Дисциплина: Компьютерная графика

Лабораторная работа №2

Алгоритм Бразенхайма для рисования линии

Выполнила: Маслова Анастасия Сергеевна

№ Студ.билета: 1032216455

Группа: НКНбд-01-21

Задание: написать компьютерную программу для построения геометрического фрактала «Дерево Пифагора», используя функцию myline, являющуюся реализацией алгоритма Бразенхайма для рисования линии.

Ход работы:

Алгоритм построения дерева Пифагора был описан в 1-ой лабораторной работе.

Алгоритм Бразенхайма (Брезенхэма) – алгоритм построения прямой линии на экране компьютера, позволяющий определить, какие пиксели нужно закрасить, чтобы получить близкое приближение прямой линии между двумя заданными точками.   
  
Данный алгоритм описывает процесс построения отрезка между двумя точками на плоскости. Начальная точка имеет координаты (x\_0, y\_0), а конечная точка - (x\_1, y\_1). Предполагается, что отрезок идет вправо и вниз, то есть горизонтальное расстояние (x\_1 - x\_0) больше вертикального расстояния (y\_1 - y\_0). Для каждого столбца x в диапазоне от x\_0 до x\_1, мы хотим определить ближайшую строку y к линии и нарисовать точку (x, y).

Формула для вычисления координаты y лежит на прямой, проходящей через две заданные точки: y = ((y\_1 - y\_0) / (x\_1 - x\_0)) \* (x - x\_0) + y\_0. Однако точное значение y не обязательно вычислять. Мы можем заметить, что при каждом шаге увеличения x, значение y уменьшается, и мы добавляем к x единицу и значение наклона s = (y\_1 - y\_0) / (x\_1 - x\_0) (которое можно вычислить заранее). На каждом шаге мы либо сохраняем текущее значение y, либо уменьшаем его на 1. Выбор между сохранением текущего значения y и уменьшением его на 1 зависит от значения ошибки, которая представляет собой вертикальное расстояние между текущим значением y и точным значением y для текущего x. При увеличении x, значение ошибки увеличивается на значение наклона s. Если ошибка превышает 1.0, это означает, что линия стала ближе к следующему целочисленному значению y. В этом случае мы увеличиваем y на 1.0, одновременно уменьшая значение ошибки на 1.0. Таким образом, мы можем построить отрезок между двумя точками, определяя ближайшие значения y для каждого столбца x и рисуя точки (x, y) на плоскости. Этот алгоритм позволяет нам эффективно и точно нарисовать линию между двумя точками, даже если нам необходимо только приближенное значение y.

Существует алгоритм Бразенхайма для рисования окружностей, который схож алгоритмом для рисования линии. Этот алгоритм пошагово генерирует очередные точки окружности, выбирая на каждом шаге для занесения пиксела точку растра, ближайшую к истинной окружности, так чтобы ошибка была минимальной.

Код:

*#include <stdlib.h>*

*#include <stdio.h>*

*#include <conio.h>*

*#include <math.h>*

*#include "graphics.h"*

**void** myline**(int** x1**,** **int** y1**,** **int** x2**,** **int** y2**)** **{**

**int** deltax **=** abs**(**x2 **-** x1**);**

**int** deltay **=** abs**(**y2 **-** y1**);**

**int** signx **=** x1 **>** x2 **?** **-1** **:** **1;**

**int** signy **=** y1 **>** y2 **?** **-1** **:** **1;**

**int** error **=** deltax **-** deltay**;**

putpixel**(**x2**,** y2**,** WHITE**);**

**while(**x1 **!=** x2 **||** y1 **!=** y2**)**

**{**

putpixel**(**x1**,** y1**,** WHITE**);**

**int** error2 **=** error **\*** **2;**

**if(**error2 **>** **-**deltay**)**

**{**

error **-=** deltay**;**

x1 **+=** signx**;**

**}**

**if(**error2 **<** deltax**)**

**{**

error **+=** deltax**;**

y1 **+=** signy**;**

**}**

**}**

**}**

**void** mycircle**(int** x0**,** **int** y0**,** **int** radius**)** **{**

**int** x **=** **0;**

**int** y **=** radius**;**

**int** delta **=** **1** **-** **2** **\*** radius**;**

**int** error **=** **0;**

**while(**y **>=** **0)** **{**

putpixel**(**x0 **+** x**,** y0 **+** y**,** WHITE**);**

putpixel**(**x0 **+** x**,** y0 **-** y**,** WHITE**);**

putpixel**(**x0 **-** x**,** y0 **+** y**,** WHITE**);**

putpixel**(**x0 **-** x**,** y0 **-** y**,** WHITE**);**

error **=** **2** **\*** **(**delta **+** y**)** **-** **1;**

**if(**delta **<** **0** **&&** error **<=** **0)** **{**

**++**x**;**

delta **+=** **2** **\*** x **+** **1;**

**continue;**

**}**

error **=** **2** **\*** **(**delta **-** x**)** **-** **1;**

**if(**delta **>** **0** **&&** error **>** **0)** **{**

**--**y**;**

delta **+=** **1** **-** **2** **\*** y**;**

**continue;**

**}**

**++**x**;**

delta **+=** **2** **\*** **(**x **-** y**);**

**--**y**;**

**}**

**}**

**void** pifagor**(double** x0**,** **double** y0**,** **double** a**,** **double** n**,** **double** fi**,** **double** alpha**)** **{**

**if(**n**==0){return;}**

**double** l**,** ll**,** x1**,** y1**,** x2**,** y2**,** x3**,** y3**,** x4**,** y4**,** xx**,** yy**,** xo**,** yo**,** r**,** rr**;**

n**--;**

x1**=**x0**;**

y1**=**y0**;**

x4**=**x0**+**a**\***cos**(**fi**);**

y4**=**y0**-**a**\***sin**(**fi**);**

x2**=**x0**-**a**\***sin**(**fi**);**

y2**=**y0**-**a**\***cos**(**fi**);**

x3**=**x4**-**a**\***sin**(**fi**);**

y3**=**y4**-**a**\***cos**(**fi**);**

xo **=** **(**x3**+**x1**)/2;**

yo **=** **(**y1**+**y3**)/2;**

r **=** **(**a**\***sqrt**(2))/2;**

myline**(**x1**,**y1**,**x2**,**y2**);**

myline**(**x2**,**y2**,**x3**,**y3**);**

myline**(**x3**,**y3**,**x4**,**y4**);**

myline**(**x4**,**y4**,**x1**,**y1**);**

mycircle**(**xo**,** yo**,** r**);**

l **=** a**\***cos**(**alpha**);**

ll **=** a**\***sin**(**alpha**);**

xx**=**x2**+**l**\***sin**((**M\_PI**/2)-**alpha**-**fi**);**

yy**=**y2**-**l**\***cos**((**M\_PI**/2)-**alpha**-**fi**);**

myline**(**x2**,** y2**,** xx**,** yy**);**

myline**(**xx**,**yy**,**x3**,**y3**);**

pifagor**(**x2**,**y2**,** l**,** n**,** fi**+**alpha**,** alpha**);**

pifagor**(**xx**,**yy**,** ll**,** n**,** alpha**+**fi**-**M\_PI**/2,** alpha**);**

**}**

**int** main**()** **{**

**int** gddriver **=** DETECT**,** gmode**,** errorcode**;**

initgraph**(&**gddriver**,** **&**gmode**,** ""**);**

pifagor**(1000,** **800,** **200,** **4,** **0,** M\_PI**/6);**

getch**();**

closegraph**();**

**return** **0;**

**}**

Результат работы программы:

