**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра САПР**

**ОТЧЁТ**

**По лабораторной работе №4**

**По дисциплине «Программирование»**

Студент гр. 3351 \_\_\_\_\_\_\_\_\_\_\_\_ Морозов А.А.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ Рыжов Н.Г.

Санкт-Петербург

2024

цель работы

**Задание**

Написать программу – графический редактор, позволяющую автоматизировать процесс создания вышивки сначала на экране компьютера, а затем переноса данных в виде последовательности длины стежков в автоматическую швейную машину в качестве программы управления.

(0,0)

(x,y)

Y

X

Пользователь с помощью курсора выбирает нужную точку экрана и нажимает «Enter» или левую кнопку мыши, фиксируя координату (0,0) начала рисунка в декартовой системе координат. Для задания очередного стежка курсор перемещается в другую точку, но не дальше, чем на максимальную длину стежка, и т. д., образуя узор вышивки. Окончание ввода рисунка производится по нажатию клавиши «Esc».

**Требования по выполнению задания**

1. Последовательность стежков хранится в виде связного списка, фиксирующего для каждой точки смещение по осям X и Y относительно предыдущей точки.
2. Предусмотреть возможность отмены последнего введенного стежка.
3. Программа должна обеспечивать: сохранение рисунка (списка последовательности стежков) в виде файла, возможность начала нового рисунка, либо продолжение работы с ранее сохраненным в файле рисунком.
4. Параметры работы программы должны сохраняться в отдельном файле.
5. По окончании работы вывести на экран список координат всех стежков.
6. Рисунок не должен ограничиваться размерами экрана.
7. Длина стежка ограничена заданным максимальным значением.
8. Реализовать команды смещения рисунка влево/вправо/вверх/вниз.
9. Реализовать команды масштабирования (увеличения и уменьшения размера изображения), без изменения фактических размеров, сохраняемых в файле.
10. От последней зафиксированной точки рисунка до курсора отобразить линию другого цвета, отличающегося от основного рисунка. При превышении максимальной длины стежка линия не отображается.
11. Выводить на экран параметры курсора относительно последнего стежка: смещение x, y и расстояние.
12. Программа должна иметь удобный пользовательский интерфейс.

**Текст программы**

#include <SFML/Graphics.hpp>

#include <vector>

#include <fstream>

#include <iostream>

#include <cmath>

#include <sstream>

#include <Windows.h>

using namespace std;

// Структура для хранения точек

// Каждая точка имеет координаты x и y

struct Point {

int x;

int y;

};

// Глобальные переменные

// vector points - здесь хранятся все точки, которые будут рисоваться на экране

vector<Point> points;

// scaleFactor - масштабирующий коэффициент, изменяет размер рисунка на экране

float scaleFactor = 1.0f;

// translation - смещение рисунка на экране

sf::Vector2f translation;

// maxStepDistance - максимальное расстояние между точками, если расстояние больше, то новая точка не добавляется

int maxStepDistance = 100;

// showCoordinates - показывать или нет координаты курсора мыши относительно последней точки

bool showCoordinates = false;

// lineColor - цвет линий, соединяющих точки

sf::Color lineColor = sf::Color::Red;

// gridSize - размер сетки, которая будет рисоваться на экране

int gridSize = 50;

int gridOffsetX = 0; // Смещение сетки по X

int gridOffsetY = 0; // Смещение сетки по Y

const int sizeG = 200;

const int VideoMode800 = 800;

const int VideoMode600 = 600;

const int sth16 = 16;

const int sth10 = 10;

// Для отображения информации в окне

bool showFileName = true;

bool showSize = true;

bool showPathsCount = true;

int minX = INT\_MAX;

int minY = INT\_MAX;

int maxX = INT\_MIN;

int maxY = INT\_MIN;

// Функция для добавления новой точки

// Принимает координаты точки на экране (screenX, screenY)

// Преобразует их в координаты точки в системе координат рисунка (x, y)

// Проверяет, что расстояние между новой точкой и последней точкой не больше maxStepDistance

// Добавляет новую точку в vector points

void addPoint(int screenX, int screenY) {

int x = static\_cast<int>((screenX - translation.x) / scaleFactor);

int y = static\_cast<int>((screenY - translation.y) / scaleFactor);

if (!points.empty()) {

Point lastPoint = points.back();

int dx = x - lastPoint.x;

int dy = y - lastPoint.y;

float distance = sqrt(dx \* dx + dy \* dy);

if (distance > maxStepDistance) {

return;

}

}

points.push\_back({ x, y });

// Обновляем минимальные и максимальные координаты

minX = min(minX, offsetX);

minY = min(minY, offsetY);

maxX = max(maxX, offsetX);

maxY = max(maxY, offsetY);

}

// Функция для сохранения точек в файл

// Принимает имя файла (filename)

// Открывает файл для записи

// Записывает в файл координаты всех точек из vector points

// Выводит сообщение об успешном сохранении или об ошибке при сохранении

void savePointsToFile(const string& filename) {

ofstream file(filename);

if (file.is\_open()) {

for (const auto& point : points) {

file << point.x << " " << point.y << "\n";

}

file.close();

cout << "Точки сохранены в файл: " << filename << endl;

for (const auto& stitch : stitches) {

cout << "(" << stitch.offsetX << ", " << stitch.offsetY << ")" << endl;

}

}

else {

cerr << "Ошибка при сохранении файла: " << filename << endl;

}

}

// Функция для загрузки точек из файла

// Принимает имя файла (filename)

// Открывает файл для чтения

// Читает из файла координаты точек и добавляет их в vector points

// Выводит сообщение об успешном загрузке или об ошибке при загрузке

void loadPointsFromFile(const string& filename) {

points.clear();

ifstream file(filename);

if (file.is\_open()) {

int x, y;

while (file >> x >> y) {

points.push\_back({ x, y });

}

file.close();

cout << "Точки загружены из файла: " << filename << endl;

}

else {

cerr << "Ошибка при загрузке файла: " << filename << endl;

}

}

bool createNewDrawing() {

cout << "-=-=- Главное меню -=-=-" << endl;

cout << "1. Новый рисунок" << endl;

cout << "2. Загрузить рисунок" << endl;

cout << "-=-=-=-=-=-=-=-=-=-=-=-=\n" << endl;

int choice;

cin >> choice;

if (choice == 1) {

return true;

}

else if (choice == 2) {

return false;

}

else {

cout << "Некорректный выбор. Попробуйте еще раз." << endl;

return createNewDrawing();

}

}

int main() {

setlocale(LC\_ALL, "Russian");

// Создание окна приложения

// sf::VideoMode(800, 600) - размер окна (ширина, высота)

// "Редактор" - заголовок окна

sf::RenderWindow window(sf::VideoMode(VideoMode800, VideoMode600), "Редактор");

std::string filename;

// Создание текста для вывода информации о координатах курсора

sf::Text cursorInfo;

cursorInfo.setCharacterSize(sth16); // размер шрифта

cursorInfo.setFillColor(sf::Color::Black); // цвет текста

cursorInfo.setPosition(sth10, sth10); // координаты текста на экране

bool isNewDrawing = createNewDrawing();

if (!isNewDrawing) {

string filename;

cout << "Введите имя файла для открытия: ";

cin >> filename;

loadPointsFromFile(filename);

}

// Главный цикл приложения

// window.isOpen() - проверяет, что окно приложения не закрыто

while (window.isOpen()) {

sf::Event event;

// Обновляем текст с информацией о рисунке

ostringstream infoOss;

infoOss << "Количество стежков: " << stitches.size() << "\n";

infoOss << "Размер рисунка: " << (maxX - minX) << ", " << (maxY - minY);

infoOss << "Файл: " << (isNewDrawing ? "new\_drawing.txt" : filename) << "\n";

// Выводим имя файла, либо "new\_drawing.txt", если рисунок новый

infoText.setString(infoOss.str());

// window.pollEvent(event) - получает следующее событие из очереди событий

// пока в очереди есть события, цикл while будет выполняться

while (window.pollEvent(event)) {

// обработка событий

if (event.type == sf::Event::Closed) {

// событие закрытия окна, выходим из цикла

window.close();

}

else if (event.type == sf::Event::MouseButtonPressed && event.mouseButton.button == sf::Mouse::Left) {

// событие нажатия левой кнопки мыши, добавляем новую точку

addPoint(event.mouseButton.x, event.mouseButton.y);

}

else if (event.type == sf::Event::KeyPressed && event.key.code == sf::Keyboard::M) {

// событие нажатия клавиши M, сохраняем точки в файл

savePointsToFile("drawing.txt");

}

else if (event.type == sf::Event::KeyPressed) {

// Обработка команд перемещения окна просмотра и изменения масштаба

switch (event.key.code) {

case sf::Keyboard::Left: // Смещение рисунка

translation.x += 10 / scaleFactor;

break;

case sf::Keyboard::Right:

translation.x -= 10 / scaleFactor;

break;

case sf::Keyboard::Up:

translation.y += 10 / scaleFactor;

break;

case sf::Keyboard::Down:

translation.y -= 10 / scaleFactor;

break;

case sf::Keyboard::Q: // Переключение отображения названия файла

showFileName = !showFileName;

break;

case sf::Keyboard::W: // Переключение отображения размера рисунка

showSize = !showSize;

break;

case sf::Keyboard::E: // Переключение отображения количества стежков

showPathsCount = !showPathsCount;

break;

case sf::Keyboard::Equal: // Уменьшение масштаба

scaleFactor \*= 1.1f;

maxStepDistance \*= 1.1f; // Увеличиваем максимальную длину линии пропорционально масштабу

break;

case sf::Keyboard::Hyphen: // Увеличение масштаба

scaleFactor \*= 0.9f;

maxStepDistance \*= 0.9f; // Уменьшаем максимальную длину линии пропорционально масштабу

break;

case sf::Keyboard::B: // Смещение сетки к первой точке рисунка

if (!points.empty()) {

translation.x = -points.front().x \* scaleFactor + window.getSize().x / 2;

translation.y = -points.front().y \* scaleFactor + window.getSize().y / 2;

}

break;

default:

break;

}

}

}

window.clear(sf::Color::White); // очищаем экран, заливаем его белым цветом

// Отрисовка сетки

if (showCoordinates) {

sf::VertexArray grid(sf::Lines, 2 \* (window.getSize().x / gridSize + window.getSize().y / gridSize));

int index = 0;

for (int x = gridOffsetX % gridSize; x < window.getSize().x; x += gridSize) {

grid[index].position = sf::Vector2f(x, 0);

grid[index + 1].position = sf::Vector2f(x, window.getSize().y);

grid[index].color = sf::Color(sizeG, sizeG, sizeG);

grid[index + 1].color = sf::Color(sizeG, sizeG, sizeG);

index += 2;

}

for (int y = gridOffsetY % gridSize; y < window.getSize().y; y += gridSize) {

grid[index].position = sf::Vector2f(0, y);

grid[index + 1].position = sf::Vector2f(window.getSize().x, y);

grid[index].color = sf::Color(sizeG, sizeG, sizeG);

grid[index + 1].color = sf::Color(sizeG, sizeG, sizeG);

index += 2;

}

window.draw(grid);

}

// Создание и настройка текста для параметров

sf::Font font;

sf::Text paramsInfo;

paramsInfo.setFont(font);

paramsInfo.setCharacterSize(16);

paramsInfo.setFillColor(sf::Color::Black);

// Формирование строки с параметрами

std::stringstream paramsSS;

if (showFileName) {

paramsSS << "Файл: " << filename << "\n";

}

if (showSize) {

int width = points.empty() ? 0 : (points.back().x - points.front().x);

int height = points.empty() ? 0 : (points.back().y - points.front().y);

paramsSS << "Размер: " << width << "x" << height << "\n";

}

if (showPathsCount) {

paramsSS << "Количество стежков: " << points.size();

}

// Установка текста и позиции

paramsInfo.setString(paramsSS.str());

paramsInfo.setPosition(10, 10);

// Отображение текста с параметрами

window.draw(paramsInfo);

// рисуем линии, соединяющие точки

sf::VertexArray lines(sf::LinesStrip, points.size());

int i = 0;

for (const auto& point : points) {

lines[i].position = sf::Vector2f(point.x \* scaleFactor + translation.x, point.y \* scaleFactor + translation.y);

lines[i].color = sf::Color::Black;

++i;

}

window.draw(lines);

if (!points.empty()) {

// рисуем линию от последней точки до текущего положения курсора мыши

sf::Vector2i mousePosition = sf::Mouse::getPosition(window);

sf::Vector2f lastPointPosition(points.back().x \* scaleFactor + translation.x, points.back().y \* scaleFactor + translation.y);

sf::Vector2f direction = sf::Vector2f(mousePosition.x, mousePosition.y) - lastPointPosition;

float distance = sqrt(direction.x \* direction.x + direction.y \* direction.y);

if (distance > maxStepDistance) {

direction \*= maxStepDistance / distance;

}

sf::Vertex cursorLine[] = { sf::Vertex(lastPointPosition, lineColor), sf::Vertex(lastPointPosition + direction, lineColor) };

window.draw(cursorLine, 2, sf::Lines);

}

if (!points.empty()) {

// выводим информацию о координатах курсора мыши относительно последней точки

sf::Vector2i mousePosition = sf::Mouse::getPosition(window);

sf::Vector2f lastPointPosition(points.back().x, points.back().y);

sf::Vector2f offsetFromLastPoint = sf::Vector2f(mousePosition.x, mousePosition.y) - lastPointPosition;

float distance = sqrt(offsetFromLastPoint.x \* offsetFromLastPoint.x + offsetFromLastPoint.y \* offsetFromLastPoint.y);

ostringstream oss;

oss << "Положение точки (x, y): " << offsetFromLastPoint.x << ", " << offsetFromLastPoint.y << "\n";

oss << "Длина: " << distance;

cursorInfo.setString(oss.str());

}

else {

cursorInfo.setString("Больше точек нет.");

}

window.draw(cursorInfo); // рисуем текст на экране

window.display(); // обновляем экран, показываем все, что было нарисовано

}

return 0;

}

**Описание программы**

Программное обеспечение – Microsoft Visual Studio 2022. Язык программирования – C++. Программа, графический редактор, предоставляет пользователю возможность создавать рисунки вышивки на экране компьютера для последующего переноса данных в виде последовательности длины стежков в автоматическую швейную машину. Пользователь может рисовать с помощью левой кнопки мыши. Также кнопка M – сохраняет рисунок, кнопка Z – удаляет последнее действие, кнопки Q, W, E, служат для отображения данных о рисунке.

**Руководство пользователя**

Программа выводит главное меню в котором можно выбрать: создать новый рисунок или загрузить уже готовый. В случае неверного ввода меню перезапустится. (Рис. 1)

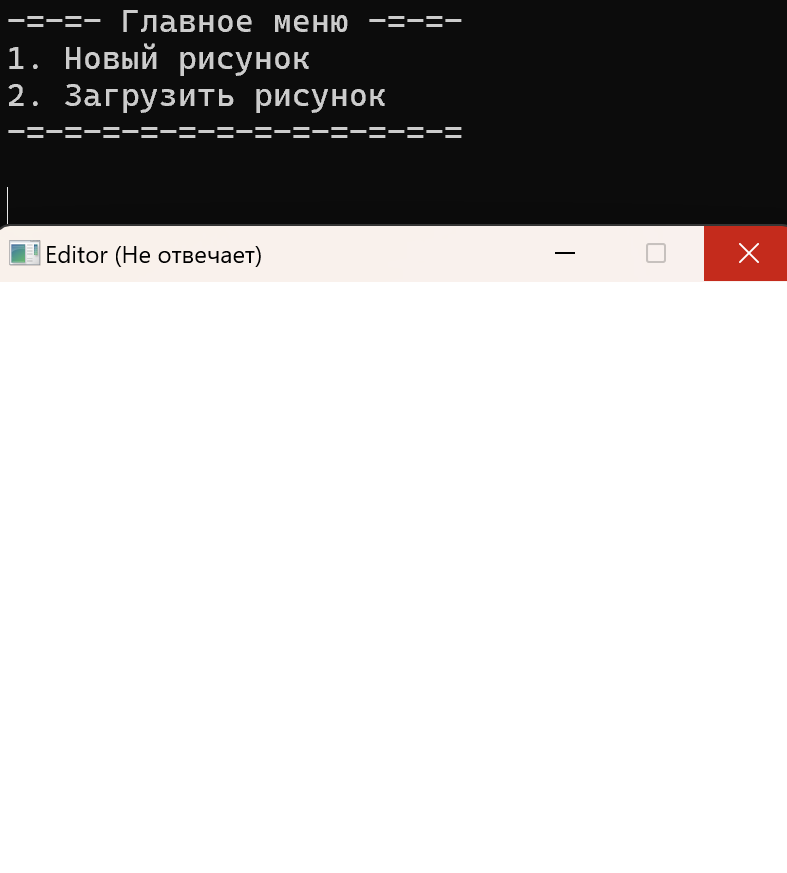


Рис. 1 – Главное меню

При выборе варианта работы программы с созданием нового рисунка пользователь сразу может начинать работу. При нажатии на левую кнопку мыши создаётся стежок – линия рисунка. С помощью стежков можно создать практически любой узор. Также при нажатии кнопок Q, W, E можно вывести информацию о рисунке в окно редактора. (Рис. 2)

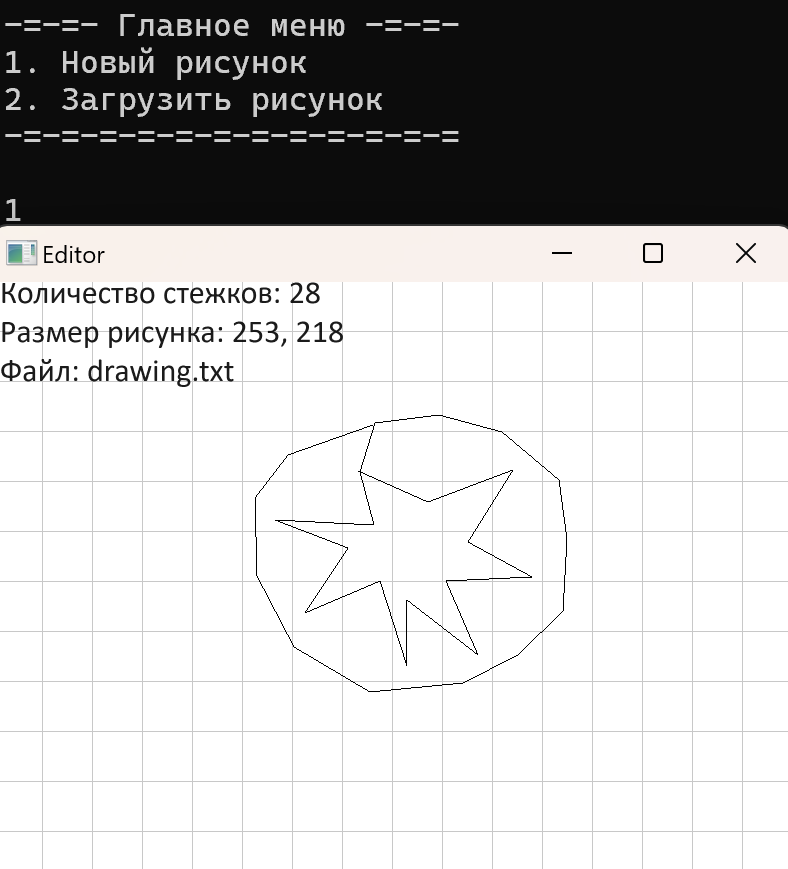


Рис. 2 – Создание рисунка

С помощью нажатия кнопки Z можно отменять ранее созданные стежки. Данное действие отменить нельзя. (Рис. 3)

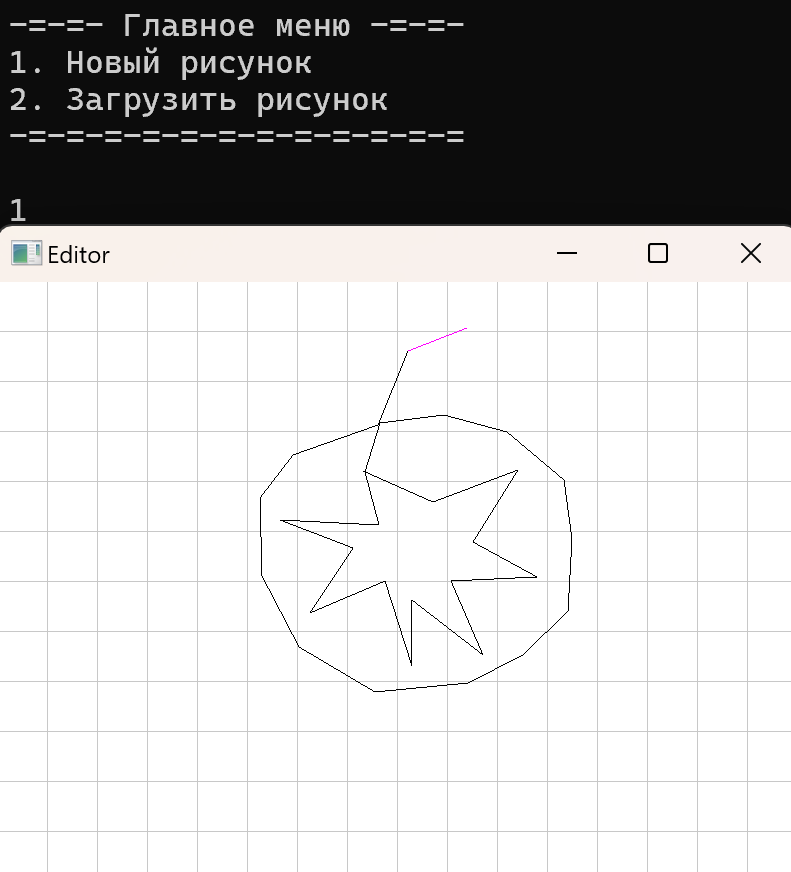


Рис. 3 – Отмена стежков

Чтобы сохранить рисунок необходимо нажать на кнопку M. Это также выведет список координат стежков в консоль. (Рис. 4)

