UNIWERSYTET
TECHNOLOGICZNO-HUMANISTYCZNY
im. Kazimierza Pułaskiego w Radomiu

WYDZIAŁ TRANSPORTU, ELEKTROTECHNIKI I INFORMATYKI

Kierunek: INFORMATYKA

Przedmiot: BIBLIOTEKA GRAFICZNA OPENGL

Autor: DR ARTUR HERMANOWICZ

Konfiguracja programu do pracy z OpenGL

1. Metody interfejsu GLEventListener

Najwygodniejszą metodą skorzystania z możliwości włączenia biblioteki OpenGL w Javie jest skorzystanie z interfejsu **GLEventListener**. Można to uczynić w następujący sposób:

```
public class p03 extends JFrame implements GLEventListener
```

W ten sposób tworzona klasa (tu p03) dziedziczy nie tylko właściwości klasy **JFrame**, ale również interfejsu **GLEventListener**. Uzyskujemy dostęp nie tylko do funkcjonalności biblioteki swing, ale również do biblioteki OpenGL.

Interfejs **GLEventListener** wymaga aby klasa miała zaimplementowane cztery metody: **display**, **dispose**, **init** oraz **reshape**.

Metoda **display** jest wywoływana każdorazowo, gdy generowana jest zawartość okna. Tu znajduje się kod, który jest wysyłany do karty graficznej. Oczywiście nie oznacza to, że nie można wykorzystywać podprogramów lub innych klas.

Metoda **dispose** służy do zwalniania zbędnych już zasobów. Mechanizm automatycznego zwalniania pamięci w Javie nie oznacza, że jest ona zbędna. Nie zapewni on na przykład zwolnienia bloku pamięci zarezerwowanej na karcie graficznej.

Metoda **init** wywoływana jest w celu wstępnego zainicjalizowania ustawień na początku wykonywania programu. Tu umieszczamy wszelkie wstępne ustawienia, jak np. dotyczące oświetlenia.

Metoda **reshape** wywoływane jest przy każdorazowej zmianie wymiarów okna. Ze względu na to, że ustawienia dotyczące ustawień perspektywy zależne są między innymi od wymiarów okna jest to dobre miejsce do wykorzystania w tym celu.

2. Konfiguracja widoku, perspektywy, kamery

Domyślnie w OpenGL ustawione jest rzutowanie 2D. W celu umożliwienia wykorzystania trzeciego wymiaru należy zmodyfikować domyślne ustawienia. Można tego dokonać w metodzie **reshape**. Umożliwi to dodatkowo aktualizację ustawień w przypadku gdyby wymiary okna miały ulegać zmianie w trakcie wykonywania programu.

Pierwszą rzeczą jest ustawienie widoku OpenGL, czyli wyznaczenie fragmentu obszaru klienta okna, za którego generowanie odpowiedzialna będzie biblioteka OpenGL:

```
gl.glViewport(0, 0, width, height);
```

Metoda **glViewport** jest bardzo prosta w użyciu. Przyjmuje cztery parametry oznaczające odpowiednio: lewą współrzędną lewego górnego rogu widoku (x), górną współrzędną lewego górnego rogu widoku (y), szerokość widoku, wysokość widoku. W tym przypadku będzie to obszar począwszy od punktu (0, 0) o szerokości **width** oraz wysokości **height**. Warto zwrócić uwagę, że parametry **width** i **height** są przekazywane przez metodę **reshape**.

Do skonfigurowania rzutowania perspektywicznego zostanie wykorzystana metoda **gluPerspective** z biblioteki narzędziowej GLU. Wymaga ona parametru opisującego stosunek szerokości do wysokości widoku. Prawidłowe jego wyliczenie jest niezbędne w celu zapewnienia właściwych proporcji generowanego obrazu, np. aby okrąg był okręgiem a nie elipsą. Obliczamy stosunek szerokości do wysokości (**aspect**), zabezpieczając się w prosty sposób przed błędem dzielenia przez zero:

```
if(height==0)
     height=1;
float aspect=(float)width/height;
```

