МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Компьютерная графика»
Тема: Примитивы OpenGL

Студент гр. 0382 Корсунов А.А.

Преподаватель Герасимова Т.В.

Санкт-Петербург
2023

Цель работы.

На базе разработанной оболочки из 1-ой лабораторной работы разработать программу, реализующую представление тестов отсечения (glScissor), прозрачности (glAlphaFunc), смешения цветов (glBlendFunc) в библиотеке OpenGL на базе разработанных в предыдущей работе примитивов.

Теоретические положения.

Управление режимами работы в OpenGL осуществляется при помощи двух команд - glEnable и glDisable, одна из которых включает, а вторая выключает некоторый режим.

void glEnable(GLenum cap)
void glDisable(GLenum cap)

Обе команды имеют один аргумент – cap, который может принимать значения определяющие тот или иной режим, например, GL_ALPHA_TEST, GL_BLEND, GL_SCISSOR_TEST и многие другие.

Тест отсечения

Режим GL_SCISSOR_TEST разрешает отсечение тех фрагментов объекта, которые находятся вне прямоугольника "вырезки".
Прямоугольник "вырезки" определяется функцией glScissor:

void glScissor(GLint x, GLint y, GLsizei width, GLsizei height);
где параметры

- x, y определяют координаты левого нижнего угла прямоугольника «вырезки», исходное значение - (0,0).
- width, height - ширина и высота прямоугольника «вырезки».

Тест прозрачности

Режим GL_ALPHA_TEST задает тестирование по цветовому параметру альфа.Функция glAlphaFunc устанавливает функцию тестирования параметра альфа.

void glAlphaFunc(GLenum func, GLclampf ref)

где параметр – func может принимать следующие значения:

GL_NEVER – никогда не пропускает

GL_LESS – пропускает, если входное значение альфа меньше, чем значение ref

GL_EQUAL – пропускает, если входное значение альфа равно значению ref

GL_LEQUAL – пропускает, если входное значение альфа меньше или равно значения ref

GL_GREATER – пропускает, если входное значение альфа больше, чем значение ref

GL_NOTEQUAL – пропускает, если входное значение альфа не равно значению ref

GL_GEQUAL – пропускает, если входное значение альфа больше или равно значения ref

GL_ALWAYS – всегда пропускается, по умолчанию,

а параметр ref – определяет значение, с которым сравнивается входное значение альфа. Он может принимать значение от 0 до 1, причем 0 представляет наименьшее возможное значение альфа, а 1 – наибольшее. По умолчанию ref равен 0.

Тест смешения цветов

Режим GL_BLEND разрешает смешивание поступающих значений цветов

RGBA со значениями, находящимися в буфере цветов.
Функция glBlendFunc устанавливает пиксельную арифметику.

void glBlendFunc(GLenum sfactor, GLenum dfactor);
где параметры

- sfactor устанавливает способ вычисления входящих факторов смешения RGBA. Может принимать одно из следующих значений – GL_ZERO, GL_ONE, GL_DST_COLOR, GL_ONE_MINUS_DST_COLOR, GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA, GL_DST_ALPHA, GL_ONE_MINUS_DST_ALPHA и GL_SRC_ALPHA_SATURATE.

- dfactor устанавливает способ вычисления факторов смешения RGBA, уже находящихся в буфере кадра. Может принимать одно из следующих значений – GL_ZERO, GL_ONE, GL_SRC_COLOR, GL_ONE_MINUS_SRC_COLOR, GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA, GL_DST_ALPHA и GL_ONE_MINUS_DST_ALPHA.

Прозрачность лучше организовывать используя команду glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA). Такой же вызов применяют для устранения ступенчатости линий и точек. Для устранения ступенчатости многоугольников применяют вызов команды glBlendFunc(GL_SRC_ALPHA_SATURATE, GL_ONE).

Задание.

На базе разработанной вами оболочки из 1 работы разработать программу реализующую представление тестов отсечения (glScissor), прозрачности (glAlphaFunc), смешения цветов (glBlendFunc) в библиотеке OpenGL на базе разработанных вами в предыдущей работе примитивов. Разработанная на базе шаблона программа должна быть пополнена возможностями остановки интерактивно различных атрибутов тестов через вызов соответствующих элементов интерфейса пользователя

Ход работы.

1. Тест отсечения.

а) Демонстрация работы.

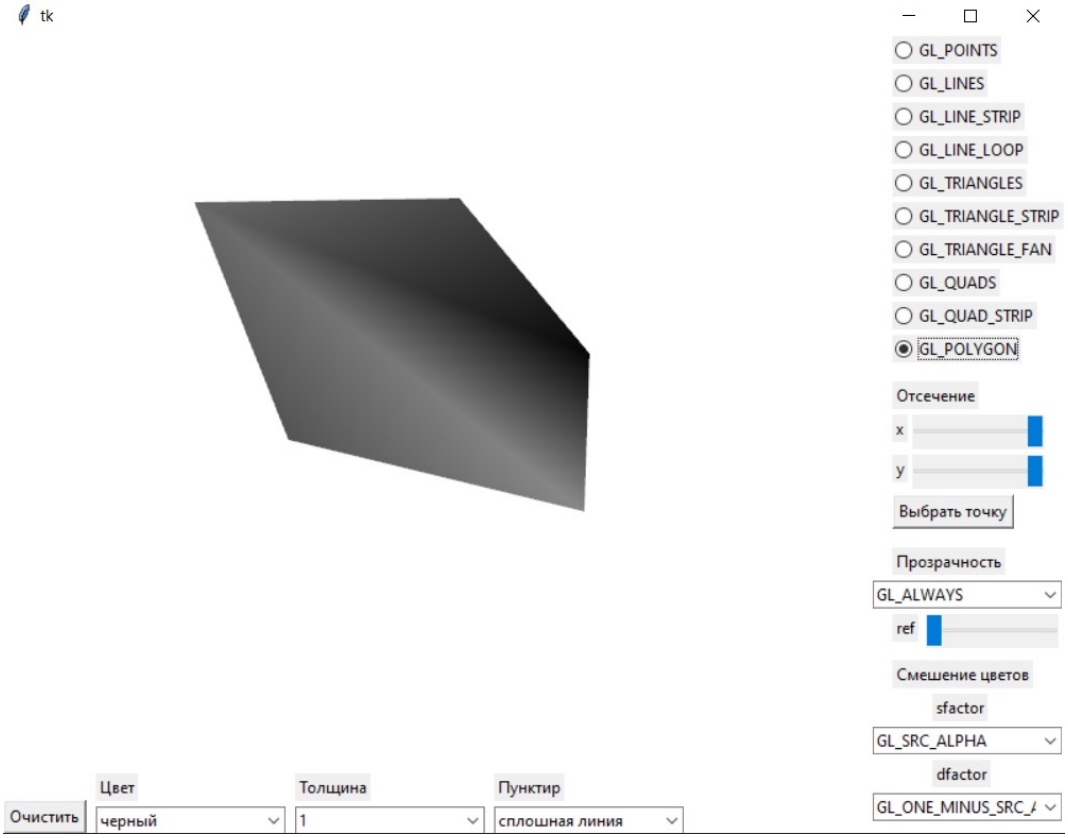

Рисунок 1 — Тест отсечения, виджет

За тест отсечения отвечают 3-и виджета (рис. 1, справа): «x», «y» – увеличение отсечения по координатам x и y соответственно, кнопка «Выбрать точку» позволяет выбрать последнюю созданную вершину как левый верхний угол отсечения (сама же вершина при нажатии будет удалена).

Для примера пусть в центр прошлой фигуры будет добавлена вершина, после чего она будет выбрана левым верхним углом отсечения, а затем будет уменьшены размеры отсечения путем уменьшения значений x и y:

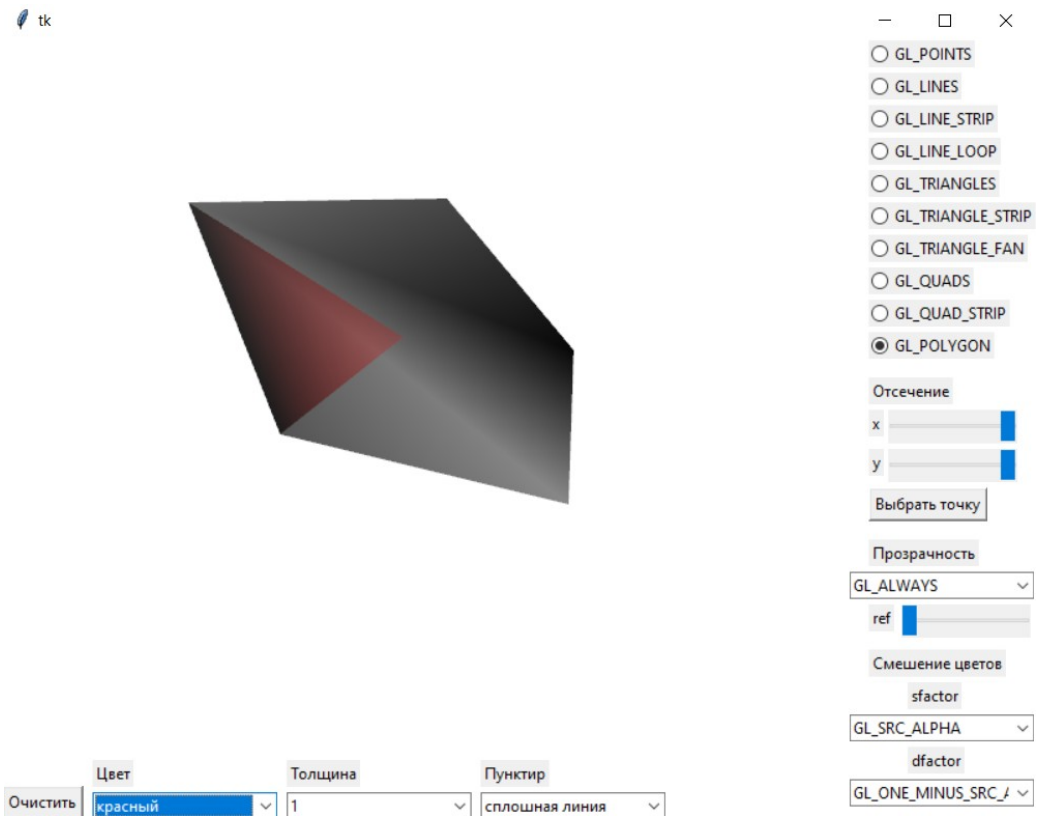

Рисунок 2 — Добавление новой вершины к предыдущей фигуре

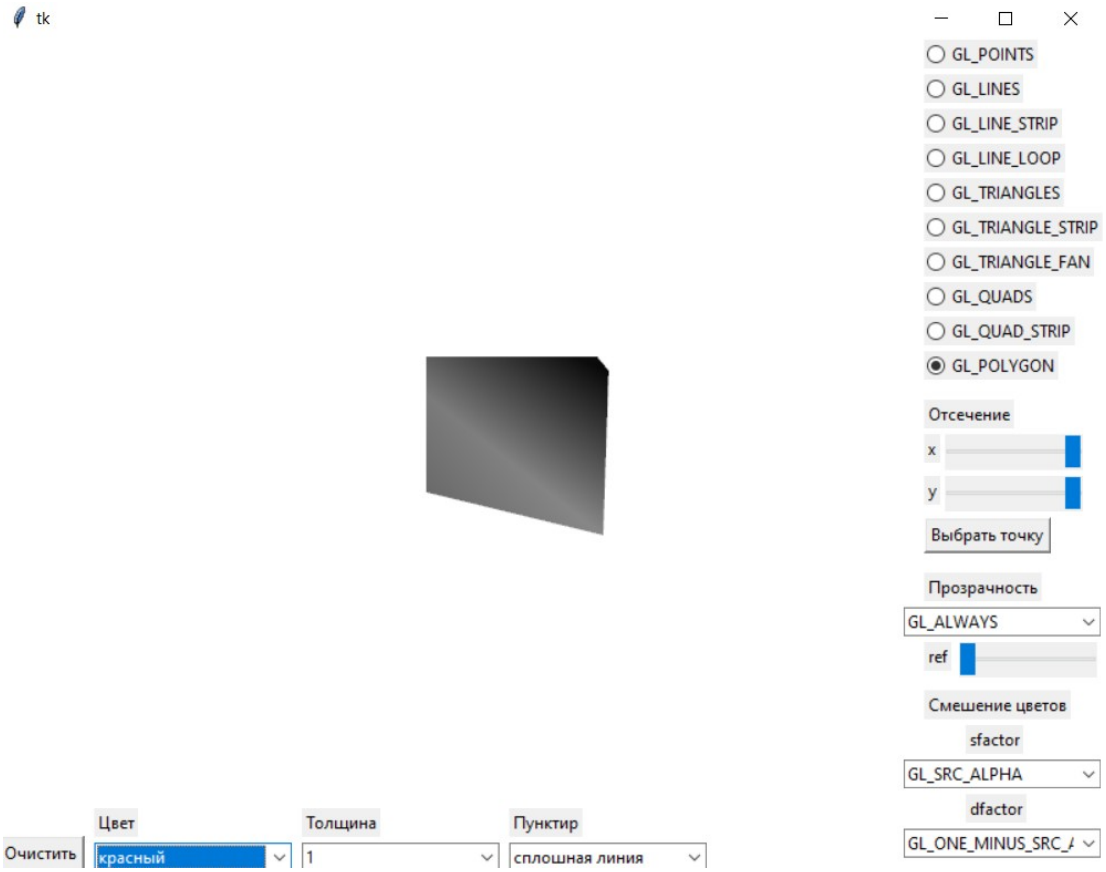

Рисунок 3 — Добавленная на прошлом шаге вершина была выбрана левым верхним углом отсечения

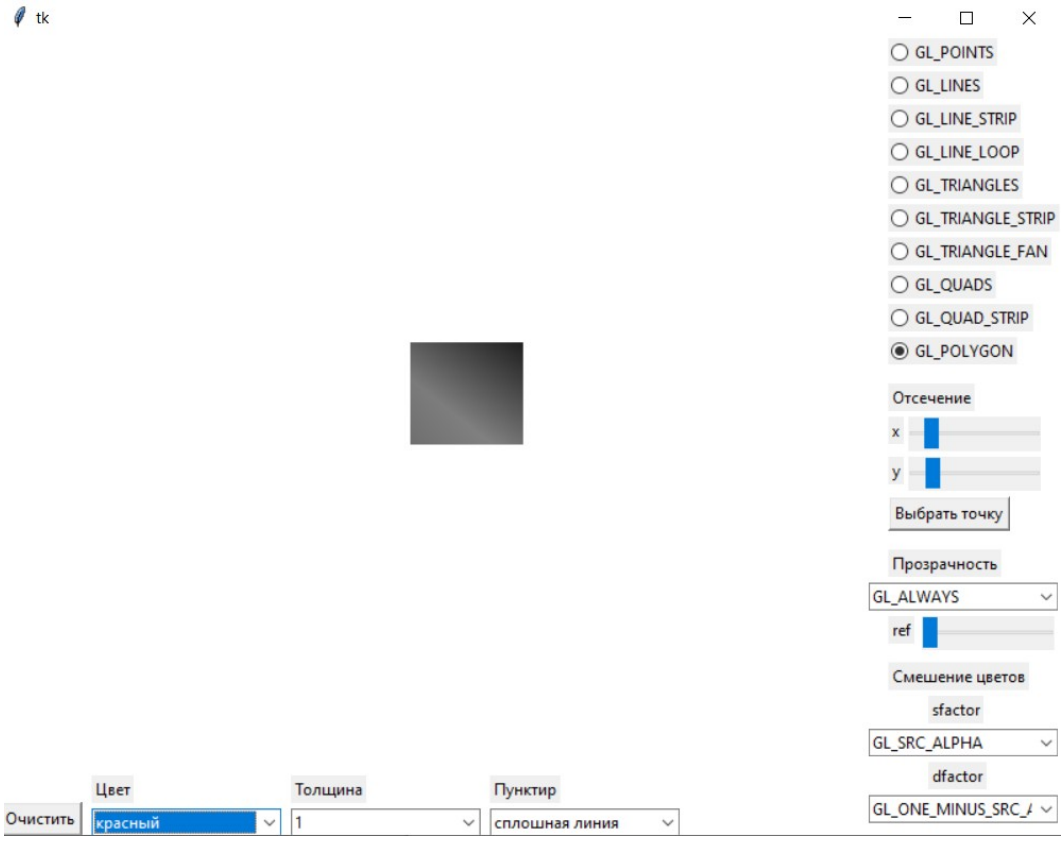

Рисунок 4 — Уменьшение размеров отсечения

б) Фрагменты кода.

```
...
GL.glEnable(GL.GL_ALPHA_TEST)
GL.glEnable(GL.GL_SCISSOR_TEST)
GL.glEnable(GL.GL_BLEND)

scissor_top_left(start_point_scissor[0], start_point_scissor[1], int(scale_scissor_x.get()),
        int(scale_scissor_y.get()))
GL.glAlphaFunc(get_Alpha(), float(scale_Alpha_ref.get()))
GL.glBlendFunc(get_sfactor(), get_dfactor())
...
```

с помощью *GL.glEnable(GL.GL_SCISSOR_TEST)* включается режим отсечения, а с помощью функции *scissor_top_left* передаются координаты левого верхнего угла и размеры отсечения

```
...
def scissor_top_left(x, y, w, h):  # вспомогательная функция для теста отсечения
   GL.glScissor(x, window_height - h - y, w, h)
...
```

Эта функция использует glScissor, изменяя входные параметры, это необходимо, т. к. система координат glScissor отличается от системы координат остальных функций программы. Значения, устанавливающиеся в эту функцию, передаются пользователем через виджеты (таким же образом значения передавались и в предыдущей лабораторной работе).