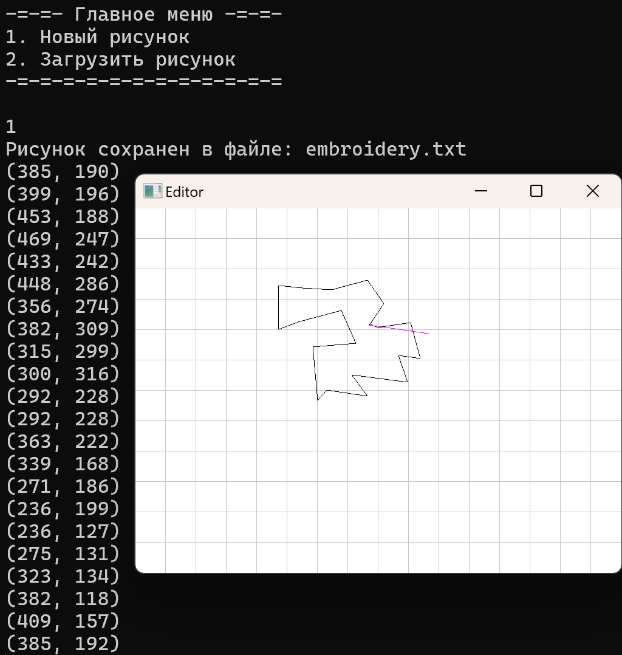


Рис. 4 – Сохранение

Также пользователь может загрузить в редактор уже готовый рисунок. Для этого в главном меню необходимо выбрать второй пункт. (Рис. 5)

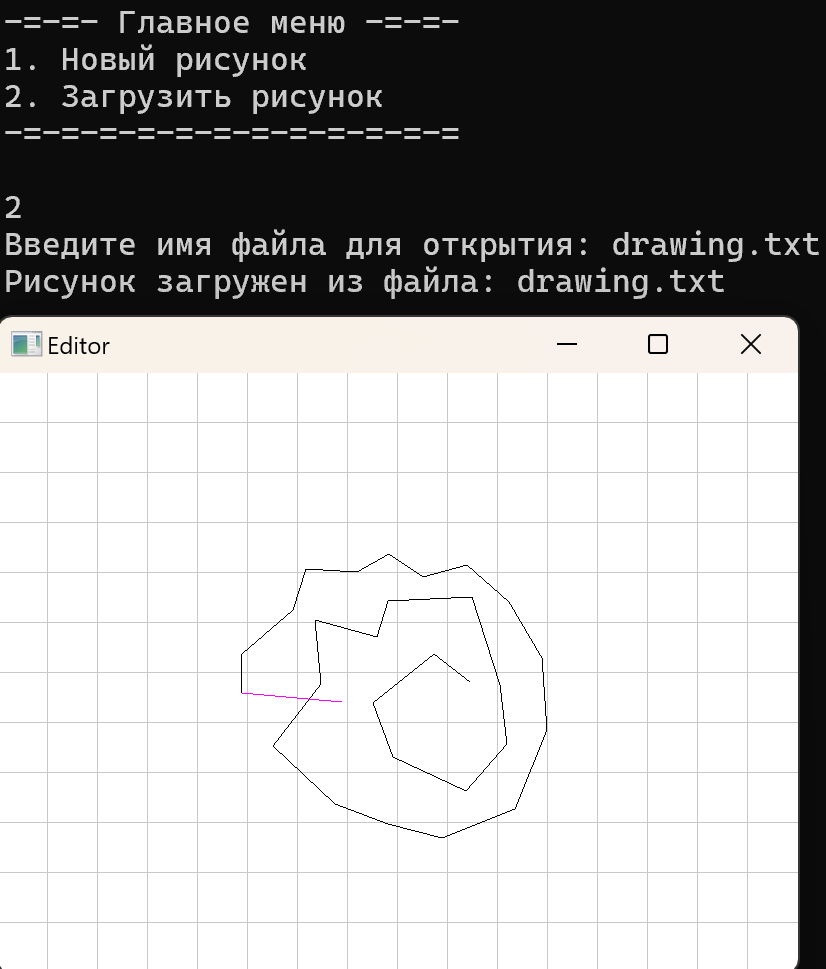


Рис. 5 – Загрузка рисунка

**Пути дальнейшего улучшения программы**

1. Улучшение визуальной части программы
2. Добавление возможности редактировать максимальную длину стежка