Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИУК «Информатика и управление»***

КАФЕДРА ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

ЛАБОРАТОРНАЯ РАБОТА №1

«ВВЕДЕНИЕ В OPENGL»

ДИСЦИПЛИНА: «Компьютерная графика»

Выполнил: студент гр. ИУК4-41Б _____(___Дубовицкий Д.А.____)
(Подпись) (Ф.И.О.)

Проверил: _____(___Глебов С.А.____)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга , 2023 г.

Цель: формирование практических навыков по работе с проекционной матрицей средствами OpenGL, а также созданию простейших анимаций графических примитивов и их адаптации в абсолютных и относительных оконных координатах.

Задачи:

1. Сформировать представление о методах и секторе решаемых OpenGL задач
2. Изучить основные принципы работы OpenGL, представлять и понимать основные реализации OpenGL
3. Знать типы данных OpenGL и специфику именования переменных, понимать основные принципы трехмерного программирования компьютерной графики, иметь представление о проекциях, уметь создавать типовой проект в различных средах разработки (Visual Studio), иметь представление о двойной буферизации.

Вариант 8

Задание

Выполнить пульсирующее масштабирование с вращением прямоугольника

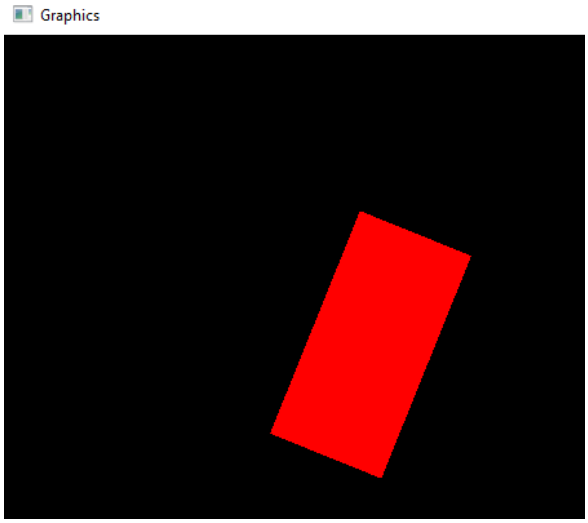
Листинг программы:

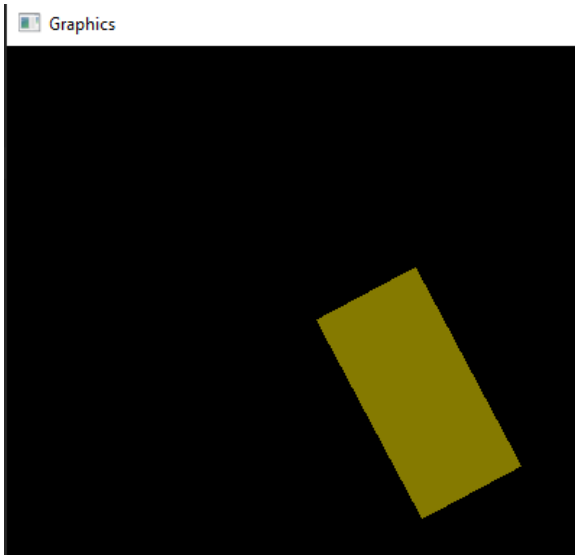
```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *

# window size
width = 600
height = 600

# rectangle position and size
rect_x = -0.3
rect_y = -0.3
rect_width = 0.5
rect_height = 1.0

# rotation angle and speed
angle = 0.0
angle_speed = 1

# scale factor and speed
scale = 1.0
scale_speed = 0.01

# pulsation parameters
color_speed = 0.08
color_factor = 0.0


def draw_rect():
    # set color
    glColor3f(1.0 - color_factor, color_factor, 0.0)

    # draw rectangle
    glBegin(GL_QUADS)
    glVertex2f(rect_x, rect_y)
    glVertex2f(rect_x + rect_width, rect_y)
    glVertex2f(rect_x + rect_width, rect_y + rect_height)
    glVertex2f(rect_x, rect_y + rect_height)
    glEnd()
```

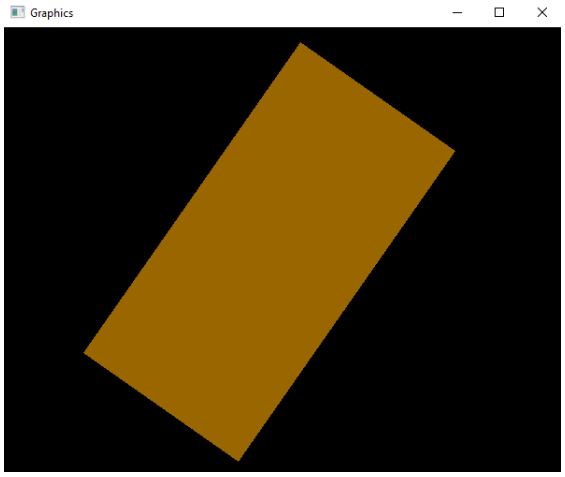
```
def display():
    global angle, scale, color_factor, scale_speed, color_speed

    # clear screen
    glClear(GL_COLOR_BUFFER_BIT)

    # set projection
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0)

    # set modelview
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    # rotate and scale rectangle
    glTranslatef(rect_x + rect_width / 2, rect_y + rect_height / 2, 0.0)
    glRotatef(angle, 0.0, 0.0, 1.0)
    glScalef(scale, scale, 1.0)
    glTranslatef(-rect_x - rect_width / 2, -rect_y - rect_height / 2, 0.0)

    # draw rectangle
    draw_rect()

    # update rotation angle
    angle += angle_speed

    # update scale factor
    scale += scale_speed
    if scale <= 0.5 or scale >= 1.5:
        scale_speed = -scale_speed

    # update pulsation factor
    color_factor += color_speed
    if color_factor <= 0.0 or color_factor >= 1.0:
        color_speed = -color_speed

    # update screen
    glutSwapBuffers()


def idle():
    glutPostRedisplay()


# initialize window
glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)
glutInitWindowSize(width, height)
glutCreateWindow("Graphics")

# set background color
glClearColor(0, 0, 0, 0.0)

# set callback functions
glutDisplayFunc(display)
glutIdleFunc(idle)
```

```
# start main loop
glutMainLoop()
```

Результаты работы программы:







Листинг 2:

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


def RenderScene():
    glClear(GL_COLOR_BUFFER_BIT)
```

```
    glFlush()


def SetupRC():
    glClearColor(0.0, 1.0, 0.5, 1.0)


glutInit()
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB)
glutCreateWindow("DANIIL")
glutDisplayFunc(RenderScene)
SetupRC()
glutMainLoop()
```

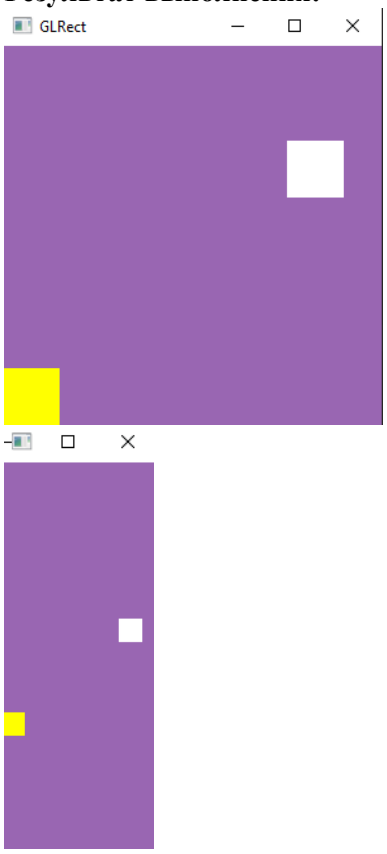
Результат выполнения:

Листинг 3:

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


def RenderScene():
    glClear(GL_COLOR_BUFFER_BIT)
    glColor3f(1.0, 1.0, 1.0)
    glRectf(50.0, 20.0, 80.0, 50.0)
    glColor3f(1, 1, 0)
    glRectf(-100, -70, -70, -100)
    glFlush()


def SetupRC():
    glClearColor(0.6, 0.4, 0.7, 1.0)


def ChangeSize(w, h):
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    aspectRatio = w / h
    if w <= h:
        glOrtho(-100.0, 100.0, -100/aspectRatio, 100.0/aspectRatio, 1.0, -1.0)
    else:
        glOrtho(-100.0 * aspectRatio, 100.0 * aspectRatio, -100.0, 100.0, 1.0, -1.0)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()


glutInit()
```

```
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB)
glutCreateWindow("GLRect")
glutDisplayFunc(RenderScene)
glutReshapeFunc(ChangeSize)
SetupRC()
glutMainLoop()
```

Результат выполнения:

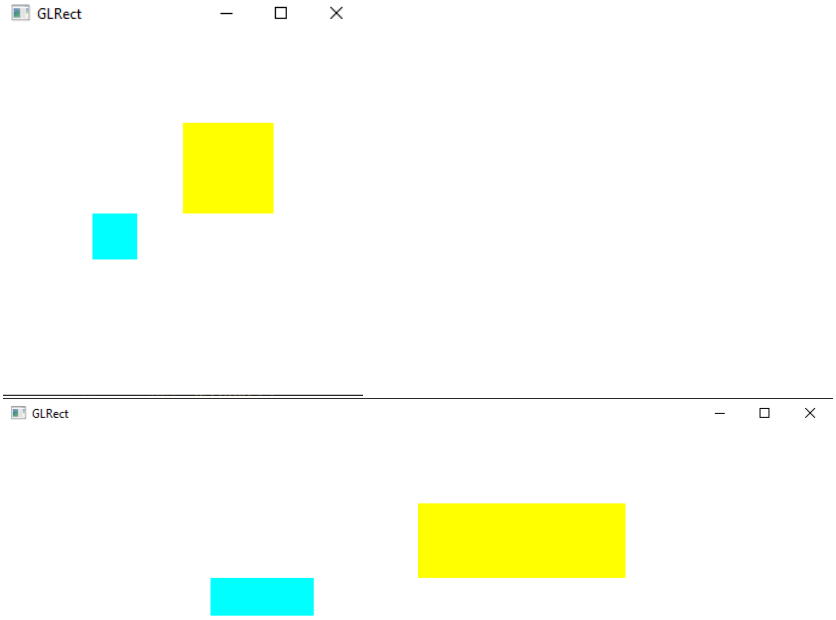


Листинг 4:

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


def RenderScene():
    glClear(GL_COLOR_BUFFER_BIT)
    glColor3f(0.0, 1.0, 1.0)
    glRectf(-50.0, 0.0, -25.0, -25.0)
    glColor3f(1.0, 1.0, 0.0)
    glRectf(0.0, 0.0, 50.0, 50.0)
    glFlush()


def SetupRC():
    glClearColor(1.0, 1.0, 1.0, 1.0);


def ChangeSize( w, h):
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(-100.0, 100.0, -100, 100.0, 1.0,-1.0)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
```

```
glutInit()
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB)
glutCreateWindow("GLRect")
glutDisplayFunc(RenderScene)
glutReshapeFunc(ChangeSize)
SetupRC()
glutMainLoop()
```

Результат выполнения:



Вывод: в ходе выполнения лабораторной работы были сформированы практические навыки по работе с проекционной матрицей средствами OpenGL, а также созданию простейших анимаций графических примитивов и их адаптации в абсолютных и относительных оконных координатах.