Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Операционные системы и системное программирование (ОСиСП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

Сетевая игра «Точки»

БГУИР КП I–40 01 01 318 ПЗ

Выполнил

студент: гр. 951003 Насанович Г.Б.

Проверил: Яскевич Д.А.

Минск 2021

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Лапицкая Н.В. 2021г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Насановичу Григорию Борисовичу*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы Сетевая игра «Точки»

2. Срок сдачи законченной работы 11.06.2021г.

3. Исходные данные к работе Возможность играть по сети.Правила игры “Точки”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Анализ литературных источников\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2 Моделирование предметной области \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3 Проектирование программного средства \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Тестирование и проверка работоспособности программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

\_*Приложение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма в формате А1*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе Яскевич Д.А.*.*

7.Дата выдачи задания *7.09.2021г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объема работы):

*Раздел 1. Введение к 15.09.2021г. – 15 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2, к 15.10.2021г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3, к 15.11.2021г. – 60% готовности работы*

*Раздел 4 к 15.12.2021 – 90 % готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*оформление пояснительной записки и графического материала к 01.06.2021*

*– 100 % готовности работы. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 13.12.2021г. по 17.12.2021г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Яскевич Д.А.

*(подпись)*

Задание принял к исполнению\_\_\_\_\_\_\_\_\_\_\_\_\_\_*Насанович Г.Б. 7.09.2021г.*

*(дата и подпись студента)*

Содержание

[Введение 5](#_Toc90394716)

[1 Анализ литературных источников 7](#_Toc90394717)

[1.1 Анализ существующих аналогов 7](#_Toc90394718)

[1.2 Постановка задачи 9](#_Toc90394719)

[2 Моделирование предметной области 10](#_Toc90394720)

[2.1 Теоретические сведения по теме 10](#_Toc90394721)

[3 Проектирование программного средства 11](#_Toc90394722)

[3.1 Общая информация 11](#_Toc90394723)

[3.2 Разработка алгоритма добавления новых точек 11](#_Toc90394724)

[3.3 Разработка алгоритма нахождения замкнутых контуров 13](#_Toc90394725)

[3.4 Разработка алгоритма нахождения уникальных замкнутых областей 15](#_Toc90394726)

[3.5 Разработка алгоритма обработки TCP сообщений 17](#_Toc90394727)

[4 Тестирование и проверка работоспособности программного средства 20](#_Toc90394728)

[4.1 Тестирование игрового процесса 20](#_Toc90394729)

[Заключение 22](#_Toc90394730)

[Список использованных источников 23](#_Toc90394731)

[Приложение А 24](#_Toc90394732)

Введение

Точки – логическая [настольная игра](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D1%82%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0) для двух человек, сочетающая позиционное стратегическое планирование с тактическим перебором вариантов. Ведётся на плоском игровом поле, расчерченном на клетки одинакового размера. Игроки поочерёдно ставят точки двух цветов в перекрестия линий. Цель – окружить точки соперника замыканием вокруг них непрерывной цепи своих точек.

Игра точки появилась в Советском Союзе в 1970-1980х годах и стала популярной, можно сказать, народной игрой. Возможно ее популярность была обусловлена тем обстоятельством, что для нее достаточно стандартного листа бумаги, разлинованного в клетку, и при этом игра несет в себе глубокие стратегические идеи.

Игра точки произошла от древней игры го, пришедшей к нам из Юго-Восточной Азии. В 1975-1976 в журнале "Наука и жизнь" был опубликован ряд статей про го. Авторами статей являлись Валерий Асташкин и Георгий Нилов. Помимо описания правил игры, авторы отметили ее сложность. Во многих городах страны начали появляться клубы любителей го, начали проводиться турниры. Однако для воспитанных на шахматной логике европейцев было тяжело понять смысл борьбы за территорию, что было главной целью го и люди начали просить изменить правила. Тогда в своих статьях авторы предложили упростить правила игры и сделать ее более понятной и доступной. Также авторы предложили играть в го на тетрадных листах бумаги в клетку.  Так появилась совсем новая игра, которая в народе стала называться «точки».

Основной целью игры точки стал захват точек соперника и защита собственных. Окружение точек соперника сплошной цепью рядом стоящих собственных точек приводило к закрашиванию области окружения, а очки за захваченные точки шли в копилку игроку. Захваченные точки больше не могли участвовать в игре. Размер захваченной территории не шел в зачет игры, как это было в го.

С появлением доступа к Интернету любители игры точки стали создавать программы, сайты и сообщества в сети. Это позволило размыть границы и объединить игроков, живущих на огромной территории. Сообщество игроков сформировало единые правила игры, создало теорию и обучение. Сам уровень игры значительно вырос. Если отдельным группам игроков было сложно выдумывать новые комбинации и распространять свой опыт, то большое интернет-сообщество быстро продуцировало и накапливало новые знания.

В нынешнее время Точки являются достаточно популярной игрой, в основном среди школьников и студентов, и быстро развиваются (создаются новые программы для игры, сайты, проводятся турниры и т.д.). Наиболее распространены точки в России, на Украине, в Белоруссии, Казахстане. Известны, но не так популярны в других бывших советских республиках. Относительно многочисленное и устойчивое сообщество игроков сформировалось в Польше. Некоторое время игра культивировалась в Чехии. Кроме того, немало игроков представляют страны, в которых имеются большие диаспоры выходцев из СССР, России и других постсоветских государств (Германия, США, Израиль).

В данной пояснительной записке подробно документирован процесс работы над курсовым проектом: приведены теоретические сведения из справочной литературы, обоснован выбор используемых средств разработки, дано описание основных моделей и алгоритма, приведены результаты тестирования, составлено руководство пользователя и подведены итоги курсовой работы.

# Анализ литературных источников

## Анализ существующих аналогов

Прежде чем приступить к разработке своего программного средства, были изучены схожие по задачам программы и приложения.

**Точки онлайн** – самое популярное бесплатное приложение для игры в точки на вашем смартфоне.

**Плюсы**:

* Возможность игры одному;
* Возможность игры с ещё одним пользователем;
* Возможность игры онлайн;
* Возможность выбрать размер поля и уровень сложности;
* Наличие инструкции по использованию;
* Возможность получать награды и участвовать в рейтинге

**Минусы:**

* Обилие рекламы;
* Долгая загрузка игрового поля

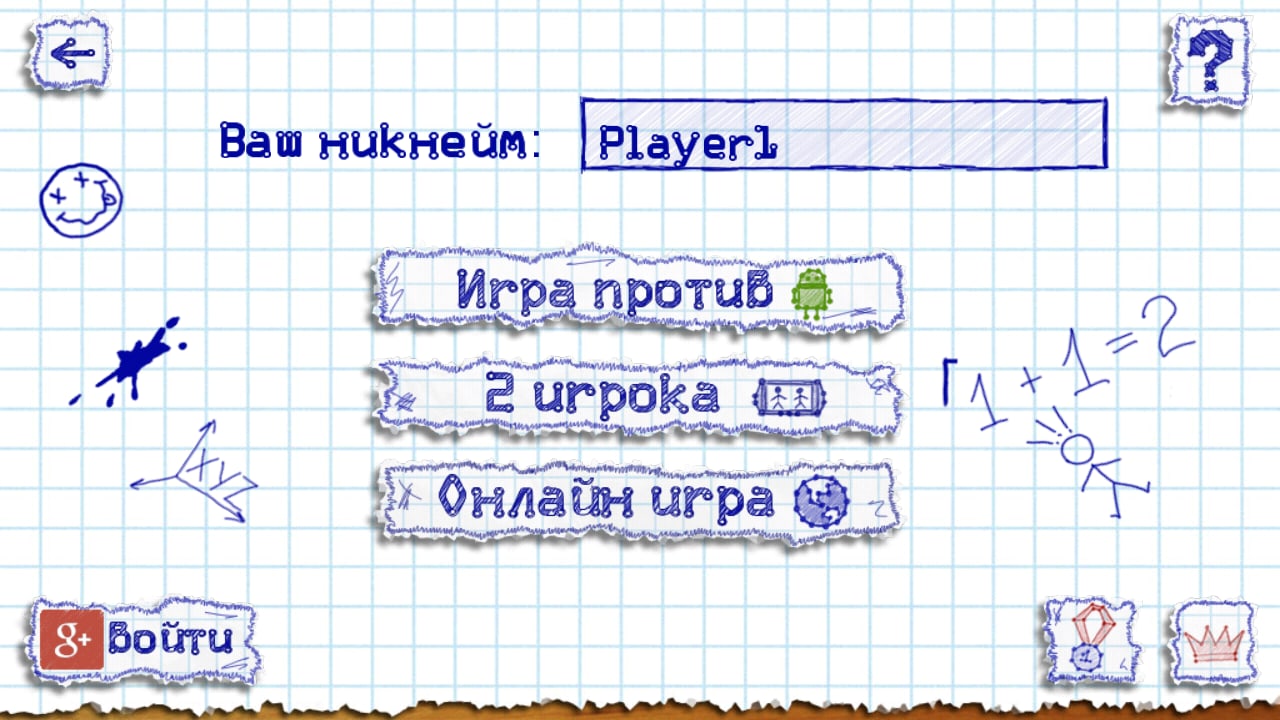


Рисунок 1.1 – Скриншот Точки онлайн

**notAgo** – второе по количеству скачиваний приложение для смартфона для игры в точки.

**Плюсы**:

* Возможность игры одному;
* Возможность игры с ещё одним пользователем;
* Обучение с пошаговой инструкцией;
* Бесконечное поле;
* Выбор уровня сложности при игре с машиной;
* Рейтинг при онлайн игре;
* Возможность игры онлайн;
* Игра онлайн без авторизации или с авторизацией;
* Чат для обсуждения;
* Удобный и понятный пользовательский интерфейс;

**Минусы**:

* Ограничение по количеству выйгранных точек, что ведёт к завершению игры;
* Неудобная работа на игровом поле;
* Большая загруженность ведёт к невозможности найти партнеров для игры

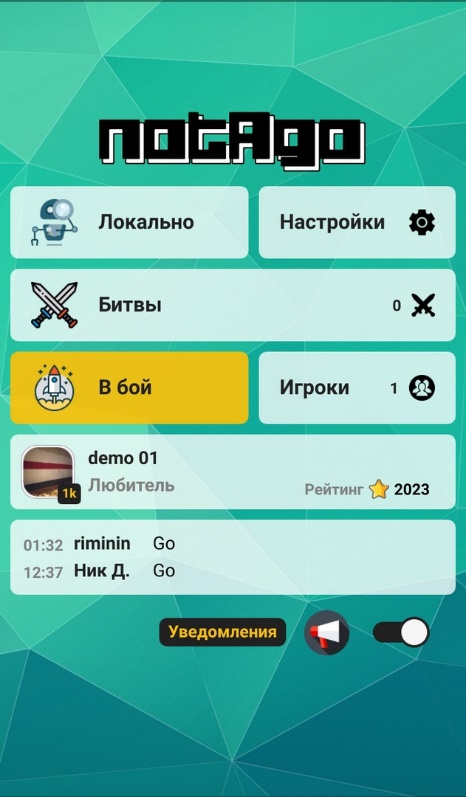


Рисунок 1.2 – Скриншот notAgo

**Cпортивные точки** – приложение с самым большим рейтингом из представленных приложений с игрой «Точки».

**Плюсы**:

* Полезные советы при загрузке;
* Игра онлайн без авторизации или с авторизацией;
* Обучение с подробной инструкцией и возможными заданиями для выполнения;
* Чат с обсуждениями;
* Рейтинг пользователей;
* Возможность просмотра сыгранных по дате игры и ещё незаконченных партий;
* Получение наград;
* Возможность предложить ничью или сдаться

**Минусы**:

* Долгая загрузка после входа в приложение;
* Неудобный интерфейс при игре

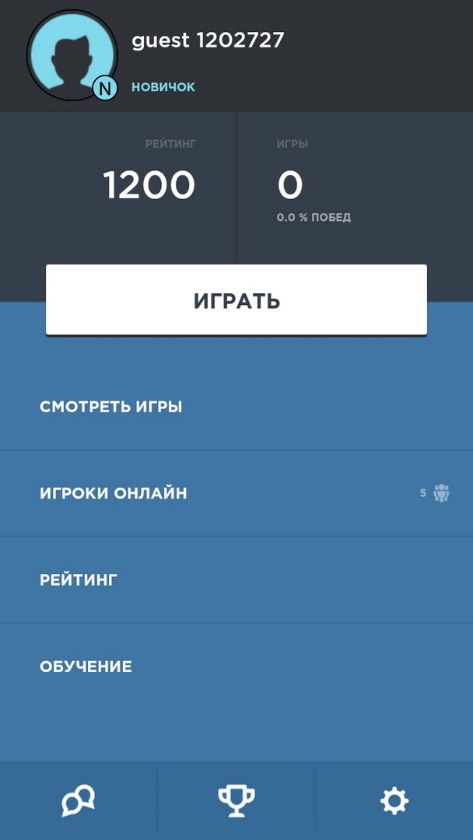
****

Рисунок 1.2 – Скриншот Спортивные точки

## Постановка задачи

Для максимально удобного взаимодействия пользователя с проектируемым программным средством должен быть создан информативный и понятный пользовательский интерфейс. Также ПС должно иметь достаточный функционал для обеспечения удобной работы.

Должны быть реализованы следующие функции:

* Возможность ставить точки в указанных полях;
* Смена игрока при выполнении хода;
* Подсчёт выигранных точек;
* Обводка выигранных точек;
* Возможность начать новую игру;
* Возможность игры по сети

# Моделирование предметной области

## Теоретические сведения по теме

Правила игры в точки похожи на [правила игры Го](http://www.gambiter.ru/go/item/17-pravila-igry-go.html), но более легки для понимания, поэтому игра является как бы "облегчённой" версией Го. Вместо камней используются точки разных цветов, наносимые на бумагу ручкой или карандашом. При игре карандашами одного цвета для различия противников наравне с точками используются крестики и нолики, либо крестики и точки. Эти особенности делают игру точки более доступной для широких масс людей.

Поле игры - лист бумаги в клетку любого выбранного размера.

Соперники по очереди ставят точки на пересечении линий листа в клетку, каждый своим цветом.

Часто используется начальная расстановка точек – по две точки разного цвета в центре игрового поля (аналогично [игре отелло-реверси](http://www.gambiter.ru/reversi/item/3-igra-reversi-pravila.html)) – для более активной игры.

Цель игры – окружить и взять в плен больше точек противника. Менее популярный вариант правил – захватить больше территории (данный вариант более близок к игре Го).

Окружение должно быть построено так, чтобы расстояние между точками составляло не более одной клетки - по прямой или по диагонали. Пленные точки в окружении участвовать не могут. Почти во всех вариантах игры территория без пленных точек противника не захватывается. Ставить точку в территорию, захваченную противником, запрещается либо эта точка считается захваченной, если не завершает собой окружения, разрывающего линию.

# Проектирование программного средства

## Общая информация

Для данного курсового проекта была выбрана модульная архитектура. Данная архитектура хорошо подходит для сложных программ, потому что она позволяет серьезно упростить процессы проектирования и создания конечного программного продукта. Модульная архитектура так же обеспечивает гибкость системы, дает возможность масштабировать систему и использовать принципы нисходящего проектирования.

Для передачи данных между пользователями были выбраны два протокола UDP и TCP. При помощи UDP запроса вновь подключенный пользователь отправляет в сеть UDP запрос, который содержит его IP адрес и игровое имя. При получении UDP запроса получатель создает TCP соединение с полученным IP адресом и отправляет по этому соединению свое игровое имя.

## Разработка алгоритма добавления новых точек

Для добавления новых точек необходимо получать информацию о позиционировании курсора и информация о игроке, который хочет поставить данную точку. Для удачного добавления точки должны выполнятся следующие требования: данная точка не должна принадлежать другому игроку и не должна быть закрашена.



Рисунок 3.1- Схема алгоритма добавления новых точек.

## Разработка алгоритма нахождения замкнутых контуров

Определение замкнутого контура начинается с последней добавленной точки. Переменная i задает положение следующей точки, после начальной. Чтобы найти все возможные замкнутые контуры необходимо проверить все близлежащий точки к последней поставленной. Если точка с i-тым смещением того же цвета, что и начальная, необходимо перейти к этой точке. После этого производится поиск еще не посещенных точек того же цвета, которые граничат с нашей новой начальной точкой. Алгоритм прекращает работать, когда рядом нет не посещенных точек такого цвета либо когда конечная точка и начальная точка совпали, то есть нашелся замкнутый контур. После завершения алгоритма, замкнутый контур добавляется для отрисовки.



