**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 6**

Тема: Создание шейдерных анимационных эффектов в OpenGL

Студент: Черных Сергей Дмитриевич

Группа: 08-305

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2022

1. Постановка задачи

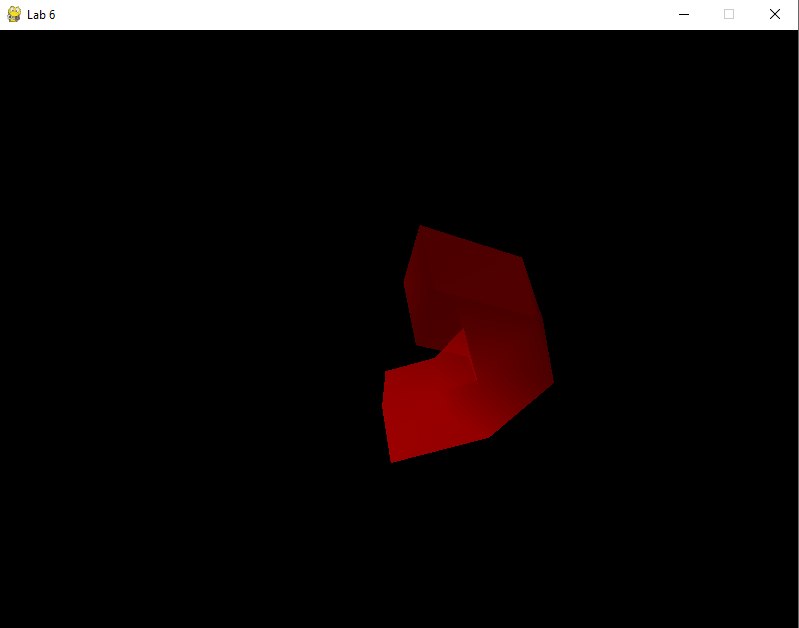
Тема: Создание шейдерных анимационных эффектов в OpenGL 2.1

Задание: для поверхности, созданной в л.р. №5, обеспечить выполнение следующего шейдерного эффекта: Анимация. Прозрачность изменяется по синусоидальному закону.

