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 4



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. Teksturowanie
   - Generowanie tekstur
   - Powiązanie obiektu z teksturą
   - Tworzenie obrazów tekstury
   - Nakładanie tekstur
   - Filtrowanie tekstur
   - Przykład
2. Dodatkowe efekty graficzne
   - Mgła
   - Blending
   - Przezroczystość
   - Przykład
3. 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.

Teksturowanie

Teksturowanie jest techniką w grafice komputerowej polegającą na nakładaniu na **obiekty** *(wielokąty)* **obrazów** *(tekstur)*, które przedstawiają brakujące szczegóły.

Teksturowanie

Teksturowanie jest techniką w grafice komputerowej polegającą na nakładaniu na **obiekty** *(wielokąty)* **obrazów** *(tekstur)*, które przedstawiają brakujące szczegóły.

Rodzaje tekstur w OpenGL

- Tekstury **jednowymiarowe** (1D)
- Tekstury **dwuwymiarowe** (2D)
- Tekstury **trójwymiarowe** (3D)

Teksturowanie

Tekstury **jednowymiarowe** są to płaskie obrazy w postaci paska o szerokości lub wysokości jednego piksela.

Teksturowanie

Tekstury **jednowymiarowe** są to płaskie obrazy w postaci paska o szerokości lub wysokości jednego piksela. Drugim typem tekstur są tekstury **dwuwymiarowe**, są to płaskie obrazy o zadanej szerokości i wysokości. Tekstury takie składają się z **teksli** czyli grup barwnych pikseli. Najczęściej tekstury 2D używane są w celu zastąpienia powierzchni o skomplikowanej geometrii lub w celu stworzenia efektu realistycznego w danej scenie.

Teksturowanie

Tekstury **jednowymiarowe** są to płaskie obrazy w postaci paska o szerokości lub wysokości jednego piksela. Drugim typem tekstur są tekstury **dwuwymiarowe**, są to płaskie obrazy o zadanej szerokości i wysokości. Tekstury takie składają się z **teksli** czyli grup barwnych pikseli. Najczęściej tekstury 2D używane są w celu zastąpienia powierzchni o skomplikowanej geometrii lub w celu stworzenia efektu realistycznego w danej scenie. Trzecim i ostatnim typem tekstur są tekstury **trójwymiarowe** *(wolumetryczne)*, są to trójwymiarowe obrazy o zadanej szerokości, wysokości oraz głębokości. Tekstury takie również składają się z teksli. Teksle w przypadku tekstur 3D są bryłami sześciennymi nazywanymi **wokselami** *(pikselami trójwymiarowymi)*.

Teksturowanie

Tekstury **jednowymiarowe** są to płaskie obrazy w postaci paska o szerokości lub wysokości jednego piksela. Drugim typem tekstur są tekstury **dwuwymiarowe**, są to płaskie obrazy o zadanej szerokości i wysokości. Tekstury takie składają się z **teksli** czyli grup barwnych pikseli. Najczęściej tekstury 2D używane są w celu zastąpienia powierzchni o skomplikowanej geometrii lub w celu stworzenia efektu realistycznego w danej scenie. Trzecim i ostatnim typem tekstur są tekstury **trójwymiarowe** *(wolumetryczne)*, są to trójwymiarowe obrazy o zadanej szerokości, wysokości oraz głębokości. Tekstury takie również składają się z teksli. Teksle w przypadku tekstur 3D są bryłami sześciennymi nazywanymi **wokselami** *(pikselami trójwymiarowymi)*.

Uwaga

Najczęściej stosowane są **tekstury dwuwymiarowe 2D**.

Teksturowanie

Aby włączyć teksturowanie w OpenGL, należy za pomocą funkcji **glEnable** ustawić odpowiednią parametr maszyny stanu (w zależności od tego jaki rodzaj teksturowania chcemy włączyć).

Teksturowanie

Aby włączyć teksturowanie w OpenGL, należy za pomocą funkcji **glEnable** ustawić odpowiednią parametr maszyny stanu (w zależności od tego jaki rodzaj teksturowania chcemy włączyć).