Рисунок 3.2- Схема алгоритма нахождения замкнутых контуров.

## Разработка алгоритма нахождения уникальных замкнутых областей

Для проверки на уникальность замкнутой области необходимо пройти по всему списку уже замкнутых областей и проверить на соответствие добавляемую область на равенство им. Проверка начинается со сравнение длинны добавляемой области и уже существующей, если длинна различна, то эти области различны и можно переходить к следующей области для проверки, если длинна совпадает, то необходима проверить на соответствие каждую точку. Если точки хотя бы одна точка уникальна, то мы переходим к проверке следующего элемента списка, иначе- область не уникальна и проверка заканчивается. Проверка продолжается до того момента, когда все элементы списка проверены.



Рисунок 3.3- Схема алгоритма нахождения уникальных замкнутых областей

## Разработка алгоритма обработки TCP сообщений

Для передачи сообщений был разработан собственный формат запросов. В первом байте передается тип запроса, во втором – код сообщения, с третьего до последнего - передаваемый контент.

Для передачи имени и состояния пользователя был выбран первый тип сообщений. Получение кода 1 – код для подключения, код 2 – код для создания матча, 3 – код для отправки новой точки, 4 – код для окончания игры.

Сообщения, которые начинаются на 2, обозначают начало игры. Во втором байте этого сообщения хранится тип данных сообщений. Получение кода 0 обозначает, что пользователь отказывает в подключении, код 1 – код для создания нового матча, код 2 – положительный ответ на запрос о создании нового матча.

По коду 3 передаются данные о новых точка. Координаты по горизонтали хранятся в 2-3 байтах сообщения, координаты по вертикали – в 4-5 байтах.

Код 4 предназначен для окончания игры. Получение кода 1 обозначает, что пользователь сдался.



Рисунок 3.4- Схема алгоритма обработки TCP сообщений (часть 1)



Рисунок 3.5- Схема алгоритма обработки TCP сообщений (часть 2)

# Тестирование и проверка работоспособности программного средства

## Тестирование игрового процесса

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Тестируемая функциональность | Последовательность действий | Ожидаемый результат | Полученный результат |
| 1 | Правильность нахождения замкнутых контуров | 1. Создать замкнутый контур с вражескими точками внутри | Закрашивание области и добавления очков для закрашивающего | Тест пройден |
| 2 | 1. Создать замкнутый контур.  2. Обвести уже созданный контур точками другого цвета | Перекрашивание всей захваченной области. Уменьшение количества очков у другого игрока и зачисления их другому | Тест пройден |
| 3 | 1. Создать замкнутый контур вокруг своей точки | Никаких изменений в контурах | Тест пройден |
| 4 | 1. Создание двух контуров, замыкание в которых произойдет по добавлению одной точки  2. Добавление данной точки | Создание двух отдельных контуров | Тест пройден |
| 5 | Правильность сетевого взаимодействия | 1. Подключиться к другому игроку.  2. Завершить программу | Получение другим игроком сообщение о выходе | Тест пройден |

Таблица 1- Тестирование программного продукта

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 |  | 1. Попытаться подключиться к играющему человеку | Отказ подключения | Тест пройден |

# Заключение

Используя среду программирования Microsoft Visual Studio было разработано программное средство «Точки».

В ходе данной курсовой работы были проведены анализ поставленной задачи, исследование существующих аналогов, работа с литературой, разработка алгоритмов, экспериментальные исследования созданного приложения, составление руководства пользователя.

Все возможности программы, которые обсуждались при постановке задачи, предоставляются пользователям. При поиске решения программной реализации были использованы наиболее подходящие алгоритмы, структуры данных и методы программирования.

Были проведены многочисленные тесты, в ходе которых явные неисправности обнаружены не были. Количество тестов было достаточным, чтобы утверждать, что приложение имеет стабильную работоспособность.

На выходе было получено программное средство, реализующее следующие возможности:

* Возможность ставить точки в указанных полях
* Смена игрока при выполнении хода
* Подсчёт выйгранных точек
* Обводка выйгранных точек
* Возможность играть по сети

# Список использованных источников

[1] Сайт «Википедия» [Электронный ресурс]. – Электронные данные. – режим доступа: https://ru.wikipedia.org/цшлш/Точки\_(игра), свободный.

[2] Серебряная, Л.В. Марина, И.М. Структуры и алгоритмы обработки данных: учебно-метод. пособие для студ. спец. «Программное обеспечение информационных технологий» всех форм обуч. / Л. В. Серебряная, И. М. Марина. – Минск: БГУИР, 2012. – 49 с.: ил.

[3] ГОСТ 19.701–90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – Введ. 1992–01–01. – М.: Изд-во стандартов, 1991.

[4] Сайт MSDN- документация по операционной системе Microsoft Windows[Электронный ресурс]. Электронные данные- режим доступа: https://MSDN.com , свободный.

**Приложение А**

#include "PointInternet.h"

#include <Windows.h>

#include <windowsx.h>

#include "PlayGround.h"

#include <CommCtrl.h>

#include "Main.h"

#include "TCPIPList.h"

#include "stdlib.h"

typedef struct MainWindow {

HWND HListBox;

} \*PMainWindows;

const wchar\_t\* const WindowsClass = L"PointGame";

const wchar\_t\* const Child = L"Child";

PPointInternet Internet;

static HWND HListBox;

static HWND HEdit;

static HWND HButton;

static HWND HButtonStartGame;

static char BufferName[16];

LRESULT CALLBACK ChildProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM

lParam) {

switch (msg)

{

case(WM\_CREATE) :

{

HListBox=CreateWindow(WC\_LISTBOX, NULL,

WS\_CHILD | WS\_VISIBLE | LBS\_STANDARD |

LBS\_WANTKEYBOARDINPUT,

0, 0, 200, 200,

hwnd, (HMENU)ID\_LIST, NULL, NULL);

HEdit = CreateWindow(WC\_EDITW, NULL, WS\_CHILD |

WS\_VISIBLE|WS\_BORDER, 210, 0, 120, 20, hwnd,

NULL, NULL, NULL);

HButton = CreateWindow(WC\_BUTTON, L"Регистрация",

WS\_CHILD | WS\_VISIBLE, 210, 20, 120, 20, hwnd,

BUTTON\_ED\_ID, NULL, NULL);

HButtonStartGame = CreateWindow(WC\_BUTTON,

L"Начать игру", WS\_CHILD | WS\_VISIBLE, 210, 40,

120, 20, hwnd, BUTTON\_START\_GAME, NULL, NULL);

Button\_Enable(HButtonStartGame, FALSE);

memset(BufferName, 0, sizeof(BufferName));

break;

}

case(WM\_CLOSE):

case(WM\_DESTROY):

{

ShowWindow(hwnd, SW\_HIDE);

return 0;

}

case(WM\_KEYDOWN):

{

break;

}

case(WM\_COMMAND):

{

switch(wParam)

{

case BUTTON\_START\_GAME: {

int ChooseItem = SendMessage(HListBox,

LB\_GETCURSEL, 0, 0);

if (ChooseItem >= 0)

{

HWND parent = GetWindowLong(hwnd, 0);

SendMessage(parent, MESSAGE\_STARTGAME\_OUT,

ChooseItem, 0);

SendMessage(hwnd, WM\_DESTROY, 0, 0);

}

break;

}

case(EDIT\_ID): {

break;

}

case(BUTTON\_ED\_ID):

{

TCHAR\* Name=malloc(sizeof(TCHAR)\*MaxNameSize);

memset(Name, "\0", sizeof(TCHAR) \* MaxNameSize);

Edit\_GetText(HEdit, Name, sizeof(TCHAR) \*

MaxNameSize);

HWND parent = GetWindowLong(hwnd, 0);

SendMessage(parent, MESSAGE\_NEWNAME, Name,0);

Edit\_Enable(HEdit, FALSE);

Button\_Enable(HButton, FALSE);

Button\_Enable(HButtonStartGame, TRUE);

break;

}

default:

break;

}

break;

}

case WM\_SIZING:

{

break;

}

case WM\_GETMINMAXINFO:

{

LPMINMAXINFO lpMMI = (LPMINMAXINFO)lParam;

lpMMI->ptMinTrackSize.x = 350;

lpMMI->ptMinTrackSize.y = 235;

lpMMI->ptMaxTrackSize.x = 350;

lpMMI->ptMaxTrackSize.y = 235;

break;

}

case MESSAGE\_UPDATELIST: {

SendMessage(HListBox, LB\_RESETCONTENT, 0, 0);

PTCPList UserList = GetUsers(wParam);

PTCPConnect TCP;

int i = 0;

while (TCP= GetUserOFIndex(UserList, i))

{

SendMessage(HListBox, LB\_ADDSTRING, 0,

(TCHAR\*)TCP->PlayerName);

i++;

}

RemoveUsers(UserList);

}

default:

break;

}

return DefWindowProc(hwnd, msg, wParam, lParam);

}

HWND WINAPI CreateChildWindow(HWND hwndParent) {

WNDCLASSEX w;

memset(&w, 0, sizeof(WNDCLASSEX));

w.cbSize = sizeof(WNDCLASSEX);

w.cbWndExtra = 4;

w.cbClsExtra = 0;

w.style = 0;

w.lpfnWndProc = ChildProc;

w.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

int err = GetLastError();

w.lpszMenuName = NULL;

w.lpszClassName = L"ChildWClass";

w.hCursor = LoadCursor(0, IDC\_HAND);

RegisterClassEx(&w);

HWND child;

child = CreateWindowEx(0, L"ChildWClass", (LPCTSTR)NULL,

WS\_OVERLAPPEDWINDOW| WS\_MINIMIZEBOX, 0, 0,

400, 235, NULL, NULL, NULL, NULL);

SetWindowLong(child, 0, hwndParent);

ShowWindow(child, SW\_SHOWDEFAULT);

UpdateWindow(child);

return child;

}

LRESULT WINAPI MainWindowProcc(HWND hwnd,UINT msg,WPARAM wParam,LPARAM

lParam)

{

switch (msg)

{

case(WM\_CREATE):

{

PplayGround Play = GetWindowLong(hwnd, 0);

if (!Play)

{

HWND child= CreateChildWindow(hwnd);

Play = CreatePlayGround(hwnd);

SetWindowLong(hwnd, 0, Play);

SetWindowLong(hwnd, 4, child);

InvalidateRect(hwnd, NULL, FALSE);

}

SetFocus(GetWindowLong(hwnd, 4));

break;

}

case(WM\_PAINT):

{

PplayGround Play = GetWindowLong(hwnd, 0);

PlayGroundRedraw(Play);

int res=PlayGroundEndGame(Play);

if (res == 1)

{

PPointInternet Internet = GetWindowLong(hwnd, 8);

InternetSendMessage(Internet->ChooseUser, '3', 1);

Internet->ChooseUser == NULL;

}

break;

}

case WM\_COMMAND:

switch (wParam)

{

case MAIN\_WINDOW\_BUTTON\_START:

{

HWND ChildHandle = GetWindowLong(hwnd, 4);

ShowWindow(ChildHandle, SW\_SHOW);

int er = GetLastError();

UpdateWindow(ChildHandle);

SetFocus(ChildHandle);

break;

}

case MAIN\_WINDOW\_LEAVE\_GAME: {

HWND ChildHandle = GetWindowLong(hwnd, 4);

EnableWindow(hwnd, TRUE);

EnableWindow(ChildHandle, TRUE);

ShowWindow(ChildHandle, SW\_SHOW);

UpdateWindow(ChildHandle);

PlayGroundClean(GetWindowLong(hwnd, 0));

PPointInternet Internet = GetWindowLong(hwnd, 8);

if ((Internet != NULL) && (Internet->ChooseUser != NULL))

{

InternetSendMessage(Internet->ChooseUser, "41", 2);

}

break;

}

}

break;

// Обработка сообщений от нажатий пользователя на этом компьютере

//

//

case(WM\_LBUTTONDOWN):

{

InvalidateRect(hwnd, NULL, FALSE);

PPointInternet Internet = GetWindowLong(hwnd, 8);

if (Internet == NULL)

break;

if (Internet->ChooseUser == NULL)

break;

PplayGround Play = GetWindowLong(hwnd, 0);

if (Play->YouStep == 0)

break;

int res;

if (!Play->EndGame)

{

res=PlayGroundAddPoint(Play, lParam);

InvalidateRect(hwnd, NULL, FALSE);

}

char DGram[16] = { 0 };

DGram[0] = 3;

DGram[1] = res >> 24;

DGram[2] = res >> 16;

DGram[3] = res >> 8;

DGram[4] = res >> 0;

InternetSendMessage(Internet->ChooseUser, DGram, 5);

break;

}

case(WM\_SIZING):

{

/\*PplayGround Play = GetWindowLong(hwnd, 0);

PlayGroundResize(Play);

InvalidateRect(hwnd, NULL, FALSE);\*/

break;

}

case WM\_GETMINMAXINFO:

{

LPMINMAXINFO lpMMI = (LPMINMAXINFO)lParam;

lpMMI->ptMinTrackSize.x = MINIMAL\_WIDTH\_WINDOW;

lpMMI->ptMinTrackSize.y = MINIMAL\_HEIGHT\_WINDOW;

lpMMI->ptMaxTrackSize.x = MINIMAL\_WIDTH\_WINDOW;

lpMMI->ptMaxTrackSize.y = MINIMAL\_HEIGHT\_WINDOW;

break;

}

case WM\_SETFOCUS: {

break;

}

case(WM\_DESTROY):

{

PostQuitMessage(0);

break;

}

case(WM\_KEYDOWN): {

return 0;

break;

}

case MESSAGE\_CONNECTION\_ADDPOINT: {

PplayGround Play = GetWindowLong(hwnd, 0);

PlayGroundAddPointInt(Play, wParam);

InvalidateRect(hwnd, NULL, FALSE);

break;

}

case MESSAGE\_NEWNAME: {

wchar\_t\* Name = wParam;

PPointInternet Internet = CreateInternetConnection(hwnd, Name);

SetWindowLong(hwnd, 8, Internet);

UDPSendName(GetWindowLong(hwnd, 8));

break;

}

case MESSAGE\_UPDATELIST: {

HWND child = GetWindowLong(hwnd, 4);

SendMessage(child, MESSAGE\_UPDATELIST, wParam, lParam);

break;

}

case MESSAGE\_STARTGAME\_OUT: {

PPointInternet Internet = GetWindowLong(hwnd, 8);

PTCPConnect Connection = GetIP(Internet->TCPList, wParam);

Internet->ChooseUser = Connection;

InternetSendMessage(Connection, "21", 2);

PplayGround Play = GetWindowLong(hwnd, 0);

Play->YouStep = 1;

EnableWindow(hwnd, FALSE);

EnableWindow(GetWindowLong(hwnd, 4), FALSE);

int res = MessageBox(hwnd, L"Ожидание ответа пользователя",

L"Ожидание ответа пользователя", MB\_OKCANCEL |

MB\_ICONINFORMATION);

if (res == IDOK)

{

break;

}

else {

InternetSendMessage(Connection, "20", 2);

Internet->ChooseUser = NULL;

Play->YouStep = 0;

}

EnableWindow(hwnd, TRUE);

EnableWindow(GetWindowLong(hwnd, 4), TRUE);

break;

}

case MESSAGE\_CONNECTION\_SUCCESS: {

PPointInternet Internet = GetWindowLong(hwnd, 8);

MessageBox(hwnd, L"Игра начата", L"игра начата", MB\_OK |

MB\_ICONINFORMATION);

Internet->ChooseUser = wParam;

EnableWindow(hwnd, TRUE);

EnableWindow(GetWindowLong(hwnd, 4), TRUE);

SendMessageW(GetWindowLong(hwnd, 4), WM\_CLOSE, 0, 0);

break;

}

case MESSAGE\_CONNECTION\_CLOSE: {

PPointInternet Internet = GetWindowLong(hwnd, 8);

Internet->ChooseUser = NULL;

EnableWindow(hwnd, TRUE);

EnableWindow(GetWindowLong(hwnd, 4), TRUE);

break;

}

case MESSAGE\_CONNECTION\_END: {

PPointInternet Internet = GetWindowLong(hwnd, 8);

Internet->ChooseUser = NULL;

break;

}

case MESSAGE\_STARTGAME\_IN:

{

PTCPConnect Connection = wParam;

PPointInternet Internet = GetWindowLong(hwnd, 8);

if (Internet->ChooseUser != NULL)

InternetSendMessage(Connection, "20", 2);

EnableWindow(hwnd, FALSE);

EnableWindow(GetWindowLong(hwnd, 4), FALSE);

int res = MessageBox(hwnd, Connection->PlayerName, L"Принять

приглашение?", MB\_YESNO | MB\_ICONINFORMATION);

if (res == IDYES)

{

PplayGround Play = GetWindowLong(hwnd, 0);

Play->YouStep = 0;

Internet->ChooseUser = Connection;

InternetSendMessage(Connection, "22", 2);

SendMessage(hwnd,MESSAGE\_CONNECTION\_SUCCESS,wParam,0);

}

else

{

InternetSendMessage(Connection, "20", 2);

EnableWindow(hwnd, TRUE);

EnableWindow(GetWindowLong(hwnd, 4), TRUE);

}

break;

}

case MESSAGE\_CONNECTION\_FAILD:

{

PPointInternet Internet = GetWindowLong(hwnd, 8);

PTCPConnect Connection = (PTCPConnect)wParam;

if (Internet->ChooseUser!=NULL)

if (Internet->ChooseUser->TCPThread == Connection-

>TCPThread)

{

Internet->ChooseUser = NULL;

}

EnableWindow(hwnd, TRUE);

EnableWindow(GetWindowLong(hwnd, 4), TRUE);

break;

}

case MESSAGE\_CONNECTION\_LEAVE: {

MessageBox(hwnd, L"Пользователь покинул игру", L"Ответ

пользователя", MB\_OK | MB\_ICONINFORMATION);

PPointInternet Internet = GetWindowLong(hwnd, 8);

Internet->ChooseUser = NULL;

PplayGround Play= GetWindowLong(hwnd, 0);

PlayGroundClean(Play);

InvalidateRect(hwnd, NULL, FALSE);

}

}

return DefWindowProc(hwnd,msg,wParam, lParam);

}

INT WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR

lpCmdLine, INT nCmdShow)

{

WNDCLASSEX wcex;

MSG msg;

HWND Mainhwnd;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = 0;

wcex.lpfnWndProc = MainWindowProcc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 12;

wcex.hInstance = hInstance;

wcex.hIcon = 0;

wcex.hCursor = LoadCursor(0, IDC\_HAND);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = NULL;

wcex.lpszClassName = WindowsClass;

wcex.hIconSm = 0;

RegisterClassEx(&wcex);

Mainhwnd = CreateWindowEx(0, WindowsClass, L"Игра точки",

(WS\_OVERLAPPEDWINDOW | WS\_VISIBLE), 0, 0, MINIMAL\_WIDTH\_WINDOW, MINIMAL\_HEIGHT\_WINDOW, 0, 0, hInstance, NULL);

HWND HBStart = CreateWindowEx(0, L"BUTTON", L"Список игроков",

(WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON), MINIMAL\_WIDTH\_WINDOW - RIGHT\_PLACE, 0, 120, 30, Mainhwnd, (HMENU)MAIN\_WINDOW\_BUTTON\_START, 0, NULL);

HWND HBLeaveGame = CreateWindowEx(0, L"BUTTON", L"Покинуть игру",

(WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON), MINIMAL\_WIDTH\_WINDOW - RIGHT\_PLACE, 40, 120, 30, Mainhwnd, (HMENU)MAIN\_WINDOW\_LEAVE\_GAME, 0, NULL);

while (GetMessage(&msg, 0, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

#include "PlayGround.h"

#include <Windows.h>

#include <windowsx.h>

#include "Main.h"

#define BLUE 1

#define RED 2

#define CLEANTEXT " "

#define STEPRED "Ходит красный"

#define STEPBLUE "Ходит синий"

#define REDCOUNT "Красные точки -"

#define BLUECOUNT "Синие точки -"

#define RIGHTCOLUMN RIGHT\_PLACE

#pragma warning(disable : 4996)

int PlayGroundClean(PplayGround Play)

{

Play->CountBlue = 0;

Play->CountRed = 0;

Play->EndGame = 0;

for (int i = 0; i < Play->WidthPoint; i++)

{

for (int j = 0; j < Play->HeightPoint; j++)

{

Play->PointArr[i \* Play->HeightPoint + j].brush = 0;

Play->PointArr[i \* Play->HeightPoint + j].color = 0;

Play->PointArr[i \* Play->HeightPoint + j].view = 0;

}

}

Play->YouStep = 0;

PlayGroundBackDraw(Play);

}

PplayGround CreatePlayGround(HWND hwnd)

{

PplayGround Play = (PplayGround)HeapAlloc(GetProcessHeap(), HEAP\_ZERO\_MEMORY, sizeof(struct \_playGround));

Play->hwndSelf = hwnd;

Play->player.blue = CreateSolidBrush(0xff0000);

Play->player.red = CreateSolidBrush(0x0000ff);

Play->Heap = HeapCreate(HEAP\_GENERATE\_EXCEPTIONS, 0x0f0000, 0);

Play->WidthPoint = MINIMALWIDTH;

Play->HeightPoint = MINIMALHEIGHT;

RECT rc;

GetWindowRect(hwnd, &rc);

Play->hdcBack = CreateCompatibleDC(GetDC(hwnd));

Play->hbmBack = CreateBitmap(rc.right- RIGHTCOLUMN, rc.bottom-20, 1, 32, NULL);

SelectObject(Play->hdcBack, Play->hbmBack);

rc.right -= RIGHTCOLUMN;

rc.bottom -= 25;

Play->playGroundsize=rc;

PlayGroundCalculatePlace(Play);

PlayGroundBackDraw(Play);

return Play;

}

int PlayGroundResize(PplayGround Play)

{

RECT RC;

GetWindowRect(Play->hwndSelf, &RC);

RC.right -= RIGHTCOLUMN;

RC.bottom -= 25;

DeleteObject(Play->hbmBack);

Play->hbmBack = CreateBitmap(RC.right, RC.bottom, 1, 32, NULL);

SelectObject(Play->hdcBack, Play->hbmBack);

Play->playGroundsize = RC;

PlayGroundCalculatePlace(Play);

PlayGroundBackDraw(Play);

}

int PlayGroundBackDraw(PplayGround Play)

{

int r = r = Play->rad;

//HBRUSH br = CreateHatchBrush(HS\_FDIAGONAL, Play->player.red);

HBRUSH br = CreateSolidBrush(0xFFFFFF);

SelectObject(Play->hdcBack, br);

FillRect(Play->hdcBack, &Play->playGroundsize, br);

HBRUSH brush = CreateSolidBrush(0xFFFFFF);

HPEN pen= CreatePen(PS\_SOLID, 1, 0x000000);

HPEN lpen = SelectObject(Play->hdcBack, pen);

int i = 0;

while ((i) < (Play->WidthPoint))

{

MoveToEx(Play->hdcBack, Play->PointArr[i \* Play->HeightPoint].point.x, Play->PointArr[i \* Play->HeightPoint].point.y, NULL);

LineTo(Play->hdcBack, Play->PointArr[(i + 1) \* Play->HeightPoint - 1].point.x, Play->PointArr[(i + 1) \* Play->HeightPoint - 1].point.y);

i++;

}

int j = 0;

while (j < Play->HeightPoint)

{

MoveToEx(Play->hdcBack, Play->PointArr[j].point.x, Play->PointArr[j].point.y, NULL);

LineTo(Play->hdcBack, Play->PointArr[(i - 1) \* Play->HeightPoint + j].point.x, Play->PointArr[(i - 1) \* Play->HeightPoint + j].point.y);

j++;

}

i = 0;

while (i < Play->WidthPoint)

{

j = 0;

while (j < Play->HeightPoint)

{

SelectObject(Play->hdcBack, brush);

if (Play->PointArr[i \* Play->HeightPoint + j].color == RED)

SelectObject(Play->hdcBack, Play->player.red);

if (Play->PointArr[i \* Play->HeightPoint + j].color == BLUE)

SelectObject(Play->hdcBack, Play->player.blue);

Ellipse(Play->hdcBack, Play->PointArr[i \* Play->HeightPoint + j].point.x - r, Play->PointArr[(i)\*Play->HeightPoint + j].point.y - r, Play->PointArr[i \* Play->HeightPoint + j].point.x + r, Play->PointArr[(i)\*Play->HeightPoint + j].point.y + r);

j++;

}

i++;

}

PSPISOK Psp = Play->spis;

while (Psp != NULL)

{

if (Psp->color == RED)

