**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**

**(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кафедра |  |  |  |  |
|  |  | шифр |  | наименование кафедры, по которой выполняется работа |
| Дисциплина |  |  | | |
|  |  | наименование дисциплины | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УЧЕБНО-ПРАКТИЧЕСКАЯ РАБОТА | | | | |
|  | 2 |  |
|  | номер задания (при наличии) |  |
| ПРИМИТИВЫ OPENGL, ОСНОВНЫЕ ПРИЁМЫ ПОСТРОЕНИЯ ДВУМЕРНЫХ ОБЪЕКТОВ. ОСНОВЫ ШЕЙДЕРОВ | | | |
|  | | | |

при наличии указать тему учебно-практической работы и (или) номер варианта

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ОБУЧАЮЩИЙСЯ** | | | | | | |
| группы | | | |  | | о722б |
|  |  | Смыгалов Д. Д. | | | | |
| подпись |  | фамилия и инициалы | | | | |
| 24.10.2024 | | | | |
| дата сдачи | | | | |
| **ПРОВЕРИЛ** | | | | | | |
|  |  | Снижко Е. А. | | | | |
| подпись |  | фамилия и инициалы | | | | |
| Оценка / балльная оценка | | |  | | | |
|  | | | | |
| дата проверки | | | | |

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

**СОДЕРЖАНИЕ**

[**1 Задание 1** 3](#_Toc180651529)

[**2 Задание 2** 4](#_Toc180651530)

[**3 Задание 3** 4](#_Toc180651531)

[**4 Задание 4** 5](#_Toc180651532)

[**5 Задание 5** 6](#_Toc180651533)

[**5 Задание 6** 10](#_Toc180651534)

[**7 Задание 7** 11](#_Toc180651535)

[**8 Задание 8** 13](#_Toc180651536)

# **1 Задание 1**

* 1. Построить точки, расположенные в вершинах правильного n-угольника (координаты вершин рекомендуется высчитывать в цикле). Установить режим сглаживания для точек. Экспериментально определить максимальный размер точки, при котором возможно сглаживание.
  2. Текст программы

    def draw\_n\_gon\_points(self):

        glPointSize(10.0)

        glBegin(GL\_POINTS)

        for i in range(self.n):

            angle = 2 \* math.pi \* i / self.n

            x = math.cos(angle)

            y = math.sin(angle)

            glVertex2f(x, y)

        glEnd()

1.3 Результат работы программы

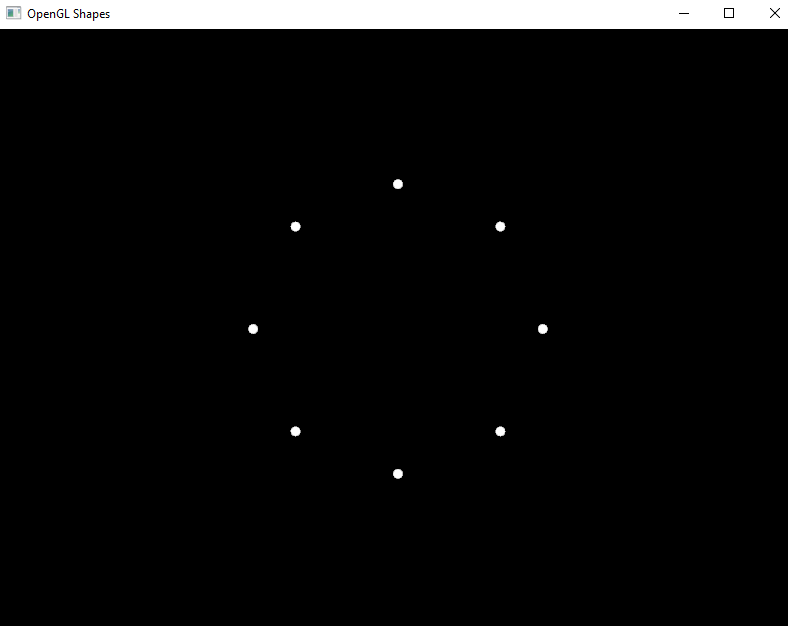


Рисунок 1

# **2 Задание 2**

2.1 Используя примитив для вывода линий, нарисовать правильный n-угольник. Изменить толщину линий.