2. Тест прозрачности.

Для теста прозрачности были реализованы 2-а виджета (рис. 1, справа): первый позволяет выбрать один из режимов (режимы описаны в теоретических положениях), второй позволяет изменять параметр «ref» с помощью «ползунка» (от 0 до 1). При создании вершин, последний аргумент (отвечающий за прозрачность) задается случайно в диапазоне от 0 до 1.

а) Демонстрация работы.

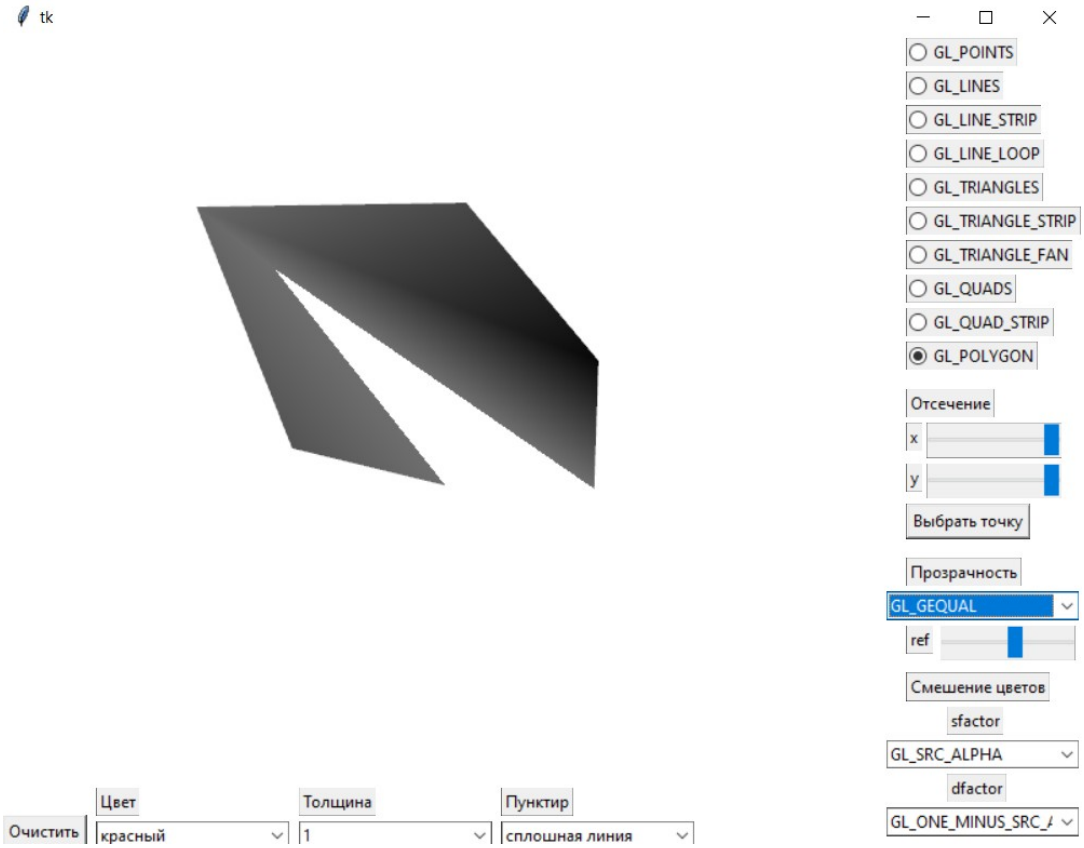

Рисунок 5 - Для начальной фигуры (рис. 1) был выбран режим GL_GEQUAL и изменено значение ref

б) Фрагменты кода.

```
...
GL.glEnable(GL.GL_ALPHA_TEST)
GL.glEnable(GL.GL_SCISSOR_TEST)
GL.glEnable(GL.GL_BLEND)

scissor_top_left(start_point_scissor[0], start_point_scissor[1], int(scale_scissor_x.get()),
            int(scale_scissor_y.get()))
GL.glAlphaFunc(get_Alpha(), float(scale_Alpha_ref.get()))
GL.glBlendFunc(get_sfactor(), get_dfactor())
...
```

с помощью *GL.glEnable(GL.GL_ALPHA_TEST)* включается режим прозрачности, а с помощью функции *GL.glAlphaFunc* передаются режим и ref для теста прозрачности.

get_Alpha, scale_Alpha_ref.get - функции, которые позволяют получать значения из виджетов, установленных пользователем.

3. Тест смешения цветов.

Для теста смешения цветов были реализованы 2-а виджета (рис. 1, справа): оба позволяют выбрать 1 из режимов, описанных в теоретических положениях для параметров sfactor и dfactor.

Для прозрачности по умолчанию выставлены режимы GL_SRC_ALPHA для sfactor и GL_ONE_MINUS_SRC_ALPHA для dfactor (рис. 1)

а) Демонстрация работы.