{

SelectObject(Play->hdcBack, Play->player.red);

}

if (Psp->color == BLUE)

SelectObject(Play->hdcBack, Play->player.blue);

Polygon(Play->hdcBack, Psp->ArrOfPoint, Psp->CountPoint);

Psp = Psp->NextPoint;

}

SelectObject(Play->hdcBack, lpen);

DeleteObject(pen);

DeleteObject(brush);

}

int PlayGroundCalculatePlace(PplayGround Play)

{

int i = 0, j = 0;

int Size\_x, Size\_y;

Size\_x = (int)(Play->playGroundsize.right / Play->WidthPoint);

Size\_y = (int)(Play->playGroundsize.bottom / Play->HeightPoint);

if (Size\_x > Size\_y)

Play->rad = (int)(Size\_x / 5);

else

Play->rad = (int)(Size\_y / 5);

while (i < Play->WidthPoint)

{

j = 0;

while (j < Play->HeightPoint)

{

// gпоиграться с этими значениями, чтобы не было выхода за границы экрана, а то не учитываются радиусы в конечном итоге

Play->PointArr[i \* Play->HeightPoint + j].point.x = i \* Size\_x + Play->rad \* 2;

Play->PointArr[i \* Play->HeightPoint + j].point.y = j \* Size\_y + Play->rad \* 2;

//Play->PointArr[i \* Play->HeightPoint + j].color = 0;

//Play->PointArr[i \* Play->HeightPoint + j].brush = 0;

j++;

}

i++;

}

}

int PlayGroundCalculatePoint(PplayGround Play)

{

int i = 0, j = 0;

int count = 0;

Play->CountRed = Play->CountBlue = 0;

while (i < Play->WidthPoint)

{

j = 0;

while (j < Play->HeightPoint)

{

if (Play->PointArr[i \* Play->HeightPoint + j].brush == RED)

Play->CountRed++;

if (Play->PointArr[i \* Play->HeightPoint + j].brush == BLUE)

Play->CountBlue++;

j++;

}

i++;

}

}

PSPISOK PlayGroundCleanSpis(PplayGround Play)

{

PSPISOK temp = Play->spis;

while (temp != NULL)

{

PSPISOK tmp = temp;

temp = temp->NextPoint;

HeapFree(Play->Heap, 0, tmp->ArrOfPoint);

HeapFree(Play->Heap, 0, tmp);

}

return NULL;

}

int PlayGroundRedraw(PplayGround Play)

{

RECT rc;

GetWindowRect(Play->hwndSelf, &rc);

rc.right -= RIGHTCOLUMN;

PAINTSTRUCT ps;

HDC windowhdc = BeginPaint(Play->hwndSelf,&ps);

BitBlt(windowhdc, rc.left, rc.top, rc.right, rc.bottom, Play->hdcBack, 0, 0, SRCCOPY);

if (Play->player.color == BLUE)

{

TextOutA(windowhdc, MINIMAL\_WIDTH\_WINDOW - 180, MINIMAL\_HEIGHT\_WINDOW - 70, CLEANTEXT, strlen(CLEANTEXT));

TextOutA(windowhdc, MINIMAL\_WIDTH\_WINDOW - 180, MINIMAL\_HEIGHT\_WINDOW - 70, STEPBLUE, strlen(STEPBLUE));

}

if (Play->player.color == RED)

{

TextOutA(windowhdc, MINIMAL\_WIDTH\_WINDOW - 180, MINIMAL\_HEIGHT\_WINDOW - 70, STEPRED, strlen(STEPRED));

}

if (Play->player.color)

{

char text1[3] = { '\0' };

itoa(Play->CountRed, text1, 10);

TextOutA(windowhdc, MINIMAL\_WIDTH\_WINDOW - 180, MINIMAL\_HEIGHT\_WINDOW - 100, REDCOUNT, strlen(REDCOUNT));

TextOutA(windowhdc, MINIMAL\_WIDTH\_WINDOW - 50, MINIMAL\_HEIGHT\_WINDOW - 100, text1, 3);

char text2[3] = { '\0' };

itoa(Play->CountBlue, text2, 10);

TextOutA(windowhdc, MINIMAL\_WIDTH\_WINDOW - 180, MINIMAL\_HEIGHT\_WINDOW - 120, BLUECOUNT, strlen(BLUECOUNT));

TextOutA(windowhdc, MINIMAL\_WIDTH\_WINDOW - 50, MINIMAL\_HEIGHT\_WINDOW - 120, text2, 3);

}

EndPaint(Play->hwndSelf, &ps);

}

int PlayGroundBrush(PplayGround Play, int color)

{

int i = 0, j = 0;

int brush = 0, count = 0, ResultWertical = 0, ResultHorizontal = 0;

while (i < Play->WidthPoint)

{

j = 0;

count = 0;

while (j < Play->HeightPoint)

{

if (Play->PointArr[i \* Play->HeightPoint + j].view == 1)

{

count++;

}

j++;

}

j = 0;

brush = 0;

while ((j < Play->HeightPoint) && (count > 0))

{

if ((brush) && (Play->PointArr[i \* Play->HeightPoint + j].view != 1) && (Play->PointArr[i \* Play->HeightPoint + j].brush != color))

{

ResultWertical++;

}

if ((Play->PointArr[i \* Play->HeightPoint + j].view == 1))

{

brush = !brush;

count--;

}

j++;

}

i++;

}

j = 0;

while (j < Play->HeightPoint)

{

i = 0;

count = 0;

while (i < Play->WidthPoint)

{

if (Play->PointArr[i \* Play->HeightPoint + j].view == 1)

{

count++;

}

i++;

}

i = 0;

brush = 0;

while ((i < Play->WidthPoint) && (count > 0))

{

if ((brush) && (Play->PointArr[i \* Play->HeightPoint + j].view != 1) && (Play->PointArr[i \* Play->HeightPoint + j].brush != color))

{

ResultHorizontal++;

}

if ((Play->PointArr[i \* Play->HeightPoint + j].view == 1))

{

brush = !brush;

count--;

}

i++;

}

j++;

}

if (ResultWertical == 0)

return 0;

if (ResultHorizontal == 0)

return 0;

if ((ResultWertical - ResultHorizontal) != 0)

return 0;

i = 0;

while (i < Play->WidthPoint)

{

j = 0;

count = 0;

while (j < Play->HeightPoint)

{

if (Play->PointArr[i \* Play->HeightPoint + j].view == 1)

{

count++;

}

j++;

}

j = 0;

brush = 0;

while ((j < Play->HeightPoint) && (count > 0))

{

if ((brush) && (Play->PointArr[i \* Play->HeightPoint + j].view != 1) && (Play->PointArr[i \* Play->HeightPoint + j].brush != color))

{

Play->PointArr[i \* Play->HeightPoint + j].brush = color;

}

if ((Play->PointArr[i \* Play->HeightPoint + j].view == 1))

{

Play->PointArr[i \* Play->HeightPoint + j].view = 0;

brush = !brush;

count--;

}

j++;

}

i++;

}

return 1;

}

int PlayGroundIsUniq(PplayGround Play, PPOINT point, int count, int color)

{

PSPISOK psp = Play->spis;

PPOINT pp;

int Max\_x = 0, Max\_y = 0, Min\_x = MAXINT, Min\_y = MAXINT;

int i, tmp = 1;

if (count < 4)

return 0;

while (psp != NULL)

{

tmp = 0;

i = 0;

if (psp->CountPoint != count)

tmp = 1;

else

{

pp = psp->ArrOfPoint;

while (((point + i)->x == (pp + i)->x) && ((point + i)->y == (pp + i)->y) && (i <= count))

{

i++;

}

if (i != count)

tmp = 1;

}

psp = psp->NextPoint;

}

if (!tmp)

return 0;

return PlayGroundBrush(Play, color);

}

int PlayGroundTryBrush(PplayGround Play, int i, int j)

{

int Direction = 0, d1 = 0, loop1 = 0, d2;

int Near\_x = 0;

int StartRedraw = 0;

int res = 0;

int Near\_y = 0;

PPOINT PArrOfPoint, SavePoint = NULL;

DWORD err;

int CountPoint = 0;

int NewPP;

while (Direction < 8)

{

int index = 0;

while (index < Play->HeightPoint \* Play->WidthPoint)

{

Play->PointArr[index].view = 0;

index++;

}

index = 0;

/\*

тут не хватает проверок на выход за границы массива,в принципе это не вызывает прямо капитальных проблем, но надо пофиксить.

ошибки в индексации особенно заметны при проставлении точек в первой и последней строке массива

\*/

switch (Direction)

{

case 0:

if (Play->PointArr[i \* Play->HeightPoint + j].color != Play->PointArr[(i)\*Play->HeightPoint + j + 1].color)

{

Near\_x = i;

Near\_y = j + 1;

loop1 = 1;

d1 = Direction;

d2 = Direction + 1;

}

break;

case 1:

if (Play->PointArr[i \* Play->HeightPoint + j].color != Play->PointArr[(i + 1) \* Play->HeightPoint + j + 1].color)

{

Near\_x = i + 1;

Near\_y = j + 1;

loop1 = 1;

d1 = Direction;

d2 = Direction + 1;

}

break;

case 2:

if (Play->PointArr[i \* Play->HeightPoint + j].color != Play->PointArr[(i + 1) \* Play->HeightPoint + j].color)

{

Near\_x = i + 1;

Near\_y = j;

loop1 = 1;

d1 = Direction;

d2 = Direction + 1;

loop1 = 1;

}

break;

case 3:

if (Play->PointArr[i \* Play->HeightPoint + j].color != Play->PointArr[(i + 1) \* Play->HeightPoint + j - 1].color)

{

Near\_x = i + 1;

Near\_y = j - 1;

loop1 = 1;

d1 = Direction;

d2 = Direction + 1;

loop1 = 1;

}

break;

case 4:

if (Play->PointArr[i \* Play->HeightPoint + j].color != Play->PointArr[(i)\*Play->HeightPoint + j - 1].color)

{

Near\_x = i;

Near\_y = j - 1;

loop1 = 1;

d1 = Direction;

d2 = Direction + 1;

loop1 = 1;

}

break;

case 5:

if (Play->PointArr[i \* Play->HeightPoint + j].color != Play->PointArr[(i - 1) \* Play->HeightPoint + j - 1].color)

{

Near\_x = i - 1;

Near\_y = j - 1;

loop1 = 1;

d1 = Direction;

d2 = Direction + 1;

loop1 = 1;

}

break;

case 6:

if (Play->PointArr[i \* Play->HeightPoint + j].color != Play->PointArr[(i - 1) \* Play->HeightPoint + j].color)

{

Near\_x = i - 1;

Near\_y = j;

loop1 = 1;

d1 = Direction;

d2 = Direction + 1;

loop1 = 1;

}

break;

case 7:

if (Play->PointArr[i \* Play->HeightPoint + j].color != Play->PointArr[(i - 1) \* Play->HeightPoint + j + 1].color)

{

Near\_x = i - 1;

Near\_y = j + 1;

loop1 = 1;

d1 = Direction;

d2 = 0;

loop1 = 1;

}

break;

}

if (loop1)

{

CountPoint = 0;

CountPoint++;

NewPP = 1;

SavePoint = PArrOfPoint = (PPOINT)HeapAlloc(Play->Heap, HEAP\_ZERO\_MEMORY, CountPoint \* sizeof(POINT));

(PArrOfPoint + CountPoint - 1)->x = Play->PointArr[i \* Play->HeightPoint + j].point.x;

(PArrOfPoint + CountPoint - 1)->y = Play->PointArr[i \* Play->HeightPoint + j].point.y;

Play->PointArr[i \* Play->HeightPoint + j].view = 1;

}

while (loop1)

{

switch (d2)

{

case 0:

//

if ((Play->PointArr[(Near\_x + 1) \* Play->HeightPoint + Near\_y].color == Play->PointArr[i \* Play->HeightPoint + j].color) && (!Play->PointArr[(Near\_x + 1) \* Play->HeightPoint + Near\_y].view) && (Play->PointArr[(Near\_x + 1) \* Play->HeightPoint + Near\_y].brush == 0))

{

d1 = d2 = 6;

Play->PointArr[(Near\_x + 1) \* Play->HeightPoint + Near\_y].view = 1;

CountPoint++;

PArrOfPoint = SavePoint = (PPOINT)HeapReAlloc(Play->Heap, 0, SavePoint, CountPoint \* sizeof(POINT));

\*(PArrOfPoint + CountPoint - 1) = Play->PointArr[(Near\_x + 1) \* Play->HeightPoint + Near\_y].point;

}

else

{

if ((Near\_x == i) && (Near\_y == j))

StartRedraw = 1;

else

Near\_x++;

}

break;

case 1:

//

if ((Play->PointArr[(Near\_x + 1) \* Play->HeightPoint + Near\_y].color == Play->PointArr[i \* Play->HeightPoint + j].color) && (!Play->PointArr[(Near\_x + 1) \* Play->HeightPoint + Near\_y].view) && (Play->PointArr[(Near\_x + 1) \* Play->HeightPoint + Near\_y].brush == 0))

{

Play->PointArr[(Near\_x + 1) \* Play->HeightPoint + Near\_y].view = 1;

d2 = 6;

d1 = 6;

CountPoint++;

SavePoint = (PPOINT)HeapReAlloc(Play->Heap, 0, SavePoint, CountPoint \* sizeof(POINT));

PArrOfPoint = SavePoint;

\*(PArrOfPoint + CountPoint - 1) = Play->PointArr[(Near\_x + 1) \* Play->HeightPoint + Near\_y].point;

}

else

{

if ((Near\_x == i) && (Near\_y == j))

StartRedraw = 1;

else

Near\_x++;

}

break;

case 2:

//

if ((Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y - 1].color == Play->PointArr[i \* Play->HeightPoint + j].color) && (!Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y - 1].view) && (Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y - 1].brush == 0))

{

Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y - 1].view = 1;

d2 = 0;

d1 = 0;

CountPoint++;

SavePoint = (PPOINT)HeapReAlloc(Play->Heap, 0, SavePoint, CountPoint \* sizeof(POINT));

PArrOfPoint = SavePoint;

\*(PArrOfPoint + CountPoint - 1) = Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y - 1].point;

}

else

{

if ((Near\_x == i) && (Near\_y == j))

StartRedraw = 1;

else

Near\_y--;

}

break;

case 3:

if ((Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y - 1].color == Play->PointArr[i \* Play->HeightPoint + j].color) && (!Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y - 1].view) && (Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y - 1].brush == 0))

{

Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y - 1].view = 1;

d1 = d2 = 0;

CountPoint++;

SavePoint = (PPOINT)HeapReAlloc(Play->Heap, 0, SavePoint, CountPoint \* sizeof(POINT));

PArrOfPoint = SavePoint;

\*(PArrOfPoint + CountPoint - 1) = Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y - 1].point;

}

else

{

if ((Near\_x == i) && (Near\_y == j))

StartRedraw = 1;

else

Near\_y--;

}

break;

case 4:

if ((Play->PointArr[(Near\_x - 1) \* Play->HeightPoint + Near\_y].color == Play->PointArr[i \* Play->HeightPoint + j].color) && (!Play->PointArr[(Near\_x - 1) \* Play->HeightPoint + Near\_y].view) && (Play->PointArr[(Near\_x - 1) \* Play->HeightPoint + Near\_y].brush == 0))

{

Play->PointArr[(Near\_x - 1) \* Play->HeightPoint + Near\_y].view = 1;

d1 = d2 = 2;

CountPoint++;

SavePoint = (PPOINT)HeapReAlloc(Play->Heap, 0, SavePoint, CountPoint \* sizeof(POINT));

PArrOfPoint = SavePoint;

\*(PArrOfPoint + CountPoint - 1) = Play->PointArr[(Near\_x - 1) \* Play->HeightPoint + Near\_y].point;

}

else

{

if ((Near\_x == i) && (Near\_y == j))

StartRedraw = 1;

else

Near\_x--;

}

break;

case 5:

if ((Play->PointArr[(Near\_x - 1) \* Play->HeightPoint + Near\_y].color == Play->PointArr[i \* Play->HeightPoint + j].color) && (!Play->PointArr[(Near\_x - 1) \* Play->HeightPoint + Near\_y].view) && (Play->PointArr[(Near\_x - 1) \* Play->HeightPoint + Near\_y].brush == 0))

{

Play->PointArr[(Near\_x - 1) \* Play->HeightPoint + Near\_y].view = 1;

d1 = d2 = 2;

CountPoint++;

SavePoint = (PPOINT)HeapReAlloc(Play->Heap, 0, SavePoint, CountPoint \* sizeof(POINT));

