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 2



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. Typy danych oraz konwencja nazw funkcji
   - Typy danych
   - Konwencja nazw funkcji
2. Przykład
   - Kompilacja programu
   - Obsługa okna
   - Funkcje zwrotne
3. Organizacja oraz manipulowanie przestrzenią
   - Manipulowanie przestrzenią
   - Rzutowanie
   - Rzutowanie perspektywiczne
   - Rzutowanie ortogonalne
   - Kamera
   - Macierz rzutowania i modelowania
4. 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.

Typy danych

Biblioteka OpenGL oraz wszystkie biblioteki pomocnicze i użytkowe zaimplementowane są w języku C.

Typy danych

Biblioteka OpenGL oraz wszystkie biblioteki pomocnicze i użytkowe zaimplementowane są w języku C. W celu pełnej przenośności kodu źródłowego między różnymi platformami systemowymi i sprzętowymi wykorzystują one specjalne stworzone typy danych, które są typami **opakowującymi** standardowe typy danych języka C.

Typy danych

Typy danych		
Typy danych OpenGL	**Reprezentacja**	**Język C**
GLbyte	8-bit całkowita	signed char
GLshort	16-bit całkowita	short
GLint, GLsizei	32-bit całkowita	long
GLfloat, GLclampf	32-bit zmiennoprzecinkowa	float
GLdouble, GLclampd	64-bit zmiennoprzecinkowa	double
GLubyte, GLboolean	8-bit całkowita bez znaku	unsigned char
GLushort	16-bit całkowita bez znaku	unsigned short
GLuint, GLenum, GLbitfield	32-bit całkowita bez znaku	unsigned long

- **clamp** - *„color amplitude”* (wykorzystywane przy składaniu kolorów)

Typy danych

Typy danych		
Typy danych OpenGL	**Reprezentacja**	**Język C**
GLbyte	8-bit całkowita	signed char
GLshort	16-bit całkowita	short
GLint, GLsizei	32-bit całkowita	long
GLfloat, GLclampf	32-bit zmiennoprzecinkowa	float
GLdouble, GLclampd	64-bit zmiennoprzecinkowa	double
GLubyte, GLboolean	8-bit całkowita bez znaku	unsigned char
GLushort	16-bit całkowita bez znaku	unsigned short
GLuint, GLenum, GLbitfield	32-bit całkowita bez znaku	unsigned long

- **clamp** - *„color amplitude”* (wykorzystywane przy składaniu kolorów)
- **GLboolean** – typ logiczny (True, False)

Typy danych

Typy danych		
Typy danych OpenGL	**Reprezentacja**	**Język C**
GLbyte	8-bit całkowita	signed char
GLshort	16-bit całkowita	short
GLint, GLsizei	32-bit całkowita	long
GLfloat, GLclampf	32-bit zmiennoprzecinkowa	float
GLdouble, GLclampd	64-bit zmiennoprzecinkowa	double
GLubyte, GLboolean	8-bit całkowita bez znaku	unsigned char
GLushort	16-bit całkowita bez znaku	unsigned short
GLuint, GLenum, GLbitfield	32-bit całkowita bez znaku	unsigned long

- **clamp** - *„color amplitude”* (wykorzystywane przy składaniu kolorów)
- **GLboolean** – typ logiczny (True, False)
- **GLenum** – zmienna wyliczeniowa

Typy danych

Typy danych		
Typy danych OpenGL	**Reprezentacja**	**Język C**
GLbyte	8-bit całkowita	signed char
GLshort	16-bit całkowita	short
GLint, GLsizei	32-bit całkowita	long
GLfloat, GLclampf	32-bit zmiennoprzecinkowa	float
GLdouble, GLclampd	64-bit zmiennoprzecinkowa	double
GLubyte, GLboolean	8-bit całkowita bez znaku	unsigned char
GLushort	16-bit całkowita bez znaku	unsigned short
GLuint, GLenum, GLbitfield	32-bit całkowita bez znaku	unsigned long

- **clamp** - *„color amplitude”* (wykorzystywane przy składaniu kolorów)
- **GLboolean** – typ logiczny (True, False)
- **GLenum** – zmienna wyliczeniowa
- **GLbitfield** – pole bitowe

Konwencja nazw funkcji

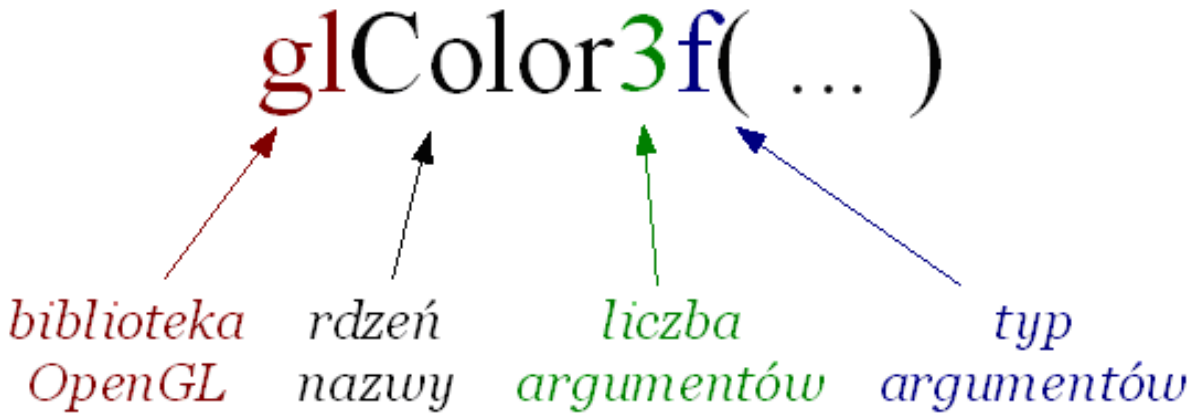
Wszystkie funkcje biblioteki **OpenGL** oraz **bibliotek pomocniczych** nazywane są zgodnie z określonym schematem. *Konwencja nazewnictwa* informuje o danej bibliotece, z której pochodzi funkcja oraz o *liczbie* i *typie* przyjmowanych argumentów.

