Programowanie Grafiki Komputerowej

OpenGL - wprowadzenie do programowania grafiki komputerowej, transformacje 3D, oświetlenie i materiały, teksturowanie, dodatkowe efekty graficzne

mgr inż. Maciej Lasota

Katedra Informatyki, Politechnika Świętokrzyska w Kielcach

część 3



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



Politechnika Świętokrzyska

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

" Program Rozwojowy Potencjału Dydaktycznego Politechniki Świętokrzyskiej w Kielcach: *kształcenie w nowoczesnych obszarach techniki* "

Program Operacyjny Kapitał Ludzki, Priorytet IV Szkolnictwo wyższe i nauka, Działanie 4.1. Wzmocnienie i rozwój potencjału dydaktycznego uczelni oraz zwiększenie liczby absolwentów kierunków o kluczowym znaczeniu dla gospodarki opartej na wiedzy, Poddziałanie 4.1.1. Wzmocnienie potencjału dydaktycznego uczelni.

Umowa UDA–POKL.04.01.01-00395/09-00

Spis treści

1. Obsługa kolorów
   - Bufor kolorów oraz głębi
2. Rysowanie prymitywów
   - Rysowanie pojedynczego punktu
   - Rysowanie linii
   - Rysowanie pozostałych prymitywów
   - Tryb wypełniania
   - Widoczność i ukrywanie ścian
   - Przykład
3. Transformacje 3D
   - Translacja
   - Rotacja
   - Skalowanie
   - Przykład
4. Oświetlenie i materiały
   - Wektory normalne
   - Oświetlenie
   - Materiały
   - Śledzenie kolorów
   - Przykład
5. Bibliografia

Uwagi

- Animacje są uruchamiane, gdy plik PDF jest wyświetlany na pełnym ekranie.
- Animacja jest uruchamiana dopiero po naciśnięciu spacji (program Adobe Reader).
- Klatki animacji wyświetlane są co 3 sekundy.
- Symulacje są uruchamiane po kliknięciu w ikonę z odpowiednim systemem operacyjnym. W systemie Linux należy dodatkowo kliknąć dwukrotnie w nazwę programu.
- Do uruchamiania symulacji z pliku PDF, wymagany jest program Adobe Reader w wersji 5.0 lub nowszej.

Obsługa kolorów

Obsługą kolorów w bibliotece OpenGL zajmuje się rodzina funkcji **glColor**. Funkcje z tej rodziny mogą przyjmować *trzy* lub *cztery* argumenty.

Obsługa kolorów

Obsługą kolorów w bibliotece OpenGL zajmuje się rodzina funkcji **glColor**. Funkcje z tej rodziny mogą przyjmować *trzy* lub *cztery* argumenty.

Funkcje do obsługi kolorów

glColor3f(GLfloat R, GLfloat G, GLfloat B)
glColor4f(GLfloat R, GLfloat G, GLfloat B, GLfloat A)

Obsługa kolorów

W przypadku pierwszej funkcji kolor podaje się w postaci trzech składowych **RGB**, natomiast w przypadku drugiej funkcji oprócz koloru **RGB** podaje się czwarty argument będący składową **alfa**.

Bufor kolorów oraz głębi

W OpenGL wyróżnimy cztery rodzaje buforów.

Rodzaje buforów

- Bufor kolorów
- Bufor głębi
- Bufor szablonu (wzorca)
- Bufor akumulacyjny

Bufor kolorów oraz głębi

Najczęściej stosowane są *dwa pierwsze*. **Bufor kolorów** odpowiedzialny jest za prawidłowe wyświetlanie kolorów, natomiast **bufor głębi** odpowiedzialny jest za poprawne renderowanie obiektów w przestrzeni 3D.

Bufor kolorów oraz głębi

Najczęściej stosowane są *dwa pierwsze*. **Bufor kolorów** odpowiedzialny jest za prawidłowe wyświetlanie kolorów, natomiast **bufor głębi** odpowiedzialny jest za poprawne renderowanie obiektów w przestrzeni 3D.

Funkcja służąca do czyszczenia buforów

glClear(GLbitfield mask)

Bufor kolorów oraz głębi

Jako argument funkcja przyjmuje stałe określającą rodzaj bufora, który chcemy wyczyścić: **GL_COLOR_BUFFER_BIT**, **GL_DEPTH_BUFFER_BIT**, **GL_ACCUM_BUFFER_BIT**, **GL_STENCIL_BUFFER_BIT**.

