| *voenmeh* | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| --- | --- |
| БГТУ.СМК-Ф-4.2-К5-01 |

| Факультет |  | О |  | Естественнонаучный |
| --- | --- | --- | --- | --- |
|  |  | шифр |  | наименование |
| Кафедра |  | О7 |  | Информационные системы и программная инженерия |
|  |  | шифр |  | наименование |
| Дисциплина |  | Компьютерная геометрия и графика | | |

Отчёт по практической работе № 2

| Примитивы OpenGL, основные приемы |
| --- |
| построения двумерных объектов |
|  |

| Выполнил студент группы | | | |  | И508Б |
| --- | --- | --- | --- | --- | --- |
| Кабиров К.Р. | | | | | |
| Фамилия И.О. | | | | | |
| **РУКОВОДИТЕЛЬ** | | | | | |
|  | |  |  | | |
| Фамилия И.О. Подпись | | | | | |
| Оценка |  | | | |  |
| «\_\_\_\_\_» |  | | | | 2022г. |

САНКТ-ПЕТЕРБУРГ

2022 г

**Вариант 3**

**Задания к работе**

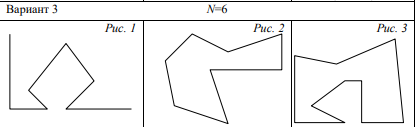


Рисунок 1 – Задания по варианту

1. Построить точки, расположенные в вершинах правильного *N*-угольника. Установить режим сглаживания для точек. Экспериментально определить максимальный размер точки, при котором возможно сглаживание.

Содержимое функции DrawGLScene:

glClear(**GL\_COLOR\_BUFFER\_BIT**); *// очистка экрана*

*// glEnable(GL\_POINT\_SMOOTH); // сглаживание*

glPointSize(20); *//размер точки*

glColor3f(0.2, 0.78, 0.0);

double x, y;

double radius = 0.7;

glBegin(**GL\_POINTS**);

for (int i = 0; i < N; i++)

{

x = radius \* cos(2.0\***M\_PI**\*i/N);

y = radius \* sin(2.0\***M\_PI**\*i/N);

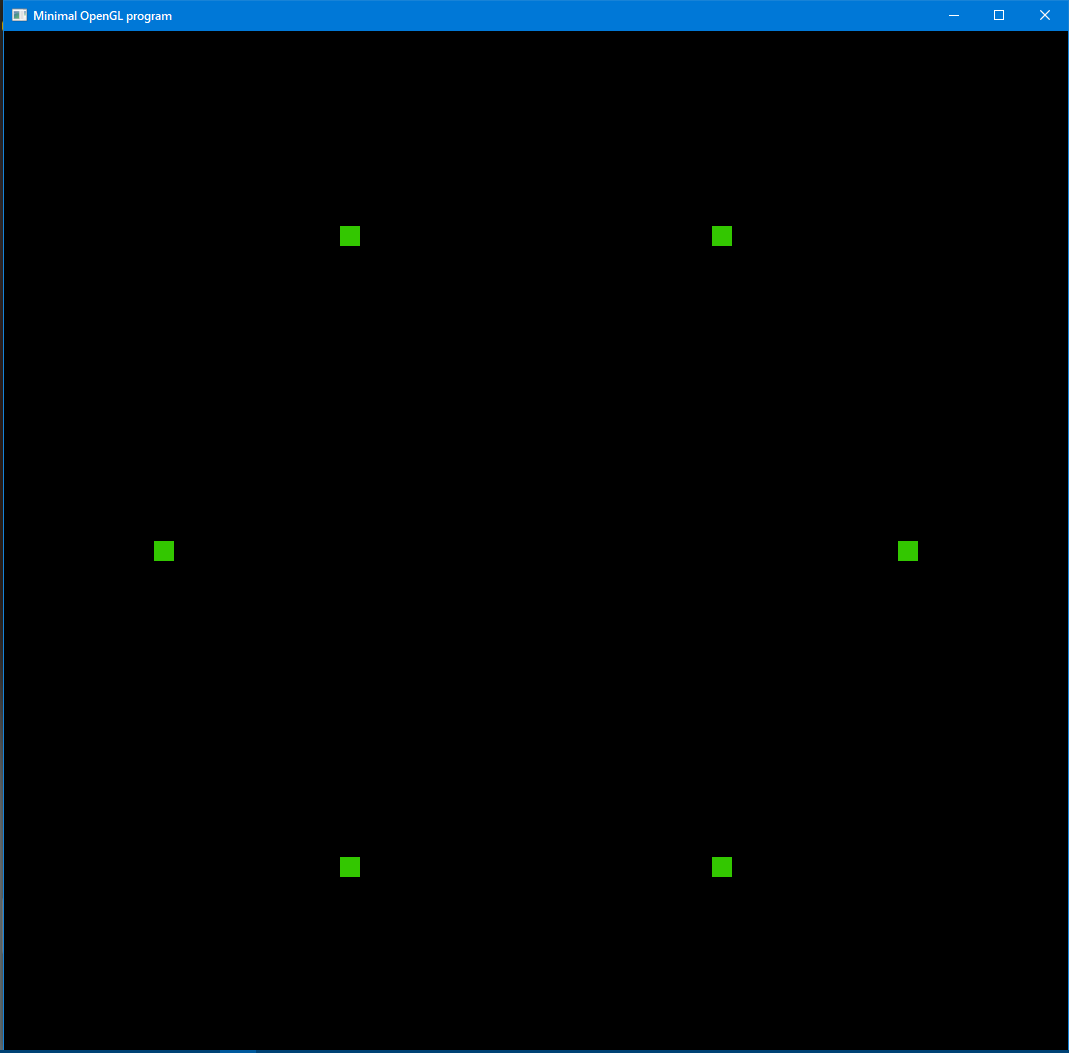
glVertex2d(x, y);

}

*// glDisable(GL\_POINT\_SMOOTH);*

glEnd();

Установленный максимальный размер точки: 20.