Teraz można przystąpić do samego ustawiania perspektywy:

```
gl.glMatrixMode(GL2.GL_PROJECTION);
gl.glLoadIdentity();
glu.gluPerspective(45.0, aspect, 1.0, 10.0);
glu.gluLookAt(0.0f, 0.0f, 5.0f,
     0.0f, 0.0f, 0.0f,
     0.0f, 1.0f, 0.0f);
gl.glMatrixMode(GL2.GL_MODELVIEW);
```

OpenGL domyślnie ustawiony ma tryb macierzy **GL_MODELVIEW**. W celu zmiany ustawień rzutowania przełączamy pierwszą instrukcją tryb macierzy w **GL_PROJECTION**, a po zakończeniu ustawień przywracamy stan poprzedni wracając w tryb macierzy **GL_MODELVIEW** (ostatnia instrukcja).

Wszystkie przekształcenia mają charakter multiplikatywny, nakładają się na siebie. Aby wprowadzić nowe ustawienia należy „zresetować" macierz przekształceń metodą **glLoadIdentity**. Teraz można dokonać ustawienia perspektywy przy pomocy metody **gluPerspective**. Przyjmuje ona cztery parametry: kąt pola widzenia w kierunku pionowym, stosunek szerokości do wysokości widoku, tzw. płaszczyznę bliższą i dalszą obszaru widzenia. Dwa ostatnie parametry mają szczególnie duże znaczenie praktyczne. W powyższym kodzie to liczby 1 i 10. Oznacza to, że obiekty, które znajdują się odległości mniejszej niż 1 lub większej niż 10 od obserwatora nie zostaną wyświetlone. Czasami wystarczy nieco zwiększyć zasięg widzenia, aby zagubiony obiekt „odnalazł się". Nie warto jednak przesadzać z zakresem pola widzenia. Odległe obiekty w praktyce nie będą i tak widoczne lub będą rozmiarów piksela. Zezwalając na ich generowanie tracimy na wydajności aplikacji.

Kolejna metoda w powyższym kodzie (**gluLookAt**) umożliwia ustawienie kamery (obserwatora). Przyjmuje ona dziewięć parametrów. Trzy pierwsze to odpowiednio współrzędne x, y i z położenia kamery. Trzy następne to położenie punktu, na który skierowana jest kamera. Trzy ostatnie to tzw. wektor pionu i zawiera informację, gdzie jest „góra" kamery (obserwatora).

3. Oświetlenie w OpenGL i jego konfiguracja

Oświetlenie w OpenGL modelowane jest przy pomocy trzech jego rodzajów: otaczające, rozproszone, odbłysków.

Światło otaczające (ang. *ambient*) oświetla wszystkie obiekty równomiernie i nie ma zdefiniowanego kierunku. Jest to domyślne ustawienie oświetlenia w OpenGL. Stosowane samodzielnie nie umożliwia uzyskanie realistycznego obrazu, ale wspomaga ten efekt przy zastosowaniu innych rodzajów oświetlenia. W pewnym sensie jest to część oświetlenia, której nie jesteśmy w stanie precyzyjnie zdefiniować, a bierze udział w oświetleniu sceny jako światło pochodzące z nieznanych źródeł, odbijające się od obiektów itp.

Światło rozproszone (ang. *diffuse*) charakteryzuje się przede wszystkim tym, że ma zdefiniowany kierunek, pod którym pada na powierzchnie obiektu. Natężenie oświetlenia obliczane jest zgodnie z prawem Lamberta i jest proporcjonalne do kosinusa kąta pomiędzy wektorem oświetlenia (np. łączącym źródło światła z oświetlanym punktem) oraz wektorem normalnym (prostopadłym) do powierzchni w danym punkcie. W ten sposób obiekty, na które światło pada pionowo są jaśniej oświetlone, a im bardziej wektor oświetlenia odchylony jest od wektora normalnego (pionu) tym słabsze oświetlenie.

Światło odbłysków (ang. *specular*) to światło, które posiada kierunek i jest odbijane w jedną stronę. Typowym przykładem są refleksy świetlne (ostre, jasne plamy) na karoseriach samochodów w słoneczny dzień lub „zajączki" puszczane przez dzieci przy pomocy lusterek. Światło to umożliwia tworzenie bardziej realistycznych obrazów obiektów z twardych połyskliwych materiałów jak np. metal.

