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»

|  |  |
| --- | --- |
| **Подп.и** |  |
| **Инв** |  |
| **Вза** |  |
| **Подп.и** |  |
| **Инв** |  |

**Компьютерная игра эндшпиль «Король, 2 слона – Король, конь»**

Руководитель разработки:  
Кандидат технических наук,

Доцент кафедры ИВК  
Шишкин Вадим Викторинович

«\_\_» \_\_\_\_\_\_\_\_ 2024 г.

Исполнитель:

студент гр. ИСТбд-23

Сипатов Егор Валерьевич

«\_\_» \_\_\_\_\_\_\_\_ 2024 г.

2024

**Содержание**

Оглавление

[**ТЕХНИЧЕСКОЕ ЗАДАНИЕ** 3](#_Toc193979559)

[**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА** 9](#_Toc193979560)

[**РУКОВОДСТВО ПРОГРАММИСТА** 19](#_Toc193979561)

[**Листинг программы** 26](#_Toc193979562)

[**2. Источники, использованные при разработке** 32](#_Toc193979563)

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»

|  |  |
| --- | --- |
| **Подп.и** |  |
| **Инв** |  |
| **Вза** |  |
| **Подп.и** |  |
| **Инв** |  |

**Компьютерная игра эндшпиль «Король, 2 слона – Король, конь»**

Р.02069337. №23/748-Вариант 21

Листов 6

Исполнитель:

студент гр. ИСТбд-23

Сипатов Егор Валерьевич

«\_\_» \_\_\_\_\_\_\_\_ 2024 г.

2024

**Введение**

**Наименование и условное обозначение приложения:**

*Название:* Chess Endgame

*Условное обозначение:* CE

**Реализуемая игра:**

*Название игры:* Эндшпиль «Король, 2 слона – Король, конь»

**Краткий свод правил игры:**

Эндшпиль «Король, 2 слона – Король, конь» является одной из классических позиций в шахматах, где у одной стороны остаются король и 2 слона, а у второй стороны король и конь. Цель игрока состоит в том, чтобы благодаря двум слонам защитить короля, найти для него наиболее выгодное положение и поставить мат сопернику.

1. **Основание для разработки**

Основание: учебный план направления 09.03.02 "Информационные системы и технологии" и распоряжение по факультету.

1. **Требования к программе или программному изделию**
   1. Функциональное назначение

**Функциональное назначение приложения:**

Приложение Chess Endgame (CE) предназначено для автоматизации процесса обучения и тренировки шахматного эндшпиля «Король, 2 слона – Король, конь». Оно позволяет пользователям рассматривать различные комбинации, возможные при данном эндшпиле.

**Перечень автоматизируемых процессов:**

1 Определение состояния игры:

Проверка на шах и мат/пат: Функция CheckCheckMate автоматизирует проверку на наличие шаха, мата и пата.

2 Определение возможных ходов:

Создание списка Variants должно быть автоматизировано. Программа должна самостоятельно определять все возможные легальные ходы для текущего игрока, учитывая положение фигур и правила игры.

* 1. Требования к функциональным характеристикам
     1. Требования к структуре приложения

1. Модульная структура

* Модули должны быть независимыми и взаимозаменяемыми. Каждый модуль должен выполнять одну конкретную задачу и взаимодействовать с другими модулями через четко определенные интерфейсы.
* Минимальная зависимость между модулями. Минимизировать количество зависимостей между модулями, чтобы изменения в одном модуле минимально влияли на другие.
* Разделение ответственности. Разделять функционал на отдельные модули, чтобы каждый модуль отвечал за строго определённую часть приложения.

1. Основные модули

* Логика игры (Game Logic):

Содержит классы и методы, отвечающие за управление состоянием игры, проверку допустимости ходов, реализацию правил шахматного эндшпиля. Должен включать в себя классы для представления шахматной доски, фигур, а также алгоритмы для генерации и оценки ходов.

* Пользовательский интерфейс (UI):

Отвечает за отображение игровой доски, фигур и других элементов интерфейса. Управляет взаимодействием с пользователем, включая обработку кликов мыши и нажатий клавиш.

* Авторизация и шифрование:

Реализует механизмы авторизации пользователя и шифрования данных.

Предоставляет API для проверки подлинности пользователя и защиты конфиденциальной информации.

* Анализ партий (Analysis):

Может включать в себя базу данных для хранения информации о сыгранных партиях.

* 1. Требования к надежности

Приложение должно стабильно функционировать при нормальном режиме эксплуатации, избегая неожиданных сбоев и аварийных завершений. Перезагрузка - как способ восстановления.

* 1. Требования к информационной и программной совместимости

1. *Операционная система:*

* Windows 10 или выше
* macOS Catalina (10.15) или выше
* Ubuntu 20.04 LTS или выше

1. *Библиотеки:*

* pygame 2.0+
* math (встроена в стандартную библиотеку Python)
* bcrypt 3.2.2+
* sqlite3 (встроена в стандартную библиотеку Python)
* os (встроена в стандартную библиотеку Python)
* json (встроена в стандартную библиотеку Python)
* datetime (встроена в стандартную библиотеку Python)
* copy (встроена в стандартную библиотеку Python)
* sys (встроена в стандартную библиотеку Python)

1. *База данных:*

* SQLite 3.33+

1. *Язык программирования:*

* Python 3.8+

1. *Среда разработки:*

* PyCharm Community Edition (2021.1 или выше)
  1. Требования к маркировке и упаковке

Определяются заданием на курсовую работу.

* 1. Требования к транспортированию и хранению
     1. Условия транспортирования

Требования к условиям транспортирования не предъявляются.

* + 1. Условия хранения

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

* + 1. Сроки хранения

Срок хранения – до окончания срока учебы.

1. **Требования к программной документации**

Определяются заданием на курсовую работу.

1. **Стадии и этапы разработки**

Определяются заданием на курсовую работу.

1. **Порядок контроля и приемки**

Определяются заданием на курсовую работу.

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»

|  |  |
| --- | --- |
| **Подп.и** |  |
| **Инв** |  |
| **Вза** |  |
| **Подп.и** |  |
| **Инв** |  |

**Компьютерная игра эндшпиль «Король, 2 слона – Король, конь»**

Р.02069337. №23-748 Вариант 21

Листов 11

Исполнитель:

студент гр. ИСТбд-23

Сипатов Егор Валерьевич

«\_\_» \_\_\_\_\_\_\_\_ 2024 г.

2024

**Введение**

В данной курсовой работе разрабатывается приложение "Компьютерный эндшпиль: Король, пешка - Король, пешка". Программа представляет собой реализацию шахматной игры с графическим интерфейсом, разработанную на языке Python с использованием библиотеки **pygame**. Программа позволяет игроку рассмотреть все возможные варианты данного эндшпиля.

Реализованное приложение включает в себя графический интерфейс. Приложение разработано на языке Python с использованием библиотеки Pygame для графического интерфейса,.

**1. Проектная часть**

**1.1. Постановка задачи на разработку приложения**

Задача на разработку приложения определяется заданием на курсовую работу. Детализируется в разработанном техническом задании. Приложение должно реализовать игру в шахматный эндшпиль с участием короля, двух слонов и коня.

**1.2. Математические методы**

Для реализации искусственного интеллекта используется алгоритм Minimax с альфа-бета отсечением. Этот алгоритм позволяет оценить наилучший ход для искусственного интеллекта на основе оценки текущего состояния игры. Оценка состояния игры производится на основе разницы в количестве фигур на доске.

1. Оценка состояния игры:

Для оценки текущего состояния игры используется простая функция, которая учитывает разницу в количестве фигур и их ценность. Каждая фигура имеет определённую ценность:

* Король: 1000
* Ферзь: 9
* Ладья: 5
* Слон: 3
* Конь: 3
* Пешка: 1.

Это позволяет алгоритму принимать решения на основе текущей позиции.

Выбор данных методов обоснован их эффективностью в решении задач анализа игровых ситуаций, а также их широкой распространённостью в шахматных программах.

**1.3. Архитектура и алгоритмы**

1.3.1 Архитектура

Приложение состоит из нескольких модулей, каждый из которых отвечает за определённую функциональность:

1. auth.py: Отвечает за аутентификацию пользователей (регистрация и вход). Использует библиотеку bcrypt для хеширования паролей.

* Функции модуля:
* register (username, password): Регистрация нового пользователя.
* Проверяет длину имени пользователя и пароля.
* Хеширует пароль с использованием bcrypt.
* Сохраняет пользователя в базе данных.
* login (username, password): Вход пользователя в систему.
* Проверяет наличие пользователя в базе данных.
* Сравнивает хешированный пароль с введённым паролем.
* Возвращает результат входа.

1. game.py: Реализует логику игры, включая управление доской, ходы фигур и проверку состояния игры.

* Класс Move:
* Свойства:
* start\_row, start\_col: Начальная позиция хода.
* end\_row, end\_col: Конечная позиция хода.
* piece\_moved: Фигура, которая совершает ход.
* piece\_captured: Фигура, которая была захвачена.
* is\_pawn\_promotion: Флаг, указывающий на превращение пешки.
* promotion\_choice: Выбор фигуры для превращения пешки.
* Методы:
* \_\_eq\_\_ (self, other): Сравнение двух ходов.
* get\_chess\_notation (self): Возвращает шахматную нотацию хода.
* to\_dict (self): Сериализация хода в словарь.
* from\_dict (cls, move\_dict): Десериализация хода из словаря.
* Класс Game:
* Свойства:
* white\_player, black\_player: Имена игроков.
* game\_id: Идентификатор игры.
* board: Текущее состояние доски.
* white\_to\_move: Флаг, указывающий, чей ход (белых или чёрных).
* move\_log: История ходов.
* selected\_square: Выбранная клетка.
* valid\_moves: Допустимые ходы.
* checkmate, stalemate: Флаги, указывающие на мат или пат.
* en\_passant\_possible: Возможность взятия на проходе.
* promotion\_choice: Выбор фигуры для превращения пешки.
* start\_time, end\_time: Время начала и окончания игры.
* result: Результат игры.
* Методы:
* \_\_init\_\_ (self, white\_player, black\_player, game\_id): Инициализация игры.
* create\_initial\_board (self): Создание начальной доски.
* load\_game (self, game\_id): Загрузка игры из базы данных.
* reconstruct\_board (self): Восстановление доски из истории ходов.
* make\_move (self, move, update\_state): Выполнение хода.
* undo\_move (self): Отмена последнего хода.
* get\_valid\_moves (self): Получение всех допустимых ходов.
* get\_all\_possible\_moves (self): Получение всех возможных ходов.
* in\_check (self, white): Проверка, находится ли король под шахом.
* check\_game\_state (self): Проверка состояния игры (шах, мат, пат).
* save\_current\_game (self): Сохранение текущего состояния игры.
* save\_game\_completion (self): Сохранение завершённой игры.
* export\_pgn (self): Экспорт игры в формате PGN.
* draw (self, win, images, selected\_square, valid\_moves): Отрисовка игры.
* draw\_board (self, win, selected\_square, valid\_moves): Отрисовка доски.
* draw\_pieces (self, win, images): Отрисовка фигур.
* draw\_game\_state (self, win): Отрисовка состояния игры.
* is\_move\_valid (self, move): Проверка, является ли ход допустимым.
* get\_piece\_moves (self, r, c): Получение всех ходов для фигуры на заданной позиции.

1. main.py: Главный модуль, который управляет запуском приложения и взаимодействием с пользователем.

* Функции модуля:
* auth\_screen (): Экран аутентификации.
* login\_prompt (): Ввод данных для входа.
* register\_prompt (): Ввод данных для регистрации.
* select\_mode (username): Выбор режима игры.
* view\_games (username): Просмотр списка игр.
* resume\_game (game): Возобновление игры.
* game\_screen (mode, white\_player, black\_player): Экран игры.
* main (): Главная функция, запускающая приложение.

1. settings.py: Хранит основные настройки приложения, такие как размеры окна, цвета и путь к ресурсам.

**1.4. Тестирование**

**Mind map**

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

Чек лист

1. Основные компоненты

* Инициализация игры
  + pygame.init()
  + wind = display.set\_mode((640, 640))
  + display.set\_caption('Chess')
  + clock = time.Clock()
  + font.init()
* Доска и фигуры
  + RectList (генерация клеток доски)
  + Board (матрица 8x8 для хранения фигур)
  + AttackDict (правила движения фигур)
* Функции
  + Генерация начальной позиции: MakeBoard()
  + Отрисовка:
    - Фона: DrawBg()
    - Фигур: DrawPieces()
  + Проверка шаха: CheckShah(B\_W)
  + Получение допустимых ходов: ShowVariants(x, y)
  + Проверка мата/пата: CheckCheckMate(B\_W)
* Игровой цикл
  + Обработка событий: event.get()

2. Основные функции

* MakeBoard()
  + Размещение фигур на доске:
    - Ладьи (B1, B1)
    - Короли (K1, K0)
    - Конь (H0)
  + Учет правил размещения (например, король не может быть под шахом).
* DrawBg()
  + Отрисовка фона доски с чередующимися цветами.
* DrawPieces()
  + Загрузка изображений фигур (например, K1.png, K0.png) и их отрисовка на доске.
* CheckShah(B\_W)
  + Проверка, находится ли король под шахом.
  + Анализ всех возможных атак противника.
* ShowVariants(x, y)
  + Вычисление допустимых ходов для выбранной фигуры.
  + Удаление ходов, которые ставят короля под шах.
* CheckCheckMate(B\_W)
  + Проверка мата или пата:
    - Если ни одна фигура не может сделать ход:
      * Мат → король под шахом.
      * Пат → король не под шахом.

1.4.1 Описание отчета о тестировании

Отчет о тестировании предназначен для проверки корректности работы приложения и его соответствия требованиям технического задания.

* + 1. Цель тестирования

1. Проверка основных функций

* Убедиться, что доска (**Board**) и фигуры отображаются правильно.
* Проверить, что фигуры перемещаются согласно правилам шахмат.
* Убедиться, что рокировка, взятие фигур и другие специальные ходы работают корректно.

2. Проверка начальной расстановки фигур

* Убедиться, что функция **MakeBoard** корректно размещает фигуры на доске:
  + Фигуры не пересекаются.
  + Король не находится под шахом в начальной позиции.
  + Все фигуры находятся в пределах доски.

3. Логика игры (game.py)

Цель: Проверить корректность работы логики игры.

Ожидаемый результат:

* При запуске новой игры начальная доска должна быть корректно инициализирована.
* Пешка не должна ходить назад или на занятую клетку, выводится ошибка.
* При установке позиции мата игра должна завершаться с сообщением "Мат".

4. Производительность

* Проверить, что программа работает плавно и без задержек:
  + Обновление доски (**display.update()**).
  + Ограничение FPS (**clock.tick(60)**).

