Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компьютерные системы и сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

«Шашки онлайн»

БГУИР КР 1-40 01 01 115 ПЗ

Студент: гр. 251001 Кривицкий Ф. Ю.

Руководитель: Красковский П.Н.

Минск 2024

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 4](#_Toc152667987)

[ВВЕДЕНИЕ 5](#_Toc152667988)

[1 АНАЛИЗ ПРОТОТИПОВ, ЛИЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМУ ПРОГРАММНОМУ СРЕДСТВУ 6](#_Toc152667989)

[1.1 Анализ литературных источников 6](#_Toc152667990)

[1.2 Анализ существующих аналогов 7](#_Toc152667991)

[1.3 Постановка задачи 8](#_Toc152667992)

[2 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАМНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ 9](#_Toc152667993)

[2.1 Описание функциональных требований к программному средству 9](#_Toc152667994)

[2.2 Спецификация функциональных требований 9](#_Toc152667995)

[3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 11](#_Toc152667996)

[3.1 Построение дерева возможных ходов 11](#_Toc152667997)

[3.2 Разработка алгоритмов подсчёта оценки 13](#_Toc152667998)

[3.3 Добавление ходов 14](#_Toc152668002)

[3.4 Оптимизация перебора 15](#_Toc152668006)

[4 СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА 17](#_Toc152668008)

[4.1 Проектирование модулей программного средства 17](#_Toc152668009)

[4.2 Разработка модулей программного средства 17](#_Toc152668010)

[4.3 Проектирование основных структур данных 19](#_Toc152668011)

[5 ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ 20](#_Toc152668013)

[5.1 Тестирование и проверка работоспособности программного средства 20](#_Toc152668014)

[5.2 Анализ полученных результатов 22](#_Toc152668015)

[6 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ 24](#_Toc152668021)

[ЗАКЛЮЧЕНИЕ 28](#_Toc152668022)

[ПРИЛОЖЕНИЕ А 30](#_Toc152668025)

[ПРИЛОЖЕНИЕ Б 35](#_Toc152668026)

# **ВВЕДЕНИЕ**

**Русские шашки** — традиционный и наиболее популярный вид [шашек](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D1%88%D0%BA%D0%B8) в России, [странах](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D1%82%D1%81%D0%BE%D0%B2%D0%B5%D1%82%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%BE) бывшего [СССР](https://ru.wikipedia.org/wiki/%D0%A1%D0%A1%D0%A1%D0%A0) и в [Израиле](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D1%80%D0%B0%D0%B8%D0%BB%D1%8C). Цель игры — лишить противника возможности хода путём взятия или запирания всех его шашек (в [обратных русских шашках](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D1%8B%D0%B5_%D1%80%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B5_%D1%88%D0%B0%D1%88%D0%BA%D0%B8) цель противоположна — лишить себя возможности хода).

[Шашечная доска](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D1%88%D0%B5%D1%87%D0%BD%D0%B0%D1%8F_%D0%B4%D0%BE%D1%81%D0%BA%D0%B0) располагается между партнерами таким образом, чтобы слева от играющего находилось тёмное угловое поле. В начальной позиции у каждой стороны по 12 шашек, которые занимают первые три ряда с каждой стороны. Игра ведется по тёмным полям. Шашки делятся на простые и дамки. В начальном положении все шашки простые. При достижении последнего (восьмого от себя) горизонтального ряда простая шашка превращается в дамку. Дамка обычно обозначается либо двумя шашками, поставленными друг на друга, либо перевернутой шашкой.

Подробнее о правилах русских шашек:

https://shashki.ru/variations/draughts64

Цель данной курсовой работы – разработать онлайн-приложение для игры в русские шашки, которое включает игровой движок для русских шашек, который будет находить наилучшие ходы для текущей ситуации на доске и производить их, уметь строить дерево ходов на несколько вперёд, перебирать все возможные ходы игрока и оценивать ситуацию на доске некоторым числом, которое объективно отображало бы позицию. Также создать графическую оболочку для приложения с целью удобного взаимодейтсвия программы и пользователя.

Данная пояснительная записка содержит следующие основные разделы:

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству;
2. Анализ требований к программному средству и разработка функциональных требований;
3. Проектирование программного средства;
4. Создание (конструирование) программного средства;
5. Тестирование, проверка работоспособности и анализ полученных результатов;
6. Руководство по установке и использованию.
7. **АНАЛИЗ ПРОТОТИПОВ, ЛИЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ**
   1. **Анализ литературных источников**

Когда создавались первые вычислительные машины, их воспринимали только как дополнение к человеческому разуму. И до недавнего времени так и было. Программисты учили компьютеры играть в шахматы с 1960-х годов. И тогда победа у игрока-новичка уже считалась большим прогрессом. О серьёзных матчах даже не задумывались.

В 1980-х программа Belle достигла рейтинга Эло в 2250 пунктов, что примерно соответствует рейтингу мастера спорта. И с того времени развитие компьютерных шахмат вышло на совершенно новый уровень. Сначала честь человечества не смог защитить Гарри Каспаров в 1996 году, а сегодня уже создана нейросеть с рейтингом около 5000 Эло, что в разы превосходит даже сильнейших игроков.

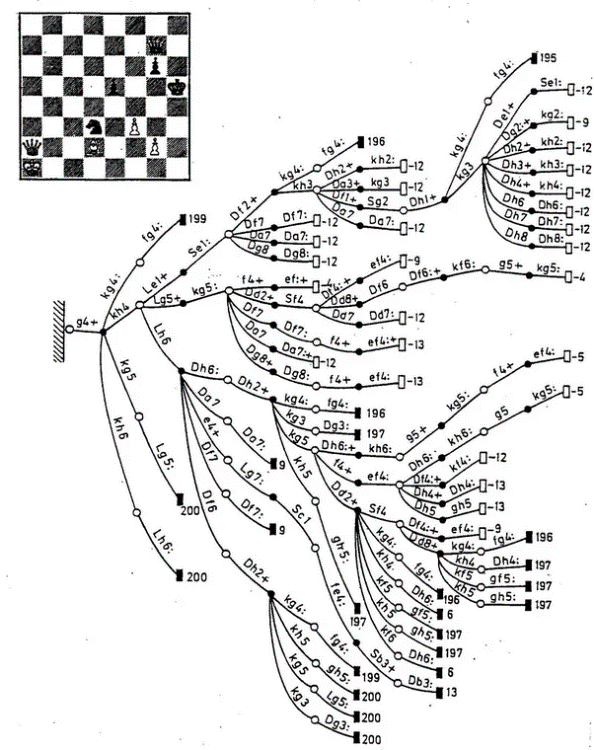
Основой движка является построение дерева ходов. Идея алгоритмов была взята из open-source движка для шахмат «Stockfish». Неофициальное название игрового движка для русских шашек «Vobla» пошло от дословного перевода на русский язык «Stockfish», поскольку «Vobla» строится на алгоритмах, адаптированных под русские шашки.

Рисунок 1.1 – Пример дерева возможных ходов Stockfish

* 1. **Анализ существующих аналогов**

Chess.com — сайт для игры в шахматы онлайн, форум, шахматная социальная сеть, а также одноимённая компания, владеющая данным сайтом. По данным Alexa.com, является самым популярным шахматным сайтом в мире.

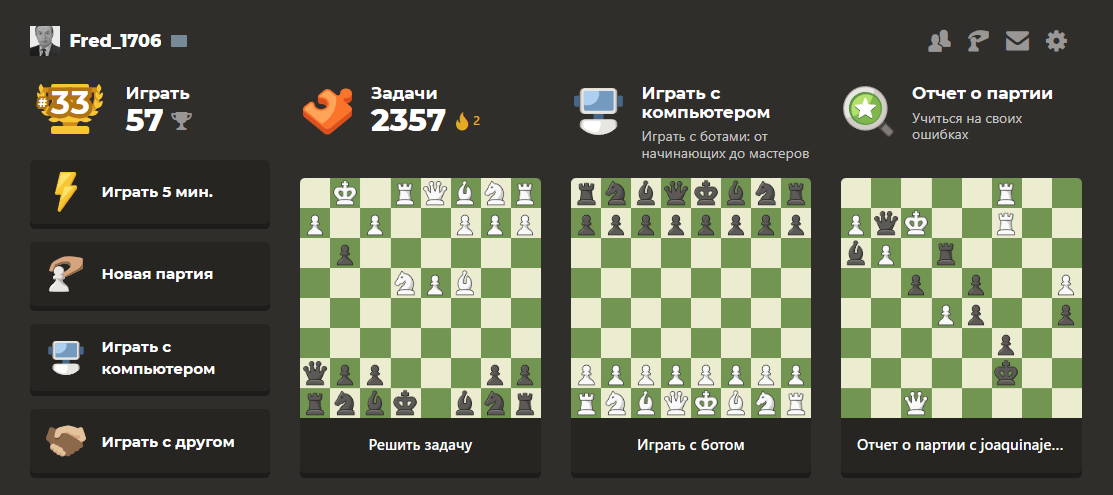


Рисунок 1.2 – Сайт «chess.com»

С января 2012 года на сайте проводятся так называемые «матчи смерти», ныне называемые Speed Chess Championship, в которых два заранее известных шахматиста в течение 3 часов без перерыва играют партии в блиц (по 5 минут, по 3 минуты, по 1 минуте, все с добавлением 1 секунды на ход). К участию в этих акциях привлекаются такие известные гроссмейстеры, как Хикару Накамура, Дмитрий Андрейкин, Максим Вашье-Лаграв.

В октябре 2013 года компания chess.com приобрела голландский шахматный новостной сайт chessvibes.com. В 2014 году было заявлено, что на сайте с момента его запуска было проведено более 1 миллиарда партий, из которых 100 миллионов — по переписке.

На сайте chess.com есть возможность анализа партий при помощи движка «Stockfish», а также курсы по обучению игре и решению задач.

* 1. **Постановка задачи**

В своём проекте, в соответствии с заданием, я буду использовать следующие алгоритмы:

* Метод построения дерева всех возможных ходов для текущей ситуации;
* Метод частичного перебора возможных ходов;
* Метод оценки текущей позиции на доске;
* Метод minimax с альфа-бета отсечением для оптимизации перебора ходов.

После построения ходов в глубину необходимо построить все возможные сбития, иначе оценка позиции может быть необъективна. Сбивания при построении считаются за единицу глубины.

Задачей курсовой работы будет онлайн-приложение, включающее игровой движок, который будет просичтывать ходы за время, зависящее от глубины просчета. При слишком быстром просчете ходов, следует увеличить глубину:

* Глубина 4: менее 1 мс;
* Глубина 8: от 1 до 20 мс;
* Глубина 10: от 20 мс до 1 с;
* Глубина 12: от 1 с до 5 с.

Приложение должно позволять анализировать партию после ее завершения, в частности классифицировать ходы по их статусу исходя их оценки, показывать точность ходов черных и точность ходов белых.

Алгоритмы оценки ходов и их точности должны быть похожи на алгоритмы сайта «chess.com»

1. **АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАМНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ**
   1. **Описание функциональных требований к программному средству**

Цель разработки программного средства (ПС) – создать онайлн-приложение для шашек, которое будет позволять анализировать сыгранные партии. Для реализации любого программного средства, необходимо чётко определить функциональные требования, которые в дальнейшем послужат базисом для оформления и реализации тематики курсовой работы. Используя данные, полученые из раздела «Анализ литературных источников», опираясь на изученные аналоги и вводные данные курсовой работы, для реализации движка выделяются следующие основные требования к функционалу к программному средству:

* Выбор цвета шашек для пользователя;
* Выбор сложности движка, то есть глубины просчета ходов;
* Возможность осуществления ходов;
* Построение дерева ходов и выбор наилучшего хода для движка;
* Определение победы, ничьей, поражения для игрока;
* Высвечивание сообщения при завершении партии;
* Возможность играть по сети.
  1. **Спецификация функциональных требований**

1. Выбор цвета шашек для пользователя:

* По кнопке на стартовом экране пользователь может выбрать цвет, за который он хочет играть;
* Если цвет – черный, то доска дожна отображаться в перевернутом виде.

1. Выбор сложности движка, то есть глубины просчета ходов:

* Должны быть следующие уровни сложности:
  + Easy (4 хода);
  + Medium (8 ходов);
  + Hard (10 ходов);
  + Impossible (12 ходов).

1. Возможность осуществления ходов:

* Возможность осуществлять все те и только те возможные ходы, которые позволяют правила при данной ситуации на доске;
* Должна быть возможность ходить при помощи мыши.
* Ходы должны осуществляться путем drag-and-roll.

1. Построение дерева ходов и выбор наилучшего хода для движка:

* Должна происходить сортировка ходов по оценке после прохода всех элементов оптимизированного дерева;
* Движок должен ходить по лучшему из вариантов.

1. Определение победы, ничьей, поражения для игрока:

* Победа и поражение должны определяться по правилам ФШР;
* Ничья должна определяться только по согласию двух игроков.

1. Высвечивание сообщения при завершении партии:

* При окончании партии должно высвечиваться соответствующее сообщение;
* Текст сообщения должен исходить из итогов партии.

1. Возможность играть по сети:

* В меню должен быть соответствующий пункт;
* Необходимо реализовать контроль времени «Рапид» (10 минут на партию для каждого из игроков)

1. **ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**
   1. **Построение дерева возможных ходов**

Каждый узел дерева содержит координаты хода, который привел к образованию этого узла; доску, которая получится при этом ходе; и цвет узла, определяющий очередь совершения хода. Если ход белых, то узел содержит True, в противном случае – False. Также при создании узла считается оценка текущего положения на доске, которая будет играть ключевую роль при выборе наилучшего хода.

Корневым элементом дерева считается узел, содержащий начальную доску, ход белых и координаты хода (0; 0; 0; 0). Затем, если нет необходимости бить, строятся все возможные ходы для текущей ситуации (всего существует 7 вариантов первого хода для белых) и добавляются как дочерние элементы для узла, у них становится противоположным цвет. Если на каком-либо этапе построения дерева возникает необходимость сбивать, то цвет узла дерева меняется на противоположный только после завершения сбиваний.

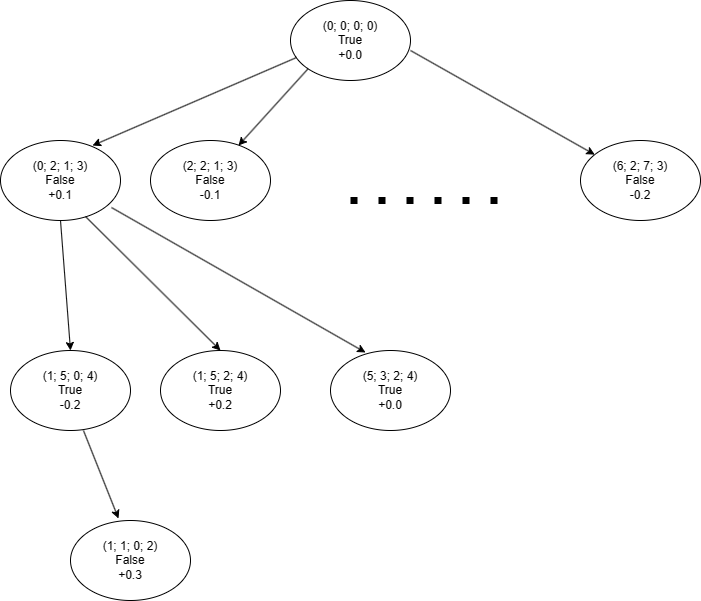
Алгоритм продолжается, пока высота дерева не достигнет некоторой определенной глубины.

Рисунок 3.1.1 – Пример дерева возможных ходов

После построения дерева идет этап оценки каждого из дочерних элементов корневого узла путем рекурсивного концевого обхода. Алгоритм следующий:

1. Если в корне стоит True, то выбрать ход с максимальной оценкой;
2. Иначе выбрать ход с минимальной.

Для примера возьмем узел (0; 2; 1; 3). В корне стоит False, а значит

необходимо выбрать ход с минимальной оценкой. У дочерних узлов (1; 5; 2; 4) и (5; 3; 2; 4) нет потомков, а значит их оценка никак не поменяется. У узла

(1; 5; 0; 4) есть единственный потомок (1; 1; 0; 2), поэтому оценка (1; 5; 0; 4) станет равной оценке (1; 1; 0; 2).

Вернёмся к узлу (0; 2; 1; 3). Минимальная оценка из его дочерних у хода

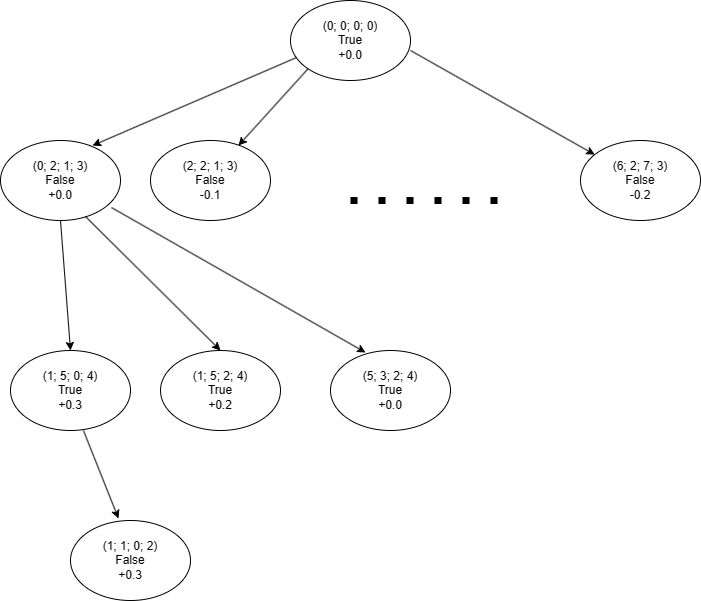
(5; 3; 2; 4). Присваиваем её и получаем дерево, которое прошло алгоритм оценки ходов:

Рисунок 3.1.2 – Оценённое дерево возможных ходов

Далее происходит сортировка дочерних элементов корневого элемента по оценке и выбор наилучшего хода. Узел с наилучшей оценкой становится корневым, остальные удаляются из памяти. Алгоритм продолжается до тех пор, пока существует следующий ход. При отсутствии такового, объявляется конец игры. Кто не смог сделать ход – проиграл.

* 1. **Разработка алгоритмов подсчёта оценки**
     1. **Подсчёт оценки для обычных шашек**

Позиция на доске характеризуется двумя важными составляющими: расположение шашек и их количество. Иногда бывает, что три шашки в центре могут держать четыре шашка по краям доски, поэтому пришлось ввести доску оценки позиции для каждой шашки:

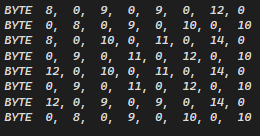
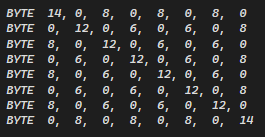


Рисунок 3.2.1 – Оценка позиции обычной шашки

Исходя из рисунка 3.2.1 видно, что хорошо оцениваются шашки, стоящие ближе к центру. По словам известного шашиста Остапчука М.Г., важными позициями сделаны клетки e1 и g1, а также симметричные им относительно центра b8 и d8. Эти позиции являются ключевыми, открывать их стоит только в крайних ситуациях. Позиции недалеко от центра, но не на краях доски считаются плохими из-за сложности сделать «петлю». Вместо двух шашек понадобятся три.

* + 1. **Подсчёт оценки для дамок**

Хорошей позицией считается край доски и центральная диагональ, поскольку при такой позиции дамку поймать сложно или невозможно. Плохой позицией – все остальные.

**** Рисунок 3.2.2 – Оценка позиции для дамки

* + 1. **Подсчёт итоговой оценки позиции**

Итоговая оценка считается по формуле:

Оценка = Б – Ч + 5 \* (Бд - Чд)

Б – сумма оценок белых шашек;

Ч – сумма оценок черных шашек;

Бд – сумма оценок белых дамок;

Чд – сумма оценок чёрных дамок.

Дамка считается в 5 раз ценнее обычной шашки, поэтому движок будет стараться создать максимальное количество дамок.

Если нет ни одной белой шашки или дамки на доске, то алгоритм выдает оценку = -100, что означает победу черных. Соответственно, при отсутствии черных шашек или дамок оценка будет = 100.

* 1. **Добавление ходов**

После каждого хода игровой движок заново просчитывает все ходы в выбранную зараннее глубину. В бета-версии программы было реализовано дерево в памяти, однако динамическое выделение достаточно долгое, из-за чего было решено превратить дерево в абстрактное дерево путем рекурсии.

В таблице 3.3 представлено количество узлов в полном дереве без оптимизации в зависимости от глубины просчета.

Таблица 3.3 – Зависимоть количества узлов от глубины

|  |  |
| --- | --- |
| Глубина просчёта | Количество узлов |
| 1 | >1 |
| 2 | >8 |
| 3 | >69 |
| 4 | >575 |
| 5 | >4787 |
| 6 | >39818 |
| 7 | >331203 |

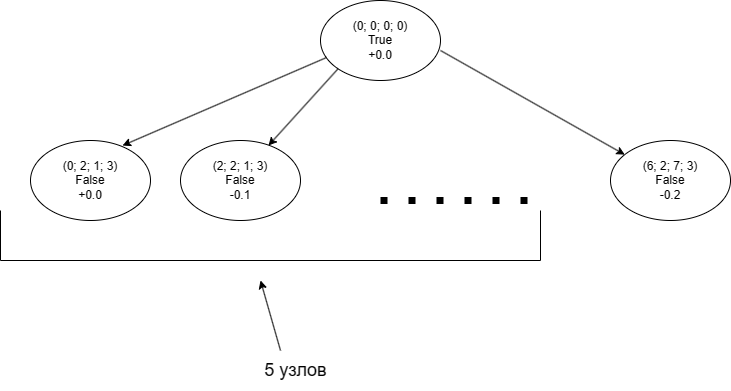
Зависимость количества узлов от глубины определяется по формуле:

Число 8.31794 было получено на практике как среднее количество возможных ходов. Количество узлов растёт экспоненциально, поэтому в данной курсовой работе не будет рассматриваться глубина просчёта более 7 ходов.

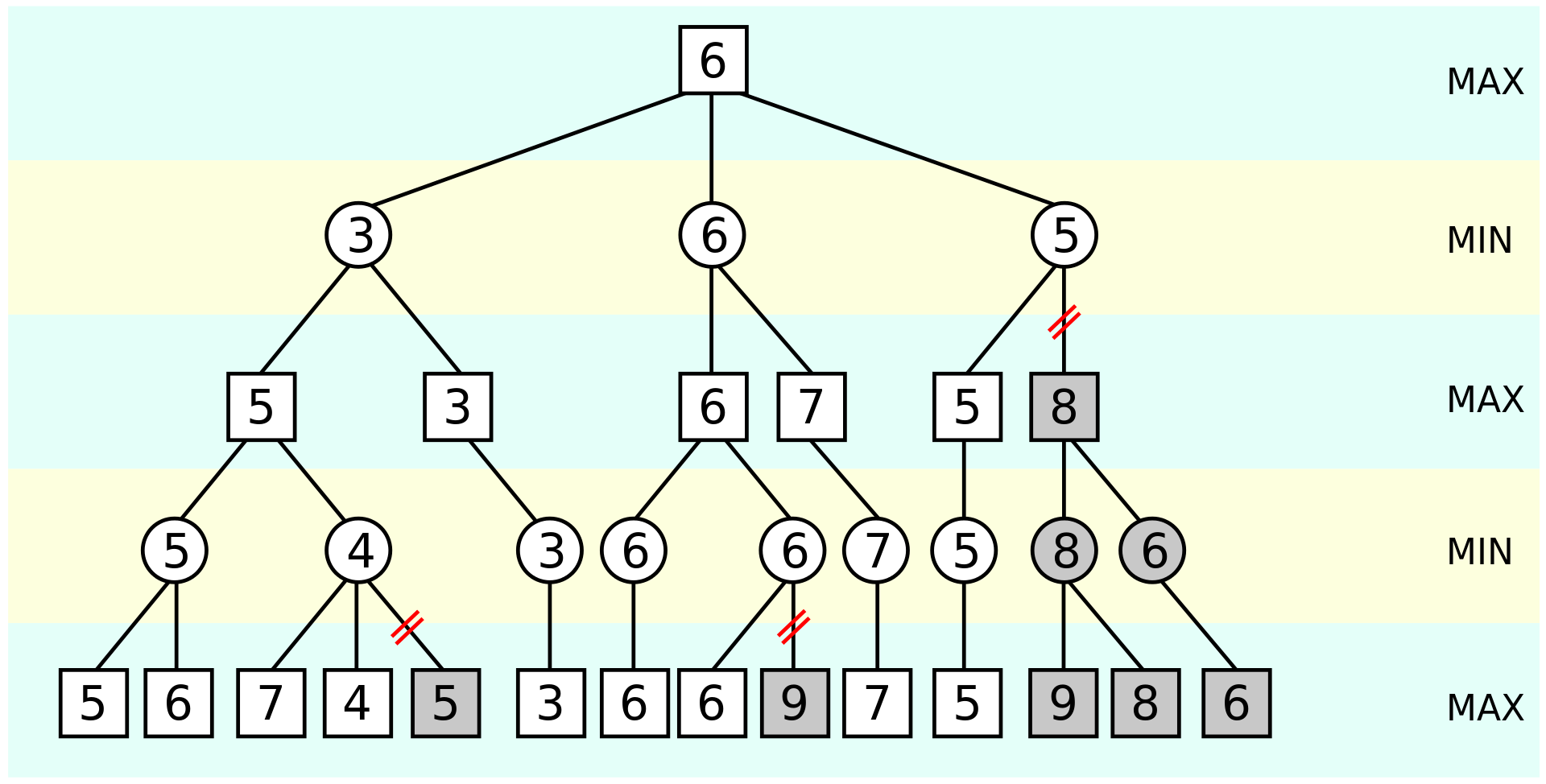
* 1. **Оптимизация перебора**

Полный перебор всех возможных долгий по времени и вызывает большие затраты оперативной памяти. Необходимо ввести оптимизацию перебора. Был использован следующий принцип:

1. В самом начале строится полное дерево ходов
2. Сортировка по оценке

…

1. Добавление узлов только в 5 самых лучших
2. Сортировка первых 5 ходов

Альфа-бета-отсечение (англ. alpha-beta pruning) — алгоритм поиска, стремящийся сократить количество узлов, оцениваемых в дереве поиска алгоритмом минимакса. Предназначен для антагонистических игр и используется для машинной игры (в компьютерных шахматах, компьютерном го и других). В основе алгоритма лежит идея, что оценивание ветви дерева поиска может быть досрочно прекращено (без вычисления всех значений оценивающей функции), если было найдено, что для этой ветви значение оценивающей функции в любом случае хуже, чем вычисленное для предыдущей ветви.

1. **СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА**
   1. **Проектирование модулей программного средства**

Разработка программного средства выполнялась на основе алгоритмов, представленных в разделе 3, а также на основе спецификации функциональных требований. Полный текст подпрограмм данных модулей представлен в Прило­жении Б.

В программном средстве используется 4 модуля:

* Модуль PossibleMoves – сожержит методы для построения возможных ходов исходя из текущего положения на доске;
* Модуль Engine – содержит класс Engine, который работает с абстрактным деревом ходов и является внешней оболочкой движка.
* Модуль Controls – содержит контролы кнопок, доски и другие игровые контроллы, необходимые для взаимодейтсвия пользователя и программы;
* Модуль Forms – содержит различные классы форм, каждая из которых включает окно для отрисовки и набор контроллов;
  1. **Разработка модулей программного средства**

Основные подпрограммы, использованные в разработке форм, перечислены в таблицах 4.2.1-4.2.4:

Таблица 4.2.1 – Основные подпрограммы модуля PossibleMoves

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя подпрог-раммы** | **Описание** | **Заголовок подпрограммы** | **Имя парамет-ра** | **Назначение параметра** |
| PMFill() | Построение массива возможных ходов. | bool PMFill(TField& field, MOVE\_TYPE type, TAM& AllMoves, mytype\* len, bool turn, mytype x, mytype y, mytype vector) | TField& field, MOVE\_TYPE type, TAM& AllMoves, mytype\* len, bool turn, mytype x, mytype y, mytype vec | Параметры положения:  поле, массив ходов, длина массива, очередь хода, параметры сбивания. |

Таблица 4.2.2 – Основные подпрограммы модуля Forms

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TAnalysicsForm() | Класс формы анализа | TAnalysicsForm(RenderWindow& renwin, std::vector<MoveData>& data); | win  data | Окно для отрисовки  Вектор ходов |
| TStartForm() | Класс формы выбора параметров игры | TStartForm(); |  |  |
| TEngineForm() | Класс формы партии против движка | TEngineForm(); |  |  |
| TPvpForm() | Класс формы партии против другого игрока | TPvpForm(); |  |  |

Таблица 4.2.3 – Основные подпрограммы модуля Engine

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| evaluate() | Оценка текущего состояния доски | void evaluate(MoveData& data, mytype depth); | data  depth | Параметры положения  Глубина просчета |
| PlayerMove() | Ход игрока | MOVE\_RESULT PlayerMove(MoveData& data); | data | Данные о ходе |
| EngineMove() | Ход движка | MOVE\_RESULT EngineMove(MoveData& data, mytype depth); | data  depth | Параметры положения  Глубина просчета |

Таблица 4.2.4 – Основные подпрограммы модуля Controls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GameController() | Класс для контроля правильности ходов и состояния игры | GameController() |  |  |
| TBoard() | Класс доски, который позволяет произовдить ходы | TBoard() |  |  |
| TCommentSection() | Класс, позволяющий посмотреть комментарии к ходам. | TCommentSection() |  |  |
| TObject() | Класс, родительский для всех отрисовываемых объектов | TObject() |  |  |

* 1. **Проектирование основных структур данных**

Таблица 4.3.1 – Основные структуры данных

|  |  |  |
| --- | --- | --- |
| **Название** | **Назначение** | **Структура** |
| TField | Массив, который хранит игровую доску | typedef mytype TField[8][8]; |
| TAM | Массив, который хранит все возможные ходы | typedef mytype TAM[100][4]; |
| InitB | Массив, хранящий начальную доску | BYTE 1, 0, 1, 0, 0, 0, 2, 0  BYTE 0, 1, 0, 0, 0, 2, 0, 2  BYTE 1, 0, 1, 0, 0, 0, 2, 0  BYTE 0, 1, 0, 0, 0, 2, 0, 2  BYTE 1, 0, 1, 0, 0, 0, 2, 0  BYTE 0, 1, 0, 0, 0, 2, 0, 2  BYTE 1, 0, 1, 0, 0, 0, 2, 0  BYTE 0, 1, 0, 0, 0, 2, 0, 2 |
| AssessD | Массив оценок для дамки на каждой клетке доски | BYTE 14,0, 8, 0, 8, 0, 8, 0  BYTE 0, 12,0, 6, 0, 6, 0, 8  BYTE 8, 0, 12,0, 6, 0, 6, 0  BYTE 0, 6, 0, 12,0, 6, 0, 8  BYTE 8, 0, 6, 0, 12,0, 6, 0  BYTE 0, 6, 0, 6, 0, 12,0, 8  BYTE 8, 0, 6, 0, 6, 0, 12,0  BYTE 0, 8, 0, 8, 0, 8, 0, 14 |
| AssessS | Массив оценок для шашки на каждой клетке доски | BYTE 8, 0, 9, 0, 9, 0, 12,0  BYTE 0, 8, 0, 9, 0, 10,0,10  BYTE 8, 0, 10,0, 11,0, 14,0  BYTE 0, 9, 0, 11,0, 12,0,10  BYTE 12, 0,10, 0, 11,0,14,0  BYTE 0, 9, 0,11,0,12, 0,10  BYTE 12,0, 9, 0, 9, 0,14,0  BYTE 0, 8, 0, 9, 0, 10,0,10 |

1. **ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ**
   1. **Тестирование и проверка работоспособности программного средства**

Тестирование – важный этап разработки, поскольку при написании программного кода неизбежны. Тестирование – сравнение реального и ожидаемого поведения программы. Основная цель – выявление несоответствия функционирования программы спецификации и устранение их для повышения качества программного средства. Проводилось функциональное тестирование программы.

Функциональные тесты, проведенные над программой, представлены в таблице 5.1.

Таблица 5.1 – Результаты функционального тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Но-мер  теста | Тестируе-мая функцио-нальность | Последовательность действий | Ожидаемый результат | Получен-ный результат |
| 1 | Запуск приложения | 1) Нажатие на ярлык “Vobla” | Приложение запустилось | Тест пройден |
| 2 | Выбор сложности “impossible” и чёрного цвета | 1) Выбор в меню цвета “Black”  2) Выбор в меню сложности “impossible”  3) Нажатие кнопки “Start” | Игра запущена, выбран чёрный цвет и сложность “impossible”. Доска отображается в перевёрнутом виде. | Тест пройден |
| 3 | Ход шашкой | 1) Нажатие на клетку доски с шашкой  2) Перенесение курсора мыши на желаемый ход  3) Отпускание курсора мыши | Успешный ход | Тест пройден |

Продолжение таблицы 5.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | Невозможный ход шашкой | Выполнение теста 3, но отпустить мышку на заведомо невозможной для хода клетке | Ход не произошёл | Тест пройден |
| 5 | Проверка ничьей | Сведение партии к ничьей | Завершение партии и высвечивание текста “Draw” | Тест пройден |
| 6 | Проверка победы | Победа в партии | Завершение партии и высвечивание текста “You win” | Тест пройден |
| 7 | Проверка поражения в партии | Поражение в партии | Завершение партии и высвечивание текста “You lose” | Тест пройден |
| 8 | Начало онлайн-партии | 1) Выбор «vs Player»  2) Нажатие «Start»  3) Ожидание подключния другого игрока | Онлайн-партия начата | Тест пройден |

* 1. **Анализ полученных результатов**

Исходя из полученных результатов в разделе 5.1 можно отметить, что движок не выдал ошибок, связанных с ошибками памяти, ее переполнением или зависанием программы.

Программа была протестирована на сайте https://logic-games.spb.ru/checkers, где показала 2021 ЭЛО, что соответствует званию КМС. Статистика игрового движка:



Ошибок, связанных с интерфейсом, отрисовкой доски или оценки не обнаружено. Все тесты программы сохранены на данном сайте. Единственный минус данного движка в том, что он плохо умеет побеждать. У движка нет базы дебютов и эндшпилей, каждый раз движок просчитывает все с начала. Чем выше рейтинг, тем выше вероятность получить ничью, и, соответственно, не увеличить рейтинг. До рейтинга 1900 движок давал 83% побед, где было 4% ничьих и 13% поражений по истечению времени на ход.

1. **РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ**

Для того, чтобы начать использовать программное средство, необходимо запустить файл Vobla.exe.

После открытия программы появляется окно. Пользователю предстоит выбрать режим игры: против другого игрока или против движка. По умолчанию высветится окно для выбора игры против игрока. Следует ввести IP адрес и порт сервера для подключения к нему.

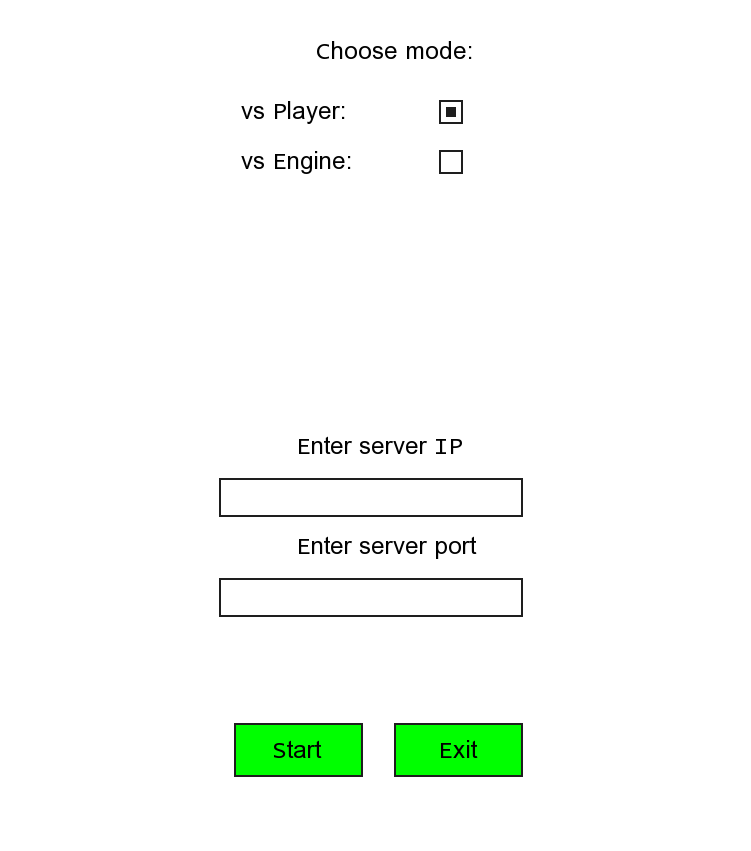


Рисунок 6.1.1 – Интерфейс программы

При выборе «vs Engine» появляется возможность выбора цвета шашек и сложности движка. Меню показано на рисунке 6.1.2.

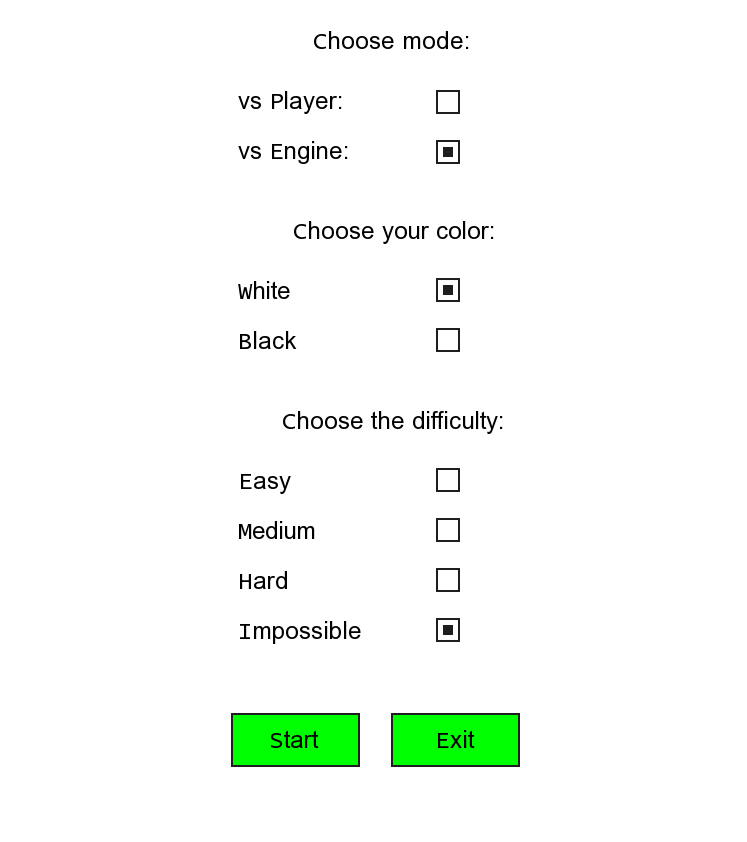


Рисунок 6.1.2 – Интерфейс программы

После выбора режима игры против движка появится окно, показанное на рисунке 6.2.1. Кнопка «Exit» позволяет выйти в главное меню. Кнопка «Analysics» позволяет проанализировать партию на ошибки: появится отдельное окно с анализом партии. Кнопка «Flip» перевернет доску так, как видит ее ваш соперник.

Ходить необходимо следующим образом: брать левой кнопкой мыши определенную шашку, перетаскивать ее на другую клетку и отпускать левую кнопку мыши. При невозможности осуществить ход по правилам ФШР, ход осуществлен не будет.

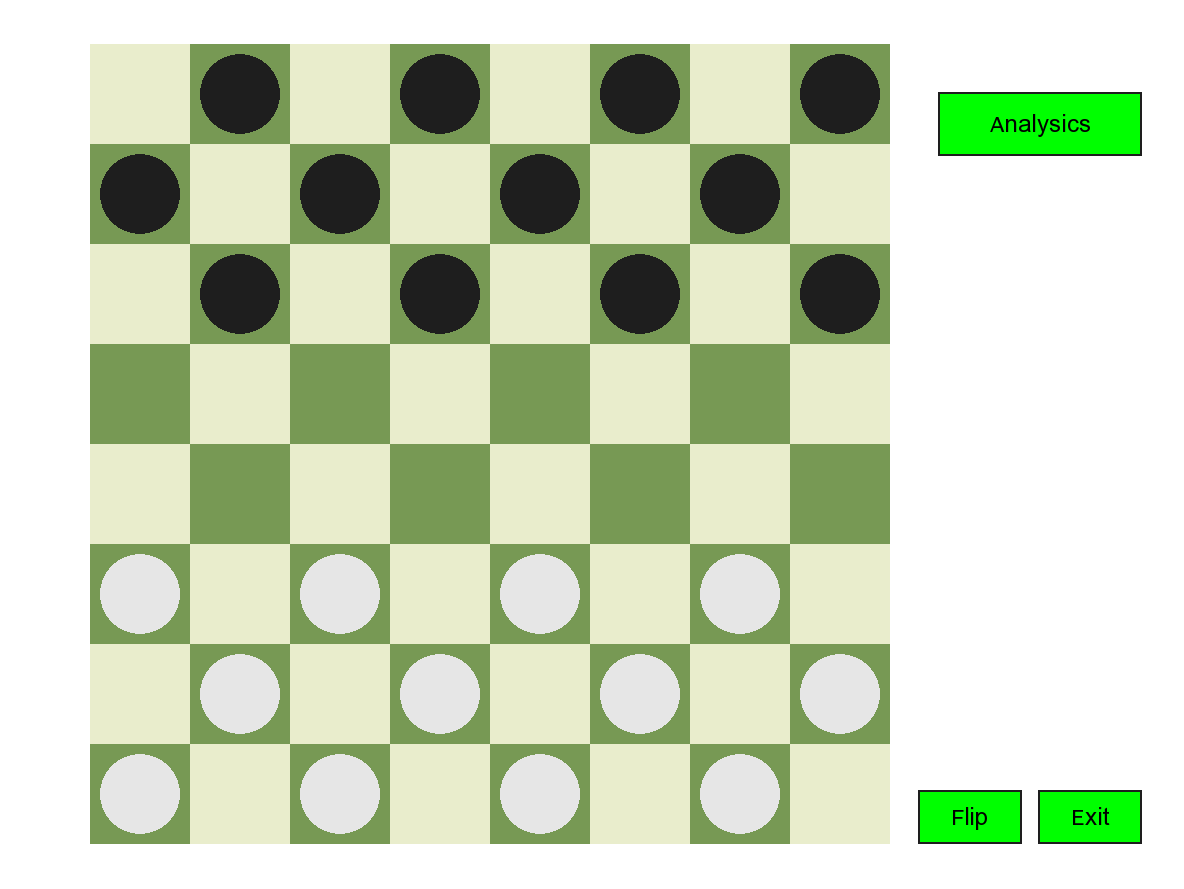
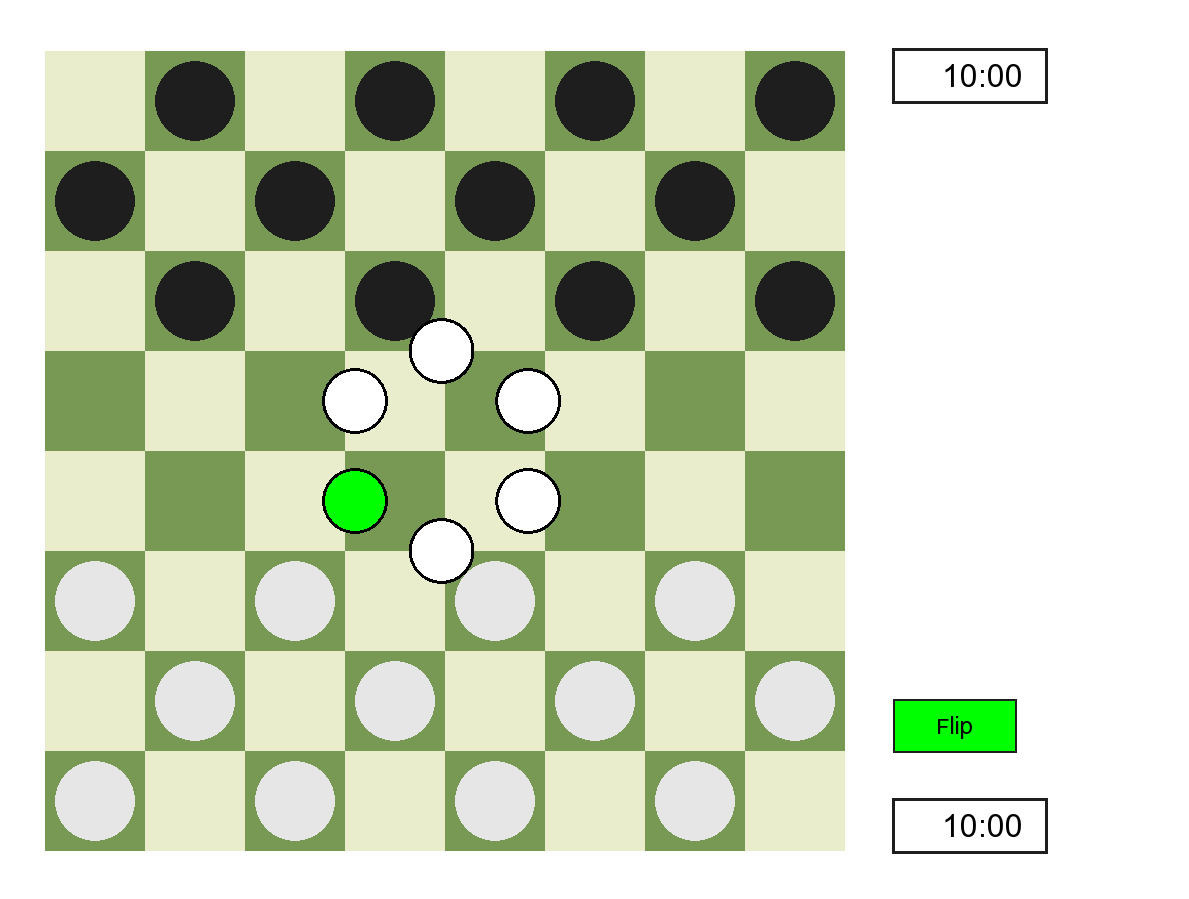


Рисунок 6.2 – Основной экран игры

Если пользователь выбрал режим игры «vs Player», у него появится форма, представленная на рисунке 6.3. Загрузка будет длиться до того момента, пока соперник не подключится к игре. Далее все аналогично игре против движка за исключением отсутствия кнопки «Exit» и «Analysics» во избежание нечетсной игры от пользователей.

 Рисунок 6.3 – Экран игры против игрока

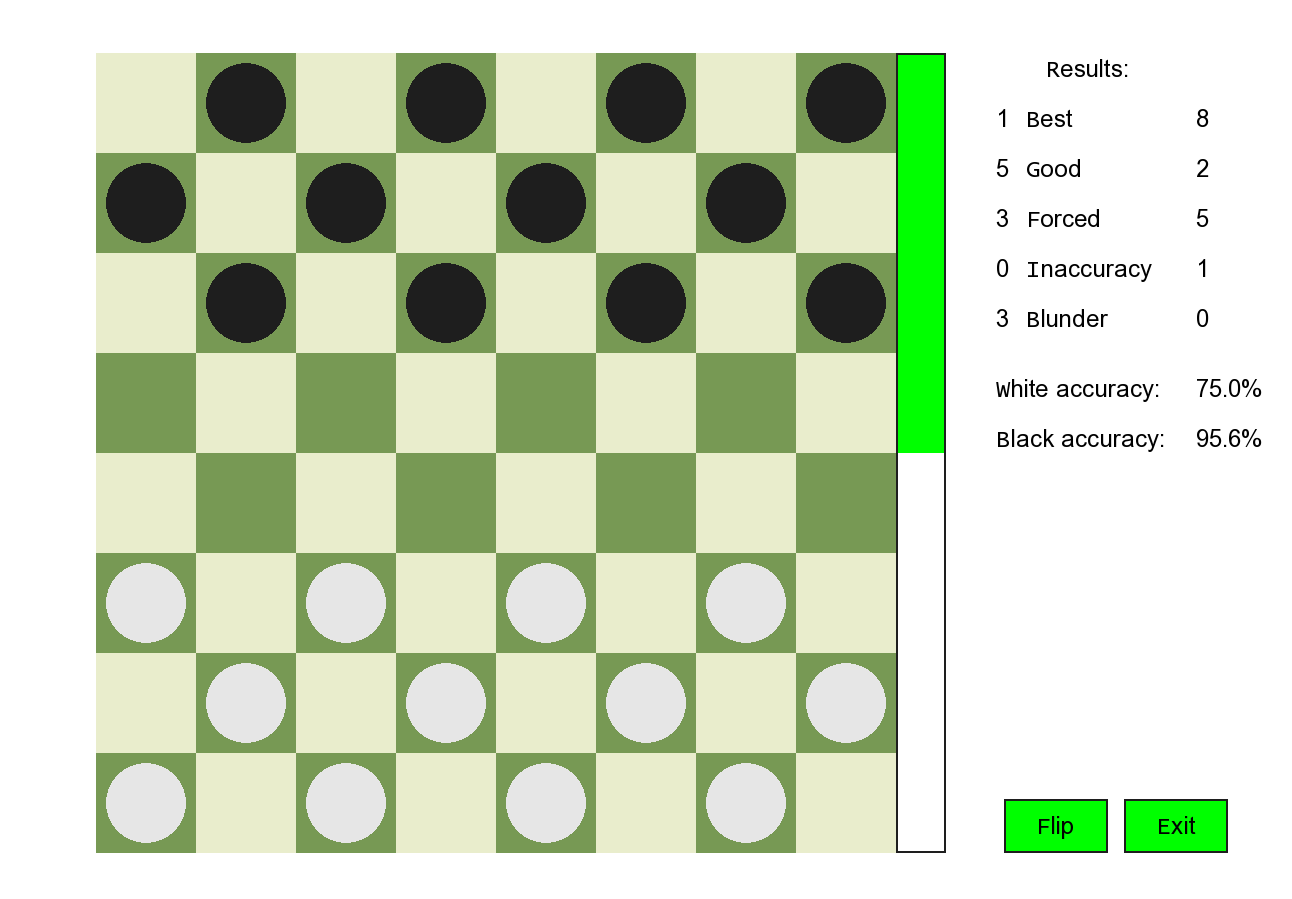


Рисунок 6.4 – Экран анализа партии

# **ЗАКЛЮЧЕНИЕ**

В процессе выполнения курсового проекта была проанализирована предметная область, рассмотрены существующие аналоги и выявлены их преимущества и недостатки. В качестве языка разработки использовался Visual C++ с подключением библиотеки SFML.

На этапе проектирования были разработаны блок-схемы алгоритмов. В соответствии с поставленной целью было разработано программное средство "Игровой движок для русских шашек".

В ходе курсового проекта мною было изучено дерево как структура данных, алгоритмы работы с ним. Из-за экспоненциального роста количества элементов в дереве, необходимо было использовать альфа- и бета-отсечение плохих ходов. Успешно был подключен модуль на языке ассемблера, который ускоряет просчет ходов в три раза, меняя логику алгоритмов только на уровне инструкций.

Был проведён матч между движком «Аврора» (разработчик Александр Свирин), который занял 5-ое место на чемпионате мира и данным программным средством, созданным в рамках курсовой работы 3-го семестра (далее – «Vobla»). Матч закончился достойным поражением «Vobla» на тридцать третьем ходу, что является весьма впечатлительным успехом. На вычисления «Vobla» потратил в два раза меньше времени и в 8 раз меньше нагружал процессор.

«Vobla» не имеет предела совершенству. Некоторые алгоритмы можно прописать на языке ассемблера, улучшить оптимизацию дерева перебора, прописать дебюты и эндшпили, прописать построение треугольника Петрова. Тогда можно будет всерьёз считать «Vobla» полноценным аналогом «Stockfish» для русских шашек.

# **Список использованной литературы**

[1] Stockfish. [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Stockfish.

[2] chess.com [Электронный ресурс] – Режим доступа:

https://www.chess.com/terms/stockfish-chess-engine.

[3] Русские шашки (википедия). [Электронный ресурс] – Режим доступа:

<https://en.wikipedia.org/wiki/Russian_draughts>

[4] Правила русских шашек. [Электронный ресурс] – Режим доступа:

<https://shashki.ru/variations/draughts64>

[5] Серебряная, Л. В. Структуры и алгоритмы обработки данных: учеб.-метод. пособие / Л. В. Серебряная, И. М. Марина. – Минск: БГУИР, 2013. – 51 с.

[6] Документация библиотеки SFML. [Электронный ресурс] – Режим доступа:

<https://www.sfml-dev.org/learn.php>

# [7] Шахматные алгоритмы, которые думают почти так же, как человек, только лучше. [Электронный ресурс] – Режим доступа: https://habr.com/ru/companies/skillfactory/articles/544040

# **ПРИЛОЖЕНИЕ А**



Рисунок A.1 – Блок-схема программы



Рисунок А.2 – Ход соперника



Рисунок А.3 – Ход игрока



Рисунок А.4 – Построение дерева ходов



Рисунок А.5 – Построение дерева сбиваний

# **ПРИЛОЖЕНИЕ Б**

.DATA

InitB BYTE 1, 0, 1, 0, 0, 0, 2, 0

BYTE 0, 1, 0, 0, 0, 2, 0, 2

BYTE 1, 0, 1, 0, 0, 0, 2, 0

BYTE 0, 1, 0, 0, 0, 2, 0, 2

BYTE 1, 0, 1, 0, 0, 0, 2, 0

BYTE 0, 1, 0, 0, 0, 2, 0, 2

BYTE 1, 0, 1, 0, 0, 0, 2, 0

BYTE 0, 1, 0, 0, 0, 2, 0, 2

AssessD BYTE 14, 0, 8, 0, 8, 0, 8, 0

BYTE 0, 12, 0, 6, 0, 6, 0, 8

BYTE 8, 0, 12, 0, 6, 0, 6, 0

BYTE 0, 6, 0, 12, 0, 6, 0, 8

BYTE 8, 0, 6, 0, 12, 0, 6, 0

BYTE 0, 6, 0, 6, 0, 12, 0, 8

BYTE 8, 0, 6, 0, 6, 0, 12, 0

BYTE 0, 8, 0, 8, 0, 8, 0, 14

AssessS BYTE 8, 0, 9, 0, 9, 0, 12, 0

BYTE 0, 8, 0, 9, 0, 10, 0, 10

BYTE 8, 0, 10, 0, 11, 0, 14, 0

BYTE 0, 9, 0, 11, 0, 12, 0, 10

BYTE 12, 0, 10, 0, 11, 0, 14, 0

BYTE 0, 9, 0, 11, 0, 12, 0, 10

BYTE 12, 0, 9, 0, 9, 0, 14, 0

BYTE 0, 8, 0, 9, 0, 10, 0, 10

.CODE

CheckCoord PROC

mov rax, 0

cmp cl, 0

jl Ending

cmp cl, 7

jg Ending

cmp dl, 0

jl Ending

cmp dl, 7

jg Ending

mov rax, 1

Ending:

ret

CheckCoord ENDP

CheckCoord4 PROC

push rbp

mov rbp, rsp

xor rax, rax

cmp cl, 0

jl Ending

cmp cl, 7

jg Ending

cmp dl, 0

jl Ending

cmp dl, 7

jg Ending

cmp r8b, 0

jl Ending

cmp r8b, 7

jg Ending

cmp r9b, 0

jl Ending

cmp r9b, 7

jg Ending

mov rax, 1

Ending:

pop rbp

ret

CheckCoord4 ENDP

GetMode PROC

push rbp

mov rbp, rsp

movsx rax, byte ptr[rbp + 48]

test rax, rax

jz Mode0

cmp rax, 2

je Mode23

cmp rax, 3

je Mode23

jmp Mode14

Mode0:

cmp rdx, r9

jng m0y1notBiggery2

cmp rcx, r8

jng m0x1notBiggerx2

mov rax, 4

jmp Ending

m0x1notBiggerx2:

mov rax, 3

jmp Ending

m0y1notBiggery2:

cmp rcx, r8

jng m02x1notBiggerx2

mov rax, 2

jmp Ending

m02x1notBiggerx2:

mov rax, 1

jmp Ending

Mode14:

sub rcx, rdx

sub r8, r9

cmp rcx, r8

jg m14Bigger

jl m14Lower

jmp Ending

m14Bigger:

mov rax, 2

jmp Ending

m14Lower:

mov rax, 3

jmp Ending

Mode23:

add rcx, rdx

add r8, r9

cmp rcx, r8

jg m23Bigger

jl m23Lower

jmp Ending

m23Bigger:

mov rax, 4

jmp Ending

m23Lower:

mov rax, 1

Ending:

pop rbp

ret

GetMode ENDP

;---------------------------------------------- BInit

BInit PROC

mov rdi, rcx

lea rsi, OFFSET InitB

mov rcx, 8

rep movsq

ret

BInit ENDP

;---------------------------------------------- BInit END

;---------------------------------------------- BCopy

BCopy PROC

mov rdi, rcx

mov rsi, rdx

mov rcx, 8

rep movsq

ret

BCopy ENDP

;---------------------------------------------- BCopy END

;---------------------------------------------- PosDamka

PosDamka PROC

mov rsi, rcx

mov rcx, 4

metka:

cmp byte ptr[rsi], 2

jnz noteq1

mov byte ptr[rsi], 4

noteq1:

add rsi, 15

cmp byte ptr[rsi], 1

jnz noteq2

mov byte ptr[rsi], 3

noteq2:

inc rsi

loop metka

ret

PosDamka ENDP

;---------------------------------------------- PosDamka END

;---------------------------------------------- asmMove

Move PROC

push rbp

mov rbp, rsp

and rdx, 0FFh

and r8, 0FFh

and r9, 0FFh

movsx r10, byte ptr[rbp + 48]

mov rsi, rcx

mov rdi, rcx

shl rdx, 3 ;imul rdx, 8

add rsi, rdx

add rsi, r8

shl r9, 3 ;imul r9, 8

add rdi, r9

add rdi, r10

mov al, byte ptr[rsi]

mov byte ptr[rdi], al

mov byte ptr[rsi], 0

call PosDamka

pop rbp

ret

Move ENDP

;---------------------------------------------- asmMove END

;---------------------------------------------- asmBeat

Beat PROC

push rbp

mov rbp, rsp

and rdx, 0FFh

and r8, 0FFh

and r9, 0FFh

movsx r10, byte ptr[rbp + 48]

mov rsi, rcx

mov rdi, rcx

shl rdx, 3 ;imul rdx, 8

add rsi, rdx

add rsi, r8

shl r9, 3 ;imul r9, 8

add rdi, r9

add rdi, r10

mov al, byte ptr[rsi]

mov byte ptr[rdi], al

mov byte ptr[rsi], 0

cmp rdx, r9

jg Bigger

jmp Lower

Bigger:

cmp r8, r10

jg BBig

jmp BLow

BBig:

sub rsi, 9

jmp Ending

BLow:

sub rsi, 7

jmp Ending

Lower:

cmp r8, r10

jg LBig

jmp LLow

LBig:

add rsi, 7

jmp Ending

LLow:

add rsi, 9

Ending:

mov byte ptr [rsi], 0

call PosDamka

pop rbp

ret

Beat ENDP

;---------------------------------------------- asmBeat END

;---------------------------------------------- SMCheck

SMCheck PROC

push rbp

mov rbp, rsp

xor rax, rax

mov rsi, rcx

mov rcx, [rbp + 48]

call CheckCoord4

test rax, rax

jz Ending

and r9, 0FFh

xor rax, rax

shl r9, 3 ;imul r9, 8

add r9b, cl

add rsi, r9

cmp byte ptr[rsi], 0

jnz Ending

mov rax, 1

Ending:

pop rbp

ret

SMCheck ENDP

;---------------------------------------------- SMCheck END

;---------------------------------------------- SBCheck END

SBCheck PROC

push rbp

mov rbp, rsp

mov rsi, rcx ;field

mov cl, byte ptr [rbp + 48] ;y2

mov rdi, rsi

call CheckCoord4

test rax, rax

jz Ending

xor rax, rax

mov r10b, byte ptr [rbp + 56] ;turn

inc r10b

and r9, 0FFh

mov r13, r9

shl r13, 3 ;imul r13, 8

add r13b, cl

add rdi, r13

mov r12b, byte ptr[rdi]

test r12b, r12b

jnz Ending

;x1 - dl

;y1 - r8b

;x2 - r9b

;y2 - cl

;temp - r11b

;x0 - r12b

;y0 - r13b

mov r12b, dl

mov r13b, r8b

cmp dl, r9b

jg B

inc r12b

cmp r8b, cl

jg LB

inc r13b

jmp LOR

LB:

dec r13b

jmp LOR

B:

dec r12b

cmp r8b, cl

jg BB

inc r13b

jmp LOR

BB:

dec r13b

LOR:

and r12, 0FFh

shl r12, 3; imul r12, 8

add r12b, r13b

add rsi, r12

mov r12b, byte ptr[rsi]

cmp r12b, r10b

je Okey

add r10b, 2

cmp r12b, r10b

je Okey

jmp Ending

Okey:

mov rax, 1

Ending:

pop rbp

ret

SBCheck ENDP

;---------------------------------------------- SBCheck END

;---------------------------------------------- getAssess

getAssess PROC

push rbp

mov rbp, rsp

mov rbx, rcx

mov rsi, OFFSET AssessS

mov rdi, OFFSET AssessD

mov r15, rsi

add r15, 63

xor r8, r8

xor r9, r9

xor r10, r10

xor r11, r11

xor rax, rax

mov rcx, 64

LoopMetka:

mov dl, byte ptr[rbx]

test dl, dl

jz Next

cmp dl, 1

je Piece1

cmp dl, 2

je Piece2

cmp dl, 3

je Piece3

jmp Piece4

Piece1:

add r8b, byte ptr[rsi]

jmp Next

Piece2:

add r9b, byte ptr[r15]

jmp Next

Piece3:

add r10b, byte ptr[rdi]

jmp Next

Piece4:

add r11b, byte ptr[rdi]

Next:

inc rsi

inc rdi

dec r15

inc rbx

loop LoopMetka

xor r12, r12

add r12, r8

add r12, r10

mov rax, -100

test r12, r12

jz Ending

xor r12, r12

add r12, r9

add r12, r11

mov rax, 100

test r12, r12

jz Ending

xor r12, r12

mov r12, r10

sub r12, r11

mov r13, r12 ;imul prepare

shl r12, 2 ;imul r12, 5

add r12, r13 ;imul end

add r12, r8

sub r12, r9

mov rax, r12

Ending:

pop rbp

ret

getAssess ENDP

;---------------------------------------------- getAssess END

;---------------------------------------------- NTBDamka

NTBDamka PROC

push rbp

mov rbp, rsp

;r15 - temp

;r14 - y0

;r13 - x0

;r12 - y

;r11 - x

;r10 - mode

;rsi - field

;rdi - field[xi][yi]

mov rsi, rcx

mov rdi, rcx

xor r15, r15

mov r15b, r9b

inc r15

mov r11b, dl

mov r12b, r8b

and r11, 0FFh

and r12, 0FFh

mov r13, r11

mov r14, r12

mov r10b, byte ptr[rbp+48]

cmp r10b, 4

je next4

xor rax, rax

mov al, r11b ;mov al, 8

shl al, 3 ;mul r11b

add al, r12b

add rdi, rax

while4:

inc r11b

inc r12b

cmp r11b, 7

jg endwhile4

cmp r12b, 7

jg endwhile4

add rdi, 9

cmp byte ptr[rdi], 0

jne endwhile4

jmp while4

endwhile4:

cmp r11b, 7

jge next4

cmp r12b, 7

jge next4

mov r9b, r15b

cmp byte ptr[rdi], r9b

je nextif4

add r9b, 2

cmp byte ptr[rdi], r9b

je nextif4

jmp next4

nextif4:

add rdi, 9

cmp byte ptr[rdi], 0

jne next4

mov rax, 1

jmp Ending

next4:

cmp r10b, 3

je next3

mov r11, r13

mov r12, r14

mov rdi, rsi

xor rax, rax

mov al, r11b ;mov al, 8

shl al, 3 ;mul r11b

add al, r12b

add rdi, rax

while3:

dec r11b

inc r12b

cmp r11b, 0

jl endwhile3

cmp r12b, 7

jg endwhile3

sub rdi, 7

cmp byte ptr[rdi], 0

jne endwhile3

jmp while3

endwhile3:

cmp r11b, 0

jle next3

cmp r12b, 7

jge next3

mov r9b, r15b

cmp byte ptr[rdi], r9b

je nextif3

add r9b, 2

cmp byte ptr[rdi], r9b

je nextif3

jmp next3

nextif3:

sub rdi, 7

cmp byte ptr[rdi], 0

jne next3

mov rax, 1

jmp Ending

next3:

cmp r10b, 2

je next2

mov r11, r13

mov r12, r14

mov rdi, rsi

xor rax, rax

mov al, r11b ;mov al, 8

shl al, 3 ;mul r11b

add al, r12b

add rdi, rax

while2:

inc r11b

dec r12b

cmp r11b, 7

jg endwhile2

cmp r12b, 0

jl endwhile2

add rdi, 7

cmp byte ptr[rdi], 0

jne endwhile2

jmp while2

endwhile2:

cmp r11b, 7

jge next2

cmp r12b, 0

jle next2

mov r9b, r15b

cmp byte ptr[rdi], r9b

je nextif2

add r9b, 2

cmp byte ptr[rdi], r9b

je nextif2

jmp next2

nextif2:

add rdi, 7

cmp byte ptr[rdi], 0

jne next2

mov rax, 1

jmp Ending

next2:

cmp r10b, 1

je next1

mov r11, r13

mov r12, r14

mov rdi, rsi

xor rax, rax

mov al, r11b ;mov al, 8

shl al, 3 ;mul r11b

add al, r12b

add rdi, rax

while1:

dec r11b

dec r12b

cmp r11b, 0

jl endwhile1

cmp r12b, 0

jl endwhile1

sub rdi, 9

cmp byte ptr[rdi], 0

jne endwhile1

jmp while1

endwhile1:

cmp r11b, 0

jle next1

cmp r12b, 0

jle next1

mov r9b, r15b

cmp byte ptr[rdi], r9b

je nextif1

add r9b, 2

cmp byte ptr[rdi], r9b

je nextif1

jmp next1

nextif1:

sub rdi, 9

cmp byte ptr[rdi], 0

jne next1

mov rax, 1

jmp Ending

next1:

mov rax, 0

Ending:

pop rbp

ret

NTBDamka ENDP

;---------------------------------------------- NTBDamka END

;---------------------------------------------- DamkaBeat

DamkaBeat PROC

push rbp

mov rbp, rsp

;r15 - mode

;r14 - y2

;r13 - x2

;r12 - y1

;r11 - x1

mov r15b, byte ptr[rbp + 56]

mov r14b, byte ptr[rbp + 48]

mov r13b, r9b

mov r12b, r8b

mov r11b, dl

mov rsi, rcx

mov rdi, rcx

xor rcx, rcx

xor rax, rax

mov al, r11b ;mov al, 8

shl al, 3 ;mul r11b

add al, r12b

add rsi, rax

xor rax, rax

mov al, r13b ;mov al, 8

shl al, 3 ;mul r13b

add al, r14b

add rdi, rax

mov al, byte ptr[rsi]

mov byte ptr[rdi], al

cmp r15b, 1

jne next1

mov cl, r13b

add cl, r14b

sub cl, r11b

sub cl, r12b

loop1:

cmp cl, 0

jle next

mov byte ptr[rsi], 0

add rsi, 9

inc r11b

inc r12b

sub cl, 2

jmp loop1

next1:

cmp r15b, 2

jne next2

mov cl, r11b

add cl, r14b

sub cl, r12b

sub cl, r13b

loop2:

cmp cl, 0

jle next

mov byte ptr[rsi], 0

sub rsi, 7

dec r11b

inc r12b

sub cl, 2

jmp loop2

next2:

cmp r15b, 3

jne next3

mov cl, r13b

add cl, r12b

sub cl, r11b

sub cl, r14b

loop3:

cmp cl, 0

jle next

mov byte ptr[rsi], 0

add rsi, 7

inc r11b

dec r12b

sub cl, 2

jmp loop3

next3:

cmp r15b, 4

jne next

mov cl, r11b

add cl, r12b

sub cl, r13b

sub cl, r14b

loop4:

cmp cl, 0

jle next

mov byte ptr[rsi], 0

sub rsi, 9

dec r11b

dec r12b

sub cl, 2

jmp loop4

next:

xor rcx, rcx

cmp r11b, r13b

jl x2

cmp r12b, r14b

jl x1y2

mov cl, r11b

sub cl, r13b

loop01:

test cl, cl

jz Ending

mov byte ptr[rsi], 0

sub rsi, 9

dec cl

jmp loop01

x1y2:

mov cl, r11b

sub cl, r13b

loop02:

test cl, cl

jz Ending

mov byte ptr[rsi], 0

sub rsi, 7

dec cl

jmp loop02

x2:

cmp r12b, r14b

jl x2y2

mov cl, r13b

sub cl, r11b

loop03:

test cl, cl

jz Ending

mov byte ptr[rsi], 0

add rsi, 7

dec cl

jmp loop03

x2y2:

mov cl, r13b

sub cl, r11b

loop04:

test cl, cl

jz Ending

mov byte ptr[rsi], 0

add rsi, 9

dec cl

jmp loop04

Ending:

pop rbp

ret

DamkaBeat ENDP

;---------------------------------------------- DamkaBeat END

;---------------------------------------------- amountOfDamka

amountOfDamka PROC

push rbp

mov rbp, rsp

mov rsi, rcx

xor rax, rax

mov rcx, 64

Metka:

cmp byte ptr[rsi], 2

jle notDamka

inc rax

notDamka:

inc rsi

loop Metka

pop rbp

ret

amountOfDamka ENDP

;---------------------------------------------- amountOfDamka END

END

.DATA

.CODE

Check PROC

mov rax, 0

cmp cl, 0

jl Ending

cmp cl, 7

jg Ending

cmp dl, 0

jl Ending

cmp dl, 7

jg Ending

mov rax, 1

Ending:

ret

Check ENDP

;---------------------------------------------- asmAdd

asmAdd PROC

push rbp

mov rbp, rsp

xor rbx, rbx

mov bl, byte ptr [rdx]

mov al, bl

imul rbx, 4

add rcx, rbx

mov r10b, byte ptr[rbp+48]

mov r11b, byte ptr[rbp+56]

mov byte ptr[rcx], r8b

inc rcx

mov byte ptr[rcx], r9b

inc rcx

mov byte ptr[rcx], r10b

inc rcx

mov byte ptr[rcx], r11b

inc al

mov byte ptr[rdx], al

pop rbp

ret

asmAdd ENDP

;---------------------------------------------- asmAdd END

;---------------------------------------------- FillDamkaMoves

FillDamkaMoves PROC

push rbp

mov rbp, rsp

;x - r8

;y - r9

;x0 - r10

;y0 - r11

;\*len - r12

;len - r13

;temp field - r15

;field - rsi

;AllMoves - rdi

mov rsi, rcx

mov rdi, rdx

mov r12, r8

mov r13b, byte ptr[r8]

mov r8, r9

mov r9, [rbp+48]

and r8, 0FFh

and r9, 0FFh

and r13, 0FFh

mov r10, r8

mov r11, r9

mov rax, 8

mul r8b

add rax, r9

add rsi, rax

mov rax, 4

mul r13b

add rdi, rax

mov r15, rsi

loop1:

inc r8

inc r9

mov rcx, r8

mov rdx, r9

call Check

test rax, rax

jz next1

add rsi, 9

cmp byte ptr[rsi], 0

jnz next1

mov byte ptr[rdi], r10b

inc rdi

mov byte ptr[rdi], r11b

inc rdi

mov byte ptr[rdi], r8b

inc rdi

mov byte ptr[rdi], r9b

inc rdi

inc r13

jmp loop1

next1:

mov r8, r10

mov r9, r11

mov rsi, r15

loop2:

dec r8

inc r9

mov rcx, r8

mov rdx, r9

call Check

test rax, rax

jz next2

sub rsi, 7

cmp byte ptr[rsi], 0

jnz next2

mov byte ptr[rdi], r10b

inc rdi

mov byte ptr[rdi], r11b

inc rdi

mov byte ptr[rdi], r8b

inc rdi

mov byte ptr[rdi], r9b

inc rdi

inc r13

jmp loop2

next2:

mov r8, r10

mov r9, r11

mov rsi, r15

loop3:

inc r8

dec r9

mov rcx, r8

mov rdx, r9

call Check

test rax, rax

jz next3

add rsi, 7

cmp byte ptr[rsi], 0

jnz next3

mov byte ptr[rdi], r10b

inc rdi

mov byte ptr[rdi], r11b

inc rdi

mov byte ptr[rdi], r8b

inc rdi

mov byte ptr[rdi], r9b

inc rdi

inc r13

jmp loop3

next3:

mov r8, r10

mov r9, r11

mov rsi, r15

loop4:

dec r8

dec r9

mov rcx, r8

mov rdx, r9

call Check

test rax, rax

jz next4

sub rsi, 9

cmp byte ptr[rsi], 0

jnz next4

mov byte ptr[rdi], r10b

inc rdi

mov byte ptr[rdi], r11b

inc rdi

mov byte ptr[rdi], r8b

inc rdi

mov byte ptr[rdi], r9b

inc rdi

inc r13

jmp loop4

next4:

mov byte ptr [r12], r13b

pop rbp

ret

FillDamkaMoves ENDP

;---------------------------------------------- FillDamkaMoves END

END

#pragma once+

#include <SFML/Graphics.hpp>

#include "Engine.h"

using namespace sf;

#define schar signed char

typedef mytype Coord[4];

extern Font font;

const int fontSize = 24;

const int tileSize = 100;

class TLabel {

Text text;

bool visible;

public:

TLabel();

void setText(std::string txt);

void setPos(int x, int y);

void draw(RenderWindow& win);

void setVisible(bool toSet);

void setThickness(int thick);

void setFontSize(int fontSize);

void setColor(Color color);

void setOutlineColor(Color color);

};

class TObject

{

protected:

RectangleShape background;

int x, y, width, height;

TObject();

bool visible;

public:

void setVisible(bool toSet);

virtual ~TObject() {}

virtual void setPos(int tx, int ty);

virtual void setSize(int twidth, int theight);

void setColor(Color color);

void setThickness(int thickness);

virtual void draw(RenderWindow& win);

};

class TClickable : public TObject {

protected:

Vector2f pressPos;

virtual void onPress() = 0;

virtual void onRelease() = 0;

public:

bool isPressed(Vector2f& pos);

TClickable();

};

class TButton : public TClickable {

Text text;

int len;

void normText();

public:

TButton();

void setText(std::string toSet);

void setPos(int tx, int ty) override;

void setSize(int twidth, int theight);

void onPress();

void onRelease();

void draw(RenderWindow& win);

};

class TChoice : public TClickable {

bool isSelected;

RectangleShape in;

public:

TChoice();

void setPos(int tx, int ty) override;

void setSize(int twidth, int theight);

void onPress();

void onRelease();

void draw(RenderWindow& win);

void setStatus(bool status);

bool getStatus();

};

class TBar : public TObject {

protected:

RectangleShape first;

float value;

int posX;

RectangleShape second;

Text text;

virtual void setValue(float toSet) = 0;

TBar();

public:

void setFirstColor(Color color);

void setSecondColor(Color color);

void draw(RenderWindow& win);

};

class TProgressBar : public TBar {

inline void setWidth();

inline void setTextPosition();

inline void setString();

public:

TProgressBar();

void setPos(int tx, int ty) override;

void setSize(int twidth, int theight) override;

void setValue(float toSet);

};

class TAssessBar : public TBar {

inline void setHeight();

inline void setTextColor();

inline void setTextPosition();

inline void setString();

bool isFlip;

public:

TAssessBar();

void setPos(int tx, int ty) override;

void setSize(int twidth, int theight) override;

void setValue(float toSet);

void flip();

};

class TCommentSection : TObject{

std::vector<Text> vText;

std::vector<char> values;

public:

TCommentSection();

void setPos(int x0, int y0) override;

void setValues(std::vector<MoveData>& vdata);

void draw(RenderWindow& win);

};

class TClock : public TObject {

volatile bool gameIsGoing;

volatile bool yourTurn;

sf::Thread\* thread;

int value;

Text text;

std::string getStringTime(int seconds);

void tictac();

public:

TClock();

void update(int seconds);

void start();

void pause();

void draw(RenderWindow& win);

void setPos(int tx, int ty) override;

void release();

void stop();

};

class TBoard : TObject {

bool flipped;

TField field = {};

bool red[8][8] = {};

mytype x1, y1, x2, y2;

MOVE\_STATUS comment;

Texture forced, best, good, inac, blunder;

bool isCaptured;

mytype cx = -1, cy = -1;

public:

void capture(int posx, int posy);

void uncatch();

void redReset();

TBoard();

void setPos(int x0, int y0) override;

void setField(TField& toSet);

void getCoord(Vector2f start, Vector2f end, mytype\* coord);

void redSet(mytype x1, mytype y1, mytype x2, mytype y2);

void redReset(mytype x1, mytype y1, mytype x2, mytype y2);

void flip();

void draw(RenderWindow& win, int posx, int posy);

void setComment(MOVE\_STATUS comment, mytype x1, mytype y1, mytype x2, mytype y2);

};

class GameController {

protected:

Engine engine;

MOVE\_TYPE type;

mytype x, y, vector;

bool locked = false;

void getData(MoveData& source);

void setData(MoveData& dest);

public:

TField field;

float assess;

int curr, head;

bool turn;

std::vector<MoveData> gameMoves;

GameController();

virtual MOVE\_RESULT PlayerMove(mytype x1, mytype y1, mytype x2, mytype y2);

MOVE\_RESULT EngineMove(mytype depth);

void getPrev();

void getNext();

void getCurr();

};

class AnalysicsController {

bool turn;

Engine engine;

MOVE\_TYPE type;

mytype x, y, vector;

int curr, head;

void getData(MoveData& source);

public:

TField field;

float assess;

MOVE\_STATUS comment;

mytype x1, y1, x2, y2;

std::vector<MoveData> gameMoves;

AnalysicsController();

void evaluate(int index, int depth);

void setMoves(std::vector<MoveData>& tgameMoves);

void getPrev();

void getNext();

void getCurr();

};

class TInput : public TClickable {

bool isSelected;

Text text;

int limit;

bool checkchar(char toCheck);

public:

bool letters = false;

bool dot = false;

bool numbers = false;

TInput();

void onPress() override;

void onKeyPress(char pressed);

void onRelease() override;

void draw(RenderWindow& win) override;

void setPos(int x0, int y0) override;

void setSize(int w, int h) override;

void setLimit(int lim);

std::string getText();

};

class TWait {

const float s3 = 1.73205;

CircleShape mas[6];

int current = 0;

int radius = 100;

int x = 0, y = 0;

bool visible = true;

void setPos();

public:

TWait();

void setNext();

void setPos(int tx, int ty);

void setRadius(int tradius);

void setVisible(bool toSet);

void draw(RenderWindow& win);

};

#pragma once

#include "PossibleMoves.h"

#include <vector>

enum MOVE\_STATUS {

FORCED,

BEST,

GOOD,

INACCURACY,

BLUNDER,

};

typedef struct \_MoveData {

TField field;

TField oldfield;

MOVE\_TYPE type;

mytype x, y, vector;

mytype coord[4];

float assess;

bool turn;

MOVE\_STATUS comment;

} MoveData;

enum MOVE\_RESULT {

INVALID\_COORD,

ONE\_MORE,

SUCCESS,

WIN,

DRAW,

LOSE,

};

class Engine {

private:

mytype find(mytype x1, mytype y1, mytype x2, mytype y2);

void fill(TField& field, MOVE\_TYPE type, mytype x, mytype y, mytype vector, bool turn, int depth);

void fill(TField& field, MOVE\_TYPE type, mytype x, mytype y, mytype vector, bool turn);

float mmAB(TField& field, mytype x, mytype y, mytype vector, int depth, float alpha, float beta, bool turn);

float mmAB(TField& field, int depth, float alpha, float beta, bool turns);

public:

std::vector<MoveData> moves;

MOVE\_RESULT PlayerMove(MoveData& data);

MOVE\_RESULT EngineMove(MoveData& data, mytype depth);

void evaluate(MoveData& data, mytype depth);

};

#pragma once

#include <SFML/Network.hpp>

#include "controls.h"

extern Font font;

extern Image icon;

extern bool open;

extern bool turn;

extern bool pvp;

extern int depth;

extern TcpSocket socket;

extern TcpListener listener;

class TAnalysicsForm {

RectangleShape background;

RenderWindow& win;

TButton exitB;

TBoard board;

TAssessBar bar;

TButton flipB;

TProgressBar pbar;

AnalysicsController control;

TCommentSection section;

void drawprogress();

void draw();

public:

TAnalysicsForm(RenderWindow& renwin, std::vector<MoveData>& data);

void poll();

};

class TStartForm {

RenderWindow win;

RectangleShape background;

std::vector<TLabel> vLabel;

std::vector<TChoice> vChoice;

std::vector<TInput> vInput;

TButton startB, exitB;

int masDepth[4] = { 4, 8, 10, 12 };

void draw();

public:

TStartForm();

void poll();

};

class TEngineForm {

bool LP = false;

bool LR = false;

RenderWindow win;

RectangleShape background;

TButton exitB, flipB, analysicsB;

TLabel resultLabel;

TBoard board;

GameController control;

Thread\* engineThread;

void draw(int posx, int posy);

void engineMove();

public:

TEngineForm();

~TEngineForm();

void poll();

};

class TPvpForm {

bool turn = true;

bool LP = false;

bool LR = false;

bool opponentMoveReceived = false;

bool connected = false;

std::vector<int> vMoves;

TClock clock1, clock2;

RectangleShape background;

RenderWindow win;

TButton exitB, flipB, analysicsB;

TBoard board;

TWait wait;

GameController control;

TLabel lDraw, lLose, lWin;

enum Type {

INIT,

MOVEREQ,

MOVEREPLY,

};

void addMove(mytype x1, mytype y1, mytype x2, mytype y2);

void sendMove(mytype x1, mytype y1, mytype x2, mytype y2);

void receive();

void draw(int posx, int posy);

void loading();

public:

TPvpForm();

void poll();

};

#pragma once

#define mytype signed char

typedef mytype TField[8][8];

typedef mytype TAM[100][4];

typedef enum MOVE\_TYPE {

MOVE,

BEAT

};

extern "C" mytype GetMode(mytype, mytype, mytype, mytype, mytype);

extern "C" bool CheckCoord(mytype, mytype);

extern "C" void BInit(TField&);

extern "C" void BCopy(TField&, TField&);

extern "C" void Move(TField&, mytype, mytype, mytype, mytype);

extern "C" void Beat(TField&, mytype, mytype, mytype, mytype);

extern "C" bool SMCheck(TField&, mytype, mytype, mytype, mytype);

extern "C" bool SBCheck(TField&, mytype, mytype, mytype, mytype, bool);

extern "C" int getAssess(TField&);

extern "C" bool NTBDamka(TField&, mytype, mytype, bool, mytype);

extern "C" void DamkaBeat(TField&, mytype, mytype, mytype, mytype, mytype);

extern "C" mytype amountOfDamka(TField&);

bool NTBDamkaOneMore(TField& field, mytype x, mytype y, bool turn, mytype mode);

bool PMFill(TField&, MOVE\_TYPE, TAM&, mytype\*, bool type, mytype x, mytype y, mytype vector);

#include <SFML/Network.hpp>

#include <SFML/System.hpp>

#include "GameController.h"

#include <iostream>

#include <vector>

#include <functional>

#include <thread>

#define mytype unsigned char

using std::cout;

const unsigned short PORT = 5000;

using namespace sf;

using std::endl;

using std::cout;

enum Type {

INIT,

MOVEREQ,

DRAWREQ

};

enum GAME\_RESULT {

FIRST\_WIN,

SECOND\_WIN,

THERE\_DRAW,

GAME\_IS\_GOING

};

struct Client {

int id;

TcpSocket\* socket;

bool draw;

int time;

};

std::vector<Client> clients;

typedef struct {

int\* time1;

int\* time2;

bool\* turn;

} ClockData;

void handleTime(ClockData& data) {

while (true) {

sf::sleep(sf::milliseconds(1000));

if (\*(data.turn)) {

(\*(data.time1))--;

if (\*(data.time1) == 0) {

break;

}

}

else {

(\*(data.time2))--;

if (\*(data.time2) == 0) {

break;

}

}

}

}

static GAME\_RESULT handleMove(GameController& control, Client& tfirst, Client& tsecond) {

Client first = tfirst;

Client second = tsecond;

bool tturn = control.turn;

if (!tturn) {

auto temp = first;

first = second;

second = temp;

}

Packet packet;

Packet packet1;

Packet packet2;

mytype x1, y1, x2, y2;

uint8\_t type;

uint8\_t result;

if (first.socket->receive(packet) == Socket::Done) {

packet >> type;

packet1.clear();

packet2.clear();

if (type == DRAWREQ) {

first.draw = true;

if (first.draw && second.draw) {

second.socket->send(packet);

return THERE\_DRAW;

}

else {

packet2 << (uint8\_t)DRAWREQ;

second.socket->send(packet);

}

}

else {

first.draw = false;

second.draw = false;

packet >> x1 >> y1 >> x2 >> y2;

if (result = control.PlayerMove(x1, y1, x2, y2), result != INVALID\_COORD) {

if (result == WIN) {

packet1 << (uint8\_t)MOVEREQ << (uint8\_t)WIN << x1 << y1 << x2 << y2 << 600 << 600;

packet2 << (uint8\_t)MOVEREQ << (uint8\_t)LOSE << x1 << y1 << x2 << y2 << 600 << 600;

}

else {

if (tturn) {

packet1 << (uint8\_t)MOVEREQ << result << x1 << y1 << x2 << y2 << first.time << second.time;

packet2 << (uint8\_t)MOVEREQ << result << x1 << y1 << x2 << y2 << second.time << first.time;

}

else {

packet1 << (uint8\_t)MOVEREQ << result << x1 << y1 << x2 << y2 << second.time << first.time;

packet2 << (uint8\_t)MOVEREQ << result << x1 << y1 << x2 << y2 << first.time << second.time;

}

}

first.socket->send(packet1);

second.socket->send(packet2);

}

else {

if (tturn) {

packet1 << (uint8\_t)MOVEREQ << (uint8\_t)INVALID\_COORD << x1 << y1 << x2 << y2 << first.time << second.time;

}

else {

packet1 << (uint8\_t)MOVEREQ << (uint8\_t)INVALID\_COORD << x1 << y1 << x2 << y2 << second.time << first.time;

}

first.socket->send(packet);

}

}

}

return GAME\_IS\_GOING;

}

static void handleGame(int index1, int index2) {

Packet packet;

Client first = clients[index1];

Client second = clients[index2];

first.draw = false;

first.time = 600;

second.draw = false;

second.time = 600;

GameController control;

ClockData clockdata;

clockdata.time1 = &first.time;

clockdata.time2 = &second.time;

clockdata.turn = &control.turn;

Thread timeThread(&handleTime, clockdata);

timeThread.launch();

packet << (uint8\_t)INIT << (uint8\_t)1 << 600;

first.socket->send(packet);

packet.clear();

packet << (uint8\_t)INIT << (uint8\_t)0 << 600;

second.socket->send(packet);

packet.clear();

while (handleMove(control, first, second)) { };

timeThread.terminate();

delete first.socket;

delete second.socket;

}

int main() {

TcpSocket\* socket = new TcpSocket;

TcpListener listener;

if (listener.listen(PORT) != Socket::Done) {

cout << "Failed to bind port " << PORT << std::endl;

return 1;

}

cout << "Server started, listening on port " << PORT << std::endl;

while (true) {

if (listener.accept(\*socket) == Socket::Done) {

Client newClient;

newClient.id = clients.size();

newClient.socket = socket;

clients.push\_back(newClient);

cout << "Accepted new client: " << socket->getRemoteAddress() << " " << socket->getRemotePort() << endl;

if (clients.size() % 2 == 0) {

std::thread(&handleGame, clients.size() - 2, clients.size() - 1).detach();

}

socket = new TcpSocket;

}

}

return 0;

}

#include <SFML/Network.hpp>

#include <SFML/System.hpp>

#include "GameController.h"

#include <iostream>

#include <vector>

#include <functional>

#include <thread>

#include <condition\_variable>

#include <atomic>

#define mytype unsigned char

using std::cout;

using std::endl;

using std::mutex;

using std::condition\_variable;

using std::atomic\_bool;

const unsigned short PORT = 5000;

using namespace sf;

#define TIME 600

enum Type {

INIT,

MOVEREQ,

DRAWREQ,

RESIGN

};

enum GAME\_RESULT {

FIRST\_WIN,

SECOND\_WIN,

FIRST\_DISCONNECT,

SECOND\_DISCONNECT,

THERE\_DRAW,

GAME\_IS\_GOING,

TIMEOUT\_RESULT

};

struct Client {

int id;

TcpSocket\* socket;

bool draw;

int time;

};

std::vector<Client> clients;

typedef struct {

int\* time1;

int\* time2;

bool\* turn;

atomic\_bool\* gameActive;

condition\_variable\* cv;

mutex\* mtx;

Client\* first;

Client\* second;

GameController\* control;

} ClockData;

void handleTime(ClockData& data) {

while (\*(data.gameActive)) {

std::this\_thread::sleep\_for(std::chrono::seconds(1));

std::unique\_lock<mutex> lock(\*(data.mtx));

if (\*(data.turn)) {

(\*(data.time1))--;

if (\*(data.time1) == 0) {

\*(data.gameActive) = false;

data.cv->notify\_one();

Packet packet;

packet << (uint8\_t)MOVEREQ << (uint8\_t)LOSE << 0 << 0 << 0 << 0;

data.first->socket->send(packet);

packet.clear();

packet << (uint8\_t)MOVEREQ << (uint8\_t)WIN << 0 << 0 << 0 << 0;

data.second->socket->send(packet);

break;

}

}

else {

(\*(data.time2))--;

if (\*(data.time2) == 0) {

\*(data.gameActive) = false;

data.cv->notify\_one();

Packet packet;

packet << (uint8\_t)MOVEREQ << (uint8\_t)WIN << 0 << 0 << 0 << 0;

data.first->socket->send(packet);

packet.clear();

packet << (uint8\_t)MOVEREQ << (uint8\_t)LOSE << 0 << 0 << 0 << 0;

data.second->socket->send(packet);

break;

}

}

}

}

void resignOrDraw(Client& first, Client& second, std::atomic<bool>& secondResign, std::atomic<bool>& theredraw) {

Packet packet;

uint8\_t status;

while (second.socket->receive(packet) == Socket::Done) {

packet >> status;

if (status == DRAWREQ) {

std::cout << "second draw" << endl;

second.draw = true;

if (!first.draw) {

packet.clear();

packet << (uint8\_t)DRAWREQ;

first.socket->send(packet);

}

else {

theredraw = true;

packet.clear();

packet << (uint8\_t)MOVEREQ << (uint8\_t)DRAW << 0 << 0 << 0 << 0;

first.socket->send(packet);

second.socket->send(packet);

}

}

else if (status == RESIGN) {

std::cout << "second resign" << endl;

secondResign = true;

Packet packet1, packet2;

packet1 << (uint8\_t)MOVEREQ << (uint8\_t)WIN << 0 << 0 << 0 << 0;

packet2 << (uint8\_t)MOVEREQ << (uint8\_t)LOSE << 0 << 0 << 0 << 0;

first.socket->send(packet1);

second.socket->send(packet2);

}

}

}

GAME\_RESULT handleMove(GameController& control, Client& tfirst, Client& tsecond) {

Packet packet;

mytype x1, y1, x2, y2;

uint8\_t type;

uint8\_t result;

Client first = tfirst, second = tsecond;

if (!control.turn) {

std::swap(first, second);

}

std::atomic<bool> secondResign(false);

std::atomic<bool> theredraw(false);

std::cout << "receive thread to second" << endl;

sf::Thread thread(std::bind(&resignOrDraw, std::ref(first), std::ref(second), std::ref(secondResign), std::ref(theredraw)));

thread.launch();

if (first.socket->receive(packet) == Socket::Done) {

std::cout << "Packet from first" << endl;

thread.terminate();

if (secondResign) {

return (control.turn ? SECOND\_DISCONNECT : FIRST\_DISCONNECT);

}

if (theredraw) {

return THERE\_DRAW;

}

packet >> type;

if (type == RESIGN) {

Packet packet1, packet2;

packet1 << (uint8\_t)MOVEREQ << (uint8\_t)LOSE << 0 << 0 << 0 << 0;

packet2 << (uint8\_t)MOVEREQ << (uint8\_t)WIN << 0 << 0 << 0 << 0;

first.socket->send(packet1);

second.socket->send(packet2);

return (control.turn ? SECOND\_WIN : FIRST\_WIN);

}

else if (type == DRAWREQ) {

if (control.turn) {

tfirst.draw = true;

}

else {

tsecond.draw = true;

}

if (second.draw) {

packet.clear();

packet << (uint8\_t)MOVEREQ << (uint8\_t)DRAW << 0 << 0 << 0 << 0;

first.socket->send(packet);

second.socket->send(packet);

return THERE\_DRAW;

}

else {

packet.clear();

packet << (uint8\_t)DRAWREQ;

second.socket->send(packet);

}

}

else {

tfirst.draw = false;

tsecond.draw = false;

packet >> x1 >> y1 >> x2 >> y2;

result = control.PlayerMove(x1, y1, x2, y2);

Packet packet1, packet2;

if (result != INVALID\_COORD) {

if (result == WIN) {

packet1 << (uint8\_t)MOVEREQ << (uint8\_t)WIN << x1 << y1 << x2 << y2;

packet2 << (uint8\_t)MOVEREQ << (uint8\_t)LOSE << x1 << y1 << x2 << y2;

return (control.turn ? SECOND\_WIN : FIRST\_WIN);

}

else {

packet1 << (uint8\_t)MOVEREQ << result << x1 << y1 << x2 << y2;

packet2 = packet1;

}

first.socket->send(packet1);

second.socket->send(packet2);

}

else {

packet1 << (uint8\_t)MOVEREQ << (uint8\_t)INVALID\_COORD << x1 << y1 << x2 << y2;

first.socket->send(packet1);

}

}

return GAME\_IS\_GOING;

}

thread.terminate();

return (GAME\_RESULT)(3 - control.turn);

}

void handleGame(int index1, int index2) {

Packet packet;

Client first = clients[index1];

Client second = clients[index2];

first.draw = false;

first.time = TIME;

second.draw = false;

second.time = TIME;

GameController control;

ClockData clockdata;

clockdata.time1 = &first.time;

clockdata.time2 = &second.time;

clockdata.turn = &control.turn;

atomic\_bool gameActive(true);

clockdata.gameActive = &gameActive;

condition\_variable cv;

clockdata.cv = &cv;

mutex mtx;

clockdata.mtx = &mtx;

clockdata.first = &first;

clockdata.second = &second;

clockdata.control = &control;

packet << (uint8\_t)INIT << (uint8\_t)1 << TIME;

first.socket->send(packet);

packet.clear();

packet << (uint8\_t)INIT << (uint8\_t)0 << TIME;

second.socket->send(packet);

packet.clear();

Thread timeThread(&handleTime, std::ref(clockdata));

GAME\_RESULT status = GAME\_IS\_GOING;

while (status == GAME\_IS\_GOING) {

{

std::lock\_guard<mutex> lock(mtx);

status = handleMove(control, first, second);

}

std::unique\_lock<mutex> lock(mtx);

cv.wait\_for(lock, std::chrono::milliseconds(1), [&gameActive] { return !gameActive; });

if (!gameActive) {

status = TIMEOUT\_RESULT;

}

}

if (status == FIRST\_DISCONNECT) {

packet.clear();

packet << (uint8\_t)MOVEREQ << (uint8\_t)WIN << 0 << 0 << 0 << 0;

second.socket->send(packet);

}

else if (status == SECOND\_DISCONNECT) {

packet.clear();

packet << (uint8\_t)MOVEREQ << (uint8\_t)WIN << 0 << 0 << 0 << 0;

first.socket->send(packet);

}

timeThread.terminate();

delete first.socket;

delete second.socket;

}

int main() {

TcpSocket\* socket = new TcpSocket;

TcpListener listener;

if (listener.listen(PORT) != Socket::Done) {

return 1;

}

cout << "Server started, listening on port " << PORT << std::endl;

while (true) {

if (listener.accept(\*socket) == Socket::Done) {

Client newClient;

newClient.id = clients.size();

newClient.socket = socket;

clients.push\_back(newClient);

cout << "Accepted new client: " << socket->getRemoteAddress() << " " << socket->getRemotePort() << endl;

if (clients.size() % 2 == 0) {

std::thread(&handleGame, clients.size() - 2, clients.size() - 1).detach();

}

socket = new TcpSocket;

}

}

return 0;

}

ВЕДОМОСТЬ

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КР 1-40 01 01 115 ПЗ | | | | Пояснительная записка | | | | 63 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР 251001 115 СП | | | | "Шашки онлайн"  А1, схема программы, чертеж | | | | Формат А1 | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КР 1-40 01 01 115 ПЗ | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Шашки онлайн  Ведомость курсовой работы |  | | | | Лист | Листов |
| Разраб. | | Кривицкий Ф.Ю. |  |  | Т |  | |  | 63 | 63 |
| Пров. | | Красковский П.Н. |  |  | Кафедра ПОИТ  гр. 251001 | | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |