

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

ОТЧЕТ
по лабораторной работе №2
по дисциплине «Объектно-ориентированное программирование»
Тема:

Студентка гр. 4383

Жданова К.В.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2025

Цель работы:

Продолжение разработки игрового приложения. Создание системы взаимодействующих классов для возможности использования заклинания. Построение UML-диаграммы, отображающей структуру классов, их взаимосвязи и архитектурные решения.

Задание.

На 6/3/1 баллов:

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

Примечания:

Интерфейс заклинания должен быть унифицирован, чтобы их можно было единообразно использовать через интерфейс. Не должно быть методов в интерфейсе, которые не используются каким-то классом наследником.

Избегайте явных проверок на тип данных.

Основные теоретические положения.

Интерфейс — структура программы или синтаксиса, определяющая отношение с объектами, объединёнными некоторым поведением.

Виртуальный метод — это метод в базовом классе, который может быть переопределён в его дочерних классах для изменения поведения.

Чистый виртуальный метод (или абстрактный метод) — это метод, объявленный в базовом классе, но не имеющий реализации в нем. Он не имеет тела и предназначен для того, чтобы его реализация была обязательно переопределена в производных классах. Класс, содержащий хотя бы один чистый виртуальный метод, называется абстрактным классом и объекты такого класса нельзя создать

Абстрактный класс в объектно-ориентированном программировании — базовый класс, который не предполагает создания экземпляров. Абстрактные классы реализуют на практике один из принципов ООП — полиморфизм. Абстрактный класс может содержать абстрактные методы и свойства. Абстрактный метод не реализуется для класса, в котором описан, однако должен быть реализован для его неабстрактных потомков.

Указатель `unique_ptr<T>` представляет указатель на тип `T`, который является "уникальным" в том смысле, что что может быть только один объект `unique_ptr`, который содержит один и тот же адрес. То есть не может одновременно быть двух или более объектов `unique_ptr<T>`, которые указывают один и тот же адрес памяти.

Полиморфный класс — это класс, который поддерживает полиморфизм, то есть способность объектов разных классов реагировать на один и тот же вызов метода по-разному

Выполнение работы.

В рамках лабораторной работы была построена UML-диаграмма, отображающая структуру классов, их взаимосвязи и ключевые архитектурные решения. UML-диаграмма представлена на Рисунке 1.

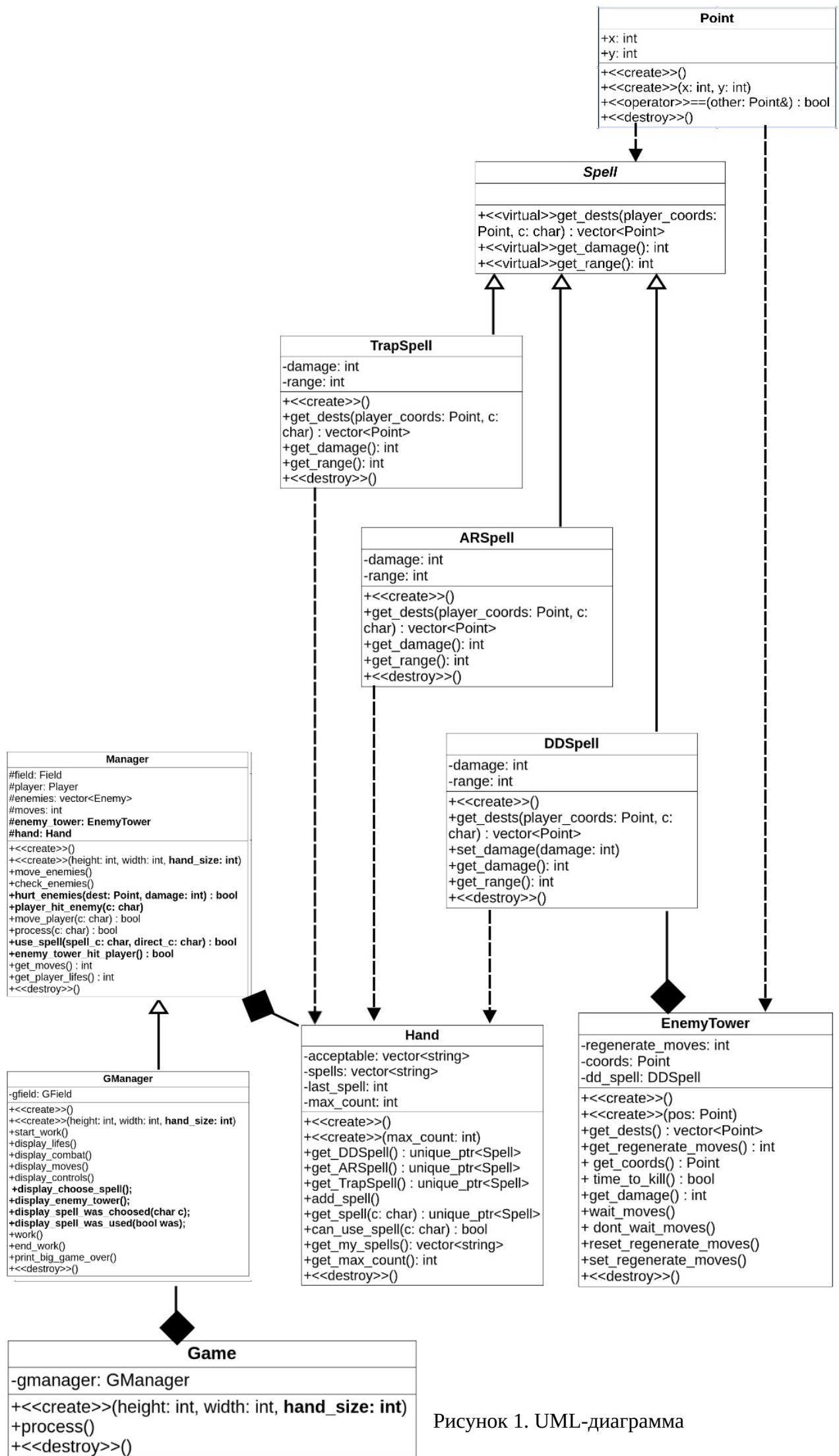


Рисунок 1. UML-диаграмма

Был создан интерфейс карточки заклинания. В ней определен чистый виртуальный метод `vector<Point> get_dests(char c)`, который переопределен в каждом из классов-наследников-реализаций интерфейса. Этот метод возвращает массив координат, по которым будет наноситься урон при использовании заклинания.

Реализованные заклинания: Заклинание прямого урона (`DDSpell`), заклинание урона по площади (`ARSpell`), заклинание ловушки (`TrapSpell`). Их реализации отличаются только расчётом возвращаемых координат. `TrapSpell` возвращает массив из одной координаты — местонахождения игрока.

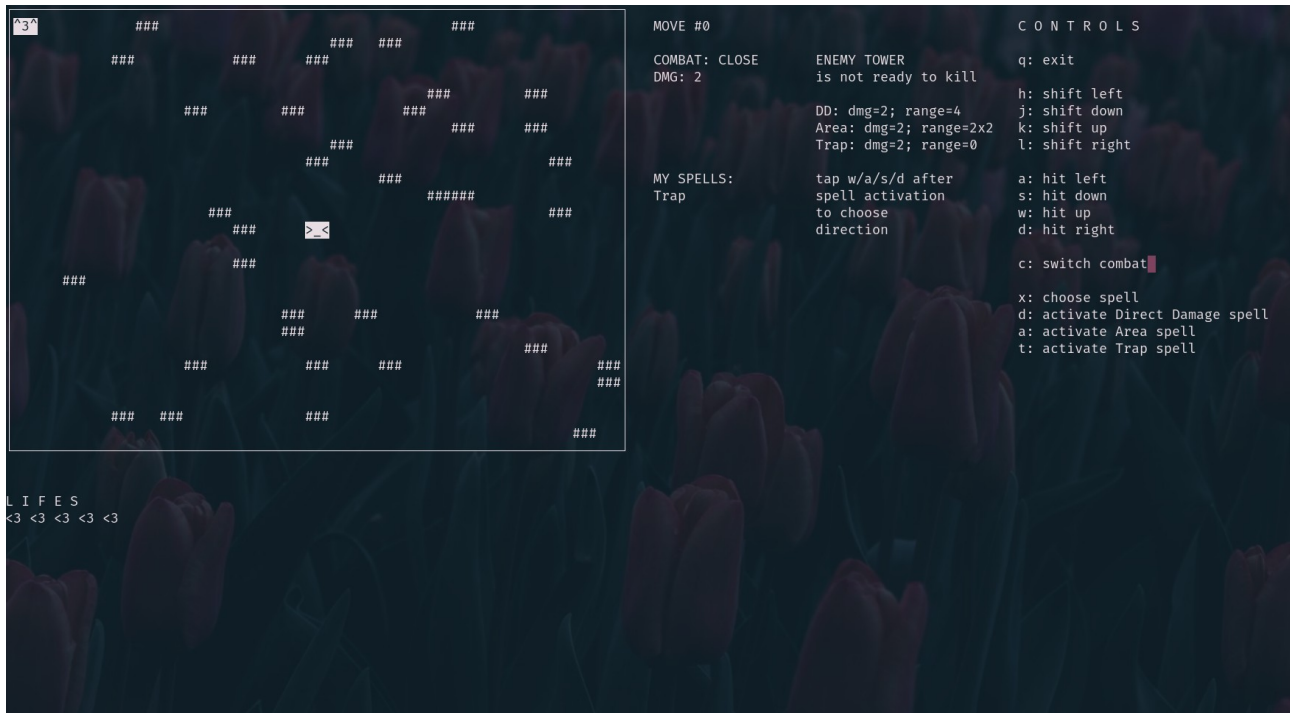
Был реализован класс руки игрока `Hand`. Она хранит в себе список строк — названий заклинаний, а не самих объектов заклинаний. При необходимости вызвать заклинание — объект создается и используется. То есть объекты карточек заклинаний не хранятся, а живут только в момент использования. В методе `unique_ptr<Spell> get_spell` — вызывается соответствующий метод получения заклинания и возвращается указатель на объект `Spell` — так как каждый из классов-карточек заклинаний является наследником класса `Spell`. Заклинания добавляются в руку поочередно.

Был реализован класс Вражеской башни. Она располагается в центре поля. Если после последней попытки успешного удара по игроку прошло достаточно ходов — поле `regenerate_moves` — она начинает пытаться снова ударить игрока. Если попытка была не успешна (игрок не находился в области удара) — она продолжает пытаться его ударить каждый ход, пока не наступит успешная попытка.

В класс `Manager` в поля добавились классы `Hand` и `EnemyTower`, а также метод использования заклинания и работы Вражеской башни.

В класс `GManager` добавились методы вывода информации о новых полях, а также вывод информации о выборе заклинания.

На данный момент игровое приложение имеет следующий вид, представленный на Рисунке 2.



Выводы:

Была продолжена разработка консольного игрового приложения на основе системы взаимодействующих классов для управления игровыми сущностями и игровым полем. Была построена UML-диаграмма, отображающая структуру новодобавленных классов, их взаимосвязи и архитектурные решения.