Механики

Core gameplay
Explore Neo-Eden Zones → 2. Engage in Tactical Combat → 3. Manage Virus/Clones → 4. Interact with Factions → 5. Upgrade Gear/Implants → REPEAT

Свободная форма

1. Тактический бой
   - Пошаговая система с action points, укрытиями, дронами и ловушками.
   - Уникальность: игрок может взламывать вражеские дроны прямо во время боя, изменяя динамику сражения.

2. Система Клонов/Андроидов
   - После смерти — возрождение в новом теле с рандомными трейдами (например: +25% урон, но шанс самоуничтожения).
   - Каждое возрождение увеличивает вирус, ухудшая статы, но открывает мутации.

3. Дроновая тактика
   - Игрок может развивать дронов как бойцов, взломщиков или медиков.
   - Дроны адаптируются к стратегии игрока.

4. События и исследования
   - Процедурные ивенты с выбором действия, часто зависящим от текущих статов (например, здоровье < 50% → появляется доп. опция).

5. Репутационная система фракций
   - Действия влияют на репутацию и открывают фракционные награды/локации.
   - Высокая репутация с враждующими фракциями может вызвать "события-предательства".

Формализованное описание механик

ER-модель: Пример на основе системы клонов
Сущности:

Player — ID, текущее тело, вирус
CloneBody — ID, трейды, вирус, бонусы

Trait — ID, название, эффект
Mutation — ID, эффект, вирус-кост

Связи:
Player (1) — (M) CloneBody
CloneBody (M) — (M) Trait
Player (M) — (M) Mutation

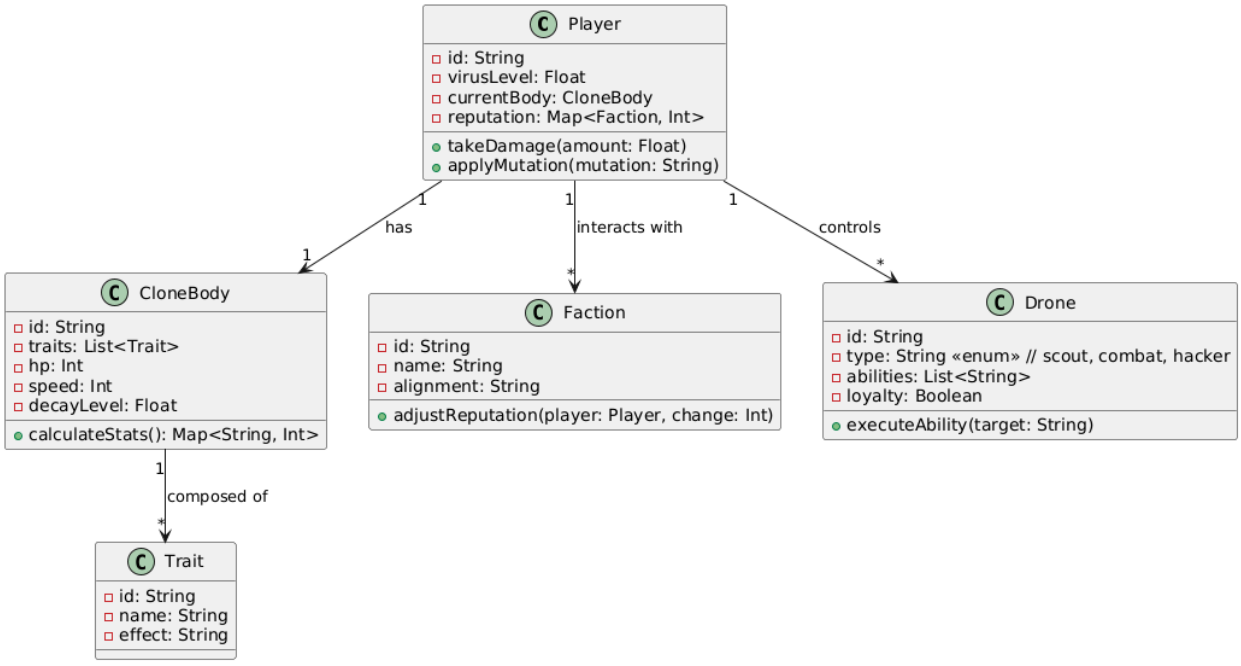
Формула урона

$$DAMAGE = BASE\_DAMAGE \times (1 + WEAPON\_MOD - ENEMY\_ARMOR\_CLASS)$$

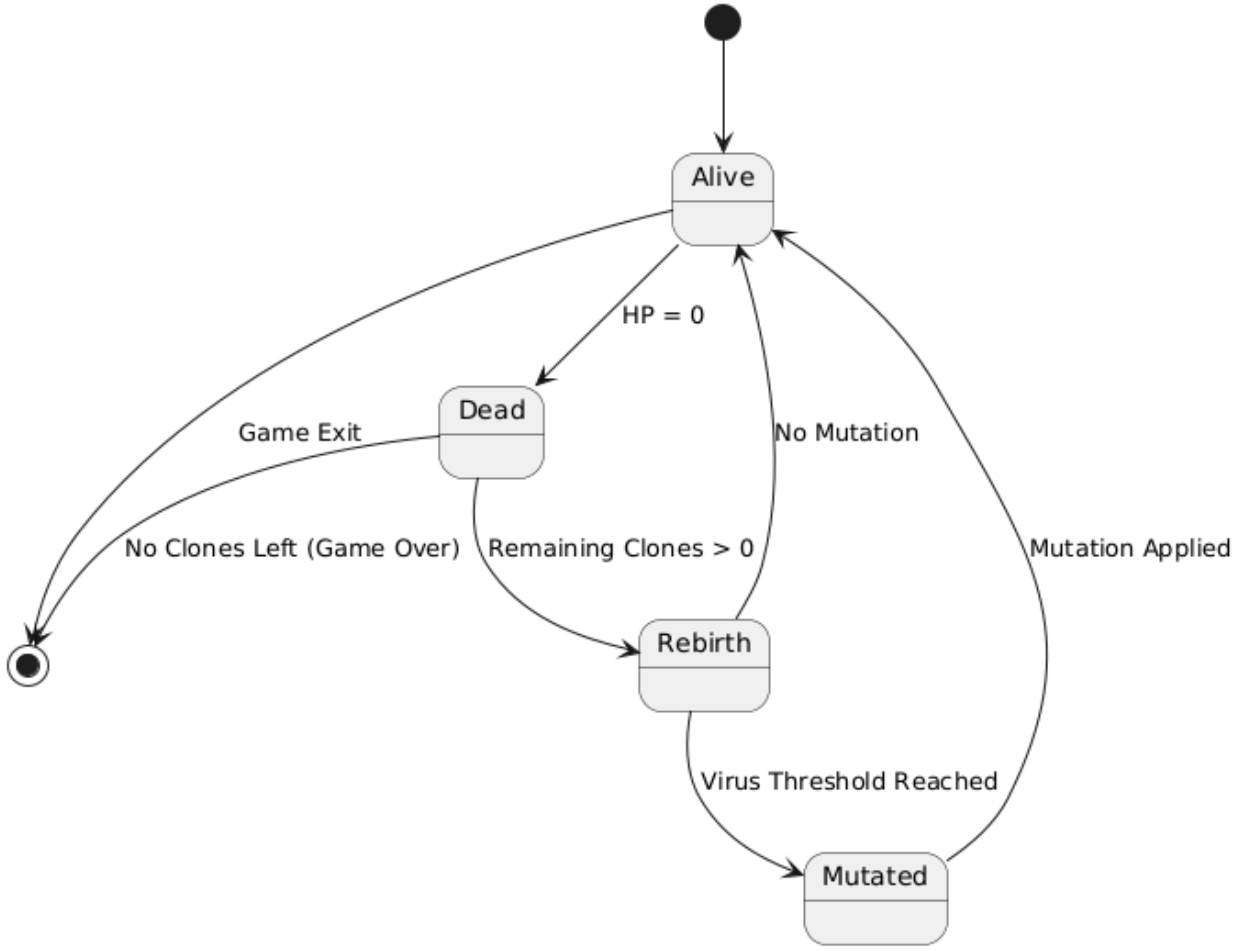
Параметры:

- BASE_DAMAGE: 20
- WEAPON_MOD: от +0.2 (Plasma Cutter) до +0.8 (Quantum Blade)
- ENEMY_ARMOR_CLASS: от 0.1 до 0.6

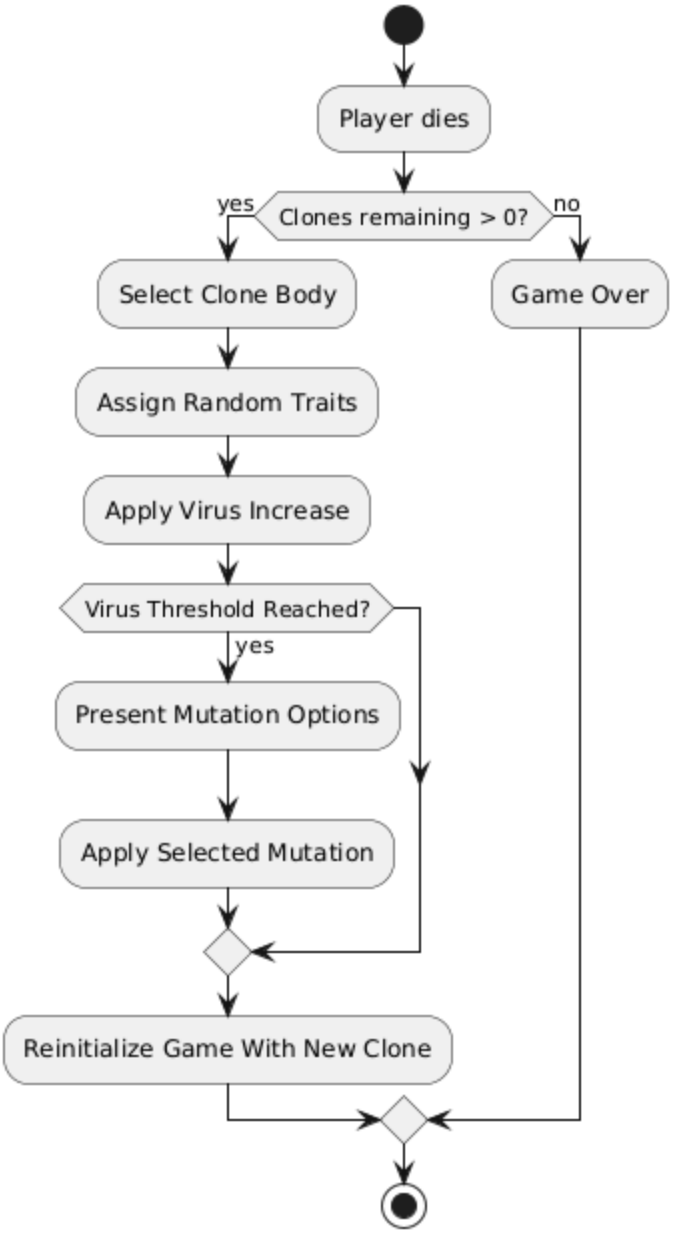
UML Class Diagram: описывает классы Player, CloneBody, Trait, Faction, Drone.



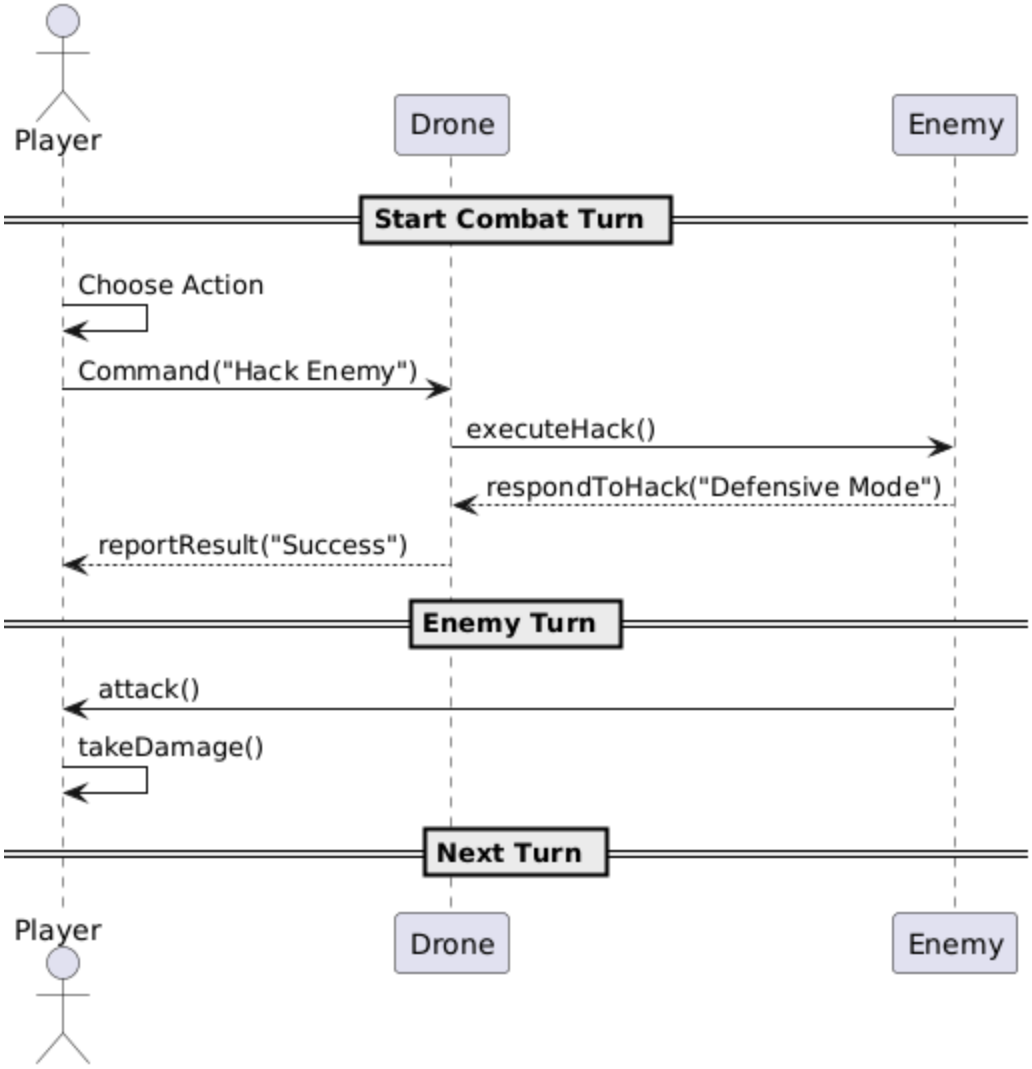
UML State Machine: например, состояния игрока: Alive → Dead → Rebirth → Mutated.



UML Activity: цепочка действий при смерти: Die → SelectClone → ApplyTrait → AddVirus → Reinit.



UML Sequence: взаимодействие игрока, дрона и врага в бою.



Общая архитектура

Неформальное описание архитектуры

1. Подсистема Рендеринга

- Отвечает за отрисовку интерфейса в терминальном виде: ASCII-карта, боевые логи, HUD.
- Состоит из: RenderEngine, UIScreenManager, AsciiMapRenderer.

2. Игровая логика (Game Core)

- Управляет состоянием игрока, боевой системой, клонами, вирусом и фракциями.
- Состоит из: GameStateManager, CombatSystem, CloneManager, FactionSystem.

3. ИИ и Поведение

- Контролирует врагов и дронов в бою (например, адаптивное поведение при взломе).
- Состоит из: EnemyAIController, DroneBehavior, DecisionEngine.

4. Процедурная генерация

- Генерирует зоны, события, врагов и лут.
- Состоит из: ZoneGenerator, EventSpawner, LootDistributor.

