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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Объектно-ориентирование программирование»
Тема: Реализация игры на C++

Студент гр. 4381 _____ Бабин Д.Е.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2025

Цель работы.

Разработать систему карточных заклинаний для пошаговой игры, реализующую унифицированный интерфейс применения заклинаний, механизм ограниченной "руки" игрока с возможностью её пополнения, а также полиморфное поведение различных типов заклинаний (прямого урона, урона по площади, ловушек, призыва союзников и улучшения), обеспечивающую их взаимодействие с игровыми объектами (врагами, зданиями) и включающую интеллектуальное поведение вражеской башни с ограничениями на использование способностей.

Задание.

Лабораторная работа №2

На 6/3/1 баллов:

1. Создать интерфейс карточки заклинания. Заклинание должно применяться игроком. На использование заклинания игрок тратит один ход.
2. 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.
3. 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.
4. 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:

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

6. 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:

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

8. Реализовать интерфейс заклинание улучшения. Заклинание улучшает следующее используемое заклинание:

a. Заклинание прямого урона - увеличивает радиус применения

b. Заклинание урона по площади - увеличивает площадь

c. Заклинание ловушки - увеличивает урон

d. Заклинание призыва - призывает больше союзников

e. 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.

Выполнение работы.

1. Интерфейс карточки заклинания (spellCard)

Методы интерфейса:

`virtual bool use_card(Visitor& visitor, int x, int y) = 0`

- Назначение: Унифицированный метод применения заклинания через паттерн Visitor

- Реализация: Каждое заклинание реализует свою логику применения через соответствующий метод Visitor

- Соответствие требованиям: Позволяет единообразно применять все типы заклинаний

`virtual int get_price() const = 0`

- Назначение: Получение стоимости заклинания для покупки

- Использование: Проверка возможности покупки карты игроком

`virtual void upgrade(int level) = 0`

- Назначение: Улучшение характеристик заклинания

-Особенности: Реализует механику улучшения для всех типов заклинаний

2. Класс "руки" игрока (handPlayer)

Конструктор и деструктор:

`handPlayer(int size_hand)`

-Инициализация: Создает все типы заклинаний и устанавливает максимальный размер руки

- Начальное состояние: Рука пуста, текущий размер = 0

`~handPlayer()`

- Очистка: Освобождает память от всех созданных заклинаний

Ключевые методы:

`void find_and_use(Visitor& visitor, std::string str, int x, int y)`

Логика обработки:

- Покупка карты: проверка денег и свободного места в руке

- Улучшение карты: применение улучшения к целевой карте

- Использование карты: применение эффекта на поле

- Валидация: Проверка координат, наличия карты в руке, условий применения

`void first_random_card(Visitor& visitor)`

- Реализация требования: Изначально рука содержит одно случайное заклинание

- Механика: Случайный выбор из 5 доступных заклинаний

3. Заклинание прямого урона (Lightning)

Характеристики:

- Урон: 11

- Стоимость: 30

- Ширина атаки: 2 клетки

`bool use_card(Visitor& visitor, int x, int y)`

- Проверки: Нахождение врага в зоне поражения (2x2 клетки)

- Эффект: Нанесение урона врагу через `visitor.visitEnemy()`

- Требование: Заклинание не используется если враг не в радиусе

4. Заклинание урона по площади (Poison)

Характеристики:

- Урон: 6

- Стоимость: 30

- Радиус: 6 клеток

`bool use_card(Visitor& visitor, int x, int y)`

- Особенности: Наносит урон по области через visitor

- Требование: Используется даже если в области никого нет

5. Заклинание ловушки (Trap)

Характеристики:

- Урон: 6
- Стоимость: 35

`bool use_card(Visitor& visitor, int x, int y)`

- Размещение: Устанавливает ловушку на карту через `visitor.visitMap()`
- Активация: При наступлении врага на клетку
- Удаление: Ловушка исчезает после срабатывания

6. Вражеская башня (Tower)

Механика атаки:

`void GameModel::tower_attack()`

- Проверка радиуса: Игрок в радиусе 7 клеток
- Урон: Половина урона от карты Poison (3 единицы)

7. Заклинание призыва (Teammate)

Характеристики:

- Стоимость: 80
- Количество призывов: 1 (увеличивается при улучшении)

`bool use_card(Visitor& visitor, int x, int y)`

- Создание союзника: Через `visitor.visitGame()` добавляет Leon'a на карту
- Автономное движение: Реализовано в `GameModel::leon_journey()` с случайным перемещением

8. Заклинание улучшения (Improve)

Характеристики:

- Стоимость: 60

- Уровень: 1 (увеличивается при улучшении)

`bool use_card(Visitor& visitor, int x, int y)`

- Механика: Не применяется напрямую, а сохраняется в руке для улучшения следующей карты

Система улучшений:

В `handPlayer::find_and_use()`:

- Обнаружение улучшения: Поиск карты Improve в руке

- Смысл: улучшение применяется к целевой карте

-Накопительный эффект: Уровень улучшения увеличивается при последовательном применении

Конкретные улучшения:

- Lightning: Увеличивает ширину атаки

- Poison: Увеличивает радиус применения

- Trap: Увеличивает урон

- Teammate: Увеличивает количество призываемых союзников

- Improve: Увеличивает уровень для следующего улучшения

9. Паттерн Visitor

Класс `Visitor` обеспечивает единообразное взаимодействие с игровой моделью:

- `visitEnemy()` - для заклинаний урона

- `visitGame()` - для призыва союзников

- `visitMap()` - для установки ловушек

- `visitPlayer()` - для проверки и списания денег

- `visitButtons()` - для обновления интерфейса

Ниже на рисунках представлены UML-диаграмма и визуализация игры.

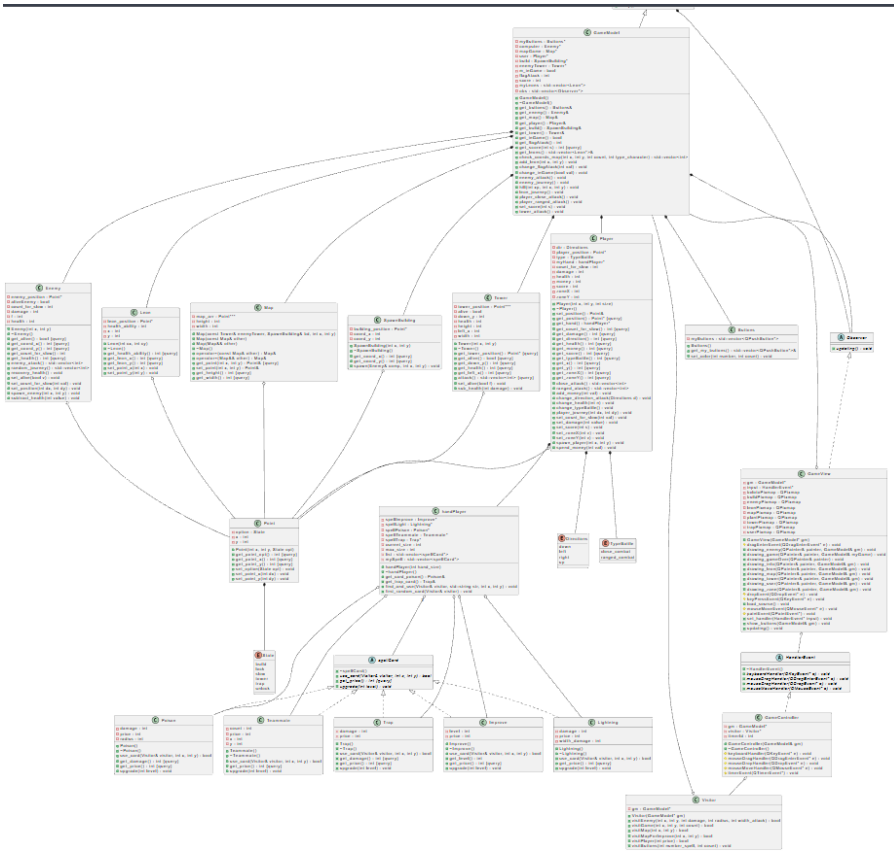

Рисунок 1

Рисунок 2

Выводы.

В ходе лабораторной работы была успешно разработана комплексная система карточных заклинаний для пошаговой игры, реализующая полиморфную архитектуру на основе единого интерфейса spellCard. Система демонстрирует эффективное применение объектно-ориентированного программирования через наследование и полиморфизм, где каждый конкретный тип заклинания (Lightning, Poison, Trap, Teammate, Improve) реализует уникальное поведение при сохранении общего контракта использования.

Ключевым достижением является реализация механизма "руки" игрока (handPlayer) с ограниченной емкостью и системой покупки карт, что добавляет стратегический элемент в геймплей. Особого внимания заслуживает система улучшений (Improve), обеспечивающая синергию между заклинаниями через накопление баффов и модификацию характеристик без явных проверок типов.

Графический интерфейс (Buttons) обеспечивает интуитивное взаимодействие с цветовой индикацией состояний карт, а реализация автономного поведения призванных существ (Leon) добавляет глубины игровому процессу. Архитектура системы позволяет легко расширять функциональность добавлением новых типов заклинаний без изменения существующего кода, что подтверждает соблюдение принципа открытости/закрытости.

Все поставленные задачи выполнены в полном объеме: создана унифицированная система заклинаний с различными типами воздействия, реализована ограниченная рука игрока с механизмом пополнения, обеспечено взаимодействие заклинаний с игровыми объектами и реализованы специализированные поведения для каждого типа магии.