Terminy oddawania ćwiczeń:

Ćwiczenie nr 1 – Szkielet aplikacji wykorzystującej bibliotekę OpenGL dla środowiska Windows z wykorzystaniem Visual C++

Podstawowe zasady:

- Wszystkie wątki wykorzystują jeden OpenGL Rendering Context (hRC) <->GDI Device Context (hDC)
- Wiele RC może wykorzystywać w jednym oknie
- Tylko jeden RC może być aktywny w jednym wątku

Sposób rysowania w MFC:

- Ustawić windows' pixel format
- Utwórz RC
- Ustaw RC jako bieżący

Tworzenie aplikacji w środowisku Microsoft Visual C++/MFC[1] (na podstawie [1]):

1) Utworzyć nowy project "MFC AppWizard (exe)"; nazwa projektu: Ex1

Microsoft Visual Studio ver. 6.0	**Microsoft Visual Studio 2005**
	File->New->Project: Visual C++->MFC->MFC Applicatio, Name: Ex1, Solution Name: Ex1

2) Opcje kompilatora:

Microsoft Visual Studio ver. 6.0	**Microsoft Visual Studio 2005**
Single Document Interface, Database support: None Compond Document Support: None Docking Toolbar: OFF (optional) Initial Status Bar: OFF (optional) Printing an Print Preview: OFF Context-Sensitive Help: OFF (optional) 3D Controls: ON (optional) Use the MFC Standard style of project Generate Source File Comments: Yes Use the MFC library as a shared DLL.	Single Document Interface, MFC Standard, „MFC In a static library" lub „MFC In a shared DLL", Compound document support: None Database support: None Initial Status Bar: OFF (optional) Toolbars: None Context-sensitive Help: None ActiveX controls: None

3) Opcje linkera

Microsoft Visual Studio ver. 6.0	**Microsoft Visual Studio 2005**
Project-Settings->Link->General-> Object/Library Modules: opengl32.lib glu32.lib glaux.lib	Project->Properties: Linker->Input->Additional dependencies: opengl32.lib glu32.lib glaux.lib

4) zmiany w pliku "stdafx.h"

```
#define VC_EXTRALEAN // Exclude rarely-used stuff from Windows
// headers
#include <afxwin.h> // MFC core and standard components
```

[1] Przykłady tworzenia aplikacji w innych środowiskach mogą być wykorzystane: GLX dla X Windows w [5], [7], Linux [9]; WGL- dla Windows 9x/NT/... , patrz MSDN; PGL – dla IBM OS/2 WARP [8]; GLUT – inne [4]; MESA dla środowiska Java.

```
#include <afxext.h> // MFC extensions
#include <gl\gl.h>
#include <gl\glu.h>
//#include <gl\glaux.h>
//#pragma comment(lib,opengl32.lib) // zamiast pkt.3 (w niektórych implementacjach)
//#pragma comment(lib,glu32.lib)    //zamiast pkt.3 (w niektórych implementacjach)
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h> // MFC support for Windows 95 Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT
```

5) Zmiany w klasie

```
BOOL CEx1View::PreCreateWindow(CREATESTRUCT& cs){
      cs.style |= (WS_CLIPCHILDREN | WS_CLIPSIBLINGS);
      return CView::PreCreateWindow(cs);
}
```

6) Nowa metoda (tekst pogrubiony – nowy tekst do samodzielnego wprowadzenia lub korekty):

```
BOOL CEx1View::SetWindowPixelFormat(HDC hDC){
PIXELFORMATDESCRIPTOR pixelDesc;

        pixelDesc.nSize = sizeof(PIXELFORMATDESCRIPTOR);
        pixelDesc.nVersion = 1;
        pixelDesc.dwFlags = PFD_DRAW_TO_WINDOW |
                PFD_DRAW_TO_BITMAP | PFD_SUPPORT_OPENGL |
                PFD_SUPPORT_GDI | PFD_STEREO_DONTCARE;
        pixelDesc.iPixelType = PFD_TYPE_RGBA;
        pixelDesc.cColorBits = 32;
        pixelDesc.cRedBits = 8;         pixelDesc.cRedShift = 16;
        pixelDesc.cGreenBits = 8;       pixelDesc.cGreenShift = 8;
        pixelDesc.cBlueBits = 8;        pixelDesc.cBlueShift = 0;
        pixelDesc.cAlphaBits = 0;       pixelDesc.cAlphaShift = 0;
        pixelDesc.cAccumBits = 64;
        pixelDesc.cAccumRedBits = 16;
        pixelDesc.cAccumGreenBits = 16;
        pixelDesc.cAccumBlueBits = 16;
        pixelDesc.cAccumAlphaBits = 0;
        pixelDesc.cDepthBits = 32;
        pixelDesc.cStencilBits = 8;
        pixelDesc.cAuxBuffers = 0;
        pixelDesc.iLayerType = PFD_MAIN_PLANE;
        pixelDesc.bReserved = 0;        pixelDesc.dwLayerMask = 0;
        pixelDesc.dwVisibleMask = 0;    pixelDesc.dwDamageMask = 0;
        m_GLPixelIndex = ChoosePixelFormat( hDC, &pixelDesc);
        if (m_GLPixelIndex==0){ // Let's choose a default index.
                m_GLPixelIndex = 1;
                if (DescribePixelFormat(hDC, m_GLPixelIndex,
                        sizeof(PIXELFORMATDESCRIPTOR), &pixelDesc)==0)
                        return FALSE;
        }
        if (SetPixelFormat( hDC, m_GLPixelIndex, &pixelDesc)==FALSE)
                return FALSE;
        return TRUE;
}
```

7) Nowe zmienne klasy view

```
int m_GLPixelIndex; // protected
```

```
HGLRC m_hGLContext; // protected
```

8) WM_CREATE (tego typu opis oznacza utworzenie metody przechwytującej zdarzenie/komunikat WM_CREATE przez klasę CEx1View: VC6.0 – class wizard, VC2005 i nowsze – domyślnie ukryte okienko po prawej stronie edytowanego tekstu).

```
int CEx1View::OnCreate(LPCREATESTRUCT lpCreateStruct){
      if (CView::OnCreate(lpCreateStruct) == -1)
            return -1;

HWND hWnd = GetSafeHwnd();
HDC hDC = ::GetDC(hWnd);

      if (SetWindowPixelFormat(hDC)==FALSE)
            return 0;

      return 0;
}
```

9) Nowa metoda

```
BOOL CEx1View::CreateViewGLContext(HDC hDC){
      m_hGLContext = wglCreateContext(hDC);
      if (m_hGLContext == NULL)
            return FALSE;
      if (wglMakeCurrent(hDC, m_hGLContext)==FALSE)
            return FALSE;
      return TRUE;
}
```

10) WM_CREATE

```
int CEx1View::OnCreate(LPCREATESTRUCT lpCreateStruct){
      if (CView::OnCreate(lpCreateStruct) == -1)
            return -1;

HWND hWnd = GetSafeHwnd();
HDC hDC = ::GetDC(hWnd);

      if (SetWindowPixelFormat(hDC)==FALSE)
            return 0;
      if (CreateViewGLContext(hDC)==FALSE)
            return 0;

      return 0;
}
```

11) WM_DESTROY

```
void CEx1View::OnDestroy(){
      if (wglGetCurrentContext()!=NULL)
// make the rendering context not current
           wglMakeCurrent(NULL, NULL) ;

      if (m_hGLContext!=NULL){
            wglDeleteContext(m_hGLContext);
            m_hGLContext = NULL;
      }
// Now the associated DC can be released.
      CView::OnDestroy();
}
```

12) Modyfikacja konstruktora klasy View

```
CEx1View::CEx1View(){
      m_hGLContext = NULL;
```

```
      m_GLPixelIndex = 0;
}
```

I. Sprawdź i przeanalizuj działanie programu.

13a) WM_SIZE

```
void CEx1View::OnSize(UINT nType, int cx, int cy){
      CView::OnSize(nType, cx, cy);

GLsizei width, height;
GLdouble aspect;

      width = cx;
      height = cy;
      if (cy==0)
            aspect = (GLdouble)width;
      else
            aspect = (GLdouble)width/(GLdouble)height;

      glViewport(0, 0, width, height);//poczatek u.ws. lewy gorny rog
//    glViewport(width/4, height/4, width/2, height/2);

      glMatrixMode(GL_PROJECTION);//Nast•pne 2 wiersze b•d• modyfikowały m. PROJECTION
      glLoadIdentity();//inicjalizacja
      gluOrtho2D(0.0, 500.0*aspect, 0.0, 500.0);

      glMatrixMode(GL_MODELVIEW); //Nast•pny wiersz b•dzie modyfikował m. MODELVIEW
      glLoadIdentity();
}
```

14a) WM_PAINT

```
void CEx1View::OnPaint(){
      CPaintDC dc(this); // device context for painting (added by
                         // ClassWizard)
// Wyswietlana scena - poczatek
      glLoadIdentity();
      glClear(GL_COLOR_BUFFER_BIT);
      glBegin(GL_POLYGON);//zmiana stanu OpenGL'a

//Kodowanie składowych: red, green, blue, alpha; 0...1;0-ciemne odcienie
      glColor4f(1.0f, 0.0f, 0.0f, 1.0f);//Red;
      glVertex2f(100.0f, 50.0f);

      glColor4f(0.0f, 1.0f, 0.0f, 1.0f);//Green
      glVertex2f(450.0f, 400.0f);

      glColor4f(0.0f, 0.0f, 1.0f, 1.0f);//Blue
      glVertex2f(450.0f, 50.0f);

      glEnd();
// Wyswietlana scena - koniec

      glFlush();//start processing buffered OpenGL routines
}
```

II. Sprawdź i przeanalizuj działanie programu.
Uwaga – podobny szkielet programu będzie wykorzystywany w następnych ćwiczeniach.

13b) WMPAINT

```
void CEx1View::OnPaint(){
CPaintDC dc(this); // device context for painting
```

```
      CEx1Doc* pDoc = GetDocument();
      pDoc->RenderScene();
}
```

14b) Zmiany w klasie dokumentu (zmienne)

```
enum GLDisplayListNames{
      ArmPart=1  // This is a identifier for the display list that we will be creating to draw the parts of the arm.
};
double m_transY;// This is the y offset of the arm from the world coordinate system origin
double m_transX;// This is the x offset of the arm from the world coordinate system origin
double m_angle2;// This is the angle of the second part of the arm with respect to the first part.
double m_angle1;// This is the angle of the first part of the arm with respect to the world coordinate axis.
```

14c) Zmiany w klasie dokumentu

```
CEx1Doc::CEx1Doc(){
      m_transY=100;m_transX=100;
      m_angle2=15; m_angle1=15;
}

void CEx1Doc::RenderScene(void){
      glClear(GL_COLOR_BUFFER_BIT); //Wstepna wersja
//Przykład 1
      glColor4f(1.0f, 0.0f, 0.0f, 1.0f);
      glCallList(ArmPart);
//Przykład 2
//    glPushMatrix();
//      glTranslated( m_transX, m_transY, 0);
//      glRotated( m_angle1, 0, 0, 1);
//      glColor4f(1.0f, 0.0f, 0.0f, 1.0f);
//      glCallList(ArmPart);
//    glPopMatrix();
//Przykład 3
//    glPushMatrix();
//      glTranslated( m_transX, m_transY, 0);
//      glRotated( m_angle1, 0, 0, 1);
//      glPushMatrix();
//        glTranslated( 90, 0, 0);
//        glRotated( m_angle2, 0, 0, 1);
//        glColor4f(0.0f, 1.0f, 0.0f, 1.0f);
//        glCallList(ArmPart);
//      glPopMatrix();
//      glColor4f(1.0f, 0.0f, 0.0f, 1.0f);
//      glCallList(ArmPart);
//    glPopMatrix();

      glFlush();
}

BOOL CEx1Doc::OnNewDocument(){
      if (!CDocument::OnNewDocument())
            return FALSE;

      glNewList(ArmPart);              //Original OpenGL
//    glNewList(ArmPart, GL_COMPILE); // Microsoft VC++ API

      glBegin(GL_POLYGON);
      glVertex2f(-10.0f, 10.0f);
      glVertex2f(-10.0f, -10.0f);
      glVertex2f(100.0f, -10.0f);
      glVertex2f(100.0f, 10.0f);
```

```
      glEnd();
      glEndList();
      return TRUE;
}
```

III. Sprawdź i przeanalizuj działanie programu dla Przykładu 1, 2 i 3.

15c) Klasa view Dodanie możliwości poruszania myszką

```
CPoint m_RightDownPos; // Initialize to (0,0)
CPoint m_LeftDownPos;  // Initialize to (0,0)
BOOL m_RightButtonDown; // Initialize to FALSE
BOOL m_LeftButtonDown;  // Initialize to FALSE
```

16c) Nowe metody dla WM_LBUTTONDOWN, WM_LBUTTONUP, WM_RBUTTONDOWN, WM_RBUTTONUP.

```
void CEx1View::OnLButtonUp(UINT nFlags, CPoint point){
      m_LeftButtonDown = FALSE;
      CView::OnLButtonUp(nFlags, point);
}
void CEx1View::OnLButtonDown(UINT nFlags, CPoint point){
      m_LeftButtonDown = TRUE; m_LeftDownPos = point;
      CView::OnLButtonDown(nFlags, point);
}
void CEx1View::OnRButtonUp(UINT nFlags, CPoint point){
      m_RightButtonDown = FALSE;
      CView::OnRButtonUp(nFlags, point);
}
void CEx1View::OnRButtonDown(UINT nFlags, CPoint point){
      m_RightButtonDown = TRUE; m_RightDownPos = point;
      CView::OnRButtonDown(nFlags, point);
}
```

17c) WM_MOUSEMOVE

```
void CEx1View::OnMouseMove(UINT nFlags, CPoint point){
      if (m_RightButtonDown){
      CEx1Doc* pDoc = GetDocument();
      CSize rotate = m_RightDownPos - point;
            m_RightDownPos = point;
            pDoc->m_angle1 += rotate.cx/3;
            pDoc->m_angle2 += rotate.cy/3;
            InvalidateRect(NULL,FALSE);
      }
      if (m_LeftButtonDown){
      CEx1Doc* pDoc = GetDocument();
      CSize translate = m_LeftDownPos - point;
            m_LeftDownPos = point;
            pDoc->m_transX -= translate.cx/3;//Warto•• mo•e ulec zmianie
            pDoc->m_transY += translate.cy/3;//Warto•• mo•e ulec zmianie
            InvalidateRect(NULL,FALSE);
      }
      CView::OnMouseMove(nFlags, point);
}
```

18c) Modyfikacja CEx1View::SetWindowPixelFormat

```
pixelDesc.dwFlags = PFD_DRAW_TO_WINDOW |
      PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER |
      PFD_STEREO_DONTCARE;
```

19c) Dodanie na koncu CEx1View::OnSize

```
      glDrawBuffer(GL_BACK);
```

20c) Dodanie na koncu CEx1View::OnPaint:

```
      SwapBuffers(dc.m_ps.hdc);
```

IV, V. Sprawdź i przeanalizuj działanie programu dla Example2 i Example3.

13d) Modyfikacja `CEx1View::OnSize`

```
void CEx1View::OnSize(UINT nType, int cx, int cy){
CView::OnSize(nType, cx, cy);
GLsizei width, height;
GLdouble aspect;

      width = cx;
      height = cy;
      if (cy==0)
            aspect = (GLdouble)width;
      else
            aspect = (GLdouble)width/(GLdouble)height;
      glViewport(0, 0, width, height);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluPerspective(45, aspect, 1, 10.0);
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      glDrawBuffer(GL_BACK);
//    glEnable(GL_LIGHTING);
////  glEnable(GL_DEPTH_TEST);
}
```

14d) Modyfikacja `CEx1Doc::RenderScene`

```
//GLfloat RedSurface[] = { 1.0f, 0.0f, 0.0f, 1.0f};
//GLfloat GreenSurface[] = { 0.0f, 1.0f, 0.0f, 1.0f};
//GLfloat BlueSurface[] = { 0.0f, 0.0f, 1.0f, 1.0f};
////GLfloat LightAmbient[] = { 0.1f, 0.1f, 0.1f, 0.1f };
////GLfloat LightDiffuse[] = { 0.7f, 0.7f, 0.7f, 0.7f };
////GLfloat LightSpecular[] = { 0.0f, 0.0f, 0.0f, 0.1f };
////GLfloat LightPosition[] = { 5.0f, 5.0f, 5.0f, 0.0f };

void CEx1Doc::RenderScene(void){
      glClear(GL_COLOR_BUFFER_BIT);
//////glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
//////glLightfv(GL_LIGHT0, GL_AMBIENT, LightAmbient);
//////glLightfv(GL_LIGHT0, GL_DIFFUSE, LightDiffuse);
//////glLightfv(GL_LIGHT0, GL_SPECULAR, LightSpecular);
//////glLightfv(GL_LIGHT0, GL_POSITION, LightPosition);
//////glEnable(GL_LIGHT0);

      glPushMatrix();
        glTranslated(0.0, 0.0, -8.0);
        glRotated(m_angle1, 1.0, 0.0, 0.0);
        glRotated(m_angle2, 0.0, 1.0, 0.0);

//      glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, RedSurface);
        glBegin(GL_POLYGON);
          glNormal3d( 1.0, 0.0, 0.0);
          glVertex3d( 1.0, 1.0, 1.0);
          glVertex3d( 1.0, -1.0, 1.0);
          glVertex3d( 1.0, -1.0, -1.0);
          glVertex3d( 1.0, 1.0, -1.0);
        glEnd();
        glBegin(GL_POLYGON);
          glNormal3d( -1.0, 0.0, 0.0);
          glVertex3d( -1.0, -1.0, 1.0);
          glVertex3d( -1.0, 1.0, 1.0);
```

```
          glVertex3d( -1.0, 1.0, -1.0);
          glVertex3d( -1.0, -1.0, -1.0);
        glEnd();

//      glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, GreenSurface);
        glBegin(GL_POLYGON);
          glNormal3d( 0.0, 1.0, 0.0);
          glVertex3d( 1.0, 1.0, 1.0);
          glVertex3d( -1.0, 1.0, 1.0);
          glVertex3d( -1.0, 1.0, -1.0);
          glVertex3d( 1.0, 1.0, -1.0);
        glEnd();
        glBegin(GL_POLYGON);
          glNormal3d( 0.0, -1.0, 0.0);
          glVertex3d( -1.0, -1.0, 1.0);
          glVertex3d( 1.0, -1.0, 1.0);
          glVertex3d( 1.0, -1.0, -1.0);
          glVertex3d( -1.0, -1.0, -1.0);
        glEnd();

//      glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, BlueSurface);
        glBegin(GL_POLYGON);
          glNormal3d( 0.0, 0.0, 1.0);
          glVertex3d( 1.0, 1.0, 1.0);
          glVertex3d( -1.0, 1.0, 1.0);
          glVertex3d( -1.0, -1.0, 1.0);
          glVertex3d( 1.0, -1.0, 1.0);
        glEnd();
        glBegin(GL_POLYGON);
          glNormal3d( 0.0, 0.0, -1.0);
          glVertex3d( -1.0, 1.0, -1.0);
          glVertex3d( 1.0, 1.0, -1.0);
          glVertex3d( 1.0, -1.0, -1.0);
          glVertex3d( -1.0, -1.0, -1.0);
        glEnd();
      glPopMatrix();
}
```

VI, VII, VIII, IX. Sprawdź i przeanalizuj działanie programu; wykasuj pojedyńcze komentarze „//” i powtórz ten punkt (3 testy).
Uwaga – podobny szkielet programu będzie wykorzystywany w następnych ćwiczeniach.

Wykorzystanie biblioteki GLUT 3.7 [6], [10].

a) Zainstaluj biblioteke
   - Prekompilowany kod znajduje się pod adresem http://www.opengl.org/resources/libraries/glut/
   - Dynamicznie linkowane pliki (DLL) należy skopiować do katalogu, w którym będzie uruchamiana aplikacja (np. Debug).
   - Przykład konfiguracji:
     X:\glut-3.7\include w zmiennej INCLUDE.
     X:\glut-3.7\lib\glut\glut32.lib oraz **opengl32.lib** w zmiennej LIB.
     X:\glut-3.7\lib\glut\glut32.dll w zmiennej PATH.
     dla
     \\neo\common\tgk jako dysk sieciowy **X: .**

b) **gcc plik.c -lGL -lGLU -lglut**

```
//#include "stdafx.h"
//#include <windows.h>
#include <GL/gl.h>
#include <GL/glut.h>
#include <stdlib.h>

void MyDisplay(void){
// Wyswietlana scena - poczatek
	glLoadIdentity();
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POLYGON);//zmiana stanu OpenGL'a
	glColor4f(1.0f, 0.0f, 0.0f, 1.0f);//Red;
	glVertex2f(100.0f, 50.0f);
	glColor4f(0.0f, 1.0f, 0.0f, 1.0f);//Green
	glVertex2f(450.0f, 400.0f);
	glColor4f(0.0f, 0.0f, 1.0f, 1.0f);//Blue
	glVertex2f(450.0f, 50.0f);
	glEnd();
// Wyswietlana scena - koniec
	glFlush();//start processing buffered OpenGL routines
}

void MyInit(void){
	glClearColor (0.0, 0.0, 0.0, 0.0);//select clearing (background) color
/* initialize viewing values */
	glViewport(0, 0, 300, 300);//pocz•tek u.ws. lewy górny róg
	glMatrixMode(GL_PROJECTION);//Nast•pne 2 wiersze b•d• modyfikowały m. PROJECTION
	glLoadIdentity();//inicjalizacja
	gluOrtho2D(0.0, 500.0*1.2, 0.0, 500.0);
	glMatrixMode(GL_MODELVIEW); //Nast•pny wiersz b•dzie modyfikował m. MODELVIEW
	glLoadIdentity();
}

/*int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR     lpCmdLine,
                     int       nCmdShow){
	glutInit(&__argc, __argv);              */
int main(int argc, char** argv){
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);//single buffer and RGBA
	glutInitWindowSize(250, 250);//initial window size
	glutInitWindowPosition(100, 100);
	glutCreateWindow("My window");//create widnow, hello title bar
	MyInit();
```

```
        glutDisplayFunc(MyDisplay);//register display function (call-back)

/*      glutSetMenu(menu);
        glutReshapeFunc(NULL);
        glutReshapeFunc(NULL);
        glutKeyboardFunc(NULL);
        glutKeyboardFunc(NULL);
        glutMouseFunc(NULL);
        glutMouseFunc(NULL);
        glutMotionFunc(NULL);
        glutMotionFunc(NULL);
        glutVisibilityFunc(NULL);
        glutVisibilityFunc(NULL);
        glutMenuStateFunc(NULL);
        glutMenuStateFunc(NULL);
        glutMenuStatusFunc(NULL);
        glutMenuStatusFunc(NULL);
        glutSpecialFunc(NULL);
        glutSpecialFunc(NULL);
        glutSpaceballMotionFunc(NULL);
        glutSpaceballMotionFunc(NULL);
        glutSpaceballRotateFunc(NULL);
        glutSpaceballRotateFunc(NULL);
        glutSpaceballButtonFunc(NULL);
        glutSpaceballButtonFunc(NULL);
        glutButtonBoxFunc(NULL);
        glutButtonBoxFunc(NULL);
        glutDialsFunc(NULL);
        glutDialsFunc(NULL);
        glutTabletMotionFunc(NULL);
        glutTabletMotionFunc(NULL);
        glutTabletButtonFunc(NULL);
        glutTabletButtonFunc(NULL);
        glutTimerFunc(100, NULL, 1); … */

        glutMainLoop();//enter main loop and process events
        return 0;
}
```

X. Sprawdź i przeanalizuj działanie programu.
Uwaga – podobny szkielet programu będzie wykorzystywany w następnych ćwiczeniach.

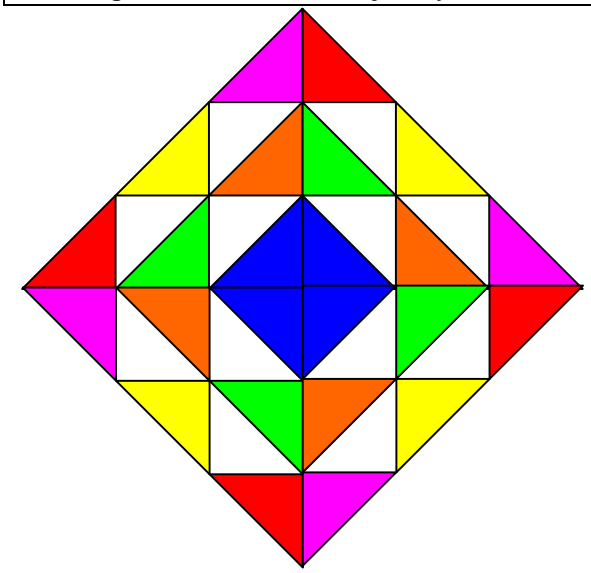
Literatura uzupełniająca do następnych ćwiczeń [2], [3].

Literatura:

[1] Alan Oursland *"Using OpenGL in Visual C++"*
[2] www.sgi.com/software/opengl/
[3] msdn.microsoft.com/library/default.asp?URL=/library/psdk/opengl/int01_2v58.htm
[4] Jeff Molofee et al, *"NeHe OpenGL Tutorial"* , Last rev. Jan 2000
[5] Mark Kilgard *"OpenGL Programming for the X Window System"* Addison-Wesley Developers Press, 1996
[6] *"OpenGL Programming Guide."* 2nd ed., Addison-Wesley Publishing Company
[7] ftp://sgigate.sgi.com/pub/opengl/doc/
[8] http://www.austin.ibm.com/software/OpenGL/
[9] Norman Lin "Linux 3D Graphics Programming". Wordware Publishing 2001
[10] http://www.pobox.com/~ndr/glut.html , http://reality.sgi.com/mjk/glut3/glut3.html , http://www.opengl.org , http://www.sgi.com/Technology/openGL

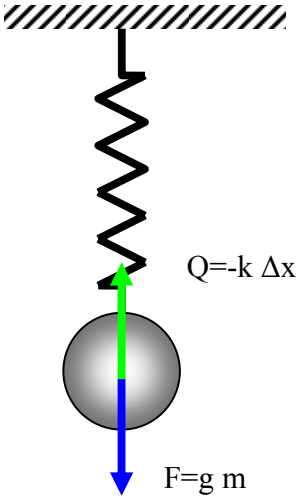
Ćwiczenie nr 2 – Budowa podstawowych obiektów oraz opis ich ruchu

Zadanie. Narysować następujący obiekt (zbudowany z 24 trójkątów) oraz wprowadzić go w ruch obrotowy względem początkowego środka ciężkości. Wewnętrzne trójkąty obracają się w przeciwnym kierunku (3 obroty/1obrót całości) oraz oddalają się od początkowego środka ciężkości z prędkością 3 długości przyprostokątnej na 10 obrotów całej figury.
Można pominąć analizę wzajemnych zderzeń trójkątów.



Ćwiczenie nr 3 – Modelowanie rzeczywistych obiektów i zjawisk

Zadanie. Zamodelować ruch sprężyny oraz kuli połączonych według zamieszczonego poniżej rysunku z uwzględnieniem praw fizyki i ich rzeczywistego wyglądu.



Dodatkowe założenia:

- Kula wykonana z drewna/szkła
- Sprężyna wykonana z drutu stalowego
- W początkowej chwili sprężyna jest naciągnięta
- Zakładamy zerową prędkość początkowa kuli.
- Model w pełni 3-D, zastosować tekstury do modelowania powierzchni.
- Powierzchnia sprężyny zamodelowana na podstawie wzoru:

$$\begin{cases} x_i = \cos(t_i)\cdot(3+\cos(u_i)) \\ y_i = \sin(t_i)\cdot(3+\cos(u_i)) \\ z_i = 0.6\cdot t_i + \sin(u_i) \end{cases}$$

$$t = 0,\ldots,8\pi; u = 0,\ldots,2\pi$$

- Uwzględnić górne i dolne wykończenie sprężyny (sfera-cylinder-sfera-cylinder).

Podpowiedź – dla uproszczenia zastosować równanie ruchu podane w postaci analitycznej.

Projekt

Wykonać model jednego z wymienionych niżej obiektów uwzględniający: kształt obiektu, tekstury, cienie, ruch obiektu, możliwość zmiany szybkości ruchu oraz położenia kamery.

1. Chodzący robot.
2. Rękę wykonującą gesty.
3. Twarz ludzką z mimiką.
4. Poruszające się zwierzę (np. pies, kot).
5. Drzewo poruszające gałęziami.
6. Grupę krzewów poruszających gałęziami.
7. Pięć wahadeł umieszczonych wzdłuż linii i zderzających się ze sobą.
8. Ryba płynąca w wodzie i wyskakująca z niej, co pewien czas.
9. Latający motyl.
10. Łódkę pływającą na falach.
11. Jadący samochód.
12. Lecący samolot.
13. Układ słoneczny.
14. Inne o podobnym poziomie trudności do uzgodnienia z prowadzącym.

Przykład: