**SimpleDungeon — Design Document (v0.1)**

**Автор:** Илингин Даниил  
**Дата:** 08.09.2025  
**Статус:** Draft  
**Версия документа:** 0.1

**1. Резюме**

SimpleDungeon — это консольная Role Play Game с пошаговыми боями, исследованием подземелий, взаимодействием с NPC.  
Основная цель: создать структурированный и расширяемый проект, который позволит добавлять новые механики (скиллы, предметы, квесты) без полного переписывания кода. Для этого в будущем планируется отдельный редактор.

**2. Цели и метрики успеха**

* Сделать удобную архитектуру для расширения игры (новые монстры, предметы, механики).
* Поддержка сохранений через JSON (возможность продолжать игру).
* Время загрузки данных < 1 сек.
* Отрисовка интерфейса без наложений и артефактов.
* Покрытие ключевых механик тестами (DataImporter, боевая система, выполнение квестов, проведение диалогов, торговля с NPC и т.д.).

**3. Scope (объём)**

**Включено**

* Экран главного меню.
* Система боёв: HP, скорость, очередь ходов, лог событий.
* Система предметов и инвентаря.
* Экран взаимодействия с NPC.
* Сохранение/загрузка прогресса из JSON.

**Не включено (в будущем)**

* Мультиплеер.

**4. Stakeholders (участники)**

* **Автор и разработчик:** Илингин Даниил.
* **Тестирование:** вручную самим разработчиком (позже — автоматизация).
* **Будущие пользователи:** игроки, любящие текстовые RPG.

**5. Пользовательские сценарии**

1. Игрок запускает игру → видит главное меню → может начать новую игру или загрузить сохранение.
   * Заметка: нажатие клавиши вызывает корректный экран.
2. Игрок входит в бой с монстром → появляются шкалы HP и скорости → игрок и монстр делают ходы.
   * Заметка: HP уменьшается корректно, лог событий отображается без перекрытий.
3. Игрок находит предмет → добавляет в инвентарь → может использовать его в бою.
   * Заметка: предметы хранятся в JSON и загружаются при старте.
4. Игрок взаимодействует с NPC → появляется экран диалога → можно вернуться назад.
   * Заметка: клавиша Q возвращает именно в предыдущее меню.

**6. Нефункциональные требования**

* **Производительность:** бой и переходы между экранами должны происходить без видимой задержки (<100 ms).
* **Совместимость:** Windows/Linux (консоль).
* **Расширяемость:** лёгкое добавление новых монстров, предметов и NPC через JSON.
* **Надёжность:** при ошибке загрузки данных игра должна выводить сообщение и использовать дефолтные настройки.

**7. Архитектура и структура проекта**

**Основные модули:**

* **Program.cs** — точка входа.
* **Game.cs** — основной цикл игры.
* **Screens/** — экраны (меню, бой, взаимодействие, титульный экран).
* **Entities/** — сущности (Player, Enemy, NPC, Item).
* **Battle/** — логика боя, очередь ходов, расчёт урона.
* **Data/** — DataImporter, JSON-файлы.
* **Utils/** — вспомогательные классы (отрисовка, лог).

**Диаграмма (упрощённая)**

Program → Game → ScreenManager

| | |

| | └─> BattleScreen / MenuScreen / InteractionScreen

| └─> DataImporter → JSON

└─> Entities (Player, Enemy, NPC, Item)

**8. Детали реализации**

**DataImporter**

* Загружает сущности из JSON (монстры, предметы, NPC).
* Методы: LoadEnemies(), LoadItems(), LoadNPCs().
* При ошибке: логировать и возвращать пустой список.

**Боевая система**

* Очередь ходов на основе **скорости** (чем выше скорость, тем раньше ход).
* Формула урона: Damage = Attack - Defense + Random(0..crit).
* Лог боя: отображается снизу, максимум N строк.
* События боя записываются в список и выводятся построчно.

**Экран взаимодействия**

* При нажатии Q игрок возвращается именно в предыдущее меню (а не в корневое).

**Инвентарь**

* JSON структура:
* {
* "id": 1,
* "name": "Healing Potion",
* "effect": "heal",
* "value": 20
* }
* Методы: UseItem(itemId), AddItem(item).

**9. Тестирование**

* **Unit-тесты:**
  + DataImporter (корректная загрузка/ошибки).
  + Формула урона (граничные значения).
  + Очередь ходов (правильный порядок).
* **Integration-тесты:**
  + Переходы между экранами.
  + Использование предмета в бою.
* **Manual tests:**
  + UI: перекрытие логов, отображение HP/Speed.

**10. План разработки**

**Этап 1 (готово частично)**

* Главный экран, меню, отрисовка.
* Система боя (HP, скорость).
* Загрузка данных из JSON.

**Этап 2 (в процессе)**

* Инвентарь.
* Исправление перекрытия лога.
* Корректный возврат из меню (Q).

**Этап 3 (план)**

* Добавить предметы с эффектами (баффы, дебаффы).
* Диалоги с NPC.
* Расширение системы квестов.

**11. Риски и планы снижения**

* **Риск:** JSON-файлы повредятся → игра не загрузится.
  + *Митигейшн:* дефолтные значения при ошибке.
* **Риск:** код станет слишком связанным (hardcode).
  + *Митигейшн:* вынос логики в отдельные классы, больше интерфейсов.

**12. Приложения**

* [GitHub репозиторий](https://github.com/Lunlis-Console/SimpleDungeon.git)
* Глоссарий:
  + **HP** — здоровье.
  + **Speed** — скорость, влияет на порядок хода.
  + **NPC** — неигровой персонаж.