МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: “Компьютерная игра эндшпиль

«Король, ферзь - Король ладья, 2 пешки»”

**Инв.№ подл.**

**Подп. и дата**

**Взам.инв. №**

**Инв.№ дубл.**

**Подп. и дата**

Р.02069337. 23/685 11 ПЗ-01

Листов 67

Руководитель разработки:

к.т.н, доцент кафедры "ИВК"

Шишкин Вадим Викторинович

« » 2024 г.

Исполнитель:

студент гр. ИСТбд-21

Казаров Дмитрий Сергеевич

« » 2024 г.

2024

Содержание

[**ТЕХНИЧЕСКОЕ ЗАДАНИЕ** 3](#_Toc185873834)

[**Пояснительная записка** 8](#_Toc185873835)

[**Руководство программиста** 15](#_Toc185873836)

[**Тестовая документация** 20](#_Toc185873837)

[**Листинг кода** 29](#_Toc185873838)

[**Заключение** 56](#_Toc185873839)

[**Источники, использованные при разработке** 58](#_Toc185873840)

[**ПРИЛОЖЕНИЕ 1** 59](#_Toc185873841)

[**ПРИЛОЖЕНИЕ 2** 60](#_Toc185873842)

[**ПРИЛОЖЕНИЕ 3** 62](#_Toc185873843)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на курсовую работу

по дисциплине «Алгоритмы и структуры данных»

Тема: “Компьютерная игра эндшпиль

«Король, ферзь - Король ладья, 2 пешки»”

**Инв.№ подл.**

**Подп. и дата**

**Взам.инв. №**

**Инв.№ дубл.**

**Подп. и дата**

Р.02069337. 23/685 11 ТЗ-01

Листов 6

Руководитель разработки:

к.т.н, доцент кафедры "ИВК"

Шишкин Вадим Викторинович

« » 2024 г.

Исполнитель:

студент гр. ИСТбд-21

Казаров Дмитрий Сергеевич

« » 2024 г.

2024

|  |  |
| --- | --- |
|  | УТВЕРЖДЕНО  Распоряжением по кафедре ИВК  №\_\_\_ от «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2024 |

# **ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

на выполнение курсовой работы

по дисциплине «Алгоритмы и структуры данных»

Специальность 09.03.02 – «Информационные системы и технологии»

Тема: «**Разработка компьютерной игра эндшпиль “Король, ферзь-Король ладья, 2 пешки”**»

**1. Введение**

Целью разработки программного обеспечения является создание симулятора шахматного эндшпиля "Король, ферзь - Король, ладья, 2 пешки", предназначенного для обучения пользователей шахматным стратегиям и тактикам в данной позиции.

**2. Основания для разработки**

Разработка осуществляется на основании учебного плана дисциплины «Алгоритмы и структуры данных» направления подготовки «Информационные системы и технологии» в соответствии с требованиями кафедры "Информационные системы" Ульяновского государственного технического университета.

**3. Назначение разработки**

Программа предназначена для визуализации и симуляции шахматного эндшпиля "Король, ферзь - Король, ладья, 2 пешки". Программное обеспечение должно обеспечивать учебно-тренировочный процесс и развитие навыков шахматной игры в данном эндшпиле.

**4. Требования к программе**

**4.1 Требования к функциональным характеристикам**

**4.1.1 Программа должна обеспечивать выполнение следующих функций:**

Загрузка и визуализация окна, отрисовка изображений и текстовых уведомлений, воспроизведение звуковых эффектов с помощью библиотеки **PyGame***;*

Форма регистрации и авторизации для допуска пользователя к игровому процессу. Хеширование паролей пользователей для обеспечения безопасности данных и защиты учетных записей от несанкционированного доступа, используя модуль **HashLib**;

Отображение шахматной доски размером 8х8 клеток. Генерация случайных чисел для случайного и точного размещения шахматных фигур данного эндшпиля на доске, с помощью модуля **random**.

Разработка режимов игры «Игрок против игрока» и «Игрок против ПК», с использованием алгоритма **MiniMax с альфа-бета отсечением.** Диапазон поиска ходов 3.

Реализация перемещения фигур в строгом соответствии с правилами шахмат. Определение окончания игры согласно общепринятым правилам игры (мат, пат, ничья);

Работа с файловой системой, включая проверку существования директорий, создание папок для сохранений пользователей и удаление файлов, с использованием модуля **os** для управления файловыми операциями.

Сохранение и загрузка состояния игры, управление данными пользователей, такими как учетные записи и настройки, с помощью модуля **json** для обеспечения надежного хранения данных.

Управление завершением работы приложения и обработка системных выходов с использованием модуля **sys** для обеспечения корректного завершения игры.

**4.1.2 Организация входных и выходных данных**

Входные данные: начальная позиция фигур, действия пользователя (ходы фигур).

Выходные данные: визуализация текущего состояния доски, сохранение состояния игры в файл.

**4.2 Требования к надежности**

Программа должна корректно обрабатывать некорректные действия пользователя (например, невозможность совершения хода), выводить соответствующие уведомления и обеспечивать защиту от сбоев при сохранении и загрузке файлов игры.

**4.3 Требования к составу и параметрам технических средств**

Программа должна функционировать на персональных компьютерах с операционными системами Windows 7 и выше, минимальный объем оперативной памяти — 2 ГБ, поддержка Python версии 3.8 и выше.

**4.4 Требования к информационной и программной совместимости**

* Программа должна быть совместима с операционой системой Windows.
* Программа должна быть написана на Python с использованием библиотеки PyGame для интерфейса.

**4.5 Условия и срок хранения**

Обеспечение свободного доступа к проекту в репозитории до окончания срока учебы.

**5. Требования к программной документации**

1. В состав программной документации должны входить:
2. Техническое задание (составленное в соответствии с ГОСТ 19.201-78);
3. Руководство пользователя;
4. Пояснительная записка;
5. Тестовая документация (чек-листы, тест-кейсы, отчет о тестировании).

**6. Стадии и этапы разработки**

1. Анализ требований к проекту (срок выполнения: 05.10.2024 — 20.10.2024).
2. Разработка технического задания (срок выполнения: 21.10.2024 — 30.10.2024).
3. Проектирование архитектуры программы (срок выполнения: 31.10.2024 — 10.11.2024).
4. Реализация программы (срок выполнения: 11.11.2024 — 30.11.2024).
5. Тестирование и исправление ошибок (срок выполнения: 01.12.2024 — 05.12.2024).
6. Разработка эксплуатационной документации (срок выполнения: 05.12.2024 — 20.12.2024).
7. Сдача программного продукта и приемка (срок выполнения: 27.12.2024).

**6.1 Распределение обязанностей в команде разработчиков**

**•** Дизайнер, Full-Stack разработчик проекта: Казаров Дмитрий Сергеевич.

**7. Порядок контроля и приемки**

**7.1 Контроль осуществляется посредством проведения тестов, проверяющих функциональность и надежность программы по следующим критериям:**

Контроль и приемка программного обеспечения осуществляются в соответствии с планом тестирования, который включает проверку следующих функциональных возможностей:

1. Корректная работа системы регистрации, авторзиации, шифрования, а так же процесса игры, включая отображение доски и всех фигур, их перемещение;
2. Функции сохранения и загрузки игры, уникальной для каждого пользователя;
3. Проверка всех возможных сценариев окончания игры (мат, пат, ничья).

**7.2 Приемка проекта осуществляется в рамках защиты курсовых проектов, где проводится повторное тестирование и оценка работы.**

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

на курсовую работу

по дисциплине «Алгоритмы и структуры данных»

Тема: “Компьютерная игра эндшпиль

«Король, ферзь - Король ладья, 2 пешки»”

**Инв.№ подл.**

**Подп. и дата**

**Взам.инв. №**

**Инв.№ дубл.**

**Подп. и дата**

Р.02069337. 23/685 11 ПЗ-02

Листов 8

Руководитель разработки:

к.т.н, доцент кафедры "ИВК"

Шишкин Вадим Викторинович

« » 2024 г.

Исполнитель:

студент гр. ИСТбд-21

Казаров Дмитрий Сергеевич

« » 2024 г.

2024

# **Пояснительная записка**

на выполнение курсовой работы

по дисциплине «Алгоритмы и структуры данных»

Специальность 09.03.02 – «Информационные системы и технологии»

Тема: «**Разработка компьютерной игра эндшпиль “Король, ферзь-Король ладья, 2 пешки”**»

**Введение**

Разрабатываемое приложение представляет собой шахматную игру с фокусом на эндшпильные позиции, разработанную с использованием языка программирования Python и библиотеки Pygame для графического интерфейса и работы со звуком.

Целью приложения является обучение игроков эндшпильным позициям через практику игры против компьютера или другого игрока. Программа демонстрирует навыки программирования и создания пользовательских интерфейсов для реализации логических игр.

**Основания для разработки**

Задание для курсового проекта по учебному плану направления «Информационные системы и технологии».

**Постановка задачи**

Цель разработки заключается в создании шахматной игры с фокусом на эндшпильные позиции, которая включает в себя:

- Регистрация и авторизация в системе;

- Глубокий выбор настроек игры:

-- Выбор режима игры: против компьютера или другого игрока;

-- Выбор начального набора фигур и выбор цвета;

- Сохранение и загрузка партий;

- Интерактивная и динамичная инструкция к игре.

**Структура программы**

***Классы****:*

1. **Button**: класс для создания и управления кнопками интерфейса

2. **Piece**: класс для создания и управления шахматными фигурами

3. **ChessBoard**: основной класс для управления игровым процессом

***Основные функции****:*

1. **load\_images** - загрузка изображений фигур

2. **setup\_pieces** - расстановка фигур на доске

3. **draw** - отрисовка игрового поля и интерфейса

4. **handle\_events** - обработка событий

5. **get\_valid\_moves** - получение возможных ходов

6. **is\_check/is\_checkmate/is\_stalemate** - проверка шаха/мата/пата

7. **save\_game/load\_game** - сохранение и загрузка игры

8. **authenticate/register** - авторизация и регистрация пользователей

***Функциональное назначение:***

1. Управление игровым процессом

2. Проверка правильности ходов

3. Реализация искусственного интеллекта

4. Сохранение/загрузка игр

5. Авторизация пользователей

6. Отображение подсказок

7. Воспроизведение звуков

***Проектная часть***

***1.1 Математические методы***

- Алгоритм минимакс с альфа-бета отсечением для ИИ

- Оценочная функция позиции

- Хеширование позиций для транспозиционной таблицы

***1.2 Архитектура и алгоритмы***

***1.2.1 Архитектура***

- Игровое поле - шахматная доска 8x8

- Фигуры - объекты с определенными правилами хода

- Интерфейс - кнопки управления и информационные панели

- Система сохранений - JSON файлы для хранения партий

- Система авторизации - хеширование паролей

***1.2.2 Алгоритмы***

- Алгоритм проверки возможных ходов

- Алгоритм определения шаха/мата/пата

- Алгоритм ИИ на основе минимакса

- Алгоритм оценки позиции

- Алгоритм сохранения/загрузки игры

**Стадии и этапы разработки**

1. Проектирование архитектуры

2. Реализация базовой механики

3. Добавление ИИ

4. Реализация сохранений

5. Добавление авторизации

6. Тестирование и отладка

7. Документирование

**Тестирование**

***Цель тестирования***

Проверка корректности работы всех игровых механик и пользовательского интерфейса.

***Методика тестирования***

Тестирование проводилось методом "черного ящика" с проверкой всех возможных сценариев использования.

***Тестовые сценарии***

1. Проверка интерфейса и правильности ходов фигур

2. Проверка определения шаха/мата/пата

3. Проверка работы ИИ

4. Проверка сохранения/загрузки

5. Проверка авторизации

6. Стресс-тестирование

Выявленные ошибки были исправлены, программа работает стабильно.

**Использованные методы и алгоритмы**

В данном проекте использованы различные методы и алгоритмы, которые обеспечивают функциональность шахматной игры. Одним из ключевых алгоритмов является Минимакс с альфа-бета отсечением, реализованный в методе get\_best\_move. Этот алгоритм представляет собой рекурсивный подход для поиска оптимального хода в играх с полной информацией, таких как шахматы. Он создает дерево возможных ходов на заданную глубину, где максимизирующий игрок (обычно белые) стремится получить максимальную оценку, а минимизирующий игрок (черные) — минимальную. Альфа-бета отсечение позволяет значительно сократить количество рассматриваемых вариантов, пропуская те ветви, которые заведомо хуже уже найденных. Это достигается за счет использования параметров alpha и beta, которые хранят лучшие результаты для максимизирующего и минимизирующего игроков соответственно. Если в процессе поиска оказывается, что beta меньше или равна alpha, то дальнейшее исследование этой ветви становится бессмысленным, и она отсеивается. Применение этого алгоритма выгодно, так как оно позволяет существенно сократить время вычислений, сохраняя при этом точность поиска оптимального хода. Это особенно важно в условиях ограниченного времени, когда необходимо быстро принимать решения.

Эвристическая оценка позиции, реализованная в методе evaluate\_board, играет важную роль в определении лучшего хода. Она оценивает текущую позицию на доске, учитывая материальное преимущество, то есть ценность фигур, а также позиционные факторы, такие как расположение пешек, защита короля, контроль центра и близость фигур к королю противника. Для каждого из этих факторов используется система весов, что позволяет более точно оценивать позиции и принимать более обоснованные решения.

Транспозиционная таблица, представленная в поле transposition\_table, служит для оптимизации процесса поиска. Она кэширует уже вычисленные позиции, сохраняя их хеш и оценку, что позволяет избежать повторных вычислений одинаковых позиций. Это значительно ускоряет процесс поиска, особенно в сложных позициях, где возможны многочисленные повторения.

Метод \_move\_priority отвечает за приоритизацию ходов, что также способствует оптимизации альфа-бета отсечения. Он сортирует ходы по приоритету, учитывая такие факторы, как взятие фигур противника, продвижение пешек, контроль центра и атака на короля противника. Это позволяет сначала рассматривать наиболее перспективные ходы, что может привести к более быстрому отсечению менее выгодных вариантов.

Алгоритмы проверки правил, такие как is\_valid\_move, is\_check, is\_checkmate, is\_stalemate и is\_piece\_between, обеспечивают корректность игры, проверяя правильность ходов и определяя такие состояния, как шах, мат и пат. Это гарантирует, что игра будет проходить в соответствии с правилами шахмат.

Система аутентификации, использующая хеширование паролей с помощью SHA-256 и сохранение учетных данных в формате JSON, обеспечивает безопасность пользовательских данных. Она позволяет проверять корректность логина и пароля, защищая учетные записи от несанкционированного доступа.

Система сохранений позволяет сохранять состояние игры в формате JSON, включая историю ходов, что дает возможность загружать сохраненные игры и продолжать их в любое время. Это удобно для пользователей, которые хотят вернуться к игре позже.

Графический интерфейс, реализованный с использованием библиотеки pygame, обеспечивает визуализацию игры. Он включает в себя систему кнопок и меню, анимации и звуковые эффекты, а также отображение подсказок и сообщений, что делает игру более интерактивной и приятной для пользователя.

Искусственный интеллект в игре представлен различными уровнями сложности, которые определяются глубиной поиска. Он использует эвристическую оценку позиции и оптимизацию с помощью альфа-бета отсечения, а также приоритизацию перспективных ходов, что делает его достаточно сильным соперником для игроков.

Управление состоянием игры включает в себя отслеживание текущего хода, историю ходов и обработку особых ситуаций, таких как шах, мат и пат. Также предусмотрено превращение пешек, что добавляет игре реалистичности и полноты.

**Заключение**

Разработанная программа полностью соответствует поставленным задачам и обеспечивает возможность практики шахматных эндшпилей. Программа имеет удобный пользовательский интерфейс и возможность сохранения прогресса.

В заключение, данный проект представляет собой комплексную реализацию шахматной игры, в которой гармонично сочетаются различные алгоритмы и методы. Использование Минимакс с альфа-бета отсечением обеспечивает эффективный поиск оптимальных ходов, а эвристическая оценка позиции позволяет принимать более обоснованные решения. Транспозиционная таблица и приоритизация ходов значительно ускоряют процесс поиска, делая игру более динамичной. Алгоритмы проверки правил гарантируют соблюдение шахматных норм, а система аутентификации и сохранений обеспечивает безопасность и удобство для пользователей. Графический интерфейс, созданный с помощью pygame, делает игру визуально привлекательной и интерактивной. Искусственный интеллект, адаптируемый под различные уровни сложности, предлагает достойное сопротивление игрокам. Все эти компоненты вместе создают увлекательный и реалистичный опыт игры в шахматы, подходящий как для новичков, так и для опытных игроков.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

РУКОВОДСТВО ПРОГРАММИСТА

на курсовую работу

по дисциплине «Алгоритмы и структуры данных»

Тема: “Компьютерная игра эндшпиль

«Король, ферзь - Король ладья, 2 пешки»”

**Инв.№ подл.**

**Подп. и дата**

**Взам.инв. №**

**Инв.№ дубл.**

**Подп. и дата**

Р.02069337. 23/685 11 РП-01

Листов 6

Руководитель разработки:

к.т.н, доцент кафедры "ИВК"

Шишкин Вадим Викторинович

« » 2024 г.

Исполнитель:

студент гр. ИСТбд-21

Казаров Дмитрий Сергеевич

« » 2024 г.

2024

# **Руководство программиста**

на выполнение курсовой работы

по дисциплине «Алгоритмы и структуры данных»

Специальность 09.03.02 – «Информационные системы и технологии»

Тема: «**Разработка компьютерной игра эндшпиль “Король, ферзь-Король ладья, 2 пешки”**»

**1. Назначение и условия применения программы**

**1.1. Назначение программы**

Программа представляет собой симулятор шахматного эндшпиля. Она визуализирует шахматную доску, фигуры и позволяет играть как против другого игрока, так и против компьютера.

**1.2. Функции программы**

Программа выполняет следующие функции:

• Визуализация шахматной доски и фигур

• Симуляция шахматной партии

• Управление ходами фигур

• Обработка взаимодействия между фигурами

• Настройка параметров игры

• Сохранение и загрузка партий

• Авторизация и регистрация пользователей

**1.3. Условия применения**

Для использования программы необходимы:

• Python версии 3.x

• Библиотеки: **pygame, json, os, random, hashlib, textwrap**

• Графические ресурсы (папка **assets**):

  - Фоновое изображение (**assets/background.jpg)**

  - Изображения фигур (**assets/black\_king.png, assets/white\_king.png и т.д**.)

  - Звуковой файл для хода фигуры (**assets/soundChess.mp3**)

• Файл данных пользователей (**users.json**)

• Папка для сохранений (**saves**)

**2. Характеристика программы**

**2.1. Общие сведения**

Программа реализована с использованием объектно-ориентированного подхода. Основными классами являются ChessBoard и Piece, которые описывают поведение шахматной доски и фигур соответственно.

**2.2. Основные характеристики**

Программа использует графический интерфейс на основе библиотеки pygame. Игра происходит в реальном времени, где отображаются все элементы. Движение фигур и обработка ходов происходят в реальном времени.

**3. Обращение к программе**

**3.1. Класс Piece**

**3.1.1. Метод \_\_init\_\_(self, color, type, position)**

  Назначение: инициализация объекта шахматной фигуры.

**3.1.2. Метод draw(self, screen)**

  Назначение: отрисовка фигуры на экране.

**3.1.3. Метод move\_to(self, new\_position)**

  Назначение: перемещение фигуры на новую позицию.

**3.2. Класс ChessBoard**

**3.2.1. Метод \_\_init\_\_(self)**

  Назначение: инициализация объекта шахматной доски.

**3.2.2. Метод setup\_pieces(self, piece\_set, color)**

  Назначение: установка фигур на доску.

**3.2.3. Метод add\_move\_to\_history(self, piece, start\_pos, end\_pos, *captured*=None)**

  Назначение: добавление хода в историю.

**3.2.4. Метод is\_piece\_between(self, start\_pos, end\_pos)**

  Назначение: проверка наличия фигур между двумя позициями.

**3.2.5. Метод is\_check(self, color)**

  Назначение: проверка состояния шаха для указанного цвета.

**3.2.6. Метод is\_checkmate(self, color)**

  Назначение: проверка состояния мата для указанного цвета.

**3.2.7. Метод is\_stalemate(self, color)**

  Назначение: проверка состояния пата для указанного цвета.

**3.2.8. Метод draw(self, screen)**

  Назначение: отрисовка шахматной доски и фигур на экране.

**3.2.9. Метод promote\_pawn(self, pawn)**

  Назначение: превращение пешки в другую фигуру.

**3.2.10. Метод get\_next\_save\_number(self)**

  Назначение: получение следующего номера сохранения.

**3.2.11. Метод delete\_all\_saves(self)**

  Назначение: удаление всех сохранений текущего пользователя.

**3.2.12. Метод save\_game(self)**

  Назначение: сохранение текущей игры.

**3.2.13. Метод load\_game(self)**

  Назначение: загрузка сохраненной игры.

**3.2.14. Метод show\_message(self, text, *duration*=180)**

  Назначение: отображение сообщения на экране.

**3.2.15. Метод get\_piece\_at(self, position)**

  Назначение: получение фигуры на указанной позиции.

**3.2.16. Метод is\_valid\_move(self, piece, new\_pos)**

  Назначение: проверка корректности хода фигуры.

**3.2.17. Метод get\_valid\_moves(self, piece)**

  Назначение: получение списка допустимых ходов для фигуры.

**3.2.18. Метод handle\_click(self, pos)**

  Назначение: обработка клика мыши на доске.

**3.2.19. Метод handle\_events(self)**

  Назначение: обработка событий (клики мыши, нажатия клавиш).

**3.2.20. Метод clear\_board(self)**

  Назначение: очистка доски от фигур.

**3.2.21. Метод show\_instructions(self)**

  Назначение: отображение инструкции к игре.

**3.2.22. Метод show\_new\_game\_menu(self)**

  Назначение: отображение меню новой игры.

**3.2.23. Метод show\_login\_menu(self)**

  Назначение: отображение меню авторизации.

**3.2.24. Метод show\_register\_menu(self)**

  Назначение: отображение меню регистрации.

**3.2.25. Метод authenticate(self, username, password)**

  Назначение: проверка авторизации пользователя.

**3.2.26. Метод is\_username\_taken(self, username)**

  Назначение: проверка занятости никнейма.

**3.2.27. Метод register(self, username, password)**

  Назначение: регистрация нового пользователя.

**3.2.28. Метод draw\_message(self, screen)**

  Назначение: отрисовка сообщения на экране.

**3.2.29. Метод evaluate\_board(self)**

  Назначение: оценка текущего состояния доски.

**3.2.30. Метод get\_best\_move(self, depth, color, *alpha*=float('-inf'), *beta*=float('inf'),** ***use\_transposition*=True)**

  Назначение: получение лучшего хода для указанного цвета.

**3.2.31. Метод hash\_board(self)**

  Назначение: получение хеша текущего состояния доски.

**3.2.32. Метод \_move\_priority(self, piece, move)**

  Назначение: получение приоритета хода для фигуры.

**3.2.33. Метод get\_best\_promotion(self, piece, move)**

  Назначение: получение лучшей фигуры для превращения пешки.

**4. Входные и выходные данные**

**4.1. Входные данные**

• Параметры игры (режим игры, цвет игрока, набор фигур)

• Пользовательские действия (нажатия кнопок управления, клики мыши)

**4.2. Выходные данные**

• Визуальное отображение шахматной доски и фигур

• Звуковые сигналы при ходах фигур

• Сообщения о состоянии игры (шах, мат, пат и т.д.)

**5. Сообщения**

Программа выводит следующие сообщения:

• Информация о успешной регистрации и авторизации, а так же возможных ошибках. Предупреждения при попытке выполнить недопустимые действия. Сообщения о состоянии игры (шах, мат, пат и т.д.).

**6. Используемые технические средства**

Для работы программы используются следующие библиотеки Python:

• **pygame**: создание графического интерфейса и обработка событий;

• **json**: работа с сохранениями и данными пользователей;

• **os**: работа с файловой системой;

• **random**: генерация случайных чисел;

• **hashlib**: шифрование паролей;

• **textwrap**: форматирование текста.

**7. Особенности реализации**

• Программа использует объектно-ориентированный подход для моделирования поведения шахматных фигур и доски;

• Графический интерфейс включает в себя шахматную доску, панель управления и область для отображения сообщений;

• Предусмотрена возможность настройки параметров игры через отдельное меню;

• Реализована возможность игры как против другого игрока, так и против компьютера;

• Поддерживается авторизация и регистрация пользователей, а также сохранение и загрузка партий.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

ТЕСТОВАЯ ДОКУМЕНТАЦИЯ

на курсовую работу

по дисциплине «Алгоритмы и структуры данных»

Тема: “Компьютерная игра эндшпиль

«Король, ферзь - Король ладья, 2 пешки»”

**Инв.№ подл.**

**Подп. и дата**

**Взам.инв. №**

**Инв.№ дубл.**

**Подп. и дата**

Р.02069337. 23/685 11 ТД-01

Листов 10

Руководитель разработки:

к.т.н, доцент кафедры "ИВК"

Шишкин Вадим Викторинович

« » 2024 г.

Исполнитель:

студент гр. ИСТбд-21

Казаров Дмитрий Сергеевич

« » 2024 г.

2024

# **Тестовая документация**

на выполнение курсовой работы

по дисциплине «Алгоритмы и структуры данных»

Специальность 09.03.02 – «Информационные системы и технологии»

Тема: «**Разработка компьютерной игра эндшпиль “Король, ферзь-Король ладья, 2 пешки”**»

**1. Описание тестируемых функций**

В процессе тестирования игры необходимо проверить следующие функции:

**1. handle\_click(pos):**

   - Описание: Обрабатывает нажатие на игровое поле, перемещает фигуры и проверяет состояние игры.

   - Тестируемые моменты:

     - Корректное определение позиции нажатия.

     - Корректное перемещение фигур.

     - Проверка на шах, мат и пат.

     - Обновление истории ходов.

     - Корректное отображение возможных ходов.

**2. handle\_events():**

   - Описание: Обрабатывает события, такие как нажатия кнопок и движения мыши.

   - Тестируемые моменты:

     - Корректная обработка нажатий кнопок.

     - Корректная обработка нажатий на игровое поле.

     - Обработка событий выхода из игры.

**3. draw(screen):**

   - Описание: Отрисовывает игровое поле, фигуры, кнопки и сообщения.

   - Тестируемые моменты:

     - Корректное отображение игрового поля.

     - Корректное отображение фигур.

     - Корректное отображение кнопок и их состояний.

     - Корректное отображение сообщений.

**4. save\_game():**

   - Описание: Сохраняет текущее состояние игры в файл.

   - Тестируемые моменты:

     - Корректное сохранение состояния игры.

     - Проверка на наличие ошибок при сохранении.

     - Проверка корректности сохраненного файла.

**5. load\_game():**

   - Описание: Загружает сохраненное состояние игры из файла.

   - Тестируемые моменты:

     - Корректная загрузка состояния игры.

     - Проверка на наличие ошибок при загрузке.

     - Проверка корректности загруженного состояния.

**6. show\_login\_menu():**

   - Описание: Отображает меню авторизации.

   - Тестируемые моменты:

     - Корректное отображение меню авторизации.

     - Проверка ввода логина и пароля.

     - Проверка корректности авторизации.

**7. show\_register\_menu():**

   - Описание: Отображает меню регистрации.

   - Тестируемые моменты:

     - Корректное отображение меню регистрации.

     - Проверка ввода логина и пароля.

     - Проверка корректности регистрации.

**8. promote\_pawn(pawn):**

   - Описание: Обрабатывает превращение пешки в другую фигуру.

   - Тестируемые моменты:

     - Корректное отображение меню выбора фигуры.

     - Проверка корректности выбора фигуры.

     - Проверка корректности обновления состояния игры после превращения.

**9. get\_best\_move(depth, color, alpha, beta, use\_transposition):**

   - Описание: Вычисляет лучший ход для текущего игрока.

   - Тестируемые моменты:

     - Корректное вычисление лучшего хода.

     - Проверка корректности работы алгоритма минимакс.

     - Проверка корректности работы альфа-бета отсечения.

**10. evaluate\_board():**

   - Описание: Оценивает текущее состояние доски.

   - Тестируемые моменты:

     - Корректная оценка состояния доски.

     - Проверка корректности оценки для различных ситуаций (шах, мат, пат).

**2. Mind map**

Для тестирования игры можно организовать карту разума (mind map), чтобы структурировать процесс (Приложение 1)

**3. Чеклист**

|  | **Тестируемая функция** | **Шаги выполнения** | **Ожидаемый результат** | **Статус** |
| --- | --- | --- | --- | --- |
| 1 | handle\_click(pos) | Нажатие на игровое поле | Фигуры перемещаются, состояние игры обновляется | Пройдено/Не пройдено |
| 2 | handle\_events() | Обработка событий | События обрабатываются корректно | Пройдено/Не пройдено |
| 3 | draw(screen) | Отрисовка элементов | Элементы отображаются корректно | Пройдено/Не пройдено |
| 4 | save\_game() | Сохранение игры | Игра сохраняется корректно | Пройдено/Не пройдено |
| 5 | load\_game() | Загрузка игры | Игра загружается корректно | Пройдено/Не пройдено |
| 6 | show\_login\_menu() | Отображение меню авторизации | Меню отображается корректно | Пройдено/Не пройдено |
| 7 | show\_register\_menu() | Отображение меню регистрации | Меню отображается корректно | Пройдено/Не пройдено |
| 8 | promote\_pawn(pawn) | Превращение пешки | Превращение происходит корректно | Пройдено/Не пройдено |
| 9 | get\_best\_move(depth, color, alpha, beta, use\_transposition) | Вычисление лучшего хода | Лучший ход вычисляется корректно | Пройдено/Не пройдено |
| 10 | evaluate\_board() | Оценка состояния доски | Состояние доски оценивается корректно | Пройдено/Не пройдено |

**4. Набор тест-кейсов**

**Тест-кейс 1: Проверка нажатия на игровое поле**

•   Цель: Убедиться, что нажатие на игровое поле обрабатывается корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Нажать на игровое поле.

2.  Проверить перемещение фигур и обновление состояния игры.

•   Ожидаемый результат: Фигуры перемещаются, состояние игры обновляется. Отсутствие сбоев.

**Тест-кейс 2: Проверка обработки событий**

•   Цель: Проверить, что события обрабатываются корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Взаимодействовать с элементами интерфейса (кнопки, игровое поле).

2.  Проверить корректность обработки событий.

•   Ожидаемый результат: События обрабатываются корректно. Отсутствие сбоев.

**Тест-кейс 3: Проверка отрисовки элементов**

•   Цель: Проверить, что элементы отрисовываются корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Наблюдать за отрисовкой игрового поля, фигур, кнопок и сообщений.

•   Ожидаемый результат: Элементы отображаются корректно. Отсутствие сбоев.

**Тест-кейс 4: Проверка сохранения игры**

•   Цель: Проверить, что игра сохраняется корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Нажать кнопку "Сохранить игру".

2.  Проверить корректность сохраненного файла.

•   Ожидаемый результат: Игра сохраняется корректно. Отсутствие сбоев.

**Тест-кейс 5: Проверка загрузки игры**

•   Цель: Проверить, что игра загружается корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Нажать кнопку "Загрузить игру".

2.  Проверить корректность загруженного состояния.

•   Ожидаемый результат: Игра загружается корректно. Отсутствие сбоев.

**Тест-кейс 6: Проверка отображения меню авторизации**

•   Цель: Убедиться, что меню авторизации отображается корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Нажать кнопку "Авторизация".

2.  Проверить корректность отображения меню.

•   Ожидаемый результат: Меню отображается корректно. Отсутствие сбоев.

**Тест-кейс 7: Проверка отображения меню регистрации**

•   Цель: Убедиться, что меню регистрации отображается корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Нажать кнопку "Регистрация".

2.  Проверить корректность отображения меню.

•   Ожидаемый результат: Меню отображается корректно. Отсутствие сбоев.

**Тест-кейс 8: Проверка превращения пешки**

•   Цель: Проверить, что превращение пешки происходит корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Переместить пешку на последнюю горизонталь.

2.  Выбрать фигуру для превращения.

•   Ожидаемый результат: Превращение происходит корректно. Отсутствие сбоев.

**Тест-кейс 9: Проверка вычисления лучшего хода**

•   Цель: Проверить, что лучший ход вычисляется корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Запустить функцию вычисления лучшего хода.

2.  Проверить корректность вычисленного хода.

•   Ожидаемый результат: Лучший ход вычисляется корректно. Отсутствие сбоев.

**Тест-кейс 10: Проверка оценки состояния доски**

•   Цель: Проверить, что состояние доски оценивается корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Запустить функцию оценки состояния доски.

2.  Проверить корректность оценки для различных ситуаций (шах, мат, пат).

•   Ожидаемый результат: Состояние доски оценивается корректно. Отсутствие сбоев.

**Тест-кейс 11: Проверка корректности отображения истории ходов**

•   Цель: Убедиться, что история ходов отображается корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Сделать несколько ходов.

2.  Проверить отображение истории ходов.

•   Ожидаемый результат: История ходов отображается корректно. Отсутствие сбоев.

**Тест-кейс 12: Проверка корректности отображения шаха**

•   Цель: Убедиться, что шах отображается корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Создать ситуацию шаха.

2.  Проверить отображение шаха.

•   Ожидаемый результат: Шах отображается корректно. Отсутствие сбоев.

**Тест-кейс 13: Проверка корректности хода бота при шахе**

•   Цель: Убедиться, что ход бота при шахе работает корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Сделать несколько ходов.

2.  Поставить боту шах

•   Ожидаемый результат: Бот успешно уходит из под шаха. Отсутствие сбоев.

**Тест-кейс 14: Проверка корректности работы функции загрузки прогресса**

•   Цель: Убедиться, что функция загрузки прогресса работает корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Загрузить сохраненный прогресс.

2.  Проверить корректность загруженного прогресса.

•   Ожидаемый результат: Прогресс загружается корректно. Отсутствие сбоев.

**Тест-кейс 15: Проверка корректности отображения мата**

•   Цель: Убедиться, что мат отображается корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Создать ситуацию мата.

2.  Проверить отображение мата.

•   Ожидаемый результат: Мат отображается корректно. Отсутствие сбоев.

**Тест-кейс 16: Проверка корректности отображения пата**

•   Цель: Убедиться, что пат отображается корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Создать ситуацию пата.

2.  Проверить отображение пата.

•   Ожидаемый результат: Пат отображается корректно. Отсутствие сбоев.

**Тест-кейс 17: Проверка корректности работы функции подсказки**

•   Цель: Убедиться, что функция подсказки работает корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Запустить функцию подсказки.

•   Ожидаемый результат: Подсказка работает корректно. Отсутствие сбоев.

**Тест-кейс 18: Проверка корректности работы функции сохранения настроек**

•   Цель: Убедиться, что функция сохранения настроек работает корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Изменить настройки игры.

2.  Сохранить настройки.

3.  Перезапустить игру и проверить сохраненные настройки.

•   Ожидаемый результат: Настройки сохраняются корректно. Отсутствие сбоев.

**Тест-кейс 19: Проверка корректности работы функции загрузки настроек**

•   Цель: Убедиться, что функция загрузки настроек работает корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Загрузить сохраненные настройки.

2.  Проверить корректность загруженных настроек.

•   Ожидаемый результат: Настройки загружаются корректно. Отсутствие сбоев.

**Тест-кейс 20: Проверка корректности работы функции сохранения прогресса**

•   Цель: Убедиться, что функция сохранения прогресса работает корректно.

•   Предусловия: Игра запущена.

•   Шаги:

1.  Сохранить прогресс игры.

2.  Перезапустить игру и проверить сохраненный прогресс.

•   Ожидаемый результат: Прогресс сохраняется корректно. Отсутствие сбоев.

# **Листинг кода**

на выполнение курсовой работы

по дисциплине «Алгоритмы и структуры данных»

Специальность 09.03.02 – «Информационные системы и технологии»

Тема: «**Разработка компьютерной игра эндшпиль “Король, ферзь-Король ладья, 2 пешки”**»

import pygame  
import sys  
import json  
import os  
import random  
import hashlib  
import textwrap  
  
pygame.init()  
info = pygame.display.Info()  
screen\_width = info.current\_w  
screen\_height = info.current\_h  
tile\_size = min(screen\_width, screen\_height) // 9  
screen = pygame.display.set\_mode((screen\_width, screen\_height), pygame.FULLSCREEN)  
pygame.display.set\_caption("Шахматный эндшпиль")  
clock = pygame.time.Clock()  
font = pygame.font.Font(None, 36)  
  
background = pygame.image.load("assets/background.jpg")  
background = pygame.transform.scale(background, (screen\_width, screen\_height))  
  
move\_sound = pygame.mixer.Sound("assets/soundChess.mp3")  
move\_sound.set\_volume(0.5)  
  
def load\_images():  
 pieces = {}  
 for color in ["black", "white"]:  
 for piece in ["king", "queen", "rook", "pawn", "bishop", "knight"]:  
 image = pygame.image.load(f"assets/{color}\_{piece}.png").convert\_alpha()  
 pieces[f"{color}\_{piece}"] = pygame.transform.scale(image, (tile\_size, tile\_size))  
 pieces[f"{color}\_{piece}\_small"] = pygame.transform.scale(image, (tile\_size // 2, tile\_size // 2))  
 return pieces  
  
board\_color\_1 = (235, 235, 208)  
board\_color\_2 = (92, 86, 83)  
highlight\_color = (186, 202, 68)  
button\_color = (70, 130, 180)  
button\_hover\_color = (100, 149, 237)  
selected\_piece = None  
current\_turn = "white"  
  
pieces = load\_images()  
is\_authenticated = False  
play\_against\_pc = False  
player\_color = "white"  
current\_user = None  
  
class Button:  
 def \_\_init\_\_(self, x, y, width, height, text, color=None):  
 self.rect = pygame.Rect(x, y, width, height)  
 self.text = text  
 self.is\_hovered = False  
 self.custom\_color = color  
 self.disabled = False  
  
 def draw(self, screen):  
 if self.custom\_color:  
 color = self.custom\_color  
 else:  
 color = button\_hover\_color if self.is\_hovered else button\_color  
 if self.disabled:  
 color = (169, 169, 169)  
 pygame.draw.rect(screen, color, self.rect, border\_radius=10)  
 pygame.draw.rect(screen, (50, 50, 50), self.rect, 2, border\_radius=10)  
  
 text\_shadow = font.render(self.text, True, (0, 0, 0))  
 text\_surface = font.render(self.text, True, (255, 255, 255))  
 text\_rect = text\_surface.get\_rect(center=self.rect.center)  
 screen.blit(text\_shadow, (text\_rect.x + 2, text\_rect.y + 2))  
 screen.blit(text\_surface, text\_rect)  
  
 def handle\_event(self, event):  
 if self.disabled:  
 return False  
 if event.type == pygame.MOUSEMOTION:  
 self.is\_hovered = self.rect.collidepoint(event.pos)  
 elif event.type == pygame.MOUSEBUTTONDOWN:  
 if self.is\_hovered:  
 return True  
 return False  
  
class Piece:  
 def \_\_init\_\_(self, color, type, position):  
 self.color = color  
 self.type = type  
 self.position = position  
 self.image = pieces[f"{color}\_{type}"]  
 self.has\_moved = False  
  
 def draw(self, screen):  
 board\_offset\_x = (screen\_width - 8 \* tile\_size) // 2  
 board\_offset\_y = (screen\_height - 8 \* tile\_size) // 2  
 x, y = self.position  
 screen.blit(self.image, (board\_offset\_x + x \* tile\_size, board\_offset\_y + y \* tile\_size))  
  
 def move\_to(self, new\_position):  
 self.position = new\_position  
 self.has\_moved = True  
  
class ChessBoard:  
 def \_\_init\_\_(self):  
 self.pieces = {}  
 self.board\_offset\_x = (screen\_width - 8 \* tile\_size) // 2  
 self.board\_offset\_y = (screen\_height - 8 \* tile\_size) // 2  
 self.message = ""  
 self.message\_timer = 0  
 self.moves\_history = []  
 self.current\_save = None  
 self.valid\_moves = [] *# список для хранения возможных ходов* self.captured\_pieces = [] *# список для хранения съеденных фигур* self.game\_over = False *# переменная для отслеживания конца игры* self.piece\_values = {  
 'king': 10000,  
 'queen': 90,  
 'rook': 50,  
 'bishop': 35,  
 'knight': 32,  
 'pawn': 10  
 }  
 self.transposition\_table = {}  
  
 button\_width = 200  
 button\_height = 50  
 container\_width = 300  
 button\_x = self.board\_offset\_x - container\_width - 50 + (container\_width - button\_width) // 2  
 button\_spacing = 20  
  
 container\_height = 700  
 total\_buttons\_height = (button\_height + button\_spacing) \* 6 - button\_spacing  
 start\_y = (container\_height - total\_buttons\_height) // 2 + 200  
  
 self.buttons = {  
 'login': Button(button\_x, start\_y, button\_width, button\_height, "Авторизация"),  
 'new\_game': Button(button\_x, start\_y + button\_height + button\_spacing, button\_width, button\_height,  
 "Новая игра"),  
 'save\_game': Button(button\_x, start\_y + (button\_height + button\_spacing) \* 2, button\_width, button\_height,  
 "Сохранить игру"),  
 'load\_game': Button(button\_x, start\_y + (button\_height + button\_spacing) \* 3, button\_width, button\_height,  
 "Загрузить игру"),  
 'instructions': Button(button\_x, start\_y + (button\_height + button\_spacing) \* 4, button\_width,  
 button\_height,  
 "Инструкция"),  
 'exit': Button(button\_x, start\_y + (button\_height + button\_spacing) \* 5, button\_width, button\_height,  
 "Выход из игры")  
 }  
 def setup\_pieces(self, piece\_set, color):  
 pieces = {}  
 positions = [(x, y) for x in range(8) for y in range(8)]  
 random.shuffle(positions)  
  
 def get\_random\_position():  
 while positions:  
 pos = positions.pop()  
 if pos[1] not in [0, 7]: *# условие для исключения крайних верхних и нижних клеток для пешек* return pos  
 return None  
  
 def create\_pieces():  
 if piece\_set == "king\_queen":  
 if color == "white":  
 return {  
 "white\_king": Piece("white", "king", positions.pop()),  
 "white\_queen": Piece("white", "queen", positions.pop()),  
 "black\_king": Piece("black", "king", positions.pop()),  
 "black\_rook": Piece("black", "rook", positions.pop()),  
 "black\_pawn\_1": Piece("black", "pawn", get\_random\_position()),  
 "black\_pawn\_2": Piece("black", "pawn", get\_random\_position())  
 }  
 else:  
 return {  
 "black\_king": Piece("black", "king", positions.pop()),  
 "black\_queen": Piece("black", "queen", positions.pop()),  
 "white\_king": Piece("white", "king", positions.pop()),  
 "white\_rook": Piece("white", "rook", positions.pop()),  
 "white\_pawn\_1": Piece("white", "pawn", get\_random\_position()),  
 "white\_pawn\_2": Piece("white", "pawn", get\_random\_position())  
 }  
 elif piece\_set == "king\_rook\_pawns":  
 if color == "white":  
 return {  
 "white\_king": Piece("white", "king", positions.pop()),  
 "white\_rook": Piece("white", "rook", positions.pop()),  
 "white\_pawn\_1": Piece("white", "pawn", get\_random\_position()),  
 "white\_pawn\_2": Piece("white", "pawn", get\_random\_position()),  
 "black\_king": Piece("black", "king", positions.pop()),  
 "black\_queen": Piece("black", "queen", positions.pop())  
 }  
 else:  
 return {  
 "black\_king": Piece("black", "king", positions.pop()),  
 "black\_rook": Piece("black", "rook", positions.pop()),  
 "black\_pawn\_1": Piece("black", "pawn", get\_random\_position()),  
 "black\_pawn\_2": Piece("black", "pawn", get\_random\_position()),  
 "white\_king": Piece("white", "king", positions.pop()),  
 "white\_queen": Piece("white", "queen", positions.pop())  
 }  
  
 while True:  
 positions = [(x, y) for x in range(8) for y in range(8)]  
 random.shuffle(positions)  
 pieces = create\_pieces()  
 self.pieces = pieces  
  
 if not self.is\_check("white") and not self.is\_check("black"):  
 break  
  
 return pieces  
  
 def add\_move\_to\_history(self, piece, start\_pos, end\_pos, captured=None):  
 piece\_symbol = {  
 'king': 'K', 'queen': 'Q', 'rook': 'R', 'pawn': '', 'bishop': 'B', 'knight': 'N'  
 }  
 start\_coord = f"{chr(97 + start\_pos[0])}{8 - start\_pos[1]}"  
 end\_coord = f"{chr(97 + end\_pos[0])}{8 - end\_pos[1]}"  
 move\_text = f"{piece\_symbol[piece.type]}{start\_coord}-{end\_coord}"  
 if captured:  
 move\_text += "x"  
 self.moves\_history.append(move\_text)  
  
 def is\_piece\_between(self, start\_pos, end\_pos):  
 x1, y1 = start\_pos  
 x2, y2 = end\_pos  
  
 dx = 0 if x2 == x1 else (x2 - x1) // abs(x2 - x1)  
 dy = 0 if y2 == y1 else (y2 - y1) // abs(y2 - y1)  
  
 current\_x, current\_y = x1 + dx, y1 + dy  
 while (current\_x, current\_y) != (x2, y2):  
 if self.get\_piece\_at((current\_x, current\_y)):  
 return True  
 current\_x += dx  
 current\_y += dy  
  
 return False  
  
 def is\_check(self, color):  
 king\_pos = None  
 for piece in self.pieces.values():  
 if piece.type == "king" and piece.color == color:  
 king\_pos = piece.position  
 break  
 if king\_pos is None:  
 return False  
 for piece in self.pieces.values():  
 if piece.color != color:  
 if piece.type == "knight":  
 x\_diff = abs(king\_pos[0] - piece.position[0])  
 y\_diff = abs(king\_pos[1] - piece.position[1])  
 if (x\_diff == 2 and y\_diff == 1) or (x\_diff == 1 and y\_diff == 2):  
 return True  
 elif self.is\_valid\_move(piece, king\_pos):  
 if not self.is\_piece\_between(piece.position, king\_pos):  
 return True  
 return False  
  
 def is\_checkmate(self, color):  
 if not self.is\_check(color):  
 return False  
  
 pieces\_list = list(self.pieces.values())  
 for piece in pieces\_list:  
 if piece.color == color:  
 valid\_moves = self.get\_valid\_moves(piece)  
 for move in valid\_moves:  
 original\_pos = piece.position  
 captured\_piece = self.get\_piece\_at(move)  
 piece.move\_to(move)  
 if captured\_piece:  
 captured\_key = None  
 for k, v in self.pieces.items():  
 if v == captured\_piece:  
 captured\_key = k  
 break  
 if captured\_key:  
 del self.pieces[captured\_key]  
  
 still\_in\_check = self.is\_check(color)  
  
 piece.move\_to(original\_pos)  
 if captured\_piece and captured\_key:  
 self.pieces[captured\_key] = captured\_piece  
  
 if not still\_in\_check:  
 return False  
 return True  
  
 def is\_stalemate(self, color):  
 if self.is\_check(color):  
 return False  
  
 pieces\_list = list(self.pieces.values())  
 for piece in pieces\_list:  
 if piece.color == color:  
 valid\_moves = self.get\_valid\_moves(piece)  
 for move in valid\_moves:  
 original\_pos = piece.position  
 captured\_piece = self.get\_piece\_at(move)  
 piece.move\_to(move)  
 if captured\_piece:  
 captured\_key = None  
 for k, v in self.pieces.items():  
 if v == captured\_piece:  
 captured\_key = k  
 break  
 if captured\_key:  
 del self.pieces[captured\_key]  
  
 legal\_move = not self.is\_check(color)  
  
 piece.move\_to(original\_pos)  
 if captured\_piece and captured\_key:  
 self.pieces[captured\_key] = captured\_piece  
  
 if legal\_move:  
 return False  
 return True  
  
 def draw(self, screen):  
 screen.blit(background, (0, 0))  
  
 for row in range(8):  
 for col in range(8):  
 color = board\_color\_1 if (row + col) % 2 == 0 else board\_color\_2  
 pygame.draw.rect(screen, color,  
 pygame.Rect(self.board\_offset\_x + col \* tile\_size,  
 self.board\_offset\_y + row \* tile\_size,  
 tile\_size, tile\_size))  
  
 for piece in self.pieces.values():  
 piece.draw(screen)  
  
 if selected\_piece:  
 valid\_moves = self.get\_valid\_moves(selected\_piece)  
 for move in valid\_moves:  
 target\_piece = self.get\_piece\_at(move)  
 if target\_piece and target\_piece.color != selected\_piece.color:  
 pygame.draw.circle(screen, (255, 165, 0),  
 (self.board\_offset\_x + move[0] \* tile\_size + tile\_size // 2,  
 self.board\_offset\_y + move[1] \* tile\_size + tile\_size // 2),  
 10)  
 else:  
 pygame.draw.circle(screen, highlight\_color,  
 (self.board\_offset\_x + move[0] \* tile\_size + tile\_size // 2,  
 self.board\_offset\_y + move[1] \* tile\_size + tile\_size // 2),  
 10)  
  
 for i in range(8):  
 pygame.draw.rect(screen, (26, 7, 4),  
 (self.board\_offset\_x - 30, self.board\_offset\_y + i \* tile\_size, 30, tile\_size))  
 text = font.render(str(8 - i), True, (255, 255, 255))  
 screen.blit(text, (self.board\_offset\_x - 25, self.board\_offset\_y + i \* tile\_size + tile\_size // 3))  
  
 pygame.draw.rect(screen, (26, 7, 4),  
 (self.board\_offset\_x - 30 + i \* tile\_size, self.board\_offset\_y + 8 \* tile\_size,  
 tile\_size + 30, 30))  
 text = font.render(chr(97 + i), True, (255, 255, 255))  
 screen.blit(text, (  
 self.board\_offset\_x + i \* tile\_size + tile\_size // 3, self.board\_offset\_y + 8 \* tile\_size + 5))  
  
 pygame.draw.rect(screen, (26, 7, 4),  
 (self.board\_offset\_x - 30, self.board\_offset\_y - 30, 8 \* tile\_size + 60, 30))  
  
 pygame.draw.rect(screen, (26, 7, 4),  
 (self.board\_offset\_x + 8 \* tile\_size, self.board\_offset\_y - 30, 30, 8 \* tile\_size + 60))  
  
 if self.message and self.message\_timer > 0:  
 message\_surface = pygame.Surface((screen\_width, 100), pygame.SRCALPHA)  
 message\_surface.fill((0, 0, 0, 128))  
  
 text\_shadow = font.render(self.message, True, (0, 0, 0))  
 text\_surface = font.render(self.message, True, (255, 255, 255))  
 text\_rect = text\_surface.get\_rect(center=(screen\_width // 2, 50))  
 message\_surface.blit(text\_shadow, (text\_rect.x + 4, text\_rect.y + 4))  
 message\_surface.blit(text\_surface, text\_rect)  
  
 screen.blit(message\_surface, (0, 0))  
 self.message\_timer -= 1  
  
 container\_width = 300  
 container\_height = screen\_height - 400  
 buttons\_x = self.board\_offset\_x - container\_width - 50  
 buttons\_y = 170  
 moves\_x = self.board\_offset\_x + 8 \* tile\_size + 50  
 moves\_y = 170  
  
 pygame.draw.rect(screen, (235, 235, 208),  
 (buttons\_x, buttons\_y, container\_width, container\_height))  
 pygame.draw.rect(screen, (26, 7, 4),  
 (buttons\_x, buttons\_y, container\_width, container\_height), 6)  
  
 title\_font = pygame.font.Font(None, 52)  
 title\_text\_1 = title\_font.render("Шахматный", True, (0, 0, 0))  
 title\_text\_2 = title\_font.render("эндшпиль", True, (0, 0, 0))  
  
 title\_rect\_1 = title\_text\_1.get\_rect(center=(buttons\_x + container\_width // 2, buttons\_y + 70))  
 title\_rect\_2 = title\_text\_2.get\_rect(center=(buttons\_x + container\_width // 2, buttons\_y + 115))  
  
 shadow\_offset = 2  
 shadow\_color = (150, 150, 150)  
  
 shadow\_rect\_1 = title\_rect\_1.copy()  
 shadow\_rect\_1.center = (title\_rect\_1.centerx + shadow\_offset, title\_rect\_1.centery + shadow\_offset)  
 screen.blit(title\_font.render("Шахматный", True, shadow\_color), shadow\_rect\_1)  
  
 shadow\_rect\_2 = title\_rect\_2.copy()  
 shadow\_rect\_2.center = (title\_rect\_2.centerx + shadow\_offset, title\_rect\_2.centery + shadow\_offset)  
 screen.blit(title\_font.render("эндшпиль", True, shadow\_color), shadow\_rect\_2)  
  
 screen.blit(title\_text\_1, title\_rect\_1)  
 screen.blit(title\_text\_2, title\_rect\_2)  
  
 for button in self.buttons.values():  
 button.draw(screen)  
  
 pygame.draw.rect(screen, (235, 235, 208),  
 (moves\_x, moves\_y, container\_width, container\_height))  
 pygame.draw.rect(screen, (26, 7, 4),  
 (moves\_x, moves\_y, container\_width, container\_height), 6)  
  
 if self.current\_save:  
 save\_text = font.render("Текущее сохранение:", True, (0, 0, 0))  
 screen.blit(save\_text, (moves\_x + 10, moves\_y + 10))  
 save\_name = font.render(f"{self.current\_save}", True, (0, 0, 0))  
 screen.blit(save\_name, (moves\_x + 10, moves\_y + 40))  
 turn\_text = font.render(  
 f"Ход {'белых' if current\_turn == 'white' else 'черных'}",  
 True, (0, 0, 0))  
 screen.blit(turn\_text, (moves\_x + 10, moves\_y + 80))  
  
 elif not self.game\_over:  
 if not self.pieces:  
 turn\_text = font.render("Добро пожаловать!", True, (0, 0, 0))  
 else:  
 turn\_text = font.render(  
 f"Ход {'белых' if current\_turn == 'white' else 'черных'}",  
 True, (0, 0, 0))  
 screen.blit(turn\_text, (moves\_x + 10, moves\_y + 80))  
 else:  
 turn\_text = font.render("Конец игры!", True, (0, 0, 0))  
 screen.blit(turn\_text, (moves\_x + 10, moves\_y + 80))  
  
 history\_text = font.render("История ходов:", True, (0, 0, 0))  
 screen.blit(history\_text, (moves\_x + 10, moves\_y + 120))  
  
 visible\_moves = 11  
 start\_index = max(0, len(self.moves\_history) - visible\_moves)  
  
 for i, move in enumerate(self.moves\_history[start\_index:]):  
 move\_number = start\_index + i + 1  
 move\_text = font.render(f"{move\_number}. {move}", True, (0, 0, 0))  
 screen.blit(move\_text, (moves\_x + 10, moves\_y + 160 + i \* 30))  
  
 small\_piece\_size = tile\_size // 2  
 captured\_x = moves\_x + 10  
 captured\_y = moves\_y + container\_height - small\_piece\_size \* 2 - 20  
  
 captured\_text = font.render("Съеденные фигуры:", True, (0, 0, 0))  
 screen.blit(captured\_text, (captured\_x, captured\_y - 30))  
  
 white\_captured = [p for p in self.captured\_pieces if p.color == "white"]  
 black\_captured = [p for p in self.captured\_pieces if p.color == "black"]  
  
 for i, piece in enumerate(black\_captured):  
 if i < 8:  
 x = captured\_x + i \* small\_piece\_size  
 y = captured\_y  
 small\_image = pieces[f"{piece.color}\_{piece.type}\_small"]  
 screen.blit(small\_image, (x, y))  
  
 for i, piece in enumerate(white\_captured):  
 if i < 8:  
 x = captured\_x + i \* small\_piece\_size  
 y = captured\_y + small\_piece\_size  
 small\_image = pieces[f"{piece.color}\_{piece.type}\_small"]  
 screen.blit(small\_image, (x, y))  
  
 def promote\_pawn(self, pawn):  
 *# Создание поверхности для меню выбора фигуры* menu\_surface = pygame.Surface(screen.get\_size(), pygame.SRCALPHA)  
 for i in range(screen\_height):  
 alpha = min(128, i // 16)  
 pygame.draw.line(menu\_surface, (70, 130, 180, alpha), (0, i), (screen\_width, i))  
  
 menu\_width = 800  
 menu\_height = 300  
 menu\_rect = pygame.Rect((screen\_width - menu\_width) // 2,  
 (screen\_height - menu\_height) // 2,  
 menu\_width, menu\_height)  
  
 piece\_types = ["queen", "rook", "bishop", "knight"]  
 piece\_rects = []  
 piece\_size = tile\_size  
 spacing = 20  
 total\_width = len(piece\_types) \* piece\_size + (len(piece\_types) - 1) \* spacing  
 start\_x = menu\_rect.centerx - total\_width // 2 + 70  
  
 for i, piece\_type in enumerate(piece\_types):  
 piece\_image = pieces[f"{pawn.color}\_{piece\_type}"]  
 piece\_rect = piece\_image.get\_rect()  
 piece\_rect.centerx = start\_x + i \* (piece\_size + spacing)  
 piece\_rect.centery = menu\_rect.centery  
 piece\_rects.append((piece\_rect, piece\_type))  
  
 selecting = True  
 while selecting:  
 screen.blit(background, (0, 0))  
 screen.blit(menu\_surface, (0, 0))  
  
 pygame.draw.rect(screen, (235, 235, 208), menu\_rect)  
 pygame.draw.rect(screen, (26, 7, 4), menu\_rect, 6)  
  
 title = font.render("Выберите фигуру для превращения:", True, (0, 0, 0))  
 title\_rect = title.get\_rect(center=(screen\_width // 2, menu\_rect.top + 30))  
 screen.blit(title, title\_rect)  
  
 for rect, piece\_type in piece\_rects:  
 screen.blit(pieces[f"{pawn.color}\_{piece\_type}"], rect)  
  
 pygame.display.flip()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 elif event.type == pygame.MOUSEBUTTONDOWN:  
 for rect, piece\_type in piece\_rects:  
 if rect.collidepoint(event.pos):  
 new\_piece = Piece(pawn.color, piece\_type, pawn.position)  
 for key, value in self.pieces.items():  
 if value == pawn:  
 self.pieces[key] = new\_piece  
 break  
 selecting = False  
 return  
  
 def get\_next\_save\_number(self):  
 user\_save\_dir = os.path.join('saves', current\_user)  
 existing\_saves = [f for f in os.listdir(user\_save\_dir) if f.startswith('save') and f.endswith('.json')]  
 if not existing\_saves:  
 return 1  
 numbers = [int(f.replace('save', '').replace('.json', '')) for f in existing\_saves]  
 max\_number = max(numbers)  
 next\_number = max\_number + 1 if max\_number < 9 else 1  
 return next\_number  
  
 def delete\_all\_saves(self):  
 user\_save\_dir = os.path.join('saves', current\_user)  
 saves = [f for f in os.listdir(user\_save\_dir) if f.startswith('save') and f.endswith('.json')]  
 for save in saves:  
 os.remove(os.path.join(user\_save\_dir, save))  
 self.current\_save = None  
 self.show\_message("Все сохранения удалены!")  
  
 def save\_game(self):  
 if not self.pieces or self.game\_over:  
 self.show\_message("Нечего сохранять!")  
 return  
  
 user\_save\_dir = os.path.join('saves', current\_user)  
 if not os.path.exists(user\_save\_dir):  
 os.makedirs(user\_save\_dir)  
  
 save\_number = self.get\_next\_save\_number()  
 filename = os.path.join(user\_save\_dir, f'save{save\_number}.json')  
 game\_state = {  
 'pieces': [(p.color, p.type, p.position) for p in self.pieces.values()],  
 'moves\_history': self.moves\_history,  
 'current\_turn': current\_turn,  
 'captured\_pieces': [(p.color, p.type) for p in self.captured\_pieces],  
 'play\_against\_pc': play\_against\_pc,  
 'player\_color': player\_color  
 }  
 with open(filename, 'w') as f:  
 json.dump(game\_state, f)  
 self.current\_save = filename  
 if save\_number < 9:  
 self.show\_message(f"Игра сохранена как {filename}!")  
 else:  
 self.show\_message(f"Сохранено в {filename} Дальнейшие сохранения будут в save1.json. Очистите память")  
  
 def load\_game(self):  
 user\_save\_dir = os.path.join('saves', current\_user)  
 saves = [f for f in os.listdir(user\_save\_dir) if f.startswith('save') and f.endswith('.json')]  
 if not saves:  
 self.show\_message("Нет доступных сохранений!")  
 return  
  
 menu\_surface = pygame.Surface(screen.get\_size(), pygame.SRCALPHA)  
 for i in range(screen\_height):  
 alpha = min(128, i // 16)  
 pygame.draw.line(menu\_surface, (70, 130, 180, alpha), (0, i), (screen\_width, i))  
  
 menu\_width = 400  
 menu\_height = len(saves) \* 60 + 220  
 menu\_rect = pygame.Rect((screen\_width - menu\_width) // 2,  
 (screen\_height - menu\_height) // 2,  
 menu\_width, menu\_height)  
  
 pygame.draw.rect(menu\_surface, (235, 235, 208), menu\_rect)  
 pygame.draw.rect(menu\_surface, (26, 7, 4), menu\_rect, 6)  
  
 title = font.render("Выберите нужное сохранение:", True, (0, 0, 0))  
 title\_rect = title.get\_rect(center=(screen\_width // 2, menu\_rect.top + 30))  
  
 save\_buttons = []  
 button\_height = 50  
 button\_spacing = 10  
 start\_y = menu\_rect.top + 70  
  
 for i, save in enumerate(saves):  
 button = Button(menu\_rect.centerx - 100, start\_y + i \* (button\_height + button\_spacing),  
 200, button\_height, save)  
 save\_buttons.append((button, save))  
  
 back\_button = Button(menu\_rect.centerx - 100,  
 start\_y + len(saves) \* (button\_height + button\_spacing),  
 200, button\_height, "Назад")  
  
 delete\_button = Button(menu\_rect.centerx - 150,  
 start\_y + len(saves) \* (button\_height + button\_spacing) + button\_height + button\_spacing,  
 300, button\_height, "Удалить все сохранения", (255, 0, 0))  
  
 selecting = True  
 while selecting:  
 screen.blit(background, (0, 0))  
 screen.blit(menu\_surface, (0, 0))  
 screen.blit(title, title\_rect)  
  
 for button, \_ in save\_buttons:  
 button.draw(screen)  
  
 back\_button.draw(screen)  
 delete\_button.draw(screen)  
  
 pygame.display.flip()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 elif event.type == pygame.MOUSEBUTTONDOWN:  
 if delete\_button.rect.collidepoint(event.pos):  
 self.delete\_all\_saves()  
 selecting = False  
 break  
 elif back\_button.rect.collidepoint(event.pos):  
 selecting = False  
 break  
  
 for button, save\_file in save\_buttons:  
 if button.rect.collidepoint(event.pos):  
 try:  
 with open(os.path.join(user\_save\_dir, save\_file), 'r') as f:  
 game\_state = json.load(f)  
 self.pieces = {}  
 for color, type, position in game\_state['pieces']:  
 piece\_key = f"{color}\_{type}"  
 if len([p for p in self.pieces.values() if  
 p.type == type and p.color == color]) > 0:  
 piece\_key += f"\_{len([p for p in self.pieces.values() if p.type == type and p.color == color]) + 1}"  
 self.pieces[piece\_key] = Piece(color, type, tuple(position))  
 self.moves\_history = game\_state['moves\_history']  
 global current\_turn, play\_against\_pc, player\_color  
 current\_turn = game\_state.get('current\_turn', 'white')  
 play\_against\_pc = game\_state.get('play\_against\_pc',  
 False) *# Загружаем режим игры против ПК* player\_color = game\_state.get('player\_color', 'white') *# Загружаем цвет игрока* self.captured\_pieces = []  
 if 'captured\_pieces' in game\_state:  
 for color, type in game\_state['captured\_pieces']:  
 self.captured\_pieces.append(Piece(color, type, (-1, -1)))  
 self.current\_save = save\_file  
 self.show\_message(f"Игра загружена из {save\_file}!")  
 selecting = False  
 except:  
 self.show\_message("Ошибка загрузки игры!")  
 elif event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_ESCAPE:  
 selecting = False  
  
 def show\_message(self, text, duration=180):  
 self.message = text  
 self.message\_timer = duration  
  
 def get\_piece\_at(self, position):  
 for piece in self.pieces.values():  
 if piece.position == position:  
 return piece  
 return None  
  
 def is\_valid\_move(self, piece, new\_pos):  
 if not (0 <= new\_pos[0] < 8 and 0 <= new\_pos[1] < 8):  
 return False  
  
 target\_piece = self.get\_piece\_at(new\_pos)  
 if target\_piece and target\_piece.color == piece.color:  
 return False  
  
 x\_diff = abs(new\_pos[0] - piece.position[0])  
 y\_diff = abs(new\_pos[1] - piece.position[1])  
  
 if piece.type == "king":  
 return x\_diff <= 1 and y\_diff <= 1  
 elif piece.type == "queen":  
 return (x\_diff == y\_diff or x\_diff == 0 or y\_diff == 0) and not self.is\_piece\_between(piece.position,  
 new\_pos)  
 elif piece.type == "rook":  
 return (x\_diff == 0 or y\_diff == 0) and not self.is\_piece\_between(piece.position, new\_pos)  
 elif piece.type == "bishop":  
 return x\_diff == y\_diff and not self.is\_piece\_between(piece.position, new\_pos)  
 elif piece.type == "knight":  
 return (x\_diff == 2 and y\_diff == 1) or (x\_diff == 1 and y\_diff == 2)  
 elif piece.type == "pawn":  
 direction = -1 if piece.color == "white" else 1  
 if target\_piece:  
 return y\_diff == 1 and x\_diff == 1 and (new\_pos[1] - piece.position[1]) == direction  
 else:  
 if not piece.has\_moved:  
 return x\_diff == 0 and (new\_pos[1] - piece.position[1]) == direction \* (  
 1 if y\_diff == 1 else 2) and not self.is\_piece\_between(piece.position, new\_pos)  
 return x\_diff == 0 and (new\_pos[1] - piece.position[1]) == direction and y\_diff == 1  
  
 return False  
  
 def get\_valid\_moves(self, piece):  
 valid\_moves = []  
 for x in range(8):  
 for y in range(8):  
 if self.is\_valid\_move(piece, (x, y)):  
 original\_pos = piece.position  
 captured\_piece = self.get\_piece\_at((x, y))  
 piece.move\_to((x, y))  
  
 if captured\_piece:  
 captured\_key = None  
 for k, v in self.pieces.items():  
 if v == captured\_piece:  
 captured\_key = k  
 break  
 if captured\_key:  
 del self.pieces[captured\_key]  
  
 if not self.is\_check(piece.color):  
 valid\_moves.append((x, y))  
  
 piece.move\_to(original\_pos)  
 if captured\_piece and captured\_key:  
 self.pieces[captured\_key] = captured\_piece  
  
 return valid\_moves  
  
 def handle\_click(self, pos):  
 global selected\_piece, current\_turn  
 board\_x = (pos[0] - self.board\_offset\_x) // tile\_size  
 board\_y = (pos[1] - self.board\_offset\_y) // tile\_size  
  
 if not (0 <= board\_x < 8 and 0 <= board\_y < 8):  
 return  
  
 clicked\_pos = (board\_x, board\_y)  
 clicked\_piece = self.get\_piece\_at(clicked\_pos)  
  
 if selected\_piece:  
 if clicked\_piece and clicked\_piece.color == selected\_piece.color:  
 selected\_piece = clicked\_piece  
 self.valid\_moves = self.get\_valid\_moves(selected\_piece)  
 elif (board\_x, board\_y) in self.valid\_moves:  
 start\_pos = selected\_piece.position  
 captured = clicked\_piece is not None  
 selected\_piece.move\_to(clicked\_pos)  
  
 if clicked\_piece:  
 captured\_key = None  
 for k, v in self.pieces.items():  
 if v == clicked\_piece:  
 captured\_key = k  
 break  
 if captured\_key:  
 del self.pieces[captured\_key]  
 self.captured\_pieces.append(clicked\_piece)  
  
 *# Автоматическое превращение пешки бота в ферзя* if selected\_piece.type == "pawn":  
 if (selected\_piece.color == "white" and clicked\_pos[1] == 0) or \  
 (selected\_piece.color == "black" and clicked\_pos[1] == 7):  
 if play\_against\_pc and current\_turn != player\_color:  
 selected\_piece.type = "queen"  
 selected\_piece.image = pieces[f"{selected\_piece.color}\_queen"]  
 else:  
 self.promote\_pawn(selected\_piece)  
  
 self.add\_move\_to\_history(selected\_piece, start\_pos, clicked\_pos, captured)  
 next\_color = "black" if current\_turn == "white" else "white"  
 if self.is\_check(next\_color):  
 if self.is\_checkmate(next\_color):  
 self.show\_message(f"Шах и мат! Победа {current\_turn}!")  
 self.game\_over = True  
 else:  
 self.show\_message("Шах!")  
 elif self.is\_stalemate(next\_color):  
 self.show\_message("Пат! Ничья!")  
 self.game\_over = True  
  
 current\_turn = next\_color  
 selected\_piece = None  
 self.valid\_moves = []  
 move\_sound.play()  
  
 pieces\_backup = self.pieces.copy()  
 captured\_backup = self.captured\_pieces.copy()  
  
 if play\_against\_pc and current\_turn != player\_color and not self.game\_over:  
 best\_move = self.get\_best\_move(2, current\_turn)  
 if best\_move and isinstance(best\_move, tuple) and len(best\_move) == 2:  
 piece, move = best\_move  
 if isinstance(piece, Piece):  
 self.handle\_click((self.board\_offset\_x + piece.position[0] \* tile\_size + tile\_size // 2,  
 self.board\_offset\_y + piece.position[1] \* tile\_size + tile\_size // 2))  
 else:  
 self.show\_message("Некорректный ход!")  
 selected\_piece = None  
 self.valid\_moves = []  
 elif clicked\_piece and clicked\_piece.color == current\_turn:  
 selected\_piece = clicked\_piece  
 self.valid\_moves = self.get\_valid\_moves(selected\_piece)  
  
 if len(self.pieces) == 2 and all(piece.type == "king" for piece in self.pieces.values()):  
 self.show\_message("Пат! Ничья!")  
 self.game\_over = True  
  
 def handle\_events(self):  
 global selected\_piece, current\_turn, is\_authenticated, play\_against\_pc, player\_color  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 elif event.type == pygame.MOUSEBUTTONDOWN:  
 mouse\_pos = pygame.mouse.get\_pos()  
  
 for button\_name, button in self.buttons.items():  
 if button.handle\_event(event):  
 if button\_name == 'login':  
 self.show\_login\_menu()  
 selected\_piece = None  
 elif button\_name == 'new\_game' and is\_authenticated:  
 self.show\_new\_game\_menu()  
 selected\_piece = None  
 elif button\_name == 'new\_game' and not is\_authenticated:  
 self.show\_message("Для начала новой игры, пожалуйста, авторизуйтесь!")  
 elif button\_name == 'save\_game' and is\_authenticated:  
 self.save\_game()  
 elif button\_name == 'save\_game' and not is\_authenticated:  
 self.show\_message("Нечего сохранять!")  
 elif button\_name == 'load\_game' and is\_authenticated:  
 self.load\_game()  
 selected\_piece = None  
 elif button\_name == 'load\_game' and not is\_authenticated:  
 self.show\_message("Для загрузки игры, пожалуйста, авторизуйтесь!")  
 elif button\_name == 'instructions':  
 self.show\_instructions()  
 elif button\_name == 'exit':  
 pygame.quit()  
 sys.exit()  
 return  
  
 self.handle\_click(mouse\_pos)  
  
 elif event.type == pygame.MOUSEMOTION:  
 for button in self.buttons.values():  
 button.handle\_event(event)  
  
 elif event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_ESCAPE:  
 pygame.quit()  
 sys.exit()  
  
 def clear\_board(self):  
 self.pieces.clear()  
 self.captured\_pieces.clear()  
  
 def show\_instructions(self):  
 instructions = [  
 ("", "assets/first\_page.jpg"),  
 ("", "assets/second\_page.jpg"),  
 ("", "assets/third\_page.jpg"),  
 ("", "assets/fourth\_page.jpg"),  
 ("", "assets/fifth\_page.jpg"),  
 ("", "assets/sixth\_page.jpg"),  
 ("", "assets/seventh\_page.jpg"),  
 ]  
  
 instruction\_surface = pygame.Surface(screen.get\_size(), pygame.SRCALPHA)  
 for i in range(screen\_height):  
 alpha = min(128, i // 16)  
 pygame.draw.line(instruction\_surface, (70, 130, 180, alpha), (0, i), (screen\_width, i))  
  
 container\_width = 900  
 container\_height = screen\_height - 100  
 container\_rect = pygame.Rect((screen\_width - container\_width) // 2,  
 50,  
 container\_width, container\_height)  
  
 current\_index = 0  
 showing\_instructions = True  
 while showing\_instructions:  
 screen.blit(background, (0, 0))  
 screen.blit(instruction\_surface, (0, 0))  
  
 pygame.draw.rect(screen, (236,233,218), container\_rect)  
 pygame.draw.rect(screen, (26, 7, 4), container\_rect, 6)  
  
 title\_font = pygame.font.Font(None, 52)  
 title\_text = title\_font.render("Инструкция к игре", True, (0, 0, 0))  
 title\_rect = title\_text.get\_rect(center=(container\_rect.centerx, container\_rect.top + 40))  
 screen.blit(title\_text, title\_rect)  
  
 text, image\_path = instructions[current\_index]  
  
 wrapped\_text = textwrap.fill(text, width=70) *# 70 - количество символов в строке* if image\_path:  
 image = pygame.image.load(image\_path).convert\_alpha()  
 image\_width = container\_width - 40 *# Отступы по 20 пикселей с каждой стороны в инструкции* image\_height = int(image.get\_height() \* (image\_width / image.get\_width()))  
 image = pygame.transform.scale(image, (image\_width, image\_height))  
 screen.blit(image, (container\_rect.left + 20, container\_rect.top + 100))  
 text\_y\_position = container\_rect.top + 120 + image\_height  
 else:  
 text\_y\_position = container\_rect.top + 100  
  
 for i, line in enumerate(wrapped\_text.split('\n')):  
 text\_surface = font.render(line, True, (0, 0, 0))  
 screen.blit(text\_surface, (container\_rect.left + 20, text\_y\_position + i \* 30))  
  
 *# Пагинация* page\_text = f"{current\_index + 1}/{len(instructions)}"  
 page\_surface = font.render(page\_text, True, (0, 0, 0))  
 page\_rect = page\_surface.get\_rect(center=(container\_rect.centerx, container\_rect.bottom - 30))  
 screen.blit(page\_surface, page\_rect)  
  
 left\_button = pygame.Rect(container\_rect.left + 20, container\_rect.bottom - 60, 50, 40)  
 right\_button = pygame.Rect(container\_rect.right - 70, container\_rect.bottom - 60, 50, 40)  
 pygame.draw.rect(screen, (70, 130, 180), left\_button)  
 pygame.draw.rect(screen, (70, 130, 180), right\_button)  
  
 left\_text = font.render("<-", True, (255, 255, 255))  
 right\_text = font.render("->", True, (255, 255, 255))  
 screen.blit(left\_text, left\_button.move(15, 5))  
 screen.blit(right\_text, right\_button.move(15, 5))  
  
 close\_button\_size = 40  
 close\_button\_rect = pygame.Rect(container\_rect.right - close\_button\_size - 10,  
 container\_rect.top + 10,  
 close\_button\_size,  
 close\_button\_size)  
  
 pygame.draw.rect(screen, (255, 50, 20), close\_button\_rect)  
 close\_text = font.render("X", True, (255, 255, 255))  
 screen.blit(close\_text, close\_button\_rect.move(10, 5))  
  
 pygame.display.flip()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 elif event.type == pygame.MOUSEBUTTONDOWN:  
 if left\_button.collidepoint(event.pos):  
 current\_index = (current\_index - 1) % len(instructions)  
 elif right\_button.collidepoint(event.pos):  
 current\_index = (current\_index + 1) % len(instructions)  
 elif close\_button\_rect.collidepoint(event.pos):  
 showing\_instructions = False  
 elif event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_LEFT:  
 current\_index = (current\_index - 1) % len(instructions)  
 elif event.key == pygame.K\_RIGHT:  
 current\_index = (current\_index + 1) % len(instructions)  
 elif event.key == pygame.K\_ESCAPE:  
 showing\_instructions = False  
  
 def show\_new\_game\_menu(self):  
 global current\_turn, play\_against\_pc, player\_color  
 saved\_state = {  
 'pieces': self.pieces.copy(),  
 'captured\_pieces': self.captured\_pieces.copy(),  
 'moves\_history': self.moves\_history.copy(),  
 'current\_turn': current\_turn,  
 'game\_over': self.game\_over  
 }  
  
 menu\_surface = pygame.Surface(screen.get\_size(), pygame.SRCALPHA)  
 for i in range(screen\_height):  
 alpha = min(128, i // 16)  
 pygame.draw.line(menu\_surface, (70, 130, 180, alpha), (0, i), (screen\_width, i))  
  
 menu\_width = 800  
 menu\_height = 400  
 menu\_rect = pygame.Rect((screen\_width - menu\_width) // 2,  
 (screen\_height - menu\_height) // 2,  
 menu\_width, menu\_height)  
  
 options = {  
 "Противник:": ["Игрок", "ПК"],  
 "Сторона:": ["Белые", "Черные"],  
 "Набор фигур:": ["Король, ферзь", "Король, ладья, 2 пешки"]  
 }  
  
 selected\_options = {  
 "Противник:": 0,  
 "Сторона:": 0,  
 "Набор фигур:": 0  
 }  
  
 selecting = True  
 start\_new\_game = False  
 while selecting:  
 screen.blit(background, (0, 0))  
 screen.blit(menu\_surface, (0, 0))  
  
 pygame.draw.rect(screen, (235, 235, 208), menu\_rect)  
 pygame.draw.rect(screen, (100, 100, 100), menu\_rect, 2)  
  
 title = font.render("Настройки новой игры", True, (0, 0, 0))  
 title\_rect = title.get\_rect(center=(screen\_width // 2, menu\_rect.top + 30))  
 screen.blit(title, title\_rect)  
  
 for i, (option, values) in enumerate(options.items()):  
 option\_text = font.render(option, True, (0, 0, 0))  
 screen.blit(option\_text, (menu\_rect.left + 50, menu\_rect.top + 80 + i \* 80))  
  
 for j, value in enumerate(values):  
 value\_text = font.render(value, True, (0, 0, 0) if selected\_options[option] != j else (255, 0, 0))  
 screen.blit(value\_text, (menu\_rect.left + 250 + j \* 200, menu\_rect.top + 80 + i \* 80))  
  
 start\_button = Button(menu\_rect.centerx - 5, menu\_rect.bottom - 70, 200, 50, "Начать игру")  
 start\_button.draw(screen)  
  
 back\_button = Button(menu\_rect.centerx - 240, menu\_rect.bottom - 70, 200, 50, "Назад")  
 back\_button.draw(screen)  
  
 pygame.display.flip()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 elif event.type == pygame.MOUSEBUTTONDOWN:  
 mouse\_pos = pygame.mouse.get\_pos()  
 if start\_button.rect.collidepoint(mouse\_pos):  
 start\_new\_game = True  
 selecting = False  
 elif back\_button.rect.collidepoint(mouse\_pos):  
 self.pieces = saved\_state['pieces']  
 self.captured\_pieces = saved\_state['captured\_pieces']  
 self.moves\_history = saved\_state['moves\_history']  
 current\_turn = saved\_state['current\_turn']  
 self.game\_over = saved\_state['game\_over']  
 selecting = False  
 else:  
 for i, (option, values) in enumerate(options.items()):  
 for j, value in enumerate(values):  
 value\_rect = pygame.Rect(menu\_rect.left + 200 + j \* 150, menu\_rect.top + 80 + i \* 80,  
 150, 40)  
 if value\_rect.collidepoint(mouse\_pos):  
 selected\_options[option] = j  
  
 elif event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_ESCAPE:  
 self.pieces = saved\_state['pieces']  
 self.captured\_pieces = saved\_state['captured\_pieces']  
 self.moves\_history = saved\_state['moves\_history']  
 current\_turn = saved\_state['current\_turn']  
 self.game\_over = saved\_state['game\_over']  
 selecting = False  
  
 if start\_new\_game:  
 self.clear\_board()  
 if selected\_options["Набор фигур:"] == 0:  
 self.pieces = self.setup\_pieces("king\_queen",  
 "white" if selected\_options["Сторона:"] == 0 else "black")  
 else:  
 self.pieces = self.setup\_pieces("king\_rook\_pawns",  
 "white" if selected\_options["Сторона:"] == 0 else "black")  
 self.moves\_history = []  
 self.current\_save = None  
 self.captured\_pieces = []  
 self.game\_over = False  
 current\_turn = "white"  
 global play\_against\_pc  
 play\_against\_pc = selected\_options["Противник:"] == 1  
 player\_color = "white" if selected\_options["Сторона:"] == 0 else "black"  
  
 def show\_login\_menu(self):  
 global is\_authenticated, current\_user  
 menu\_surface = pygame.Surface(screen.get\_size(), pygame.SRCALPHA)  
 for i in range(screen\_height):  
 alpha = min(128, i // 16)  
 pygame.draw.line(menu\_surface, (70, 130, 180, alpha), (0, i), (screen\_width, i))  
  
 menu\_width = 500  
 menu\_height = 400  
 menu\_rect = pygame.Rect((screen\_width - menu\_width) // 2,  
 (screen\_height - menu\_height) // 2,  
 menu\_width, menu\_height)  
  
 pygame.draw.rect(menu\_surface, (235, 235, 208), menu\_rect)  
 pygame.draw.rect(menu\_surface, (26, 7, 4), menu\_rect, 6)  
  
 title = font.render("Авторизация", True, (0, 0, 0))  
 title\_rect = title.get\_rect(center=(screen\_width // 2, menu\_rect.top + 30))  
  
 username\_input = ""  
 password\_input = ""  
 active\_input = None  
  
 login\_button = Button(menu\_rect.centerx - 125, menu\_rect.bottom - 190, 250, 50, "Войти")  
 register\_button = Button(menu\_rect.centerx - 125, menu\_rect.bottom - 130, 250, 50, "Регистрация")  
 back\_button = Button(menu\_rect.centerx - 125, menu\_rect.bottom - 70, 250, 50, "Назад")  
  
 selecting = True  
 message = ""  
 message\_timer = 0  
 while selecting:  
 screen.blit(background, (0, 0))  
 screen.blit(menu\_surface, (0, 0))  
 screen.blit(title, title\_rect)  
  
 username\_text = font.render("Никнейм:", True, (0, 0, 0))  
 screen.blit(username\_text, (menu\_rect.left + 20, menu\_rect.top + 80))  
 username\_input\_rect = pygame.Rect(menu\_rect.left + 150, menu\_rect.top + 80, 300, 30)  
 pygame.draw.rect(screen, (255, 255, 255), username\_input\_rect)  
 pygame.draw.rect(screen, (0, 0, 0), username\_input\_rect, 2)  
 username\_surface = font.render(username\_input + ('|' if active\_input == "username" else ''), True,  
 (0, 0, 0))  
 screen.blit(username\_surface, (username\_input\_rect.x + 5, username\_input\_rect.y + 5))  
  
 password\_text = font.render("Пароль:", True, (0, 0, 0))  
 screen.blit(password\_text, (menu\_rect.left + 20, menu\_rect.top + 130))  
 password\_input\_rect = pygame.Rect(menu\_rect.left + 150, menu\_rect.top + 130, 300, 30)  
 pygame.draw.rect(screen, (255, 255, 255), password\_input\_rect)  
 pygame.draw.rect(screen, (0, 0, 0), password\_input\_rect, 2)  
 password\_surface = font.render("\*" \* len(password\_input) + ('|' if active\_input == "password" else ''),  
 True, (0, 0, 0))  
 screen.blit(password\_surface, (password\_input\_rect.x + 5, password\_input\_rect.y + 5))  
  
 login\_button.draw(screen)  
 register\_button.draw(screen)  
 back\_button.draw(screen)  
  
 if message and message\_timer > 0:  
 message\_surface = pygame.Surface((screen\_width, 100), pygame.SRCALPHA)  
 message\_surface.fill((0, 0, 0, 128))  
 text\_shadow = font.render(message, True, (0, 0, 0))  
 text\_surface = font.render(message, True, (255, 255, 255))  
 text\_rect = text\_surface.get\_rect(center=(screen\_width // 2, 50))  
 screen.blit(message\_surface, (0, 0))  
 screen.blit(text\_shadow, (text\_rect.x + 2, text\_rect.y + 2))  
 screen.blit(text\_surface, text\_rect)  
 message\_timer -= 1  
  
 pygame.display.flip()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 elif event.type == pygame.MOUSEBUTTONDOWN:  
 if username\_input\_rect.collidepoint(event.pos):  
 active\_input = "username"  
 elif password\_input\_rect.collidepoint(event.pos):  
 active\_input = "password"  
 elif login\_button.rect.collidepoint(event.pos):  
 if self.authenticate(username\_input, password\_input):  
 is\_authenticated = True  
 current\_user = username\_input  
 self.buttons['login'].disabled = True  
 self.message = f"Добрый день, {username\_input}!"  
 self.message\_timer = 180  
 selecting = False  
 else:  
 message = "Неверный логин или пароль!"  
 message\_timer = 180  
 elif register\_button.rect.collidepoint(event.pos):  
 self.show\_register\_menu()  
 selecting = False  
 elif back\_button.rect.collidepoint(event.pos):  
 selecting = False  
 elif event.type == pygame.KEYDOWN:  
 if active\_input == "username":  
 if event.key == pygame.K\_BACKSPACE:  
 username\_input = username\_input[:-1]  
 elif event.key not in [pygame.K\_ESCAPE, pygame.K\_TAB, pygame.K\_DELETE, pygame.K\_SPACE] and len(username\_input) < 16:  
 username\_input += event.unicode  
 elif active\_input == "password":  
 if event.key == pygame.K\_BACKSPACE:  
 password\_input = password\_input[:-1]  
 elif event.key not in [pygame.K\_ESCAPE, pygame.K\_TAB, pygame.K\_DELETE, pygame.K\_SPACE] and len(password\_input) < 16:  
 password\_input += event.unicode  
  
 def show\_register\_menu(self):  
 menu\_surface = pygame.Surface(screen.get\_size(), pygame.SRCALPHA)  
 for i in range(screen\_height):  
 alpha = min(128, i // 16)  
 pygame.draw.line(menu\_surface, (70, 130, 180, alpha), (0, i), (screen\_width, i))  
  
 menu\_width = 500  
 menu\_height = 400  
 menu\_rect = pygame.Rect((screen\_width - menu\_width) // 2,  
 (screen\_height - menu\_height) // 2,  
 menu\_width, menu\_height)  
  
 pygame.draw.rect(menu\_surface, (235, 235, 208), menu\_rect)  
 pygame.draw.rect(menu\_surface, (26, 7, 4), menu\_rect, 6)  
  
 title = font.render("Регистрация", True, (0, 0, 0))  
 title\_rect = title.get\_rect(center=(screen\_width // 2, menu\_rect.top + 30))  
  
 username\_input = ""  
 password\_input = ""  
 active\_input = None  
  
 register\_button = Button(menu\_rect.centerx - 125, menu\_rect.bottom - 130, 250, 50, "Зарегистрироваться")  
 back\_button = Button(menu\_rect.centerx - 125, menu\_rect.bottom - 70, 250, 50, "Назад")  
  
 selecting = True  
 message = ""  
 message\_timer = 0  
 while selecting:  
 screen.blit(background, (0, 0))  
 screen.blit(menu\_surface, (0, 0))  
 screen.blit(title, title\_rect)  
  
 username\_text = font.render("Никнейм:", True, (0, 0, 0))  
 screen.blit(username\_text, (menu\_rect.left + 20, menu\_rect.top + 80))  
 username\_input\_rect = pygame.Rect(menu\_rect.left + 150, menu\_rect.top + 80, 300, 30)  
 pygame.draw.rect(screen, (255, 255, 255), username\_input\_rect)  
 pygame.draw.rect(screen, (0, 0, 0), username\_input\_rect, 2)  
 username\_surface = font.render(username\_input + ('|' if active\_input == "username" else ''), True,  
 (0, 0, 0))  
 screen.blit(username\_surface, (username\_input\_rect.x + 5, username\_input\_rect.y + 5))  
  
 password\_text = font.render("Пароль:", True, (0, 0, 0))  
 screen.blit(password\_text, (menu\_rect.left + 20, menu\_rect.top + 130))  
 password\_input\_rect = pygame.Rect(menu\_rect.left + 150, menu\_rect.top + 130, 300, 30)  
 pygame.draw.rect(screen, (255, 255, 255), password\_input\_rect)  
 pygame.draw.rect(screen, (0, 0, 0), password\_input\_rect, 2)  
 password\_surface = font.render("\*" \* len(password\_input) + ('|' if active\_input == "password" else ''),  
 True, (0, 0, 0))  
 screen.blit(password\_surface, (password\_input\_rect.x + 5, password\_input\_rect.y + 5))  
  
 register\_button.draw(screen)  
 back\_button.draw(screen)  
  
 if message and message\_timer > 0:  
 message\_surface = pygame.Surface((screen\_width, 100), pygame.SRCALPHA)  
 message\_surface.fill((0, 0, 0, 128))  
 text\_shadow = font.render(message, True, (0, 0, 0))  
 text\_surface = font.render(message, True, (255, 255, 255))  
 text\_rect = text\_surface.get\_rect(center=(screen\_width // 2, 50))  
 screen.blit(message\_surface, (0, 0))  
 screen.blit(text\_shadow, (text\_rect.x + 2, text\_rect.y + 2))  
 screen.blit(text\_surface, text\_rect)  
 message\_timer -= 1  
  
 pygame.display.flip()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
 elif event.type == pygame.MOUSEBUTTONDOWN:  
 if username\_input\_rect.collidepoint(event.pos):  
 active\_input = "username"  
 elif password\_input\_rect.collidepoint(event.pos):  
 active\_input = "password"  
 elif register\_button.rect.collidepoint(event.pos):  
 if len(username\_input) < 5:  
 message = "Никнейм должен содержать минимум 5 символов!"  
 message\_timer = 180  
 elif len(password\_input) < 5:  
 message = "Пароль должен содержать минимум 5 символов!"  
 message\_timer = 180  
 elif self.is\_username\_taken(username\_input):  
 message = "Данный пользователь уже зарегистрирован"  
 message\_timer = 180  
 else:  
 self.register(username\_input, password\_input)  
 self.show\_message(f"Добро пожаловать, {username\_input}! Пожалуйста, авторизуйтесь!", 180)  
 selecting = False  
 elif back\_button.rect.collidepoint(event.pos):  
 selecting = False  
 elif event.type == pygame.KEYDOWN:  
 if active\_input == "username":  
 if event.key == pygame.K\_BACKSPACE:  
 username\_input = username\_input[:-1]  
 elif event.key not in [pygame.K\_ESCAPE, pygame.K\_TAB, pygame.K\_DELETE, pygame.K\_SPACE] and len(username\_input) < 16:  
 username\_input += event.unicode  
 elif active\_input == "password":  
 if event.key == pygame.K\_BACKSPACE:  
 password\_input = password\_input[:-1]  
 elif event.key not in [pygame.K\_ESCAPE, pygame.K\_TAB, pygame.K\_DELETE, pygame.K\_SPACE] and len(password\_input) < 16:  
 password\_input += event.unicode  
  
 def authenticate(self, username, password):  
 try:  
 with open('users.json', 'r') as f:  
 users = json.load(f)  
 if username in users:  
 hashed\_password = hashlib.sha256(password.encode()).hexdigest()  
 return users[username]['password'] == hashed\_password  
 except FileNotFoundError:  
 return False  
 except json.JSONDecodeError:  
 return False  
 return False  
  
 def is\_username\_taken(self, username):  
 try:  
 with open('users.json', 'r') as f:  
 users = json.load(f)  
 return username in users  
 except FileNotFoundError:  
 return False  
 except json.JSONDecodeError:  
 return False  
  
 def register(self, username, password):  
 try:  
 with open('users.json', 'r') as f:  
 users = json.load(f)  
 except (FileNotFoundError, json.JSONDecodeError):  
 users = {}  
  
 hashed\_password = hashlib.sha256(password.encode()).hexdigest()  
 users[username] = {'password': hashed\_password}  
  
 with open('users.json', 'w') as f:  
 json.dump(users, f)  
  
 *# Create user save directory* user\_save\_dir = os.path.join('saves', username)  
 if not os.path.exists(user\_save\_dir):  
 os.makedirs(user\_save\_dir)  
  
 def draw\_message(self, screen):  
 if self.message and self.message\_timer > 0:  
 message\_surface = pygame.Surface((screen\_width, 100), pygame.SRCALPHA)  
 message\_surface.fill((0, 0, 0, 128))  
 text\_shadow = font.render(self.message, True, (0, 0, 0))  
 text\_surface = font.render(self.message, True, (255, 255, 255))  
 text\_rect = text\_surface.get\_rect(center=(screen\_width // 2, 50))  
 message\_surface.blit(text\_shadow, (text\_rect.x + 4, text\_rect.y + 4))  
 message\_surface.blit(text\_surface, text\_rect)  
 screen.blit(message\_surface, (0, 0))  
 self.message\_timer -= 1  
  
 def evaluate\_board(self):  
 score = 0  
  
 white\_pawns = len([p for p in self.pieces.values() if p.type == 'pawn' and p.color == 'white'])  
 black\_pawns = len([p for p in self.pieces.values() if p.type == 'pawn' and p.color == 'black'])  
  
 for piece in list(self.pieces.values()):  
 value = self.piece\_values[piece.type]  
  
 *# Приоритет развития пешек в начале игры* if piece.type == 'pawn':  
 if (white\_pawns > 0 and piece.color == 'white') or (black\_pawns > 0 and piece.color == 'black'):  
 value \*= 2 *# ценность пешек* if 2 <= piece.position[0] <= 5:  
 if piece.color == 'white' and piece.position[1] > 4:  
 value \*= 1.5  
 elif piece.color == 'black' and piece.position[1] < 3:  
 value \*= 1.5  
  
 if piece.type != 'king':  
 for enemy in list(self.pieces.values()):  
 if enemy.color != piece.color and enemy.type == 'king':  
 dist = abs(enemy.position[0] - piece.position[0]) + abs(enemy.position[1] - piece.position[1])  
 if dist <= 2:  
 value \*= 3.0  
 elif dist <= 4:  
 value \*= 1.5  
  
 if piece.type == 'king':  
 protectors = sum(1 for p in self.pieces.values()  
 if p.color == piece.color and p != piece  
 and abs(p.position[0] - piece.position[0]) + abs(  
 p.position[1] - piece.position[1]) <= 1)  
 value \*= (1 + 0.2 \* protectors)  
  
 if piece.type == 'pawn':  
 if piece.color == 'white':  
 value += (7 - piece.position[1]) \* 5  
 if 2 <= piece.position[0] <= 5:  
 value += 10  
 for other\_pawn in [p for p in self.pieces.values() if  
 p.type == 'pawn' and p.color == 'white' and p != piece]:  
 if piece.position[0] == other\_pawn.position[0]:  
 value -= 10  
 else:  
 value += piece.position[1] \* 5  
 if 2 <= piece.position[0] <= 5:  
 value += 10  
 for other\_pawn in [p for p in self.pieces.values() if  
 p.type == 'pawn' and p.color == 'black' and p != piece]:  
 if piece.position[0] == other\_pawn.position[0]:  
 value -= 10  
  
 if piece.color == 'white':  
 score += value  
 else:  
 score -= value  
  
 if self.is\_checkmate('black'):  
 score += 10000  
 elif self.is\_checkmate('white'):  
 score -= 10000  
  
 return score  
  
 def get\_best\_move(self, depth, color, alpha=float('-inf'), beta=float('inf'), use\_transposition=True):  
 if depth == 0 or self.game\_over:  
 return None, self.evaluate\_board()  
  
 best\_move = None  
 if color == 'white':  
 best\_value = float('-inf')  
 else:  
 best\_value = float('inf')  
  
 if use\_transposition:  
 board\_hash = self.hash\_board()  
 if board\_hash in self.transposition\_table:  
 return self.transposition\_table[board\_hash]  
  
 pieces = [p for p in self.pieces.values() if p.color == color]  
  
 pieces.sort(key=lambda p: self.\_move\_priority(p, p.position), reverse=True)  
  
 for piece in pieces:  
 valid\_moves = self.get\_valid\_moves(piece)  
 valid\_moves.sort(key=lambda m: self.\_move\_priority(piece, m), reverse=True)  
  
 for move in valid\_moves:  
 target\_piece = self.get\_piece\_at(move)  
  
 original\_pos = piece.position  
 original\_pieces = self.pieces.copy()  
  
 piece.move\_to(move)  
 if target\_piece:  
 target\_key = None  
 for k, v in original\_pieces.items():  
 if v == target\_piece:  
 target\_key = k  
 break  
 if target\_key:  
 del self.pieces[target\_key]  
  
 was\_promoted = False  
 if piece.type == 'pawn':  
 if (color == 'white' and move[1] == 0) or (color == 'black' and move[1] == 7):  
 piece.type = 'queen'  
 was\_promoted = True  
  
 \_, evaluation = self.get\_best\_move(depth - 1, 'black' if color == 'white' else 'white', alpha, beta, use\_transposition)  
  
 if was\_promoted:  
 piece.type = 'pawn'  
 piece.move\_to(original\_pos)  
 self.pieces = original\_pieces.copy()  
  
 if color == 'white':  
 if evaluation > best\_value:  
 best\_value = evaluation  
 best\_move = (piece, move)  
 alpha = max(alpha, evaluation)  
 else:  
 if evaluation < best\_value:  
 best\_value = evaluation  
 best\_move = (piece, move)  
 beta = min(beta, evaluation)  
  
 if beta <= alpha:  
 break  
  
 if use\_transposition:  
 self.transposition\_table[board\_hash] = (best\_move, best\_value)  
  
 return best\_move, best\_value  
  
 def hash\_board(self):  
 return hash(tuple((p.color, p.type, p.position) for p in self.pieces.values()))  
  
 def \_move\_priority(self, piece, move):  
 priority = 0  
 target = self.get\_piece\_at(move)  
  
 if target:  
 priority += self.piece\_values[target.type] \* 15  
  
 if piece.type == 'pawn':  
 if (piece.color == 'white' and move[1] == 0) or (piece.color == 'black' and move[1] == 7):  
 priority += 1000  
  
 if 2 <= move[0] <= 5:  
 priority += 100  
  
 if piece.color == 'white':  
 priority += (7 - move[1]) \* 20  
 else:  
 priority += move[1] \* 20  
  
 if 2 <= move[0] <= 5 and 2 <= move[1] <= 5:  
 priority += 80  
  
 for enemy in self.pieces.values():  
 if enemy.color != piece.color and enemy.type == 'king':  
 dist = abs(enemy.position[0] - move[0]) + abs(enemy.position[1] - move[1])  
 if dist <= 2:  
 priority += 200  
 elif dist <= 4:  
 priority += 100  
  
 return priority  
  
 def get\_best\_promotion(self, piece, move):  
 return 'queen'  
  
  
def main():  
 global play\_against\_pc, current\_turn, player\_color  
 board = ChessBoard()  
 last\_move\_time = 0  
 move\_delay = 200  
  
 while True:  
 current\_time = pygame.time.get\_ticks()  
  
 if (play\_against\_pc and current\_turn != player\_color and  
 not board.game\_over and current\_time - last\_move\_time >= move\_delay):  
  
 best\_move, \_ = board.get\_best\_move(3, current\_turn)  
  
 if best\_move:  
 piece, move = best\_move  
 board.handle\_click((  
 board.board\_offset\_x + piece.position[0] \* tile\_size + tile\_size // 2,  
 board.board\_offset\_y + piece.position[1] \* tile\_size + tile\_size // 2  
 ))  
  
 pygame.time.wait(100)  
  
 board.handle\_click((  
 board.board\_offset\_x + move[0] \* tile\_size + tile\_size // 2,  
 board.board\_offset\_y + move[1] \* tile\_size + tile\_size // 2  
 ))  
  
 if piece.type == 'pawn' and (move[1] == 0 or move[1] == 7):  
 piece.type = 'queen'  
  
 last\_move\_time = current\_time  
  
 screen.fill((255, 255, 255))  
 board.handle\_events()  
 board.draw(screen)  
 pygame.display.flip()  
 clock.tick(60)  
  
if \_\_name\_\_ == "\_\_main\_\_":

# **Заключение**

В результате выполнения курсовой работы была разработана программа "Шахматный эндшпиль" на языке Python с использованием библиотеки Pygame. В ходе разработки были успешно решены следующие задачи:

1. Создана полноценная игровая система с графическим интерфейсом, включающая:

   - Реализацию шахматной доски и фигур

   - Систему ходов с учетом всех правил шахмат

   - Механику эндшпиля с различными наборами фигур

   - Искусственный интеллект для игры против компьютера

2. Разработана система авторизации и хранения данных:

   - Регистрация и авторизация пользователей

   - Сохранение и загрузка игровых партий

   - Ведение истории ходов

   - Защита пользовательских данных

3. Внедрен искусственный интеллект с использованием:

   - Алгоритма минимакс с альфа-бета отсечением

   - Эвристической оценки позиций

   - Транспозиционной таблицы для оптимизации

   - Приоритезации ходов для улучшения качества игры

4. Реализован удобный пользовательский интерфейс:

   - Интуитивно понятное управление

   - Визуальная обратная связь

   - Подсветка возможных ходов

   - Система подсказок и сообщений

Программная документация разработана в соответствии с ГОСТ 19.201-78 "Техническое задание. Требования к содержанию и оформлению". Код программы структурирован и прокомментирован согласно стандартам.

В процессе разработки были получены практические навыки:

- Объектно-ориентированного программирования

- Работы с графическими библиотеками

- Создания искусственного интеллекта для игр

- Проектирования пользовательских интерфейсов

- Работы с файловой системой

- Обеспечения безопасности данных

Созданная программа полностью соответствует поставленным требованиям и может быть использована как для обучения шахматным эндшпилям, так и для развлекательных целей.

# **Источники, использованные при разработке**

1. Пирогов, А.А. Python для начинающих. Изучаем программирование с нуля. – М.: Издательство "БХВ-Петербург", 2020.
2. "Документация Pygame" [Электронный ресурс] // Официальный сайт библиотеки Pygame: https://www.pygame.org/docs/ – Режим доступа: свободный.
3. ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения». – М.: Издательство стандартов, 1990.
4. В.В. Шишкин, Д.С. Афонин РАЗРАБОТКА ЛОГИЧЕСКИХ КОМПЬЮТЕРНЫХ ИГР С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ В СРЕДЕ ПИТОН
5. Лутц М. Изучаем Python. - СПб.: Символ-Плюс, 2019. - 1280 с.
6. Свейгарт Эл. Автоматизация рутинных задач с помощью Python. - М.: Вильямс, 2020. - 592 с.
7. Доусон М. Программируем на Python. - СПб.: Питер, 2019. - 416 с.
8. Pygame Documentation. [Электронный ресурс]. - Режим доступа: https://www.pygame.org/docs/
9. Python Documentation. [Электронный ресурс]. - Режим доступа: https://docs.python.org/3/

# **ПРИЛОЖЕНИЕ 1**

***MINDMAP***

# **ПРИЛОЖЕНИЕ 2**

**АННОТАЦИЯ**

Проект представляет собой интерактивную шахматную игру - эндшпиль, реализованную с использованием библиотеки Pygame. Основная цель проекта - создание платформы для игры в шахматы, которая поддерживает как игру между двумя игроками, так и игру против компьютера

**Основные функции и возможности:**

1. **Графический интерфейс**:

* Использование Pygame для создания визуально привлекательного интерфейса, включающего шахматную доску, фигуры и элементы управления.
* Поддержка полноэкранного режима для более комфортной игры.

2. **Аутентификация пользователей**:

* Возможность регистрации новых пользователей и входа в систему для сохранения личных данных и прогресса.
* Хранение данных пользователей в формате JSON с использованием хеширования паролей для обеспечения безопасности.

3. **Игра против компьютера**:

* Реализация алгоритма для игры против компьютера, который включает оценку доски и выбор оптимального хода.
* Использование таблицы транспозиций для оптимизации поиска лучшего хода.

**4. Сохранение и загрузка игры**:

* Возможность сохранять текущий прогресс игры в файл и загружать его в будущем.
* Поддержка нескольких сохранений для каждого пользователя.

5. **Интерактивные инструкции**:

* Встроенные инструкции, которые помогают пользователям понять правила игры и управление.
* Пошаговое руководство с иллюстрациями для облегчения обучения.

6. **Различные режимы игры**:

* Возможность выбора между игрой против другого игрока или компьютера.
* Настройка начального набора фигур и цвета стороны для более разнообразного игрового опыта.

7. **Звуковые эффекты**:

* Воспроизведение звуковых эффектов при выполнении ходов для улучшения погружения в игру.

**8. Обработка событий и управление игрой**:

* Реализация системы обработки событий для управления игрой, включая клики мыши и нажатия клавиш.
* Поддержка различных игровых состояний, таких как шах, мат и пат.

Проект демонстрирует навыки работы с графическими интерфейсами, алгоритмами искусственного интеллекта и управления состоянием игры. Он может служить основой для дальнейшего развития, включая добавление более сложных алгоритмов ИИ, расширенные возможности для многопользовательской игры и улучшение пользовательского интерфейса.

# **ПРИЛОЖЕНИЕ 3**

**Инструкция по компиляции игры в исполняемый .exe файл**

Для компиляции проекта на Python в исполняемый файл .exe, используйте библиотеку PyInstaller. Следуйте приведенным ниже шагам:

**1**. **Установка PyInstaller**:

1.1. Убедитесь, что Python и pip установлены на вашем компьютере.

1.2. Установите PyInstaller, выполнив команду:

pip install pyinstaller

**2. Подготовка проекта**:

2.1. Убедитесь, что все необходимые файлы и ресурсы (например, изображения, звуки) находятся в правильных директориях относительно основного скрипта chess/ishodnik.py.

3. **Создание исполняемого файла**:

3.1. Перейдите в директорию, где находится ваш скрипт chess/ishodnik.py.

3.2. Выполните следующую команду для создания .exe файла:

pyinstaller --onefile --windowed chess/ishodnik.py

*Прим.: --onefile: создает один исполняемый файл. --windowed: убирает консольное окно (только для GUI приложений).*

4. **Проверка результата**:

4.1. После выполнения команды, в директории dist появится файл ishodnik.exe. Это и есть исполняемый файл.

5. **Тестирование**:

5.1. Запустите ishodnik.exe из директории dist и убедитесь, что программа работает корректно.

6. **Упаковка ресурсов**:

Проект использует внешние файлы (изображения или звуки), убедитесь, что они включены в сборку.

PyInstaller автоматически включает файлы, которые импортируются в коде, но для дополнительных файлов может потребоваться указать их вручную.

Пример команды с добавлением ресурсов:

pyinstaller --onefile --windowed --add-data "assets;assets" chess/ishodnik.py

Здесь assets;assets указывает PyInstaller включить папку assets в сборку.

7. **Настройка файла спецификации (опционально)**:

7.1. PyInstaller создает файл спецификации .spec, который можно настроить для более сложных конфигураций сборки.

7.2. Отредактируйте этот файл и затем выполните:

pyinstaller your\_project.spec

Убедитесь, что все зависимости и ресурсы правильно настроены, чтобы избежать ошибок при запуске .exe файла