МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | |  | |  |  |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | |  | | РУКОВОДСТВО ПРОГРАММИСТА  на курсовую работу  по дисциплине «Алгоритмы и структуры данных»  Тема «Компьютерная игра эндшпиль Король, 2 слона-Король, ладья» | | | | | | |
|  | | |  | |  | | | |
|  | | Исполнитель  студент гр. ИСТбд-22  Камчаров М.С.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | | | |
|  | | 2024 | | | | | | |

**1. Введение**

Данный проект представляет собой полноценную шахматную игру с современным графическим интерфейсом, который делает игровой процесс максимально удобным и привлекательным. Игра включает в себя широкий спектр возможностей, позволяя пользователям наслаждаться игрой как друг против друга, так и с искусственным интеллектом.

Основные функции:

* Поддержка игры между двумя игроками или игроком и ботом.
* Регистрация и авторизация пользователей.
* Настраиваемые параметры игры, включая выбор противника.
* Визуализация шахматной доски с возможностью подсветки доступных ходов.



Рисунок 1 – Подсвечивание ходов.

**2. Установка и настройка окружения**

**Зависимости:**

* Python версии 3.7 или выше.
* Библиотеки:
  + pygame
  + pygame-widgets
  + bcrypt
  + tkinter

**Инструкции:**

* Убедитесь, что Python установлен и доступен в командной строке.
* Установите зависимости:
  + pip install pygame pygame-widgets bcrypt
* Скачайте проект и перейдите в его корневую папку.
* Запустите игру командой:
  + python main.py

**3. Обзор архитектуры**

Программа состоит из нескольких модулей, каждый из которых отвечает за отдельный аспект функциональности шахматной игры. Основные файлы и их роли:

* `main.py`: точка входа в программу, управляет запуском игры, обработкой событий и игровым циклом.
* `chessboard.py`: реализует шахматную доску, обработку ходов и основные правила игры.
* `infopanel.py`: управляет информационной панелью, включая отображение таймеров и истории ходов.
* `menu.py`: реализует элементы меню для управления игрой.
* `config.py`: содержит класс для загрузки и сохранения настроек конфигурации игры.
* `resloader.py`: обеспечивает загрузку и кеширование ресурсов, таких как изображения и шрифты.
* Файлы фигур (`Bishop.py`, `Rook.py`, `King.py` и др.): реализуют поведение конкретных фигур.

**4. Ключевые компоненты**

**4.1. main.py**

* **Класс SettingsWindow:**

Отвечает за отображение окон авторизации, регистрации и выбора противника.

* **center\_window(width=None, height=None)** Центрирует окно на экране в зависимости от его размеров.
  + **Параметры:**
    - width (int) — ширина окна (опционально).
    - height (int) — высота окна (опционально).
  + **Пример использования:**
* **login()** Управляет процессом авторизации пользователя. Поддерживает регистрацию новых учетных записей с сохранением хэша пароля.
  + **Функциональность:**
* **Проверяет существующие учетные записи в файле accounts.json.**
* **Отображает сообщения об ошибке или успехе операции.**
  + **Пример использования:**
  + if settings\_window.login().show():

print("Успешный вход")

* **choose\_enemy()** Показывает окно выбора режима игры — против другого игрока или бота.
  + **Возвращаемое значение:**

"player" или "bot" в зависимости от выбора.

* **show()** Запускает главное окно авторизации и управления.
* **Класс Chess:**

Отвечает за реализацию логики игры и взаимодействие с пользователем.

* **\_\_init\_\_()** Выполняет начальную инициализацию игры, включая настройку экрана и создание объектов.
  + **Функциональность:**
    - Устанавливает размер экрана в зависимости от разрешения.
    - Инициализирует доску (Board), загрузчик ресурсов (ResLoader).
* **draw(events)** Отвечает за визуализацию состояния игры, включая доску, фигуры и панель информации.
  + **Параметры:**
    - events (list) — список событий Pygame.
* **bot\_move(bot, qres)** Реализует выполнение хода ИИ, используя алгоритм Minimax.
  + **Параметры:**
    - bot (Minimax) — объект ИИ.
    - qres (queue.Queue) — очередь для обмена данными.
* **player\_move(events)** Обрабатывает действия игрока, такие как выбор фигуры и её перемещение.
* **start\_game()** Основной цикл игры, в котором управляются состояния игры, выполняются ходы и обновляется интерфейс.



Рисунок 2 – Основной экран игры с активной партией.

**4.2. chessboard.py**

* **Класс Board:**
  + Основной класс для работы с шахматной доской.
  + Методы:
* new\_game(fen=None): Инициализирует новую партию на основе позиции FEN.
* save\_game(): Сохраняет текущую позицию в конфигурационный файл.
* game\_over(result=None): Проверяет завершение игры и устанавливает результат.
* draw(): Отрисовывает доску, фигуры и вспомогательные элементы, такие как подсветка ходов.
* on\_click(mx, my): Обрабатывает щелчки мыши по доске для выполнения хода.
* is\_in\_check(color): Проверяет, находится ли король указанного цвета под шахом.
* is\_in\_checkmate(color): Проверяет состояние мата для указанного цвета.



Рисунок 3 – Пример подсвечивания (шах черным).



Рисунок 4 – Пример подсвечивания и также вывод в меню информацию (мат черным).



Рисунок 5 – Пример вывода информации в меню (пат белым).

* **Класс Square:**

Клетка доски, отображающая фигуры и состояния (например, подсветка).

* **\_\_init\_\_(x, y, width, height, board)** Инициализирует клетку с заданными координатами и размерами.
  + **Параметры:**
    - x, y (int) — координаты клетки.
    - width, height (int) — размеры клетки.
    - board (Board) — ссылка на объект доски.
* **set\_figure(figure)** Устанавливает фигуру на клетку. Если передан None, клетка становится пустой.
* **draw()** Отрисовывает клетку и фигуру (если она присутствует).



Рисунок 6 – Крупный план клетки с фигурой.

**4.3. bot.py**

* **Класс Minimax:**
  + Логика шахматного бота на основе алгоритма Minimax.
  + Методы:
* evaluateBoard(): Оценивает позицию на доске, используя весовые коэффициенты для фигур.
* getBestMove(qres): Возвращает лучший ход для текущей позиции.
* minimax(depth, alpha, beta, is\_maximizing): Основной рекурсивный метод алгоритма Minimax с обрезкой альфа-бета.

**4.4. infopanel.py**

* **Класс InfoPanel:**
  + Отвечает за отображение информации о ходе игры, включая таймеры и историю ходов.
  + Методы:
* draw(events, bot\_thread): Отрисовывает панель информации, включая меню, таймеры и историю.
* on\_click(mx, my): Реагирует на клики мыши в области панели.
* **Класс Timers:**
  + Управляет игровыми таймерами.
  + Методы:
* update(color, ms): Обновляет оставшееся время для указанного цвета.
* reset(): Сбрасывает таймеры до начального значения.

**4.5. menu.py**

* **Класс Menu:**
  + Управляет отображением и функциональностью кнопок меню (например, новая игра, сохранение).
  + Методы:
* button\_new\_game(): Создает кнопку "Новая игра", которая запускает партию заново.
* button\_save\_config(): Создает кнопку "Сохранить", которая сохраняет текущую позицию.
* draw(events): Отрисовывает элементы меню и обрабатывает события кнопок.



Рисунок 7 – Отображение информации о ходе игры, включая таймеры и историю ходов.

**4.6. Файлы фигур (bishop.py, rook.py, king.py)**

* **Общие принципы:**
  + Каждая фигура наследуется от базового класса Figure.
  + Важно переопределить метод get\_possible\_moves(), который возвращает список возможных ходов фигуры с учетом текущей позиции.

**Bishop (слон):**

* Методы:
  + get\_possible\_moves(): Генерирует все возможные диагональные ходы для слона.

**Rook (ладья):**

* Методы:
  + get\_possible\_moves(): Генерирует все возможные ходы по горизонтали и вертикали для ладьи.

**King (король):**

* Методы:
  + get\_possible\_moves(): Генерирует все возможные ходы для короля (на одну клетку в любом направлении).

**4.7. config.py**

* **Класс Config:**
  + Управляет настройками игры и их сохранением в файл settings.json.
  + Методы:
* load\_config(): Загружает параметры из файла настроек, если он существует.
* save\_config(): Сохраняет текущие параметры в файл settings.json.
* **Ключевые параметры:**
  + START\_POSITION: Стартовая позиция в формате FEN.
  + PLAYER\_COLOR: Цвет фигур игрока.
  + PLAYER\_IS\_BOT: Флаг, указывающий, играет ли бот за игрока.
  + ENEMY\_IS\_PLAYER: Флаг, указывающий, является ли противник человеком.
  + DIFFICULTY: Уровень сложности бота.
  + TIME\_LIMIT: Лимит времени для каждого игрока.

**5. Работа с графическим интерфейсом**

**5.1. Pygame**

**Pygame** используется для отображения шахматной доски, фигур и графических элементов интерфейса.

* **Создание окна:**
  + Окно игры создается с помощью pygame.display.set\_mode().
  + Пример: self.screen = pygame.display.set\_mode((800, 600))
* **Отрисовка элементов:**
  + Каждый графический элемент (доска, фигуры, подсветка ходов) рисуется с помощью методов draw().
  + Используются функции:
* pygame.draw.rect(): Рисование прямоугольников (например, клеток доски).
* pygame.draw.circle(): Рисование подсветки.
* blit(): Вывод изображений фигур на экран.
* **События:**
  + Обработка событий (щелчки мыши, нажатия клавиш) через цикл pygame.event.get().
  + Пример:

for event in pygame.event.get():

if event.type == pygame.MOUSEBUTTONDOWN:

handle\_click(event.pos)

**5.2. Tkinter**

**Tkinter** используется для создания окон регистрации и выбора противника.

* **Настройка окон:**
  + Создание окна с помощью Tk():

self.root = Tk()

self.root.title("Авторизация")

* + Центрирование окна с использованием метода center\_window().
* **Элементы интерфейса:**
  + Используются виджеты ttk для создания кнопок, полей ввода и меток:

user\_entry = ttk.Entry(frm, textvariable=user\_var, width=25)

login\_btn = ttk.Button(frm, text="Войти", command=enter)

* **События:**
  + Привязка действий к кнопкам через параметр command (например, обработка входа или регистрации).
  + Tkinter отлично подходит для создания компактных окон, которые дополняют основной интерфейс Pygame.

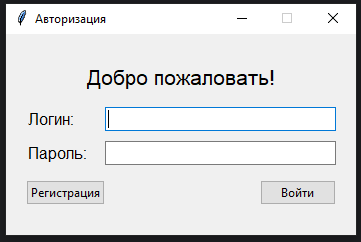
**

Рисунок 8 – Авторизации на базе Tkinter

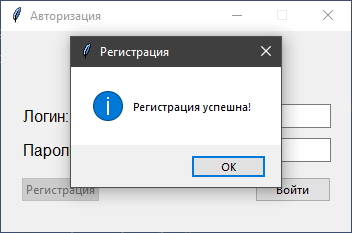


Рисунок 9 – Регистрация на базе Tkinter.

**6. Алгоритмы игры и бота**

Алгоритмы, лежащие в основе работы шахматного бота, являются важной частью общей функциональности игры. Они обеспечивают интеллектуальный уровень игры и делают партию с ботом увлекательной и непредсказуемой.

#### Реализация бота (bot.py)

Бот построен на основе классического алгоритма минимакс с использованием альфа-бета отсечения. Это позволяет минимизировать количество просматриваемых вариантов, сохраняя при этом точность вычислений. Подход обеспечивает высокую производительность даже на глубине анализа, достаточной для создания конкурентного уровня игры.

* **Алгоритм минимакс:** позволяет просчитывать возможные ходы вперёд, оценивая каждую позицию с точки зрения игрока и противника.
* **Альфа-бета отсечение:** значительно снижает количество анализируемых ходов, обрезая заведомо невыгодные варианты.

Пример работы алгоритма:

1. На каждом уровне дерева ходов анализируются все возможные варианты.
2. Бот оценивает позицию с использованием весовых коэффициентов для фигур.
3. Учитываются сценарии атаки, защиты и стратегического контроля центра доски.

#### Метод bot\_move() в main.py

Для выполнения хода бота используется метод bot\_move(). Он запускается в отдельном потоке, что позволяет игре оставаться отзывчивой и не блокировать интерфейс во время расчёта. Это особенно важно для поддержания плавности игрового процесса и приятного пользовательского опыта.

Преимущества многопоточности:

* Пользователь может просматривать доску или совершать другие действия, пока бот обдумывает ход.
* Исключение задержек и зависаний интерфейса.

Этот подход создаёт баланс между сложностью вычислений и комфортом игрока, что делает взаимодействие с игрой интуитивным и приятным.

**Заключение**

Проект представляет собой современное решение для шахматной игры с широкими возможностями как для начинающих игроков, так и для опытных любителей. В основе разработки лежит гибкая модульная структура, которая делает программу удобной для доработки, а также расширения функционала в будущем. Каждый элемент проекта, начиная от визуализации доски и заканчивая алгоритмами искусственного интеллекта, тщательно продуман и направлен на создание комфортного игрового опыта.

Одной из ключевых особенностей является использование мощного алгоритма минимакса для работы бота, что позволяет ему предугадывать возможные ходы соперника и подбирать оптимальную стратегию. В сочетании с графическим интерфейсом, реализованным на основе Pygame, это обеспечивает высокую интерактивность и привлекательность игры.

Важным аспектом является также поддержка гибких настроек, включая изменение уровня сложности, выбор цвета фигур и возможность игры как с реальными противниками, так и с ботом. Удобная система авторизации и регистрации через окна Tkinter позволяет игрокам сохранять свои достижения и статистику, что добавляет проекту персонализации.

Дальнейшее развитие проекта может включать добавление новых режимов игры, расширение возможностей кастомизации, улучшение алгоритмов искусственного интеллекта и адаптацию интерфейса для мобильных устройств. Эти шаги позволят сделать проект ещё более универсальным и востребованным среди широкой аудитории.

Проект демонстрирует не только высокий уровень технической реализации, но и внимание к деталям, что делает его отличным примером качественного программного продукта.