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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Объектно-ориентированное программирование»
Тема: Реализация игры с использованием ООП.

Студент гр. 4384 Боков М.О..

Преподаватель Жангиров Т.Р.

Санкт-Петербург
2025

Цель работы.

Разработка игры с использованием принципов объектно-ориентированного программирования, демонстрирующей взаимодействие классов.

Задание.

1. Создать класс(ы) игры, который реализует основной цикл игры, и которому передаются команды от пользователя. Игровой цикл состоит из следующих шагов:
   a. Начало игры
   b. Запуск уровня
   c. Ход игрока. Ход, атака или применение заклинания.
   d. Ход союзников - если имеются
   e. Ход врагов
   f. Ход вражеской базы и башни - если имеются

Условие прохождения уровня студент определяет самостоятельно. Если игрок проигрывает, то игроку должно предлагаться начать заново игру, либо выйти из программы.
Все взаимодействие должно происходить через классы игры.

2. Реализовать систему сохранения и загрузки игры. Пользователь должен иметь возможность сохранить игру в любой момент. Пользователь должен иметь возможность загружаться при запуске программы (или выбрать новую), либо во время игры. Сохранения должны оставаться в консистентном состоянии между запусками игры.

3. Добавить обработку исключительных ситуаций для загрузки/сохранения, например, невозможность записать в файл, нельзя загрузиться так как файл не существует или в нем некорректные данные

Выполнение работы.

В ходе выполнения работы была написана программа на языке C++, реализующая простую игру логикой взаимодействия между игроком и

врагами. Программа также реализует игровой цикл систему сохранений и обработки ошибок

Класс Game

Назначение:

Управление основным игровым циклом, обработка главного меню, координация уровней и взаимодействие между всеми подсистемами игры.

Причины реализации:

1. Инкапсуляция логики управления игровым процессом
2. Централизация управления состоянием игры
3. Обеспечение возможности перезапуска игры после поражения

Поля:

- currentGame— текущий экземпляр игрового движка
- saveManager — менеджер сохранения/загрузки
- isRunning— флаг работы основного цикла
- justLoaded— флаг загрузки игры

Методы:

- start() — запуск основного цикла игры с главным меню
- runLevel() — выполнение игрового цикла для текущего уровня
- processMainMenuInput() — обработка выбора пользователя в главном меню
- playerTurn() — обработка хода игрока
- enemiesTurn() — обработка хода врагов
- setupLevel() — настройка начального состояния уровня
- askForRestart() — предложение перезапустить игру после поражения

Связь с классами:

text
Game — GameEngine

Управляет созданием и уничтожением игрового движка

Передает управление в игровом цикле

Game — SaveManager
Использует для сохранения и загрузки состояния игры

Класс SaveManager

Назначение:

Управление сохранением и загрузкой состояния игры.

Причины реализации:

Инкапсуляция сохранения и загрузки игры

сохранения игры

Методы:

saveGame() — сохранение всей игры в файл

loadGame() — загрузка игры из файла

saveGameState() — сохранение состояния GameEngine

loadGameState() — загрузка состояния GameEngine

saveFieldState() — сохранение состояния игрового поля

loadFieldState() — загрузка состояния игрового поля

связь с другими класами

SaveManager — GameEngine

Сохраняет и восстанавливает состояние игрока и поля

SaveManager — GameField

Работает с полем, врагами и ловушками на поле

SaveManager — EnemyTower

Сохраняет параметры башен

Обработка исключений:

- GameException — базовый класс всех игровых исключений
- SaveGameException — ошибки при сохранении (невозможность создания файла, ошибка записи)
- LoadGameException — базовый класс для ошибок загрузки
- FileNotFoundException — файл сохранения не найден
- CorruptedSaveException — файл повреждён или имеет неверный формат

Архитектура.

Архитектура построена на принципах ООП. Game выступает игровым циклом. GameEngine реализует игровую логику, GameField отвечает за состояние игрового поля, включая клетки Cell, позиции объектов Position. SaveManager реализует систему сохранения и загрузки игры. обработка исключений GameException, SaveGameException, LoadGameException, FileNotFoundException, CorruptedSaveException обеспечивает надежную обработку ошибок при операциях с файлами. Инкапсуляция защищает состояние объектов.

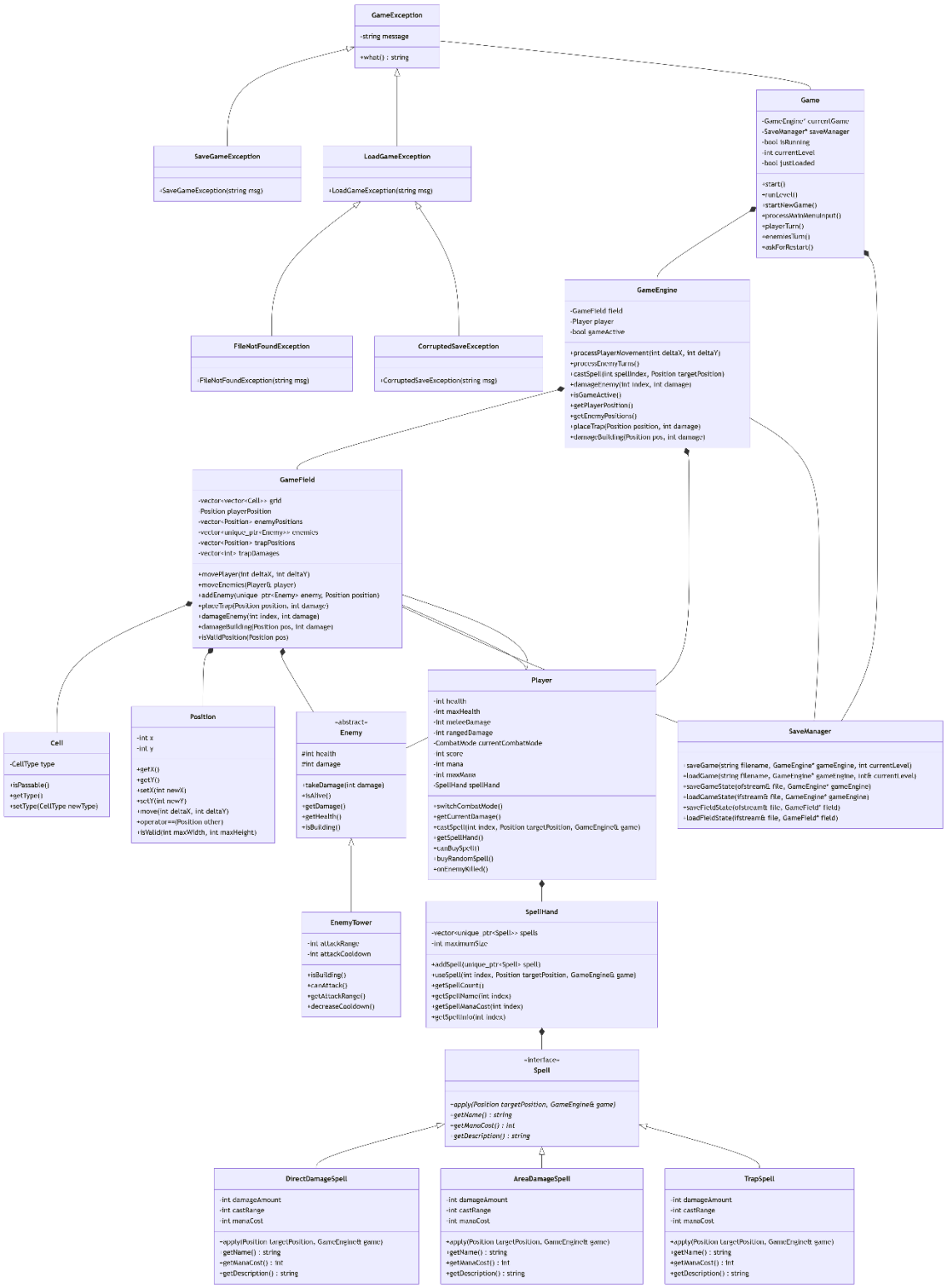

Рис. 1 UML-диаграмма классов.

Выводы.

В ходе лабораторной работы была разработана игра на языке C++, написанная с применением объектно-ориентированного программирования. В процессе выполнения были созданы и связаны между собой, каждый из которых выполняет определенную роль в архитектуре программы.

Разработанная структура демонстрирует принципы инкапсуляции, наследования и слабой связанности компонентов, что облегчает поддержку и масштабирование кода.