Z oświetleniem nierozerwalnie związane są właściwości materiału, z którego wykonany jest obiekt. Obliczenia wykonywane są w niezwykle prosty sposób. Kolor wynikowy to iloczyn składowa po składowej oświetlenia i właściwości materiału. Na przykład materiał o składowych RGB (1, 1, 0) oznacza, że jest to obiekt, który będzie obijał składowe czerwoną i zieloną światła a pochłaniał składową niebieską. Jeżeli oświetlimy go światłem o składowych RGB (1, 0, 1) to wygenerowany zostanie obiekt czerwony, gdyż natężenie oświetlenia wyniesie (1*1, 1*0, 0*1), czyli (1, 0, 0).

Poniższy kod przedstawia przykładowe ustawienia materiałów:

```
float matSpec[]={1.0f,1.0f,1.0f,1.0f};
gl.glMaterialfv(GL2.GL_FRONT, GL2.GL_SPECULAR, matSpec, 0);
gl.glMateriali(GL2.GL_FRONT, GL2.GL_SHININESS, 128);
gl.glEnable(GL2.GL_COLOR_MATERIAL);
gl.glColorMaterial(GL2.GL_FRONT, GL2.GL_AMBIENT_AND_DIFFUSE);
```

Pierwsze trzy instrukcje dotyczą ustawienia w jaki sposób obiekt będzie reagował na światło odbłysków. Zdefiniowaną w pierwszej instrukcji tablicę z wartościami odbijanego światła rozbłysków dla danego materiału ustawiamy metodą **glMaterialfv** w drugiej instrukcji. Parametr **GL_SHININESS** w instrukcji trzeciej steruje intensywnością rozbłysków.

Znacząco można uprościć ustawienia dla światła otaczającego i rozproszonego zezwalając OpenGL na pobieranie tych ustawień na podstawie koloru obiektu poprzez włączenie metodą **glEnable** opcji **GL_COLOR_MATERIAL**. W ten sposób tablice dla tych rodzajów oświetlenia będą automatycznie tworzone na podstawie tego co ustawione zostanie dla danego obiektu przy pomocy metody **glColor**.

Ostatnia instrukcja w powyższym kodzie oznacza, że ustawienia materiału dla światła otaczającego i rozproszonego mają dotyczyć przednich ścianek obiektów.

Mając ustawione właściwości materiału można pokusić się o włączenie oświetlenia:

```
gl.glEnable(GL2.GL_LIGHTING);
```

Warto jednakże wcześniej włączyć przynajmniej model światła otaczającego:

```
float ambientLight[]={0.1f,0.1f,0.1f,1.0f};
gl.glLightModelfv(GL2.GL_LIGHT_MODEL_AMBIENT, ambientLight, 0);
```

W powyższym przypadku oznacza to, że światło otaczające będzie miało natężenia 0,1 dla każdej składowej.

W OpenGL można zdefiniować osiem źródeł światła, oznaczonych od **GL_LIGHT0** do **GL_LIGHT7**:

```
float ambient[]={0.1f,0.1f,0.1f,1.0f};
float diffuse[]={0.5f,0.5f,0.5f,1.0f};
float specular[]={1.0f,1.0f,1.0f,1.0f};
float position[]={-2.0f,2.0f,3.0f,1.0f};

gl.glLightfv(GL2.GL_LIGHT0,GL2.GL_AMBIENT,ambient,0);
gl.glLightfv(GL2.GL_LIGHT0, GL2.GL_DIFFUSE, diffuse,0);
gl.glLightfv(GL2.GL_LIGHT0, GL2.GL_SPECULAR, specular,0);
gl.glLightfv(GL2.GL_LIGHT0, GL2.GL_POSITION, position,0);

gl.glEnable(GL2.GL_LIGHT0);
```

Dla każdego źródła światła zaczynamy od zdefiniowania tablic parametrów dla poszczególnych rodzajów oświetlenia z nim związanych. Takie źródło światła dodatkowo może mieć konkretne położenie w generowanej scenie (tablica **position**), co umożliwia symulowanie np. światła żarówki.

Sekwencja instrukcji **glLightfv** służy do przesłania zawartości wcześniej zdefiniowanych parametrów w odpowiednich tablicach.

Ostatnia instrukcja to włączenie źródła światła o identyfikatorze **GL_LIGHT0**.