  
Рисунок 2 – Вершины правильного шестиугольника без сглаживания

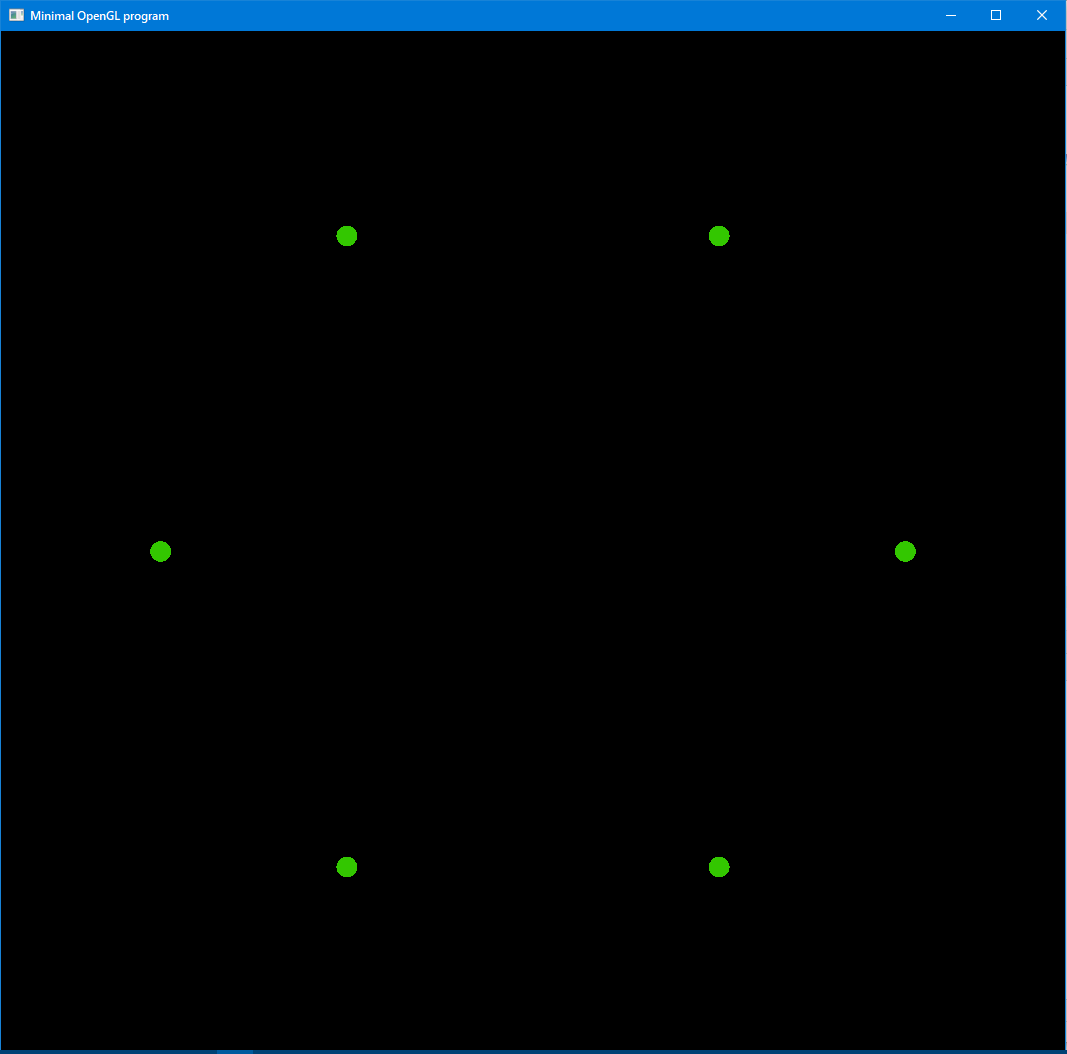


Рисунок 3 – Вершины правильного шестиугольника со сглаживанием

1. Используя примитив для вывода линий, нарисовать правильный *N* -угольник. Изменить тип и ширину линий.

Содержимое функции *DrawGLScene*:

glClear(**GL\_COLOR\_BUFFER\_BIT**); *// очистка экрана*

double x[N], y[N];

double radius = 0.7;

x[0] = radius;

y[0] = 0;

glColor3f(0.2, 0.78, 0.0);

glLineWidth(2); *// толщина линии 2 пикселя*

glLineStipple(2, 3855); *// 0000111100001111*

glEnable(**GL\_LINE\_STIPPLE**);

glBegin(**GL\_LINE\_LOOP**);

for (int i = 1; i < N; i++)

{

x[i] = radius \* cos(2.0\***M\_PI**\*i/N);

y[i] = radius \* sin(2.0\***M\_PI**\*i/N);

glVertex2d(x[i-1], y[i-1]);

glVertex2d(x[i], y[i]);

}

glEnd();

glDisable(**GL\_LINE\_STIPPLE**);

Ширина линий установлена на 2 , тип линий - пунктирная.

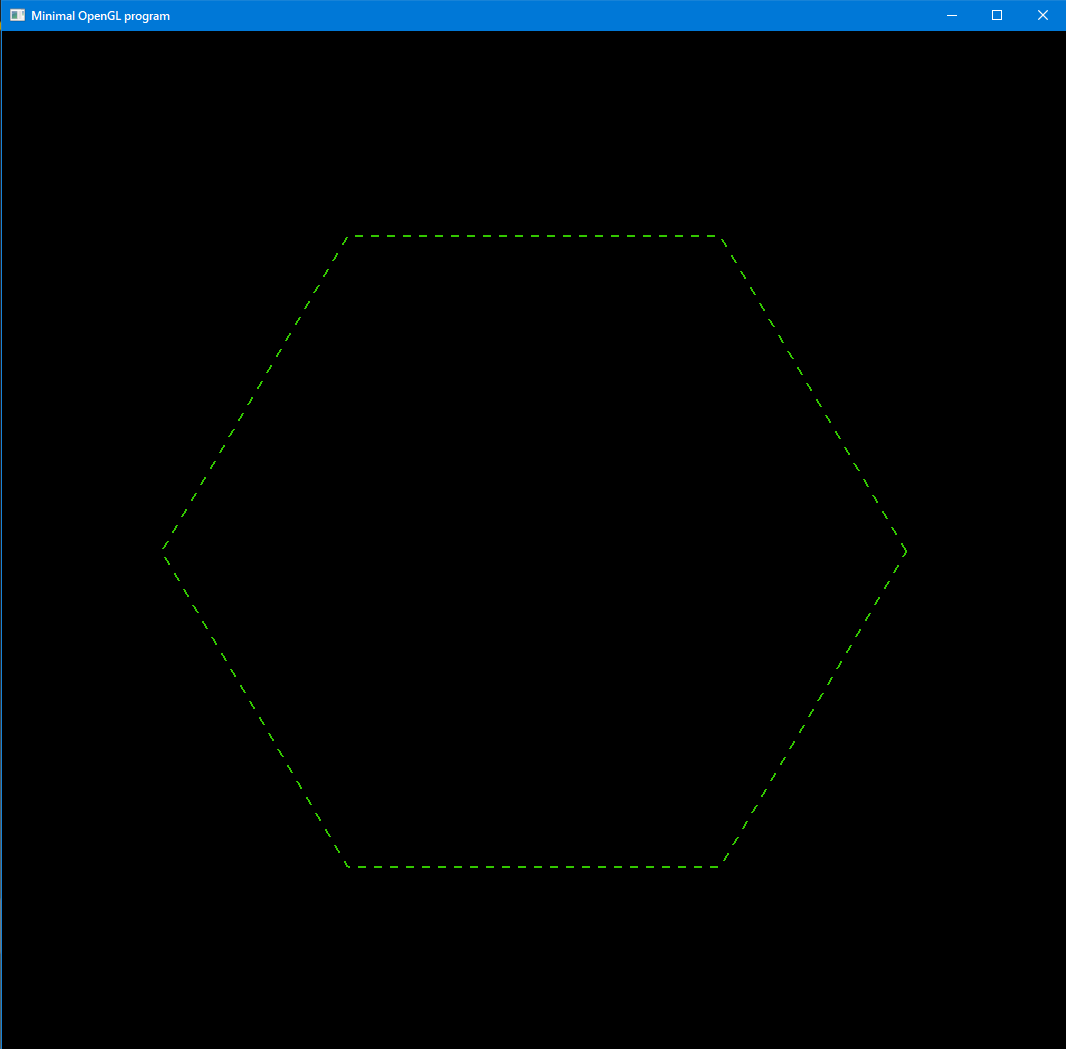


Рисунок 4 – Правильный семиугольник пунктирными линиями

1. Используя примитив для вывода ломаной линии, нарисовать фигуру, изображенную на рис. 1.

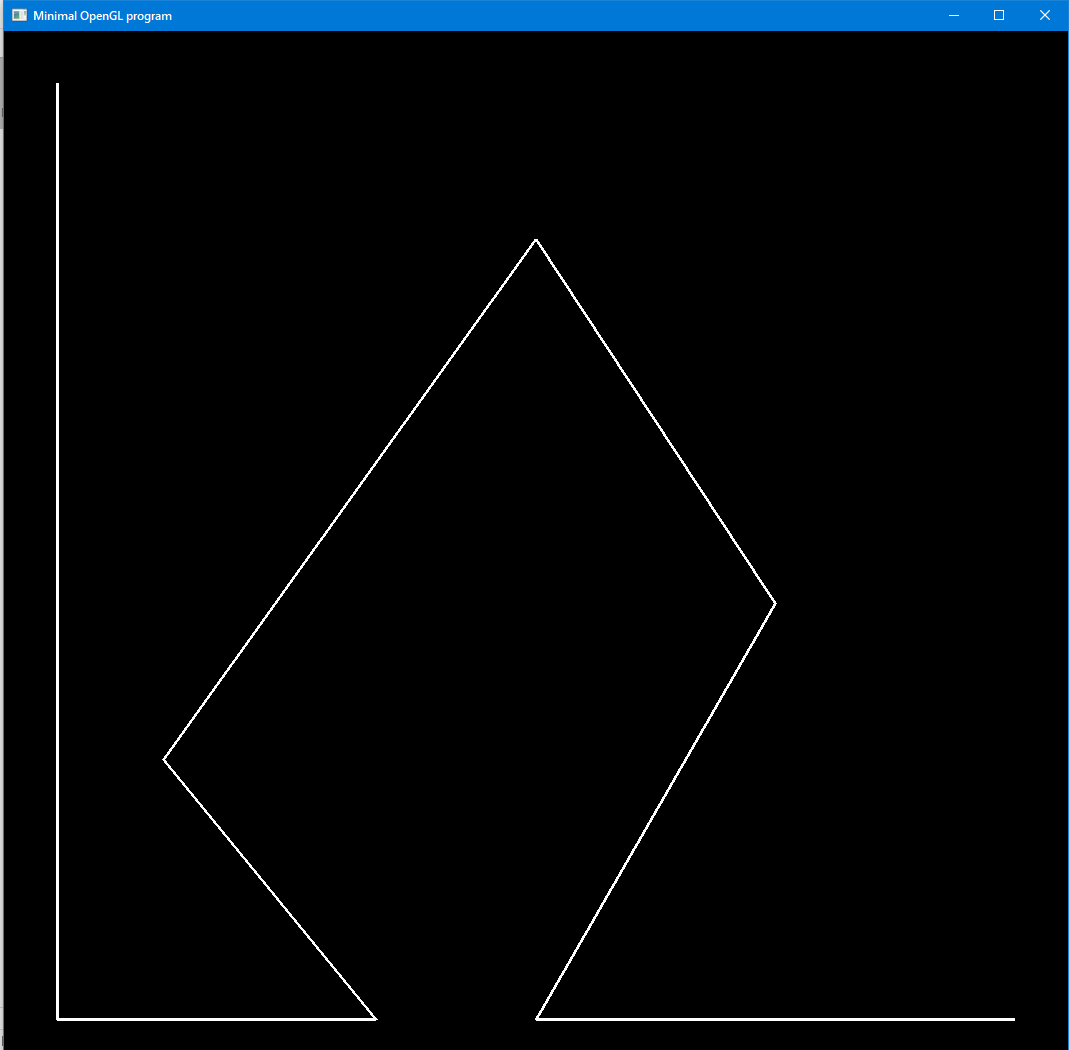


Рисунок 5 – Первая фигура

Содержимое функции *DrawGLScene*:

glClear(**GL\_COLOR\_BUFFER\_BIT**); *// очистка экрана*

glColor3f(1.0, 1.0, 1.0);

glLineWidth(3); *// толщина линии 3 пикселя*

glLineStipple(1, 0xF0F0); *// тип линии - пунктирная*

glBegin(**GL\_LINE\_STRIP**);

glVertex2d(-0.9, 0.9);

glVertex2d(-0.9, -0.9);

glVertex2d(-0.3, -0.9);

glVertex2d(-0.7, -0.4);

glVertex2d(0.0, 0.6);

glVertex2d(0.45, -0.1);

glVertex2d(0.0, -0.9);

glVertex2d(0.9, -0.9);

glEnd();

1. Используя примитив для вывода замкнутой ломаной, нарисовать фигуру, изображенную на рис. 2.

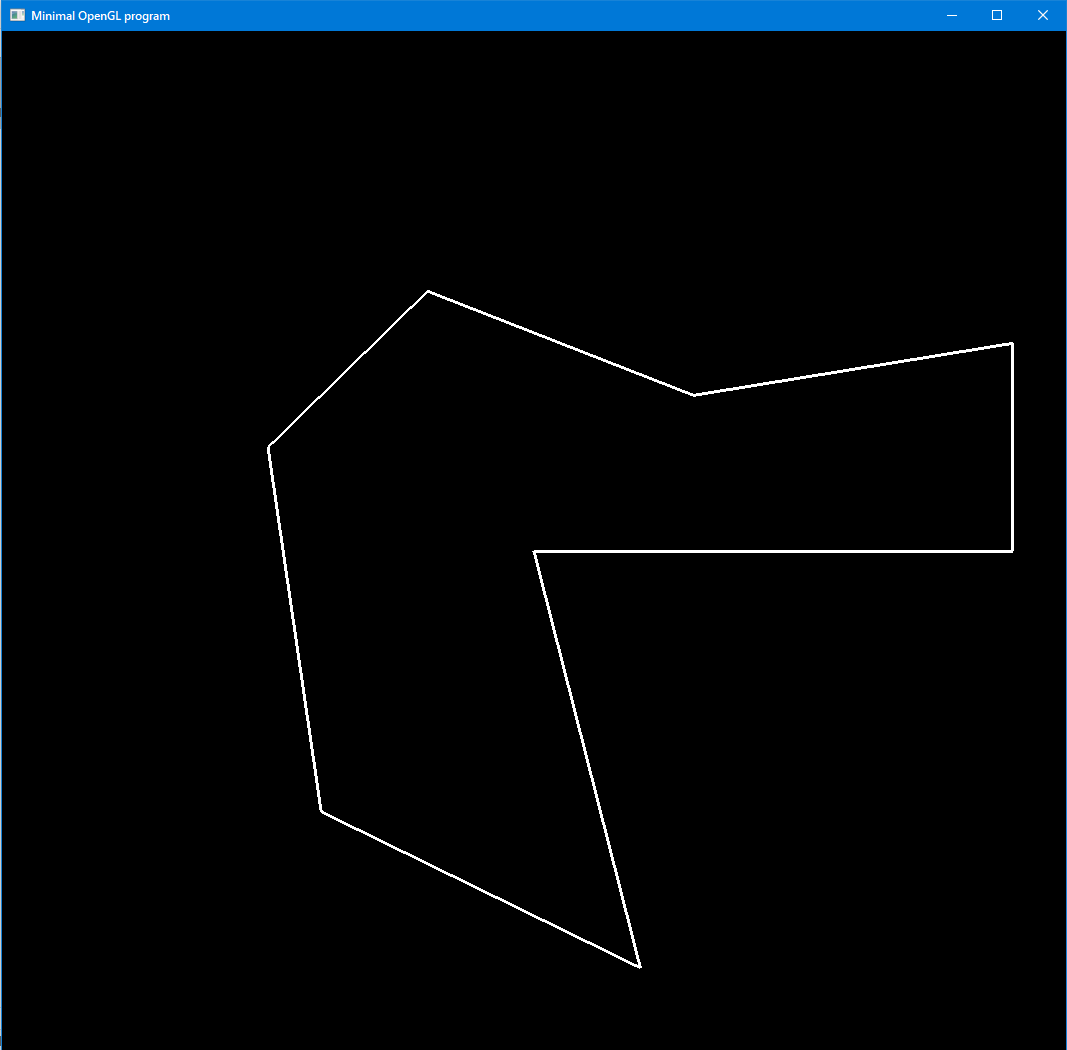


Рисунок 6 – Вторая фигура

Содержимое функции *DrawGLScene*:

glClear(**GL\_COLOR\_BUFFER\_BIT**); *// очистка экрана*

glColor3f(1.0, 1.0, 1.0);

glLineWidth(3); *// толщина линии 3 пикселя*

glLineStipple(1, 0xF0F0); *// тип линии - пунктирная*

glBegin(**GL\_LINE\_LOOP**);

glVertex2d(0.0, 0.0);

glVertex2d(0.9, 0.0);

glVertex2d(0.9, 0.4);

glVertex2d(0.3, 0.3);

glVertex2d(-0.2, 0.5);

glVertex2d(-0.5, 0.2);

glVertex2d(-0.4, -0.5);

glVertex2d(0.2, -0.8);

glEnd();

1. Построить фигуру, изображенную на рис. 2, разбив ее на треугольники (каждый треугольник окрашен случайным цветом). Выполнить три варианта построений с использованием примитивов:

А) треугольник;

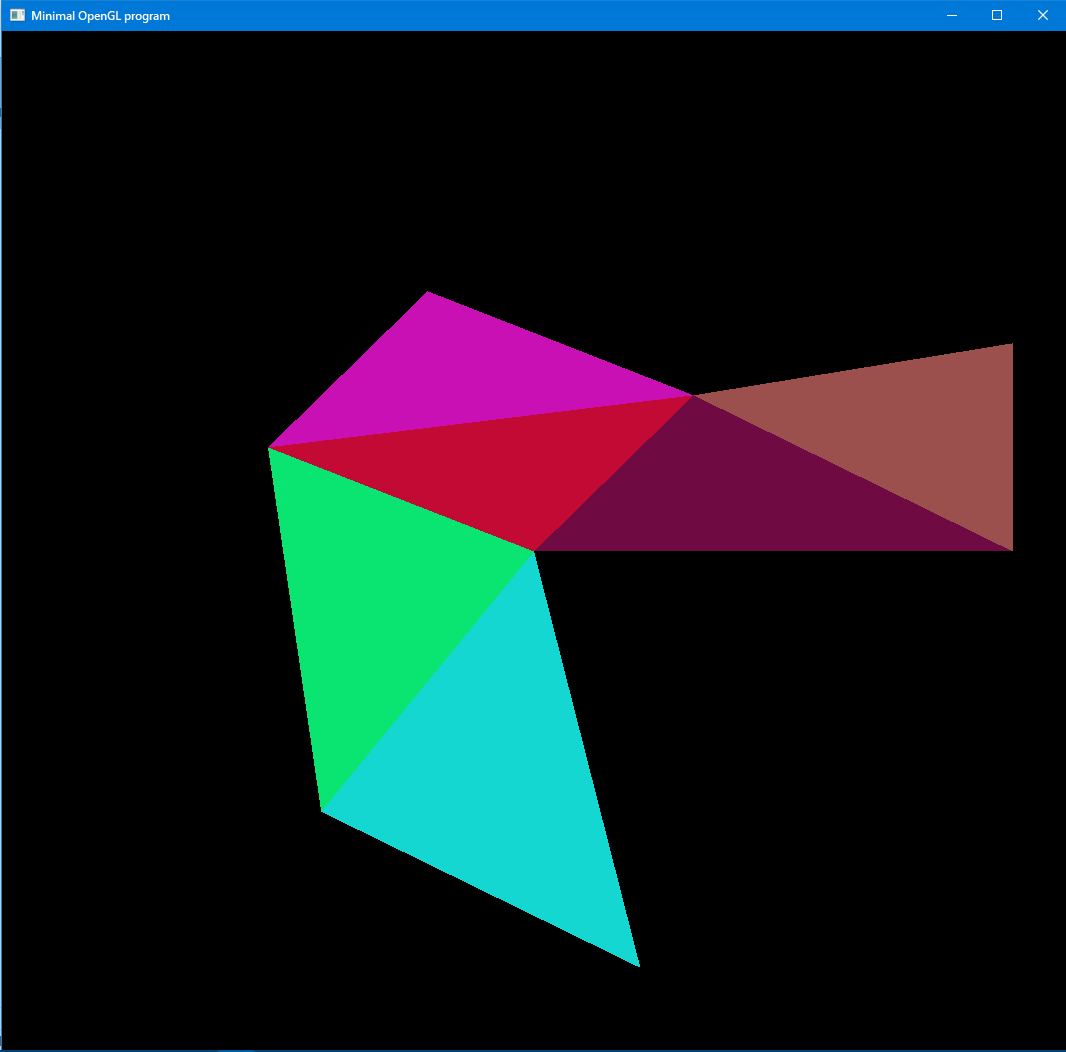


Рисунок 7 – Вторая фигура треугольниками

Содержимое функции *DrawGLScene*:

glClear(**GL\_COLOR\_BUFFER\_BIT**);

srand(time(**NULL**));

glBegin(**GL\_TRIANGLES**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(0.9, 0.0);

glVertex2d(0.3, 0.3);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.9, 0.0);

glVertex2d(0.9, 0.4);

glVertex2d(0.3, 0.3);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.3, 0.3);

glVertex2d(-0.2, 0.5);

glVertex2d(-0.5, 0.2);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(-0.5, 0.2);

glVertex2d(-0.4, -0.5);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(0.3, 0.3);

glVertex2d(-0.5, 0.2);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(-0.4, -0.5);

glVertex2d(0.2, -0.8);

glEnd();

Б) лента треугольников;

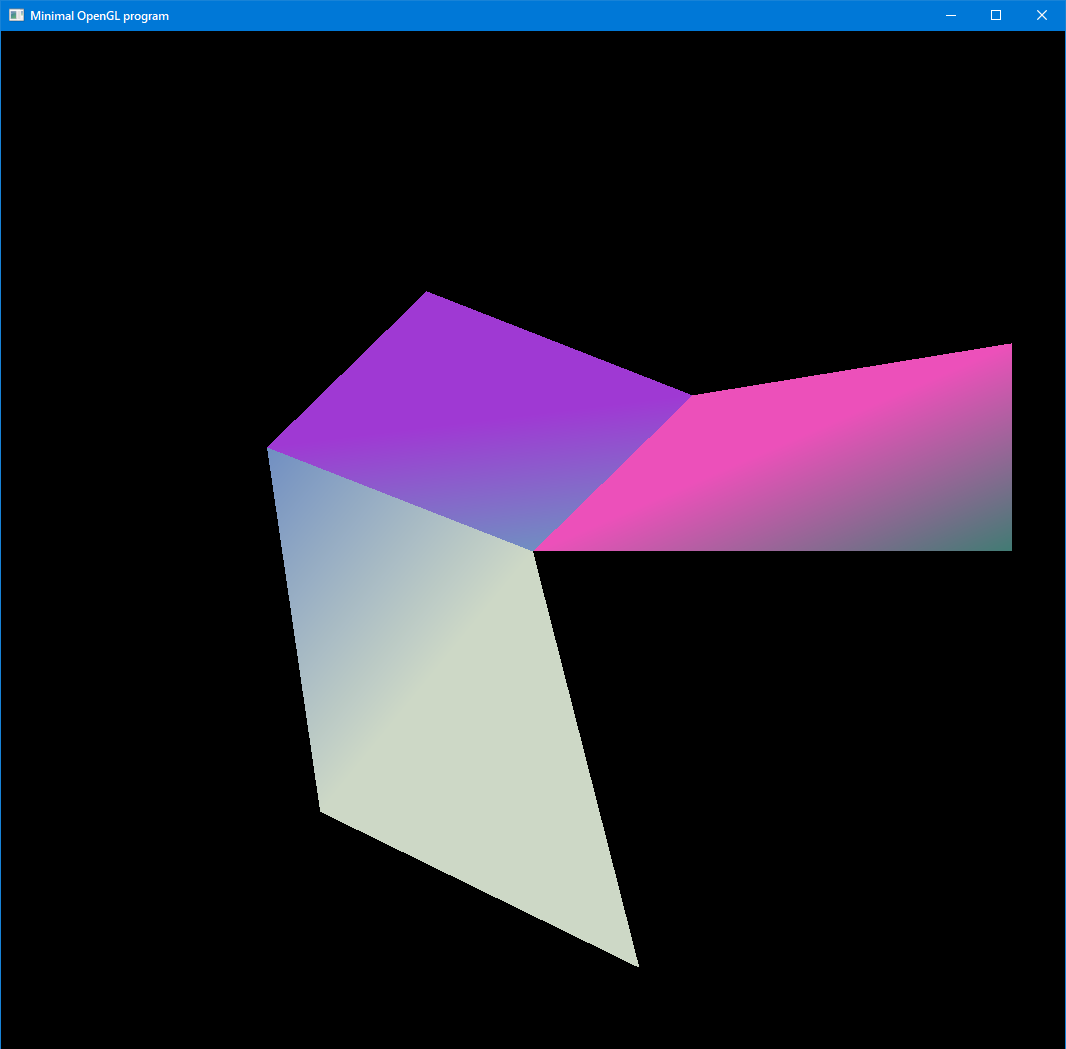


Рисунок 8 – Вторая фигура лентой треугольников

Содержимое функции *DrawGLScene*:

glClear(GL\_COLOR\_BUFFER\_BIT); // очистка экрана

srand(time(NULL));

glBegin(GL\_TRIANGLE\_STRIP);

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glVertex2d(0.0, 0.0);

glVertex2d(0.9, 0.4);

glVertex2d(0.9, 0.0);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.9, 0.4);

glVertex2d(0.0, 0.0);

glVertex2d(0.9, 0.0);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.9, 0.4);

glVertex2d(0.0, 0.0);

glVertex2d(0.3, 0.3);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(0.3, 0.3);

glVertex2d(-0.5, 0.2);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(-0.2, 0.5);

glVertex2d(0.3, 0.3);

glVertex2d(-0.5, 0.2);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(-0.4, -0.5);

glVertex2d(-0.5, 0.2);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(-0.4, -0.5);

glVertex2d(0.2, -0.8);

В) веер треугольников.

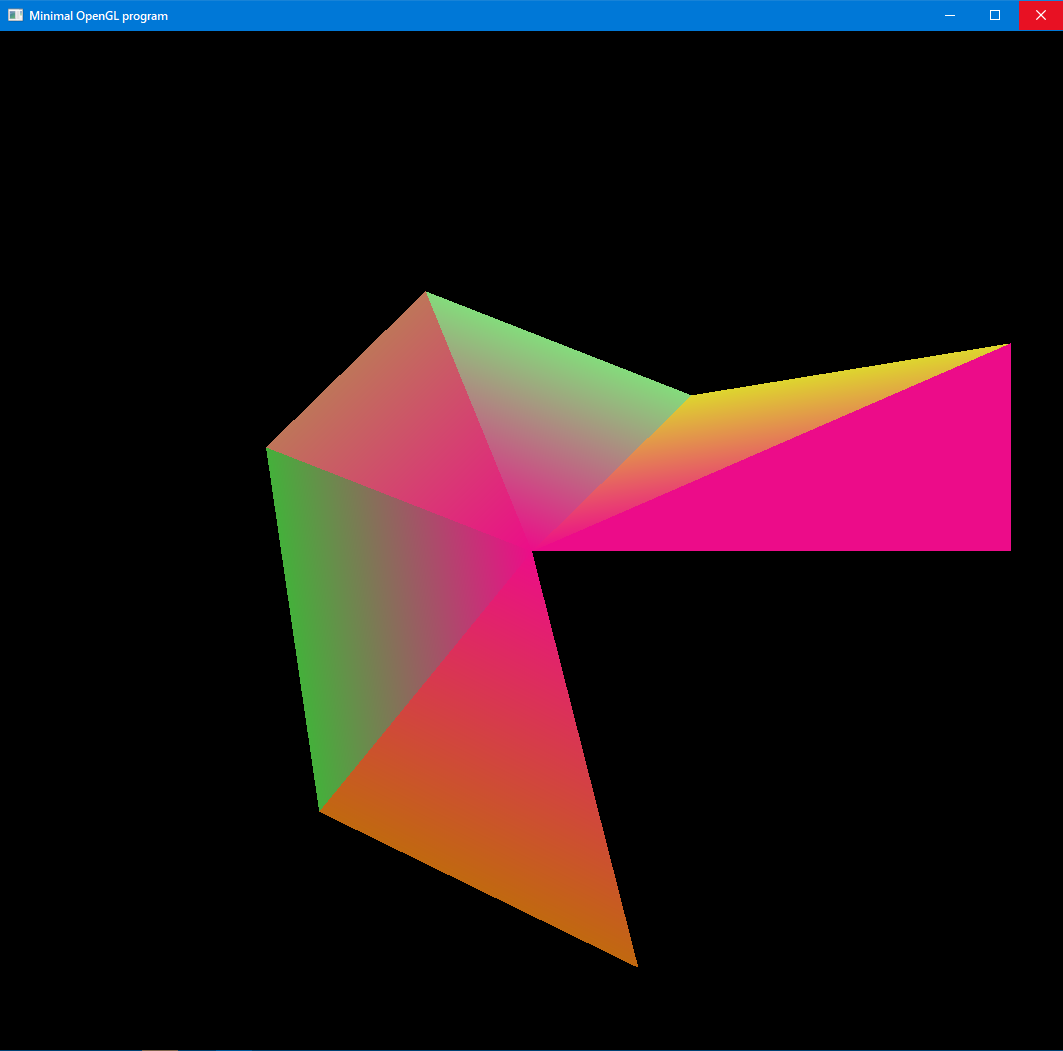


Рисунок 9 – Вторая фигура веером треугольников

Содержимое функции *DrawGLScene*:

glClear(**GL\_COLOR\_BUFFER\_BIT**); *// очистка экрана*

srand(time(**NULL**));

glBegin(**GL\_TRIANGLE\_FAN**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(0.9, 0.0);

glVertex2d(0.9, 0.4);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(0.3, 0.3);

glVertex2d(0.9, 0.4);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(0.3, 0.3);

glVertex2d(-0.2, 0.5);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(-0.2, 0.5);

glVertex2d(-0.5, 0.2);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(-0.5, 0.2);

glVertex2d(-0.4, -0.5);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.0, 0.0);

glVertex2d(-0.4, -0.5);

glVertex2d(0.2, -0.8);

glEnd();

Чем различаются результаты при изменении способа тонирования? – по умолчанию способ тонирования задан плавным – фигуры окрашиваются градиентно. Чтобы изменить его, необходимо перед командными скобками вызвать функцию *glShadeModel(GL\_FLAT)*

1. Используя примитив для вывода многоугольников, построить правильный *N* -угольник.

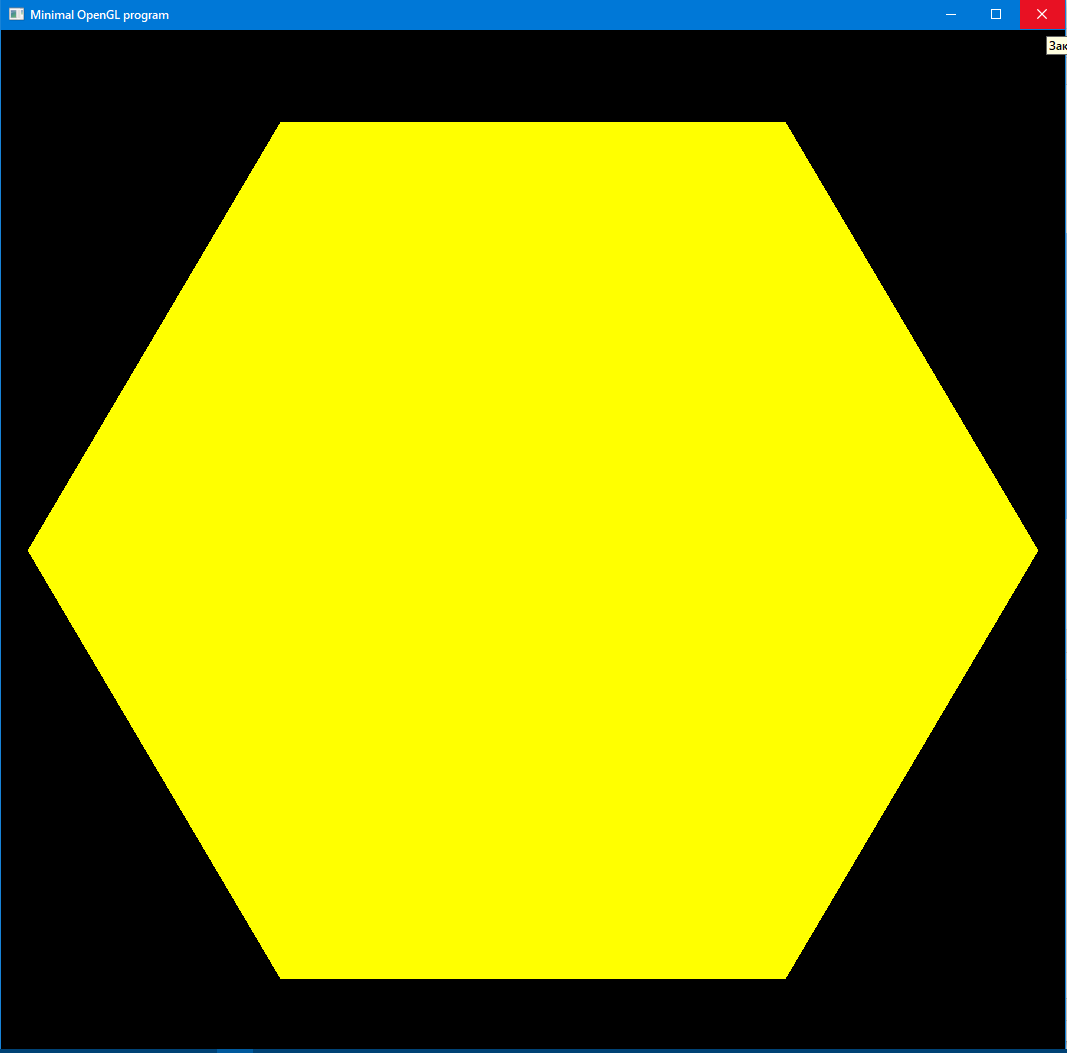


Рисунок 10 – Правильный шестиугольник

Содержимое функции *DrawGLScene*:

glClear(**GL\_COLOR\_BUFFER\_BIT**); *// очистка экрана*

double x[N], y[N];

double radius = 0.95;

glColor3f(1.0, 1.0, 0.0);

glBegin(**GL\_POLYGON**);

for (int i = 0; i < N; i++)

{

x[i] = radius \* cos(2.0\***M\_PI**\*i/N);

y[i] = radius \* sin(2.0\***M\_PI**\*i/N);

glVertex2d(x[i], y[i]);

}

glEnd();

1. Построить невыпуклый многоугольник, изображенный на рис.3, представив его в виде совокупности отдельных многоугольников, назначив каждому из них свой цвет. Посмотреть результат работы программы для различных способов тонирования.

Содержимое функции *DrawGLScene*:

glClear(GL\_COLOR\_BUFFER\_BIT); // очистка экрана

srand(time(NULL));

glBegin(GL\_QUADS);

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glVertex2d(0.20, 0.20); // 2

glVertex2d(0.20, -0.30); // 3

glVertex2d(0.70, -0.30); // 4

glVertex2d(0.60, 0.70); // 5

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glVertex2d(0.00, 0.20); // 1

glVertex2d(0.20, 0.20); // 2

glVertex2d(0.60, 0.70); // 5

glVertex2d(-0.10, 0.40); // 6

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glVertex2d(0.00, 0.20); // 1

glVertex2d(-0.10, 0.40); // 6

glVertex2d(-0.70, 0.60); // 7

glVertex2d(-0.70, -0.18); // 11

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glVertex2d(-0.70, -0.18); // 11

glVertex2d(-0.70, -0.30); // 8

glVertex2d(0.00, -0.30); // 9

glVertex2d(-0.46, -0.05); // 10

glEnd();

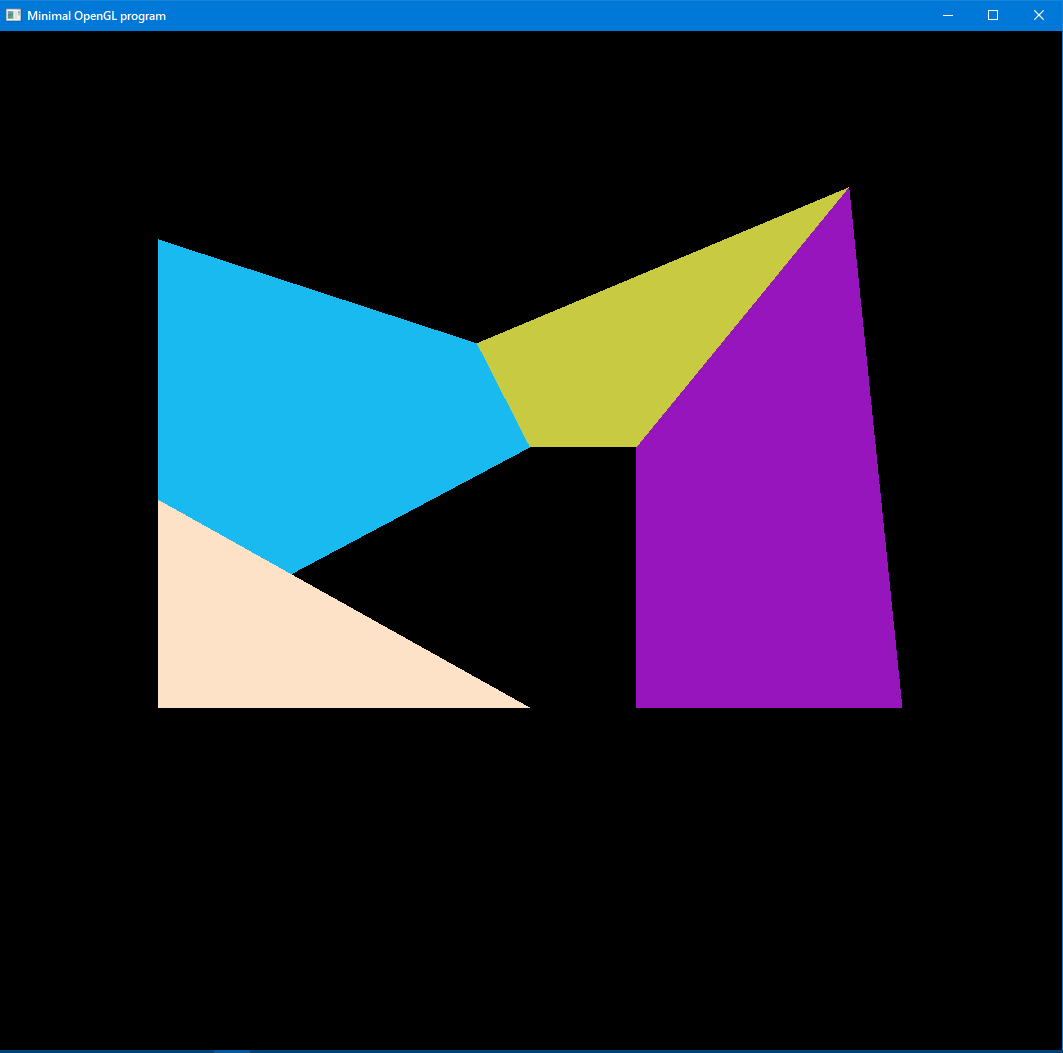


Рисунок 11 – Невыпуклый многоугольник

1. Изменить программу предыдущей задачи таким образом, чтобы:

А) лицевые грани изображались только вершинами;

Содержимое функции *DrawGLScene*:

glClear(**GL\_COLOR\_BUFFER\_BIT**); *// очистка экрана*

srand(time(**NULL**));

glPointSize(5);

glBegin(**GL\_QUADS**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.20, 0.20); *// 2*

glVertex2d(0.60, 0.70); *// 5*

glVertex2d(0.70, -0.30); *// 4*

glVertex2d(0.20, -0.30); *// 3*

glEnd();

glBegin(**GL\_POINTS**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.00, 0.20); *// 1*

glVertex2d(0.20, 0.20); *// 2*

glVertex2d(0.60, 0.70); *// 5*

glVertex2d(-0.10, 0.40); *// 6*

glEnd();

glBegin(**GL\_POINTS**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.00, 0.20); *// 1*

glVertex2d(-0.10, 0.40); *// 6*

glVertex2d(-0.70, 0.60); *// 7*

glVertex2d(-0.70, 0.1); *// 11*

glVertex2d(-0.46, -0.05); *// 10*

glEnd();

glBegin(**GL\_TRIANGLES**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(-0.70, 0.1); *// 11*

glVertex2d(0.00, -0.30); *// 9*

glVertex2d(-0.70, -0.30); *// 8*

glEnd();

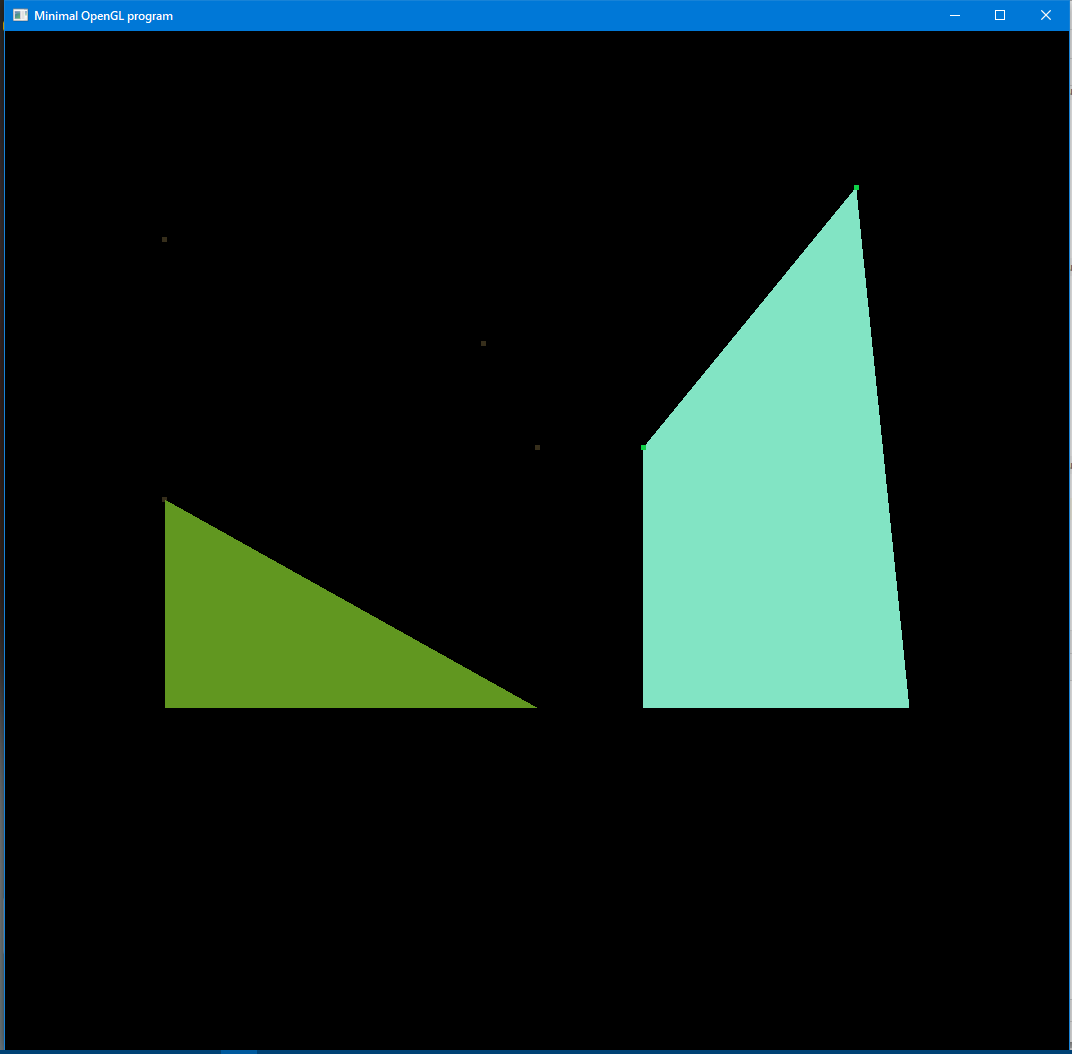


Рисунок 12 – Лицевые грани изображаются только вершинами

Б) лицевые грани изображались закрашенными, а обратные – линиями;

Содержимое функции *DrawGLScene*:

glClear(**GL\_COLOR\_BUFFER\_BIT**); *// очистка экрана*

srand(time(**NULL**));

glPointSize(5);

glBegin(**GL\_POLYGON**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.20, 0.20); *// 2*

glVertex2d(0.60, 0.70); *// 5*

glVertex2d(0.70, -0.30); *// 4*

glVertex2d(0.20, -0.30); *// 3*

glEnd();

glBegin(**GL\_POLYGON**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.00, 0.20); *// 1*

glVertex2d(0.20, 0.20); *// 2*

glVertex2d(0.60, 0.70); *// 5*

glVertex2d(-0.10, 0.40); *// 6*

glEnd();

glBegin(**GL\_POLYGON**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.00, 0.20); *// 1*

glVertex2d(-0.10, 0.40); *// 6*

glVertex2d(-0.70, 0.60); *// 7*

glVertex2d(-0.70, 0.1); *// 11*

glVertex2d(-0.46, -0.038); *// 10*

glEnd();

glBegin(**GL\_LINES**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(-0.70, 0.1); *// 11*

glVertex2d(0.00, -0.30); *// 9*

glVertex2d(0.00, -0.30); *// 9*

glVertex2d(-0.70, -0.30); *// 8*

glVertex2d(-0.70, -0.30); *// 8*

glVertex2d(-0.70, 0.1); *// 11*

glEnd();

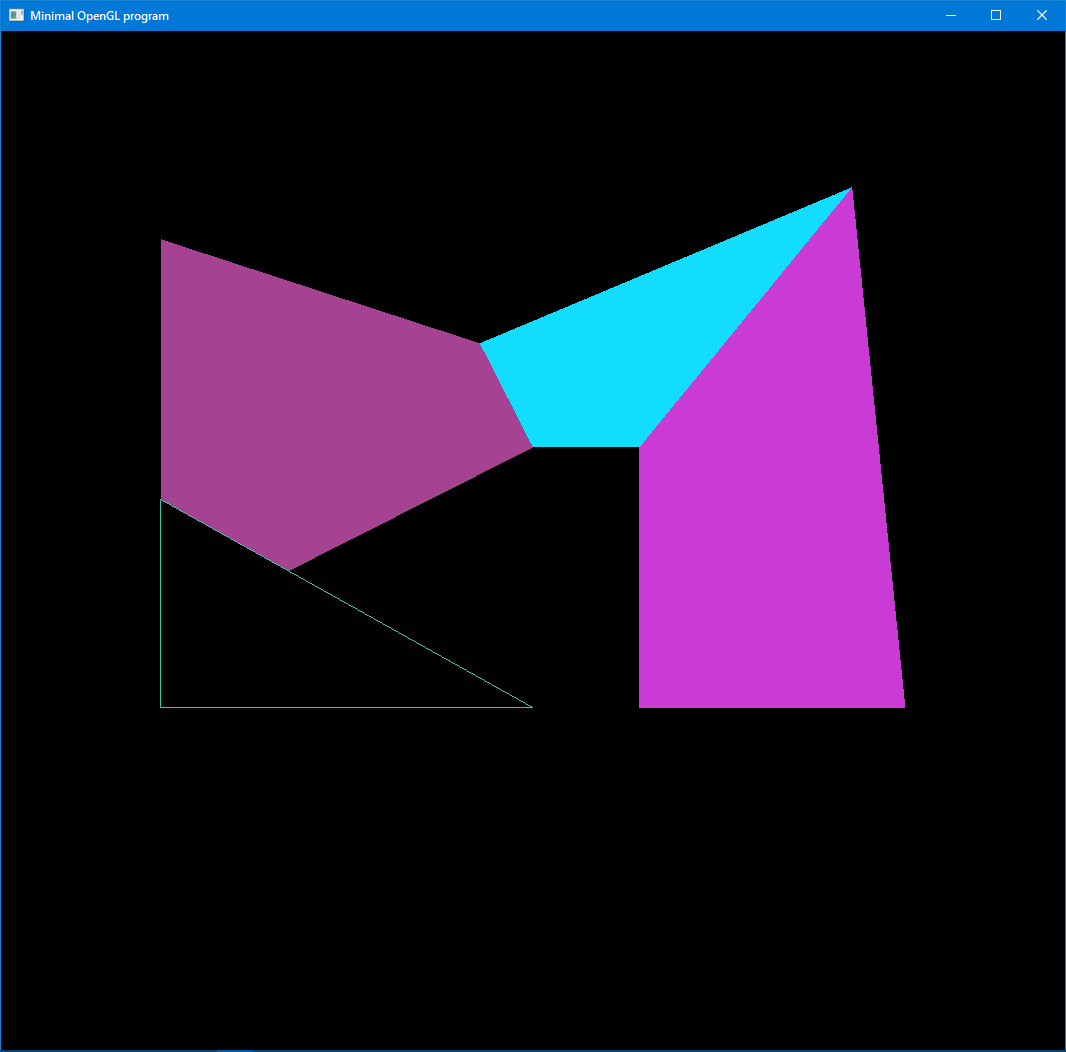


Рисунок 13 – Лицевые грани закрашены, обратные изображаются линиями

В) лицевые и обратные грани изображались линиями (каркасное изображение).

Содержимое функции *DrawGLScene*:

glClear(**GL\_COLOR\_BUFFER\_BIT**); *// очистка экрана*

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

glLineWidth(5);

srand(time(**NULL**));

glPointSize(5);

glBegin(**GL\_QUADS**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.20, 0.20); *// 2*

glVertex2d(0.60, 0.70); *// 5*

glVertex2d(0.70, -0.30); *// 4*

glVertex2d(0.20, -0.30); *// 3*

glEnd();

glBegin(**GL\_QUADS**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.00, 0.20); *// 1*

glVertex2d(0.20, 0.20); *// 2*

glVertex2d(0.60, 0.70); *// 5*

glVertex2d(-0.10, 0.40); *// 6*

glEnd();

glBegin(**GL\_POLYGON**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(0.00, 0.20); *// 1*

glVertex2d(-0.10, 0.40); *// 6*

glVertex2d(-0.70, 0.60); *// 7*

glVertex2d(-0.70, 0.1); *// 11*

glVertex2d(-0.46, -0.038); *// 10*

glEnd();

glBegin(**GL\_TRIANGLES**);

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glVertex2d(-0.70, 0.1); *// 11*

glVertex2d(0.00, -0.30); *// 9*

glVertex2d(-0.70, -0.30); *// 8*

glEnd();

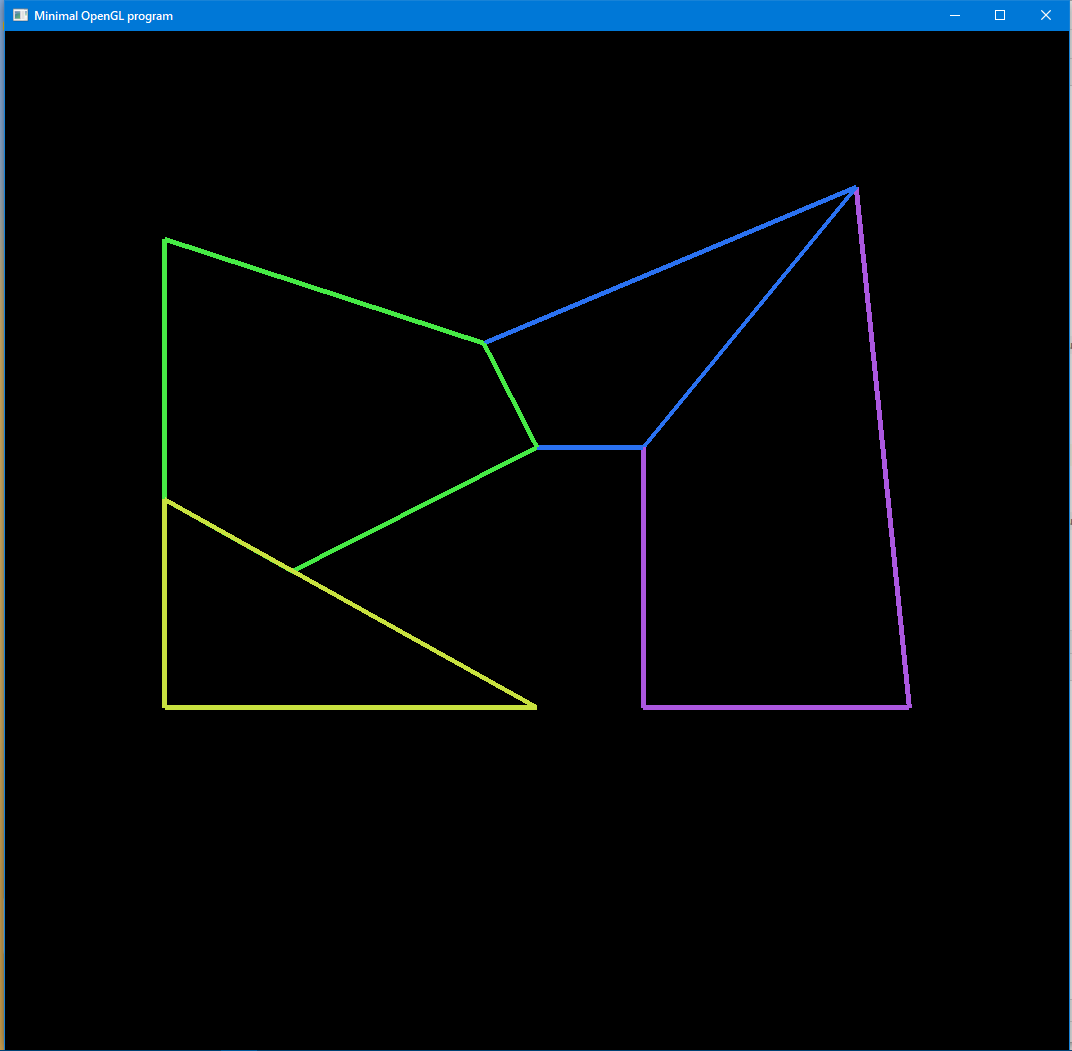


Рисунок 14 – Каркасное изображение