**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**



ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

**Лабораторная работа №1**

по дисциплине: Компьютерная графика

тема: «Растровые алгоритмы»

Выполнил: ст. группы ПВ-223

Пахомов Владислав Андреевич

Проверили:

ст. пр. Осипов Олег Васильевич

Белгород 2024 г.

**Лабораторная работа №1  
Растровые алгоритмы  
Вариант 8**

**Цель работы:** изучение алгоритмов Брезенхейма растеризации графических примитивов: отрезков, окружностей.

**Задания для выполнения к работе:**

1. Изучить целочисленные алгоритмы Брезенхейма для растеризации окружности и линии.
2. Разработать алгоритм и составить программу для построения на экране изображения в соответствии с номером варианта (по журналу старосты). В качестве исходных данных взять указанные в таблице №1.

**Задание:**

Реализовать вращение 4-конечной звезды против часовой стрелки.

Пусть – ширина экрана, – высота экрана. Диаметр описанной вокруг равностороннего треугольника окружности примем равным . Диаметр вписанной в треугольник окружности будет в два раза меньше описанного r = R / 2. Сторона треугольника будет равна . Введём также центр экрана . Вектор определяющие точки треугольника будут равны:

С окружностями с радиусом R и r и центром C получим равносторонний треугольник с описанной и вписанной окружностью.

Высота и ширина четырёхконечной звезды будет равна радиусу вписанной окружности, однако пока что обозначим её центр в точке (0, 0). Также введём некоторый отступ от центра для частей, где соединяются выступы звезды. Он будет равен . Тогда звезда задаётся координатами

Для выполнения трансформаций над звездой будем использовать SRT-матрицу. Для преобразования координат нужно переменожить каждую из них на SRT матрицу:

Где соответственно отвечают за размер, вращение и перемещение вектора. Для двухмерного пространства достаточно будет использовать трёхмерные матрицы. Матрица составляется следующим образом:

В нашем случае размер не изменяется (, а смещение относительно центра = С (. Для рисования варианта задания представлен текст программы на C++:

int W = frame.width, H = frame.height;

// Размер рисунка возьмём меньше (7 / 8), чтобы он не касался границ экрана

float a = 7.0f / 8 \* ((W < H) ? W - 1 : H - 1) / sqrt(2);

if (a < 1) return; // Если окно очень маленькое, то ничего не рисуем

float angle = -global\_angle; // Угол поворота

a = a / 2;

coordinate C = { W / 2, H / 2 };

// Рисуем описанную окружность

frame.Circle((int)C.x, (int)C.y, (int)a, COLOR(0, 0, 0));

// Рисуем вписанную окружность

frame.Circle((int)C.x, (int)C.y, (int)(a \* 0.5), COLOR(0, 0, 0));

//Рисуем треугольник

double t = (3 \* a) / sqrt(3);

coordinate triangleA = { C.x, C.y - a };

coordinate triangleB = { C.x - t / 2, C.y + a / 2 };

coordinate triangleC = { C.x + t / 2, C.y + a / 2 };

frame.DrawLine(triangleA.x + 0.5, triangleA.y + 0.5, triangleB.x + 0.5, triangleB.y + 0.5, { 56, 93, 138 });

frame.DrawLine(triangleC.x + 0.5, triangleC.y + 0.5, triangleB.x + 0.5, triangleB.y + 0.5, { 56, 93, 138 });

frame.DrawLine(triangleA.x + 0.5, triangleA.y + 0.5, triangleC.x + 0.5, triangleC.y + 0.5, { 56, 93, 138 });

Matrix S = { 1, 0, 0,

0, 1, 0,

0, 0, 1 };

Matrix R = { cos(angle), -sin(angle), 0,

sin(angle), cos(angle), 0,

0, 0, 1 };

Matrix T = { 1, 0, W / 2.0,

0, 1, H / 2.0,

0, 0, 1 };

Matrix SRT = (T.multiply(R)).multiply(S);

double starOffset = a / 12;

coordinate star[8] = {

{ 0, a / 2 },

{ starOffset, starOffset },

{ a / 2, 0 },

{ starOffset, -starOffset },

{ 0, -a / 2 },

{ -starOffset, -starOffset },

{ -a / 2, 0 },

{ -starOffset, starOffset } };

for (int i = 0; i < 8; i++)

{

Vector pointVector = { star[i].x, star[i].y, 1 };

pointVector = SRT.multiply(pointVector);

star[i].x = pointVector.vector[0];

star[i].y = pointVector.vector[1];

}

for (int i = 0; i < 8; i++)

{

int i2 = (i + 1) % 8;

frame.DrawLine( // Добавляем везде 0.5f, чтобы вещественные числа правильно округлялись при преобразовании к целому типу

int(star[i].x + 0.5f),

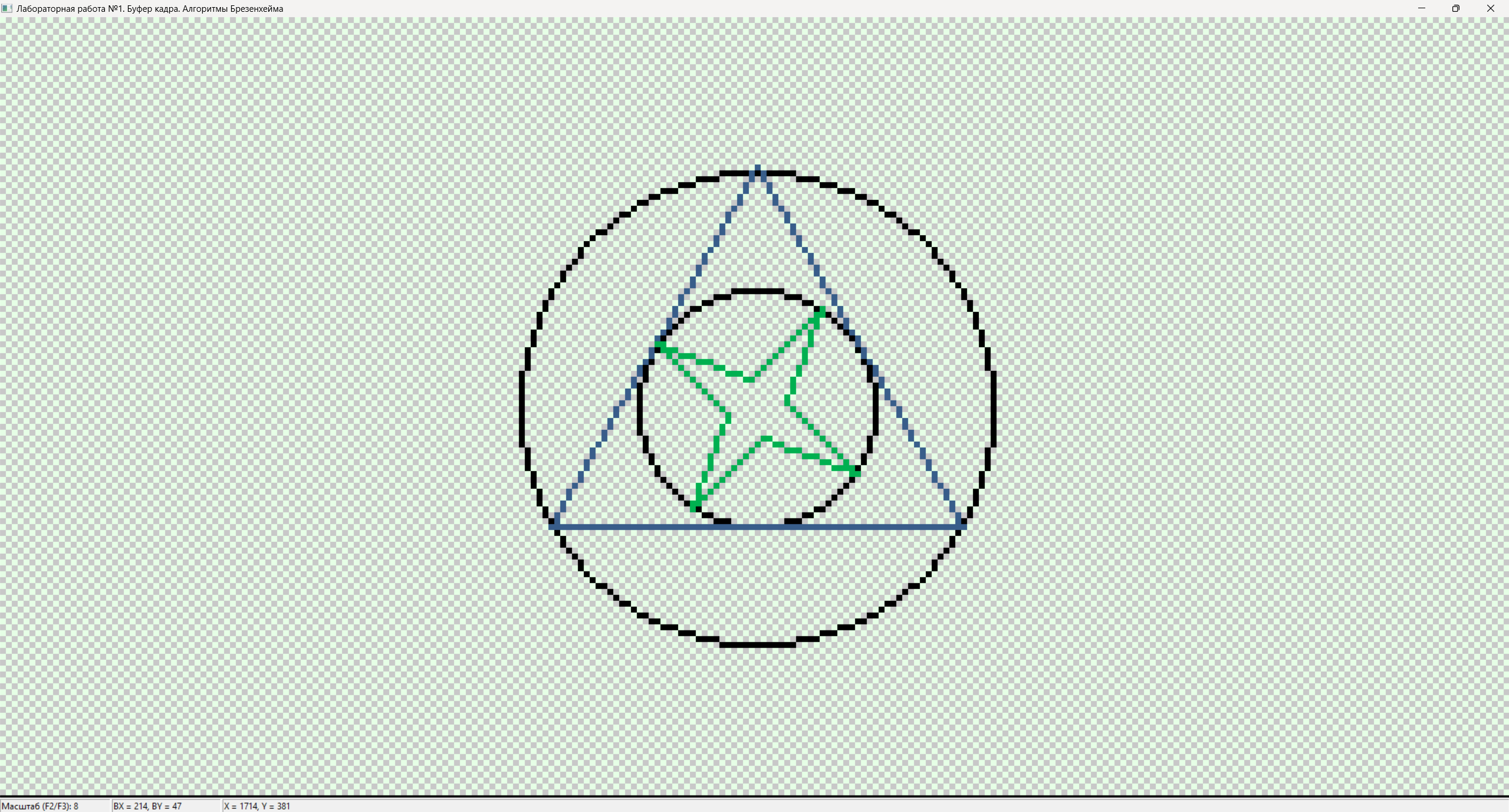
int(star[i].y + 0.5f),

int(star[i2].x + 0.5f),

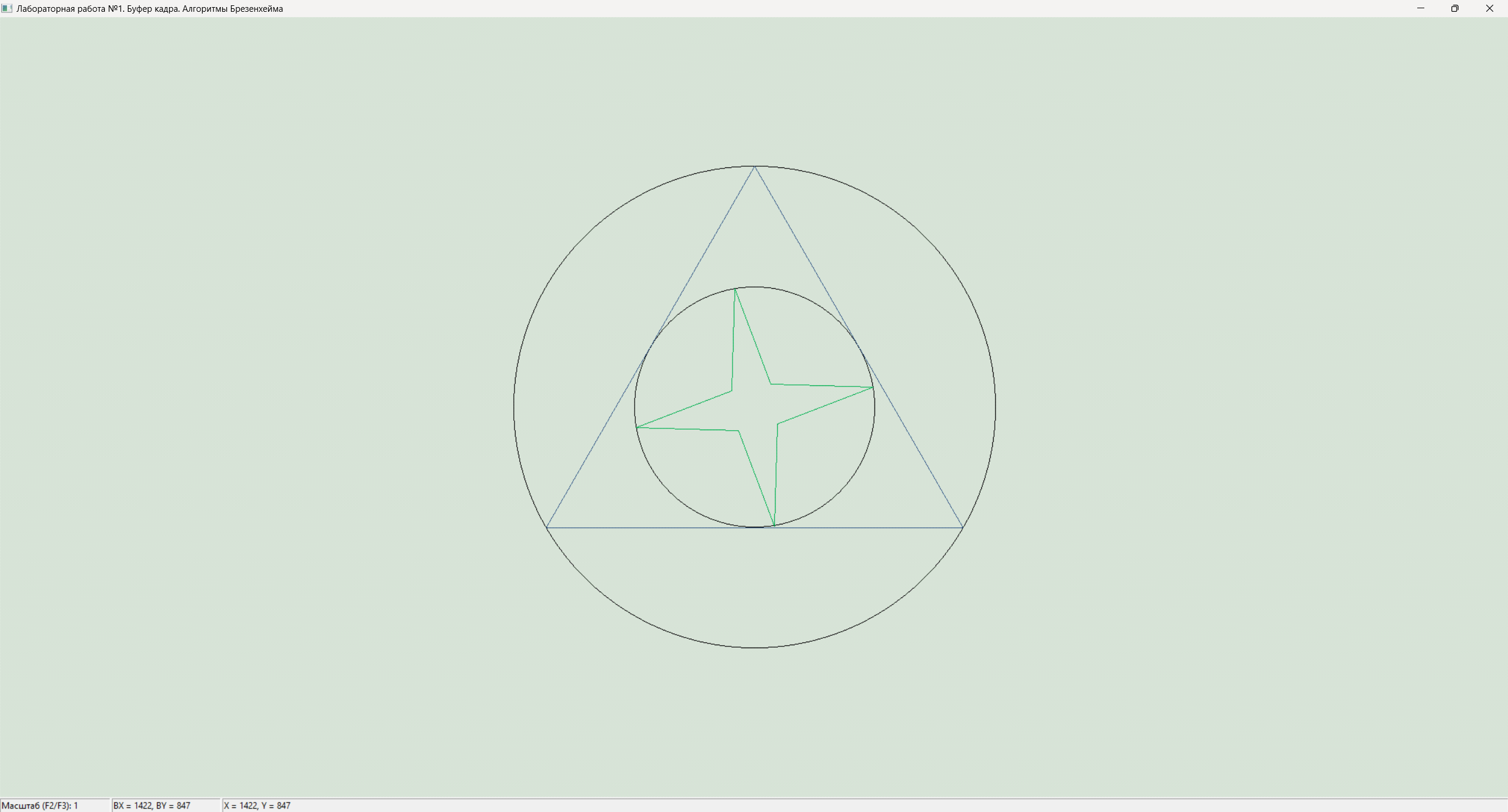
int(star[i2].y + 0.5f), COLOR(0, 176, 80));

}

Рендер при низкой детализации



Рендер при высокой детализации



**Вывод:** в ходе лабораторной работы получены навыки создания простейших ассемблерных программ с использованием пакета masm32, получены навыки пользования отладчиком x32dbg.