, że chcemy patrzeć na teksturę jak na teksturę sześcienną (po przez ustawieni pola `ViewDimension` na wartość `D3D11_SRV_DIMENSION_TEXTURECUBE`), a także określić, że chcemy korzystać tylko z najwyższego poziomu szczegółowości tekstury, gdyż więcej ich nie zdefiniowaliśmy (pole `TextureCube.MipLevels` powinno mieć wartość 1, a `TextureCube.MostDetailedMip` wartość 0).

2. Renderowanie do mapy sześcienniej

Przed wyrenderowaniem sceny do każdej ze ścian tekstury sześcienniej wywoływana będzie funkcja „SetupFace”. Należy w niej dodać:

- Zmianę macierzy widoku (przekształcenie z układu świata, do układu kamery) tak, jakby kamera znajdowała się w środku czajnika i skierowana była na renderowaną właśnie ścianę (przydatna może być funkcja `XMMatrixLookToLH`).
- Zmianę macierzy rzutowania (perspektywy) tak, aby kąt widzenia w pionie wynosił 90°, obraz był kwadratowy, a bliższa i dalsza płaszczyzna obcinania wynosiła odpowiednio `m_nearPlane` oraz `m_farPlane`.
- Zmianę współrzędnych okna widoku (`D3D11_VIEWPORT`) aby lewy górny róg znajdował się w punkcie (0; 0), okno miało wymiary równe wymiarom pojedynczej ściany tekstury sześcienniej, a bliższa i dalsza głębokość wynosiły odpowiednio 0 i 1.

Po wyrenderowaniu ściany wywołana będzie funkcja „EndFace”, w której należy przekopiować dane z tekstury tymczasowej do odpowiedniej ściany tekstury sześcienniej. Służy do tego metoda `CopySubresourceRegion` w obiekcie `m_context`. Poza teksturą docelową i źródłową istotny jest drugi parametr, który określa w której ścianie mają być umieszczone przekopiowane dane (można użyć wartości `m_face`). Pozostałe parametry można ustawić na 0.

Ostatnia zmiana na tym etapie ma miejsce w pliku „gk2_room.cpp”. W funkcji `Render` należy w pętli wyrenderować scenę (`DrawScene`) do każdej ze ścian tekstury sześcienniej dla obiektu `m_environmentMapper`.

3. Obliczanie współrzędnych tekstury.

Wprowadzenie

Współzrzednymi dla mapy sześciennnej sę wektory o trzech współzrzednych. Taki wektor wyznacza kierunek. Promień wypuszczony ze środka sześcianu w określonym kierunku przetnie jego powierzchnię w pewnym punkcie. Tak mniej więcej wygląda schemat adresowania przy użyciu mapy sześciennnej.

Jeśli świat ma się odbijać w powierzchni obiektu, to aby wyznaczyć kolor w pewnym jego punkcie musimy promień wzroku odbić od powierzchni obiektu i śledzić jego drogę po odbiciu. W przypadku mapowania środowiska z użyciem mapy sześciennnej, po prostu wykorzystujemy kierunek po odbiciu przy adresowaniu mapy sześciennnej. Aby uzyskać poprawny wynik należy uważać, aby wszystkie operacje przeprowadzać w układzie sceny.

Wykonanie

Należy uzupełnić shader wierzchołków („main” w pliku envMapVS.hlsl) tak, aby współzrzednymi tekstury były współzrzedne wektora wzroku odbitego od powierzchni obiektu. Pomocna może być funkcja „reflect”. Przekazany do niej promień wzroku musi iść od kamery w kierunku pozycji danego wierzchołka, natomiast wektor normalny przed wywołaniem tej funkcji musi być przekształcony do układu sceny. Nie należy zapomnieć o normalizacji wektorów.

4. Wyświetlanie

W metodzie „DrawTeapot” klasy Room należy podmienić domyślny efekt m_phongEffect, na nowo stworzony m_environmentMapper. Dodatkowo można odkomentować umieszczony tam kod rysujący jednostkową sferę w miejsce czajnika. Może się ona przydać do testowania poprawności generowania mapy środowiska i obliczania odbić.