* 1. Текст программы

    def draw\_n\_gon\_lines(self):

        glLineWidth(2.0)

        glBegin(GL\_LINE\_LOOP)

        for i in range(self.n):

            angle = 2 \* math.pi \* i / self.n

            x = math.cos(angle)

            y = math.sin(angle)

            glVertex2f(x, y)

        glEnd()

2.3 Результат работы программы

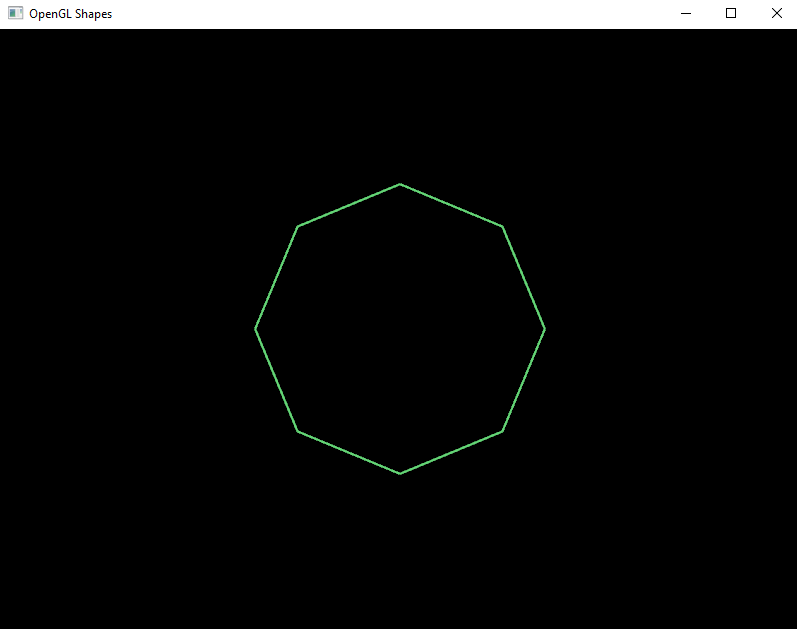


Рисунок 2

# **3 Задание 3**

3.1 Используя примитив для вывода ломаной линии, нарисовать первую фигуру, указанную в варианте.

3.2 Текст программы

    def draw\_figure1(self):

        glBegin(GL\_LINE\_STRIP)

        vertices = [

            (-1.8, -0.5), (-1.2, 1), (-0.9, -1.8),

            (-0.9, 0.4), (0, 0.4), (0, 1), (2, -1)

        ]

        for vertex in vertices:

            glVertex2f(\*vertex)

        glEnd()

3.3 Результат работы программы



Рисунок 3

# **4 Задание 4**

4.1 Используя примитив для вывода замкнутой ломаной, нарисовать вторую фигуру, указанную в варианте.

4.2 Текст программы

    def draw\_figure2(self):

        glBegin(GL\_LINE\_LOOP)

        vertices = [

            (-2, -1), (-1.2, 1.5), (1, 2), (0.5, 1), (2, 1.3), (2, 0.3), (0.5, 0.3), (0.5, -1.4), (-0.7, 0.3)

        ]

        for vertex in vertices:

            glVertex2f(\*vertex)

        glEnd()