Konwencja nazw funkcji

Wszystkie funkcje biblioteki **OpenGL** oraz **bibliotek pomocniczych** nazywane są zgodnie z określonym schematem. *Konwencja nazewnictwa* informuje o danej bibliotece, z której pochodzi funkcja oraz o *liczbie* i *typie* przyjmowanych argumentów. Nazwy wszystkich funkcji posiadają *rdzeń*. Rdzeń nazwy określa polecenie odpowiadające danej funkcji.

Konwencja nazw funkcji

Wszystkie funkcje biblioteki **OpenGL** oraz **bibliotek pomocniczych** nazywane są zgodnie z określonym schematem. *Konwencja nazewnictwa* informuje o danej bibliotece, z której pochodzi funkcja oraz o *liczbie* i *typie* przyjmowanych argumentów. Nazwy wszystkich funkcji posiadają *rdzeń*. Rdzeń nazwy określa polecenie odpowiadające danej funkcji.



Konwencja nazw funkcji

Przykładowe typy argumentów

- **f** - GLfloat
- **d** - GLdouble
- **i** - GLint
- **b** - GLbyte
- **s** - GLshort

Konwencja nazw funkcji

Przykładowe typy argumentów

- **f** - GLfloat
- **d** - GLdouble
- **i** - GLint
- **b** - GLbyte
- **s** - GLshort

Uwaga

Dodatkowo litera **„v”** informuje, że funkcja przyjmuje tablicę argumentów, np: *glVertex4fv(GLfloat arg[4])*

Kod źródłowy

Kod źródłowy w języku C

```
#include <stdlib.h>
#include <gl/gl.h>
#include <gl/glu.h>
#include <gl/glut.h>

// Glowna funkcja rysujaca
void rysuj()
{
  // Ustawienie koloru uzywanego do czyszczenia okna (R,G,B,Alfa)
  glClearColor(1.0f, 0.0f, 0.0f, 1.0f);
  // Wyczyszczenie bufora kolorow
  glClear(GL_COLOR_BUFFER_BIT);
  // Wykonanie polecen oproznienie kolejki renderowania
  glFlush();
}

int main(int argc, char *argv[])
{
  // Inicjalizacja biblioteki
  glutInit(&argc, argv);
  // Przygotowanie okna
  glutInitDisplayMode(GLUT_RGB);
  glutInitWindowSize(640, 480);
  glutCreateWindow("Hello OpenGL");
  // Ustawienie zwrotnej funkcji rysujacej
  glutDisplayFunc(rysuj);
  // Rozpoczecie glownej petli programu
  glutMainLoop();
  return(0);
}
```

Kompilacja programu

Kompilacja programu napisanego z wykorzystaniem biblioteki **OpenGL** oraz **bibliotek pomocniczych** uzależniona jest od rodzaju *języka programowania* oraz *architektury systemowej* na której kompilacja się odbywa.

Kompilacja programu

Kompilacja programu napisanego z wykorzystaniem biblioteki **OpenGL** oraz **bibliotek pomocniczych** uzależniona jest od rodzaju *języka programowania* oraz *architektury systemowej* na której kompilacja się odbywa.

Uwaga

Podczas kompilacji kodu źródłowego należy pamiętać o dołączeniu odpowiednich bibliotek oraz odpowiednich parametrów kompilatora.

Kompilacja programu

Kompilacja programu napisanego z wykorzystaniem biblioteki **OpenGL** oraz **bibliotek pomocniczych** uzależniona jest od rodzaju *języka programowania* oraz *architektury systemowej* na której kompilacja się odbywa.

Uwaga

Podczas kompilacji kodu źródłowego należy pamiętać o dołączeniu odpowiednich bibliotek oraz odpowiednich parametrów kompilatora. W przypadku biblioteki OpenGL są to następujące parametry: **–lopengl32 –lglu32 –glut32**

Obsługa okna

Inicjalizacja biblioteki użytkowej **GLUT**

glutInit(&argc, argv)

Obsługa okna

Inicjalizacja biblioteki użytkowej **GLUT**

glutInit(&argc, argv)

Inicjalizacja trybu wyświetlania oraz rozmiaru okna

glutInitDisplayMode(GLUT_RGB)
glutInitWindowSize(640, 480)
glutCreateWindow("Hello OpenGL")

Obsługa okna

Inicjalizacja biblioteki użytkowej **GLUT**

glutInit(&argc, argv)

Inicjalizacja trybu wyświetlania oraz rozmiaru okna

glutInitDisplayMode(GLUT_RGB)
glutInitWindowSize(640, 480)
glutCreateWindow("Hello OpenGL")

Inicjalizacji trybu wyświetlania z podwójnym buforem

glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE)

Obsługa okna

Inicjalizacja biblioteki użytkowej **GLUT**

glutInit(&argc, argv)

Inicjalizacja trybu wyświetlania oraz rozmiaru okna

glutInitDisplayMode(GLUT_RGB)
glutInitWindowSize(640, 480)
glutCreateWindow("Hello OpenGL")

Inicjalizacji trybu wyświetlania z podwójnym buforem

glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE)

Uwaga

Gdy używamy trybu wyświetlania z podwójnym buforem, należy zastąpić funkcję: **glFlush** na końcu głównej funkcji rysującej funkcją: **glutSwapBuffers**

Funkcje zwrotne

Funkcje zwrotne stanowią podstawowy element biblioteki użytkowej **GLUT**, wywoływane są automatycznie w momencie wystąpienia danego zdarzenia z okna głównego z wykorzystaniem tzw. **wywołań zwrotnych** *(ang. callback)*. Wyróżniamy kilka rodzajów funkcji zwrotnych.