Bufor kolorów oraz głębi

Jako argument funkcja przyjmuje stałe określającą rodzaj bufora, który chcemy wyczyścić: **GL_COLOR_BUFFER_BIT**, **GL_DEPTH_BUFFER_BIT**, **GL_ACCUM_BUFFER_BIT**, **GL_STENCIL_BUFFER_BIT**. Dodatkowo istnieje specjalna funkcja, która czyści jedynie **bufor kolorów** oraz ustawia go na wybrany kolor. Przyjmuje ona cztery argumenty określające kolor w formacie **RGBA** na jaki ma zostać wyczyszczony bufor.

Bufor kolorów oraz głębi

Jako argument funkcja przyjmuje stałe określającą rodzaj bufora, który chcemy wyczyścić: **GL_COLOR_BUFFER_BIT**, **GL_DEPTH_BUFFER_BIT**, **GL_ACCUM_BUFFER_BIT**, **GL_STENCIL_BUFFER_BIT**. Dodatkowo istnieje specjalna funkcja, która czyści jedynie **bufor kolorów** oraz ustawia go na wybrany kolor. Przyjmuje ona cztery argumenty określające kolor w formacie **RGBA** na jaki ma zostać wyczyszczony bufor.

Funkcja czyszcząca bufor kolorów

glClearColor(GLfloat R, GLfloat G, GLfloat B, GLfloat A)

Rysowanie prymitywów

Podstawową rodziną funkcji wykorzystywaną w bibliotece OpenGL do rysowania różnego rodzaju **prymitywów** *(punktów, odcinków, wielokątów)* jest **glVertex**.

Rysowanie prymitywów

Podstawową rodziną funkcji wykorzystywaną w bibliotece OpenGL do rysowania różnego rodzaju **prymitywów** *(punktów, odcinków, wielokątów)* jest **glVertex**. Funkcje z tej rodziny służą do rysowania **pojedynczego punktu**. Istnieje wiele wersji glVertex, różniących się ilością oraz typem przyjmowanych argumentów wejściowych.

Rysowanie prymitywów

W celu narysowani pojedynczego punktu za pomocą **glVertex**, należy umieścić ją wewnątrz bloku tworzonego przez parę funkcji **glBegin** oraz **glEnd**.

Rysowanie prymitywów

W celu narysowani pojedynczego punktu za pomocą **glVertex**, należy umieścić ją wewnątrz bloku tworzonego przez parę funkcji **glBegin** oraz **glEnd**. Funkcja glBegin informuje bibliotekę OpenGL o rozpoczęciu rysowania grafiki. Przyjmuje ona jeden argument będący typem rysowanego prymitywu. Typ ten określa się za pomocą odpowiedniego argumentu.

Rysowanie prymitywów

W celu narysowani pojedynczego punktu za pomocą **glVertex**, należy umieścić ją wewnątrz bloku tworzonego przez parę funkcji **glBegin** oraz **glEnd**. Funkcja glBegin informuje bibliotekę OpenGL o rozpoczęciu rysowania grafiki. Przyjmuje ona jeden argument będący typem rysowanego prymitywu. Typ ten określa się za pomocą odpowiedniego argumentu.

Uwaga

Każde wywołanie funkcji **glBegin** musi zostać zakończone bezparametrową funkcją **glEnd**.

Rysowanie prymitywów

Blok rysujący

glBegin(GLenum mode);

...

Tu następuje rysowanie prymitywów

...

glEnd();

Rysowanie prymitywów

Blok rysujący

glBegin(GLenum mode);
...
Tu następuje rysowanie prymitywów
...
glEnd();

Uwaga

Blok opisany za pomocą funkcji **glBegin** oraz **glEnd** nie może być zagnieżdżony w innym bloku.

Rysowanie pojedynczego punktu

Punkt jest najprostszym prymitywem graficznym. W celu narysowania punktu jako argument wejściowy dla funkcji **glBegin** podajemy **GL_POINTS**. Argument ten informuje bibliotekę OpenGL, aby narysowała punkt w trójwymiarowej przestrzeni kartezjańskiej.

Rysowanie pojedynczego punktu

Punkt jest najprostszym prymitywem graficznym. W celu narysowania punktu jako argument wejściowy dla funkcji **glBegin** podajemy **GL_POINTS**. Argument ten informuje bibliotekę OpenGL, aby narysowała punkt w trójwymiarowej przestrzeni kartezjańskiej.

