helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

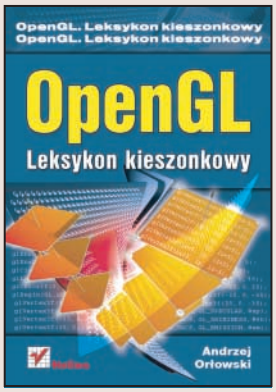
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

OpenGL. Leksykon kieszonkowy



Autor: Andrzej Orłowski
ISBN: 83-7361-968-2
Format: B6, stron: 160

Biblioteka OpenGL to narzędzie służące do tworzenia realistycznej grafiki trójwymiarowej w programach. Możliwości OpenGL możemy dziś podziwiać w grach komputerowych, animacjach, wizualizacjach i filmowych efektach specjalnych. Większość publikacji dotyczących biblioteki OpenGL przedstawia ją w kontekście języka C++, z poziomu którego faktycznie jest najczęściej wykorzystywana. Jednakże wielu programistów, którzy często rozpoczynają swoją naukę od Pascala i związanego z nim środowiska Delphi, próbuje stosować ją również w połączeniu z tym językiem.

Książka „OpenGL. Leksykon kieszonkowy" jest przeznaczona właśnie dla tych programistów. Opisuje bibliotekę OpenGL pod kątem zastosowania jej w aplikacjach tworzonych za pomocą Delphi. Przedstawia wszystkie funkcje służące do generowania i wyświetlania grafiki – począwszy od brył podstawowych, a skończywszy na krzywych NURBS i Be`ziera. Pokazuje sposoby oświetlania i teksturowania obiektów w scenie oraz metody importowania siatek stworzonych w programach 3D.

- Typy danych w OpenGL
- Metody rzutowania i wyświetlania sceny 3D na ekranie monitora
- Obiekty i bryły podstawowe
- Oświetlenie
- Umieszczanie napisów w scenie
- Korzystanie z map bitowych
- Materiały i tekstury
- Animacja
- Import obiektów 3D

Delphi to nie tylko aplikacje bazodanowe i internetowe. Przekonaj się, że w Delphi można również stworzyć realistyczną animację 3D.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Wstęp **5**

1. OpenGL dla Windows **7**
Typy Object Pascal i OpenGL 7
Przygotowanie środowiska pracy. Funkcje WGL 10
Aplikacja w oknie 13
Aplikacja pełnoekranowa 19
Komponent (VC) dla Delphi 19

2. Biblioteki OpenGL **23**
Biblioteka standardowa 23
Biblioteka GLU 59
Biblioteki GLUT 63

3. Metody zobrazowania **64**
Rzut prostokątny 64
Rzut perspektywiczny i kamera 67
Organizacja ekranu OpenGL 72

4. Obiekty podstawowe **74**
Punkt 74
Linia 78
Trójkąt 81
Czworokąt 83
Wielokąt 84

5. Bryły 86
Cylinder 87
Dysk 88
Sfera 91

6. Krzywe i powierzchnie 93
Krzywe i powierzchnie Beziera 93

7. Światło 108
Światło tła (otoczenia) 109
Położenie źródła światła w przestrzeni 110
Światło rozproszone 112
Reflektor 112
Rozmycie 113

8. Napisy 116

9. Obrazy w OpenG L 118
Mapy bitowe 118
Wczytywanie plików BMP 119
Wyświetlanie obrazów w formacie BMP 121
Kopiowanie fragmentu ekranu 124
Powiększenie 126

10. Materiał i tekstura 128
Materiał 128
Tekstura 133

11. Import siatek 140

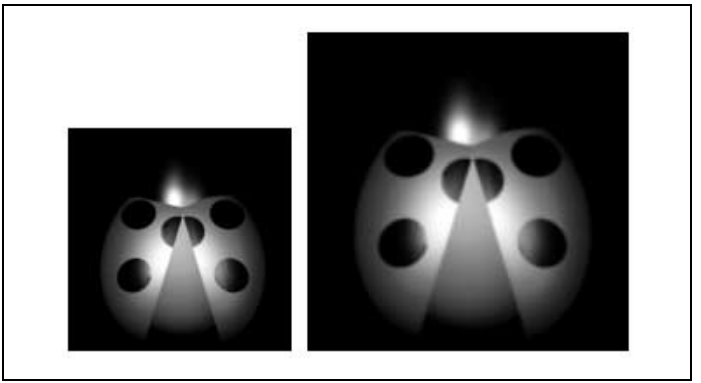
12. Ruch 148
Przemieszczenie 148
Obroty 150
Skalowanie 153

Skorowidz 154

Rozdział 4. Obiekty podstawowe

Punkt

Punkt stanowi najważniejszy element każdej rysowanej sceny. Wszystkie obiekty rysowane za pomocą procedur OpenGL są zbudowane z punktów. Punkty te mogą wyznaczać wierzchołki obiektów (linia, trójkąt) lub stanowić wypełnienie tych obiektów. Za pomocą punktów można również rysować wykresy funkcji matematycznych czy też dowolne figury ułożone z kolejnych punktów. Aby wyznaczyć położenie punktu i jego wielkość, wykorzystuje się specjalne procedury OpenGL. Bardzo ważnym zagadnieniem jest wzajemna zależność pomiędzy punktem rysowanym w OpenGL a pikselem. Piksel jest najmniejszym obszarem ekranu, jaki może przy rozdzielczości określonej parametrami danej karty graficznej w danej chwili być zapalony (zgaszony). Na rozmiar piksela programista nie ma żadnego wpływu (poza zmianą rozdzielczości ekranu). Punkt natomiast posiada rozmiar określony wielkością płaszczyzny rzutowania i zadeklarowaną wielkością. Domyślną wielkością punktu jest 1 (jeden), co nie oznacza wcale, że na ekranie będzie on odpowiadał jednemu pikselowi. W przypadku zmiany rozmiaru płaszczyzny rzutowania zmieni się również rozmiar wyświetlanego punktu o wielkości jednostkowej (może wynosić kilka lub kilkanaście pikseli). Zagadnienie to jest szczególnie istotne przy imporcie obrazków, jak również podczas wykorzystywania bitmap, których wymiary są określane w pikselach. Rezultat nieskorelowania wymiarów okna grafiki i płaszczyzny rzutowania przedstawia rysunek 4.1.



Rysunek 4.1. Zależność wymiarów wyświetlania obiektu, którego wymiary określane są w pikselach i w punktach

Obrazek z lewej strony został wykonany z wykorzystaniem procedury `glDrawPixels()`, tzn. narysowano go pikselami, natomiast ten z prawej strony jest teksturą nałożoną na kwadrat o takich samych wymiarach, ale określonych w punktach. Wymiary obrazka wynoszą 128 na 128 pikseli, wymiary kwadratu — 128 na 128 punktów. Okno grafiki określone procedurą `glViewPort()` ma wymiary 600 na 600 pikseli, a wymiary płaszczyzny rzutowania zdefiniowane procedurą `glOrtho()` wynoszą 400 na 400 punktów. Jak wynika z przedstawionego rysunku, proporcje pomiędzy rzeczywistymi rozmiarami rysunku a obiektem o tych samych wymiarach, ale wyrażonych w innych jednostkach, są oczywiste. Z powyższego wywodu wynika więc jasno, że piksel to nie to samo co punkt.

---

glPointSize()

Deklaracja: **`procedure`** `glPointSize(Grubość : GLFloat);`

Działanie: ustala nową wielkość rysowanych punktów.

Uwaga

Nowa wielkość rysowanych punktów będzie taka sama dla wszystkich punktów rysowanych po jej ustaleniu. Aby przywrócić domyślną wielkość punktu 1 (jeden), należy ponownie wywołać procedurę `glPointSize()`, z parametrem *Grubość* równym 1 (jeden).

```
glPointSize(5); // Punkt o wielkości 5 jednostek
glPointSize(1); // Powrót do wielkości domyślnej
```

glVertex*()

Deklaracja: **procedure** `glVertex2*(x, y, z, w : GL**[PGL**]);`

lub **procedure** `glVertex3*(x, y, z, w : GL**[PGL**]);`

lub **procedure** `glVertex4*(x, y, z, w : GL**[PGL**]);`

Uwaga

Znak `*` (gwiazdka) to jedno z: `d`, `f`, `i`, `s`, `dv`, `fv`, `iv` lub `sv`. Dwuznak `**` (dwie gwiazdki) to odpowiednio: `Double`, `Float`, `Int` lub `Short`.

Działanie: rysuje wierzchołek (punkt) na płaszczyźnie rzutowania, w miejscu określonym parametrami wywołania. Tworzenie takich obiektów jak linie, wielokąty itp. odbywa się za pomocą określenia położenia ich wierzchołków w przestrzeni. Cyfra przyrostka określa liczbę wymiarów w przestrzeni, w której będzie umieszczony wierzchołek. Litera *v* oznacza, że parametry do procedury będą przekazane w postaci tablicy (wektora). Najczęściej procedura ta jest wywoływana w postaci: `glVertex3f()`. Należy w tym miejscu wspomnieć, że liczby typu *Double* (*GLDouble*) i *Single* (*GLFloat*) są bez konwersji akceptowane przez koprocesor matematyczny (CPU) i z tego też względu należy z nich korzystać, oczywiście jeżeli inne względy nie

wymuszają stosowania innego typu liczb. W powyższym przykładzie uwzględniono procedurę `glPointSize()`, która określa wielkość rysowanych punktów na ekranie. Jeżeli nie wywołamy tej procedury, to wszystkie punkty będą miały domyślną wielkość równą 1 (jeden). Dla punktów, podobnie jak dla innych obiektów, można wykorzystać narzędzie do wygładzania krawędzi, czyli tzw. antialiasing. W razie zwiększenia rozmiarów punktu okaże się, że ma on kształt kwadratu, co nie zawsze jest pożądane. Aby tego uniknąć, wystarczy wywołać procedurę `glEnable()` z parametrem `GL_POINT_SMOOTH`, co spowoduje, że narożniki zostaną zaokrąglone. Należy pamiętać o wywołaniu procedury `glDisable(GL_POINT_SMOOTH)`, jeżeli kolejne punkty mają być rysowane bez wygładzania. Włączanie wygładzania dla punktów o domyślnej wielkości 1.0 jest bezcelowe i nie należy go stosować. Poniżej przedstawiono fragment kodu, którego wykonanie powoduje narysowanie punktu bez wygładzania i z wygładzaniem:

```
// Punkt
   glColor3f(0.0, 0.0, 1.0); // Kolor niebieski
// Punkt bez wygładzania
   glPointSize(95); // Wielkość punktu(95 pikseli);
   glBegin(GL_POINTS); // Rysowanie punktów
    glVertex3f(0, 0, 0); // Położenie punktu(Środek okna
                            grafiki OpenGL)
   glEnd; // Koniec rysowania
// Punkt wygładzony
   glEnable(GL_POINT_SMOOTH); // Włączenie wygładzania
                                 (antialiasing) punktu
   glPointSize(95);
   glTranslatef(100, 0, 0);
   glBegin(GL_POINTS);
    glVertex3f(0, 0, 0);
   glEnd;
   glDisable(GL_POINT_SMOOTH); // Wyłączenie wygładzania
                                  punktu
```

Rezultat działania powyższego fragmentu kodu ilustruje rysunek 4.2.

Rysunek 4.2. Punkt o wielkości 95 jednostek bez wygładzania i po włączeniu wygładzania

Uwaga

Domyślnie wygładzanie jest wyłączone, a to ze względu na bardzo znaczące spowolnienie pracy programu. Przed włączeniem wygładzania należy przeanalizować konieczność jego użycia i wyłączyć, jeżeli przestaje być faktycznie niezbędne.