Instrukcja laboratoryjna **4**	**Grafika Komputerowa 3D**
	Temat: Transformacje 3D
	Przygotował: mgr inż. Maciej Lasota

1) Transformacje 3D

Przekształcenia dostępne w OpenGL umożliwiają rzutowania trójwymiarowych współrzędnych na dwuwymiarowy ekran. Przekształcenia te umożliwiają również obracanie obiektów, przesuwanie, rozciągnie, kurczenie oraz zwijanie. W OpenGL zamiast bezpośrednio modyfikować obiekty dokonujemy przekształcenia układu współrzędnych, w którym te obiekty się znajdują . W celu dokonania podstawowych przekształceń (translacji, skalowania, rotacji) obiektu musimy modyfikować macierz modelowania (widoku modelu). W bibliotece OpenGL dostępne są gotowe funkcje umożliwiające taką modyfikację.

1.1) Translacja

Translacja polega na przesunięciu zadanego obiektu o wektor przesunięcia. Do przeprowadzenia translacji służy funkcja.

```
void glTranslatef(GLfloat x, GLfloat y, GLfloat z)
void glTranslated(GLdouble x, GLdouble y, GLdouble z)
```

Funkcja mnoży bieżąca macierz modelowania (widoku modelu) przez macierz translacji. Jako parametry przyjmuje składowe wektora przesunięcia na poszczególnych osiach układu współrzędnych. Nowa macierz staje się bieżącą macierzą modelowania.

1.2) Skalowanie

Skalowanie polega na zmianie rozmiarów zadanego obiektu o wektor skalowania. Do przeprowadzenia skalowania służy funkcja.

```
void glScalef(GLfloat x, GLfloat y, GLfloat z)
void glScaled(GLdouble x, GLdouble y, GLdouble z)
```

Funkcja mnoży bieżąca macierz modelowania (widoku modelu) przez macierz skalowania. Nowa macierz staje się bieżącą macierzą modelowania. Funkcja glScale tak naprawdę nie skaluje obiektów, ale powoduje skalowanie układu współrzędnych.

1.3) Rotacja

Rotacja polega na obrocie obiektu o zadany kąt wokół wektor obrotu. Do przeprowadzenia rotacji służy funkcja.

```
void glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z)
void glRotated(GLdouble angle, GLdouble x, GLdouble y, GLdouble z)
```

Funkcja mnoży bieżącą macierz modelowania (widoku modelu) przez macierz obrotu. Kąt obrotu określany jest w kierunku przeciwnym do ruchu wskazówek zegara, sam obrót dokonywany jest wokół osi wyznaczonej przez wektor o początku w układzie współrzędnych i końcu w punkcie (x,y,z). Nowa macierz staje się bieżącą macierzą modelowania.

Funkcja glRotate tak naprawdę nie obraca obiektów, wykonuje ona obrót układu współrzędnych wokół zdefiniowanego wektora.

2) Tryb wypełniania

Tryb wypełniania wielokątów umożliwia decydowanie o tym, jak ma być wypełniona przednia, a jak tylna strona wielokąta. Do ustawia trybu wypełniania wielokątów służy funkcja.

```
void glPolygonMode(GLenum face, GLenum mode)
```

Jej pierwszy parametr określa, których ścian będzie dotyczyć regulacja (*GL_FRONT, GL_BACK, GL_FRONT_AND_BACK*). Drugi parametr określa styl wypełniania ścian wielokąta (*GL_LINE, GL_POINT, GL_FILL*). Domyślnie jest ustawiony *GL_FILL* czyli wypełnianie całego wielokąta. *GL_LINE* oznacza że będą widocznie jedynie linie tzw. tryb wireframe, *GL_POINT* oznacza, że będą widoczne jedynie punkty wielokąta.

3) Widoczność i ukrywanie krawędzi

Często niepotrzebne jest rysowanie wewnętrznych ścian obiektów, rysowanie takie zajmuje jedynie dodatkową moc obliczeniową komputera. OpenGL umożliwia automatyczne usuwanie niewidocznych ścian. Aby ukryć niewidoczne ściany należy włączyć opcję ukrywania niewidocznych powierzchni wielokątów za pomocą funkcji glEnable z parametrem *GL_CULL_FACE*.

Następnie należy wywołać funkcję **glCullFace**.

`void glCullFace(GLenum face)`

Funkcja ta przyjmuje jeden z trzech parametrów *GL_FRONT* (ściana przednia), *GL_BACK* (ściana tylna), *GL_FRONT_AND_BACK* (ściana przednia i tylna).

Ukrywanie krawędzi wykorzystywane jest w przypadku rysowania obiektów w trybie tzw. *wireframe (siatki)*. Do ukrywani krawędzi służy specjalna funkcja OpenGL.

`void glEdgeFlag(GLenum flag)`

Funkcja ta przyjmuje jeden z dwóch parametrów *GL_TRUE* (rysuj krawędź), *GL_FALSE* (nie rysuj krawędzi).

Przykład:

```
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);

glBegin(GL_POLYGON);
      glColor3f(0.5f, 0.5f, 0.5f);
      glEdgeFlag(GL_FALSE);
      glVertex3f(0.0f, 2.0f, 0.0f);
      glEdgeFlag(GL_TRUE);
      glVertex3f(2.0f, 2.0f, 0.0f);
      glVertex3f(2.0f, -2.0f, 0.0f);
      glVertex3f(-4.0f, -2.0f, 0.0f);
      glVertex3f(-5.0f, 0.0f, 0.0f);
glEnd();
```