МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «ООП»

Тема: Разработка игровой системы с тактическими боями.

Студент гр. 4384 _____ Стукалкин М. М.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2025

Цель.

Разработать игровую систему, реализующую механику пошаговых тактических боев с управлением персонажем, врагами и специальными сооружениями. Построить архитектуру с учетом принципов ООП и расширяемости

Задание.

Создать интерфейс карточки заклинания. Заклинание должно применяться игроком. На использование заклинания игрок тратит один ход.

Создать класс “руки” игрока, которая содержит все карточки заклинаний, которые игрок может применить в свой ход. Изначально рука игрока содержит только одно случайное заклинание. Реализовать возможность получать новые заклинание игроком, например, тратить очки на покупку или после уничтожения определенного кол-ва врагов. Размер “руки” должен быть ограничен и задается через конструктор.

Реализовать интерфейс заклинанием прямого урона. Это заклинание при использовании должно наносить урон врагу или вражескому зданию, если они находятся в достижимом радиусе. Если в качестве цели не выбран враг или вражеское здание, то заклинание не используется.

Реализовать интерфейс заклинания урона по площади. Это заклинание при использовании в допустимом радиусе наносит урон по области 2 на 2 клетки. Заклинание используется, даже если там нет никого.

На 8/4/1.5 баллов:

Реализовать интерфейс заклинания ловушки. Заклинание размещает на поле ловушку, если враг наступает на клетку с ловушкой, то ему наносится урон, и ловушка пропадает.

Создать класс вражеской башни. Вражеская башня размещается на поле, и если в радиусе ее атаки появляется игрок, то применяет ослабленную версию заклинания прямого урона. Не может применять заклинание несколько ходов подряд.

На 10/5/2 баллов:

Реализовать интерфейс заклинания призыва. Заклинание создает союзника рядом с игроком, который перемещается самостоятельно.

Реализовать интерфейс заклинание улучшения. Заклинание улучшает следующее используемое заклинание:

Заклинание прямого урона - увеличивает радиус применения

Заклинание урона по площади - увеличивает площадь

Заклинание ловушки - увеличивает урон

Заклинание призыва - призывает больше союзников

Заклинание улучшение - накапливает усиление, то есть при применении следующего заклинания отличного от улучшения, все улучшения применяются сразу.

Архитектурные решения и обоснование.

1. **Интерфейс SpellCard**

Назначение: Интерфейс для всех типов заклинаний в игре.

Атрибуты: Нет, интерфейс содержит только чисто виртуальный метод.

Методы:

virtual bool apply(const SpellContext& context, GameArea& gamearea) = 0 — основной метод применения заклинания, который принимает контекст применения заклинания и игровое поле, на котором заклинание будет применено. Возвращает true, если заклинание убило цель.

virtual std::string accept(const SpellCardVisitor& visitor) const = 0 – метод, отвечающий за реализацию паттерна Visitor для вывода типа заклинания, не используя downcasting.

Обоснование: Использование интерфейса позволяет единообразно работать с разными типами заклинаний без необходимости проверять их конкретный класс.

2. **Структура SpellContext**

Назначение: SpellContext инкапсулирует всю необходимую информацию для применения заклинания.

Атрибуты:

1. Coords caster_position — координаты объекта, который использует заклинание.
2. std::vector<std::string> target_types — вектор типов врагов, к которым должно применяться заклинание
3. Hand* caster_hand — указатель на руку заклинателя (если применяет игрок, иначе nullptr).

Обоснование: использование контекста добавляет гибкость и расширяемость, при добавлении новых типов заклинаний можно расширить контекст без изменения существующего кода.

3. **Система представления заклинаний (Паттерн Visitor)**

Назначение: отделение логики представления заклинаний от их бизнес-логики, избегание использования downcasting.

class SpellCardVisitor – Интерфейс.

class SimpleTextFormatter – Реализация.

Обоснование: использование данного паттерна позволило избежать добавлений методов по типу get_type() в классы заклинаний.

Минусы: нарушение Open Close Principle – для добавления нового типа заклинания придется изменять и интерфейс, и реализацию.

4. **Класс Hand.**

Назначение: Хранит доступные игроку заклинания, управляет их количеством и процессом получения новых.

Атрибуты:

std::vector<Spell*> hand — контейнер для карточек заклинаний.

Переменные для подсчёта усилений и количества убийств, необходимых для получения новых заклинаний.

Ограничение размера руки, заданное через конструктор.

Методы:

add_random_spell() — добавляет случайное заклинание при условии, что место в руке есть.

plus_kill() — увеличивает счётчик убийств, и при достижении заданного порога вызывает добавление нового заклинания.

Геттеры и вспомогательные методы для работы с усилениями и количеством заклинаний.

Обоснование: Класс Hand централизует управление заклинаниями игрока, поддерживая ограниченный набор и прогрессию доступных умений.

5. **Классы заклинаний (DirectSpell, SquareSpell, TrapSpell, SummonSpell, UpgradeSpell).**

5.1 **Класс DamageSpell (Наследуется от SpellCard)**

Назначение: определяет в себе поля double damage и int radius.

Обоснование: в работе есть группа заклинаний, что наносит урон (DirectSpell, SquareSpell, TrapSpell), чтобы не дублировать в каждом из них одни и те же поля был создан класс родитель.

5.2 **Класс DirectSpell (Наследуется от DamageSpell и реализует интерфейс SpellCard)**

Назначение: наносит урон одной ближашей цели (врагу или вражескому зданию) в пределах досягаемого радиуса.

Логика работы метода apply(): получает накопленные усиления из контекста. Увеличивает радиус применения. Ищет все допустимые цели в увеличенном радиусе. Если цели найдены, наносит урон первой цели.

Возвращает: true если цель уничтожена, false в противном случае.

5.3 **SquareSpell (Наследуется от DamageSpell и реализует интерфейс SpellCard)**

Назначение: наносит урон всем врагам в области определенного размера вокруг точки применения.

Логика работы метода apply(): получает накопленные усиления из контекста. Увеличивает радиус области. Находит все цели в области через. Итерируется по всем найденным целям, нанося урон каждой

Возвращает: true если хотя бы одна цель была уничтожена

5.4 **TrapSpell (Наследуется от DamageSpell и реализует интерфейс SpellCard)**

Назначение: размещает ловушку на выбранной клетке поля, которая срабатывает при наступлении врага.

Логика работы метода apply(): получает накопленные усиления, увеличивая урон. Находит все пустые клетки в радиусе размещения. Предлагает игроку выбрать конкретную клетку для установки ловушки. Создает объект Trap с усиленным уроном. Размещает ловушку на выбранной клетке.

Возвращает false, так как само заклинание никого не убивает.

5.5 **SummonSpell (Реализует интерфейс SpellCard)**

Назначение: призывает союзников (Summon), которые автономно атакуют врагов и помогают игроку в бою.

Атрибуты:

1. int allies_count — базовое количество призываемых союзников.
2. int summon_range — радиус, в котором могут появиться союзники
3. double health_ally — здоровье каждого союзника
4. double damage_ally — урон, наносимый союзником

Логика работы метода apply(): увеличивает количество союзников. Находит все пустые клетки в радиусе призыва. Определяет фактическое количество союзников. Призывает союзников на свободные клетки.

Возвращает false так как заклинание никого не убивает.

5.6 **UpgradeSpell (Реализует интерфейс SpellCard)**

Назначение: накапливает усиления (upgrades), которые применяются к следующему использованному боевому заклинанию.

Логика работы метода apply(): Получает указатель на руку игрока из контекста. Увеличивает счетчик усилений. Выводит информацию о текущем количестве накопленных усилений.

Возвращает false так как не убивает врагов напрямую.

Механика накопления: усиления накапливаются при каждом использовании UpgradeSpell. Все накопленные усиления применяются к следующему боевому заклинанию. После применения боевого заклинания счетчик усилений обнуляется. Игрок может накопить несколько усилений подряд для мощной атаки.

6. **Класс Tower (Наследуется от Building)**

Назначение: здание атакующие игрока и его союзников заклинанием прямого урона с периодическим откатом способности.

Атрибуты:

1. double health — здоровье башни (наследовано от Object)
2. Coords coords — позиция на карте (наследовано от Object)
3. int step — текущий счетчик ходов до следующей атаки
4. int step_cd — длительность отката способности (cooldown)
5. int radius — радиус обнаружения и атаки игрока
6. double damage — урон, наносимый башней

Методы:

void do_turn(GameArea& gamearea) — основной метод, вызываемый каждый ход: проверка отката, поиск целей, ищет игрока и союзников в радиусе, если цели найдены, создает временный объект DirectSpell, применяет ослабленную версию заклинания прямого урона (меньший урон чем у игрока), обнуляет счетчик шагов, начиная новый цикл отката

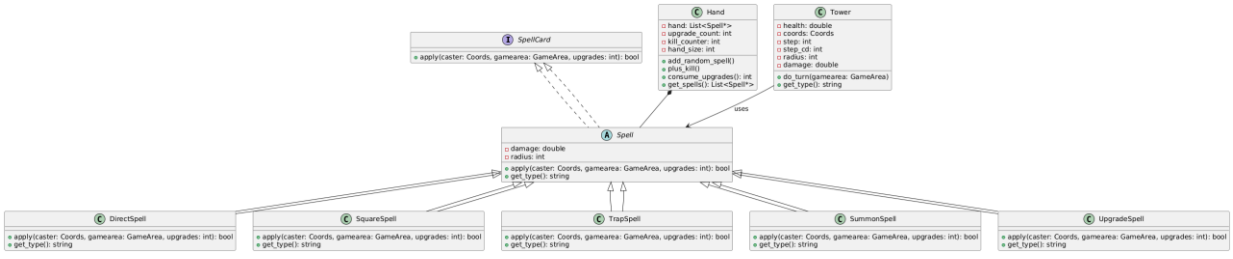

Рисунок 1 - UML-диаграмма классов

Выводы.

Разработана игровая система, реализующая механику пошаговых тактических боев с управлением персонажем, врагами и специальными

сооружениями. Архитектура построена с учетом принципов ООП и расширяемости.