

GRAFIKA KOMPUTEROWA

OpenGL w środowisku MS Windows

1. Definiowanie kształtu obiektu

OpenGL pozwala na definiowanie list poleceń, opisujących kształty poszczególnych obiektów. Każda lista ma swój numer, w programie przykładowym zostały one powiązane z nazwami obiektów za pomocą typu wyliczeniowego `GLDisplayListNames`. Przykłady list użytych do budowy sceny zawiera funkcja `CreateMaze()`.

Lista ma następującą strukturę:

```
glNewList(identyfikator_listy, GL_COMPILE);
    ... instrukcje
glEndList();
```

W programie przykładowym używamy list składających się z definicji czworokątów, np:

```
glBegin(GL_QUADS);
    glNormal3d( 1.0, 0.0, 0.0);    // normalna
    glVertex3d( 1.0, -1.0, 1.0);  // współrzędne wierzchołków
    glVertex3d( 1.0, 1.0, 1.0);
    glVertex3d( 1.0, 1.0, -6.0);
    glVertex3d( 1.0, -1.0, -6.0);
glEnd();
```

W programie przykładowym co 30ms wywoływany jest meldunek `WM_PAINT`. Obsługa tego meldunku zawiera wywołanie funkcji `RenderScene()`. W tej funkcji z kolei uruchamiane są instrukcje zawarte w zdefiniowanych wcześniej listach:

```
glCallList(identyfikator_listy);
```

2. Operacje na macierzy modelu

Obrót modelu można uzyskać przez użycie funkcji

```
void glRotated(
    GLdouble angle,    // kąt obrotu w stopniach
    GLdouble x,        // wektor, wokół którego przeprowadza się obrót
    GLdouble y,
    GLdouble z
);
```

Translację modelu uzyskujemy przy użyciu funkcji

```
void glTranslated(
    GLdouble x,        // składowe wektora translacji
    GLdouble y,
    GLdouble z
);
```

Macierze modelu można przechowywać na stosie. Dzięki temu można tworzyć oddzielne macierze dla poszczególnych elementów obiektu i uzyskiwać ruch jednego elementu względem drugiego, np.

```
glCallList(element1);    // narysowanie 1. elementu
```

```
glPushMatrix();           // odłożenie aktualnej macierzy modelu na stos
glRotated(90, 0, 1, 0); // obrócenie macierzy modelu o 90 stopni wokół OY
glCallList(element2);    // narysowanie 2. elementu, obróconego względem
                          // pierwszego o 90 stopni
glPopMatrix();           // odzyskanie pierwotnej macierzy modelu
```

3. Poruszanie się obserwatora w układzie współrzędnych

Zmiane położenia obserwatora w układzie współrzędnych uzyskujemy za pomocą funkcji

```
void gluLookAt(
    GLdouble eyex,           // położenie obserwatora
    GLdouble eyey,
    GLdouble eyez,
    GLdouble centerx,       // punkt, na który patrzymy
    GLdouble centery,
    GLdouble centerz,
    GLdouble upx,           // wektor wskazujący kierunek pionowy [0, 1, 0]
    GLdouble upy,
    GLdouble upz
);
```