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*** |  | |  | | ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  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**

Данная курсовая работа представляет собой разработку шахматной игры с возможностью взаимодействия с пользователем и встроенным ботом, предоставляющим возможность игры против компьютера. Основное внимание уделено созданию интуитивно понятного интерфейса, реализации шахматной логики и алгоритмов для искусственного интеллекта. Проект разработан на языке Python с использованием библиотеки pygame, которая предоставляет инструменты для создания графического интерфейса, а также для отображения шахматной доски и фигур.

Проект демонстрирует современные подходы к программированию и решению задач высокой вычислительной сложности, делая акцент на надежности, интерактивности и простоте использования. Внедрение искусственного интеллекта позволяет программе обеспечивать интересное взаимодействие, а широкий функционал делает игру подходящей как для начинающих игроков, так и для опытных шахматистов.

**2. Структура проекта**

Код проекта разделен на несколько файлов и классов, которые отвечают за различные аспекты работы программы:

* `main.py`: основной файл, отвечающий за запуск и управление игровым циклом, обработку событий и взаимодействие с пользователем.
* `chessboard.py`: содержит реализацию класса `Board`, который управляет состоянием шахматной доски, обработкой ходов и правилами игры.
* Классы фигур (`Bishop.py`, `Rook.py`, `King.py` и др.): реализуют отдельные типы шахматных фигур, их свойства и допустимые ходы.
* `infopanel.py`: реализует информационную панель для отображения текущего состояния игры, истории ходов и времени, отведенного на игру.
* `menu.py`: предоставляет функционал управления игрой, включая кнопки для начала новой игры и сохранения конфигурации.
* `config.py`: обрабатывает конфигурационные данные, включая стартовые настройки и параметры игрока/бота.
* `resloader.py`: отвечает за загрузку и кеширование ресурсов, таких как шрифты и изображения.

**3. Основные функции программы**

* Запуск игры: Файл `main.py` содержит класс `Chess`, который управляет созданием окна, инициализацией доски и началом игрового цикла с использованием библиотеки `pygame`.
* Обработка ходов: Пользователь может перемещать фигуры путем нажатия мыши, что обрабатывается методом `on\_click()` класса `Board`. Если текущий игрок — бот, метод `bot\_move()` рассчитывает оптимальный ход с использованием алгоритмов поиска.
* Отображение интерфейса: Метод `draw()` класса `Board` отрисовывает клетки доски, фигуры и элементы интерфейса. Класс `InfoPanel` добавляет отображение информации о ходе игры, истории и таймерах.
* Управление игровыми состояниями:

Для управления игровыми состояниями используются специальные методы:

• is\_in\_check(): Проверяет, находится ли король под шахом.

• is\_in\_checkmate(): Проверяет, находится ли король под матом (мат - положение, при котором король под шахом и не имеет возможных ходов для ухода от шаха).

• game\_over(): Определяет завершение игры (мат, пат или ничья). Этот метод анализирует текущее состояние доски и объявляет победителя или ничью.

**4. Обоснование импортируемых библиотек**

* pygame  
  Основная библиотека для реализации графического интерфейса игры. Она используется для создания окна игры, рендеринга шахматной доски и фигур, обработки событий (щелчков мыши, нажатий клавиш) и управления звуковыми эффектами. Без нее невозможно было бы визуализировать игровой процесс.
* tkinter  
  Предназначена для построения простого и удобного интерфейса (GUI) за пределами игрового окна. Используется для диалоговых окон авторизации, выбора противника и других элементов взаимодействия с пользователем. Это решение дополняет pygame, которая фокусируется на графике.
* threading  
  Применяется для выполнения фоновых задач, таких как расчеты ходов бота. Это позволяет избежать зависания игрового интерфейса при выполнении сложных вычислений.
* json  
  Используется для сохранения и загрузки настроек игры, включая данные учетных записей и конфигурацию доски. JSON формат удобен для хранения структурированных данных.
* pathlib.Path  
  Библиотека для работы с файловыми путями. Использование Path обеспечивает кроссплатформенность, позволяя программе корректно работать на Windows, macOS и Linux.
* bcrypt  
  Библиотека для безопасного хранения паролей. Хэширование паролей делает их защиту устойчивой к атакам.
* pygame\_widgets  
  Дополняет pygame функционалом для создания виджетов, таких как кнопки и слайдеры. Это упрощает создание интерактивных элементов управления в игровом интерфейсе.
* Пользовательские модули (chessboard, config, resloader)
* chessboard: Основной модуль для работы с игровой логикой шахмат. Содержит классы для шахматной доски и клеток, а также методы проверки правил игры.
* config: Управляет конфигурацией игры, включая стартовые позиции, цвет игрока, уровень сложности бота.
* resloader: Модуль для управления ресурсами игры. Загрузка и кеширование изображений, шрифтов и звуков происходит централизованно, что оптимизирует производительность.
* queue  
  Необходима для передачи данных между потоками (например, передача результатов расчета хода от бота основному потоку игры).

**5. Вычислительная модель (логика ходов)**

**Проектное решение:**

**Вычислительная модель программы отвечает за игровую механику и реализацию правил шахмат:**

* Классы фигур: Каждая шахматная фигура (например, `Bishop`, `King`, `Rook`) представлена отдельным классом, который наследуется от базового класса `Figure`. Эти классы содержат методы для определения всех возможных ходов:
* Метод `get\_possible\_moves()` определяет потенциальные ходы фигуры, учитывая ограничения на перемещение по доске (например, движения по диагонали у слона или по прямой у ладьи).
* Метод `get\_valid\_moves()` фильтрует ходы, оставляя только те, которые не ставят короля под шах, используя метод `is\_in\_check()` для проверки угрозы.
* Обработка игровых ходов:
* Метод `on\_click()` класса `Board` обрабатывает нажатия пользователя и проверяет, к какой клетке была применена команда. В зависимости от состояния игры метод инициирует перемещение фигуры, подсветку клеток или завершение хода.
* Метод `change\_side()` изменяет текущий ход игрока, обновляет историю ходов и очищает подсветку выделенных клеток.
* Проверка состояния игры:
* Метод `is\_in\_check()` анализирует, находится ли король под атакой, проверяя все возможные ходы фигур противника.
* Метод `is\_in\_checkmate()` определяет, есть ли у текущего игрока доступные ходы для выхода из шаха. Если таких ходов нет, объявляется мат.
* Метод `game\_over()` устанавливает результат игры в случае мата, пата или окончания времени.
* Алгоритм бота:
* Для выполнения ходов противника используется метод `bot\_move()`, который создает поток для выполнения вычислений с использованием класса `threading.Thread`. В этом потоке запускается метод `getBestMove()`, который рассчитывает наилучший ход, используя алгоритм минимакса или его вариации с отсечениями.

**Оценка сложности вычислительной части:**

* Определение ходов:
* Основная сложность возникает при расчете возможных ходов каждой фигуры. Например, для слона (`Bishop`) это может включать до 7 клеток в четырех направлениях, что дает сложность порядка O(N2), где N — размер доски (8 для стандартных шахмат).
* Проверка атакующих ходов всех фигур при оценке шаха также выполняется в диапазоне от O(N2) до O(N3).
* Алгоритмы бота:
* Если используется алгоритм минимакса с альфа-бета отсечением, сложность его выполнения в среднем O(BM), где B — фактор ветвления (количество возможных ходов на каждом уровне), а M — глубина поиска. В шахматах B может варьироваться от 30 до 40, а M определяется сложностью уровня.

**6. Графическая модель (интерфейс)**

**Проектное решение:**

**Графическая модель программы отвечает за отображение доски, фигур и интерфейса игры:**

* Отображение доски и фигур:
* Класс `Board` рендерит доску и каждую фигуру на экране с помощью метода `draw()`. Фигуры отображаются в зависимости от их текущей позиции, определенной методом `set\_figure()`.
* Класс `Square` отвечает за отображение отдельных клеток и фигуры, находящейся на них. Метод `draw()` визуализирует клетку, применяя подсветку при необходимости (например, при выделении клетки).
* Информационная панель:
* Класс `InfoPanel` отображает дополнительную информацию, включая время для игроков (через класс `Timers`), историю ходов и сообщения об окончании игры.
* Класс `Timers` обновляет таймеры игроков и отображает оставшееся время. Если время истекло, метод `update()` вызывает завершение игры через `board.game\_over()`.
* Меню управления:
* Класс `Menu` добавляет элементы управления, такие как кнопки для начала новой игры и сохранения состояния. Кнопки создаются с помощью библиотеки `pygame\_widgets` и обрабатывают нажатия через метод `draw( )`.

**Оценка сложности графической части:**

* Отрисовка доски и фигур:
* Отображение каждой клетки и фигуры на доске имеет сложность O(N2), так как метод `draw( )` вызывается для всех 64 клеток.
* При взаимодействии пользователя (нажатии) или изменении состояния требуется полная перерисовка доски, что также занимает O(N2).
* Обновление интерфейса и таймеров:
* Операции по обновлению текстовой информации или таймеров имеют сложность O(1) при каждом обновлении, но общая сложность обновления всех компонентов интерфейса при каждом ходе может достигать O(N2), так как доска и панель истории полностью рендерятся заново.

**Итоговая оценка сложности:**

* Вычислительная модель: Основная сложность связана с алгоритмами поиска ходов и их анализа, особенно при работе с ботом, где сложность может достигать O(BM) для алгоритма минимакса, где B — количество возможных ходов на каждом уровне, а M — глубина поиска.
* Графическая модель: Рендеринг интерфейса требует затрат O(N2), но взаимодействие пользователя и обновление отдельных элементов (например, текст и таймеры) происходит с O(1) сложностью.

**Заключение**

Данная курсовая работа представляет собой не только успешную попытку реализации компьютерной шахматной игры, но и продемонстрировала возможности применения современных технологий программирования в решении задач высокой вычислительной сложности.

Проект разработан с использованием языка Python и ряда библиотек, таких как pygame, что обеспечило высокую интерактивность и наглядность игрового интерфейса. Благодаря этому пользователи могут полноценно взаимодействовать с программой, изучая шахматные эндшпили и совершенствуя свои навыки.

Основное внимание было уделено реализации точной логики шахматных правил и созданию вычислительных моделей, которые обеспечивают корректность всех игровых процессов. Каждый элемент, начиная с движения фигур и заканчивая проверкой шаха или мата, разработан с учетом стандартов шахматной теории. Внедрение искусственного интеллекта на основе алгоритма минимакса с функцией альфа-бета-отсечения позволило программе выполнять ходы с учетом стратегических и тактических аспектов игры. Это делает игру против бота не только интересной, но и поучительной для пользователей с разным уровнем подготовки.

Особое внимание было уделено оптимизации производительности программы. Для повышения эффективности работы искусственного интеллекта были использованы структуры данных, такие как деревья поиска, и реализованы эвристические функции оценки позиций, что позволило значительно сократить время расчета ходов и улучшить качество игры бота.

Графический интерфейс был создан для максимального удобства пользователей. Отображение шахматной доски и фигур, подсветка доступных ходов, анимация передвижения фигур и информационная панель способствуют интуитивному взаимодействию с приложением. Для повышения комфорта были добавлены такие элементы управления, как кнопки меню, функция отмены хода, возможность сохранения и загрузки партий с использованием форматов JSON, а также управление настройками игры (уровень сложности бота, цвет фигур и доски).

Работа также демонстрирует высокий уровень интеграции различных библиотек и модулей, обеспечивающих не только функциональность, но и стабильность программы. Это включает в себя:

* использование многопоточности для асинхронной обработки действий бота и обеспечения плавной работы интерфейса;
* поддержку форматов данных JSON для хранения и загрузки игровых настроек и сохранений;
* гибкость в работе с графическими и аудиоресурсами для создания более реалистичной и приятной атмосферы игры.

Кроме того, проект учитывает возможность дальнейшего расширения. Благодаря модульной структуре программы можно легко добавлять:

1. Новые шахматные эндшпили для изучения;
2. Более сложные алгоритмы искусственного интеллекта, такие как нейронные сети или алгоритмы машинного обучения, для повышения уровня игры бота;
3. Поддержку дополнительных режимов игры, таких как многопользовательский режим или игра по сети;
4. Функцию анализа сыгранных партий с выводом статистики и рекомендаций для пользователей.

Таким образом, данная курсовая работа не только решает поставленную задачу, но и является примером практического применения теоретических знаний, полученных в процессе изучения курса «Алгоритмы и структуры данных». Она демонстрирует возможности современных технологий программирования и служит основой для создания более сложных и функциональных шахматных приложений.

Итогом разработки стала интерактивная и обучающая система, способная обучать пользователей шахматным тонкостям и предоставлять возможность изучать сложные игровые ситуации в удобной и визуально привлекательной форме. Этот проект можно рассматривать как успешный пример интеграции теоретических знаний, практических навыков программирования и современных подходов к разработке программного обеспечения.