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»

Студент гр. 4384 _____ Водолазко В.О.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2025

Цель работы.

Изучить принципы объектно-ориентированного программирования. Написать программу на языке C++, которая будет прототипом пошаговой игры с перемещением персонажа и сражением с врагами.

Задание.

На 6/3/1 баллов:

1. Создать класс(ы) игры, который реализует основной цикл игры, и которому передаются команды от пользователя. Игровой цикл состоит из следующих шагов:

- Начало игры
- Запуск уровня
- Ход игрока. Ход, атака или применение заклинания.
- Ход союзников - если имеются
- Ход врагов
- 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.

На 8/4/1.5 баллов:

4. Реализовать переход на следующий уровень, после прохождения уровня. При переходе на следующий уровень создается новое поле другого размера с более сильными врагами. При переходе на следующий уровень, значение жизни игрока восстанавливается, и половина его карточек заклинаний случайным образом удаляется.

На 10/5/2 баллов:

5. Реализовать прокачку игрока при переходе между уровнями. Пользователь может улучшить характеристики игрока или улучшить заклинание (что и как улучшать определяет студент). Для этого нужно расширить игровой цикл.

Примечания:

- Класс игры может знать о игровых сущностях, но не наоборот
- При работе с файлом используйте идиому RAII.
- Исключения должны обязательно обрабатываться, и программа не должна завершаться
- Исключения должны быть информативными (содержать информацию о том, что и где произошло), на разные виды исключительных ситуаций должны быть свои исключения

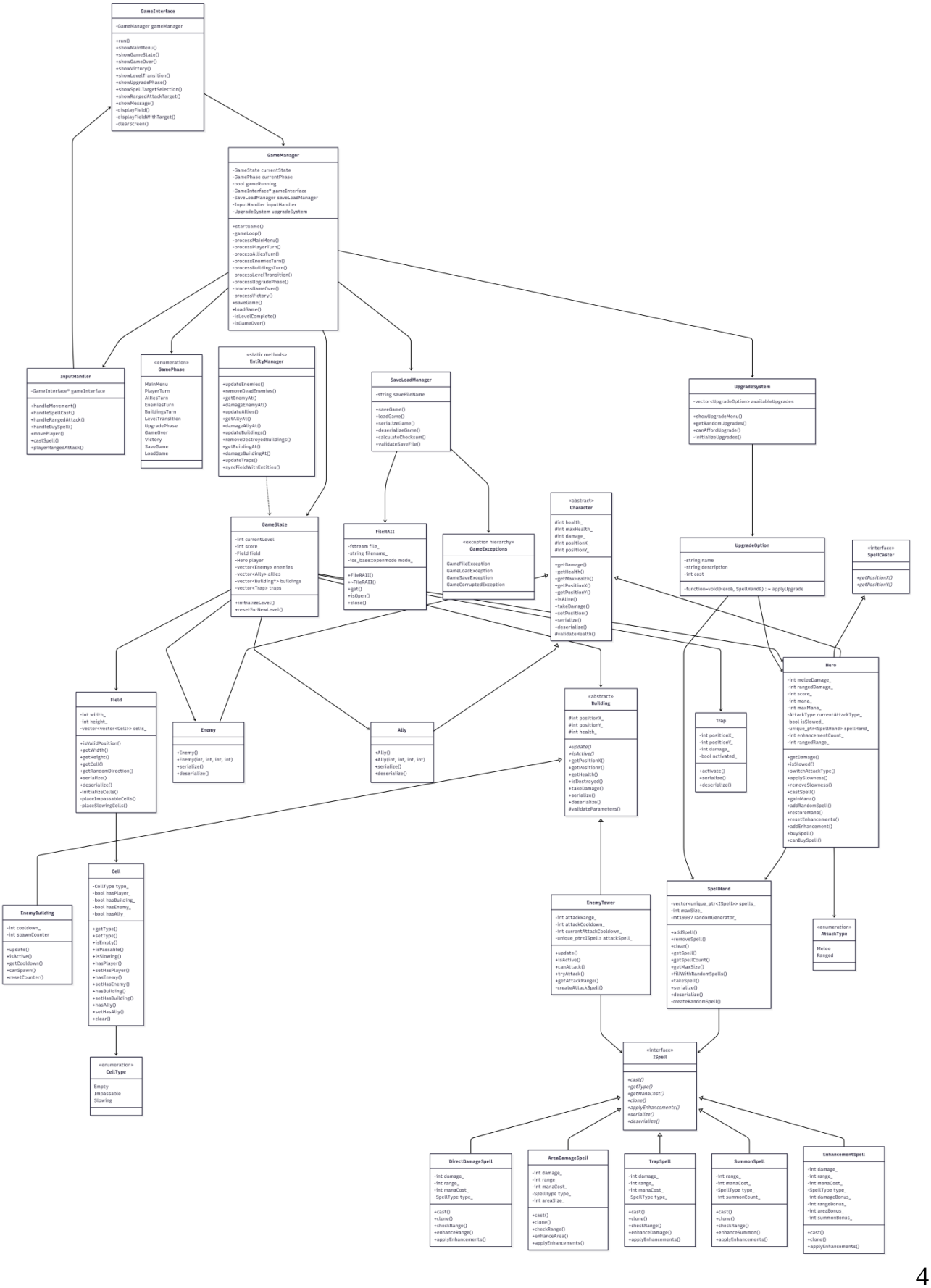
Выполнение работы.