Rodzaje teksturowania

- GL_TEXTURE_1D - teksturowanie jednowymiarowe
- GL_TEXTURE_2D - teksturowanie dwuwymiarowe
- GL_TEXTURE_3D - teksturowanie trójwymiarowe

Teksturowanie

Aby włączyć teksturowanie w OpenGL, należy za pomocą funkcji **glEnable** ustawić odpowiednią parametr maszyny stanu (w zależności od tego jaki rodzaj teksturowania chcemy włączyć).

Rodzaje teksturowania

- GL_TEXTURE_1D - teksturowanie jednowymiarowe
- GL_TEXTURE_2D - teksturowanie dwuwymiarowe
- GL_TEXTURE_3D - teksturowanie trójwymiarowe

Uwaga

Rozmiar tekstury musi być **potęgą 2** np. 16x16, 64x64, 128x512, 16x256, 16x16x16, 256x256x256 itp.. Maksymalny rozmiar tekstury określony jest możliwościami danej karty graficznej.

Generowanie tekstur

W celu wygenerowania tekstury w OpenGL używamy funkcji **glGenTextures**.

Generowanie tekstur

W celu wygenerowania tekstury w OpenGL używamy funkcji **glGen-Textures**.

Funkcja służąca do generowania tekstury

*glGenTextures(GLsizei n, GLuint *textures)*

Generowanie tekstur

W celu wygenerowania tekstury w OpenGL używamy funkcji **glGenTextures**.

Funkcja służąca do generowania tekstury

*glGenTextures(GLsizei n, GLuint *textures)*

Funkcja przyjmuje dwa argumenty. Pierwszy argument określa **ilość generowanych tekstur**. Drugi argument jest **wskaźnikiem** na element tablicy typu GLuint, w którym będzie przechowywana wygenerowana tekstura.

Powiązanie obiektu z teksturą

Powiązanie tekstury polega na przypisaniu nazwy tektury do jej danych. Do tego celu wykorzystywana jest funkcja **glBindTexture**.

Powiązanie obiektu z teksturą

Powiązanie tekstury polega na przypisaniu nazwy tektury do jej danych. Do tego celu wykorzystywana jest funkcja **glBindTexture**.

Funkcja służąca do powiązania tekstury

glBindTexture(GLenum target, GLuint texture)

Powiązanie obiektu z teksturą

Powiązanie tekstury polega na przypisaniu nazwy tektury do jej danych. Do tego celu wykorzystywana jest funkcja **glBindTexture**.

Funkcja służąca do powiązania tekstury

glBindTexture(GLenum target, GLuint texture)

Funkcja przyjmuje dwa argumenty. Pierwszy argument jest stałą określająca **rodzaj tekstury** (GL_TEXTURE_1D, GL_TEXTURE_2D itd.). Drugi argument jest **tablicą** typu GLuint, w której przechowywana będzie tekstura. Aby zwolnić obiekty tekstu wykorzystujemy funkcję **glDeleteTextures**.

Tworzenie obrazów tekstury

Gdy mamy już wygenerowaną oraz powiązaną teksturę za pomocą funkcji **glGenTextures** oraz **glBindTexture** możemy jej przypisać konkretne dane obrazowe.

Tworzenie obrazów tekstury

Gdy mamy już wygenerowaną oraz powiązaną teksturę za pomocą funkcji **glGenTextures** oraz **glBindTexture** możemy jej przypisać konkretne dane obrazowe. W OpenGL do tworzenia obrazów tekstur jednowymiarowych służy funkcja **glTexImage1D**.

Tworzenie obrazów tekstury

Gdy mamy już wygenerowaną oraz powiązaną teksturę za pomocą funkcji **glGenTextures** oraz **glBindTexture** możemy jej przypisać konkretne dane obrazowe. W OpenGL do tworzenia obrazów tekstur jednowymiarowych służy funkcja **glTexImage1D**.

Funkcja służąca do tworzenia tekstur jednowymiarowych

*glTexImage1D(GLenum target, GLint level, GLint components, GLsizei width, GLint border, GLenum format, GLenum type, const GLvoid *pixels)*

Tworzenie obrazów tekstury

Funkcja przyjmuje osiem argumentów. Argument **target** określa, która tekstura powinna zostać zdefiniowana, musi nim być GL_TEXTURE_1D.

