SCOREWARRIOR

Тестовая боевка

Цель

Необходимо продемонстрировать знание принципов проектирования, обеспечить максимальную простоту и расширяемость кода для команды.

Код должен быть понятен разработчику с квалификацией Junior (ближе к Middle) который хорошо знает основы языка и паттернов.

В этом должно помочь понимание: KISS, YAGNI, DRY, SOLID, Low Coupling и High Cohesion, желание писать простой но гибкий к расширению код.

Что не нужно делать:

- Переусложнять решение (Overengineering), делать его сложным для изучения.
- Уделять внимание производительности - мы ее не оцениваем в этом задании
- Использовать потоки, подключать библиотеки
- Тратить слишком много времени на задачу (рекомендуем не заморачиваться на мелочах, найти минимум)
    - Мы не замеряем время и никак его не ограничиваем, но рассчитываем что вы не потратите более 6-8 часов суммарно.

Задача

Делаем симуляцию битвы юнитов на карте, чтобы проводить на ней эксперименты с механиками.

Дана карта размера H*W клеток, на карте размещается N юнитов.

Один юнит занимает одну клетку (координату)

В одной клетке не может быть нескольких юнитов.

Существуют юниты разного типа с разным действиями и характеристиками.

Каждый игровой тик (ход), каждый юнит может выполнить одно действие, независимо от других юнитов.

Юниты выполняют свои действия за тик в том порядке в котором они были созданы

При запуске в программу подается список команд, по мере исполнения программа выводит произошедшие события.

Симуляция заканчивается когда нет ни одного юнита с действием на следующий тик.

Для всех юнитов:

- Параметр: Id - уникальный идентификатор юнита, целое число.
- Характеристика: HP (Health points)
    - Если равна 0, то юнит умирает. В следующем тике его не будет на карте.
    - Юнит совершает свое действие за тик даже если он умер в этом тике.
- Действие: Move
    - Юнит идет в заданную точку на карте.
        - Каждый тик может перейти в любую из 8-ми клеток вокруг него по направлению к цели
        - Если клетка занята, то юнит не совершает движения.
        - Поиск пути не нужен, на карте нет статических препятствий.

Типы юнитов:

1. Воин

- Характеристика: Strength
- Действие: Melee Attack
    - Оно приоритетнее Move.
    - Если в радиусе одной клетки есть другие юниты, то выбирается и атакуется один юнит.
    - Выбирает самого самого слабого по HP, либо самого младшего по Id, и сражается с ним до смерти врага (или своей)
    - За один тик вычитает из HP другого юнита значение равное его Strength.

2. Лучник

- Характеристики: Strength, Range и Agility
- Действие: Melee Attack - такое же как у Воина.
- Действие: Range Attack
    - Оно приоритетнее Move и Melee Attack
    - Если в радиусе от 2х клеток (включительно) до Range (включительно) есть другие юниты, то выбирает и атакует юнита.
    - Выбирает самого ближайшего, либо самого слабого по HP, либо самого младшего по Id, и сражается с ним до смерти врага (или своей)
    - За один тик вычитает из HP другого юнита значение равное его Agility.

Дальнейшие планы на развитие

Все что описано в этом разделе делать не надо, но надо учитывать в дизайне, что эти места будут точкой расширения функционала:

- Могут быть придуманы новые действия для юнитов, такие как: Лечение, Невидимость, Усиление (Бафф), Ускорение (Спринт)

    - Для действий могут вводиться новые характеристики
    - Могут быть усложнены такие действия как Range Attack (к примеру кол-во выстрелов будет ограничено)
- Могут быть придуманы новые юниты, такие как: Лекарь, Маг, Маг-Воин, Лучник-Маг - в любой комбинаторике действий и характеристик.

Ввод-вывод

Для ввода-вывода уже написана часть кода, чтобы вам ее не писать - скачайте готовый проект (предоставит HR).

Ценность парсера команд и вывода событий в лог не особо важны.

- мы ожидаем что “завтра” парсер и вывод событий могут быть дополнены другими (к примеру RPC)

По желанию вы можете поменять в реализации что угодно, но нельзя менять сам формат ввода и вывода - т.к. у нас уже заготовлены файлы сценариев.

Вы сможете рассказать про любые недостатки и проблемы на интервью.

Команды

При запуске - приложение получает в argv путь к файлу со сценарием работы, в котором описаны команды.

Существуют следующие: CREATE_MAP, SPAWN_WARRIOR, SPAWN_ARCHER, MARCH и WAIT

Пример сценария:

```
// Формат: COMMAND_NAME arg1 arg2 ...
// Для каждой команды в заданной позиции аргументов находится конкретное свойство

// Создать карту widht=10 height=10 (может быть только одна и в самом начале)
CREATE_MAP 10 10

// Разместить воина id=1 в точке x=0 y=0 с hp=10 и strength=2
SPAWN_WARRIOR 1 0 0 10 2

// Разместить лучника id=2 в точке x=9 y=0 с hp=10, strength=2, range=2, agility=3
SPAWN_ARCHER 2 9 0 5 1 5 2

// Отправить юнит id=1 в точку 9 0
MARCH 1 9 0

// Отправить юнит id=2 в точку 0 0
MARCH 2 0 0

// Подождать 10 тиков прежде чем выполнять следующие команды
WAIT 10
```

```
// Разместить воина id=3 в точке x=9 y=9 с hp=10 и strength=2
SPAWN_WARRIOR 3 9 9 10 2

// Отправить юнит id=3 в точку x=0 y=0
MARCH 3 0 0
```

События

Программа должна выводить все происходящие события в stdout.

Существуют следующие события: MAP_CREATED, UNIT_SPAWNED, MARCH_STARTED, UNIT_MOVED, UNIT_ATTACKED, MARCH_ENDED, UNIT_DIED

Пример вывода:

```
// Формат: [TICK_NUMBER] EVENT_NAME field1=value field2=value...

[1] MAP_CREATED width=10 height=10
[1] UNIT_SPAWNED unitId=1 unitType=Archer x=5 y=3
[1] UNIT_SPAWNED unitId=2 unitType=Warrior x=5 y=3
[1] MARCH_STARTED unitId=1 x=5 y=3 targetX=7 targetY=9
[2] UNIT_MOVED unitId=1 x=6 y=4
[3] UNIT_ATTACKED attackerUnitId=1 targetUnitId=2 damage=5 targetHp=0
[4] MARCH_ENDED unitId=1 x=7 y=9
[5] UNIT_DIED unitId=1
```

В процессе выполнения задачи вы можете прислать вопросы для уточнения цели и задания.

Технические требования:

ОС: Ubuntu

Компилятор: clang 15+

Стандарт: От C++17 до C++20

Система сборки: cmake (в проекте должен быть CMakeLists.txt)

Задание необходимо опубликовать в любом гит-репозитории (github, bitbucket, и т д).

- Убедитесь, что к нему есть доступ по ссылке без авторизации