Obiekty 3D

Wprowadzenie

Tworząc jakąkolwiek scenę 3D, od najprostszej po najbardziej skomplikowaną, używamy obiektów złożonych przede wszystkim z podstawowych, elementarnych obiektów, takich jak punkty, linie, trójkąty czy czworokąty. Od tego, jaki rodzaj obiektów podstawowych użyjemy, zależy ile i w jaki sposób będziemy definiować wierzchołków.

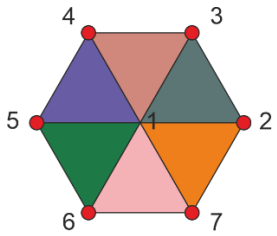
Definicję obiektu złożonego z obiektów podstawowych rozpoczyna się od wywołania funkcji glBegin z odpowiednim parametrem. Na przykład, jeśli chcemy rysować liniami, wywołanie będzie miało następującą postać:

```
glBegin(GL_LINES) // początek definicji obiektów
        glVertex3f(0.0f, 0.0f, 0.0f); // współrzędne wierzchołka
        glVertex3f(1.0f, 0.0f, 0.0f); // współrzędne wierzchołka
glEnd(); // koniec definicji obiektów
```

Polecenie glEnd kończy definicję obiektu. Oczywiście liczba zdefiniowanych wierzchołków glVertex ściśle uzależniona jest od rodzaju użytego obiektu podstawowego. Linia to dwa punkty, trójkąt to trzy itd. Ważne jest, aby pamiętać o kolejności podawania wierzchołków.

Wachlarze trójkątów – GL_TRIANGLE_FAN

GL_TRIANGLE_FAN – wprowadzane punkty powodują wygenerowanie trójkątów. Jednym z wierzchołków wszystkich tych trójkątów jest pierwszy podany punkt, dwa pozostałe to ostatni i przedostatni wprowadzony punkt. Liczba wierzchołków jest o dwa mniejsza od liczby punktów.



Trójkątów sklejanych można użyć np. do narysowania ścian bocznych i podstawy stożka. Przykładowy kod, który to realizuje jest przedstawiony poniżej.

```
glBegin(GL_TRIANGLE_FAN); // rysowanie ściany bocznej stożka
        glVertex3f(0.0f, 0.0f, 10.0f); // „czubek” stożka jest wspólnym wierzchołkiem
                                       //wszystkich trójkątów z wachlarza.
        for(angle = 0.0f; angle < (2.0f*GL_PI); angle += (GL_PI/8.0f))
        {
                x = 5.0f*sin(angle); // wyliczenie współrzędnych x i y kolejnego
                y = 5.0f*cos(angle); //wierzchołka
                if((iPivot %2) == 0) // wybieranie koloru - zielony lub czerwony
                        glColor3f(0.0f, 1.0f, 0.0f);
                else
                        glColor3f(1.0f, 0.0f, 0.0f);
                iPivot++; // inkrementacja zmiennej określającej rodzaj koloru
                glVertex2f(x, y); // def. kolejnego wierzchołka w wachlarzu trójkątów
}
glEnd(); // zakończenie rysowania trójkątów stożka
```

```
glBegin(GL_TRIANGLE_FAN); // rozpoczęcie rysowania kolejnego wachlarza
                                        //trójkątów zakrywającego podstawę stożka
        glVertex2f(0.0f, 0.0f); // środek wachlarza – początek układu współrzędnych
        for(angle = 0.0f; angle < (2.0f*GL_PI); angle += (GL_PI/8.0f))
        {
                x = 5.0f*sin(angle);
                y = 5.0f*cos(angle);
                if((iPivot %2) == 0)
                        glColor3f(0.0f, 1.0f, 0.0f);
                else
                        glColor3f(1.0f, 0.0f, 0.0f);
                iPivot++;
                glVertex2f(x, y);

        }

glEnd(); // zakończenie rysowania trójkątów stożka
```

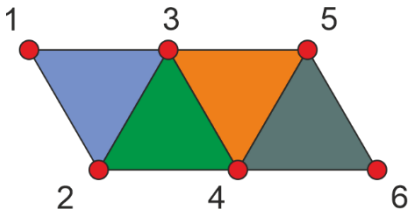
Aby powyższy kod zadziałał poprawnie, należy wcześniej zdefiniować zmienne, które są w nim używane i dołączyć bibliotekę cmath.

```
#define GL_PI 3.1415f

GLfloat x,y,angle; // przechowują wartości współrzędnych i kąta
int iPivot = 1; // do oznaczania zamiennych kolorów
```

Taśma trójkątowa – GL_TRIANGLE_STRIP

GL_TRIANGLE_STRIP – po wprowadzeniu dwóch pierwszych punktów, każdy kolejny punkt powoduje wygenerowanie trójkąta, którego wierzchołkami są: ten punkt i dwa ostatnie wprowadzone wcześniej.



Taśmy trójkątowej można użyć np. do narysowania powierzchni bocznej walca.

Ukrywanie powierzchni niewidocznych

Modelując obiekty na scenie nakazujemy bibliotece OpenGL rysowanie wszystkich elementów składających się na dany obiekt – zarówno tych widocznych, jak i tych, które w danym momencie nie są bezpośrednio widoczne dla obserwatora.
Możemy włączyć mechanizm ukrywania powierzchni niewidocznych, wówczas niewidoczne elementy nie biorą udziału w obliczeniach przy wyświetlaniu grafiki.
Włączenie ukrywania powierzchni niewidocznych odbywa się za pomocą wywołania funkcji glEnable z parametrem GL_CULL_FACE (wyłączanie analogicznie za pomocą funkcji glDisable).
W programie demo mechanizm ten jest włączony (w funkcji main).

```
// Włączenie lub wyłączenie mechanizmu eliminowania ukrytych powierzchni
if(iCull)
        glEnable(GL_CULL_FACE);
else
        glDisable(GL_CULL_FACE);
```

Testowanie głębi

Kolejny z mechanizmów wpływających na poprawne działanie zaprogramowanej sceny trójwymiarowej jest testowanie głębi. Testowanie głębi pozwala na właściwe rysowanie obiektów, tak aby elementy znajdujące się dalej i będące przesłonięte przez elementy znajdujące się bliżej nie były rysowane. Bez tego mechanizmu zawsze na wierzchu jest to, co było później narysowane. Do włączenia lub wyłączenia mechanizmu testowania głębi używamy funkcji glEnable lub glDisable z parametrem GL_DEPTH_TEST. W programie demo mechanizm ten jest włączony (w funkcji main).

```
// Włączenie lub wyłączenie mechanizmu sprawdzania głębi
if(iDepth)
	glEnable(GL_DEPTH_TEST);
else
	glDisable(GL_DEPTH_TEST);
```

Konieczne jest jeszcze użycie bufora głębi.

```
// Wyczyszczenie okna i bufora głębi
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Sposób rysowania obiektów

Biblioteka OpenGL pozwala nam na rysowanie obiektów na trzy różne sposoby: jako wypełnione obiekty, jako szkielet złożony z samych linii lub jako same punkty wskazujące kolejne wierzchołki.
Sposób rysowania obiektów ustala się za pomocą funkcji glPolygonMode z odpowiednimi parametrami: GL_BACK, GL_LINE; GL_BACK, GL_FILL; GL_BACK, GL_POINT. Pierwszy parametr ustala która część obiektu ma być rysowana w sposób określony za pomocą drugiego parametru. Przykładowo:

```
// Jeżeli ten znacznik będzie ustawiony, to wielokąty zwrócone
// tyłem do widza będą rysowane tylko w postaci szkieletu
if(iOutline)
	glPolygonMode(GL_BACK,GL_LINE);
else
	glPolygonMode(GL_BACK,GL_FILL);
```

Wypełnienie obiektów

W celu ustalenia sposobu wypełniania powierzchni wielokątów, biblioteka OpenGL udostępnia funkcję, która pozwala na wypełnianie wielokątów w sposób gładki (cieniowanie gładkie) lub w sposób płaski (cieniowanie płaskie).
Wypełnienie kolorem jednolitym ustala się za pomocą funkcji glShadeModel z parametrem GL_FLAT, natomiast wypełnienie umożliwiające płynne przejście kolorami od jednego wierzchołka do drugiego definiuje funkcja glShadeModel z parametrem GL_SMOOTH.

```
// Jeżeli ten znacznik będzie ustawiony, to wielokąty będą wypełniane jednolitym kolorem
// w przeciwnym razie nastąpi płynne przejście kolorów pomiędzy wierzchołkami
if(iFill)
	glShadeModel(GL_FLAT);
else
	glShadeModel(GL_SMOOTH);
```

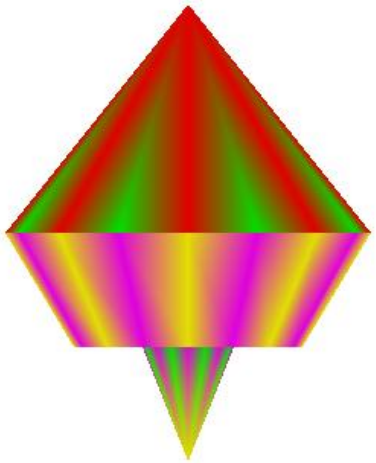
Nawinięcie wielokątów

Ważnym elementem podczas modelowania scen 3D jest nawinięcie wielokątów. Nawinięcie określa, czy ścianka obiektu jest zwrócona przodem do nas, czy też do wnętrza obiektu. Domyślnie nawinięcie w bibliotece OpenGL jest lewostronne, tzn. w kierunku przeciwnym do ruchu wskazówek zegara. Takie nawinięcie określa przód obiektu.
Aby zmienić domyślne nawinięcie, stosuje się funkcję glFront z parametrem GL_CW. Przywrócenie domyślnego nawinięcia to wywołanie funkcji glFront z parametrem GL_CCW.

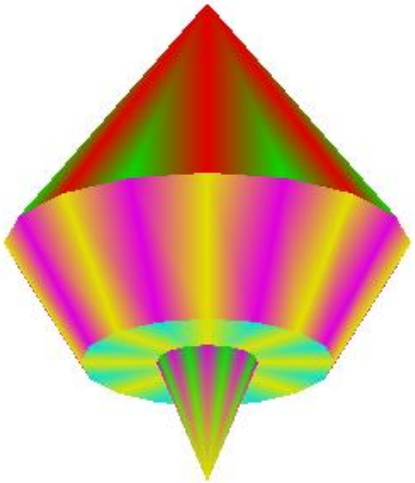
```
// Jeżeli ten znacznik będzie ustawiony, to domyślne nawinięcie zostanie zmienione //na zgodne z ruchem wskazówek zegara dla części przodem do nas,
//w przeciwnym razie powracamy do domyślnych //ustawień.
if(iClock)
        glFrontFace(GL_CW);
else
        glFrontFace(GL_CCW);
```

Zadanie do wykonania

Celem zajęć jest stworzenie sceny, w której powinien znaleźć się obiekt złożony z trójkątów sklejanych oraz wachlarzy trójkątów. Wygląd obiektu w różnych położeniach przedstawiają Rys. 1 i Rys. 2. Kolorystyka dowolna (przy zachowaniu różnych barw trójkątów w obrębie każdej części obiektu). Części niewidoczne powinny zostać ukryte i nie powinny brać udziału w obliczeniach – dodać możliwość włączania/wyłączania tego mechanizmu za pomocą klawiatury. Należy również zapewnić możliwość wyboru pełnego wypełnienia (cieniowanego i jednolitego) oraz rysowania tylko szkieletu.



Rys.1. Obiekt w położeniu nr 1



Rys. 2. Obiekt w położeniu nr 2

Poniżej kilka przykładów obiektów, jakie można stworzyć przy użyciu połączenia trójkątów sklejanych i wachlarzy trójkątów: