Język Python i jego środowisko programistyczne

Pyton:

a) Zainstaluj jedno ze środowisk programistycznych:
PyCharm (https://www.jetbrains.com/pycharm/download/#section=windows ,
https://www.jetbrains.com/pycharm/promo/anaconda/ ; osobiście proponuje PyCharma - do wygenerowania klucza musicie podać konto studenckie z domeny Politechniki Łódzkiej),
Anaconda (https://docs.anaconda.com/anaconda/install/ , **jest to środowisko, które uwielbiają studenci, studenci również chętnie używają Jupiter**),
Minianaconda (https://docs.conda.io/en/latest/miniconda.html)
Spyder (https://www.spyder-ide.org/) lub inne.

Uwaga - nowe wersje bibliotek numerycznych są pisane dla Python3.

Proponuję następujące biblioteki: Numpy, Scipy, Matplotlib, PyQt4.
Uwaga – w języku Pyton wektory i macierze są zwykle przekazywane przez referencje!!!

Literatura dodatkowa:

1. M. Lutz. O' Reilly. Python leksykon kieszonkowy. Helion
2. Jones. Python Cookbook 2013
3. M. Lutz. Learning Python 2013 (zobatrz Helion)"
4. https://pythonprogramming.net/opengl-rotating-cube-example-pyopengl-tutorial/
5. http://pyopengl.sourceforge.net/context/tutorials/index.html NeHe Translation
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

Przykładowa konfiguracja środowiska Spyder:

a) Uruchamiamy Anaconda Nvigator

CMD.exe prompt <- klawisz launch
W nowym okienku powłoki: pip install pygame
pip install pyopengl
lub pip install pyopengl PyOpenGL_accelerate

b) Druga metoda: conda install pyopengl PyOpenGL_accelerate

Zad 1 – wprowadzenie do programowania

```
# -*- coding: utf-8 -*-
#Python
import sys

import pygame
from pygame.locals import *

import OpenGL
from OpenGL.GL import *
from OpenGL.GLU import *

def myPaint():
   glBegin(GL_POLYGON)
   glColor4f(1.0,0.0,0.0,1.0)
   glVertex2f(100.0,50.0)
   glColor4f(0.0,1.0,0.0,1.0)
   glVertex2f(450.0,450.0)
   glColor4f(0.0,0.0,1.0,1.0)
   glVertex2f(450.0,50.0)
   glEnd()

#main code
pygame.init()
display = (800,600)
pygame.display.set_mode(display, DOUBLEBUF|OPENGL)

#gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)
gluOrtho2D(0.0, 500.0*(display[0]/display[1]), 0.0, 500.0);

#glTranslatef(0.0,0.0, -5)

while True:
   for event in pygame.event.get():
      if event.type == pygame.QUIT:
         pygame.quit()
         sys.exit(0)

   #glRotatef(1, 3, 1, 1)
   glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
   myPaint()
   pygame.display.flip()
   pygame.time.wait(10)
```

I. Sprawdź i przeanalizuj działanie programu.

```
# -*- coding: utf-8 -*-
#Python
import sys

import pygame
from pygame.locals import *

import OpenGL
from OpenGL.GL import *
from OpenGL.GLU import *

m_transX=0
m_transY=0
m_angle1=0
m_angle2=0
ArmPart=0

def myPaint():
#Example 1
   glColor4f(1.0, 0.0, 0.0, 1.0)
   glCallList(ArmPart)
#Example 2
#    glPushMatrix()
#    glTranslated( m_transX, m_transY, 0)
#    glRotated( m_angle1, 0, 0, 1)
#    glColor4f(1.0, 0.0, 0.0, 1.0)
#    glCallList(ArmPart)
#    glPopMatrix()
#Example 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(0.0, 1.0, 0.0, 1.0)
#    glCallList(ArmPart)
#    glPopMatrix()
#    glColor4f(1.0, 0.0, 0.0, 1.0)
#    glCallList(ArmPart)
#    glPopMatrix()
```

```
#main code
pygame.init()
display = (800,600)
pygame.display.set_mode(display, DOUBLEBUF|OPENGL)

#gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)
gluOrtho2D(0.0, 500.0*(display[0]/display[1]), 0.0, 500.0);

#glTranslatef(0.0,0.0, -5)

ArmPart=glGenLists(1)
glNewList(ArmPart, GL_COMPILE);
glBegin(GL_POLYGON);
glVertex2f(-10.0, 10.0);
glVertex2f(-10.0, -10.0);
glVertex2f(100.0, -10.0);
glVertex2f(100.0, 10.0);
glEnd();
glEndList();


m_RightDownPos=(0,0)
m_LeftDownPos=(0,0)
m_RightButtonDown=False
m_LeftButtonDown=False

while True:
   for event in pygame.event.get():
      if event.type == pygame.QUIT or event.type == pygame.KEYDOWN and event.key ==
pygame.K_q:
         pygame.quit()
         sys.exit(0)
      elif event.type == pygame.MOUSEMOTION:
         if m_LeftButtonDown:
            m_angle1 += m_LeftDownPos[0] - event.pos[0];
            m_angle2 += m_LeftDownPos[1] - event.pos[1];
            m_LeftDownPos = event.pos;
         elif m_RightButtonDown:
            m_transX -= m_RightDownPos[0] - event.pos[0];
            m_transY += m_RightDownPos[1] - event.pos[1];
            m_RightDownPos = event.pos;
      elif event.type == pygame.MOUSEBUTTONDOWN:
         if event.button == 1:
            m_LeftButtonDown = True
            m_LeftDownPos=event.pos
```

```
            elif event.button == 3:
                m_RightButtonDown = True
                m_RightDownPos=event.pos
        elif event.type == pygame.MOUSEBUTTONUP:
            if event.button == 1:
                m_LeftButtonDown = False
            elif event.button == 3:
                m_RightButtonDown = False

    #glRotatef(1, 3, 1, 1)
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    myPaint()
    pygame.display.flip()
    pygame.time.wait(10)
```

II. Sprawdź i przeanalizuj działanie programu dla przypadku Example 1, 2 i 3.
Uwaga – podobny szkielet programu wykorzystaj w następnych ćwiczeniach.

a) Zamiast ramiona (czerwony i zielony prostokąt) wstaw sześcian.
b) Użyj następujących materiałów do opisu ścian w różnych kolorach:
Np. glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, RedSurface);

```
RedSurface = ( 1.0, 0.0, 0.0, 1.0);
GreenSurface = ( 0.0, 1.0, 0.0, 1.0);
BlueSurface = ( 0.0, 0.0, 1.0, 1.0);
LightAmbient = ( 0.1, 0.1, 0.1, 0.1);
LightDiffuse = ( 0.7, 0.7, 0.7, 0.7);
LightSpecular = ( 0.0, 0.0, 0.0, 0.1);
LightPosition = ( 5.0, 5.0, 5.0, 0.0);
```

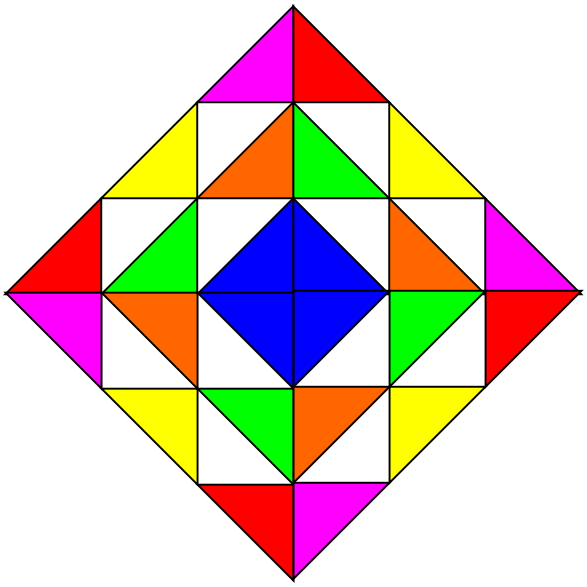
c) Użyj poprawnie następujących komend w w/w:

```
gluPerspective(45, aspect, 1, 10.0);
glDrawBuffer(GL_BACK);
glEnable(GL_LIGHTING);
glEnable(GL_DEPTH_TEST);
```

Jako wynik realizacji tego zadania wstaw końcowy kod źródłowy w języku Python oraz zrzut ekranu przedstawiający uzyskany obiekt 3D.

Zad 2 – Budowa podstawowych obiektów oraz opis ich ruchu

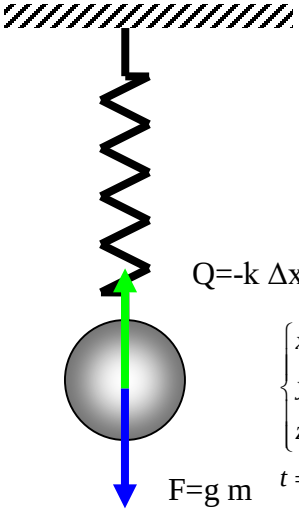
Zadanie. Narysować następujący obiekt (zbudowany z 24 trójkątów) oraz wprowadzić go w ruch obrotowy względem początkowego środka ciężkości. Wewnętrzne trójkąty obracają się w przeciwnym kierunku (3 obroty/1obrót całości) oraz oddalają się od początkowego środka ciężkości z prędkością 3 długości przyprostokątnej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



Jako wynik realizacji tego zadania wstaw końcowy kod źródłowy w języku Python oraz zrzut ekranu przedstawiający uzyskane obiekty.

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.

Jako wynik realizacji tego zadania wstaw końcowy kod źródłowy w języku Python oraz zrzut ekranu przedstawiający uzyskany obiekt 3D.

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.

Jako wynik realizacji tego zadania wstaw końcowy kod źródłowy w języku Python oraz zrzut ekranu przedstawiający uzyskany obiekt 3D.

Przykład: