Министерство науки и высшего образования

Российской Федерации

НГТУ

Лабораторная работа № 2-3

РИСОВАНИЕ ЛИНИЙ. РИСОВАНИЕ ОКРУЖНОСТЕЙ

|  |  |
| --- | --- |
| Выполнил:  Жуков Иван  Группа: АТ-34 | Проверил:  Смагин С. М. |

Новосибирск

2024

Задание

1. Разработать программу, которая будет рисовать линию, построенную по алгоритму цифрового дифференциального анализатора и по алгоритму Брезенхема на сетке с заданными параметрами (как в Лабораторной работе №1).
2. Разработать программу, которая будет рисовать окружность по алгоритму Брезенхема на сетке с заданными параметрами (как в Лабораторной работе №1).

Описание основных функций

void BresenhamLine(int x0, int y0, int x1, int y1, COLORREF color)

Функция реализует алгоритм Брезенхэма для рисования линии между двумя заданными точками. Она вычисляет разности по x и y, чтобы определить направление линии. Устанавливает начальное значение переменной ошибки. Использует цикл, чтобы последовательно рисовать пиксели (или точки) на линии, корректируя координаты в зависимости от ошибки. Цвет пикселей определяется параметром color.

void CDA\_LLine(int x1, int y1, int x2, int y2)

Эта функция реализует алгоритм ЦДА (Цифровая Дифференциальная Аналоговая) для рисования линий. Вычисляет изменения по x и y. - Устанавливает количество шагов для рисования, основываясь на большем значении из dx и dy. Инкрементально вычисляет координаты на протяжении всего пути от начальной до конечной точки, рисуя пиксели с использованием предыдущей функции CreatePoint.

void BresenhamCircle(int centerX, int centerY, int radius)

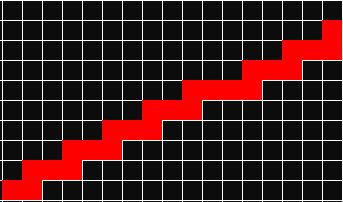
Эта функция использует алгоритм Брезенхэма для рисования окружностей. Инициализирует начальные координаты для верхней точки окружности и значение ошибки. Рисует точки окружности по всем восьми секторам, используя симметрию окружности. Увеличивает x и уменьшает y, корректируя значение ошибки (d) в зависимости от текущего положения, пока x меньше y. Как и в других функциях, используется CreatePoint для рисования пикселей окружности заданным цветом.

Описание работы программы

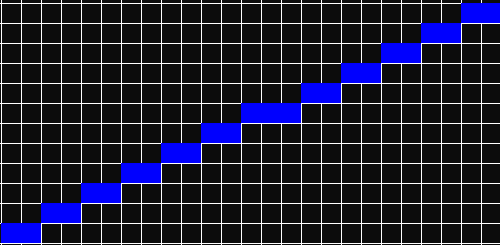
1. **Инициализация**: Программа включает в себя необходимые заголовочные файлы, такие как windows.h, iostream, locale и cmath. Определяются размеры окна (WINDOW\_WIDTH, WINDOW\_HEIGHT), размер ячейки для сетки (CELL\_SIZE), а также количество ячеек по горизонтали и вертикали (NUM\_CELLS\_X, NUM\_CELLS\_Y). Определяются дескрипторы окна (HWND) и контекста устройства (HDC).
2. **Создание графического окна**: В главной функции main() устанавливается локаль для консоли для поддержки форматирования символов. Получается дескриптор консольного окна с помощью функции GetConsoleWindow(), а затем контекст устройства для рисования с помощью GetDC().
3. **Работа с интерфейсом пользователя**: Программа предлагает пользователю выбрать, что он хочет нарисовать: линию (1) или окружность (2). После выбора, программа запрашивает необходимые координаты и параметры в зависимости от выбора пользователя: Для линии: начальные и конечные координаты, а также выбор алгоритма (Брезенха или ЦДА). Для окружности: координаты центра и радиус. Все вводимые данные очищаются в консоли для лучшей читабельности до рисования.
4. **Рисование**: В зависимости от ввода пользователя происходит вызов соответствующих функций: Если выбрана линия, то программа вызывает либо BresenhamLine(), если выбрали метод Брезенха, либо CDA\_LLine(), если выбран алгоритм ЦДА. Перед рисованием сетка также обновляется с помощью DrawGrid(). Если выбрана окружность, вызывается функция BresenhamCircle() с установленными пользователем координатами и радиусом. В процессе рисования каждая линия или окружность отображается в заданном цвете (красный для линий, зеленый для окружностей).
5. З**авершение работы**: После завершения рисования программа освобождает контекст устройства с помощью ReleaseDC() и завершает свою работу. Результат остается на экране в графическом контексте, пока окно не будет закрыто пользователем.

Скриншоты

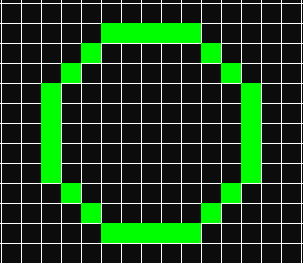
Линия по алгоритму Брезенхема:



Линия по ЦДА:



Окружность:



Листинг программы

#include <windows.h>

#include <iostream>

#include <locale>

#include <cmath>

int WINDOW\_WIDTH = 800, WINDOW\_HEIGHT = 600, CELL\_SIZE = 20;

int NUM\_CELLS\_X, NUM\_CELLS\_Y;

HWND hwnd;

HDC hdc;

void DrawGrid() {

HPEN whitePen = CreatePen(PS\_SOLID, 1, RGB(255, 255, 255));

SelectObject(hdc, whitePen);

for (int x = 0; x <= WINDOW\_WIDTH; x += CELL\_SIZE) {

MoveToEx(hdc, x, 0, NULL);