Была реализована программа, содержащая все указанные в условие лабораторной работы классы и их поля и методы, а именно:

- Классы управления игрой;
- Сохранение/загрузка игры;
- Обработка исключений сохранения/загрузки;
- Возможность перехода на следующий уровень;
- Система прокачки игрока;

Архитектура программы.

В программе реализована иерархия классов, соответствующая принципам ООП.



Основные классы:

- GameInterface – класс отображения интерфейса
- GameManager – класс-менеджер, управляет игровым циклом
- InputHandler – класс обработки пользовательского ввода
- SaveLoadManager – класс управления сохранением/загрузкой игры
- FileRAII – класс работы с файлами по принципу RAII
- GameExceptions – класс-иерархия обработки исключений
- GameState – класс-контейнер текущего состояния игры
- UpgradeSystem – класс улучшения
- EntityManager – класс управления сущностями
- GamePhase – enum-класс хранящий фазы игры

Описание классов.

Система игры

- GameInterface

  Поля класса:

  - gameManager - уникальный указатель на менеджер игры

  Методы класса:

  - run() - запуск основного цикла интерфейса
  - showMainMenu() - отображение главного меню
  - showGameState() - отображение текущего состояния игры
  - showGameOver() - отображение экрана поражения
  - showVictory() - отображение экрана победы
  - showLevelTransition() - отображение перехода между уровнями
  - showUpgradePhase() - отображение фазы улучшений
  - showSpellTargetSelection() - отображение выбора цели для заклинания

- showRangedAttackTarget() - отображение выбора цели для дальнобойной атаки
- showMessage() - отображение сообщения пользователю
- clearScreen() - очистка экрана

- GameManager

  Поля класса:

  - currentState - текущее состояние игры
  - currentPhase - текущая фаза игры
  - gameRunning - флаг выполнения игрового цикла
  - gameInterface - указатель на интерфейс игры
  - saveLoadManager - менеджер сохранения/загрузки
  - inputHandler - обработчик ввода
  - upgradeSystem - система улучшений

  Методы класса:

  startGame() - запуск игры

  gameLoop() - основной игровой цикл

  processMainMenu() - обработка главного меню

  processPlayerTurn() - обработка хода игрока

  processAlliesTurn() - обработка хода союзников

  processEnemiesTurn() - обработка хода врагов

  processBuildingsTurn() - обработка хода зданий

  processLevelTransition() - обработка перехода уровня

  processUpgradePhase() - обработка фазы улучшений

  processGameOver() - обработка завершения игры

  processVictory() - обработка победы

  saveGame() - сохранение игры

  loadGame() - загрузка игры

  isLevelComplete() - проверка завершения уровня

  isGameOver() - проверка завершения игры

- InputHandler

Поля класса:

- gameInterface - указатель на игровой интерфейс

Методы класса:

- handleMovement() - обработка движения и основных команд
- handleSpellCast() - обработка применения заклинаний
- handleRangedAttack() - обработка дальнобойной атаки
- handleBuySpell() - обработка покупки заклинаний
- movePlayer() - перемещение игрока
- castSpell() - применение заклинания
- playerRangedAttack() - выполнение дальнобойной атаки

- SaveLoadManager

  Поля класса:

  - saveFileName - имя файла для сохранения

  Методы класса:

  - saveGame() - сохранение игры в файл
  - loadGame() - загрузка игры из файла
  - serializeGame() - сериализация состояния игры
  - deserializeGame() - десериализация состояния игры
  - calculateChecksum() - расчет контрольной суммы
  - validateSaveFile() - проверка валидности файла сохранения

- FileRAII

  Поля класса:

  - file_ - файловый поток
  - filename_ - имя файла
  - mode_ - режим открытия файла

  Методы класса:

  - FileRAII() - конструктор с открытием файла
  - ~FileRAII() - деструктор с автоматическим закрытием файла
  - get() - получение ссылки на файловый поток
  - isOpen() - проверка открыт ли файл

- close() - закрытие файла

- GameExceptions

  Классы исключений:

  - GameFileException - базовое исключение для файловых операций
  - GameLoadException - исключение при загрузке игры
  - GameSaveException - исключение при сохранении игры
  - GameCorruptedException - исключение при повреждении данных

- GameState

  Поля класса:

  - currentLevel - текущий уровень игры
  - score - общий счет
  - field - игровое поле
  - player - объект героя
  - enemies - вектор врагов
  - allies - вектор союзников
  - buildings - вектор зданий
  - traps - вектор ловушек

  Методы класса:

  - resetForNewLevel() - сброс состояния для нового уровня
  - createLevel() - создание уровня

- UpgradeSystem

  Поля класса:

  - availableUpgrades - вектор доступных улучшений

  Методы класса:

  - showUpgradeMenu() - отображение меню улучшений
  - getRandomUpgrades() - получение случайных улучшений
  - canAffordUpgrade() - проверка возможности улучшения
  - initializeUpgrades() - инициализация списка улучшений

- EntityManager

Методы класса:

- updateEnemies() - обновление состояния врагов
- removeDeadEnemies() - удаление мертвых врагов
- getEnemyAt() - получение врага в позиции
- damageEnemyAt() - нанесение урона врагу в позиции
- updateAllies() - обновление состояния союзников
- getAllyAt() - получение союзника в позиции
- damageAllyAt() - нанесение урона союзнику в позиции
- updateBuildings() - обновление состояния зданий
- removeDestroyedBuildings() - удаление разрушенных зданий
- getBuildingAt() - получение здания в позиции
- damageBuildingAt() - нанесение урона зданию в позиции
- updateTraps() - обновление состояния ловушек
- syncFieldWithEntities() - синхронизация поля с сущностями

- GamePhase

  Значения:

  - MainMenu - главное меню
  - PlayerTurn - ход игрока
  - AlliesTurn - ход союзников
  - EnemiesTurn - ход врагов
  - BuildingsTurn - ход зданий
  - LevelTransition - переход между уровнями
  - UpgradePhase - фаза улучшений
  - GameOver - завершение игры
  - Victory - победа
  - SaveGame - сохранение игры
  - LoadGame - загрузка игры

Выводы.

Была изучена парадигма объектно-ориентированного программирования. Была реализована программа на языке C++ содержащая основные классы игры с необходимыми полями и методами.