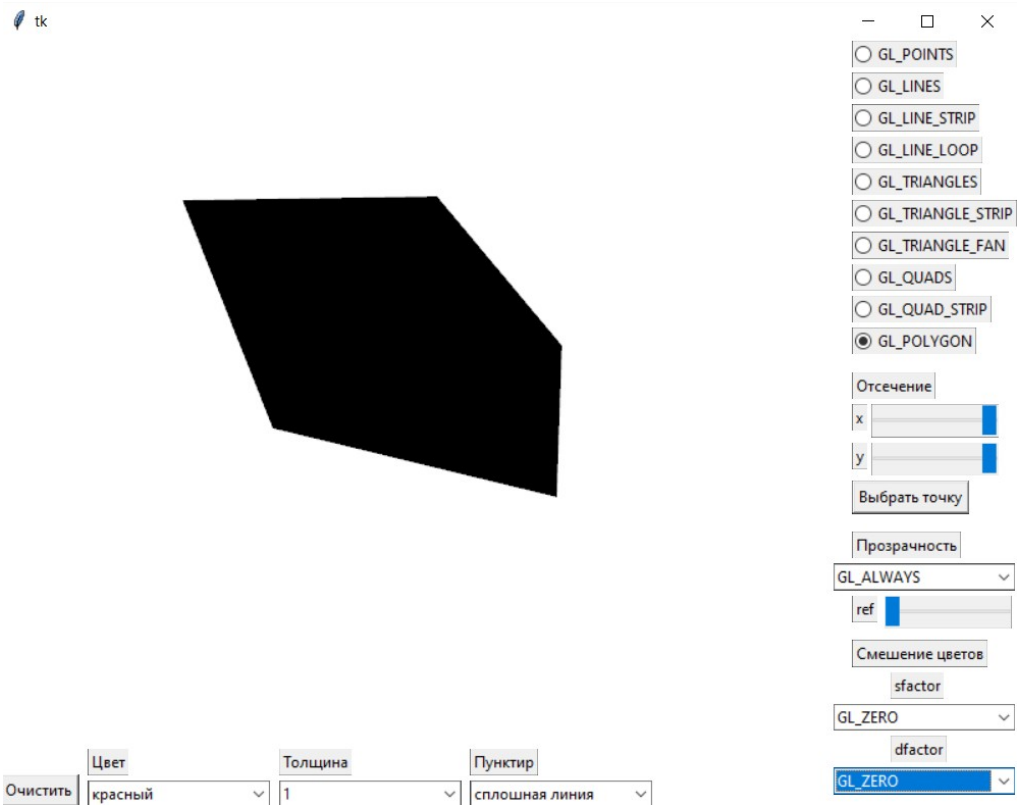

Рисунок 6 - Для начальной фигуры (рис. 1) был выбраны режимы GL_ZERO и GL_ZERO

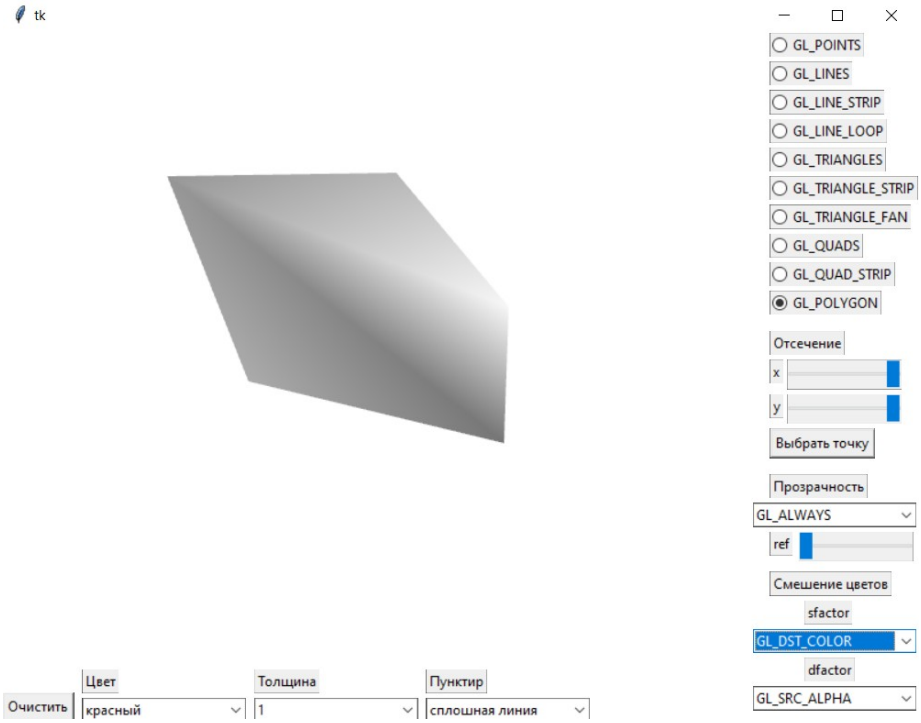

Рисунок 7 — Для начальной фигуры были выбраны режим GL_DST_COLOR и GL_SRC_ALPHA

б) Фрагменты кода.

```
...
GL.glEnable(GL.GL_ALPHA_TEST)
GL.glEnable(GL.GL_SCISSOR_TEST)
GL.glEnable(GL.GL_BLEND)

scissor_top_left(start_point_scissor[0], start_point_scissor[1], int(scale_scissor_x.get()),
            int(scale_scissor_y.get()))
GL.glAlphaFunc(get_Alpha(), float(scale_Alpha_ref.get()))
GL.glBlendFunc(get_sfactor(), get_dfactor())
...
```

с помощью *GL.glEnable(GL.GL_BLEND)* включается режим смешения цветов, а с помощью функции *GL.glBlendFunc* передаются режимы для теста смешения цветов.

get_sfactor, get_dfactor - функции, которые позволяют получать значения из виджетов, установленных пользователем.

4. Изменения функции окрашивания вершин

В предыдущей лабораторной работе виджет, который отвечал за окраску вершин менял все цвета вершин сразу, в этой лабораторной работе он был переделан — теперь цвет выбирается непосредственно для следующей создаваемой вершины (таким образом вершины теперь могут иметь разные цвета):

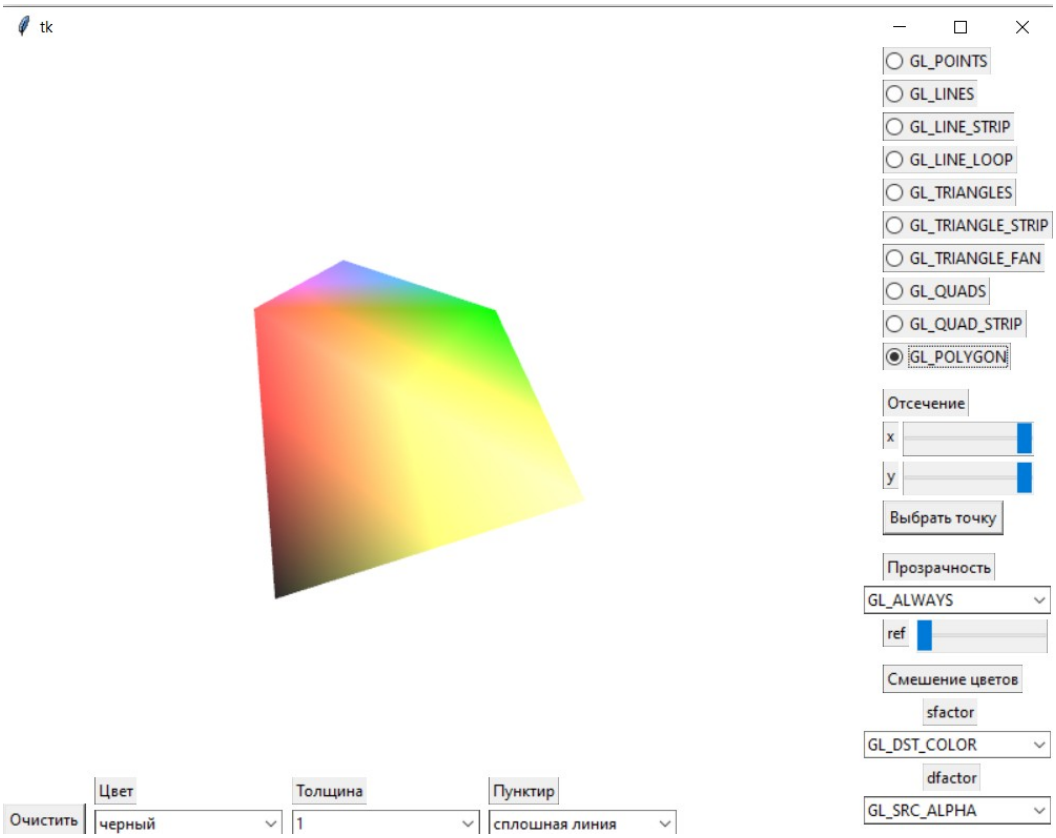

Рисунок 8 — Демонстрация изменения функции покраски вершин

Вывод.

На базе разработанной оболочки из 1-ой лабораторной работы была разработана программа, реализующая представление тестов отсечения (glScissor), прозрачности (glAlphaFunc), смешения цветов (glBlendFunc) в библиотеке OpenGL на базе разработанных в предыдущей работе примитивов.

Приложение А. Исходный код.

