# PyOpenGL – wprowadzenie



maciej.hojda@pwr.edu.pl

## 1 Zadanie nr 1

1. Utwórz projekt w języku Python z zaimportowanymi bibliotekami PyOpenGL i numpy (załącznik A).
2. Uruchom program rysujący pustą scenę (załącznik B).
3. Zmodyfikuj program tak, żeby okno miało następujące parametry: rozmiar 800x800 [px x px], kolor tła czerwony, tytuł „Lista nr 8".
4. Zapoznaj się z dokumentacją biblioteki PyOpenGL (`http://pyopengl.sourceforge.net/documentation/manual-3.0/index.html`).

## 2 Zadanie nr 2

1. Uruchom program rysujący kilka figur (załącznik C).
2. Dodaj do sceny sześciokąt foremny z wierzchołkami w różnych kolorach.
3. Uzupełnij scenę o animację przybliżającą i oddalającą narysowane figury.

## 3 Zadanie nr 3

1. Utwórz program animujący sześcian i czworościan foremny. Bryły mają być od siebie oddalone (bez przecinania). Animacja przedstawia obrót obu brył dookoła jednej osi.
2. Dodaj obsługę klawiatury (`glutKeyboardFunc`) umożliwiającą: zatrzymanie i wznowienie obrotu, obrót tylko sześcianu, obrót tylko czworościanu, zmianę kolorów i rozmiarów brył.
3. * Dodaj menu (pod prawym przyciskiem myszy) umożliwiające ukrywanie i odkrywanie każdej z brył (`glutCreateMenu`, `glutAddMenuEntry`, `glutAttachMenu` ...).

## 4 Zadanie nr 4

1. Utwórz scenę z dwudziestościanem foremnym.
2. Dodaj dowolną teksturę do wybranych ścian dwudziestościanu (zalacznik D).
3. Dodaj kamerę (`glMultMatrixf`). Kamera znajduje się na sferze w całości zawierającej w sobie utworzony dwudziestościan. Kamera jest skierowana do środka sfery. Współrzędne sferyczne (długość i szerokość) są zmieniane za pomocą myszki (tylko przy przeciągnięciu przy naciśniętym prawym przycisku myszki).

# 5 Zadanie nr 5

1. Uruchom program wykorzystujący Shader-y (załącznik E).
2. Zapoznaj się z dokumentacją
   `https://www.khronos.org/registry/OpenGL-Refpages/gl4/`
3. Napisz program rysujący kardioidę z wykorzystaniem Shader-ów.
4. Dodaj możliwość zmiany (z klawiatury) grubości linii rysującej kardioidę.

# Załączniki

# A Instalacja

1. (pomiń punkt, jeśli został już zrealizowany) Zainstaluj interpreter języka Python
   - pobierz `https://www.python.org/ftp/python/3.6.4/python-3.6.4.exe`
2. (pomiń punkt, jeśli został już zrealizowany) Zainstaluj bibliotekę PyOpenGL
   - pobierz z `https://www.lfd.uci.edu/~gohlke/pythonlibs/#pyopengl`
   (`PyOpenGL-3.1.1-cp36-cp36m-win32.whl`)
   - zainstaluj `[katalog Pythona]\Scripts\pip install [pełna ścieżka do biblioteki]`
3. (pomiń punkt, jeśli został już zrealizowany) Zainstaluj bibliotekę numpy
   - pobierz z `https://pypi.python.org/pypi/numpy`
   (`numpy-1.14.0rc1-cp36-none-win32.whl`)
   - zainstaluj `[katalog Pythona]\Scripts\pip install [pełna ścieżka do biblioteki]`

# B Pusta scena

```
from OpenGL.GL import *                         # importowanie GL
from OpenGL.GLUT import *                       # importowanie GLUT

def rysuj():
    glClear(GL_COLOR_BUFFER_BIT);               # czyszczenie bufora kolorów
    glFlush();                                  # wymuszenie wyświetlania

glutInit();                                     # inicjalizacja biblioteki
glutInitWindowSize(600, 400);                   # ustawienie rozmiaru okna
glutInitWindowPosition(0, 0);                   # ustawienie pozycji okna
glutCreateWindow(b"Zadanie nr 1");              # tytuł okna
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); # parametry okna
glutDisplayFunc(rysuj);                         # wybór głównej pętli programu
glClearColor(1.0, 1.0, 1.0, 1.0);               # ustawienie koloru tła
glutMainLoop();                                 # uruchomienie głównej pętli programu
```

# C Figury

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
```

```
# obsługa myszki
myszkax = 400;
myszkay = 300;
def myszka(x, y):
    global myszkax, myszkay;
    myszkax = x;
    myszkay = y
    glutPostRedisplay();    # zaznacz, że okno wymaga przerysowania

def rysuj():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); # czyszczenie sceny
    glLoadIdentity(); # resetowanie widoku

    glTranslatef(-4.0, -2.0, -12.0)  # przesunięcie widoku
    glRotatef(-400 + myszkax, 0.0, 1.0, 0.0); # obrót
    glRotatef(-300 + myszkay, 1.0, 0.0, 0.0); # obrót

    glColor3f(1.0, 0.5, 0.5); # kolor
    glBegin(GL_POLYGON) # rysowanie trójkąta
    glVertex3f(0.0, 1.0, 0.0)
    glVertex3f(1.205, -1.0, 0.0)
    glVertex3f(-1.205, -1.0, 0.0)
    glEnd() # koniec rysowania trójkąta

    glColor3f(0.0, 0.5, 0.5);
    glTranslatef(2.0, 2.0, 2.0) # przesunięcie widoku
    glBegin(GL_POLYGON) # rysowanie trójkąta
    glVertex3f(0.0, 1.0, 0.0)
    glVertex3f(1.205, -1.0, 0.0)
    glVertex3f(-1.205, -1.0, 0.0)
    glEnd() # koniec rysowania trójkąta

    glColor3f(1.0, 1.5, 0.5);
    glTranslatef(2.0, 2.0, 2.0) # przesunięcie widoku
    glBegin(GL_POLYGON) # rysowanie trójkąta
    glVertex3f(0.0, 1.0, 0.0)
    glVertex3f(1.205, -1.0, 0.0)
    glVertex3f(-1.205, -1.0, 0.0)
    glEnd() # koniec rysowania trójkąta

    glColor3f(1.0, 0.5, 1.0);
    glTranslatef(3.0, -2.0, 0.0) # przesunięcie widoku
    glBegin(GL_QUADS) # rysowanie prostokąta
    glVertex3f(-1.0, 1.0, 0.0);
    glVertex3f(1.0, 1.0, 0.0);
    glVertex3f(1.0, -1.0, 0.0);
    glVertex3f(-1.0, -1.0, 0.0);
    glEnd() # koniec rysowania prostokąta

    glutSwapBuffers(); # zamiana buforów - wyświetlenie trójkątów i prostokąta
```

```
def program02():
    glutInit(sys.argv); # przekazanie argumentów z wiersza poleceń do GLUT
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
        # kolory rgb, podwójne buforowanie, bufor głębokości

    glutInitWindowSize(800, 600);
    glutInitWindowPosition(300, 300);
    glutCreateWindow(b"Zadanie nr 2");

    glutDisplayFunc(rysuj);
    glutIdleFunc(rysuj);
    glutMotionFunc(myszka);

    glClearColor(0.0, 1.0, 1.0, 1.0);
    glClearDepth(1.0);
    glDepthFunc(GL_LESS); # parametr bufora głębokości
    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION); # tryb projekcji
    glLoadIdentity(); # resetuj projekcję

    gluPerspective(50.0, float(800) / float(600), 0.1, 100.0)
        # rzutowanie perspektywiczne

    glMatrixMode(GL_MODELVIEW); # tryb widoku
    glutMainLoop();

program02();
```

# D Tekstury

```
def wczytajTeksture():
    rys = Image.open("rys.bmp");
    rys = rys.tobytes("raw", "RGBX", 0, -1);

    glTexImage2D(GL_TEXTURE_2D, 0, 3, rys.size[0], rys.size[1],
     0, GL_RGBA, GL_UNSIGNED_BYTE, rys);
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

...
glEnable(GL_TEXTURE_2D);

...
glTexCoord2f(0.0, 0.0);
```

# E Shader-y

```
from OpenGL.GL import *;
from OpenGL.GLU import *;
from OpenGL.GLUT import *;
```

```
from OpenGL.GL.shaders import *;

shad = None;
ures = None;
winwidht = None;
winheight = None;

def paint():
    global shad, ures, winwidht, winheight;
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0.0, -0.0, -4.0);
    glUseProgram(shad);
    glUniform2f(ures, winwidth//2, winheight//2);

    glBegin(GL_POLYGON);
    glVertex2f(-5.0, -2.0);
    glVertex2f(5.0, -2.0);
    glVertex2f(5.0, 2.0);
    glVertex2f(-5.0, 2.0);
    glEnd();

    #glutSolidSphere(1.0, 32, 32);
    glutSwapBuffers();

def main():
    global shad, ures, winwidth, winheight;
    glutInit();
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);

    winwidth = glutGet(GLUT_SCREEN_WIDTH);
    winheight = glutGet(GLUT_SCREEN_HEIGHT);

    glutInitWindowSize(winwidth//2, winheight//2);
    glutInitWindowPosition(winwidth//4, winheight//4);
    glutCreateWindow(b"my02");
    glClearColor(1.0, 0.0, 0.0, 0.5);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(50.0, float(winwidth/winheight), 0.1, 100.0);
    glutDisplayFunc(paint);

    glMatrixMode(GL_MODELVIEW);

    shad = compileProgram(
        compileShader('''
    uniform vec2 ures;


    float getf(vec2 st) {
```

```
        if (abs( sin(st.x * 10)/2 + 0.5 - st.y) < 0.01) {
            return 1.0;
        }
        return 0.0;
    }

    void main() {
        vec2 st = gl_FragCoord.xy/ures;

        float val = getf(st);

        gl_FragColor = vec4(val, 1.0, 1.0, 1.0);
} ''', GL_FRAGMENT_SHADER), )

    ures = glGetUniformLocation(shad, "ures");

    glutMainLoop();

main();
```