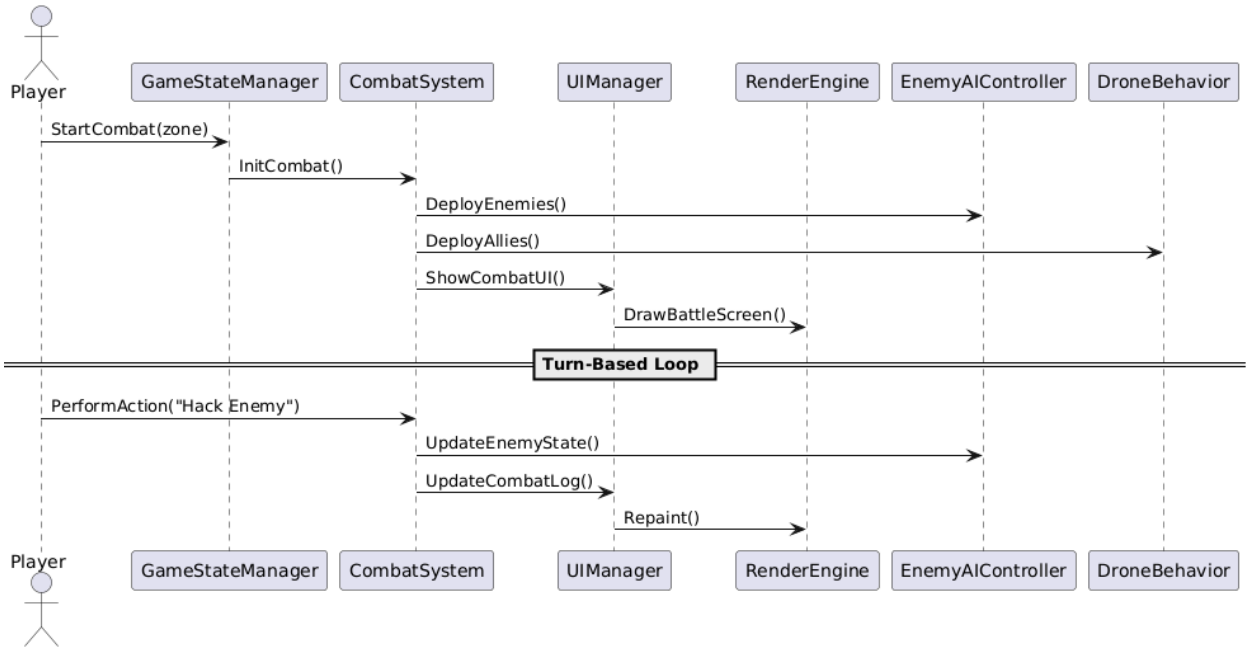
5. Система данных и сохранений

- Загружает/сохраняет клоны, репутации, прогресс.
- Состоит из: SaveManager, PlayerProfile, DataSerializer.

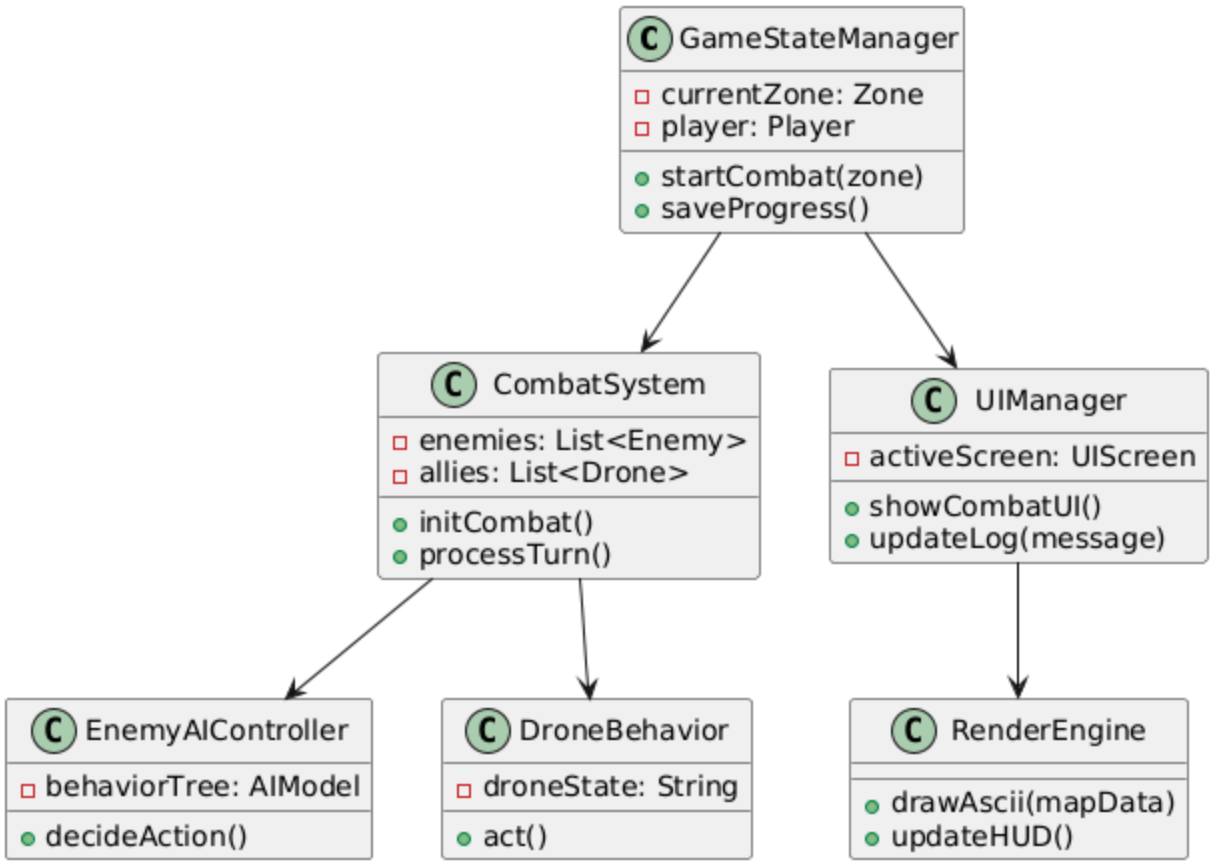
6. Нарратив и мир

- Обрабатывает MemoryFragments, TerminalLogs, ItemLore, влияет на мир через репутацию.
- Влияет на доступные зоны/события через LoreManager.

UML Sequence Diagram — как работают подсистемы при старте боя



UML Class Diagram — архитектура систем



Физическая организация

Пакет / Папка	Назначение	Файлы / Классы внутри

core/	Главная логика игры	`GameStateManager.h/.cpp`
		`CombatSystem.h/.cpp`
		`CloneManager.h/.cpp`

entities/	Сущности игрового мира	`Player.h/.cpp`
		`CloneBody.h/.cpp`
		`Trait.h/.cpp`
		`Faction.h/.cpp`
		`Drone.h/.cpp`

ai/	Враги и дроны, поведение	`EnemyAIController.h/.cpp`
		`DroneBehavior.h/.cpp`
		`DecisionEngine.h/.cpp`

render/	Отрисовка ASCII-интерфейса	`RenderEngine.h/.cpp`
		`AsciiMapRenderer.h/.cpp`
		`HUDRenderer.h/.cpp`

ui/	Управление экранами и пользовательским вводом	`UIManager.h/.cpp`
		`UIScreen.h/.cpp`
		`CombatLogScreen.h/.cpp`

procedural/	Генерация уровней и ивентов	`ZoneGenerator.h/.cpp`
		`EventSpawner.h/.cpp`
		`LootDistributor.h/.cpp`

data/	Система сохранений и профилей	`SaveManager.h/.cpp`
		`PlayerProfile.h/.cpp`
		`DataSerializer.h/.cpp`

narrative/	Лор и история	`LoreManager.h/.cpp`
		`MemoryFragment.h/.cpp`
		`TerminalLog.h/.cpp`

config/	Настройки и конфигурации	`Zone_config.json`
		`Traits.json`
		`faction_table.json`

assets/	ASCII-шрифты, шаблоны, эффекты	`Ascii_tileset.txt`
		`Fx_beep.wav`
		`intro_narration.ogg`

tests/	Юнит-тесты	`TestCombatSystem.cpp` `TestCloneSystem.cpp`

package diagram

- **core**, **ai**, **render**, **ui**, **procedural**, **narrative**, и **data** — независимые модули, импортируемые `Main.h/.cpp`.
- **core** зависит от **entities** и **data**.
- **ui** зависит от **render** и **core**.
- **ai** зависит от **entities** и частично от **procedural**.