Grafika komputerowa i komunikacja człowiek - komputer

Open GL – teksturowanie powierzchni obiektów

14.12.2022r.

Maciej Radecki
album: 253257

Prowadzący: dr inż. Jan Nikodem

Termin: Piątek TN, 16:25

1. Wstęp

Celem laboratorium było zaprezentowanie za pomocą narzędzi biblioteki OpenGL z rozszerzeniem GLUT podstawowych technik teksturowania powierzchni obiektów (przykładowo jak przeczytać obraz tekstury i nałożyć jej fragmenty na model obiektu trójwymiarowego, np. na trójkąt, wielościan czy model jajka w postaci siatki trójkątów).

Teksturowanie to proces polegający na nanoszeniu na powierzchnie elementów modelu sceny obrazów reprezentowanych za pomocą map bitowych. W ogólnym przypadku sprowadza się do trzech czynności:

1. odczytania z pliku obrazu tekstury i umieszczeniu odpowiednich danych w pamięci,
2. zdefiniowania tekstury (określenie sposobu interpretacji odczytanych z pliku danych),
3. nałożenia tekstury na elementy modelu obiektu.

Biblioteka OpenGL zapewnia jedynie funkcje realizujące definiowanie i nakładanie tekstur.

Aby możliwe było teksturowanie modelu, program z obracającym się jajkiem z poprzednich laboratoriów należy uzupełnić program o kod funkcji służącej do odczytywania obrazu tekstury z pliku graficznego w formacie TGA (targa). Następnie, w funkcji MyInit() należy dopisać kilka linii kodu istotnych dla mechanizmu renderującego OpenGL, które zawierają podstawowe funkcje definiujące właściwości procesu teksturowania: *glTexImage2D()*, *glTexParameteri()* oraz *glTexEnvi()*.

1. Funkcja *glTexImage2D()*

Funkcja ta służy do definiowania tekstury dwuwymiarowej. Określona jest jako

```
void glTexImage2D(GLenum target, GLint level, GLint components,
GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const
GLvoid *pixels)
```

Kolejne argumenty tej funkcji są zdefiniowane następująco:

- *target* oznacza rodzaj definiowanej tekstury, w przypadku naszego laboratorium zawsze będzie to wartość **GL_TEXTURE_2D** (bo mamy do czynienia tylko z teksturami 2-D),
- *level* oznacza poziom szczegółowości; jest ustawiany na 0, gdy mipmapy nie są stosowany (gdy mipmapy są używane, level jest liczbą, która określa poziom redukcji obrazu tekstury),
- *components* to ilość używanych składowych koloru (liczba od 1 do 4),
- *width* to szerokość obrazu tekstury (zawsze jest potęgą liczby 2, może być powiększona o szerokość tak zwanej ramki tekstury),
- *height* to wysokość obrazu tekstury (zawsze jest potęgą liczby 2, może być powiększona o szerokość ramki tekstury),
- *border* to szerokość ramki tekstury (może wynosić 0, 1 lub 2),
- *format* to format danych obrazu tekstury; określa co opisują dane, czy są indeksami kolorów czy ich składowymi (np. RGB) i w jakiej kolejności,
- *type* to 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,
- *pixels* to tablica o rozmiarze *width * height * components*, w której znajdują się dane z obrazem tekstury.

2. Funkcja *glTexParameteri()*

Funkcja pozwala na określenie algorytmów dodawania nowych pikseli w przypadku, gdy w obrazie tekstury jest za mało punktów, aby wypełnić obraz teksturowanego elementu oraz usuwania pikseli w przypadku, gdy w obrazie tekstury jest ich za dużo. Pozwala także określić sposób postępowania w przypadku, gdy zadane w programie współrzędne tekstury wykraczają poza zakres [0, 1]. Funkcja jest zdefiniowana następująco:

```
void glTexParameteri(GLenum target, GLenum pname, GLint param)
```

Poszczególne argumenty tej funkcji to:

- *target* – oznacza rodzaj definiowanej tekstury; w przypadku tekstur dwuwymiarowych powinien być ustawiony na **GL_TEXTURE_2D**
- *pname*
- *param*

Zestawienie wartości *pname* i *param* decyduje o szczegółach działania algorytmów teksturowania:

- gdy *pname* ma wartość **GL_TEXTURE_MIN_FILTER**, *param* opisuje w jaki sposób następuje pomniejszanie tekstury (usuwanie pikseli) i przyjmuje wartości: **GL_NEAREST** (usuwa się najbliższy punkt sąsiedni) lub **GL_LINEAR** (filtracja liniowa z odpowiednimi wagami),
- gdy *pname* ma wartość **GL_TEXTURE_MAG_FILTER**, *param* opisuje w jaki sposób następuje powiększanie tekstury (dodawanie pikseli) i przyjmuje wartości: **GL_NEAREST** (dodaje się najbliższy punkt sąsiedni) lub **GL_LINEAR** (interpolacja liniowa wykorzystująca wartości sąsiednich pikseli),
- gdy *pname* ma wartość **GL_TEXTURE_WRAP_S**, *param* opisuje sposób traktowania współrzędnej **s** tekstury, gdy wykracza ona poza zakres [0, 1] i przyjmuje wartości: **GL_CLAMP** (poza zakresem stosowany jest kolor ramki tekstury lub stały kolor) lub **GL_REPEAT** (tekstura jest powtarzana na całej powierzchni wielokąta),
- gdy *pname* ma wartość **GL_TEXTURE_WRAP_T**, *param* opisuje sposób traktowania współrzędnej **t** tekstury, gdy wykracza ona poza zakres [0, 1] i przyjmuje wartości: **GL_CLAMP** (poza zakresem stosowany jest kolor ramki tekstury lub stały kolor) lub **GL_REPEAT** (tekstura jest powtarzana na całej powierzchni wielokąta).

3. Funkcja *glTexEnvi()*

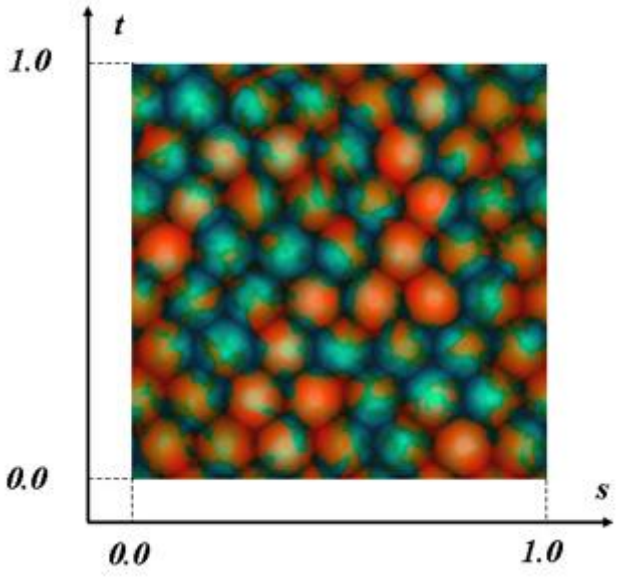
Funkcja ta służy do ustalenia trybu teksturowania (sposobu łączenia koloru piksela obrazu tekstury z kolorem piksela ekranu). Przyjmuje ona postać:

```
void glTexEnvi( GLenum target, GLenum pname, GLint param )
```

- *target* oznacza rodzaj definiowanej tekstury; należy go ustawić na **GL_TEXTURE_ENV**,
- *param* należy ustawić na **GL_TEXTURE_ENV_MOD**,
- sposób łączenia pikseli tekstury i ekranu określa argument *param*¸ który może przyjmować cztery wartości: **GL_MODULATE** (kolor piksela ekranu jest mnożony przez kolor piksela tekstury, przez co mieszają się barwy tekstury i tego, co jest teksturowane), **GL_DECAL** (piksele tekstury zastępują piksele na ekranie), **GL_BLEND** (kolor piksela ekranu jest mnożony przez kolor piksela tekstury i łączony ze stałym kolorem) i **GL_REPLACE** (działa podobnie jak **GL_DECAL**, różnica występuje wtedy, gdy wprowadzona zostaje przezroczystość).

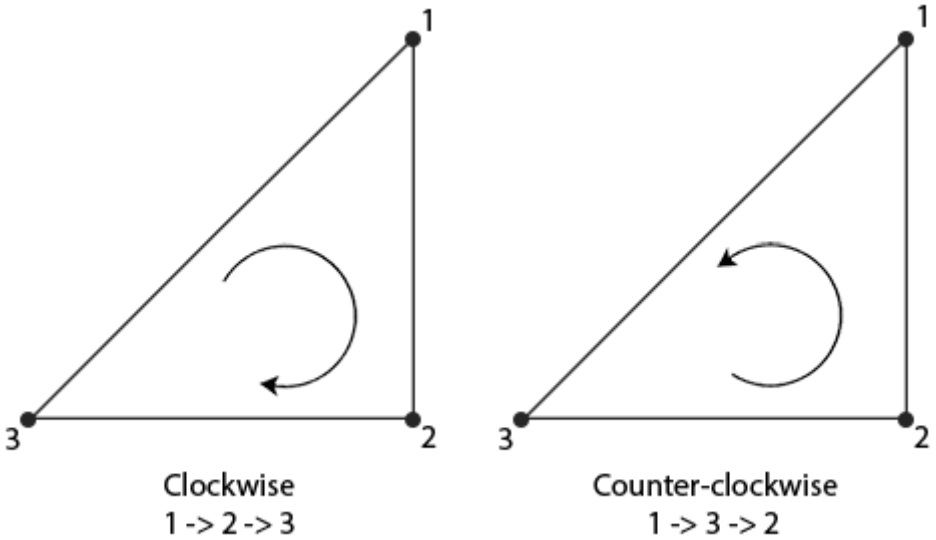
W kolejnym kroku należy udostępnić obraz tekstury zapisany w pliku graficznym w formacie TGA (w tym wypadku o rozmiarze 256 x 256 pikseli i o 24 bitowej informacji o kolorze piksela;

wybrano jedną z przykładowych tekstur z instrukcji). Na koniec należy w funkcji definiującej obiekt wprowadzić współrzędne wzorca tekstury definiujące fragment obrazu, jaki ma być naniesiony na powierzchnię trójkąta. Układ współrzędnych tekstury prezentuje się następująco:



Aby przyporządkować fragmenty przestrzeni tektury elementom modelu, należy, korzystając z funkcji *glTexCoord2f()* określić, które wierzchołki poligonu naniesionego na wzorzec tektury odpowiadają którym wierzchołkom teksturowanego modelu. Zmiana współrzędnych tekstury prowadzi do zmiany obrazu na modelu.

Warto tutaj zauważyć jeszcze jedną rzecz – w przypadku teksturowania widoczna będzie tylko część modelu. W związku z tym, aby zaoszczędzić na pamięci i mocy obliczeniowej, niewidoczna z punktu widzenia obserwatora część modelu (np. wnętrze jajka) nie powinna być teksturowana. Wykorzystywana jest do tego funkcja *glEnable(***GL_CULL_FACE***)*, która sprawia, że ściany niewidoczne nie będą teksturowane (np. wnętrze jajka). O tym, które ściany są niewidoczne, decyduje kolejność wierzchołków w trójkącie.



W powyższym przykładzie widać, że wierzchołki mają określoną kolejność w zależności od tego, czy idziemy zgodnie z ruchem wskazówek zegara, czy też przeciwnie. OpenGL analizuje wszystkie trójkąty danego modelu pod kątem kolejności wierzchołków i na podstawie tego wyciąga wnioski – trójkąty widoczne „z przodu" (*eng. front-facing*) zdefiniowane są przez wierzchołki w kolejności odwrotnej do ruchu wskazówek zegara (*eng. counter-clockwise*), natomiast trójkąty widoczne z tyłu (*eng. back-facing)* są zdefiniowane przez wierzchołki zgodne z ruchem wskazówek zegara (*eng. clockwise*).

2. Realizacja ćwiczenia

Podczas laboratorium do wykonania były dwa zadania. Pierwsze z nich polegało na stworzeniu trójwymiarowego modelu ostrosłupa i oteksturowanie go za pomocą dowolnej tekstury. Dodatkowo, model miał być stworzony w taki sposób, by użytkownik mógł ukrywać i pokazywać poszczególne ściany. Zadanie zostało zrealizowane w taki sposób, że kolejne ściany modelu można pokazać/ukryć kolejno za pomocą klawiszy 3-7 na klawiaturze.

Dodatkowo, z racji tego, że jako szkielet wykorzystano program z poprzednich laboratoriów, możliwe jest ruch wokół ostrosłupa za pomocą lewego przycisku myszy oraz przybliżanie/oddalanie się od niego za pomocą prawego przycisku myszy. Zarówno kolor materiału, z jakiego został wykonany model, jak i kolor światła padającego na niego, jest biały.

Podstawa powstała poprzez zdefiniowanie współrzędnych jej czterech wierzchołków. Ściany boczne powstały poprzez łączenie ze sobą dwóch sąsiednich wierzchołków podstawy z wierzchołkiem (0.0, 0.0, -10.0). Program wczytuje teksturę o nazwie „tekstura.tga" (w domyśle z programem załączona jest tekstura „P3_t.tga" pochodząca z instrukcji, o zmienionej nazwie).

Jako drugie zadanie należało oteksturować model jajka z poprzednich ćwiczeń w taki sposób, jakby tekstura była „naciągnięta" na jajko. Ponadto, jajko również posiada cechy poprzednich wersji jajka (oświetlenie białym światłem, obracanie i zoom za pomocą myszki). Dla tego jajka zdecydowano się na N = 60 (czyli bok kwadratu jednostki dziedziny parametrycznej zostanie podzielony na 60 części – w efekcie otrzymane zostanie dokładniejszy model jajka z lepiej nałożoną teksturą wyższej jakości). Wczytywana jest tekstura o nazwie „tekstura.tga".

Normalnie, aby model był poprawnie oteksturowany, należy pamiętać o tym, by jego kolejne wierzchołki zostały zadeklarowane w odpowiedniej kolejności (tzn. przeciwnej do kierunku wskazówek zegara), by OpenGL uznał je za zwrócone w kierunku użytkownika, dzięki czemu model zostanie oteksturowane tylko na zewnątrz, natomiast wewnątrz nie będzie nic.

Warto zauważyć, że jajko jest podzielone na połówki paskiem. Wynika to z faktu, że tekstura ta nie została zaprojektowana z myślą o nałożeniu na jajko – tekstury spotykane np. tekstury postaci w grach są dopasowane do nałożenia na ich model 3-D (posiadają odpowiedni kształt, odpowiednie elementy są odpowiednio rozdzielone np. twarz, ręce itd.).