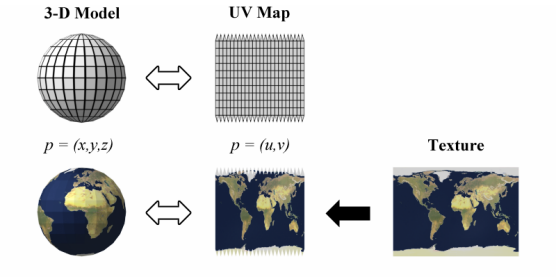
1 Wprowadzenie

Celem ćwiczenia jest pokazanie podstawowych technik teksturowania powierzchni obiektów z wykorzystaniem mechanizmów biblioteki OpenGL z biblioteką GLUT. Na przykładach zilustrowana będzie droga od przeczytania obrazu tekstury, do nałożenia jaj fragmentów na poszczególne fragmenty modelu obiektu trójwymiarowego.

2 Teksturowanie powierzchni

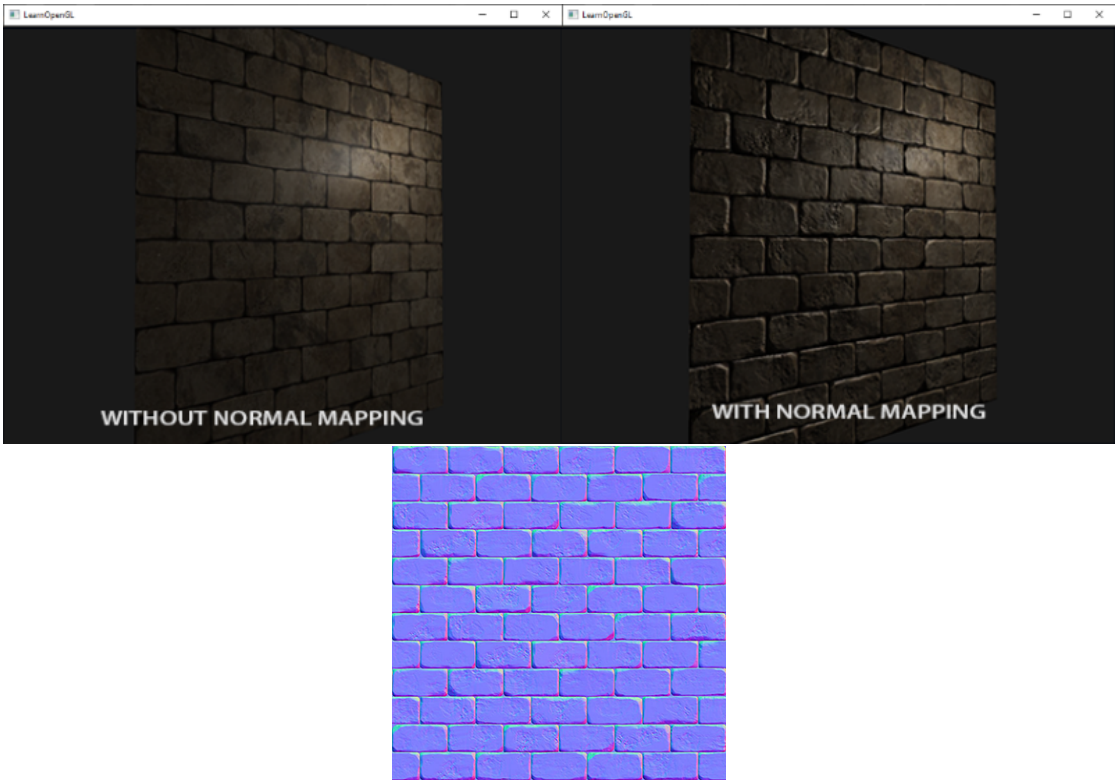
Teksturowanie polega na nanoszeniu na powierzchnie elementów modelu sceny obrazów zadanych w postaci map bitowych. Obrazy te zwykle umieszczone są w osobnych plikach. Ogólnie teksturowanie sprowadza się do trzech czynności. Po pierwsze odczytania z pliku obrazu tekstury i umieszczeniu odpowiednich danych w pamięci. Po drugie zdefiniowania tekstury, przez co należy rozumieć określenie sposobu interpretacji odczytanych z pliku danych i wreszcie po trzecie nałożenia tekstury na elementy modelu obiektu.

Proces nakładania dwuwymiarowej tekstury na trójwymiarowy obiekt nazywa się UV mapping.



Rysunek 1: Nakładanie dwuwymiarowego obrazu na trójwymiarową sferę.

Program realizuje mapowanie kolorów obieku (diffuse mapping), metodę która (w uproszczeniu) mapuje kolory z pliku graficznego bezpośrednio jako kolory fragmentów rysowanych obiektów. Dostępne jest także m.in. mapowanie wypukłości, mapowanie normalnych, mapowanie odbić zwierciadlanych, etc. Wszystkie te techniki używają tekstur do określania poziomów tych parametrów niezależnie od ilości i pozycji wierzchołków.



Rysunek 2: Przykład zastosowania metody normal mapping. Powyżej przykład przed i po zaaplikowaniu tekstury. Poniżej zastosowana tekstura.

3 Realizacja programu

3.1 Wczytywanie danych obrazu

Aby wyświetlić teksturę na rysowanych obiektach, należy najpierw wczytać ją z pliku. Posłuży do tego funkcja `LoadTGAImage()` wczytująca nasz obraz w formacie TGA.

3.2 Inicjalizacja

W funkcji `init()` pojawił się następujący kod:

```
GLint ImWidth, ImHeight, ImComponents;
GLenum ImFormat;
GLbyte* texture = LoadTGAImage("tekstury/P4_t.tga", &ImWidth, &ImHeight,
    &ImComponents, &ImFormat);
glTexImage2D(GL_TEXTURE_2D, 0, ImComponents, ImWidth, ImHeight,
    0, ImFormat, GL_UNSIGNED_BYTE, texture);
glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
free(texture);
```

Wartości kolorów pikseli obrazu są wczytywane przez funkcję `LoadTGAImage()` i umieszczane w tablicy `texture`. Następnie funkcja `glTexImage2D` tworzy teksturę z wczytanego obrazu. Funkcja przyjmuje następujące argumenty:

- `GL_TEXTURE_2D`: **target** - Rodzaj definiowanej tekstury, zawsze powinno być `GL_TEXTURE_2D`
- `0`: **level** - Poziom szczegółowości. Jest ustawiany na 0 gdy mipmapy nie są stosowane, gdy mipmapy są używane level jest liczbą określająca poziom redukcji obrazu tekstury.
- `ImComponents`: **components** - Ilość używanych składowych koloru, liczba od 1 do 4.
- `ImWidth`: **width** - Szerokość obrazu tekstury, zawsze jest potegą liczby 2, może być powiększona o szerokość tak zwanej ramki tekstury
- `ImHeight`: **height** - Wysokość obrazu tekstury, zawsze jest potęgą liczby 2, może być powiększona o szerokość ramki tekstury.
- `0`: **border** - Szerokość ramki tekstury, może wynosić 0, 1 lub 2
- `ImFormat`: **format** - Format danych obrazu tekstury, określa co opisują dane, czy są indeksami kolorów czy ich składowymi (np. R, G, B) i w jakiej są podane kolejności.
- `GL_UNSIGNED_BYTE`: **type** - Typ danych dla punktów obrazu tekstury, określa jak kodowane są dane, istnieje możliwość używania różnych formatów liczb, od 8 bitowych liczb stałoprzecinkowych, do 32 bitowych liczb zmiennoprzecinkowych.
- `texture`: **pixels** - Tablica o rozmiarze $width \times height \times components$, w ktorej znajdują się dane z obrazem tekstury.

Następnie kolejno:

1. Za pomocą funkcji `glTexEnvi()` ustawiamy funkcję teksturującą jako `GL_MODULATE`. Funkcja teksturująca określa jak łączyć kolor obrazu tekstury z kolorem obrazu ekranu. `GL_MODULATE` oznacza że kolor piksela ekranu jest mnożony przez kolor piksela tekstury, następuje w ten sposób mieszanie barw tekstury i tego co jest teksturowane.

2. Za pomocą funkcji `glTexParameteri()` określamy algortm dodawania nowych piksel w przypadku, gdy w obrazie tekstury jest za mało punktów aby wypełnić obraz teksturowanego elementu oraz usuwania pikseli, gdy w obrazie tekstury jest ich za dużo. W obu przypadkach stosujemy `GL_LINEAR` co oznacza, że w przypadku potrzeby powiększenia lub pomniejszenia tekstury będziemy stosować interpolację liniową.

3.3 Definiowanie współrzędnych tekstury dla wierzchołków

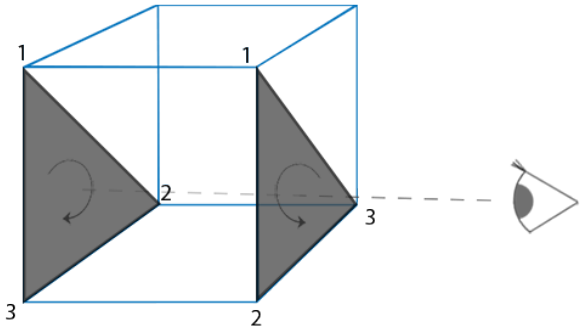
Do określania współrzędnych tekstury dla wierzchołków wykorzystano funkcję `glTexCoord2f`. Przyjmuje ona jako argumenty liczby zmiennoprzecinkowe `u` oraz `v`, oznaczające odpowiednio pozycję wierzchołka na teksturze wzdłuż osi x i y.

Przykładowo funkcja rysująca wierzchołki jaja wygląda w następujący sposób:

```
void drawVertexTextured(int u, int v) {
    glNormal3fv(normals[u][v]);
    glTexCoord2f((float)v / (N-1), (float)u / (N-1));
    glVertex3fv(points[u][v]);
}
```

3.4 Back-face culling

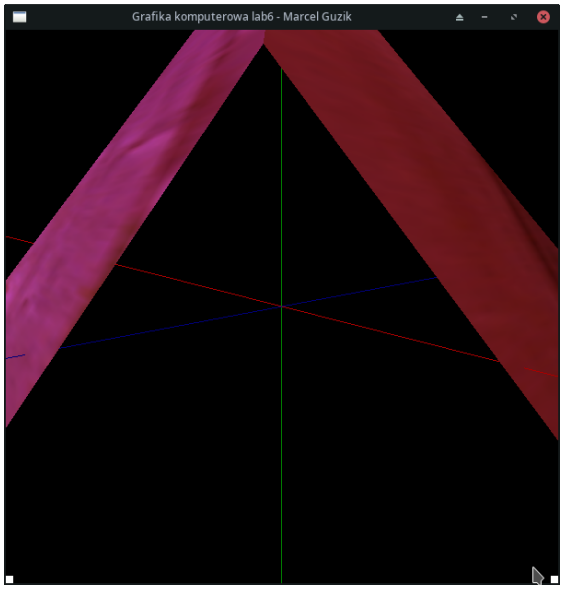
Aby nie teksturować niewidocznych dla obserwatora powierzchni wewnątrz obiektów, stosuje się mechanizm back-face culling. Mechanizm ten określa czy powierzchnia ustawiona jest do obserwatora przodem lub tyłem i teksturuje ją tylko jeśli jest ustawiona przodem. Kierunek jest ustalany na podstawie kolejności rysowania wierzchołków stanowiących tą powierzchnię, tzn. czy są rysowane zgodnie z kierunkiem wskazówek zegara lub w kierunku przeciwnym. Domyślnie powierzchnie rysowane w kierunku przeciwnym kierunkowi wskazówek zegara są określane jako stojące przodem.



Rysunek 3: Z punktu widzenia obserwatora wierzchołki stanowiące ściany po przeciwnych stronach sześcianu będą rysowane w odwrotnych kierunkach.

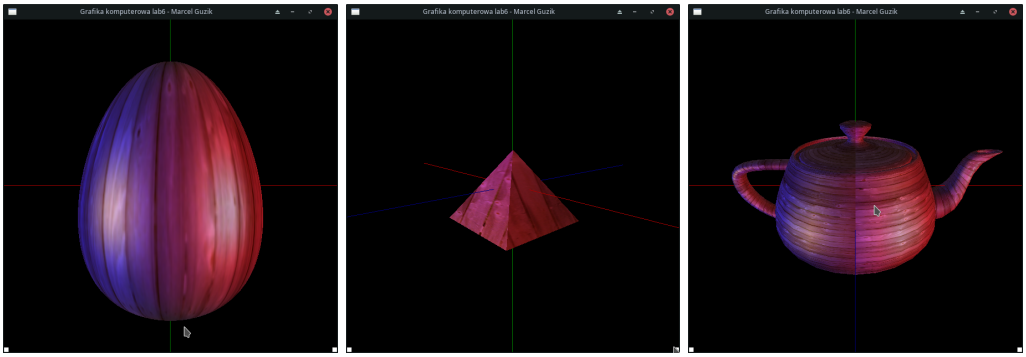
W programie proces uruchamiany jest przez funkcję `glEnable(GL_CULL_FACE)`.

```
void drawPyramid(float height) {
    glEnable(GL_CULL_FACE);
    glBegin(GL_QUADS);

    // ...

    glEnd();
    glDisable(GL_CULL_FACE);
}
```



Rysunek 4: Mechanizm back-face culling pozwala na wyeliminowanie ścian wewnętrznych obiektów z procesu teksturowania i zaoszczędzenie mocy obliczeniowej.

4 Podsumowanie



Rysunek 5: Finalna wersja programu teksturująca obiekty.

Zadanie zaprezentowało możliwości teksturowania oraz mechanizmu back-face culling w bibliotece OpenGL.