Rysowanie punktu

```
glBegin(GL_POINTS);
  glVertex3f(5.0f, 5.0f, 5.0f);
glEnd();
```

Rysowanie pojedynczego punktu

Rysowanym punktom można ustawić określony rozmiar. Funkcja **glPointSize** przyjmuje tylko jeden argument, będący wielkością punktu. Podobnie jak większość funkcji OpenGL działa ona na bieżąco. Argumenty przez nią ustawione będą oddziaływać na kolejne funkcje tak długo, aż nie wywołamy jej raz jeszcze z innym argumentem.

Funkcja służąca do ustawienia rozmiaru punktu

glPointSize(GLfloat rozmiar)

Rysowanie linii

Rysowanie linii jest zbliżone do rysowania punktu i polega na przekazaniu do glBegin argumentu **GL_LINES**.

Rysowanie linii

Rysowanie linii jest zbliżone do rysowania punktu i polega na przekazaniu do glBegin argumentu **GL_LINES**.

Rysowanie linii

```
glBegin(GL_LINES);
  glVertex3f(-5.0f, 0.0f, 3.0f);
  glVertex3f(5.0f, 3.0f, 0.0f);
glEnd();
```

Rysowanie linii

Rysowanym liniom można ustawić określony rozmiar. Funkcja **glLineWidth** przyjmuje tylko jeden argument, będący szerokości linii.

Funkcja służąca do ustawienia rozmiaru linii

glLineWidth(GLfloat rozmiar)

Rysowanie pozostałych prymitywów

Pozostałe prymitywy	
Argument	**Opis**
GL_LINE_STRIP	Odcinki połączone *(łamana)*
GL_LINE_LOOP	Odcinki połączone *(łamana zamknięta)*
GL_TRIANGLES	Trójkąty
GL_TRIANGLE_STRIP	Połączone trójkąty *(pasek trójkątów)*
GL_TRIANGLE_FAN	Połączone trójkąty *(wachlarz)*
GL_QUADS	Czworokąty
GL_QUAD_STRIP	Połączone czworokąty
GL_POLYGON	Wielokąty o dowolnej liczbie wierzchołków

Tryb wypełniania

Tryb wypełniania wielokątów umożliwia decydowanie o tym, jak ma być wypełniona przednia, a jak tylna strona wielokąta.

Tryb wypełniania

Tryb wypełniania wielokątów umożliwia decydowanie o tym, jak ma być wypełniona przednia, a jak tylna strona wielokąta.

Funkcje służące do ustawiania trybu wypełniania

glPolygonMode(GLenum face, GLenum mode)

Tryb wypełniania

Tryb wypełniania wielokątów umożliwia decydowanie o tym, jak ma być wypełniona przednia, a jak tylna strona wielokąta.

Funkcje służące do ustawiania trybu wypełniania

glPolygonMode(GLenum face, GLenum mode)

Jej pierwszy argument określa, których ścian będzie dotyczyć wypełnianie.

Tryb wypełniania

Tryb wypełniania wielokątów umożliwia decydowanie o tym, jak ma być wypełniona przednia, a jak tylna strona wielokąta.

Funkcje służące do ustawiania trybu wypełniania

glPolygonMode(GLenum face, GLenum mode)

Jej pierwszy argument określa, których ścian będzie dotyczyć wypełnianie.

Pierwszy argument

- GL_FRONT - przednia ściana
- GL_BACK - tylna ściana
- GL_FRONT_AND_BACK - przednia i tylna ściana

Tryb wypełniania

Drugi argument określa styl wypełniania ścian wielokąta.

Tryb wypełniania

Drugi argument określa styl wypełniania ścian wielokąta.

Drugi argument

- GL_LINE - widoczne będą tylko linie wielokąta
- GL_POINT - widoczne będą tylko punkty wielokąta
- GL_FILL - wypełnianie całego wielokąta (domyślnie)

Widoczność i ukrywanie ścian

Często niepotrzebne jest rysowanie wewnętrznych ścian obiektów, zajmuje to jedynie dodatkową moc obliczeniową komputera. OpenGL umożliwia automatyczne usuwanie niewidocznych ścian. Aby ukryć ściany należy włączyć opcję ukrywania niewidocznych powierzchni wielokątów.