PArrOfPoint = SavePoint;

\*(PArrOfPoint + CountPoint - 1) = Play->PointArr[(Near\_x - 1) \* Play->HeightPoint + Near\_y].point;

}

else

{

if ((Near\_x == i) && (Near\_y == j))

StartRedraw = 1;

else

Near\_x--;

}

break;

case 6:

if ((Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y + 1].color == Play->PointArr[i \* Play->HeightPoint + j].color) && (!(Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y + 1].view)) && (Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y + 1].brush == 0))

{

Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y + 1].view = 1;

d1 = d2 = 4;

CountPoint++;

SavePoint = (PPOINT)HeapReAlloc(Play->Heap, 0, SavePoint, CountPoint \* sizeof(POINT));

PArrOfPoint = SavePoint;

\*(PArrOfPoint + CountPoint - 1) = Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y + 1].point;

}

else

{

if ((Near\_x == i) && (Near\_y == j))

StartRedraw = 1;

else

Near\_y++;

}

break;

case 7:

if ((Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y + 1].color == Play->PointArr[i \* Play->HeightPoint + j].color) && (!(Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y + 1].view)) && (Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y + 1].brush == 0))

{

Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y + 1].view = 1;

d1 = d2 = 4;

CountPoint++;

SavePoint = (PPOINT)HeapReAlloc(Play->Heap, 0, SavePoint, CountPoint \* sizeof(POINT));

PArrOfPoint = SavePoint;

\*(PArrOfPoint + CountPoint - 1) = Play->PointArr[(Near\_x)\*Play->HeightPoint + Near\_y + 1].point;

}

else

{

if ((Near\_x == i) && (Near\_y == j))

StartRedraw = 1;

else

Near\_y++;

}

break;

}

d2++;

if (d2 == 8)

d2 = 0;

if (d2 == d1)

loop1 = 0;

}

if (StartRedraw)

{

if (PlayGroundIsUniq(Play, SavePoint, CountPoint, Play->PointArr[i \* Play->HeightPoint + j].color))

{

NewPP = 0;

StartRedraw = 0;

res = 1;

int op = 0;

POINT pp[4];

while ((op < CountPoint) && (op < 4))

{

pp[op] = \*(SavePoint + op);

op++;

}

if (!Play->spis)

{

Play->spis = (PSPISOK)HeapAlloc(Play->Heap, HEAP\_ZERO\_MEMORY, sizeof(SPISOK));

Play->spis->ArrOfPoint = SavePoint;

Play->spis->color = Play->PointArr[i \* Play->HeightPoint + j].color;

Play->spis->CountPoint = CountPoint;

Play->spis->NextPoint = NULL;

}

else

{

PSPISOK tmp, temp;

temp = tmp = Play->spis;

temp = (PSPISOK)HeapAlloc(Play->Heap, HEAP\_ZERO\_MEMORY, sizeof(SPISOK));

temp->ArrOfPoint = SavePoint;

temp->CountPoint = CountPoint;

temp->color = Play->PointArr[i \* Play->HeightPoint + j].color;

temp->NextPoint = tmp;

Play->spis = temp;

}

}

}

else

{

if ((SavePoint != NULL) && (NewPP))

{

HeapFree(Play->Heap, 0, SavePoint);

SavePoint = NULL;

NewPP = 0;

}

}

Direction++;

}

return res;

}

/// <summary>

/// this function search exist loop. if loop under other loop-> first

/// loop delete

/// </summary>

/// <param name="Play"></param>

void PlayGroundIsPoligonExist(PplayGround Play)

{

PSPISOK PSp = Play->spis, tmp = Play->spis;

PPOINT pp;

int i, j;

while (PSp != NULL)

{

pp = PSp->ArrOfPoint;

i = (int)(pp->x / (int)(Play->playGroundsize.right / Play->WidthPoint));

j = (int)(pp->y / (int)(Play->playGroundsize.bottom / Play->HeightPoint));

if (Play->PointArr[i \* Play->HeightPoint + j].brush != 0)

{

if (tmp == PSp)

{

PSPISOK delete = Play->spis;

Play->spis = PSp->NextPoint;

HeapFree(Play->Heap, 0, delete);

}

else

{

PSPISOK delete = PSp;

tmp->NextPoint = PSp->NextPoint;

HeapFree(Play->Heap, 0, delete);

}

}

tmp = PSp;

PSp = PSp->NextPoint;

}

}

int PlayGroundAddPointInt(PplayGround Play, int Point)

{

int i = Point >> 16;

int j = Point & (MAXINT>>16);

Play->YouStep = Play->YouStep ^ 1;

if (Play->player.color == RED)

{

Play->PointArr[i \* Play->HeightPoint + j].color = RED;

Play->player.color = BLUE;

}

else

if (Play->player.color == BLUE)

{

Play->PointArr[i \* Play->HeightPoint + j].color = BLUE;

Play->player.color = RED;

}

else

{

Play->PointArr[i \* Play->HeightPoint + j].color = RED;

Play->player.color = BLUE;

}

PlayGroundTryBrush(Play, i, j);

PlayGroundCalculatePoint(Play);

PlayGroundIsPoligonExist(Play);

PlayGroundBackDraw(Play);

}

int PlayGroundAddPoint(PplayGround Play, LPARAM lParam)

{

// получение координат

int x = GET\_X\_LPARAM(lParam);

int y = GET\_Y\_LPARAM(lParam);

// получение значение в массиве

int i = 0, j = 0;

i = (int)(x / (int)(Play->playGroundsize.right / Play->WidthPoint));

j = (int)(y / (int)(Play->playGroundsize.bottom / Play->HeightPoint));

// проверка на попадание в радиус точки

if (!((x > Play->PointArr[i \* Play->HeightPoint].point.x - Play->rad \* 2) && (x < Play->PointArr[i \* Play->HeightPoint].point.x + Play->rad \* 2)))

return -1;

if (!((y > Play->PointArr[i \* Play->HeightPoint + j].point.y - Play->rad \* 2) && (y < Play->PointArr[i \* Play->HeightPoint + j].point.y + Play->rad \* 2)))

return -1;

// проверка на закрашенность

if (!(Play->PointArr[i \* Play->HeightPoint + j].color == 0))

return;

if (!(Play->PointArr[i \* Play->HeightPoint + j].brush == 0))

return;

Play->YouStep = Play->YouStep ^ 1;

if (Play->player.color == RED)

{

Play->PointArr[i \* Play->HeightPoint + j].color = RED;

Play->player.color = BLUE;

}

else

if (Play->player.color == BLUE)

{

Play->PointArr[i \* Play->HeightPoint + j].color = BLUE;

Play->player.color = RED;

}

else

{

Play->PointArr[i \* Play->HeightPoint + j].color = RED;

Play->player.color = BLUE;

}

int res = 0;

PlayGroundTryBrush(Play, i, j);

PlayGroundCalculatePoint(Play);

PlayGroundIsPoligonExist(Play);

PlayGroundBackDraw(Play);

res = (i << 16 ) + j;

return res;

}

int PlayGroundEndGame(PplayGround Play)

{

if (Play == NULL)

return 0;

if (Play->CountBlue > 25)

{

MessageBox(Play->hwndSelf, L"Выйграл синий игрок!", L" ",

MB\_OK);

Play->EndGame = 1;

return 1;

}

if (Play->CountRed > 25)

{

MessageBox(Play->hwndSelf, L"Выйграл красный игрок!", L" ",

MB\_OK);

Play->EndGame = 1;

return 1;

}

int i = 0;

int j = 0;

int Clear = 0;

while (i < Play->WidthPoint)

{

j = 0;

while (j < Play->HeightPoint)

{

if (Play->PointArr[i\*Play->HeightPoint + j].color==0)

Clear++;

j++;

}

i++;

}

if (!Clear)

{

if (Play->CountBlue < Play->CountRed)

{

MessageBox(Play->hwndSelf, L"Выйграл красный игрок!",

L" ", MB\_OK);

Play->EndGame = 1;

return 1;

}

if (Play->CountBlue > Play->CountRed)

{

MessageBox(Play->hwndSelf,L"Выйграл синий игрок!",L" " , MB\_OK);

Play->EndGame = 1;

return 1;

}

if (Play->CountBlue == Play->CountRed)

{

MessageBox(Play->hwndSelf,L"Победтла Дружба!",L" ",MB\_OK);

Play->EndGame = 1;

return 1;

}

}

return 0;

}