Instrukcja laboratoryjna **0**	**Programowanie grafiki komputerowej**
	Temat: Przypomnienie OpenGL i GLUT
	Przygotował: dr inż. Grzegorz Łukawski, mgr inż. Maciej Lasota, dr inż. Tomasz Michno

1 Wstęp teoretyczny

OpenGL (**Open G**raphics **L**ibrary, http://www.opengl.org/) jest specyfikacją API służącą do generowania grafiki 3D, używaną obecnie w wielu aplikacjach, m. in. takich jak: aplikacje CAD, gry, programy do modelowania 3D i tworzenia efektów w filmach. OpenGL działa w architekturze klient-serwer, dzięki czemu implementacja (zależna od sprzętu) jest oddzielona od samej aplikacji korzystającej z API. Implementację specyfikacji zazwyczaj dostarczają producenci kart graficznych razem ze sterownikami (istnieje też zupełnie niezależna implementacja Mesa3D).

Obecnie OpenGL rozwijane jest wspólnie przez większość firm związanych z produkcją kart graficznych i grafiką w ramach Khronos Group (od lipca 2006, wcześniej w ramach zrzeszenia ARB). W skład grupy wchodzą m. in. nVidia, AMD/ATI, Intel, SGI, Google. Swój początek OpenGL miało w formie biblioteki IRIS GL stworzonej przez firmę Silicon Graphics na potrzeby stacji roboczych IRIS. OpenGL jest rezultatem prac firmy nad poprawą przenaszalności IRIS GL. W lipcu 1992 powstał pierwsza wersja OpenGL 1.0. Obecnie biblioteka ta jest w pełni otwarta i dostępna dla wielu platform systemowych. Umożliwia rysowanie punktów, odcinków i wielokątów w trzech wymiarach, ponadto obsługuje oświetlenie, cieniowanie, mapowanie tekstur, animacje oraz dodatkowe efekty specjalne. OpenGL nie zawiera żadnych funkcji przeznaczonych do zarządzania oknami, interakcją z użytkownikiem czy operacji wejścia/wyjścia, dlatego należy je zaimplementować własnoręcznie lub skorzystać z gotowych bibliotek pomocniczych.

GLUT (Open**GL** **U**tility **T**oolkit) jest właśnie jedną z takich bibliotek. Udostępniany jest na większość systemów operacyjnych, pozwalając na napisanie jednego, uniwersalnego kodu aplikacji. Cecha ta przyczyniła się do bardzo dużej popularności GLUT'a. Biblioteka została napisana przez Marka J. Kilgarda w celu zwiększenia przenaszalności kodu oraz ułatwienia nauki OpenGL'a. Ponieważ GLUT nie został udostępniony na wolnej licencji (Kilgard nadal posiada prawa autorskie) powstało kilka projektów alternatywnych implementacji GLUT'a. Najważniejszym jest freeGLUT (http://freeglut.sourceforge.net/), który jest w pełni kompatybilny z oryginalną biblioteką GLUT.

1.1 Instalacja OpenGL + GLUT

1.1.1 System Windows

W celu zainstalowanie bibliotek OpenGL oraz GLUT w systemie operacyjnym Windows, należy skopiować odpowiednie pliki nagłówkowe oraz pliki bibliotek (*.lib,*.a,*.dll) do odpowiednich katalogów. Poniżej została przedstawiona tabelka prezentująca nazwy plików, które należy skopiować oraz ich docelowe lokalizacje.

Pliki	Lokalizacja
gl.h glu.h glut.h	\katalog kompilatora\include\GL
opengl32.lib (*.a) glu32.lib (*.a) glut32.lib (*.a)	\katalog kompilatora\lib
opengl32.dll glu32.dll glut32.dll	\katalog systemowy\system32\

W celu kompilacji kodu źródłowego należy dodać następujące parametry (do sekcji Linker Settings->Other Linker Options):

-lopengl32 -lglu32 -glut32

1.1.2 System Linux

W systemie Linux najlepiej jest zainstalować bibliotekę freeGLUT oraz sterowniki do karty graficznej obsługujące 3D (np. udostępniane przez producenta lub społecznościowe). W celu kompilacji najlepiej jest dodać następujące parametry (do sekcji Linker Settings->Other Linker Options):

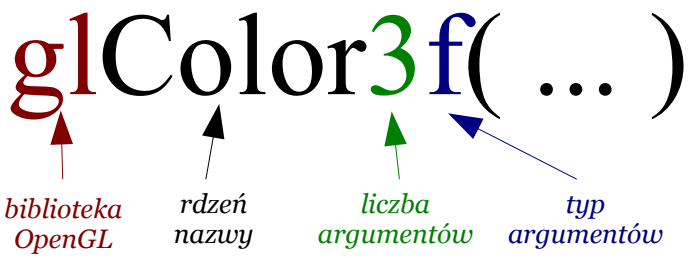
-lglut -lGL -lGLU -lX11 -lXmu -lXi -lm

1.2 Konwencja nazw funkcji i typy danych

Wszystkie nazwy funkcji wykorzystywane w bibliotekach OpenGL i GLUT posiadają przedrostki pozwalające na identyfikację, z której biblioteki pochodzą:

- gl dla OpenGL
- glut dla GLUT

Przykładowo:



Dodatkowa litera v po typie argumentu informuje, że funkcja przyjmuje tablicę argumentów, np.:
glVertex4fv(GLfloat arg[4]);
Biblioteka OpenGL, w celu pełnej przenośności kodu między różnymi platformami wprowadza swoje własne typy danych, np.:

Typ	Odpowiednik w OpenGL	Skrót w funkcji
int	GLint	i
float	GLfloat	f
double	GLdouble	d
byte (char)	GLbyte	b
short	GLshort	s

1.3 Szkielet programu

```
#include <stdlib.h>
#include <GL/glut.h>    // dodanie biblioteki GLUT

void wyswietlaj()
{
      // Tu umieszczamy kod, który zostanie
      // narysowany w obszarze widzenia
}

void rozmiar(int w, int h)
{
      // Tu umieszczamy kod odpowiedzialny
      // za ustawienie rzutowania oraz okna widoku
}

int main(int argc,char *argv[])
{
      // Tu umieszczamy kod odpowiedzialny zainicjowanie
      // biblioteki GLUT oraz przygotowanie okna do rysowania
}
```

1.4 Najważniejsze funkcje biblioteki GLUT

Nazwa funkcji	Opis
glutInit	*Funkcja inicjuje bibliotekę GLUT*
glutInitDisplayMode	*Funkcja inicjuje tryb wyświetlania zazwyczaj przyjmuje dwa parametry GLUT_DOUBLE oraz GLUT_RGB*
glutInitWindowSize	*Funkcja inicjuje rozmiar nowo tworzonego okna*
glutPositionWindow	*Funkcja określa położenia okna na ekranie monitora*
glutCreateWindow	*Funkcja tworzy nowe okno, jako parametr przyjmuje nazwę dla okna*
glutDisplayFunc	*Funkcja służy do podpięcia (rejestracji) funkcji rysujące, funkcja rysująca wywoływania jest za pomocą wywołania zwrotnego (ang. callback) z okna głównego*
glutReshapeFunc	*Funkcja służy do podpięcia (rejestracji) funkcji odpowiedzialnej za zmianę obszaru rysowani oraz rzutowania. Podpięta funkcja wywoływania jest za pomocą*
glutKeyboardFunc	*Funkcja służy do podpięcia (rejestracji) funkcji odpowiedzialnej za obsługę klawiatury.*
glutMainLoop	*Funkcja uruchamia pętle główną biblioteki GLUT, pelta ta czeka na wystąpienie odpowiedniego wyjątku.*
glutSwapBuffers	*Funkcja wykorzystywana w przypadku ustawienia podwójnego bufora GLUT_DOUBLE, służy do przełączania bufora*
glutFullScreen	*Funkcja ustawia tryb pełnoekranowy dla okna*

1.5 Podstawy rysowania

Rysowanie w aplikacjach korzystających z GLUT'a odbywa się zawsze w oddzielnej funkcji typu void, nieprzyjmującej parametrów. Podczas konfigurowania GLUT'a należy poinformować go, która z funkcji będzie służyła do wyświetlania (o tym w następnym podrozdziale).

Standardowy proces wyświetlania obrazu składa się z następujących kroków:

- **ustawienie koloru, jakim będzie wypełniany ekran przy czyszczeniu** (R, G, B, alfa),

  np. kolor czarny:

```
glClearColor(0.0, 0.0, 0.0, 1.0);
```

- **wyczyszczenie buforów**

  na początek wystarczy wyczyścić jedynie bufor kolorów:

```
glClear(GL_COLOR_BUFFER_BIT);
```

- **narysowanie obiektu/obiektów**

  np. narysowanie kwadratu polega na podaniu czterech wierzchołków (glVertex2f(x, y)) i wybraniu typu obiektu jako GL_QUADS. Więcej informacji na ten temat zostanie podanych w dalszych instrukcjach.

```
glColor3f(1.0, 0.0, 0.0);  // ustawienie koloru na czerwony
glBegin(GL_QUADS);
    glVertex2f(270, 190);
    glVertex2f(370, 190);
    glVertex2f(370, 290);
    glVertex2f(270, 290);
glEnd();
```

  W powyższym przykładzie prostokąt jest rysowany z pominięciem współrzędnej z. Dla rysowania obrazu w pełni trójwymiarowego należy użyć glVertex3f(x, y, z).

  Trójkąt można narysować za pomocą GL_TRIANGLES:

```
glBegin(GL_TRIANGLES);
    glVertex2f(270, 190);
    glVertex2f(370, 190);
    glVertex2f(370, 290);
glEnd();
```

- **wyświetlenie obrazu** – powyższe instrukcje oczekują w kolejce do wykonania, należy poinformować OpenGL, że skończyliśmy rysować i obraz można wyświetlić na ekran. Za wykonanie tej operacji odpowiedzialna jest funkcja glFlush(). W przypadku, gdy korzystamy z podwójnego buforowania (które jest zalecane), należy użyć funkcji glutSwapBuffers(), która podmieni bufory i wyświetli obraz na ekranie.

Biblioteka GLUT oprócz tworzenia okien i obsługi I/O posiada również funkcje wyświetlające na ekranie najczęściej używane bryły geometryczne stworzone z trójkątów (wystarczy je wywołać w części odpowiedzialnej za rysowanie obiektów, wszystkie figury są rysowane w środku ekranu):

Funkcja	Opis
void **glutSolidSphere(**radius, slices, stacks**)**	Rysuje wypełnioną sferę: • radius – GLdouble – promień sfery • slices - GLint – liczba południków • stacks – GLint – liczba równoleżników
void **glutWireSphere(radius, slices, stacks)**	Rysuje sferę w postaci druciaka
void **glutSolidCube(**size**)**	Rysuje wypełniony sześcian: • size – GLdouble - rozmiar
void **glutWireCube(**size**)**	Rysuje sześcian w postaci druciaka
void **glutSolidCone(**radius, height, slices, stacks**)**	Rysuje wypełniony stożek: • radius – GLdouble – promień • height – GLdouble – wysokość • slices - GLint – liczba południków • stacks – GLint – liczba równoleżników
void **glutWireCone(**radius, height, slices, stacks**)**	Rysuje stożek w postaci druciaka
void **glutSolidTorus(**inRadius, outRadius, nSides, rings**)**	Rysuje wypełniony torus (oponka, pączek z dziurą w środku): • inRadius – GLdouble – wewnętrzny promień • outRadius – GLdouble – zewnętrzny promień • nSides, rings – GLint – pełnią podobną rolę, co slices i stack w stożku i sferze
void **glutWireTorus(**inRadius, outRadius, nSides, rings**)**	Torus w postaci druciaka
void **glutSolidTeapot(**size**)**	Rysuje wypełniony czajnik/imbryk (teapot): • size – GLdouble - rozmiar
void **glutWireTeapot(**size**)**	Rysuje imbryk w postaci druciaka

Podstawy rysowania zostaną dokładniej omówione w następnych instrukcjach.

1.6 Obsługa klawiatury

Klawiatura obsługiwana jest podobnie jak rysowanie – w programie powinna zostać zadeklarowana funkcja o odpowiednich parametrach. Przy konfigurowaniu biblioteki GLUT należy ustalić, która funkcja będzie służyła do obsługi klawiszy.

Przykład (naciśnięcie klawisz ESC lub q będzie skutkowało wyjściem z programu):

```
void klawiatura(unsigned char key, int x, int y)
{
   switch (key)
   {
      case 27 :
      case 'q':
         exit(0);
      break;
   }
}
```

W przypadku, gdyby zaistniała potrzeba odmalowania ekranu po wciśnięciu klawisza, można użyć funkcji glutPostRedisplay().

1.7 Przykładowy program

```
#include <stdlib.h>
#include <GL/glut.h>

// funkcja wyświetlająca obraz
void display() {
      glClear(GL_COLOR_BUFFER_BIT);
      glColor3f(1.0, 0.0, 0.0);
      glBegin(GL_QUADS);
            glVertex2f(270, 190);
            glVertex2f(370, 190);
            glVertex2f(370, 290);
            glVertex2f(270, 290);
      glEnd();
      glutSwapBuffers();
}

// funkcja wywoływana podczas zmiany rozmiaru okna,
```

```
// funkcje w niej zawarte będą omówione w dalszych instrukcjach
void reshape(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();
}

// funkcja obsługi klawiatury
void klawiatura(unsigned char key, int x, int y)
{
    switch (key)
    {
        case 27 :
        case 'q':
            exit(0);
        break;
    }
}

int main(int argc, char *argv[]) {
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);  // ustawienie podwójnego
buforowania i kolorów RGB
      glutInitWindowSize(640, 480);    // rozmiar okna
      glutCreateWindow(argv[0]);
      glutDisplayFunc(display);        // ustawienie funkcji odpowiedzialnej za
rysowanie
      glutReshapeFunc(reshape);        // ustawienie funkcji wywoływanej przy
zmianie rozmiaru okna
      glutKeyboardFunc(klawiatura);    // ustawienie funkcji obsługi klawiatury
      glutMainLoop();                  // wejście do głównej pętli programu

      return 0;
}
```

2 Zadania

1. Zmodyfikuj przykładowy program tak, aby wyświetlał zieloną literę F.
2. Napisz program, który będzie wyświetlał podane w instrukcji bryły dostarczane przez bibliotekę GLUT (teapot itp.). Ustawienie, która z figur ma zostać wyświetlona powinno odbywać się za pomocą dowolnie wybranych klawiszy.