4.3 Результат работы программы

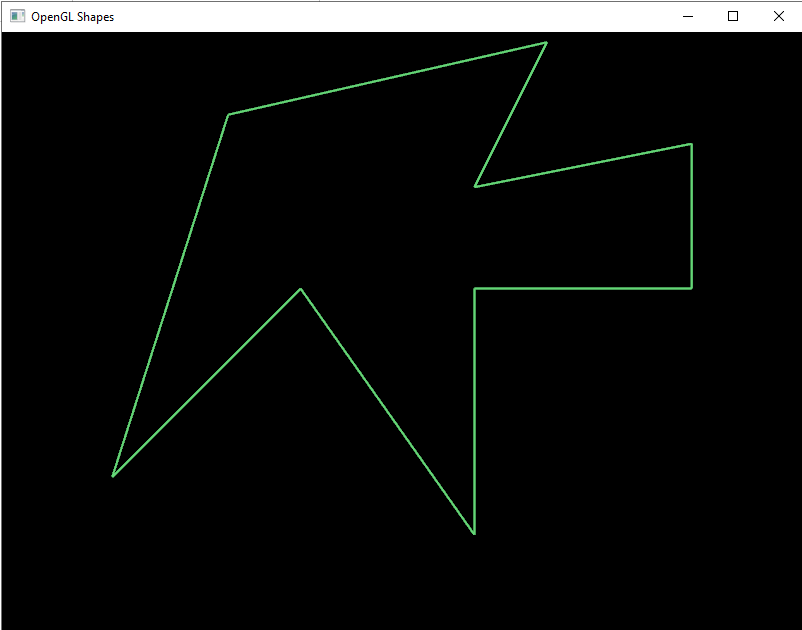


Рисунок 4

# **5 Задание 5**

5.1 Построить вторую фигуру, указанную в варианте, разбив её на треугольники (каждый треугольник окрашен случайным цветом). Выполните три варианта построений с использованием примитивов:

а) треугольник;

б) лента треугольников;

в) веер треугольников.

5.2 Текст программы

    def draw\_figure2\_triangles(self, primitive\_type):

        # Set shading mode

        if self.shading\_mode == 'flat':

            glShadeModel(GL\_FLAT)

        else:

            glShadeModel(GL\_SMOOTH)

        if primitive\_type == 'triangles':

            print("Drawing individual triangles")

            vertices = [

                (-2, -1), (-1.2, 1.5), (1, 2),

                (0.16, 1.16), (-0.7, 0.3), (2, 0.3),

                (0.5, 1), (2, 1.3), (2, 0.3),

                (0.16, 1.16), (1, 2), (0.5, 1),

                (0.5, 0.3), (0.5, -1.4), (-0.7, 0.3)

            ]

            glBegin(GL\_TRIANGLES)

            for i in range(0, len(vertices), 3):

                color = (random.random(), random.random(), random.random())

                for j in range(3):

                    glColor3f(\*color)

                    glVertex2f(\*vertices[i+j])

            glEnd()

        elif primitive\_type == 'triangle\_strip':

            print("Drawing triangle strip")

            strip1 = [

                (-2, -1),

                (-1.2, 1.5),

                (-0.7, 0.3),

                (1, 2),

                (-1, 0.6),

                (0.5, 1),

            ]

            strip2 = [

                (2, 1.3),

                (2, 0.3),

                (-0.5, 0.8),

                (-1, 0.3),

            ]

            strip3 = [

               (-0.7, 0.3),

               (0.5, -1.4),

               (0.5, 0.3),

            ]

            def draw\_strip(vertices):

                glBegin(GL\_TRIANGLE\_STRIP)

                for vertex in vertices:

                    if self.shading\_mode == 'smooth':

                        glColor3f(random.random(), random.random(), random.random())

                    else:

                        # For flat shading, color entire primitive

                        if vertices.index(vertex) % 2 == 0:

                            glColor3f(random.random(), random.random(), random.random())

                    glVertex2f(\*vertex)

                glEnd()

            draw\_strip(strip1)

            draw\_strip(strip2)

            draw\_strip(strip3)

        elif primitive\_type == 'triangle\_fan':

            print("Drawing triangle fan")

            fans = [

                {

                    'center': (-0.7, 0.3),

                    'vertices': [

                        (-2, -1),

                        (-1.2, 1.5)

                    ]

                },

                {

                    'center': (-1.2, 1.5),

                    'vertices': [

                        (-0.7, 0.3),

                        (1, 2)

                    ]

                },

                {   # Fan 3 (Triangle C)

                    'center': (-0.7, 0.3),

                    'vertices': [

                        (1, 2),

                        (-1, 0.6)

                    ]

                },

                {

                    'center': (-1, 0.6),

                    'vertices': [

                        (1, 2),

                        (0.5, 1)

                    ]

                },

                {

                    'center': (-0.5, 0.8),

                    'vertices': [

                        (2, 1.3),

                        (2, 0.3),

                        (-1, 0.3)

                    ]

                },

                {

                    'center': (-0.7, 0.3),

                    'vertices': [

                        (0.5, -1.4),

                        (0.5, 0.3)

                    ]

                }

            ]

            def draw\_fan(center, vertices):

                glBegin(GL\_TRIANGLE\_FAN)

                # Draw center point

                if not self.shading\_mode == 'smooth':

                    glColor3f(random.random(), random.random(), random.random())

                glVertex2f(\*center)

                for vertex in vertices:

                    if self.shading\_mode == 'smooth':

