UML: Диаграмма состояний (State Machine) — Боевая система

[Выбор программы] --> [Атака]

[Атака] --> [Проверка хп врага]

[Проверка хп врага] --> [Враг мертв] --> [Проверка выпадения аптечки]

[Проверка хп врага] --> [Враг жив] --> [Ответный удар врага] --> [Проверка хп игрока]

[Проверка хп игрока] --> [Игрок мертв] --> [Game Over]

[Проверка хп игрока] --> [Игрок жив] --> [Выбор программы]

UML: Диаграмма деятельности (Activity) — Исследование

[Старт] → [Выбор направления] → [Загрузка локации] → [Проверка на врагов]

→ [Бой/обход] → [Проверка на чип] → [Сбор чипа, если есть]

→ [Переход к следующей локации или завершение игры]

Архитектура

1. Подсистема рендеринга (UI / Консольный вывод)

- Отвечает за отображение текста игроку: описание локаций, боевых сообщений, состояние здоровья, доступные действия и т.д.
- Взаимодействует с другими подсистемами через текстовые команды и сообщения.

2. Подсистема управления игрой (Game Loop / Input Handler)

- Основной цикл игры.
- Считывает ввод игрока, передает его в нужную подсистему (бой, инвентарь, движение и т.д.).
- Обеспечивает смену состояний игры и переход между режимами.

3. Подсистема боевой логики

- Управляет пошаговым боем.
- Принимает решения на основе выбранной игроком боевой программы и типа врага.
- Отвечает за расчет урона и проверку условий завершения боя.

4. Подсистема инвентаря и ресурсов

- Хранит список предметов (до 15), позволяет использовать аптечки.
- Обновляет инвентарь после боя.

5. Подсистема состояния игрока и врагов

- Хранит информацию о здоровье игрока и врагов.
- Используется в боевой системе, а также влияет на рендеринг (например, отображение HP).

6. Подсистема исследования / перемещения

- Управляет переходами между локациями.
- Отслеживает, где находится игрок и какие локации уже посещены.
- Управляет получением чипов в локациях.

7. Подсистема сохранения прогресса

- Сохраняет состояние игры каждые 5 действий игрока.
- Хранит все ключевые данные: HP, инвентарь, пройденные локации, собранные чипы.

Диаграмма классов

```
class GameEngine {
 - locations: List<Location>
 - player: Player
 - saveManager: SaveManager
 + runGame()
}
class Player {
 - health: int = 500
 - inventory: Inventory
 - chipsCollected: int
 + attack(enemy: Enemy)
}
class Enemy {
 <<abstract>>
 # health: int
```

```
  # damage: int
  + takeDamage(amount: int)
}
 class SiliconLife {
  + weakPoint: bool = true
}
 class Inventory {
  - items: List<Item>
  + addItem(item: Item)
}
 class Item {
  <<abstract>>
  - name: string
}
 class MedKit {
  + healAmount: int = 150
}
 class Chip {
  + location: string
}
 class CombatManager {
  - damageTable: Map<EnemyType, Map<AttackType, int>>
  + startBattle(player: Player, enemy: Enemy)
}
 GameEngine --> Player
GameEngine --> Location
GameEngine --> SaveManager
Player --> Inventory
```

Inventory --> Item

Item <|-- MedKit

Item <|-- Chip

Enemy <|-- SiliconLife

Enemy <|-- Planter

Enemy <|-- AIRobot

CombatManager --> Player

CombatManager --> Enemy

Физическая организация

Файл	Описание
Main.cpp	Точка входа, инициализация движка.
GameEngine.h/cpp	Управление игровым циклом и системами
Player.h/cpp	Класс игрока и его методы.
Enemy.h/cpp	Базовый класс и подклассы врагов.
CombatManager.h/cpp	Логика боя и таблицы урона.
Inventory.h/cpp	Хранение предметов и ограничения.
Location.h/cpp	Описание локаций и механика сбора чипов.
SaveManager.h/cpp	Сериализация данных и автосохранение.

Диаграмма пакетов

package "Core" {

[main.cpp]

[GameEngine.h]

[GameEngine.cpp]

```
}
 package "Entities" {
  [Player.h]
  [Player.cpp]
  [Enemy.h]
  [Enemy.cpp]
  [SiliconLife.h]
  [Planter.h]
  [AIRobot.h]
}
 package "Systems" {
  [CombatManager.h]
  [CombatManager.cpp]
  [Inventory.h]
  [Inventory.cpp]
  [Location.h]
  [Location.cpp]
  [SaveManager.h]
  [SaveManager.cpp]
}
 "Core" --> "Entities" : Зависит от
"Core" --> "Systems" : Зависит от
"Systems" --> "Entities" : Зависит от
```