LineTo(hdc, x, WINDOW\_HEIGHT);

}

for (int y = 0; y <= WINDOW\_HEIGHT; y += CELL\_SIZE) {

MoveToEx(hdc, 0, y, NULL);

LineTo(hdc, WINDOW\_WIDTH, y);

}

DeleteObject(whitePen);

}

void CreatePoint(int x, int y, COLORREF color) {

HBRUSH brush = CreateSolidBrush(color);

SelectObject(hdc, brush);

RECT rect;

SetRect(&rect, x \* CELL\_SIZE, (NUM\_CELLS\_Y - y - 1) \* CELL\_SIZE,

(x + 1) \* CELL\_SIZE, (NUM\_CELLS\_Y - y) \* CELL\_SIZE);

FillRect(hdc, &rect, brush);

DeleteObject(brush);

}

void BresenhamLine(int x0, int y0, int x1, int y1, COLORREF color) {

int dx, dy, ch = 0, e, i;

x1 -= x0;

dx = abs(x1);

y1 -= y0;

dy = abs(y1);

if (!x1 && !y1) return;

if (x1) x1 = x1 < 0 ? -1 : 1;

if (y1) y1 = y1 < 0 ? -1 : 1;

if (dy > dx) {

std::swap(dx, dy);

ch = 1;

}

int dx2 = dx << 1;

int dy2 = dy << 1;

e = dy2 - dx;

for (i = 0; i <= dx; ++i) {

CreatePoint(x0, y0, color);

if (e > 0) {

if (ch)

x0 += x1;

else

y0 += y1;

e -= dx2;

}

else {

if (ch)

y0 += y1;

else

x0 += x1;

e += dy2;

}

}

}

void CDA\_LLine(int x1, int y1, int x2, int y2) {

int dx = x2 - x1;

int dy = y2 - y1;

int steps = (abs(dx) > abs(dy)) ? abs(dx) : abs(dy);

float xIncrement = dx / (float)steps;

float yIncrement = dy / (float)steps;

float x = x1;

float y = y1;

for (int i = 0; i <= steps; i++) {

CreatePoint(round(x), round(y), RGB(0, 0, 255));

x += xIncrement;

y += yIncrement;

}

}

void BresenhamCircle(int centerX, int centerY, int radius) {

int x = 0;

int y = radius;

int d = 1 - radius;

CreatePoint(centerX + x, centerY + y, RGB(0, 255, 0));

CreatePoint(centerX - x, centerY + y, RGB(0, 255, 0));

CreatePoint(centerX + x, centerY - y, RGB(0, 255, 0));

CreatePoint(centerX - x, centerY - y, RGB(0, 255, 0));

CreatePoint(centerX + y, centerY + x, RGB(0, 255, 0));

CreatePoint(centerX - y, centerY + x, RGB(0, 255, 0));

CreatePoint(centerX + y, centerY - x, RGB(0, 255, 0));

CreatePoint(centerX - y, centerY - x, RGB(0, 255, 0));

while (x < y) {

if (d < 0) {

d += 2 \* x + 3;

}

else {

d += 2 \* (x - y) + 5;

--y;

}

++x;

CreatePoint(centerX + x, centerY + y, RGB(0, 255, 0));

CreatePoint(centerX - x, centerY + y, RGB(0, 255, 0));

CreatePoint(centerX + x, centerY - y, RGB(0, 255, 0));

CreatePoint(centerX - x, centerY - y, RGB(0, 255, 0));

CreatePoint(centerX + y, centerY + x, RGB(0, 255, 0));

CreatePoint(centerX - y, centerY + x, RGB(0, 255, 0));

CreatePoint(centerX + y, centerY - x, RGB(0, 255, 0));

CreatePoint(centerX - y, centerY - x, RGB(0, 255, 0));

}

}

int main() {

std::locale::global(std::locale(""));

hwnd = GetConsoleWindow();

hdc = GetDC(hwnd);

system("cls"); // Очистка консоли

NUM\_CELLS\_X = WINDOW\_WIDTH / CELL\_SIZE;

NUM\_CELLS\_Y = WINDOW\_HEIGHT / CELL\_SIZE;

int choice;

std::cout << "Что рисовать? 1 - Линия, 2 - Окружность: ";

std::cin >> choice;

system("cls");

if (choice == 1) {

int x\_start, y\_start, x\_end, y\_end, algo\_choice;

std::cout << "Введите начальные координаты (x1, y1) для линии: ";

std::cin >> x\_start >> y\_start;

std::cout << "Введите конечные координаты (x2, y2) для линии: ";

std::cin >> x\_end >> y\_end;

std::cout << "Выберите алгоритм: 1 - Брезенхом, 2 - ЦДА: ";

std::cin >> algo\_choice;

system("cls");

DrawGrid(); // Рисуем сетку

if (algo\_choice == 1) {

BresenhamLine(x\_start, y\_start, x\_end, y\_end, RGB(255, 0, 0)); // Рисуем линию методом Брезенхома

}

else if (algo\_choice == 2) {

CDA\_LLine(x\_start, y\_start, x\_end, y\_end); // Рисуем линию методом ЦДА

}

else {

std::cout << "Неверный выбор алгоритма!" << std::endl;

}

}

else if (choice == 2) {

int centerX, centerY, radius;

std::cout << "Введите координаты центра окружности (x, y): ";

std::cin >> centerX >> centerY;

std::cout << "Введите радиус окружности: ";

std::cin >> radius;

system("cls");

DrawGrid(); // Рисуем сетку

BresenhamCircle(centerX, centerY, radius); // Рисуем окружность

}

else {

std::cout << "Неверный выбор." << std::endl;

}

ReleaseDC(hwnd, hdc);

return 0;

}