|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кафедра |  | O7 |  | Информационные системы и программная инженерия |
|  |  | шифр |  | наименование кафедры, по которой выполняется работа |
| Дисциплина |  | **Компьютерная графика и геометрия** | | |
|  |  | наименование дисциплины | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УЧЕБНО-ПРАКТИЧЕСКАЯ РАБОТА | | | | |
|  | №1 |  |
|  |  |  |
| ОБЩИЙ АЛГОРИТМ РАБОТЫ С OPENGL. | | | |
| СОЗДАНИЕ ОКОННОГО ПРИЛОЖЕНИЯ. | | | |
| СОЗДАНИЕ И ОТОБРАЖЕНИЕ ИЗОБРАЖЕНИЙ | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ОБУЧАЮЩИЙСЯ** | | | | | |
| группы | | | |  | О727Б |
|  |  | Смагин Т.В. | | | |
| подпись |  | фамилия и инициалы | | | |
|  | | | | |
| дата сдачи | | | | |
| **ПРОВЕРИЛ** | | | | | |
| к.пед.н., доцент, доцент | | | | | | |
| ученая степень, ученое звание, должность | | | | | | |
|  |  | Снижко Е.А. | | | | |
| подпись |  | фамилия и инициалы | | | | |
| Оценка / балльная оценка | | |  | | |
|  | | | | |
| дата проверки | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Санкт-Петербург | | | | | | |
|  |  |  | 20 | 24 | г. |  |

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc177045937)

[2 Реализация 4](#_Toc177045938)

[3 Демонстрация работы 8](#_Toc177045939)

# **1 Постановка задания**

Цель: создание программы-заготовки для работы с API OpenGL.

Порядок выполнения работы:

1. Создать проект в выбранной среде разработки и подключить к нему выбранные библиотеки для работы с API OpenGL.

2. Создать окно для вывода изображения.

3. С использованием шейдерных программ отобразить на окне три треугольника разных цветов. Для каждого треугольника следует создать свой фрагментный шейдер, определяющий его цвет.

# **2 Реализация**

import pygame

from pygame.locals import \*

from OpenGL.GL import \*

from OpenGL.GL.shaders import \*

vertices = [

-0.8, 0, 0.0,

-0.5, 0.8, 0.0,

-0.2, 0.0, 0.0,

0.8, 0, 0.0,

0.5, 0.8, 0.0,

0.2, 0.0, 0.0,

-0.4, -0.8, 0.0,

0.4, -0.8, 0.0,

0.0, 0.2, 0.0

]

vertex\_shader\_source = """

#version 400

in vec3 vp;

void main() {

gl\_Position = vec4(vp, 1.0);

}

"""

fragment\_shader\_1 = """

#version 400

out vec4 color;

void main() {

color = vec4(1.0, 0.0, 0.0, 1.0); // Красный

}

"""

fragment\_shader\_2 = """

#version 400

out vec4 color;

void main() {

color = vec4(0.0, 1.0, 0.0, 1.0); // Зеленый

}

"""

fragment\_shader\_3 = """

#version 400

out vec4 color;

void main() {

color = vec4(0.0, 0.0, 1.0, 1.0); // Синий

}

"""

def compile\_shader(source, shader\_type):

shader = glCreateShader(shader\_type)

glShaderSource(shader, source)

glCompileShader(shader)

return shader

def create\_shader\_program(vertex\_shader\_source, fragment\_shader\_source):

vertex\_shader = compile\_shader(vertex\_shader\_source, GL\_VERTEX\_SHADER)

fragment\_shader = compile\_shader(fragment\_shader\_source, GL\_FRAGMENT\_SHADER)

shader\_program = glCreateProgram()

glAttachShader(shader\_program, vertex\_shader)

glAttachShader(shader\_program, fragment\_shader)

glLinkProgram(shader\_program)

glDeleteShader(vertex\_shader)

glDeleteShader(fragment\_shader)

return shader\_program

def main():

pygame.init()

display = (800, 600)

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

pygame.display.set\_caption('Laba1')

glViewport(0, 0, display[0], display[1])

shader\_program\_1 = create\_shader\_program(vertex\_shader\_source, fragment\_shader\_1)

shader\_program\_2 = create\_shader\_program(vertex\_shader\_source, fragment\_shader\_2)

shader\_program\_3 = create\_shader\_program(vertex\_shader\_source, fragment\_shader\_3)

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

return

glClear(GL\_COLOR\_BUFFER\_BIT)

glUseProgram(shader\_program\_1)

glBegin(GL\_TRIANGLES)

for i in range(3):

glVertex3fv(vertices[i \* 3:i \* 3 + 3])

glEnd()

glUseProgram(shader\_program\_2)

glBegin(GL\_TRIANGLES)

for i in range(3):

glVertex3fv(vertices[9 + i \* 3:9 + i \* 3 + 3])

glEnd()

glUseProgram(shader\_program\_3)

glBegin(GL\_TRIANGLES)

for i in range(3):

glVertex3fv(vertices[18 + i \* 3:18 + i \* 3 + 3])

glEnd()

pygame.display.flip()

pygame.time.wait(10)

if \_\_name\_\_ == "\_\_main\_\_":

main()

# **3 Демонстрация работы**

Демонстрация работы получившейся программы представлена на рисунке 1.

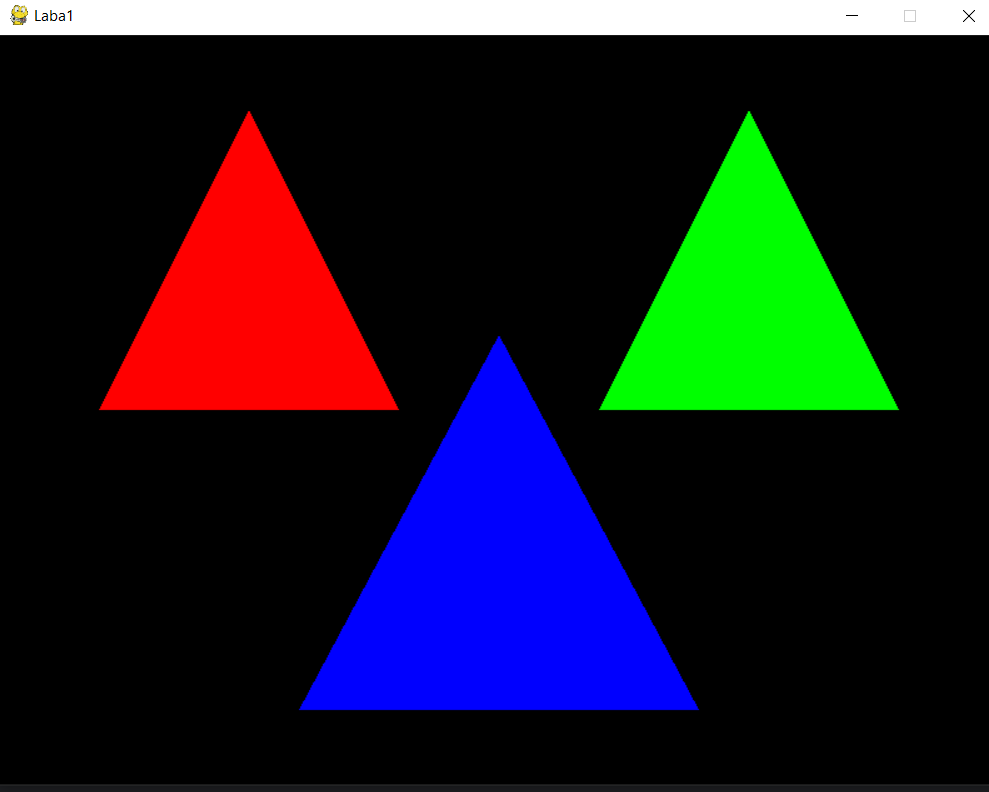


Рисунок 1 – Три треугольника