Tekstury - część 2

1. *Automatyczne generowanie współrzędnych tekstury*

   Jasne jest, że w przypadku obiektów ze skomplikowaną geometrią (są one w OpenGL przybliżane przez układy wielokątów, których jest zwykle dużo) jawne przypisywanie współrzędnych tekstury do wierzchołka (funkcja `glTexCoord2d()`) jest nierealne. Zamiast tego włącza się mechanizm automatycznego generowania współrzędnych tekstury dla każdego wierzchołka obiektu. Używamy do tego funkcji

   ```
   glTexGen*(coord, pname, param)
   ```

   lub jej wersji wektorowej `glTexGen*v(coord, pname, param)` (`*` ma zwykłe znaczenie).

   Argument `coord` przyjmuje wartość określającą, dla której współrzędnej definiujemy aktualnie sposób generowania - przyjmuje on jedną z wartości `GL_S, GL_T, GL_R, GL_Q`.

   Argument `pname` przyjmuje jedną z trzech wartości: .

   (a) `GL_TEXTURE_GEN_MODE` - wówczas argument `param` przyjmuje jedną z 5 wartości
      (1) `GL_OBJECT_LINEAR`
      (2) `GL_EYE_LINEAR`
      (3) `GL_SPHERE_MAP`
      (4) `GL_REFLECTION_MAP`
      (5) `GL_NORMAL_MAP`

   (b) `GL_OBJECT_PLANE`

   (c) `GL_EYE_PLANE`

   W przypadku a)(1) definiowana współrzędna tekstury w danym wierzchołku (X, Y, Z, W) (współrzędne jednorodne) jest określona przez równanie

   $$A\,X + B\,Y + C\,Z + D\,W, \tag{1}$$

   przy czym liczby A, B, C, D są zdefiniowane jako wektor 4-wymiarowy `wektor` i przekazane przez wywołanie (przypadek b))

   ```
   glTexGen*v(coord, GL_OBJECT_PLANE, wektor).
   ```

Liczby A, B, C, D określają pewną płaszczyznę, przy czym jeżeli $A^2 + B^2 + C^2 = 1$, to równanie (1) wyraża odległość wierzchołka (X, Y, Z, W) od tej płaszczyzny.

Przypadek a)(2) różni się tylko tym, że w obu powyższych wywołanaich w miejsce `OBJECT` wstawiamy `EYE`. Oznacza to, że działamy podobnie, ale we współrzędnych obserwatora, zamiast współrzędnych obiektu. Bardzo prosty przykład użycia tekstury jednowymiarowej do tworzenia warstwic zawarty jest w pliku `ogl82.cpp`.

Pozostałe przypadki proszę przeczytać w dokumentacji.

2. *Tekstury wielokrotne*

Tekstury wielokrotne pojawiły się w wersji 1.2 biblioteki OpenGL - istnieje osiem warstw tekstur, kodowanych przez stałe `GL_TEXTURE0,...,GL_TEXTURE7`. Definiując parametry danej warstwy (zwykle przy pomocy obiektu tekstury) musimy uczynić daną warstwę aktywną - np. dla warstwy zerowej:

```
glActiveTexture(GL_TEXTURE0);
```

Po tym, w fazie definicji, zwykle wywołujemy dwie komendy: pierwsza włącza dany typ teksturowania w warstwie, np.

```
glEnable(GL_TEXTURE_2D);
```

druga podpina wcześniej zdefiniowany obiekt tekstury do danej warstwy:

```
glBindTexture(GL_TEXTURE_2D, obiekt);
```

Mieszanie (o ile występuje) danych graficznych warstwy i-tej, wykonywane jest z efektem mieszania warstw poprzednich: $0, \ldots, i-1$. Sposób mieszania jest określony, podobnie jak w przypadku mieszania tekstur jednokrotnych z kolorem, czyli przez funkcję `glTexEnvi()` (zob. plik `opengl7.pdf`).

Każda warstwa tekstury ma swój *stos tekstur*, przy czym przejście na stos danej warstwy i załadowanie tam jakiś macierzy wymaga oczywiście uczynienia danej warstwy aktywną.

Współrzędne tekstury danej warstwy przypisywane są do wierzchołka przez funkcję (przypadek tekstur dwuwymiarowych)

```
glMultiTexCoord2d(warstwa,s,t);
```

np. `glMultiTexCoord2d(GL_TEXTURE0,0,0)`. Analogiczne funkcje istnieją dla tekstur jedno i trójwymiarowych.

3. *Teksturowanie kwadryk*

   Kwadryki stanowią specjalny rodzaj obiektu pozwalających rysować stożk, walce, dyski i kule. Tworzenie tych obiektów rozpoczynamy od stworzenia *obiektu kwadryki*:

   ```
   GLUquadricObj *obj;
   obj= gluNewQuadric();
   ```

   Następnie wypełniamy ten obiekt parametrami. Służą do tego następując funkcje

   ```
   gluQuadricDrawStyle(obiekt, styl);
   gluQuadricNormals(obiekt,normalne);
   gluQuadricOrientation(obiekt,orientacja);
   gluQuadricTexture(obiekt,tekstura);
   ```

   Argument `styl` przyjmuje następujące wartości: `GLU_FILL` (wypełnianie), `GLU_LINE` (siatka),`GLU_FILL` (zewnętrzny kontur), `GLU_POINTS` (zbiór punktów).

   Argument `normalne` przyjmuje następujące wartości: `GLU_NONE` (brak normalnych), `GLU_FLAT` (wspólna normalna dla wierzchołków danego wielokąta), `GLU_SMOOTH` (normalne generowane w każdym wierzchołku i interpolowane).

   Argument `orientacja` przyjmuje wartości `GLU_INSIDE` (normalne do środka) lub `GLU_OUTSIDE` (normalne na zewnątrz)

   Argument `tekstura` przyjmuje wartość 0 (brak teksturowania) lub 1 (automatyczne generowanie współrzędnych tekstury).

   Geometrię kwadrygi określa wywołanie jednej z funkcji

   ```
   gluCylinder(), gluDisk(), gluPartialDisk(), gluSphere()
   ```

   Dokładny opis ich parametrów (naturalnych) znajduje się w dokumentacji.

4. *Przezroczystość tekstur*

   Przezroczystość w OpenGL w najprostszy (dalece niedoskonały) sposób można zrealizować przy pomocy mieszania kolorów (tzw. *blending*), przy czym jawnie jest wtedy używana czwarta składowa koloru (dotychczas domyślnie równa 1).

   Najpierw musimy określić w jaki sposób będziemy mieszali dotychczsowe tło z nowym obiektem (wielokątem). Służy do tego funkcja

   ```
   glBlendFunc(nowy, dotychczasowy);
   ```

gdzie argumenty `nowy` i `dotychczasowy` określają ile procent koloru odpowiednio z nowego obiektu i z tła zostanie wzięte. Przykładowo jeżeli te wartości wynoszą (0,1), to mamy przypadek całkowitej nieprzezroczystości nowego obiektu,a gdy są równe (0.6,0.4) to obiekt jest w 40% nieprzezroczysty, tzn. przenika przez niego 60% koloru tła. Jednym ze sposobów przekazania wskaźnika nieprzezroczystości obiektu jest czwarta składowa geometryczna, nazywana zwykle *alfa*. Wtedy poprawny liniowy model przezroczystości pojedynczego wielokąta można zrealizwować przez wywołanie

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

Oczywiście najpierw muszą zostać wyrenderowane wszystkie obiekty nieprzezroczyste, a później przezroczyste. W przypadku scen animowanych zawierających przezroczyste bryły powyższy model się psuje, gdyż aby działał poprawnie trzeba dorysowywać do nieprzezroczystego tła wielokąty przezroczyste w kolejności od najdlaszego od obserwatora (animacja zmienia tą kolejność). Aby uniknąć zmian wizualnych zwykle stosuje się nadreprezentację tła:

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE);
```

Przykład znajduje się w pliku `ogl83.cpp`.

Jeżeli przygotujemy tekstury w trybie RGBA, to w szczególności możemy przezroczystość zrealizować przez przekazanie jej poziomu w czwartej współrzędnej tekstury, mogąc dodatkowo ustalić różny poziom przezroczystości w różnych tekselach tekstury.

5. *Zadania*

   (a) Zrealizować okno w pokoju przy pomocy tekstury częściowo przezroczystej.

   (b) Przetestować automatyczne teksturowanie kwadryk.

   (c) Nałożyć teksturę dwuwymiarową na kulę za pomocą płaszczyzny relacyjnej.