5. Основной интерфейс (main.py)

Цель: Проверить корректность работы основного интерфейса.

Ожидаемый результат:

* Программа должна запускаться корректно.

6. Настройки (settings.py)

Цель: Проверить корректность работы настроек программы.

Ожидаемый результат:

* Цвета и размеры доски должны соответствовать заданным настройкам.
* Программа должна работать в полноэкранном режиме.

1.4.5 Выводы

Тестирование показало, что приложение работает корректно и соответствует требованиям технического задания.

Рекомендации по улучшению включают:

* Добавление дополнительных тестов для проверки граничных условий.
* Улучшение алгоритма оценки состояния игры для повышения качества игры.

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»

|  |  |
| --- | --- |
| **Подп.и** |  |
| **Инв** |  |
| **Вза** |  |
| **Подп.и** |  |
| **Инв** |  |

**Компьютерная игра эндшпиль «Король, 2 слона – Король, конь»**

Р.02069337. №23/748-Вариант 21

Листов 6

Исполнитель:

студент гр. ИСТбд-23

Сипатов Егор Валерьевич

«\_\_» \_\_\_\_\_\_\_\_ 2024 г.

2024

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Приложение "Компьютерный эндшпиль: Король, 2 слона - Король, конь" предназначено для реализации шахматного эндшпиля с использованием фигур "король", "конь" и "слон". Основные функции приложения включают:

* Оценка состояния игры: Приложение использует простую функцию оценки состояния игры на основе разницы в количестве фигур.
* Графический интерфейс: Приложение предоставляет графический интерфейс для удобства взаимодействия с пользователем.
  1. **Условия, необходимые для использования приложения**

Для использования приложения необходимы следующие условия:

1. Операционная система: Windows, macOS или Linux.
2. Интерпретатор Python: Версия 3.7 или выше.
3. Библиотеки Python:

* Pygame (для графического интерфейса).

1. Требования к оборудованию:

* Процессор: x86 или x64.
* ОЗУ: 2 ГБ или больше.
* Графика: Поддержка OpenGL или DirectX.

**2. Характеристики программы**

**2.1 Характеристики приложения**

Количество строк кода: 357 строк.

Количество структур данных: 2 (класс Game и класс Move).

Количество алгоритмов: 2 (алгоритм минимакс с альфа-бета отсечениями и алгоритм оценки состояния игры).

Используемые библиотеки:

* Pygame: для создания графического интерфейса.
* Math: для математических операций.

Работа приложения:

* Приложение запускается с помощью файла main.py.
* Во время игры пользователь взаимодействует с графическим интерфейсом.

1. Экран игры: Графическое представление доски с фигурами

**Изображение выглядит как Игры, Игры и спорт в закрытом помещении, настольная игра, Настольная игра

Контент, сгенерированный ИИ, может содержать ошибки.**

**2.2 Особенности реализации приложения**

Структуры данных:

1. Класс Game:

* Представляет состояние игры, включая доску, ходы и логику.

Альтернативные варианты:

* Вместо класса Game можно было использовать простую структуру данных, но класс позволяет инкапсулировать логику игры и упрощает управление состоянием.

**3. Обращение к программе**.

**1. Методы и алгоритмы модуля game.py**

3.1 Game.\_\_init\_\_(self, white\_player, black\_player, game\_id)

* Описание: Инициализация игры.
* Параметры:
* white\_player (str): Имя игрока за белых.
* black\_player (str): Имя игрока за чёрных.
* game\_id (int): Идентификатор игры.
* Алгоритм:
* Инициализирует начальное состояние игры.
* Отменяет последний ход на доске.
* Удаляет ход из истории.
* Обновляет состояние игры.
  1. Game.get\_valid\_moves (self)
* Описание: Получение всех допустимых ходов.
* Возвращаемое значение: Список допустимых ходов.
* Алгоритм:
* Генерирует все возможные ходы.
* Фильтрует ходы, чтобы исключить ходы, приводящие к шаху.
  1. Game.get\_all\_possible\_moves (self)
* Описание: Получение всех возможных ходов.
* Возвращаемое значение: Список возможных ходов.
* Алгоритм:
* Генерирует все возможные ходы для текущего игрока.
  1. Game.in\_check (self, white)
* Описание: Проверка, находится ли король под шахом.
* Параметры:
* white (bool): Флаг, указывающий, за какого игрока проверяется шах.
* Возвращаемое значение: True, если король под шахом, иначе False.
* Алгоритм:
* Находит позицию короля.
* Проверяет, атакуется ли король фигурами противника.
  1. Game.check\_game\_state (self)
* Описание: Проверка состояния игры (шах, мат, пат).
* Алгоритм:
* Проверяет, находится ли король под шахом.
* Если ходов нет, проверяет, является ли ситуация матом или патом.
  1. Game.draw (self, win, images, selected\_square, valid\_moves)
* Описание: Отрисовка игры.
* Параметры:
* win (pygame.Surface): Поверхность для отрисовки.
* images (dict): Словарь с изображениями фигур.
* Алгоритм:
* Отрисовывает доску, фигуры и состояние игры.
  1. Game.draw\_board (self, win, selected\_square, valid\_moves)
* Описание: Отрисовка доски.
* Параметры:
* win (pygame.Surface): Поверхность для отрисовки.
* Алгоритм:
* Отрисовывает клетки доски.
  1. Game.draw\_pieces (self, win, images)
* Описание: Отрисовка фигур.
* Параметры:
* win (pygame.Surface): Поверхность для отрисовки.
* images (dict): Словарь с изображениями фигур.
* Алгоритм:
* Отрисовывает фигуры на доске.
  1. Game.draw\_game\_state (self, win)
* Описание: Отрисовка состояния игры.
* Параметры:
* win (pygame.Surface): Поверхность для отрисовки.
* Алгоритм:
* Отрисовывает сообщения о шахе, мате или пате.

**4. Описания библиотек**

4.1 pygame

* Описание: Библиотека для создания игр и мультимедийных приложений.
* Функции:
* pygame.init (): Инициализация библиотеки.
* pygame.display.set\_mode (): Создание окна.
* pygame.image.load (): Загрузка изображений.
* pygame.transform.scale (): Масштабирование изображений.
* pygame.draw.rect (): Отрисовка прямоугольников.
* pygame.draw.circle (): Отрисовка кругов.
* pygame.font.SysFont (): Создание шрифта.
* pygame.event.get (): Получение событий.
* Применение: Используется для создания графического интерфейса игры.

4.1 os

* Описание: Стандартная библиотека Python для работы с операционной системой.
* Функции:
* os.path.join(): Объединение путей.
* Применение: Используется для работы с путями к файлам.

# **Листинг программы**

import random  
  
import pygame  
from pygame import \*  
import pygame as pg  
import math  
  
wind = display.set\_mode((640, 640))  
display.set\_caption('Chess')  
clock = time.Clock()  
font.init()  
  
RectList = []  
for i in range(8):  
 for n in range(4):  
 RectList.append(pygame.Rect((n \* 160 + (i % 2) \* 80, i \* 80, 80, 80)))  
  
Board = [  
 ['.', '.', '.', '.', '.', '.', '.', '.'],  
 ['.', '.', '.', '.', '.', '.', '.', '.'],  
 ['.', '.', '.', '.', '.', '.', '.', '.'],  
 ['.', '.', '.', '.', '.', '.', '.', '.'],  
 ['.', '.', '.', '.', '.', '.', '.', '.'],  
 ['.', '.', '.', '.', '.', '.', '.', '.'],  
 ['.', '.', '.', '.', '.', '.', '.', '.'],  
 ['.', '.', '.', '.', '.', '.', '.', '.']]  
  
AttackDict = {'B': [[1, 1], [-1, -1], [1, -1], [-1, 1], 1],  
 'H': [[1, 2], [2, 1], [-1, -2], [-2, -1], [-1, 2], [-2, 1], [1, -2], [2, -1], 0],  
 'P': [[-1, -1], [1, -1], 0],  
 'p': [[-1, 1], [1, 1], 0],  
 'K': [[1, 1], [-1, -1], [1, -1], [-1, 1], [0, 1], [1, 0], [0, -1], [-1, 0], 0]  
 }  
  
  
def MakeBoard():  
 global Board  
  
 startboard = []  
 for i in range(8):  
 startboard.append([])  
 for j in range(8):  
 startboard[i].append(j)  
  
 p1 = random.randint(0, 7)  
 p2 = random.randint(0, 3) \* 2  
 if (p1 % 2 != 0): p2 += 1  
 b1 = [p1, p2]  
 startboard[p1].pop(p2)  
  
 p3 = random.randint(0, 7)  
 p4 = random.randint(0, 3) \* 2  
 if (p3 % 2 == 0): p4 += 1  
 b2 = [p3, p4]  
  
 startboard[p3].remove(p4)  
  
 K11 = random.randint(0, len(startboard) - 1)  
 K12 = startboard[K11][random.randint(0, len(startboard[K11]) - 1)]  
 k1 = [K11, K12]  
  
 startboard[K11].remove(K12)  
  
 for i in range(8):  
 if not (0 <= i - p1 + p2 <= 7): continue  
 try:  
 startboard[i].remove(i - p1 + p2)  
 except:  
 0  
 for i in range(8):  
 if not (0 <= -i + p1 + p2 <= 7): continue  
 try:  
 startboard[i].remove(-i + p1 + p2)  
 except:  
 0  
 for i in range(-1, 2):  
 if not (0 >= p1 + i >= 7): continue  
 for j in range(-1, 2):  
 if (i == 0) and (j == 0): continue  
 if not (0 >= p2 + j >= 7): continue  
 try:  
 startboard[p1 + i].remove(p2 + j)  
 except:  
 0  
 for i in range(8):  
 if not (0 <= i - p3 + p4 <= 7): continue  
 try:  
 startboard[i].remove(i - p3 + p4)  
 except:  
 0  
 for i in range(8):  
 if not (0 <= -i + p3 + p4 <= 7): continue  
 try:  
 startboard[i].remove(-i + p3 + p4)  
 except:  
 0  
 for i in range(-1, 2):  
 if not (0 >= p3 + i >= 7): continue  
 for j in range(-1, 2):  
 if (i == 0) and (j == 0): continue  
 if not (0 >= p4 + j >= 7): continue  
 try:  
 startboard[p3 + i].remove(p4 + j)  
 except:  
 0  
 for i in range(-1, 2):  
 if not (0 >= K11 + i >= 7): continue  
 for j in range(-1, 2):  
 if (i == 0) and (j == 0): continue  
 if not (0 >= K12 + j >= 7): continue  
 try:  
 startboard[K11 + i].remove(K12 + j)  
 except:  
 0  
  
 K21 = random.randint(0, len(startboard) - 1)  
 K22 = startboard[K21][random.randint(0, len(startboard[K11]) - 1)]  
 k2 = [K21, K22]  
 startboard[K21].remove(K22)  
  
 for i in [-2, -1, 1, 2]:  
 if not (0 >= K11 + i >= 7): continue  
 for j in [-2, -1, 1, 2]:  
 if (abs(i) == abs(j)): continue  
 if not (0 >= K12 + j >= 7): continue  
 try:  
 startboard[K11 + i].remove(K12 + j)  
 except:  
 0  
 for i in [-2, -1, 1, 2]:  
 if not (0 >= p1 + i >= 7): continue  
 for j in [-2, -1, 1, 2]:  
 if (abs(i) == abs(j)): continue  
 if not (0 >= p2 + j >= 7): continue  
 try:  
 startboard[p1 + i].remove(p2 + j)  
 except:  
 0  
 for i in [-2, -1, 1, 2]:  
 if not (0 >= p3 + i >= 7): continue  
 for j in [-2, -1, 1, 2]:  
 if (abs(i) == abs(j)): continue  
 if not (0 >= p3 + j >= 7): continue  
 try:  
 startboard[p3 + i].remove(p4 + j)  
 except:  
 0  
  
 h1 = random.randint(0, len(startboard) - 1)  
 h2 = startboard[h1][random.randint(0, len(startboard[h1]) - 1)]  
 h = [h1, h2]  
  
 Board[b1[0]][b1[1]] = 'B1'  
 Board[b2[0]][b2[1]] = 'B1'  
 Board[k1[0]][k1[1]] = 'K1'  
 Board[k2[0]][k2[1]] = 'K0'  
 Board[h[0]][h[1]] = 'H0'  
  
  
def DrawBg():  
 pygame.draw.rect(wind, (181, 136, 99), (0, 0, 640, 640))  
 for R in RectList:  
 pygame.draw.rect(wind, ((240, 217, 181)), R)  
  
  
def DrawPieces():  
 y = 0  
 for Brd in Board:  
 x = 0  
 for B in Brd:  
 if Board[y][x] != '.':  
 wind.blit(transform.scale(pygame.image.load(Board[y][x] + '.png'), (70, 70)), (5 + x \* 80, 5 + y \* 80))  
 x += 1  
 y += 1  
  
  
def CheckShah(B\_W): # аргумент B\_W принимает значение 0 или 1. 0-Если интересует шах белого короля, 1-если черного  
 y = 0  
 for Brd in Board: # проверка каждой строки  
 x = 0  
 for B in Brd: # проверка каждой клетки строки  
 if B != '.': # если клетка не пуста  
 if B[  
 1] != B\_W: # если найденая фигура противоположного цвета с проверяемым королём и соответственно может его атаковать  
  
 for shift in AttackDict[B[0]][0:-1]: # shift-направление аттаки, числа показывающие сдвиг по X и Y  
 pos = [x, y] # позиция найденной фигуры  
 for i in range(AttackDict[B[0]][  
 -1] \* 6 + 1): # если аттака во всё поле, то цикл повториться 7 раз, иначе-1 раз.  
 pos[0] += shift[0]  
 pos[1] += shift[1] # сместим расматриваемую позицию в соответствии с shift  
 if pos[0] > 7 or pos[0] < 0 or pos[1] > 7 or pos[  
 1] < 0: break # если X или Y рассматриваемой позиции выходит за пределы поля, то остановить проверку этого направления атаки  
 if Board[pos[1]][pos[0]] != '.':  
 if Board[pos[1]][pos[0]] != 'K' + B\_W:  
 break # если поле не пустое и на нём не стоит вражеский король, то остановить проверку этого направления атаки  
 else:  
 return True # если король в клетке всё же есть-вернуть True. Король действительно под шахом  
 x += 1  
 y += 1  
 return False # если шах так и не был обнаружен, вернуть False  
  
  
def ShowVariants(x, y): # x,y-координаты фигуры для которой нужно определить ходы  
 global Variants  
 Variants = [] # список вариантов ходов  
 B = Board[y][x] # B-фигура для которой нужно определить ходы  
 for shift in AttackDict[B[0]][0:-1]: # уже знакомый shift-сдвиг  
 pos = [x, y] # а также знакомая позиция фигуры-pos  
 for i in range(  
 AttackDict[B[0]][-1] \* 6 + 1): # если аттака во всё поле, то цикл повториться 7 раз, иначе-1 раз.  
 pos[0] += shift[0]  
 pos[1] += shift[1] # опять смещаем позицию с помощью shift  
 if pos[0] > 7 or pos[0] < 0 or pos[1] > 7 or pos[  
 1] < 0: break # если X или Y рассматриваемой позиции выходит за пределы поля, то остановить проверку этого направления  
 if Board[pos[1]][pos[0]] != '.': # если клетка не пуста  
 if Board[pos[1]][pos[0]][1] != Board[y][x][1]:  
 Variants.append([pos[0], pos[  
 1]]); break # если клетку занимает вражеская фигура то добавить её как вариант хода и остановить проверку этого направления  
 else:  
 break # если же клетку заняла дружеская фигура, то остановить эту линию ходов  
 elif B[0] != 'p' and B[  
 0] != 'P': # если клетка пуста, а рассматриваемая фигура не пешка то добавить её как вариант хода. (Пешка не может атаковать пустую клетку)  
 Variants.append([pos[0], pos[1]])  
 # Теперь дело за малым-откинуть все ходы, которые ставят своего короля под шах  
  
 ForDeletion = [] # список вариантов на удаление  
 Board[y][x] = '.' # временно уберем рассматриваемую фигуру со стола  
 for V in Variants: # переберем все варианты  
 remember = Board[V[1]][V[0]] # запоминаем клетку, на которую сейчас поставим фигуру  
 Board[V[1]][V[0]] = B # ставим фигуру на это место  
 if CheckShah(B[1]): ForDeletion.append(V) # если король под шахом-добавим этот вариант в список на удаление  
 Board[V[1]][V[0]] = remember # возвращаем клетку которую запомнили  
 Board[y][x] = B # вернём рассматриваемую фигуру на стол  
 for Del in ForDeletion: # удалим все недопустимые варианты  
 Variants.remove(Del)  
  
 if Board[y][x] == 'K0': # если рассматриваем ходы для белого короля  
 global castlingL0, castlingR0  
 if Board[7][0:5] == ['R0', '.', '.', '.',  
 'K0'] and castlingL0: # если между левой ладьёй и королём пусто, а рокировка с левой ладьёй не запрещена  
 Board[7][2], Board[7][3] = 'K0', 'K0' # временно поставим два короля в клетки через которые пройдёт король  
 if CheckShah(  
 '0') == 0: # если эти короли не получают шаха, то это значит, что все условия для рокировки есть и можно добавлять ход-рокировку  
 Variants.append([2, 7])  
 Board[7][2], Board[7][3] = '.', '.' # уберём временных королей  
  
 if Board[7][4:8] == ['K0', '.', '.', 'R0'] and castlingR0: # все тоже самое для рокировки с правой ладьёй  
 Board[7][5], Board[7][6] = 'K0', 'K0'  
 if CheckShah('0') == 0:  
 Variants.append([6, 7])  
 Board[7][5], Board[7][6] = '.', '.'  
 if Board[y][x] == 'K1':  
 global castlingL1, castlingR1  
 if Board[0][0:5] == ['R1', '.', '.', '.', 'K1'] and castlingL1:  
 Board[0][2], Board[0][3] = 'K1', 'K1'  
 if CheckShah('1') == 0:  
 Variants.append([2, 0])  
 Board[0][2], Board[0][3] = '.', '.'  
  
 if Board[0][4:8] == ['K1', '.', '.', 'R1'] and castlingR1:  
 Board[0][5], Board[0][6] = 'K1', 'K1'  
 if CheckShah('1') == 0:  
 Variants.append([6, 0])  
 Board[0][5], Board[0][6] = '.', '.'  
  
  
def CheckCheckMate(B\_W): # аргумент B\_W-как обычно, 0-интерисует мат/пат белых, 1-черных  
 global Variants  
 y = 0  
 for Brd in Board: # проверка каждой строки  
 x = 0  
 for B in Brd: # проверка каждого элемента строки  
 if B[  
 -1] == B\_W: # если найдена фигура нужного цвета то проверить, есть ли для неё хоть один вариант хода. Если да-вернуть 0-мата/пата нет  
 ShowVariants(x, y)  
 if len(Variants) > 0: Variants = [];return 0  
 x += 1  
 y += 1  
 # если дошли до этой строки, то это значит, что ни одна фигура нужного цвета не может сделать ход. Это значит что поставлен мат или пат  
 if CheckShah(B\_W):  
 Variants = [];return 1 # король под шахом-значит мат, возвращаем 1  
 else:  
 Variants = [];return 2 # король не под шахом-пат возвращаем 2  
 # обратите внимание что перед тем как вернуть значение, необходимо очистить список Variants, чтобы избежать багов  
  
  
Variants = []  
MakeBoard()  
DrawBg()  
DrawPieces()  
Turn = 0  
game = 1  
check = 0  
castlingL0, castlingR0 = True, True  
castlingL1, castlingR1 = True, True  
  
while game:  
 for e in event.get():  
 if e.type == QUIT:  
 game = 0  
  
 if e.type == pg.MOUSEBUTTONDOWN and e.button == 1: # если нажата ЛКМ  
 x, y = (e.pos) # x,y-положение мыши  
 x, y = math.floor(x / 80), math.floor(y / 80) # поделив x,y на 80 получаем клетку, на которую нажал игрок  
 if Board[y][x] != '.': # если она не пуста  
 if Board[y][x][1] == str(Turn): # и равна переменной Turn-очередь. Turn меняется каждый ход  
 ShowVariants(x, y) # получаем список доступных ходов  
 remember = [x, y] # запомним клетку на которую нажали  
 for V in Variants:  
 pygame.draw.circle(wind, (200, 200, 200), (V[0] \* 80 + 40, V[1] \* 80 + 40),  
 10) # отрисовка кружочков показывающих, куда можно сходить  
  
 if e.type == pg.MOUSEBUTTONUP and e.button == 1 and Turn != -1 and Turn != -2: # если ОТжата ЛКМ  
 x, y = (e.pos)  
 x, y = math.floor(x / 80), math.floor(y / 80) # получаем клетку в которой находиться мышка  
 if Variants.count([x, y]): # если эта клетка есть в списке возможных ходов  
  
 Board[y][x] = Board[remember[1]][  
 remember[0]] # заменяем выбранную клетку на ту что запомнили при нажатии  
 Board[remember[1]][remember[0]] = '.' # клетку с которой ушли оставляем пустой  
  
 if remember == [4, 7] and Board[y][x] == 'K0':  
 if [x, y] == [2, 7]: Board[7][0] = '.';Board[7][3] = 'R0'  
 if [x, y] == [6, 7]: Board[7][7] = '.';Board[7][5] = 'R0'  
 if remember == [4, 0] and Board[y][x] == 'K1':  
 if [x, y] == [2, 0]: Board[0][0] = '.';Board[0][3] = 'R1'  
 if [x, y] == [6, 0]: Board[0][7] = '.';Board[0][5] = 'R1'  
  
 if Board[7][  
 0] != 'R0': castlingL0 = False # если левая ладья не на месте, запретить делать с ней рокировку  
 if Board[7][  
 7] != 'R0': castlingR0 = False # если правая ладья не на месте, запретить делать с ней рокировку  
 if Board[7][  
 4] != 'K0': castlingL0 = False;castlingR0 = False # если король не на месте, запретить делать рокировку впринципе  
 if Board[0][0] != 'R1': castlingL1 = False  
 if Board[0][7] != 'R1': castlingR1 = False  
 if Board[0][4] != 'K1': castlingL1 = False;castlingR1 = False  
 Turn = 1 - Turn # очередь меняется с 0 на 1 или наоборот  
 # после смены очереди надо проверить наличие мата или пата  
 check = CheckCheckMate(str(Turn)) # check примет 1 если объявлен мат, 2-если пат, 0-в ином случае  
 if check == 1: # если мат  
 DrawBg() # рисуем доску напоследок  
 DrawPieces()  
 if Turn == 0: # и в зависимости от того чья очередь, объявляем победителя  
 wind.blit(pygame.font.SysFont(None, 30).render('BLACK WON', False, (30, 30, 30)), (260, 310))  
 if Turn == 1:  
 wind.blit(pygame.font.SysFont(None, 30).render('WHITE WON', False, (30, 30, 30)), (260, 310))  
 if check == 2: # если пат то объявляем ничью  
 wind.blit(pygame.font.SysFont(None, 30).render('DRAW', False, (30, 30, 30)), (290, 310))  
 Variants = []  
 if check == 0:  
 DrawBg()  
 DrawPieces()  
 Variants = []  
 display.update()  
 clock.tick(60)

# **2. Источники, использованные при разработке**

1. В.В. Шишкин, Д.С. Афонин РАЗРАБОТКА ЛОГИЧЕСКИХ КОМПЬЮТЕРНЫХ ИГР С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ В СРЕДЕ ПИТОН
2. ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения». – М.: Издательство стандартов, 1990.
3. Pygame Documentation. [Электронный ресурс]. - Режим доступа: https://www.pygame.org/docs/
4. Python Documentation. [Электронный ресурс]. - Режим доступа: https://docs.python.org/3/
5. Python 3. Самое необходимое / Н.А. Прохоренок, В.А. Дронов. – 2-е изд.,перераб. и доп. – СПб.: БХВ-Петербург,2020. – 608 с.: ил. – (Самое необходимое)
6. Руководство по Pygame [Электронный ресурс]. – Режим доступа: https://pythonist.ru/pygame-tutorial/
7. Pygame [Электронный ресурс]. – Режим доступа: https://egoroffartem.pythonanywhere.com/course/pygame/
8. Образовательная платформа Stepik [Электронный ресурс]. – Режим доступа: https://stepik.org/learn