Software Design Description (SDD) – Chess Game

**1. Введение**

Цель документа: описать архитектуру и проектные решения, используемые при разработке 2D-шахматной игры с использованием Python и Pygame.

Область применения: Документ предназначен для разработчиков и технических специалистов, занимающихся поддержкой и развитием проекта. Описаны границы системы, включая управление шахматной доской, взаимодействие между фигурами и правила игры.

Ссылки:

<https://github.com/elyakkos/chess_game_techdoc/blob/main/SRS.docx> - SRS-документ

**2. Общее описание системы**

Обзор системы: Игра представляет собой классическую шахматную партию для двух игроков на одном устройстве. Реализованы стандартные шахматные правила, включая рокировку, превращение пешки и выявление шаха и мата.

Архитектурные ограничения:

* Разработка выполнена на Python версии 3.11+ с использованием Pygame.
* Фиксированное разрешение окна игры — 600x600 пикселей.
* Игра предназначена для двух игроков на одном устройстве без сетевой поддержки.

**3. Архитектурный дизайн**

Архитектурный подход: используется объектно-ориентированный подход с чётким разделением логики на классы:

* Board — управление доской.
* Square — представление клетки доски.
* Piece и её наследники — фигуры с индивидуальными правилами движения.

Диаграмма архитектуры: для наглядного представления архитектуры системы используется диаграмма классов, отражающая связи между основными компонентами (Board, Square и Piece). Диаграмма представлена в разделе **12. Приложения**, где также включены дополнительные визуальные материалы.

**4. Детальное описание компонентов**

**4.1. Класс Game**

* Описание: класс Game реализован в модуле main.py и отвечает за управление основным игровым процессом. В его функционал входят запуск игрового цикла, обработка пользовательских событий и взаимодействие с другими компонентами системы.
* Методы:
  + main\_menu() — отображает главное меню с кнопками "Начать игру" и "Выход".
  + game\_loop() — реализует основной игровой цикл, обеспечивая обработку ходов, проверку игровых состояний (шах и мат) и обновление интерфейса.

**4.2. Класс Board**

* Описание: класс Board, определённый в модуле Board.py, управляет состоянием шахматной доски, размещением фигур и реализацией правил игры.
* Атрибуты:
  + width, height — размеры доски.
  + tile\_width, tile\_height — размеры одной клетки.
  + squares — список объектов Square, представляющих клетки доски.
  + turn — текущий ход (белые или чёрные).
  + selected\_piece — фигура, выбранная для выполнения хода.
* Методы:
  + setup\_board() — размещает шахматные фигуры на доске в начальной позиции.
  + handle\_click(mx, my) — обрабатывает действия пользователя, такие как выбор фигуры или выполнение хода.
  + is\_in\_check(color) — проверяет, находится ли король указанного цвета под шахом.
  + is\_in\_checkmate(color) — определяет, поставлен ли мат королю.
  + draw(display) — отвечает за отрисовку шахматной доски и фигур на экране.

**4.3. Класс Square**

* Описание: класс Square, расположенный в модуле Square.py, представляет клетку шахматной доски. Он хранит информацию о координатах, цвете и фигуре, находящейся на клетке.
* Атрибуты:
  + x, y — координаты клетки.
  + color — цвет клетки (светлый или тёмный).
  + occupying\_piece — фигура, находящаяся на клетке (если есть).
  + highlight — логический флаг, указывающий на необходимость подсветки клетки.
* Методы:
  + get\_coord() — возвращает шахматные координаты клетки (например, e4).
  + draw(display) — отрисовывает клетку на экране, включая подсветку и фигуру.

**4.4. Класс Piece и её наследники**

* Описание: класс Piece, расположенный в модуле Piece.py, является базовым для всех шахматных фигур. Его функционал включает управление позицией, правилами движения и взаимодействием с другими фигурами. Наследники класса (Pawn, Rook, Knight, Bishop, Queen, King) реализуют индивидуальные правила движения.
* Атрибуты:
  + pos — текущая позиция фигуры в виде координат (x, y).
  + color — цвет фигуры (white или black).
  + has\_moved — логический флаг, указывающий, перемещалась ли фигура ранее.
* Методы:
  + move(board, square) — перемещает фигуру на указанную клетку, если это допустимо.
  + get\_moves(board) — возвращает список доступных для перемещения клеток.
  + attacking\_squares(board) — определяет клетки, находящиеся под атакой фигуры.
* Наследники:
  + Pawn (пешка): движется вперёд, но бьёт по диагонали. При достижении последней линии превращается в выбранную фигуру.
  + Rook (ладья): перемещается по горизонтали и вертикали.
  + Knight (конь): перепрыгивает через другие фигуры, двигаясь буквой "Г".
  + Bishop (слон): перемещается по диагонали на любое количество клеток.
  + Queen (ферзь): объединяет ходы ладьи и слона.
  + King (король): движется на одну клетку в любом направлении. Может выполнять рокировку.

**5. Дизайн данных**

* Структура данных:
  + squares — список объектов Square, представляющих клетки доски.
  + config — двумерный список, представляющий начальную расстановку фигур.
* Формат данных:
  + Позиции фигур хранятся как пары координат (x, y).
  + Цвет фигур задаётся строкой "white" или "black".

**6. Интерфейсы системы**

* Внутренние интерфейсы:
  + Класс Game: управляет взаимодействием между игровыми состояниями, такими как главное меню и процесс игры. Вызывает методы класса Board для обработки хода, проверки шаха и мата, а также отрисовки доски и фигур.
  + Класс Board: взаимодействует с:
    - Square для управления клетками доски, их состоянием (подсветка, занятость фигурой) и координатами.
    - Piece и её наследниками для обработки логики перемещения фигур, определения доступных ходов, а также проверки шаха и мата.
* Внешние интерфейсы: пользователь взаимодействует с игрой через мышь:
  + ЛКМ:
    - В меню: выбор кнопок "Начать игру" или "Выход".
    - В игре: выбор фигуры и перемещение её на доступное поле.
  + ПКМ: отмена выбора фигуры (если применимо).

**7. Пользовательский интерфейс**

* Описание интерфейса:
  + Главное меню:
    - Кнопки "Начать игру" и "Выход".
  + Игровое окно:
    - Шахматная доска 8x8 клеток.
    - Заголовок окна отображает текущий ход или победителя.

**8. Требования к производительности**

* Производительность системы:
  + Игра должна поддерживать стабильную частоту кадров не менее 30 FPS.
  + Загрузка главного меню не должна превышать 2 секунд.
* Ограничения ресурсов: максимальное использование оперативной памяти — 100 МБ.

**9. Безопасность**

* Обработка ошибок: используются блоки try-except для обработки исключений при загрузке ресурсов и работе с библиотекой Pygame.
* Валидация данных: проверка наличия всех необходимых графических файлов перед началом игры.

**10. Обеспечение качества**

* Стратегия тестирования:
  + Разработка юнит-тестов для классов Board, Square, Piece и их методов.
  + Бета-тестирование с реальными пользователями.
* Критерия приемки:
  + Все шахматные правила соблюдены.
  + Игра завершает партию при достижении мата.
  + Интерфейс работает корректно и без задержек.

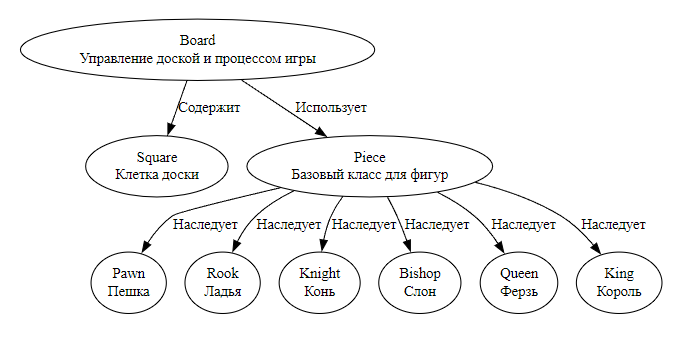
**11. Требования к среде разработки**

* Инструменты разработки:
  + Язык программирования: Python 3.11+.
  + Библиотеки: Pygame 2.6+.
  + Среда разработки: Visual Studio Code.

**12. Приложения**

**Диаграммы UML:**

* Диаграмма классов, показывающая иерархию Board, Square и Piece:



* Диаграмма состояний, отображающая переходы между главными этапами игры:

