Tekstury - część 1

1. *Informacje wstępne*

   (a) Mówiąc o teksturach będziemy się posługiwali pojęciem teksela zamiast piksela. Piksel odnosić się będzie, tak jak dotychczas, do punktu w oknie monitora.

   (b) Tekstury w OpenGL mogą być:

      i. jednowymiarowe - reprezentowane przez obrazy wymiaru $n \times 1$ lub $1 \times m$ tekseli,

      ii. dwuwymiarowe - reprezentowane przez obrazy wymiaru $m \times n$ tekseli,

      iii. trójwymiarowe - reprezentowane przez obrazy przestrzenne wymiaru $m \times n \times k$ tekseli.

   (c) Tekstury mogą zastąpić kolor wierzchołka lub mieszać się z nim w określony sposób. W szczególności przy włączonym świetle tekstury mogą się mieszać z efektami świetlnymi.

   (d) Teksturowanie trzeba włączyć:

```
glEnable(GL_TEXTURE_1D)// tekstury jednowymiarowe
glEnable(GL_TEXTURE_2D)// tekstury dwuwymiarowe
glEnable(GL_TEXTURE_3D)// tekstury trójwymairowe
```

2. *Układ współrzędnych obrazu tekstury*

   Wyjaśnimy to pojęcie dla przypadku tekstury dwuwymiarowej (pozostałe dwa przypadki staną się oczywiste).

   Obraz wymiaru $m \times n$, będący teksturą dwuwymiarową jest w OpenGL przekształcany do układu współrzędnych ciągłych s, t na obszar kwadratu $[0, 1] \times [0, 1]$. Dodatkowo w każdym kolejnym kwadracie znajduje się kolejna kopia obrazu. (Rysunek **??**). Wtedy każdemu tekselowi odpowiada obszar prostokątny o wymiarach $\frac{1}{m} \times \frac{1}{n}$.

   Tekstury dwuwymiarowe są nakładane na powierzchnie wielokątów przez jednoznaczne przypisanie wierzchołkowi wielokąta pary współrzędnych (s, t). Ponieważ każda taka para oznacza punkt punkt należący do płaszczyzny R^2, zatem jest to punkt leżący we wnętrzu któregoś teksela. Zaletą przypisywania wierzchołkom współrzędnych ciągłych zamiast bezpośrednio tekseli obrazu jest fakt, iż uniezależniamy się od używanej rozdzielczości obrazu - podajemy w rzeczywistości *miejsce na obrazie*, które chcemy przypisać do danego wierzchołka. Ponadto możemy używać współrzędnych spoza zakresu $[0, 1]$, operując na obrazie złożonym z nieskończonej ilości kopii danego obrazu.

Przypisanie współrzędnych tekstury do wierzchołka dokonuje się automatycznie, w przypadku gdy dany typ teksturowania jest włączony oraz przed definicją wierzchołka istnieją jakieś współrzędne (bieżące jak wszystko w OpenGL) tekstury. Definicję współrzędnych dwuwymiarowych wykonuje się funkcją

```
glTexCoord2*(s,t);
```

gdzie * ma znaczenie jak zwykle.

3. *Obiekt tekstury*

   Struktura programistyczna zwana w OpenGL obiektem tekstury zawiera w ogólności dwie części:

   (a) **dane graficzne - obraz**

   (b) **parametry określające sposób używania danych graficznych**.

   Obiekt taki jest identyfikowany przez wskaźnik będący liczbą całkowitą. Typowy schemat definiowania obiektu tekstury jest następujacy:

   - **Faza 1**: wygenerowanie wskaźnika dla nowego obiektu,

     ```
     int texture;

     glGenTextures(1,&texture);
     ```

     Pierwszy parametr oznacza, że generujemy pojedynczy obiekt. W ogólności możemy za jednym razem generować tablicę obiektów:

     ```
     int textures[5];

     glGenTextures(5,textures);
     ```

   - **Faza 2**: określenie tego obiektu jako aktualnie obowiązującego (OpenGL jest maszyną stanów !)

     ```
     glBindTexture(texture);
     ```

   - **Faza 3**: przypisanie obiektowi danych graficznych i parametrów - zobacz dwa kolejne punkty

4. *Dane graficzne*

   Ponownie użyjemy przypadku tekstur dwuwymiarowych dla zobrazowania zagadnienia.