Funkcje zwrotne

Funkcja rysująca

Funkcja zwrotna odpowiadająca za **rysowanie** *(generowanie obrazu)* w danym obszarze widzenia. Funkcję rysującą podpinamy (rejestrujemy) z wykorzystaniem funkcji:

Funkcje zwrotne

Funkcja rysująca

Funkcja zwrotna odpowiadająca za **rysowanie** *(generowanie obrazu)* w danym obszarze widzenia. Funkcję rysującą podpinamy (rejestrujemy) z wykorzystaniem funkcji:

Funkcja rejestrująca

glutDisplayFunc()

Funkcje zwrotne

Funkcja rysująca

Funkcja zwrotna odpowiadająca za **rysowanie** *(generowanie obrazu)* w danym obszarze widzenia. Funkcję rysującą podpinamy (rejestrujemy) z wykorzystaniem funkcji:

Funkcja rejestrująca

glutDisplayFunc()

Funkcja zmieniając rozmiar okna

Funkcja zwrotna odpowiadająca za **ustawienie rzutowania** oraz **manipulacje związane z oknem widoku**. Przyjmuje ona dwa argumenty będące rozmiarem nowego okna. Funkcję zmieniającą rozmiar okna podpinamy (rejestrujemy) z wykorzystaniem funkcji:

Funkcje zwrotne

Funkcja rysująca

Funkcja zwrotna odpowiadająca za **rysowanie** *(generowanie obrazu)* w danym obszarze widzenia. Funkcję rysującą podpinamy (rejestrujemy) z wykorzystaniem funkcji:

Funkcja rejestrująca

glutDisplayFunc()

Funkcja zmieniając rozmiar okna

Funkcja zwrotna odpowiadająca za **ustawienie rzutowania** oraz **manipulacje związane z oknem widoku**. Przyjmuje ona dwa argumenty będące rozmiarem nowego okna. Funkcję zmieniającą rozmiar okna podpinamy (rejestrujemy) z wykorzystaniem funkcji:

Funkcja rejestrująca

glutReshapeFunc()

Funkcje zwrotne

Funkcja obsługująca klawiaturę

Funkcja zwrotna odpowiadająca za obsługę **klawiatury**. Przyjmuje ona argument będący kodem ASCII wciśniętego klawisza. Funkcję obsługującą klawiaturę podpinamy (rejestrujemy) z wykorzystaniem funkcji:

Funkcje zwrotne

Funkcja obsługująca klawiaturę

Funkcja zwrotna odpowiadająca za obsługę **klawiatury**. Przyjmuje ona argument będący kodem ASCII wciśniętego klawisza. Funkcję obsługującą klawiaturę podpinamy (rejestrujemy) z wykorzystaniem funkcji:

Funkcja rejestrująca

glutKeyboardFunc()

Funkcje zwrotne

Funkcja obsługująca klawiaturę

Funkcja zwrotna odpowiadająca za obsługę **klawiatury**. Przyjmuje ona argument będący kodem ASCII wciśniętego klawisza. Funkcję obsługującą klawiaturę podpinamy (rejestrujemy) z wykorzystaniem funkcji:

Funkcja rejestrująca

glutKeyboardFunc()

Funkcja obsługująca klawisze specjalne

Funkcja zwrotna odpowiadająca za obsługę **klawiszy specjalnych**. Funkcję obsługującą klawisze specjalne podpinamy (rejestrujemy) z wykorzystaniem funkcji:

Funkcje zwrotne

Funkcja obsługująca klawiaturę

Funkcja zwrotna odpowiadająca za obsługę **klawiatury**. Przyjmuje ona argument będący kodem ASCII wciśniętego klawisza. Funkcję obsługującą klawiaturę podpinamy (rejestrujemy) z wykorzystaniem funkcji:

Funkcja rejestrująca

glutKeyboardFunc()

Funkcja obsługująca klawisze specjalne

Funkcja zwrotna odpowiadająca za obsługę **klawiszy specjalnych**. Funkcję obsługującą klawisze specjalne podpinamy (rejestrujemy) z wykorzystaniem funkcji:

Funkcja rejestrująca

glutSpecialFunc()

Funkcje zwrotne

Funkcja bezczynności

Funkcja zwrotna wywoływana w czasie **bezczynności** programu. Funkcję podpinamy (rejestrujemy) z wykorzystaniem:

Funkcje zwrotne

Funkcja bezczynności

Funkcja zwrotna wywoływana w czasie **bezczynności** programu. Funkcję podpinamy (rejestrujemy) z wykorzystaniem:

Funkcja rejestrująca

glutIdleFunc()

Funkcje zwrotne

Funkcja bezczynności

Funkcja zwrotna wywoływana w czasie **bezczynności** programu. Funkcję podpinamy (rejestrujemy) z wykorzystaniem:

Funkcja rejestrująca

glutIdleFunc()

Funkcja zegarowa

Funkcja zwrotna wywoływana za podaną ilość milisekund. Funkcję podpinamy (rejestrujemy) z wykorzystaniem:

Funkcje zwrotne

Funkcja bezczynności

Funkcja zwrotna wywoływana w czasie **bezczynności** programu. Funkcję podpinamy (rejestrujemy) z wykorzystaniem:

Funkcja rejestrująca

glutIdleFunc()

Funkcja zegarowa

Funkcja zwrotna wywoływana za podaną ilość milisekund. Funkcję podpinamy (rejestrujemy) z wykorzystaniem:

Funkcja rejestrująca

glutTimerFunc()

Układy współrzędnych

Typy układów

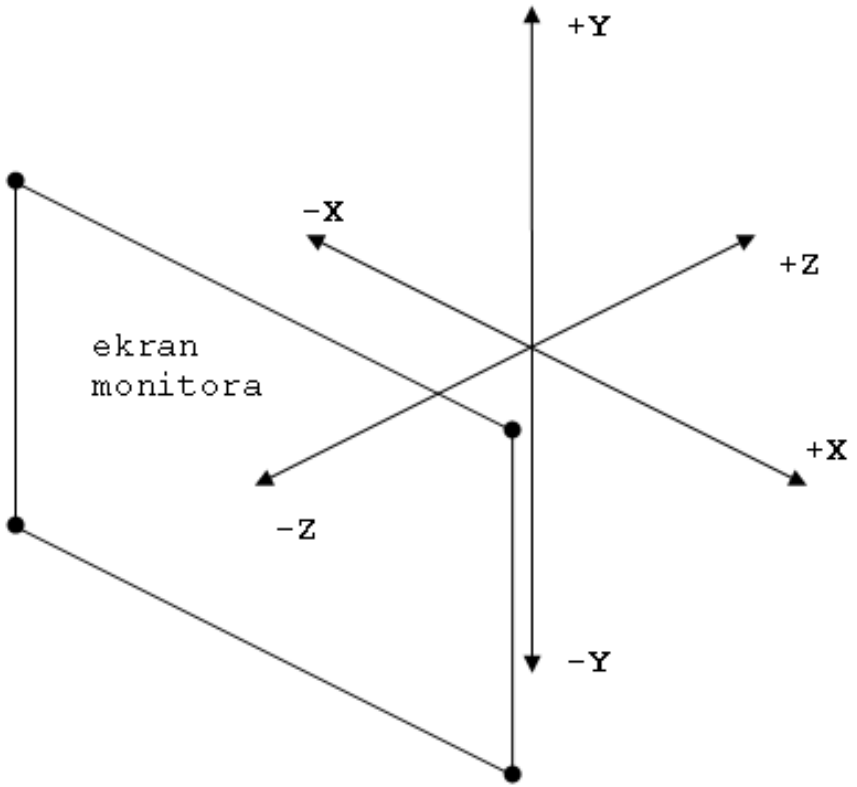
- **Lewoskrętny** układ współrzędnych

Układy współrzędnych

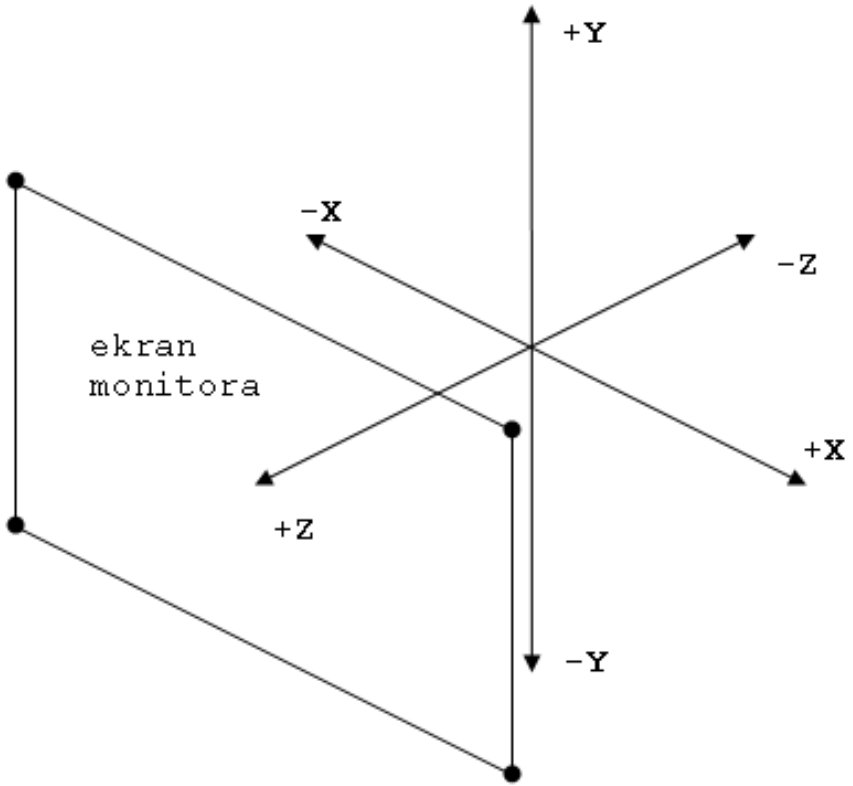
Typy układów

- **Lewoskrętny** układ współrzędnych
- **Prawoskrętny** układ współrzędnych

Lewoskrętny układ współrzędnych



Prawoskrętny układ współrzędnych



Manipulowanie przestrzenią

Przedstawione układy współrzędnych różnią się między sobą kierunkiem osi „*Z*" w układzie kartezjańskim trójwymiarowym. W bibliotece **OpenGL** obowiązuje układ **prawoskrętny** *(Direct3D umożliwia korzystanie z obydwu typów układów współrzędnych)*. Fizyczne zamienienie jednego układu w drugi jest niemożliwe.

Manipulowanie przestrzenią

Przedstawione układy współrzędnych różnią się między sobą kierunkiem osi „*Z*” w układzie kartezjańskim trójwymiarowym. W bibliotece **OpenGL** obowiązuje układ **prawoskrętny** *(Direct3D umożliwia korzystanie z obydwu typów układów współrzędnych)*. Fizyczne zamienienie jednego układu w drugi jest niemożliwe.

Uwaga

W **OpenGL** istnieje możliwość zamiany układu **lewoskrętnego** na **prawoskrętny** i na odwrót, poprzez umieszczanie przed każdą współrzędną wierzchołka osi „*Z*” znaku minusa.

Rzutowanie

Rzutowanie jest operacją polegającą na tym, aby odpowiednie piksele na płaskim ekranie monitora były wyświetlane w taki sposób, by sprawiać wrażenie trójwymiarowej głębi *(przestrzeni 3D)*.

Rzutowanie

Rzutowanie jest operacją polegającą na tym, aby odpowiednie piksele na płaskim ekranie monitora były wyświetlane w taki sposób, by sprawiać wrażenie trójwymiarowej głębi *(przestrzeni 3D)*. Rzutowanie również pozwala tworzyć przestrzeń dwuwymiarową *(przestrzeń 2D)* dzięki odpowiednim przekształceniom.

Rzutowanie

Rzutowanie jest operacją polegającą na tym, aby odpowiednie piksele na płaskim ekranie monitora były wyświetlane w taki sposób, by sprawiać wrażenie trójwymiarowej głębi *(przestrzeni 3D)*. Rzutowanie również pozwala tworzyć przestrzeń dwuwymiarową *(przestrzeń 2D)* dzięki odpowiednim przekształceniom.

Typy rzutowań

- **Rzutowanie perspektywiczne** *(wykorzystywane w grach 3D)*

Rzutowanie

Rzutowanie jest operacją polegającą na tym, aby odpowiednie piksele na płaskim ekranie monitora były wyświetlane w taki sposób, by sprawiać wrażenie trójwymiarowej głębi *(przestrzeni 3D)*. Rzutowanie również pozwala tworzyć przestrzeń dwuwymiarową *(przestrzeń 2D)* dzięki odpowiednim przekształceniom.

Typy rzutowań

- **Rzutowanie perspektywiczne** *(wykorzystywane w grach 3D)*
- **Rzutowanie ortogonalne** *(wykorzystywane w programach typu CAD)*

Rzutowanie

Rzutowanie jest operacją polegającą na tym, aby odpowiednie piksele na płaskim ekranie monitora były wyświetlane w taki sposób, by sprawiać wrażenie trójwymiarowej głębi *(przestrzeni 3D)*. Rzutowanie również pozwala tworzyć przestrzeń dwuwymiarową *(przestrzeń 2D)* dzięki odpowiednim przekształceniom.

Typy rzutowań

- **Rzutowanie perspektywiczne** *(wykorzystywane w grach 3D)*
- **Rzutowanie ortogonalne** *(wykorzystywane w programach typu CAD)*

Uwaga

OpenGL oraz **biblioteki pomocnicze** wyposażone są w specjalne funkcje, dzięki którym można uzyskać odpowiedni efekt rzutowania.

Rzutowanie perspektywiczne

W rzutowaniu **perspektywicznym** obiekty położone dalej od kamery *(obserwatora)* są mniejsze od tych położonych bliżej. OpenGL udostępnia specjalną funkcję, która tworzy odpowiednią macierz perspektywy.

Rzutowanie perspektywiczne

W rzutowaniu **perspektywicznym** obiekty położone dalej od kamery *(obserwatora)* są mniejsze od tych położonych bliżej. OpenGL udostępnia specjalną funkcję, która tworzy odpowiednią macierz perspektywy.

Funkcja odpowiedzialna za rzutowanie perspektywiczne

gluPerspective(GLdouble n, GLdouble a, GLdouble zN, GLdouble zF)

Rzutowanie perspektywiczne

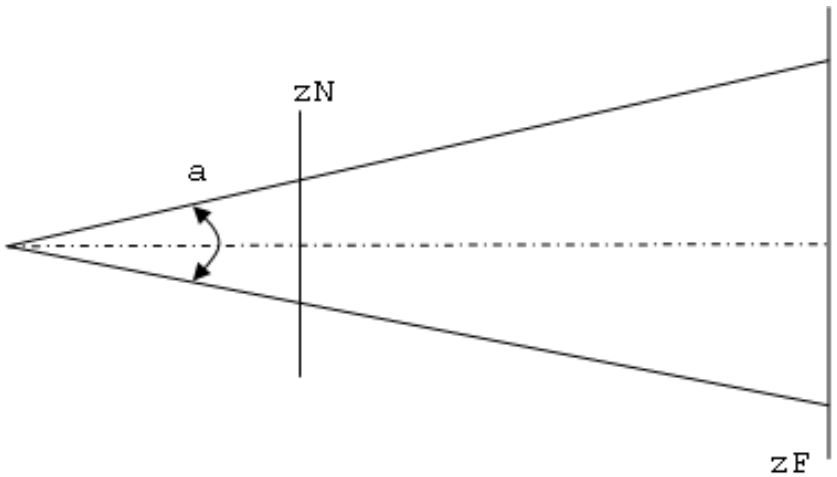
Funkcja przyjmuje cztery argumenty. Pierwszy argument określa kąt widzenia (w stopniach, zalecana wartość 60 stopni) w pionie. Drugi argument określa stosunek szerokości do wysokości okna (zazwyczaj podaje się szerokość okna, w którym wyświetlana jest grafika podzieloną przez wysokość). Dwa ostatnie argumenty określają granicę przednią i tylną, z reguły argument „zN" przyjmuje wartość 1.0.

Rzutowanie perspektywiczne

Widok perspektywiczny tworzy **stożek widzenia**, którego granicami są właśnie granica **przednia** i **tylna**. Wszystko co jest w stożku pojawia się na ekranie, obiekty znajdujące się poza stożkiem nie są wyświetlane.

Rzutowanie perspektywiczne

Widok perspektywiczny tworzy **stożek widzenia**, którego granicami są właśnie granica **przednia** i **tylna**. Wszystko co jest w stożku pojawia się na ekranie, obiekty znajdujące się poza stożkiem nie są wyświetlane.



Rzutowanie perspektywiczne

Funkcja **gluPerspective** nie należy do biblioteki OpenGL, lecz jest części biblioteki pomocniczej GLU.

Rzutowanie perspektywiczne

Funkcja **gluPerspective** nie należy do biblioteki OpenGL, lecz jest części biblioteki pomocniczej GLU. Oryginalną funkcją służącą do ustawiania **rzutowania perspektywicznego** w bibliotece OpenGL jest **glFrustum**, gluPerspective jest jedynie nakładką na tę funkcję.

Rzutowanie perspektywiczne

Funkcja **gluPerspective** nie należy do biblioteki OpenGL, lecz jest części biblioteki pomocniczej GLU. Oryginalną funkcją służącą do ustawiania **rzutowania perspektywicznego** w bibliotece OpenGL jest **glFrustum**, gluPerspective jest jedynie nakładką na tę funkcję.

Oryginalna funkcja odpowiedzialna za rzutowanie perspektywiczne

glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zN, GLdouble zF)

Rzutowanie perspektywiczne

Funkcja przyjmuje sześć argumentów, pierwsze dwa argumenty określa prawą i lewą współrzędną pionowej linii obcinania.

Rzutowanie perspektywiczne

Funkcja przyjmuje sześć argumentów, pierwsze dwa argumenty określa prawą i lewą współrzędną pionowej linii obcinania. Kolejne dwa argumenty określają górna i dolną współrzędną poziomej linii obcinania. Dwa ostatnie argumenty określają granicę przednią i tylną.

Rzutowanie ortogonalne

Nazywane inaczej **rzutowaniem prostokątnym** lub **równoległym**. Polega ono na tym, że każdy obiekt znajdujący się w obszarze rysowania niezależnie od swojej odległości od kamery *(obserwatora)* ma dokładnie tę samą wielkość.

Rzutowanie ortogonalne

Nazywane inaczej **rzutowaniem prostokątnym** lub **równoległym**. Polega ono na tym, że każdy obiekt znajdujący się w obszarze rysowania niezależnie od swojej odległości od kamery *(obserwatora)* ma dokładnie tę samą wielkość. W OpenGL dostępna jest specjalna funkcja, która tworzy rzutowanie ortogonalne.

Rzutowanie ortogonalne

Nazywane inaczej **rzutowaniem prostokątnym** lub **równoległym**. Polega ono na tym, że każdy obiekt znajdujący się w obszarze rysowania niezależnie od swojej odległości od kamery *(obserwatora)* ma dokładnie tę samą wielkość. W OpenGL dostępna jest specjalna funkcja, która tworzy rzutowanie ortogonalne.

Funkcja odpowiedzialna za rzutowanie ortogonalne

glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far)

Rzutowanie ortogonalne

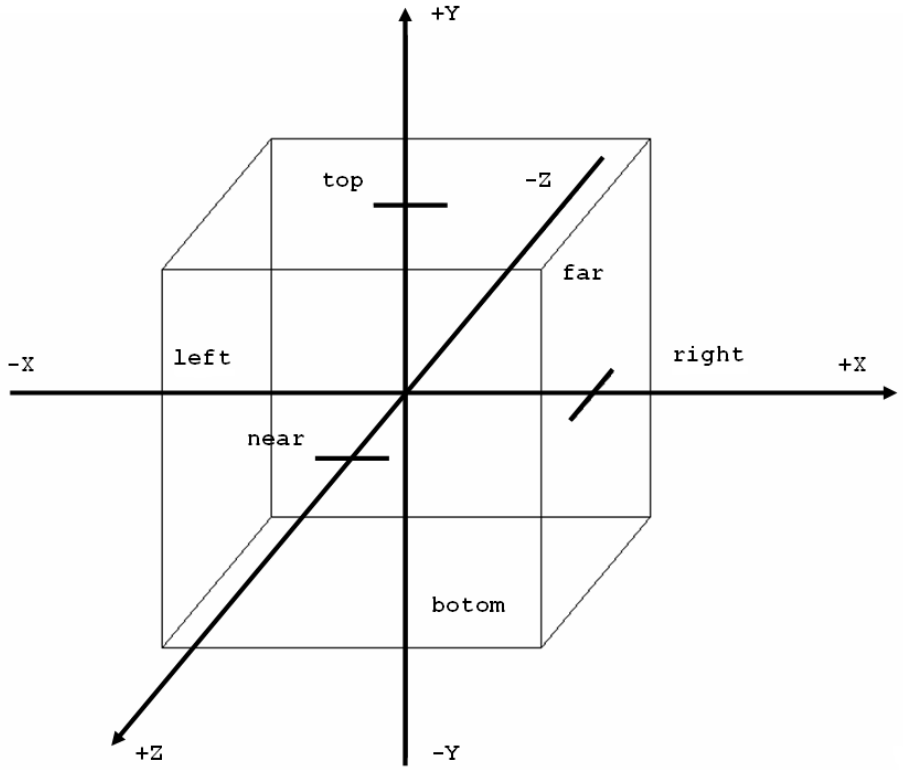
Uwaga

Funkcja glOrtho tworzy tzw. **bryłę widzenia** *(bryłę obcinania)*. Argumenty *„left”* oraz *„right”* określają szerokość bryły, argumenty *„bottom”* oraz *„top”* jej wysokość. Dwa ostatnie argumenty *„near”* i *„far”* określają głębię.

Rzutowanie ortogonalne

Uwaga

Funkcja glOrtho tworzy tzw. **bryłę widzenia** *(bryłę obcinania)*. Argumenty *„left”* oraz *„right”* określają szerokość bryły, argumenty *„bottom”* oraz *„top”* jej wysokość. Dwa ostatnie argumenty *„near”* i *„far”* określają głębię.



Rzutowanie ortogonalne

W obrębie bryły widzenia tworzone są obiekty wyświetlane na ekranie. To czy obiekt pojawi się na ekranie zależy od tego czy znajduje się on w obszarze bryły.

Rzutowanie ortogonalne

W obrębie bryły widzenia tworzone są obiekty wyświetlane na ekranie. To czy obiekt pojawi się na ekranie zależy od tego czy znajduje się on w obszarze bryły. Rzutowanie ortogonalne wykorzystywane jest często w grach **2D** oraz programach typu **CAD/CAM**.

Kamera

Oprócz samego **rzutowania** w OpenGL bardzo ważną rzeczą jest kamera. Kamera określa położenie **obserwatora** *(w przestrzeni 3D)* względem obserwowanych obiektów. OpenGL posiada funkcję tworzącą tzw. **macierz kamery**.

Kamera

Oprócz samego **rzutowania** w OpenGL bardzo ważną rzeczą jest kamera. Kamera określa położenie **obserwatora** *(w przestrzeni 3D)* względem obserwowanych obiektów. OpenGL posiada funkcję tworzącą tzw. **macierz kamery**.

Funkcja odpowiedzialna za ustawienie kamery

gluLookAt(GLdouble x, GLdouble y, GLdouble z, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz)

Kamera

Funkcja przyjmuje **dziewięć argumentów**. Pierwsze trzy *x,y,z* określają położenie kamery w przestrzeni trójwymiarowej.

Kamera

Funkcja przyjmuje **dziewięć argumentów**. Pierwsze trzy *x,y,z* określają położenie kamery w przestrzeni trójwymiarowej. Kolejne trzy argumenty *centerx*, *centery*, *centerz* określają punkt na który skierowana jest kamera *(patrzy kamera)*.

Kamera

Funkcja przyjmuje **dziewięć argumentów**. Pierwsze trzy *x,y,z* określają położenie kamery w przestrzeni trójwymiarowej. Kolejne trzy argumenty *centerx*, *centery*, *centerz* określają punkt na który skierowana jest kamera *(patrzy kamera)*. Ostatnie trzy *upx*, *upy*, *upz* określają wektor pionu kamery, dzięki tym argumentom możliwe jest obracanie kamerą.

Macierz rzutowania i modelowania

Każda pozycja punktu w przestrzeni trójwymiarowej w OpenGL obliczana jest za pomocą następującego wzoru.

Macierz rzutowania i modelowania

Każda pozycja punktu w przestrzeni trójwymiarowej w OpenGL obliczana jest za pomocą następującego wzoru.

Wzór na wyjściową pozycję punktu

*Wyjściowa pozycja punktu = Macierz rzutowania * Macierz modelowania * Wejściowa pozycja punktu*

Macierz rzutowania

Jak sama nazwa wskazuje **macierz rzutowania** *(projekcji)* związana jest z procesem rzutowania. Aby umożliwić rzutowanie **perspektywiczne** lub **ortogonalne** należy ustawić odpowiedni parametr **maszyny stanu**.

Macierz rzutowania

Jak sama nazwa wskazuje **macierz rzutowania** *(projekcji)* związana jest z procesem rzutowania. Aby umożliwić rzutowanie **perspektywiczne** lub **ortogonalne** należy ustawić odpowiedni parametr **maszyny stanu**. Służy do tego celu odpowiednia funkcja oraz parametr **GL_PROJECTION**.

Funkcja odpowiedzialna za ustawienie odpowiedniej macierzy

glMatrixMode(GL_PROJECTION)

Macierz rzutowania

Jak sama nazwa wskazuje **macierz rzutowania** *(projekcji)* związana jest z procesem rzutowania. Aby umożliwić rzutowanie **perspektywiczne** lub **ortogonalne** należy ustawić odpowiedni parametr **maszyny stanu**. Służy do tego celu odpowiednia funkcja oraz parametr **GL_PROJECTION**.

Funkcja odpowiedzialna za ustawienie odpowiedniej macierzy

glMatrixMode(GL_PROJECTION)

Uwaga

Wywołanie funkcji **glMatrixMode** z parametrem **GL_PROJECTION** informuje maszynę stanu biblioteki OpenGL, że następnie wywołane funkcję będą oddziaływać bezpośrednio na macierzy rzutowania.

Macierz rzutowania

Jak sama nazwa wskazuje **macierz rzutowania** *(projekcji)* związana jest z procesem rzutowania. Aby umożliwić rzutowanie **perspektywiczne** lub **ortogonalne** należy ustawić odpowiedni parametr **maszyny stanu**. Służy do tego celu odpowiednia funkcja oraz parametr **GL_PROJECTION**.

Funkcja odpowiedzialna za ustawienie odpowiedniej macierzy

glMatrixMode(GL_PROJECTION)

Uwaga

Wywołanie funkcji **glMatrixMode** z parametrem **GL_PROJECTION** informuje maszynę stanu biblioteki OpenGL, że następnie wywołane funkcję będą oddziaływać bezpośrednio na macierzy rzutowania. Macierz rzutowania (projekcji) potrzebna jest by dodać odpowiednią **perspektywę**.

Macierz rzutowania

Zawsze po przejściu na operacje związane z **macierzą rzutowania**, należy ją zresetować. Służy do tego celu bezparametrowa funkcja **glLoadIdentity**.

Macierz rzutowania

Zawsze po przejściu na operacje związane z **macierzą rzutowania**, należy ją zresetować. Służy do tego celu bezparametrowa funkcja **glLoadIdentity**.Funkcja odpowiedzialna jest za załadowanie do macierzy rzutowania tzw. *macierzy tożsamości*.

Macierz rzutowania

Zawsze po przejściu na operacje związane z **macierzą rzutowania**, należy ją zresetować. Służy do tego celu bezparametrowa funkcja **glLoadIdentity**.Funkcja odpowiedzialna jest za załadowanie do macierzy rzutowania tzw. *macierzy tożsamości*.

Funkcja odpowiedzialna za załadowanie macierzy tożsamości

glLoadIdentity()

Macierz rzutowania

Zawsze po przejściu na operacje związane z **macierzą rzutowania**, należy ją zresetować. Służy do tego celu bezparametrowa funkcja **glLoadIdentity**.Funkcja odpowiedzialna jest za załadowanie do macierzy rzutowania tzw. *macierzy tożsamości*.

Funkcja odpowiedzialna za załadowanie macierzy tożsamości

glLoadIdentity()

Uwaga

Dopiero po załadowaniu macierzy tożsamości można ustawić rzutowanie **perspektywiczne** lub **ortogonalne** za pomocą odpowiednich funkcji *gluPerspective*, *glFrustum*, *glOrtho*.

Macierz modelowania

W skład macierzy **modelowania** wchodzą dodatkowo **dwie macierze** (traktowane jako jedna całość).

Macierz modelowania

W skład macierzy **modelowania** wchodzą dodatkowo **dwie macierze** (traktowane jako jedna całość). Pierwszą z nich jest **macierz przekształceń** *(związana z przekształceniami w przestrzeni 3D: translacją, rotacją i skalowaniem)*, drugą zaś **macierz kamery** tworzona i ustawiana za pomocą funkcji **gluLookAt**. Aby umożliwić operacje na macierzy modelowania należy ustawić odpowiedni parametr **maszyny stanu**.

Macierz modelowania

W skład macierzy **modelowania** wchodzą dodatkowo **dwie macierze** (traktowane jako jedna całość). Pierwszą z nich jest **macierz przekształceń** *(związana z przekształceniami w przestrzeni 3D: translacją, rotacją i skalowaniem)*, drugą zaś **macierz kamery** tworzona i ustawiana za pomocą funkcji **gluLookAt**. Aby umożliwić operacje na macierzy modelowania należy ustawić odpowiedni parametr **maszyny stanu**. Służy do tego celu odpowiednia funkcja oraz parametr **GL_MODELVIEW**.

Funkcja odpowiedzialna za ustawienie odpowiedniej macierzy

glMatrixMode(GL_MODELVIEW)

Przykład ustawienia rzutowania ortogonalnego

Kod źródłowy w języku C

```
#include <stdlib.h>
#include <gl/gl.h>
#include <gl/glu.h>
#include <gl/glut.h>

void rysuj()
{
  // Tu umieszczamy kod, ktory zostanie narysowany w obszarze widzenia
}

void zmien_rozmiar(int w, int h)
{
  glViewport(0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(0.0, (GLdouble) w, 0.0, (GLdouble) h);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}

int main(int argc,char *argv[])
{
  // Ustawienie zwrotnej funkcji rysujacej
  glutDisplayFunc(rysuj);
  // Ustawienie zwrotnej funkcji odpowiedzialnej za zmiane rozmiaru okna
  glutReshapeFunc(zmien_rozmiar);
  return(0);
}
```

Przykład ustawienia rzutowania perspektywicznego

Kod źródłowy w języku C

```
#include <stdlib.h>
#include <gl/gl.h>
#include <gl/glu.h>
#include <gl/glut.h>

void rysuj()
{
  // Tu umieszczamy kod, ktory zostanie narysowany w obszarze widzenia
}

void zmien_rozmiar(int w, int h)
{
  glViewport(0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(45.0f, (GLfloat) w/ (GLfloat) h, 0.1f, 100.0f);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}

int main(int argc,char *argv[])
{
  // Ustawienie zwrotnej funkcji rysujacej
  glutDisplayFunc(rysuj);
  // Ustawienie zwrotnej funkcji odpowiedzialnej za zmiane rozmiaru okna
  glutReshapeFunc(zmien_rozmiar);
  return(0);
}
```

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 2



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