МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра САПР

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Компьютерная графика»

Тема: Формирование различных поверхностей

Студенты гр. 5371 Бергер Э. Э.

Локкина О. С.

Уруков С. Д.

Преподаватель Матвеева И. В.

Санкт-Петербург

2017

**ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ**

*Вариант 3.1:*

Сформировать билинейную поверхность на основе произвольного задания ее 4 угловых точек. Обеспечить ее поворот относительно оси OX.

В дополнение к повороту на заданный угол относительно оси OX, добавлена возможность перемещения точки наблюдения относительно осей X, Y и Z.

**ХОД РАБОТЫ**

Использовалась библиотека OpenGL с языком программирования Python 3.6.x.

В трехмерном пространстве произвольно задаются 4 угловые точки билинейной поверхности. На основе этих точек строится билинейная поверхность с помощью уравнения:

Q(u, w) = (1-u)(1-w)P0,0+u(1-w)P0,1+(1-u)wP1,0+uwP1,1, где

P0,0, P0,1, P1,0, P1,1 – заданные угловые точки,

u, w – меняются от 0 до 1.

Поворот относительно оси OX на угол α осуществляется с помощью матрицы поворота:

**ПРИМЕР ЗАПУСКА ПРОГРАММЫ**

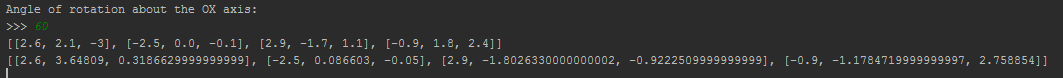
При запуске программы пользователь может ввести угол поворота поверхности относительно оси OX:



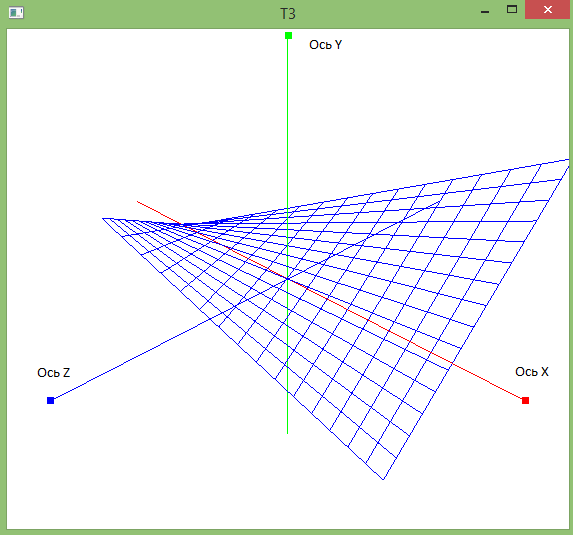
Угол вводится в градусах. Если угол будет равен 0, то откроется только одно окно, в котором будет отображаться построенная билинейная поверхность, если же угол отличен от 0, то появится второе окно, в котором будет отображена повернутая относительно OX поверхность.

После ввода угла, ниже появятся координаты случайно сгенерированных угловых точек поверхности, а также координаты точек после поворота.

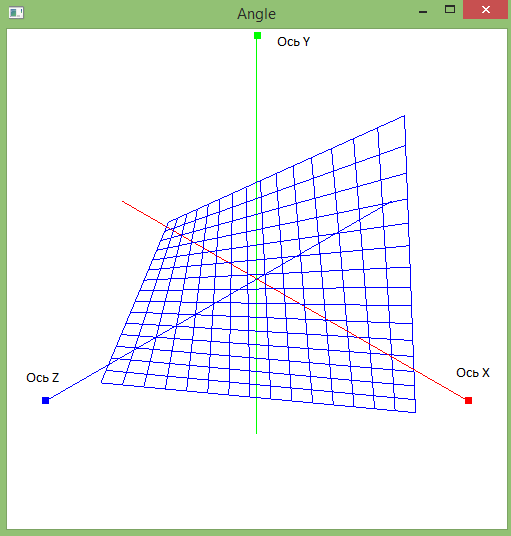
Рассмотрим пример, где угол поворота составляет 60 градусов:



В первом окне отображается исходная билинейная поверхность:



Во втором окне отображается поверхность, повернутая относительно оси OX на 60 градусов:



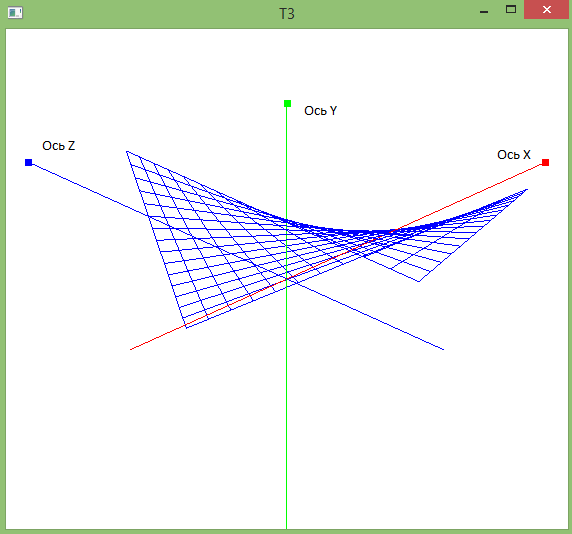
В каждом из двух окон можно перемещать точку наблюдения относительно осей X, Y и Z с помощью следующих клавиш:

q-e – перемещение относительно Z

a-d – перемещение относительно X

w-s – перемещение относительно Y

Например, перемещаясь по оси Y можно увидеть изначальную плоскость таким образом:



**ВЫВОД**

В ходе выполнения лабораторной работы были получены навыки построение билинейной поверхности на основе ее 4 угловых точек.

**КОД ПРОГРАММЫ**

from OpenGL.GL import \*

from OpenGL.GLU import \*

from OpenGL.GLUT import \*

from math import cos, sin, radians

from random import choice

window = 0

points = []

rotatedPoints = []

angle = 0

eye = (6, 6, 6)

def createPoints(count, start, stop, step=0.1):

global points

listOfNumbers = []

result = start

while result < stop:

listOfNumbers.append(round(result, 1))

result += step

for i in range(count):

points.append([choice(listOfNumbers), choice(listOfNumbers), choice(listOfNumbers)])

def keyPressed(bkey, x, y):

global eye

# Convert bytes object to string

try:

key = bkey.decode("utf-8")

# Allow to quit by pressing 'Esc' or 'q'

if key == chr(27):

sys.exit()

if key == 'w':

eye = (eye[0], eye[1] + 1, eye[2])

if key == 's':

eye = (eye[0], eye[1] - 1, eye[2])

if key == 'd':

eye = (eye[0] + 1, eye[1], eye[2])

if key == 'a':

eye = (eye[0] - 1, eye[1], eye[2])

if key == 'q':

eye = (eye[0], eye[1], eye[2] - 1)

if key == 'e':

eye = (eye[0], eye[1], eye[2] + 1)

except:

pass

glutPostRedisplay()

def initPlane(vertices):

a = 15

k = [i / a for i in range(0, a + 1)]

point = [0, 0, 0]

for u in k:

glBegin(GL\_LINE\_STRIP)

glColor3f(0, 0, 1)

for w in k:

for i in range(3):

point[i] = (1 - u) \* (1 - w) \* vertices[0][i] + (1 - u) \* w \* vertices[1][i] + u \* (1 - w) \* vertices[2][i] + u \* w \* vertices[3][i]

glVertex3dv(point)

glEnd()

for w in k:

glBegin(GL\_LINE\_STRIP)

glColor3f(0, 0, 1)

for u in k:

for i in range(3):

point[i] = (1 - u) \* (1 - w) \* vertices[0][i] + (1 - u) \* w \* vertices[1][i] + u \* (1 - w) \* vertices[2][i] + u \* w \* vertices[3][i]

glVertex3dv(point)

glEnd()

def initGL(Width, Height):

glClearColor(1.0, 1.0, 1.0, 1.0)

glClearDepth(1.0)

glDepthFunc(GL\_LESS)

glEnable(GL\_DEPTH\_TEST)

glShadeModel(GL\_SMOOTH)

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

gluPerspective(45.0, float(Width) / float(Height), 0.1, 100.0)

glMatrixMode(GL\_MODELVIEW)

def init\_axes():

glBegin(GL\_LINES)

glColor3f(1, 0, 0)

glVertex3f(-4, 0, 0)

glVertex3f(4, 0, 0)

glColor3f(0, 1, 0)

glVertex3f(0, -4, 0)

glVertex3f(0, 4, 0)

glColor3f(0, 0, 1)

glVertex3f(0, 0, -4)

glVertex3f(0, 0, 4)

glEnd()

glPointSize(7)

glBegin(GL\_POINTS)

glColor3f(1, 0, 0)

glVertex3f(4, 0, 0)

glColor3f(0, 1, 0)

glVertex3f(0, 4, 0)

glColor3f(0, 0, 1)

glVertex3f(0, 0, 4)

glEnd()

def draw3DScene():

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

glLoadIdentity()

gluLookAt(eye[0], eye[1], eye[2], 0, 0, 0, 0, 1, 0)

init\_axes()

initPlane(points)

glutSwapBuffers()

def drawRotated3DScene():

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

glLoadIdentity()

gluLookAt(eye[0], eye[1], eye[2], 0, 0, 0, 0, 1, 0)

init\_axes()

initPlane(rotatedPoints)

glutSwapBuffers()

def initWindow():

global window

glutInit(sys.argv)

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH)

glutInitWindowSize(500, 500)

glutInitWindowPosition(500, 100)

window = glutCreateWindow(b"T3")

glutDisplayFunc(draw3DScene)

glutKeyboardFunc(keyPressed)

initGL(500, 500)

def matrixMultiplication(matrixOne, matrixTwo):

result = []

for i in range(len(matrixOne)):

result.append([0] \* 3)

for j in range(3):

for k in range(3):

result[i][j] += matrixOne[i][k] \* matrixTwo[k][j]

return result

def rotateOX(angle, vertices):

angle = radians(angle)

cosine = round(cos(angle), 5)

sine = round(sin(angle), 5)

rotationMatrix = [[1, 0, 0], [0, cosine, sine], [0, -sine, cosine]]

return matrixMultiplication(vertices, rotationMatrix)

def initSubWindow():

glutInit(sys.argv)

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH)

glutInitWindowSize(500, 500)

glutInitWindowPosition(1000, 100)

SubWindow = glutCreateWindow(b"Angle")

glutSetWindow(SubWindow)

glutDisplayFunc(drawRotated3DScene)

glutKeyboardFunc(keyPressed)

initGL(500, 500)

def main():

global points, rotatedPoints

createPoints(4, -3, 3)

angle = int(input("Angle of rotation about the OX axis:\n>>> "))

initWindow()

print(points)

if angle:

rotatedPoints = rotateOX(angle, points)

print(rotatedPoints)

initSubWindow()

glutMainLoop()

main()