Tworzenie obrazów tekstury

Funkcja przyjmuje osiem argumentów. Argument **target** określa, która tekstura powinna zostać zdefiniowana, musi nim być GL_TEXTURE_1D. Argument **level** wskazuje poziom szczegółów obrazu tekstury i zwykle wynosi 0 (inne wartości używane są przy teksturach typu **mipmap**). Argument **components** określa ilość wartości koloru użytych dla każdego piksela.

Tworzenie obrazów tekstury

Funkcja przyjmuje osiem argumentów. Argument **target** określa, która tekstura powinna zostać zdefiniowana, musi nim być GL_TEXTURE_1D. Argument **level** wskazuje poziom szczegółów obrazu tekstury i zwykle wynosi 0 (inne wartości używane są przy teksturach typu **mipmap**). Argument **components** określa ilość wartości koloru użytych dla każdego piksela. Argument **width** i **border** określa rozmiar obrazu tekstury. Wartość border określa ilość pikseli ramki może mieć wartość 0, 1 lub 2. Argument **width** określa szerokość głównego obrazu tekstury (bez pikseli ramki) i musi być potęgą liczby 2.

Tworzenie obrazów tekstury

Funkcja przyjmuje osiem argumentów. Argument **target** określa, która tekstura powinna zostać zdefiniowana, musi nim być GL_TEXTURE_1D. Argument **level** wskazuje poziom szczegółów obrazu tekstury i zwykle wynosi 0 (inne wartości używane są przy teksturach typu **mipmap**). Argument **components** określa ilość wartości koloru użytych dla każdego piksela. Argument **width** i **border** określa rozmiar obrazu tekstury. Wartość border określa ilość pikseli ramki może mieć wartość 0, 1 lub 2. Argument **width** określa szerokość głównego obrazu tekstury (bez pikseli ramki) i musi być potęgą liczby 2. Argument **format** wskazuje rodzaj wartości koloru w teksturze (GL_COLOR_INDEX, GL_LUMINANCE, GL_RGB, GL_RGBA).

Tworzenie obrazów tekstury

Funkcja przyjmuje osiem argumentów. Argument **target** określa, która tekstura powinna zostać zdefiniowana, musi nim być GL_TEXTURE_1D. Argument **level** wskazuje poziom szczegółów obrazu tekstury i zwykle wynosi 0 (inne wartości używane są przy teksturach typu **mipmap**). Argument **components** określa ilość wartości koloru użytych dla każdego piksela. Argument **width** i **border** określa rozmiar obrazu tekstury. Wartość border określa ilość pikseli ramki może mieć wartość 0, 1 lub 2. Argument **width** określa szerokość głównego obrazu tekstury (bez pikseli ramki) i musi być potęgą liczby 2. Argument **format** wskazuje rodzaj wartości koloru w teksturze (GL_COLOR_INDEX, GL_LUMINANCE, GL_RGB, GL_RGBA). Argument **type** określa typ danych dla wartości pikseli. Natomiast ostatni argument **pixels** stanowi dane pikseli.

Tworzenie obrazów tekstury

Do tworzenia obrazów tekstur dwuwymiarowych służy funkcja **glTexImage2D**.

Tworzenie obrazów tekstury

Do tworzenia obrazów tekstur dwuwymiarowych służy funkcja **glTexImage2D**.

Funkcja służąca do tworzenia tekstur dwuwymiarowych

*glTexImage2D(GLenum target, GLint level, GLint components, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid *pixels)*

Tworzenie obrazów tekstury

Funkcja przyjmuje dziewięć argumentów. Przyjmowane argumenty są w większości identyczne jak w przypadku funkcji **glTexImage1D**. Jedyna różnica polega na tym, że jako **target** podajemy GL_TEXTURE_2D. Występuje też jeden dodatkowy argument **height**, który określa wysokość głównego obrazu tekstury (bez pikseli ramki) i musi być potęgą liczby 2.

Nakładanie tekstur

Nakładanie tekstur wykonuje się za pomocą rodziny funkcji **glTexCoord**.

Nakładanie tekstur

Nakładanie tekstur wykonuje się za pomocą rodziny funkcji **glTexCoord**. W zależności od rodzaju tekstury funkcje te mogą przyjmować *jeden* lub *dwa* argumenty określające punkt tekstury odpowiadający kolejnemu definiowanemu przez polecenie **glVertex** wierzchołkowi.

Nakładanie tekstur

Nakładanie tekstur wykonuje się za pomocą rodziny funkcji **glTexCoord**. W zależności od rodzaju tekstury funkcje te mogą przyjmować *jeden* lub *dwa* argumenty określające punkt tekstury odpowiadający kolejnemu definiowanemu przez polecenie **glVertex** wierzchołkowi.

Nakładanie tekstur

```
glBegin(GL_TRIANGLES);
  glTexCoord2f(0.1f, 0.0f);
  glVertex3f(0.0f,0.0f,0.0f);
  glTexCoord2f(1.0f, 0.0f);
  glVertex3f(2.0f,0.0f,0.0f);
  glTexCoord2f(0.1f, 1.0f);
  glVertex3f(0.0f,-2.0f,0.0f);
glEnd();
```

Filtrowanie tekstur

W OpenGL występują **dwa** rodzaje filtrów tekstur. Filtr **pomniejszający** oraz **powiększający**.

Filtrowanie tekstur

W OpenGL występują **dwa** rodzaje filtrów tekstur. Filtr **pomniejszający** oraz **powiększający**. Filtr pomniejszający wykorzystywany jest w przypadku, gdy obraz przypisany do tekstury na ekranie jest mniejszy niż w rzeczywistości.

Filtrowanie tekstur

W OpenGL występują **dwa** rodzaje filtrów tekstur. Filtr **pomniejszający** oraz **powiększający**. Filtr pomniejszający wykorzystywany jest w przypadku, gdy obraz przypisany do tekstury na ekranie jest mniejszy niż w rzeczywistości. Natomiast filtr powiększający wykorzystywany jest gdy obraz z tekstury na ekranie jest większy niż wczytany z pliku.

Filtrowanie tekstur

Do ustawiania parametrów tekstury (w tym też rodzaju filtra) służy funkcja **glTexParameter**.

Filtrowanie tekstur

Do ustawiania parametrów tekstury (w tym też rodzaju filtra) służy funkcja **glTexParameter**.

Funkcje służące do ustawiania parametrów tekstury

glTexParameterf(GLenum target, GLenum pname, GLfloat param)
glTexParameteri(GLenum target, GLenum pname, GLint param)

Filtrowanie tekstur

Do ustawiania parametrów tekstury (w tym też rodzaju filtra) służy funkcja **glTexParameter**.

Funkcje służące do ustawiania parametrów tekstury

glTexParameterf(GLenum target, GLenum pname, GLfloat param)
glTexParameteri(GLenum target, GLenum pname, GLint param)

Funkcje przyjmują trzy argumenty. Pierwszy argument określa **typ tekstury** (GL_TEXTURE_1D, GL_TEXTURE_2D itd.). Drugi argument to **nazwa** zmienianego parametru tekstury. Ostatni argument to nowa **wartość** zmienianego parametru tekstury.

Teksturowanie

Przykładowy program


Windows


Linux


Mgła

Mgła jest to efekt, który wnosi wiele realizmu do tworzonej sceny, można go bardzo łatwo uzyskać, a jednocześnie daje naprawdę bardzo ciekawe rezultaty.

Mgła

Mgła jest to efekt, który wnosi wiele realizmu do tworzonej sceny, można go bardzo łatwo uzyskać, a jednocześnie daje naprawdę bardzo ciekawe rezultaty. Mgła pozwala ograniczyć widoczność, powodując jednocześnie że obiekty, które znajdują się daleko, nie muszą być renderowane.

Mgła

W OpenGL istnieje możliwość ustalenia dowolnego koloru mgły (w przeciwieństwie do mgły rzeczywistej).

Mgła

W OpenGL istnieje możliwość ustalenia dowolnego koloru mgły (w przeciwieństwie do mgły rzeczywistej). Aby włączyć mgłę w OpenGL należy ustawić odpowiedni parametr maszyny stanu.

Mgła

W OpenGL istnieje możliwość ustalenia dowolnego koloru mgły (w przeciwieństwie do mgły rzeczywistej). Aby włączyć mgłę w OpenGL należy ustawić odpowiedni parametr maszyny stanu.

Włączenie mgły

glEnable(GL_FOG)

Mgła

Do tworzenia mgły w OpenGL wykorzystywana jest rodzina funkcji **glFog**.

Mgła

Do tworzenia mgły w OpenGL wykorzystywana jest rodzina funkcji **glFog**.

Funkcje do tworzenia mgły

glFogf(GLenum pname, GLfloat param)
glFogi(GLenum pname, GLint param)
*glFogfv(GLenum pname, const GLfloat *params)*
*glFogiv(GLenum pname, const GLint *params)*

Mgła

Do tworzenia mgły w OpenGL wykorzystywana jest rodzina funkcji **glFog**.

Funkcje do tworzenia mgły

glFogf(GLenum pname, GLfloat param)
glFogi(GLenum pname, GLint param)
*glFogfv(GLenum pname, const GLfloat *params)*
*glFogiv(GLenum pname, const GLint *params)*

Każda z funkcji przyjmuje po **dwa** argumenty. Pierwszy argument określa modyfikowany **parametr** mgły, drugi argument to **wartość** parametru mgły.

Mgła

Argument *"pname"*

- GL_FOG_MODE - argument określa sposób w jaki obliczana (generowana) jest mgła
- GL_FOG_START - argument określa początek mgły
- GL_FOG_END - argument określa koniec mgły
- GL_FOG_DENSITY - argument określa gęstość mgły
- GL_FOG_COLOR - argument określa kolor mgły
- GL_FOG_INDEX - argument określa indeksowany kolor mgły

Mgła

Tworzenie mgły

```
float kolor[] = {0.0f, 0.5f, 0.5f, 1.0f};
float gestosc = 0.25f;

glFogi(GL_FOG_MODE, GL_EXP);
glFogf(GL_FOG_DENSITY, gestosc);
glFogiv(GL_FOG_COLOR, kolor);
```

Mgła

Tworzenie mgły

```
float kolor[] = {0.0f, 0.5f, 0.5f, 1.0f};
float gestosc = 0.25f;

glFogi(GL_FOG_MODE, GL_EXP);
glFogf(GL_FOG_DENSITY, gestosc);
glFogiv(GL_FOG_COLOR, kolor);
```

Wartości dla argumentu *GL_FOG_MODE*

- GL_LINEAR – mgła zależy od argumentów GL_FOG_START oraz GL_FOG_END
- GL_EXP – mgła zależy od argumentu GL_FOG_DENSITY
- GL_EXP2 – mgła zależy od argumentu GL_FOG_DENSITY

Blending

Blending jest to **mieszanie kolorów**. W przypadku gdy dwa obiekty *(nie koniecznie znajdujące się w tej samej odległości od kamery)* nachodzą na siebie istnieje możliwość połączenia w pewien sposób ich wspólnych obszarów.

Blending

Blending jest to **mieszanie kolorów**. W przypadku gdy dwa obiekty *(nie koniecznie znajdujące się w tej samej odległości od kamery)* nachodzą na siebie istnieje możliwość połączenia w pewien sposób ich wspólnych obszarów. Aby włączyć blending w OpenGL należy ustawić odpowiedni parametr maszyny stanu.

Blending

Blending jest to **mieszanie kolorów**. W przypadku gdy dwa obiekty *(nie koniecznie znajdujące się w tej samej odległości od kamery)* nachodzą na siebie istnieje możliwość połączenia w pewien sposób ich wspólnych obszarów. Aby włączyć blending w OpenGL należy ustawić odpowiedni parametr maszyny stanu.

Włączenie blendingu

glEnable(GL_BLEND)

Blending

O tym jaki będzie wynikowy kolor zmieszanych części danego obiektu decyduje **funkcja mieszająca**.

Funkcja mieszająca

*Kolor Wyjściowy = Kolor Źródła * Współczynnik Mieszania Źródła + Kolor Celu * Współczynnik Mieszania Celu*

Blending

Kolor źródłowy określa piksele przesłaniające, natomiast **kolor celu** określa piksele przesłaniane. Do zmieszania kolorów dojdzie tylko wtedy, gdy *obiekt przesłaniający znajdzie się przed obiektem przesłanianym*.