Файл main.py

```
from tkinter import *
from pyopengltk import OpenGLFrame
from OpenGL import GL, GLU
from tkinter.ttk import Combobox
from tkinter.ttk import Radiobutton
from tkinter.ttk import Scale
from random import uniform


def scissor_click():  # вспомогательная функция для задания начальной точки
    if len(coordinates) > 0:
        start_point_scissor[0] = coordinates[-1][0]
        start_point_scissor[1] = coordinates[-1][1]
        del coordinates[-1]


def scissor_top_left(x, y, w, h):  # вспомогательная функция для теста отсечения
    GL.glScissor(x, window_height - h - y, w, h)


def get_dfactor():  # функция для установки dfactor
    temp_dfactor = combo_blend_dfactor.get()
    if temp_dfactor == "GL_ZERO":
        temp_dfactor = GL.GL_ZERO
    elif temp_dfactor == "GL_ONE":
        temp_dfactor = GL.GL_ONE
    elif temp_dfactor == "GL_SRC_COLOR":
        temp_dfactor = GL.GL_SRC_COLOR
    elif temp_dfactor == "GL_ONE_MINUS_SRC_COLOR":
        temp_dfactor = GL.GL_ONE_MINUS_SRC_COLOR
    elif temp_dfactor == "GL_SRC_ALPHA":
        temp_dfactor = GL.GL_SRC_ALPHA
    elif temp_dfactor == "GL_ONE_MINUS_SRC_ALPHA":
        temp_dfactor = GL.GL_ONE_MINUS_SRC_ALPHA
    elif temp_dfactor == "GL_DST_ALPHA":
        temp_dfactor = GL.GL_DST_ALPHA
    elif temp_dfactor == "GL_ONE_MINUS_DST_ALPHA":
        temp_dfactor = GL.GL_ONE_MINUS_DST_ALPHA
    return temp_dfactor


def get_sfactor():  # функция для установки sfactor
    temp_sfactor = combo_blend_sfactor.get()
    if temp_sfactor == "GL_ZERO":
        temp_sfactor = GL.GL_ZERO
    elif temp_sfactor == "GL_ONE":
```

```
        temp_sfactor = GL.GL_ONE
    elif temp_sfactor == "GL_DST_COLOR":
        temp_sfactor = GL.GL_DST_COLOR
    elif temp_sfactor == "GL_ONE_MINUS_DST_COLOR":
        temp_sfactor = GL.GL_ONE_MINUS_DST_COLOR
    elif temp_sfactor == "GL_SRC_ALPHA":
        temp_sfactor = GL.GL_SRC_ALPHA
    elif temp_sfactor == "GL_ONE_MINUS_SRC_ALPHA":
        temp_sfactor = GL.GL_ONE_MINUS_SRC_ALPHA
    elif temp_sfactor == "GL_DST_ALPHA":
        temp_sfactor = GL.GL_DST_ALPHA
    elif temp_sfactor == "GL_ONE_MINUS_DST_ALPHA":
        temp_sfactor = GL.GL_ONE_MINUS_DST_ALPHA
    elif temp_sfactor == "GL_SRC_ALPHA_SATURATE":
        temp_sfactor = GL.GL_SRC_ALPHA_SATURATE
    return temp_sfactor


def get_Alpha():  # функция для установки прозрачности
    temp_Alpha = combo_Alpha.get()
    if temp_Alpha == "GL_NEVER":
        temp_Alpha = GL.GL_NEVER
    elif temp_Alpha == "GL_LESS":
        temp_Alpha = GL.GL_LESS
    elif temp_Alpha == "GL_EQUAL":
        temp_Alpha = GL.GL_EQUAL
    elif temp_Alpha == "GL_LEQUAL":
        temp_Alpha = GL.GL_LEQUAL
    elif temp_Alpha == "GL_GREATER":
        temp_Alpha = GL.GL_GREATER
    elif temp_Alpha == "GL_NOTEQUAL":
        temp_Alpha = GL.GL_NOTEQUAL
    elif temp_Alpha == "GL_GEQUAL":
        temp_Alpha = GL.GL_GEQUAL
    elif temp_Alpha == "GL_ALWAYS":
        temp_Alpha = GL.GL_ALWAYS
    return temp_Alpha


def get_stipple():  # функция для установки пунктира
    temp_stipple = combo_stipple.get()
    if temp_stipple == "сплошная линия":
        GL.glLineStipple(1, 0xFFFF)
    elif temp_stipple == "точечный пунктир":
        GL.glLineStipple(1, 0x0101)
    elif temp_stipple == "штриховой пунктир":
        GL.glLineStipple(1, 0x00FF)
    elif temp_stipple == "штрих точка штрих":
        GL.glLineStipple(1, 0x1C47)
```

```
def get_color():  # функция для установки цвета
    temp_color = combo_color.get()
    if temp_color == "красный":
        color[0] = 255
        color[1] = 0
        color[2] = 0
        color[3] = uniform(0, 1)
    elif temp_color == "синий":
        color[0] = 0
        color[1] = 0
        color[2] = 255
        color[3] = uniform(0, 1)
    elif temp_color == "зеленый":
        color[0] = 0
        color[1] = 255
        color[2] = 0
        color[3] = uniform(0, 1)
    elif temp_color == "желтый":
        color[0] = 255
        color[1] = 255
        color[2] = 0
        color[3] = uniform(0, 1)
    elif temp_color == "черный":
        color[0] = 0
        color[1] = 0
        color[2] = 0
        color[3] = uniform(0, 1)


def clear():  # очистка списка координат вершин
    coordinates.clear()
    start_point_scissor[0] = 0
    start_point_scissor[1] = 0


def click(event):  # добавление координаты вершины по клику в окне
    if event.x < 670 and event.y < 555:
        get_color()
        temp_color = color.copy()
        coordinates.append((event.x, event.y, temp_color))


def draw():  # отрисовка
    primitive = GL.GL_POINT  # примитив по умолчанию
    line_width = combo_thickness.get()  # получение значения из виджета толщины
    temp = selected.get()  # получение примитива из выбранного виджета
```

```
    if temp == 0:  # установка примитивов для рисования (+толщины и типа линий для
примитивов вида линий)
        primitive = GL.GL_POINTS
    elif temp == 1:
        primitive = GL.GL_LINES
        GL.glLineWidth(int(line_width))
        GL.glEnable(GL.GL_LINE_STIPPLE)
        get_stipple()
    elif temp == 2:
        primitive = GL.GL_LINE_STRIP
        GL.glLineWidth(int(line_width))
        GL.glEnable(GL.GL_LINE_STIPPLE)
        get_stipple()
    elif temp == 3:
        primitive = GL.GL_LINE_LOOP
        GL.glLineWidth(int(line_width))
        GL.glEnable(GL.GL_LINE_STIPPLE)
        get_stipple()
    elif temp == 4:
        primitive = GL.GL_TRIANGLES
    elif temp == 5:
        primitive = GL.GL_TRIANGLE_STRIP
    elif temp == 6:
        primitive = GL.GL_TRIANGLE_FAN
    elif temp == 7:
        primitive = GL.GL_QUADS
    elif temp == 8:
        primitive = GL.GL_QUAD_STRIP
    elif temp == 9:
        primitive = GL.GL_POLYGON

    GL.glEnable(GL.GL_ALPHA_TEST)
    GL.glEnable(GL.GL_SCISSOR_TEST)
    GL.glEnable(GL.GL_BLEND)

    scissor_top_left(start_point_scissor[0], start_point_scissor[1], int(scale_scissor_x.get()),
                     int(scale_scissor_y.get()))
    GL.glAlphaFunc(get_Alpha(), float(scale_Alpha_ref.get()))
    GL.glBlendFunc(get_sfactor(), get_dfactor())

    GL.glBegin(primitive)  # непосредстенно отрисова виджета
    for i in range(len(coordinates)):  # в цикле отрисовка вершин по координатам из списка
        temp_color = coordinates[i][2]
        GL.glColor4f(temp_color[0], temp_color[1], temp_color[2], temp_color[3])  # установка
цвета
        GL.glVertex3f(coordinates[i][0], coordinates[i][1], 0.0)
    GL.glEnd()

    GL.glFlush()
```

```
class DrawingWindow(OpenGLFrame):  # создание класса на основе пакета pyopengltk

    def initgl(self):  # инициализация
        GL.glClear(GL.GL_COLOR_BUFFER_BIT)  # очистка буфферов от цветов ~ очищение экарана
        GL.glClearColor(1, 1, 1, 0)  # задает цвет, в который окно будет окрашиваться при его очистке ~ очищение
        # цветопередачи
        GL.glMatrixMode(GL.GL_PROJECTION)  # матрица проекции (для проецирования 3D прсостранства в 2D)
        GL.glLoadIdentity()  # единичная матрица ~ очистика
        GLU.gluOrtho2D(0, window_width, window_height, 0)  # смещецение оси координат (чтобы не возникало искажения при
        # отрисовке

    def redraw(self):  # перерисовка
        GL.glClear(GL.GL_COLOR_BUFFER_BIT)
        draw()  # функция для отрисовки


root = Tk()  # главное окно

window_width = 800  # размеры окна (ширина и высота)
window_height = 600
coordinates = []  # (координаты вершин)
color = [0, 0, 0, 0.0]  # (цвет  вершин)

app = DrawingWindow(root, width=window_width, height=window_height)  # создание окна для отрисови
app.bind('<Button 1>', click)  # биндим ПКМ по созданному окну, чтобы запоминать координаты клика
app.pack(fill=BOTH, expand=YES)  # отобразить
app.animate = 1  # для отрисовки в реальном времени

selected = IntVar()  # сюда помещается значение выбранного примитива
rad1 = Radiobutton(app, text='GL_POINTS', value=0, variable=selected)  # радиокнопки для выбора примитива
rad2 = Radiobutton(app, text='GL_LINES', value=1, variable=selected)
rad3 = Radiobutton(app, text='GL_LINE_STRIP', value=2, variable=selected)
rad4 = Radiobutton(app, text='GL_LINE_LOOP', value=3, variable=selected)
rad5 = Radiobutton(app, text='GL_TRIANGLES', value=4, variable=selected)
rad6 = Radiobutton(app, text='GL_TRIANGLE_STRIP', value=5, variable=selected)
rad7 = Radiobutton(app, text='GL_TRIANGLE_FAN', value=6, variable=selected)
rad8 = Radiobutton(app, text='GL_QUADS', value=7, variable=selected)
rad9 = Radiobutton(app, text='GL_QUAD_STRIP', value=8, variable=selected)
rad10 = Radiobutton(app, text='GL_POLYGON', value=9, variable=selected)
```

```
rad1.place(x=670, y=0)  # их отрбражение
rad2.place(x=670, y=25)
rad3.place(x=670, y=50)
rad4.place(x=670, y=75)
rad5.place(x=670, y=100)
rad6.place(x=670, y=125)
rad7.place(x=670, y=150)
rad8.place(x=670, y=175)
rad9.place(x=670, y=200)
rad10.place(x=670, y=225)

btn = Button(app, text="Очистить", command=clear)  # кнопка для очистки
btn.place(x=0, y=575)

label_color = Label(text="Цвет")  # виджеты для выбора цвета вершин
label_color.place(x=70, y=555)
combo_color = Combobox(app)
combo_color['values'] = ("красный", "синий", "зеленый", "желтый", "черный")
combo_color['state'] = 'readonly'
combo_color.current(4)
combo_color.place(x=70, y=580)

label_thickness = Label(text="Толщина")  # виджеты для выбора толщины линий
label_thickness.place(x=220, y=555)
combo_thickness = Combobox(app)
combo_thickness['values'] = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
combo_thickness['state'] = 'readonly'
combo_thickness.current(0)
combo_thickness.place(x=220, y=580)

label_stipple = Label(text="Пунктир")  # виджеты для выбора типа пунктира
label_stipple.place(x=370, y=555)
combo_stipple = Combobox(app)
combo_stipple['values'] = ("сплошная линия", "точечный пунктир", "штриховой пунктир",
"штрих точка штрих")
combo_stipple['state'] = 'readonly'
combo_stipple.current(0)
combo_stipple.place(x=370, y=580)

start_point_scissor = [0, 0]
label_scissor = Label(text="Отсечение")  # виджеты для теста отсечения
label_scissor.place(x=670, y=260)
label_scissor_x = Label(text="x")
label_scissor_x.place(x=670, y=285)
scale_scissor_x = Scale(from_=start_point_scissor[0], to=670, orient=HORIZONTAL, value=670)
scale_scissor_x.place(x=685, y=285)
label_scissor_y = Label(text="y")
label_scissor_y.place(x=670, y=315)
scale_scissor_y = Scale(from_=start_point_scissor[1], to=555, orient=HORIZONTAL, value=555)
```

```
scale_scissor_y.place(x=685, y=315)
scale_button = Button(app, text="Выбрать точку", command=scissor_click)
scale_button.place(x=670, y=345)

label_Alpha = Label(text="Прозрачность")  # виджеты для теста прозрачности
label_Alpha.place(x=670, y=385)
combo_Alpha = Combobox(app)
combo_Alpha['values'] = ("GL_NEVER", "GL_LESS", "GL_EQUAL", "GL_LEQUAL", "GL_GREATER",
"GL_NOTEQUAL", "GL_GEQUAL",
                         "GL_ALWAYS")
combo_Alpha['state'] = 'readonly'
combo_Alpha.current(7)
combo_Alpha.place(x=655, y=410)
label_Alpha_ref = Label(text="ref")
label_Alpha_ref.place(x=670, y=435)
scale_Alpha_ref = Scale(from_=0, to=1, orient=HORIZONTAL, value=0)
scale_Alpha_ref.place(x=695, y=435)

label_blend = Label(text="Смешение цветов")  # виджеты для теста смешения цветов
label_blend.place(x=670, y=470)
label_blend_sfactor = Label(text="sfactor")
label_blend_sfactor.place(x=700, y=495)
combo_blend_sfactor = Combobox(app)
combo_blend_sfactor['values'] = ("GL_ZERO", "GL_ONE", "GL_DST_COLOR",
"GL_ONE_MINUS_DST_COLOR", "GL_SRC_ALPHA",
                          "GL_ONE_MINUS_SRC_ALPHA", "GL_DST_ALPHA",
"GL_ONE_MINUS_DST_ALPHA",
                          "GL_SRC_ALPHA_SATURATE")
combo_blend_sfactor['state'] = 'readonly'
combo_blend_sfactor.current(4)
combo_blend_sfactor.place(x=655, y=520)
label_blend_dfactor = Label(text="dfactor")
label_blend_dfactor.place(x=700, y=545)
combo_blend_dfactor = Combobox(app)
combo_blend_dfactor['values'] = ("GL_ZERO", "GL_ONE", "GL_SRC_COLOR",
"GL_ONE_MINUS_SRC_COLOR", "GL_SRC_ALPHA",
                          "GL_ONE_MINUS_SRC_ALPHA", "GL_DST_ALPHA",
"GL_ONE_MINUS_DST_ALPHA")
combo_blend_dfactor['state'] = 'readonly'
combo_blend_dfactor.current(5)
combo_blend_dfactor.place(x=655, y=570)

app.mainloop()  # запуск главного цикла
```