Dany obraz wczytujemy jako teksturę na dwa sposoby. Pierwszy z nich polega na użyciu funkcji

```
glTexImage2D(GL_TEXTURE_2D, poziom dokladnosci, format wewnetrzny,
             szerokosc,wysokosc, brzeg,
             format, typ skladowych koloru,
             wskaznik na dane graficzne);
```

gdzie

- `szerokosc, wysokosc` określają wymiary obrazu tekstury - powinny być potęgami cyfry 2, tzn. np. 256×512 (w wyższych wersjach OpenGL dopuszczalne jest ominięcie tego warunku)
- `poziom dokladnosci` oznacza numer tzw. *bitmapy* (zob. niżej) - jeżeli inie używamy bitmap, to wartość tego parametru wynosi 0,
- `format wewnetrzny` oznacza format danych graficznych (ostatni parametr), czyli obrazu który będzie używany jako tekstura - oznaczamy go przez stałe, np. `GL_RGB, GL_RGBA, GL_R, GL_G, GL_B, GL_ALPHA` - zob. też dokumentacja,
- `brzeg` musi mieć wartość 0 lub 1 - dopóki nie używamy intencjonalnie tego parametru (zrobimy to na kolejnych zajęciach) należy używać wartości 0,
- `format` określa format piksela, np. `GL_RGB, GL_RGBA, GL_R,` i jednocześnie związany z nim sposób konwersji do zwykle czterobajtowego formatu koloru w oknie,
- `typ skladowych koloru` oznacza ilość pamięci na składową, np. `GL_UNSIGNED_BYTE`.

Drugi możliwy sposób wczytywania danych graficznych polega na użyciu funkcji wczytującej obraz o określonej rozdzielczości (np. 512×512) i automatycznym wygenerowaniu dla niego wersji o mniejszych rozdzielczościach - w przypadku gdy bazowy obraz ma wymiary 512×512, generowane są (w pamięci) obrazy 256×256, 128×128, ..., 4×4, 2×2 i 1×1. Oznaczane są one przez liczby całkowite określające poziom danej bitmapy - najdokładniejsza ma poziom 0. Używając parametru `poziom dokladnosci` w funkcji `glTexImage2D()` można to zrobić ręcznie przypisując dane graficzne dla każdego poziomu. Obrazy te są używane automatycznie przez bibliotekę w zależnosci od tego jak duży jest dany wielokat na rzutni w scenach animowanych.

```
gluBuild2DMipmaps(GL_TEXTURE_2D, ilosc skladowych tekstury,
                  szerokosc, wysokosc, format,
                  typ danych, dane graficzne);
```

5. *Parametry tekstury*

Omówimy na razie tylko kilka najważniejszych parametrów.

(a) *Filtracja*

- *przypadek bez bitamp* - rozpatrzmy wielokąt 3×3 piksele i nałóżmy na niego teksturę 2×2 teksele. Wtedy oczywiście danemu pikselowi przypada w teksturze obszar mniejszy niż pole teksela (Rysunek **??**), przy czym np. lewemu, górnemu pikselowi przypada obszar należący do jednego (lewgo, górnego) teksela, natomiast środkowemu pikselowi przypada obszar należący częściowo do wszystkich tekseli. W drugim przypadku musimy się zdecydować w jaki sposób wybrać z tekstury kolor dla środkowego piksela. Można to zrobić np. przez arbitralne wybranie jednego z czterech kolorów, lub też przez ich uśrednienie. Podobny problem występuje, gdy nakładamy teksturę 3×3 na wielokąt 2×2. Określenie sposobu wyboru kolorów w teksturze w takich sytuacjach dokonujemy przez wywołanie funkcji `glTexParameter*`:
- *przypadek bez bitamp*

```
//pole piksela większe niż pole teksela, wybór bezpośredni
glTexParameter(GL_TEXTURE2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

//pole piksela większe niż pole teksela, uśrednianie
glTexParameter(GL_TEXTURE2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

//pole piksela mniejsze niż pole teksela, wybór bezpośredni
glTexParameter(GL_TEXTURE2D, GL_TEXTURE_MAX_FILTER, GL_NEAREST);

//pole piksela mniejsze niż pole teksela, wybór bezpośredni
glTexParameter(GL_TEXTURE2D, GL_TEXTURE_MAX_FILTER, GL_LINEAR);
```

- *przypadek z bitamapami*
  W tej sytuacji możemy jeszcze dodatkowo dokonywać mieszania (lub bezpośredniego wyboru) pomiędzy dwoma najbliższymi bitmapami

```
//pole piksela większe niż pole teksela, wybór bezpośredni
//dla piksela i dla bitmap
glTexParameter(GL_TEXTURE2D, GL_TEXTURE_MIN_FILTER,
               GL_NEAREST_MIPMAP_NEAREST);

//pole piksela większe niż pole teksela, wybór bezpośredni
//dla piksela i uśrednianie dla bitmap
glTexParameter(GL_TEXTURE2D, GL_TEXTURE_MIN_FILTER,
```

```
                GL_NEAREST_MIPMAP_LINEAR);

//pole piksela większe niż pole teksela, uśrednianie
//dla piksela i wybór bezpośreni dla bitmap
glTexParameter(GL_TEXTURE2D, GL_TEXTURE_MIN_FILTER,
                GL_LINEAR_MIPMAP_NEAREST);

//pole piksela większe niż pole teksela, uśrednianie
//dla piksela i dla  bitmap
glTexParameter(GL_TEXTURE2D, GL_TEXTURE_MIN_FILTER,
                GL_LINEAR_MIPMAP_LINEAR);
```

(b) *Powielanie lub obcinanie tekstury*

Specyfikując współrzędne spoza zakresu $[0, 1]$ możemy na wielokąt odwzorować kilka kopii obrazu. Domyślnie OpenGL działa właśnie tak (`GL_REPEAT`). Można jednak zmienić to zachowanie tak, że na wielokącie (jego fragmencie) znajdzie się tylko przydzielony obszar z zakresu $[0, 1]\times[0, 1]$, a poza nim będzie powielony brzegowy wiersz i brzegowa kolumna, np. (`GL_CLAMP`).

```
//obcinanie w kierunku t
glTexParameter(GL_TEXTURE2D, GL_TEXTURE_WRAP_T, GL_CLAMP);

//powielanie w kierunku s
glTexParameter(GL_TEXTURE2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
```

(c) *Zastępowanie koloru lub mieszanie*

Wspomnieliśmy już że tekstura może zastąpić kolor lub w jakiś sposób się z nim zmieszać. Określenie tego dokonuje się przez wywołanie funkcji

```
glTexEnv*v(GL_TEXTURE_ENV, tryb, parametr);//parametry wektorowe
glTexEnv*(GL_TEXTURE_ENV, tryb, parametr);//parametry liczbowe
```

Jeżeli `tryb` ma postać `GL_TEXTURE_ENV_MODE`, to `parametr` może przyjmować postać

- `GL_REPLACE`
- `GL_DECAL`
- `GL_MODULATE`
- `GL_BLEND`
- `GL_ADD`
- `GL_COMBINE`

Zachowanie zależy od tego ile składowych koloru ma tekstura: np. dla `GL_RGB` mamy (szczegóły są w dokumentacji funkcji `glTexEnv()`):

- `GL_REPLACE` - zastąpienie koloru
- `GL_DECAL` - zastąpienie koloru
- `GL_MODULATE` - zmieszanie przez przemnożenie składowych koloru obowiązującego gdy teksturowanie jest wyłączone z kolorem tekstury (w szczególności pozwala oświetlać powierzchnie teksturowane)
- `GL_BLEND` - kolor tekstury używany jest jako współczynnik wagowego mieszania koloru obowiązującego przy wyłączonym teksturowaniu i specjalnego koloru zdefiniowanego przez wywołanie

  ```
  glTexEnv*v(GL_TEXTURE_ENV, GL_TEXTURE_ENV_COLOR, kolor);
  ```
- `GL_ADD` - zmieszanie przez zsumowanie składowych koloru obowiązującego gdy teksturowanie jest wyłączone z kolorem tekstury
- `GL_COMBINE` - wynik zależy od wartości trzeciego parametru wywołania

  ```
  glTexEnv*v(GL_TEXTURE_ENV, GL_COMBINE_RGB, parametr);
  ```

  Przykładowo, gdy `parametr=GL_SUBTRACT`, to zachodzi zmieszanie przez odjęcie składowych koloru obowiązującego gdy teksturowanie jest wyłączone od koloru tekstury

Uwaga: Zmiana parametrów tekstury musi odbywać się poza zakresem funkcji `glBegin()...glEnd()`.

6. *Stos tekstury*

Współrzędne każdej tekstury są automatycznie mnożone przez macierz znajdującą się na *stosie tekstur*, który jest identyfikowany przez stałą `GL_TEXTURE`. Obowiązują na nim dokładnie takie same zasady jak na pozostałych dwóch stosach, czyli współrzędne tekstury mogą być przesuwane, obracane, skalowane lub, w ogólnosci, podawane dowolnym przekształceniom macierzowym.

Uwaga: Zmiana stosu i jego macierzy musi odbywać się poza zakresem funkcji `glBegin()...glEnd()`.

7. *Typowy schemat posługiwania się teksturami*

- **Faza przygotowania obiektów tekstury** - zwykle robimy to w funkcji dodatkowej (typu `init()`), wywoływanej przed uruchomieniem rysowania sceny - definiujemy kolejne obiekty każdy wg schematu z punktu 3.
- **Używanie tekstur**

– włączenie danego typu teksturowania (punkt 1.) - zwykle w funkcji rysującej scenę

– włącznie jako bieżącego któregoś obiektu tekstury - w funkcji rysującej scenę, przed kodem danej figury geometrycznej, wywołujemy ponownie `glBindTexture(obiekt)`.

8. *Zadanie:*

W scenie zawierającej oświetlony pokój ze stołem i krzesłami przypisać tekstury drewna do stołu, tekstury metalu do krzeseł, tekstury parkietu do podłogi (wykorzystać stos tekstury do powielania wzoru parkietu). Ponadto zdefiniować obraz na scianie i przypisać mu teksturę dowolnej reprodukcji malarstwa.