                        glColor3f(random.random(), random.random(), random.random())

                    glVertex2f(\*vertex)

                glEnd()

            for fan in fans:

                draw\_fan(fan['center'], fan['vertices'])

5.3 Результат работы программы

Примитив треугольников на рисунке 5, лента – рисунок 6, веер – рисунок 7

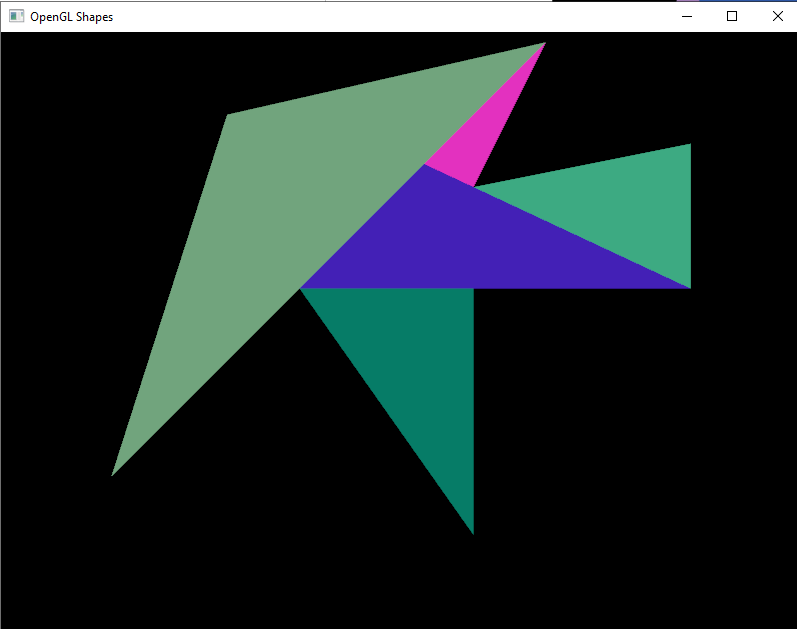


Рисунок 5

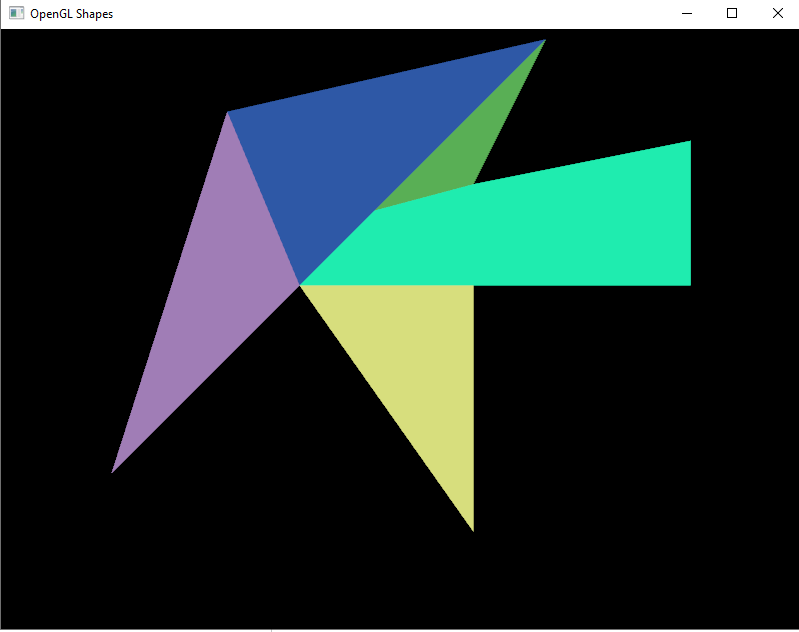


Рисунок 6

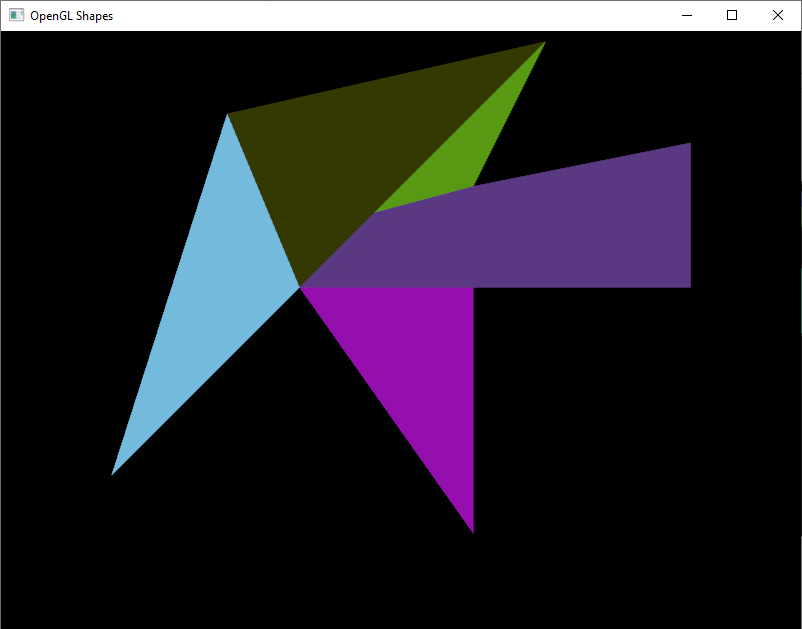


Рисунок 7

# **5 Задание 6**

6.1 Используя примитив для вывода веера треугольников, построить правильный n-угольник.

6.2 Текст программы

    def draw\_n\_gon\_fan(self):

        glBegin(GL\_TRIANGLE\_FAN)

        glVertex2f(0, 0)  # Center point

        for i in range(self.n + 1):

            angle = 2 \* math.pi \* i / self.n

            x = math.cos(angle)

            y = math.sin(angle)

            glColor3f(random.random(), random.random(), random.random())

            glVertex2f(x, y)

        glEnd()

6.3 Результат работы программы

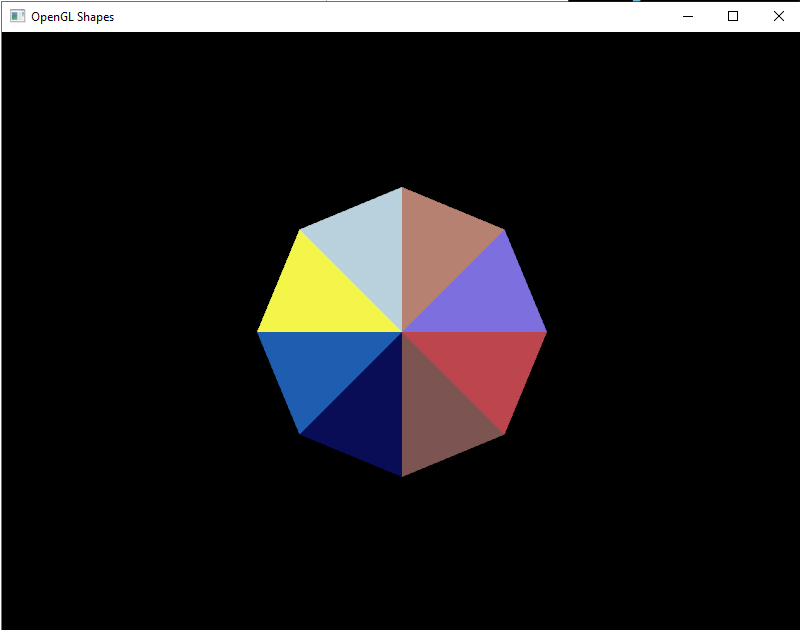


Рисунок 8

# **7 Задание 7**

7.1 Построить третью фигуру, указанную в варианте, представив её в виде совокупности отдельных треугольников, назначив 40 каждому треугольнику свой цвет. Посмотреть результат работы программы для различных способов тонирования.

7.2 Текст программы

    def draw\_figure3(self):

        triangles = [

        [(-1.5, -1.5), (-1.0, -0.5), (-0.5, -1.6)],

        [(-0.71, -1.15), (-0.5, -1.6), (0.5, -1.7)],

        [(0.5, -1.7), (-0.71, -1.15), (0.7, -1.15)],

        [(0.0, -1.15), (0.25, 0.52), (0.7, -1.15)],

        [(0.7, -1.15), (0.25, 0.52), (1.2, 0.42)],

        [(0.0, -1.15), (0.25, 0.52), (-0.5, 0.6)],

        [(-1.5, -0.5), (-0.5, 0.6), (-0.5, -0.5)],

        [(-1.5, -0.5), (-1.5, 0.7), (-0.5, 0.6)],

        [(-1.5, 0.7), (1.2, 0.42), (-1.5, 0.9)],

        [(-1.5, 0.9), (1.2, 0.42), (1.25, 0.6)]

    ]

        if self.face\_mode == 'normal':

            # Normal mode: all faces filled

            glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL)

            glBegin(GL\_TRIANGLES)

            for triangle in triangles:

                glColor3f(random.random(), random.random(), random.random())

                for vertex in triangle:

                    glVertex2f(\*vertex)

            glEnd()

            glDisable(GL\_CULL\_FACE)

        elif self.face\_mode == 'vertices\_front':

            # Mode A: front faces as vertices only

            glEnable(GL\_CULL\_FACE)

            # Draw front faces (vertices only)

            glCullFace(GL\_BACK)

            glPolygonMode(GL\_FRONT, GL\_POINT)

            glPointSize(5.0)  # Make vertices more visible

            glBegin(GL\_TRIANGLES)

            for triangle in triangles:

                glColor3f(random.random(), random.random(), random.random())

                for vertex in triangle:

                    glVertex2f(\*vertex)

            glEnd()

            glDisable(GL\_CULL\_FACE)

        elif self.face\_mode == 'filled\_front\_wire\_back':

            # Mode B: front faces filled, back faces as wireframe

            # First draw back faces as wireframe

            glEnable(GL\_CULL\_FACE)

            glCullFace(GL\_FRONT)

            glPolygonMode(GL\_BACK, GL\_LINE)

            glLineWidth(1.0)

            glBegin(GL\_TRIANGLES)

            for triangle in triangles:

                glColor3f(0.5, 0.5, 0.5)  # Grey color for back faces

                for vertex in triangle:

                    glVertex2f(\*vertex)

            glEnd()

            # Then draw front faces filled

            glCullFace(GL\_BACK)

            glPolygonMode(GL\_FRONT, GL\_FILL)

            glBegin(GL\_TRIANGLES)

            for triangle in triangles:

                glColor3f(random.random(), random.random(), random.random())

                for vertex in triangle:

                    glVertex2f(\*vertex)

            glEnd()

            glDisable(GL\_CULL\_FACE)

        else:

            # Mode C: all faces as wireframe

            glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE)

            glLineWidth(1.0)

            glBegin(GL\_TRIANGLES)

            for triangle in triangles:

                glColor3f(random.random(), random.random(), random.random())

                for vertex in triangle:

                    glVertex2f(\*vertex)

            glEnd()

7.3 Результат работы программы

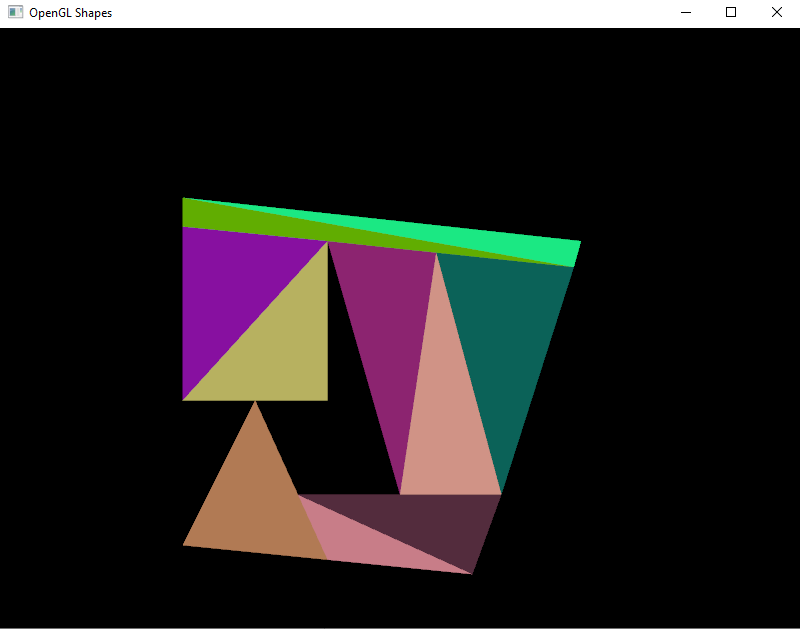


Рисунок 9

# **8 Задание 8**

8.1 Изменить программу предыдущей задачи таким образом, чтобы:

а) лицевые грани изображались только вершинами;

б) лицевые грани изображались закрашенными, а обратные – линиями; в) лицевые и обратные грани изображались линиями (каркасное изображение).

8.2 Текст программы

    def draw\_figure3(self):

        triangles = [

        [(-1.5, -1.5), (-1.0, -0.5), (-0.5, -1.6)],

        [(-0.71, -1.15), (-0.5, -1.6), (0.5, -1.7)],

        [(0.5, -1.7), (-0.71, -1.15), (0.7, -1.15)],

        [(0.0, -1.15), (0.25, 0.52), (0.7, -1.15)],

        [(0.7, -1.15), (0.25, 0.52), (1.2, 0.42)],

        [(0.0, -1.15), (0.25, 0.52), (-0.5, 0.6)],

        [(-1.5, -0.5), (-0.5, 0.6), (-0.5, -0.5)],

        [(-1.5, -0.5), (-1.5, 0.7), (-0.5, 0.6)],

        [(-1.5, 0.7), (1.2, 0.42), (-1.5, 0.9)],

        [(-1.5, 0.9), (1.2, 0.42), (1.25, 0.6)]

    ]

        if self.face\_mode == 'normal':

            # Normal mode: all faces filled

            glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL)

            glBegin(GL\_TRIANGLES)

            for triangle in triangles:

                glColor3f(random.random(), random.random(), random.random())

                for vertex in triangle:

                    glVertex2f(\*vertex)

            glEnd()

            glDisable(GL\_CULL\_FACE)

        elif self.face\_mode == 'vertices\_front':

            # Mode A: front faces as vertices only

            glEnable(GL\_CULL\_FACE)

            # Draw front faces (vertices only)

            glCullFace(GL\_BACK)

            glPolygonMode(GL\_FRONT, GL\_POINT)

            glPointSize(5.0)  # Make vertices more visible

            glBegin(GL\_TRIANGLES)

            for triangle in triangles:

                glColor3f(random.random(), random.random(), random.random())

                for vertex in triangle:

                    glVertex2f(\*vertex)

            glEnd()

            glDisable(GL\_CULL\_FACE)

        elif self.face\_mode == 'filled\_front\_wire\_back':

            # Mode B: front faces filled, back faces as wireframe

            # First draw back faces as wireframe

            glEnable(GL\_CULL\_FACE)

            glCullFace(GL\_FRONT)

            glPolygonMode(GL\_BACK, GL\_LINE)

            glLineWidth(1.0)

            glBegin(GL\_TRIANGLES)

            for triangle in triangles:

                glColor3f(0.5, 0.5, 0.5)  # Grey color for back faces

                for vertex in triangle:

                    glVertex2f(\*vertex)

            glEnd()

            # Then draw front faces filled

            glCullFace(GL\_BACK)

            glPolygonMode(GL\_FRONT, GL\_FILL)

            glBegin(GL\_TRIANGLES)

            for triangle in triangles:

                glColor3f(random.random(), random.random(), random.random())

                for vertex in triangle:

                    glVertex2f(\*vertex)

            glEnd()

            glDisable(GL\_CULL\_FACE)

        else:

            # Mode C: all faces as wireframe

            glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE)

            glLineWidth(1.0)

            glBegin(GL\_TRIANGLES)

            for triangle in triangles:

                glColor3f(random.random(), random.random(), random.random())

                for vertex in triangle:

                    glVertex2f(\*vertex)

            glEnd()

8.3 Результат работы программы

На рисунке 10 – пункт а, на рисунке 11 – пункт б, на рисунке 12 – пункт в.

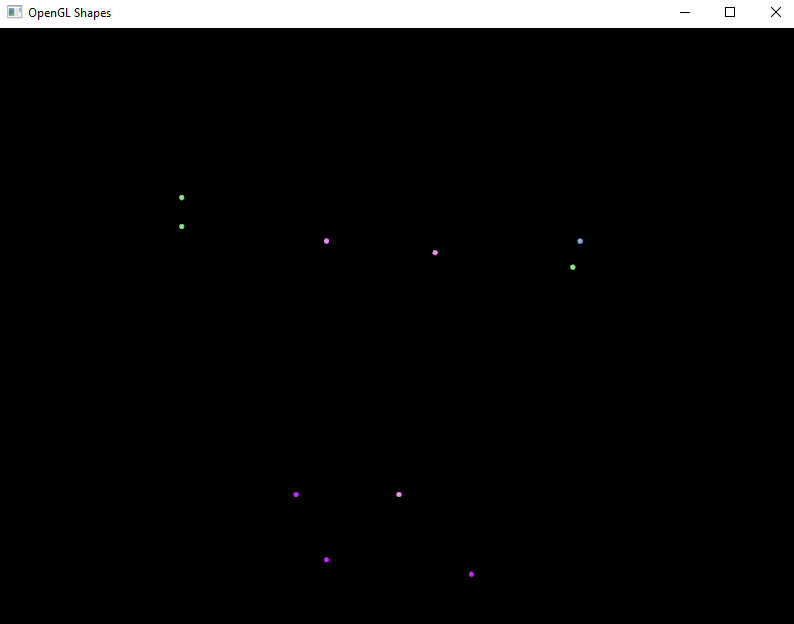
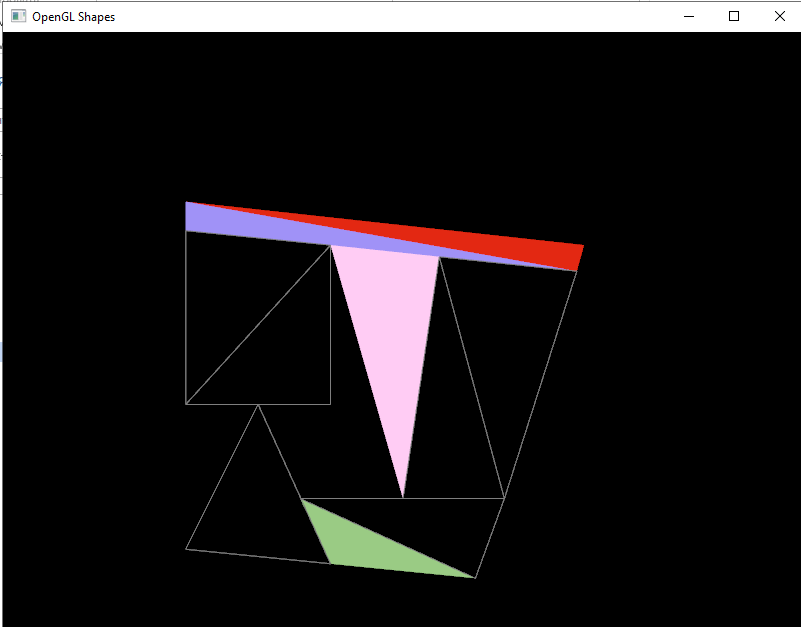


Рисунок 10

  
Рисунок 11

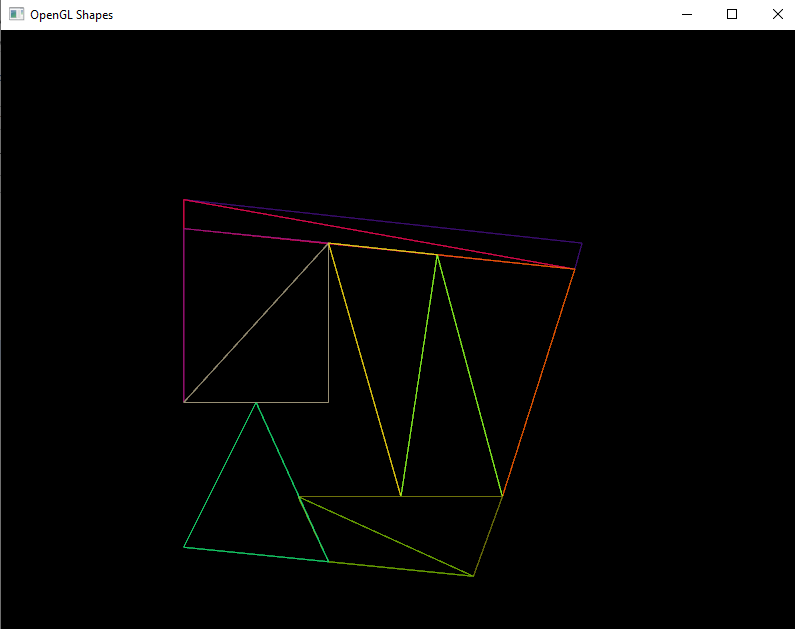


Рисунок 12