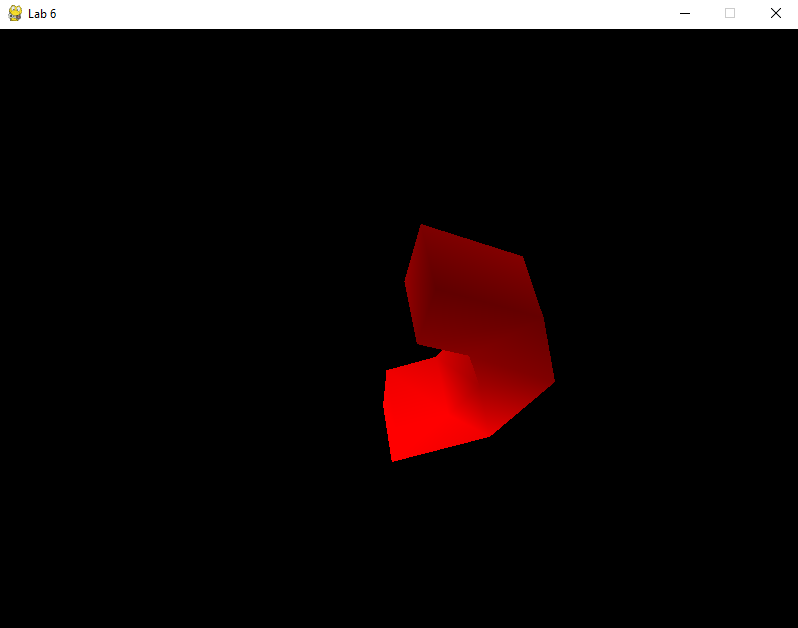
1. Описание программы

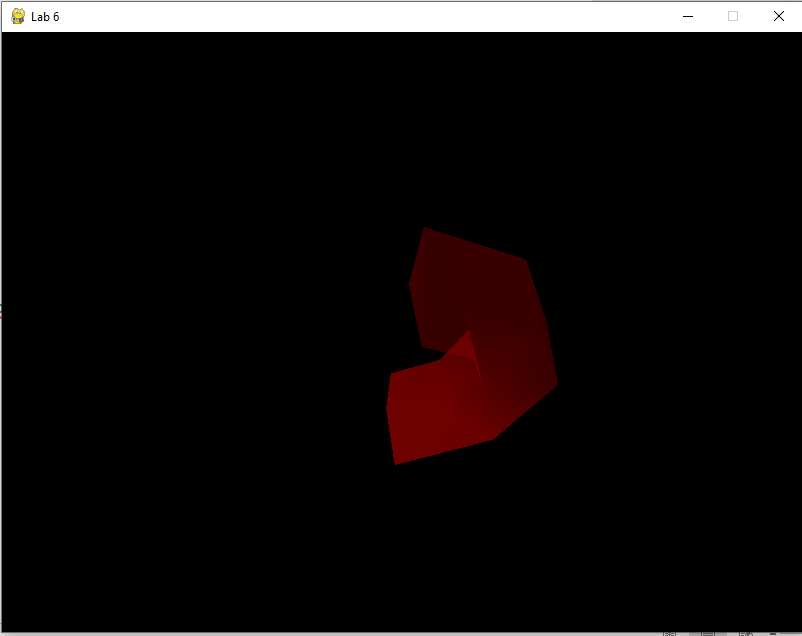
На основе л.р. №5 была разработана программа изменяющая заданный альфа канал, для изменения прозрачности фигуры. Активация эффекта происходит с помощью клавиши D, увеличение скорости изменения так же с помощью клавиши D, уменьшение скорости с помощью клавиши С, отключение эффектов с помощью клавиши A.

1. Набор тестов
   * + 1. x d
       2. x d a d a c
       3. x d d d c c
2. Результаты выполнения тестов
   * + 1. x d



* + - 1. x d a d a c



* + - 1. x d d d c c

1. Листинг программы

# Черных М80-305Б-20

# Создание шейдерных анимационных эффектов в OpenGL 2.1

# Для поверхности, созданной в л.р. №5, обеспечить выполнение следующего шейдерного эффекта:

# Анимация. Прозрачность изменяется по синусоидальному закону

import pygame

import numpy as np

from pygame.locals import \*

from OpenGL.GL import \*

from OpenGL.GLU import \*

from math import pi, sin

def make\_vertices(r\_outter=2, r\_inner=1, nt=5, a=00.5, h=1):

def cylinder(r, a=0, nt=5, h=1):

theta = np.linspace(-np.pi/2, np.pi/2, nt)

v = np.ones(nt) \* h

x = r\*np.cos(theta) - a

y = r\*np.sin(theta)

z = v

return x, y, z

x, y, z = cylinder(r\_outter, nt=nt, h=h)

x\_1, y\_1, z\_1 = cylinder(r\_outter, nt=nt, h=h)

z\_1 = np.zeros\_like(z\_1)

x = np.concatenate((x, x\_1), axis=0)

y = np.concatenate((y, y\_1), axis=0)

z = np.concatenate((z, z\_1), axis=0)

x\_2, y\_2, z\_2 = cylinder(r\_inner, nt=nt, a=a, h=h)

x\_3, y\_3, z\_3 = cylinder(r\_inner, nt=nt, a=a, h=h)

z\_3 = np.zeros\_like(z\_3)

x\_2 = np.concatenate((x\_2, x\_3), axis=0)

y\_2 = np.concatenate((y\_2, y\_3), axis=0)

z\_2 = np.concatenate((z\_2, z\_3), axis=0)

x = np.concatenate((x, x\_2), axis=0)

y = np.concatenate((y, y\_2), axis=0)

z = np.concatenate((z, z\_2), axis=0)

vert = (tuple(x), tuple(y), tuple(z))

vertices = tuple(map(list, zip(\*vert)))

return vertices

def make\_surfaces(nt=5):

i = np.array([])

j = np.array([])

k = np.array([])

for c in range(nt-1):

i = np.append(i, c)

j = np.append(j, c + 1)

k = np.append(k, c + nt)

for c in range(nt-1):

i = np.append(i, c + nt + 1)

j = np.append(j, c + nt)

k = np.append(k, c + 1)

for c in range(nt-1):

j = np.append(j, c + 2\*nt)

i = np.append(i, c + 2\*nt + 1)

k = np.append(k, c + 3\*nt)

for c in range(nt-1):

j = np.append(j, c + 3\*nt + 1)

i = np.append(i, c + 3\*nt)

k = np.append(k, c + 2\*nt + 1)

for c in range(nt-1):

j = np.append(j, 3\*nt + c)

i = np.append(i, c + 3\*nt + 1)

k = np.append(k, c + nt)

for c in range(nt-1):

j = np.append(j, c + nt + 1)

i = np.append(i, c + nt)

k = np.append(k, c + 3\*nt + 1)

for c in range(nt-1):

i = np.append(i, c + 2\*nt)

j = np.append(j, c + 2\*nt + 1)

k = np.append(k, c)

for c in range(nt-1):

i = np.append(i, c + 1)

j = np.append(j, c)

k = np.append(k, c + 2\*nt + 1)

j = np.append(j, 0)

i = np.append(i, 2\*nt)

k = np.append(k, nt)

j = np.append(j, 2\*nt)

i = np.append(i, 3\*nt)

k = np.append(k, nt)

j = np.append(j, nt - 1)

i = np.append(i, 4\*nt - 1)

k = np.append(k, 3\*nt - 1)

j = np.append(j, nt - 1)

i = np.append(i, 2\*nt - 1)

k = np.append(k, 4\*nt - 1)

i = i.astype("int")

k = k.astype("int")

j = j.astype("int")

i = tuple(i)

j = tuple(j)

k = tuple(k)

sur = (i, j, k)

surfaces = list(map(list, zip(\*sur)))

return surfaces

def Cube(r\_outter=2, r\_inner=1, nt=5, a=00.5, h=1, trns=1):

verticies = make\_vertices(r\_outter=r\_outter, r\_inner=r\_inner, nt=nt, a=a, h=h)

surfaces = make\_surfaces(nt=nt)

glBegin(GL\_TRIANGLES)

for surface in surfaces:

for vertex in surface:

COLOR = ((1, 0, 0, trns))

glMaterial(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE, COLOR)

glVertex3fv(verticies[vertex])

glEnd()

def main():

print('''\t(- =) аппроксимация, (h j) высота подковы, (r t) внешний радиус подковы,

(f g) внутренний радиус подковы, (a s) расстояние между полуокружностями подковы,

(x) вращение вокруг Ox, (y) вращение вокруг Oy, (z) вращение вокруг Oz,

стрелки перемещение по плоскости, колесико мыши масштаб,

Активация/деактивация эффекта прозраности (d a)

Увеличение/уменьшение скорости эффекта (d c)''')

pygame.init()

display = (800,600)

pygame.display.set\_caption('Lab 6')

pygame.display.set\_mode(display, DOUBLEBUF|OPENGL)

gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)

glTranslatef(0,0, -10)

glRotatef(25, 2, 1, 0)

trns\_flag = False

trns = 0

run = True

approx = 5

r\_outter = 2

r\_inner = 1

a = 0.05

h = 1

run = True

pi\_part = pi/12

pi\_in = 0

while run:

events = pygame.event.get()

for event in events:

if event.type == pygame.QUIT:

pygame.quit()

run = False

quit()

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

glTranslatef(-0.5,0,0)

if event.key == pygame.K\_RIGHT:

glTranslatef(0.5,0,0)

if event.key == pygame.K\_UP:

glTranslatef(0,1,0)

if event.key == pygame.K\_DOWN:

glTranslatef(0,-1,0)

if event.key == pygame.K\_x:

glRotatef(30, 1, 0, 0)

if event.key == pygame.K\_y:

glRotatef(30, 0, 1, 0)

if event.key == pygame.K\_z:

glRotatef(30, 0, 0, 1)

if event.key == pygame.K\_ESCAPE:

pygame.quit()

if event.key == pygame.K\_EQUALS: # approx+ =

approx += 1

if event.key == pygame.K\_MINUS: # approx- -

if approx > 3:

approx -= 1

if event.key == pygame.K\_h: # height+ h

h += 1

if event.key == pygame.K\_j: # height- j

if h > 1:

h -= 1

if event.key == pygame.K\_r: # r\_outter+ r

r\_outter += 1

if event.key == pygame.K\_t: # r\_outter- t

if r\_outter > r\_inner + 1:

r\_outter -= 1

if event.key == pygame.K\_f: # r\_inner+ r

if r\_outter > r\_inner + 1:

r\_inner += 1

if event.key == pygame.K\_g: # r\_inner- t

if r\_inner > 1:

r\_inner -= 1

if event.key == pygame.K\_a: # distance\_between\_routter\_and\_r\_inner+ a

a += 0.01

if event.key == pygame.K\_s: # distance\_between\_routter\_and\_r\_inner- s

if a > 0.03:

a -= 0.01

if event.key == pygame.K\_d: # turn on transparancy/increace frequency

pi\_part \*= 2

trns = abs(sin(pi\_in))

glEnable(GL\_ALPHA\_TEST)

glEnable(GL\_BLEND)

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA)

glDisable(GL\_CULL\_FACE)

trns\_flag = True

if event.key == pygame.K\_c: # decrease frequency of transparancy

pi\_part /= 2

if event.key == pygame.K\_a: # turn off transparancy

trns\_flag = False

glEnable(GL\_CULL\_FACE)

glCullFace(GL\_FRONT)

glDepthMask(GL\_TRUE)

glDisable(GL\_ALPHA\_TEST)

glDisable(GL\_BLEND)

if event.type == pygame.MOUSEBUTTONDOWN:

if event.button == 4:

glTranslatef(0,0,1.0)

if event.button == 5:

glTranslatef(0,0,-1.0)

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)

glLightfv(GL\_LIGHT0, GL\_POSITION, (0, 0, 2, 1))

glLightfv(GL\_LIGHT0, GL\_AMBIENT, (0, 0, 0, 1))

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, (1, 1, 1, 1))

glEnable(GL\_DEPTH\_TEST)

glEnable(GL\_LIGHTING)

glEnable(GL\_LIGHT0)

if trns\_flag:

pi\_in += pi\_part

trns = (sin(pi\_in) + 1 ) / 2

glEnable(GL\_BLEND);

glDepthMask(GL\_FALSE);

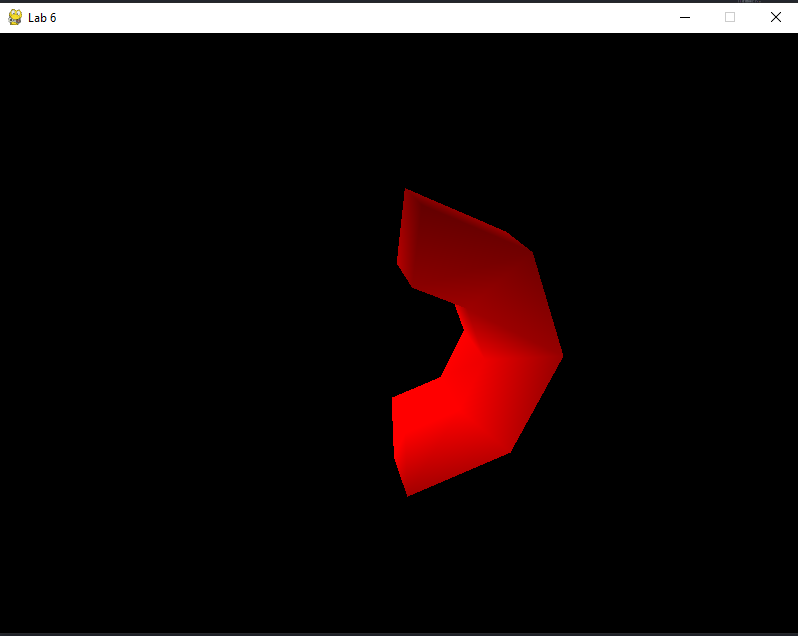
Cube(r\_outter=r\_outter, r\_inner=r\_inner, nt=approx, a=a, h=h, trns=trns)

pygame.display.flip()

pygame.time.wait(10)

if \_\_name\_\_ == '\_\_main\_\_':

main()



ЛИТЕРАТУРА

1. Документация библиотеки PyOpenGL [Электронный ресурс] <https://pypi.org/project/PyOpenGL/>
2. Введение в библиотеку PyOpenGL [Электронный ресурс] <https://stackabuse.com/brief-introduction-to-opengl-in-python-with-pyopengl/>
3. Игорь Тарасов, Программирование с использованием OpenGL [Электронный ресурс] <https://opengl.org.ru/books/open_gl/chapter4.4.html>

(дата обращения: 20.11.2022)