Widoczność i ukrywanie ścian

Często niepotrzebne jest rysowanie wewnętrznych ścian obiektów, zajmuje to jedynie dodatkową moc obliczeniową komputera. OpenGL umożliwia automatyczne usuwanie niewidocznych ścian. Aby ukryć ściany należy włączyć opcję ukrywania niewidocznych powierzchni wielokątów.

Włączenie opcji ukrywania niewidocznych ścian

glEnable(GL_CULL_FACE)

Widoczność i ukrywanie ścian

Często niepotrzebne jest rysowanie wewnętrznych ścian obiektów, zajmuje to jedynie dodatkową moc obliczeniową komputera. OpenGL umożliwia automatyczne usuwanie niewidocznych ścian. Aby ukryć ściany należy włączyć opcję ukrywania niewidocznych powierzchni wielokątów.

Włączenie opcji ukrywania niewidocznych ścian

glEnable(GL_CULL_FACE)

Następnie należy wywołać funkcję **glCullFace**, przyjmuje ona jeden z trzech argumentów.

Widoczność i ukrywanie ścian

Często niepotrzebne jest rysowanie wewnętrznych ścian obiektów, zajmuje to jedynie dodatkową moc obliczeniową komputera. OpenGL umożliwia automatyczne usuwanie niewidocznych ścian. Aby ukryć ściany należy włączyć opcję ukrywania niewidocznych powierzchni wielokątów.

Włączenie opcji ukrywania niewidocznych ścian

glEnable(GL_CULL_FACE)

Następnie należy wywołać funkcję **glCullFace**, przyjmuje ona jeden z trzech argumentów.

Argumenty

- GL_FRONT - ściana przednia
- GL_BACK - ściana tylna
- GL_FRONT_AND_BACK - ściana przednia i tylna

Rysowanie prymitywów w przestrzeni 2D

Przykładowy program


Windows


Linux


Rysowanie prymitywów w przestrzeni 3D

Przykładowy program


Windows


Linux


Transformacje 3D

Podstawowe przekształcenia dostępne w OpenGL

- Przesuwanie obiektów **(translacja)**

Transformacje 3D

Podstawowe przekształcenia dostępne w OpenGL

- Przesuwanie obiektów **(translacja)**
- Obracanie obiektów **(rotacja)**

Transformacje 3D

Podstawowe przekształcenia dostępne w OpenGL

- Przesuwanie obiektów **(translacja)**
- Obracanie obiektów **(rotacja)**
- Rozciągnie oraz kurczenie obiektów **(skalowanie)**

Transformacje 3D

Podstawowe przekształcenia dostępne w OpenGL

- Przesuwanie obiektów **(translacja)**
- Obracanie obiektów **(rotacja)**
- Rozciągnie oraz kurczenie obiektów **(skalowanie)**

Uwaga

W OpenGL zamiast bezpośrednio modyfikować obiekty dokonujemy **przekształcenia układu współrzędnych**, w którym te obiekty się znajdują.

Transformacje 3D

W celu dokonania podstawowych przekształceń obiektu (translacji, rotacji, skalowania) musimy zmodyfikować **macierz modelowania**. W bibliotece OpenGL dostępne są gotowe funkcje umożliwiające taką modyfikację.

Translacja

Translacja polega na przesunięciu zadanego obiektu o *wektor przesunięcia*.

Translacja

Translacja polega na przesunięciu zadanego obiektu o *wektor przesunięcia*.

Funkcje służące do translacji obiektu

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

Translacja

Translacja polega na przesunięciu zadanego obiektu o *wektor przesunięcia*.

Funkcje służące do translacji obiektu

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

Funkcja mnoży bieżąca **macierz modelowania** przez **macierz translacji**. Jako argumenty przyjmuje *składowe wektora przesunięcia* na poszczególnych osiach układu współrzędnych. Iloczyn macierzy staje się bieżącą macierzą modelowania.

Rotacja

Rotacja polega na obrocie obiektu o zadany kąt wokół *wektor obrotu*.

Rotacja

Rotacja polega na obrocie obiektu o zadany kąt wokół *wektor obrotu*.

Funkcje służące do rotacji obiektu

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

Rotacja

Rotacja polega na obrocie obiektu o zadany kąt wokół *wektor obrotu*.

Funkcje służące do rotacji obiektu

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

Funkcja mnoży bieżącą **macierz modelowania** 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). Iloczyn macierzy staje się bieżącą macierzą modelowania.

Skalowanie

Skalowanie polega na zmianie rozmiarów zadanego obiektu o *wektor skalowania*.

Skalowanie

Skalowanie polega na zmianie rozmiarów zadanego obiektu o *wektor skalowania*.

Funkcje służące do skalowania obiektu

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

Skalowanie

Skalowanie polega na zmianie rozmiarów zadanego obiektu o *wektor skalowania*.

Funkcje służące do skalowania obiektu

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

Funkcja mnoży bieżąca **macierz modelowania** przez **macierz skalowania**. Iloczyn macierzy staje się bieżącą macierzą modelowania.

Transformacje (translacja, rotacja, skalowanie)

Przykładowy program


Windows


Linux


Obsługa programu

- q, Q – obrót wzdłuż osi X
- w, W – obrót wzdłuż osi Y
- e, E – obrót wzdłuż osi Z
- a, A – przesunięcie wzdłuż osi X
- s, S - przesunięcie wzdłuż osi Y
- d, D - przesunięcie wzdłuż osi Z
- z, Z – skalowanie względem osi X
- x, X – skalowanie względem osi Y
- c, C – skalowanie względem osi Z
- r, R – powrót do postaci początkowej

Oświetlenie i materiały

Specyfikacja biblioteki OpenGL rozróżnia trzy niezależne rodzaje światła: **światło otoczenia** *(ambient light)*, **światło rozproszone** *(diffuse light)* oraz **światło odbite** *(specular light)*.

Oświetlenie i materiały

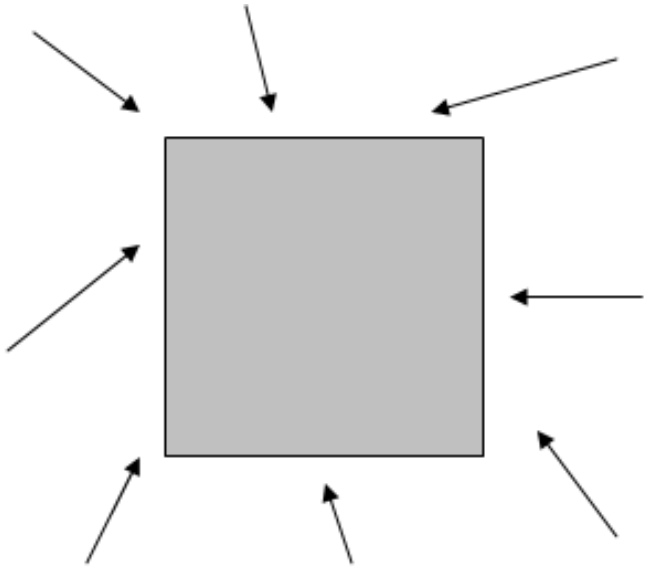
Specyfikacja biblioteki OpenGL rozróżnia trzy niezależne rodzaje światła: **światło otoczenia** *(ambient light)*, **światło rozproszone** *(diffuse light)* oraz **światło odbite** *(specular light)*. Każdy komponent światła definiowany jest przez cztery wartości **RGBA**, określające względną intensywność czerwonego, zielonego, niebieskiego oraz składowej **alfa**.

Światło otoczenia

Światło otoczenia jest to najprostszy typ światła. Rozprasza się równomiernie na powierzchni danego obiektu niezależnie od swojego źródła.

Światło otoczenia

Światło otoczenia jest to najprostszy typ światła. Rozprasza się równomiernie na powierzchni danego obiektu niezależnie od swojego źródła. Odbija się od obiektów znajdujących się w jego zasięgu, rozjaśniając wszystkie obiekty znajdujące się dookoła.

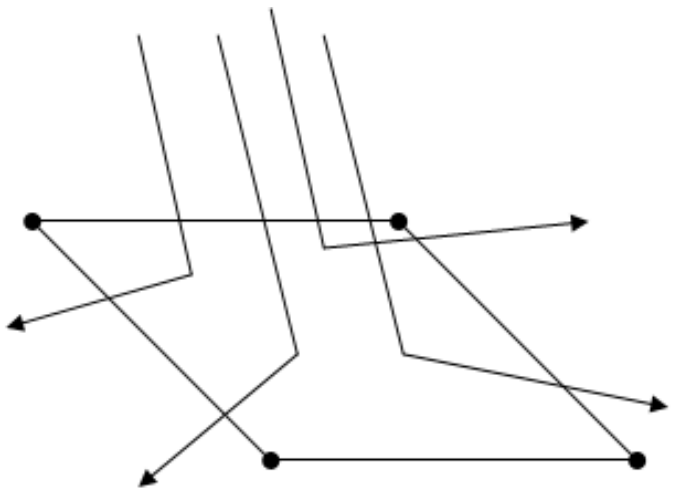


Światło rozproszone

Światło rozproszone w przeciwieństwie do światła otoczenia posiada kierunek. Oznacza to, że gdy oświetlamy światłem rozproszonym jedną ze stron danego obiektu to tylko ta strona jest oświetlona, a pozostałe nie.

Światło rozproszone

Światło rozproszone w przeciwieństwie do światła otoczenia posiada kierunek. Oznacza to, że gdy oświetlamy światłem rozproszonym jedną ze stron danego obiektu to tylko ta strona jest oświetlona, a pozostałe nie. Światło rozproszone rozprowadzane jest na powierzchni obiektu w sposób równomierny.

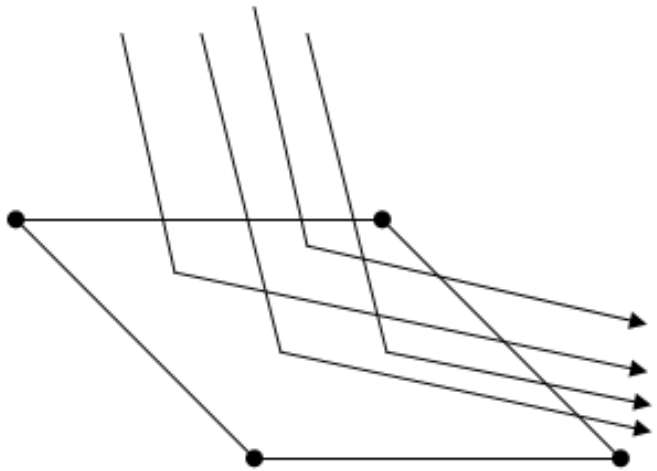


Światło odbite

Światło odbite również posiada określony kierunek, ale jest odbijane na powierzchni obiektu w konkretnym kierunku.

Światło odbite

Światło odbite również posiada określony kierunek, ale jest odbijane na powierzchni obiektu w konkretnym kierunku. Takie typ światła powoduje, że na powierzchni obiektu pojawia się rozbłysk (zazwyczaj o kolorze białym).



Wektory normalne

Aby w OpenGL oświetlenie było obliczanie poprawnie, każdej ścianie należy przypisać wektor normalny. Służą do tego celu funkcje z rodziny **glNormal**.

Wektory normalne

Aby w OpenGL oświetlenie było obliczanie poprawnie, każdej ścianie należy przypisać wektor normalny. Służą do tego celu funkcje z rodziny **glNormal**.

Funkcje służące do ustawienia wektora normalnego

glNormal3f(GLfloat x, GLfloat y, GLfloat z)
glNormal3d(GLdouble x, GLdouble y, GLdouble z)

Wektory normalne

Aby w OpenGL oświetlenie było obliczanie poprawnie, każdej ścianie należy przypisać wektor normalny. Służą do tego celu funkcje z rodziny **glNormal**.

Funkcje służące do ustawienia wektora normalnego

glNormal3f(GLfloat x, GLfloat y, GLfloat z)
glNormal3d(GLdouble x, GLdouble y, GLdouble z)

Funkcje te definiują wektor normalny dla powierzchni. Przyjmują trzy argumenty będące współrzędnymi wektora normalnego. Wektor ten powinien być jednostkowy (znormalizowany).

Oświetlenie

Aby włączyć oświetlenie w OpenGL należy ustawić odpowiedni parametr maszyny stanu.

Oświetlenie

Aby włączyć oświetlenie w OpenGL należy ustawić odpowiedni parametr maszyny stanu.

Włączenie opcji oświetlenia

glEnable(GL_LIGHTING)

Oświetlenie

Aby włączyć oświetlenie w OpenGL należy ustawić odpowiedni parametr maszyny stanu.

Włączenie opcji oświetlenia

glEnable(GL_LIGHTING)

Włącznie opcji **GL_LIGHTING** nakazuje bibliotece, aby przy obliczaniu koloru wierzchołków w scenie brała pod uwagę właściwości materiału i parametry oświetlenia.

Oświetlenie

W OpenGL dostępnych jest **osiem niezależnych** źródeł światła. Każdemu z nich możemy przypisać inne położenie, kolor otoczenia czy rozproszenia oraz kilka innych parametrów.

Oświetlenie

W OpenGL dostępnych jest **osiem niezależnych** źródeł światła. Każdemu z nich możemy przypisać inne położenie, kolor otoczenia czy rozproszenia oraz kilka innych parametrów.

Funkcje służące do ustawienia oświetlenia

glLightf(GLenum light, GLenum pname, GLfloat param)
glLighti(GLenum light, GLenum pname, GLint param)

Oświetlenie

W OpenGL dostępnych jest **osiem niezależnych** źródeł światła. Każdemu z nich możemy przypisać inne położenie, kolor otoczenia czy rozproszenia oraz kilka innych parametrów.

Funkcje służące do ustawienia oświetlenia

glLightf(GLenum light, GLenum pname, GLfloat param)
glLighti(GLenum light, GLenum pname, GLint param)

Funkcje te przyjmuje trzy argumenty. Pierwszy argument określa którego światła parametry modyfikujemy. Drugi argument określa, który parametr światła zmieniamy. Trzeci argument określa kolor, pojedynczą wartość lub wskaźnik na wartości przypisywane danemu parametrowi (określonego światła).

Oświetlenie

Argument *"light"*

- GL_LIGHTx - argument określa, którego światła parametry modyfikujemy x - jest to liczba z przedziału 0 - 7

Oświetlenie

Argument *"light"*

- GL_LIGHTx - argument określa, którego światła parametry modyfikujemy x - jest to liczba z przedziału 0 - 7

Argument *"pname"*

- GL_AMBIENT - światło otoczenia
- GL_DIFFUSE - światło rozproszone
- GL_SPECULAR - światło odbite
- GL_POSITION - pozycja światła
- GL_SPOT_CUTOFF - stożek widzenia
- GL_SPOT_DIRECTION - kierunek
- GL_SPOT_EXPONENT - płynne przejście

Materiały

Materiały określają kolor dla obiektu, na który pada światło. W OpenGL każdy materiał pokrywający zdefiniowany obiekt ma określone właściwości odbicia światła otaczającego, rozproszonego i odbitego.

Materiały

Materiały określają kolor dla obiektu, na który pada światło. W OpenGL każdy materiał pokrywający zdefiniowany obiekt ma określone właściwości odbicia światła otaczającego, rozproszonego i odbitego. Barwa obiektu widzianego przez obserwatora zależy od rodzaju światła padającego na obiekt i materiału, z jakiego jest wykonany.

Materiały

Podobnie jak światło, materiał również można opisać za pomocą współczynników ambient, diffuse i specular, które w tym wypadku oznaczają procentową zdolność materiału do **absorbowania** i **odbijania** światła.

Materiały

Podobnie jak światło, materiał również można opisać za pomocą współczynników ambient, diffuse i specular, które w tym wypadku oznaczają procentową zdolność materiału do **absorbowania** i **odbijania** światła.

Funkcje służące do zarządzania materiałami

glMaterialf(GLenum face, GLenum pname, GLfloat param)
glMateriald(GLenum face, GLenum pname, GLdouble param)
glMateriali(GLenum face, GLenum pname, GLint param)

Materiały

Podobnie jak światło, materiał również można opisać za pomocą współczynników ambient, diffuse i specular, które w tym wypadku oznaczają procentową zdolność materiału do **absorbowania** i **odbijania** światła.

Funkcje służące do zarządzania materiałami

glMaterialf(GLenum face, GLenum pname, GLfloat param)
glMateriald(GLenum face, GLenum pname, GLdouble param)
glMateriali(GLenum face, GLenum pname, GLint param)

Funkcje z rodziny **glMaterial** przyjmują trzy argumenty. Pierwszy argument określa, czy definiujemy materiał dla przednich, tylnych, czy obydwu stron wielokątów. Drugi argument określa jaką właściwość materiału definiujemy. Ostatni argument to tablica wartości składających się na kolor **RGB** lub pojedyncza wartość.

Materiały

Argument *"face"*

- GL_FRONT - przednia ściana wielokąta
- GL_BACK - tylna ściana wielokąta
- GL_FRONT_AND_BACK - przednia i tylna ściana wielokąta

Materiały

Argument *"face"*

- GL_FRONT - przednia ściana wielokąta
- GL_BACK - tylna ściana wielokąta
- GL_FRONT_AND_BACK - przednia i tylna ściana wielokąta

Argument *"pname"*

- GL_AMBIENT - definiujemy kolor materiału dla światła otoczenia
- GL_DIFFUSE - definiujemy kolor materiału dla światła rozproszonego
- GL_SPECULAR - definiujemy kolor materiału dla światła odbitego
- GL_AMBIENT_AND_DIFFUSE - definiujemy kolor materiału dla światła otoczenia i rozproszonego
- GL_SHININESS - definiujemy kolor materiału z możliwością zmiany mocy światła odbitego
- GL_EMISSION - definiujemy kolor materiału z efektem emisji, który jest dodawany do wynikowego koloru obiektu

Śledzenie kolorów

Od momentu włączenia oświetlenia, kolor **własny obiektów** ustawiany z pomocą funkcji z rodziny **glColor** jest **ignorowany**. Wynikowy kolor ustalany jest na podstawie właściwości materiałów.

Śledzenie kolorów

Od momentu włączenia oświetlenia, kolor **własny obiektów** ustawiany z pomocą funkcji z rodziny **glColor** jest **ignorowany**. Wynikowy kolor ustalany jest na podstawie właściwości materiałów. Śledzenie kolorów automatycznie wiąże kolor własny obiektu z właściwością danego materiału. W efekcie funkcje z rodziny glColor zmieniają parametry materiału.

Śledzenie kolorów

Aby włączyć śledzenie kolorów w OpenGL należy ustawić odpowiedni parametr maszyny stanu.

Śledzenie kolorów

Aby włączyć śledzenie kolorów w OpenGL należy ustawić odpowiedni parametr maszyny stanu.

Włączenie opcji śledzenia kolorów

glEnable(GL_COLOR_MATERIAL)

Śledzenie kolorów

Aby włączyć śledzenie kolorów w OpenGL należy ustawić odpowiedni parametr maszyny stanu.

Włączenie opcji śledzenia kolorów

glEnable(GL_COLOR_MATERIAL)

Przypisanie śledzonego koloru materiału do glColor odbywa się z wykorzystaniem specjalnej funkcji.

Śledzenie kolorów

Aby włączyć śledzenie kolorów w OpenGL należy ustawić odpowiedni parametr maszyny stanu.

Włączenie opcji śledzenia kolorów

glEnable(GL_COLOR_MATERIAL)

Przypisanie śledzonego koloru materiału do glColor odbywa się z wykorzystaniem specjalnej funkcji.

Funkcja służąca do śledzenia kolorów

glColorMaterial(GLenum face, GLenum mode)

Śledzenie kolorów

Aby włączyć śledzenie kolorów w OpenGL należy ustawić odpowiedni parametr maszyny stanu.

Włączenie opcji śledzenia kolorów

glEnable(GL_COLOR_MATERIAL)

Przypisanie śledzonego koloru materiału do glColor odbywa się z wykorzystaniem specjalnej funkcji.

Funkcja służąca do śledzenia kolorów

glColorMaterial(GLenum face, GLenum mode)

Funkcja przyjmuje dwa argumenty. Pierwszy argument określa stronę wielokąta, której będzie dotyczyć śledzenie kolorów. Drugi argument określa materiał, który będzie śledzony.

Oświetlenie i materiały

Przykładowy program nr 1


Windows



Linux


Przykładowy program nr 2


Windows



Linux


Bibliografia


Dave Shreiner

OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1 (7th Edition)

Addison-Wesley Professional, 2009


Jackie Neider, Tom Davis, Mason Woo

The OpenGL Programming Guide - The Redbook

http://www.opengl.org/documentation/red_book/


Richard S. Wright jr, Michael Sweet

OpenGL: księga eksperta

Helion, 1999


Kevin Hawkins, Dave Astle

OpenGL. Programowanie gier

Helion, 2003

Koniec części 3



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



Politechnika Świętokrzyska

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

" Program Rozwojowy Potencjału Dydaktycznego Politechniki Świętokrzyskiej w Kielcach: *kształcenie w nowoczesnych obszarach techniki* "

Program Operacyjny Kapitał Ludzki, Priorytet IV Szkolnictwo wyższe i nauka, Działanie 4.1. Wzmocnienie i rozwój potencjału dydaktycznego uczelni oraz zwiększenie liczby absolwentów kierunków o kluczowym znaczeniu dla gospodarki opartej na wiedzy, Poddziałanie 4.1.1. Wzmocnienie potencjału dydaktycznego uczelni.

Umowa UDA–POKL.04.01.01-00395/09-00