Blending

Kolor źródłowy określa piksele przesłaniające, natomiast **kolor celu** określa piksele przesłaniane. Do zmieszania kolorów dojdzie tylko wtedy, gdy *obiekt przesłaniający znajdzie się przed obiektem przesłanianym*. Współczynnik mieszania ustawia się za pomocą funkcji **glBlendFunc**.

Blending

Kolor źródłowy określa piksele przesłaniające, natomiast **kolor celu** określa piksele przesłaniane. Do zmieszania kolorów dojdzie tylko wtedy, gdy *obiekt przesłaniający znajdzie się przed obiektem przesłanianym*. Współczynnik mieszania ustawia się za pomocą funkcji **glBlendFunc**.

Funkcje do ustawiania współczynnika mieszania kolorów

glBlendFunc(GLenum sfactor, GLenum dfactor)

Blending

Kolor źródłowy określa piksele przesłaniające, natomiast **kolor celu** określa piksele przesłaniane. Do zmieszania kolorów dojdzie tylko wtedy, gdy *obiekt przesłaniający znajdzie się przed obiektem przesłanianym*. Współczynnik mieszania ustawia się za pomocą funkcji **glBlendFunc**.

Funkcje do ustawiania współczynnika mieszania kolorów

glBlendFunc(GLenum sfactor, GLenum dfactor)

Funkcja przyjmuje dwa argumenty. Pierwszy argument określa **współczynnik mieszania źródła**, drugi argument **współczynnik mieszania celu**. Współczynniki te podaje się w postaci odpowiednich **stałych**.

Blending

Argumenty przyjmowane przez glBlendFunc		
Stała	**Wartość RGB**	**Kanał alfa**
GL_ZERO	(0,0,0)	0
GL_ONE	(1,1,1)	1
GL_SRC_COLOR	(sR,sG,sB)	sA
GL_ONE_MINUS_SRC_COLOR	(1,1,1)–(sR,sG,sB)	1-sA
GL_DST_COLOR	(dR,dG,dB)	dA
GL_ONE_MINUS_DST_COLOR	(1,1,1)–(dR,dG,dB)	1-dA
GL_SRC_ALPHA	(sA,sA,sA)	sA
GL_ONE_MINUS_SRC_ALPHA	(1,1,1)–(sA,sA,sA)	1-sA
GL_DST_ALPHA	(dA,dA,dA)	dA
GL_ONE_MINUS_DST_ALPHA	(1,1,1)–(dA,dA,dA)	1-dA

gdzie:

sR, **sG**, **sB**, **sA** – składowe współczynnika mieszania źródła (składowe koloru źródła)

dR, **dG**, **dB**, **dA** – składowe współczynnika mieszania celu (składowe koloru celu)

Przezroczystość

Przezroczystość ustawiamy za pomocą funkcji **glBlendFunc** z argumentami dla współczynnika źródła **GL_SRC_ALPHA** oraz współczynnika celu **GL_ONE_MINUS_SRC_ALPHA**.

Przezroczystość

Przezroczystość ustawiamy za pomocą funkcji **glBlendFunc** z argumentami dla współczynnika źródła **GL_SRC_ALPHA** oraz współczynnika celu **GL_ONE_MINUS_SRC_ALPHA**.

Ustawienie przezroczystości

```
glEnable(GL_DEPTH_TEST);
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

Przezroczystość

Przezroczystość ustawiamy za pomocą funkcji **glBlendFunc** z argumentami dla współczynnika źródła **GL_SRC_ALPHA** oraz współczynnika celu **GL_ONE_MINUS_SRC_ALPHA**.

Ustawienie przezroczystości

```
glEnable(GL_DEPTH_TEST);
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

Uwaga

Aby obiekty przezroczyste były widoczne, trzeba rysować je na **końcu**. Możemy obejść ten problem za pomocą funkcji **glDepthMask** wyłączając zapis do **bufora głębi** przed rysowaniem obiektów przezroczystych *(po wyrenderowaniu obiektów przywracamy zapis do bufora)*.

Mgła, blending

Przykładowy program


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 4



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