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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Объектно-ориентированное программирование»

Тема:

Студентка гр. 4382 _____ Жданова К.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2025

Цель работы:

Разработка игрового приложения. Создание системы взаимодействующих классов для управления игровыми сущностями и игровым полем. Построение UML-диаграммы, отображающей структуру классов, их взаимосвязи и архитектурные решения.

Задание.

На 6/3/1 баллов:

Создать класс игрока, который должен хранить информацию об игроке (его жизни, урон, очки, и т. д. - студент сам определяет необходимые для работы характеристики). Объект класса игрока должен перемещаться по карте. Если у игрока кончаются жизни, то происходит конец игры.

Создать класс врага, который хранит параметры жизней и урона. Объектами класса врага управляет компьютер. При перемещении, если враг пытается перейти на клетку с игроком, то перемещение не происходит, и игроку наносится урон.

Создать класс квадратного/прямоугольного игрового поля, по которому перемещаются игрок и враги. Игровое поле не должно быть меньше 10 на 10 клеток, и не больше 25 на 25 клеток. Размеры поля задаются через конструктор. Рекомендуется для хранения информации об отдельных клетках поля создать отдельный класс.

Реализовать конструкторы перемещения и копирования для поля, а также соответствующие операторы присваивания с копированием и перемещением (должна происходить глубокая копия).

На 8/4/1.5 баллов:

Реализовать непроходимые клетки на поле. При попытке врагов или игрока перейти на такую клетку, перемещение не происходит. Заполнения поля непроходимыми клетками происходит в момент создания поля.

Добавить возможность для игрока переключаться на ближний или дальний бой с изменением значения наносимого урона. Такое переключение требует один ход.

На 10/5/2 баллов:

Добавить класс вражеского здания. Такое здание размещается на карте, и раз в несколько ходов создает нового врага возле себя. Количество ходов до создания нового врага задается в конструкторе.

Реализовать замедляющие клетки на поле. Если игрок переходит на такую клетку, то он он не может двигаться на следующий ход.

Основные теоретические положения.

Класс - единица абстракции. Универсальный тип данных, являющейся информационной моделью.

Объектно-ориентированное программирование — это парадигма программирования, при которой программа строится как система взаимодействующих объектов. Каждый объект объединяет в себе данные (атрибуты) и код (методы) для работы с ними, а также определяет, как он будет взаимодействовать с другими объектами.

Объект - экземпляр класса, хранящийся в памяти.

Модификаторы доступа - механизм, определяющий доступность полей и методов только в классе или извне.

Инвариант класса – набор утверждений, которые должны быть истинны применительно к любому объекту данного класса в любой момент времени, за исключением переходных процессов в методах объекта.

UML — язык графического описания для объектного моделирования в области разработки программного обеспечения,

Resource Acquisition Is Initialization - идиома объектно-ориентированного программирования, основная идея которой совместить получение ресурса с его инициализацией, а освобождение - с уничтожением объекта.

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.

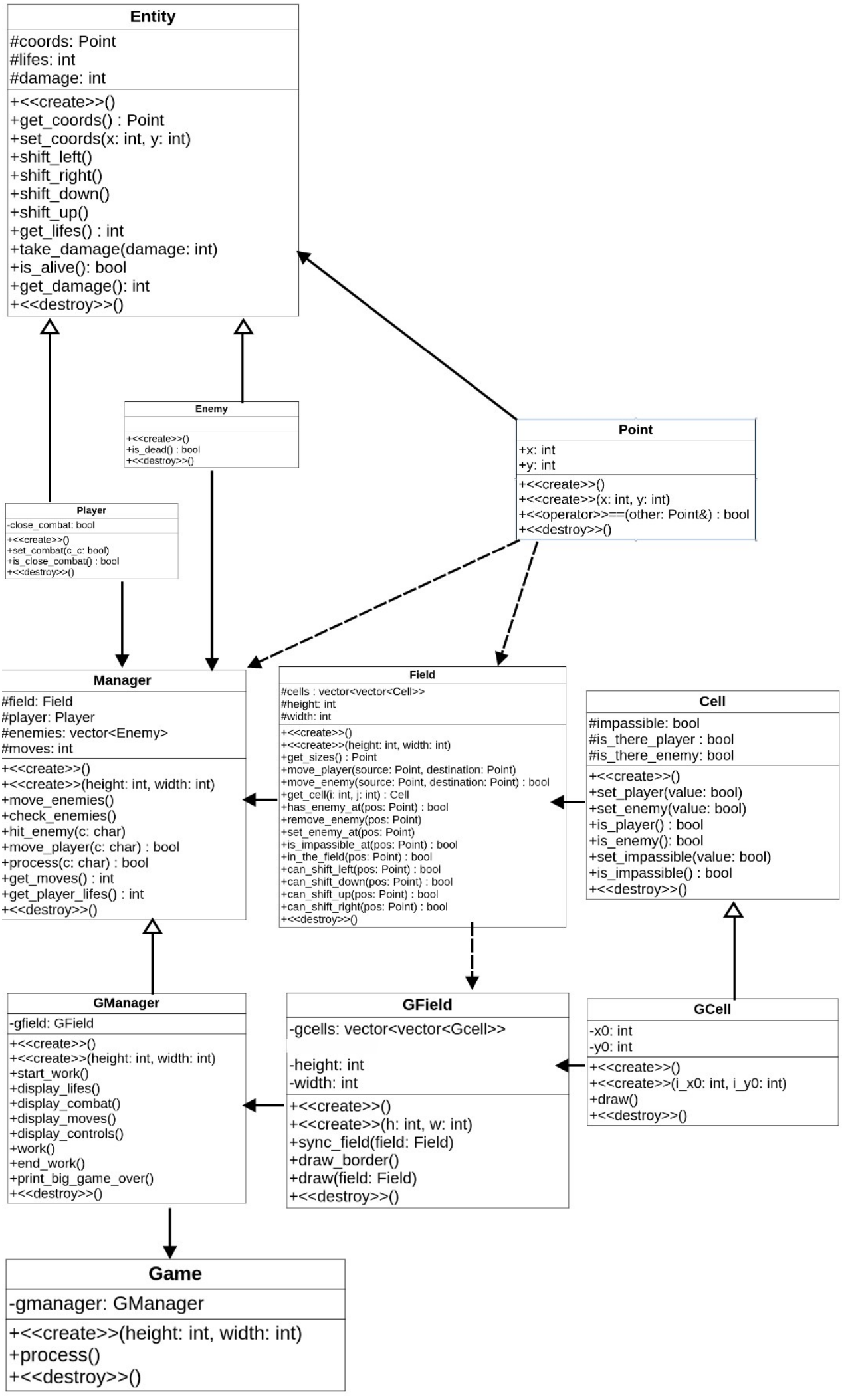

Рисунок 1. UML-диаграмма

Для разделения функционала классы были поделенны на группы — те, кто отвечает за логику и те, кто отвечает за отрисовку.

Для реализации пользовательского интерфейса использовалась библиотека ncurses.

Логическая часть может работать без графической. Графическая использует логическую часть как основу.

Класс Game запускает отрисовку игры с помощью класса GManager. GManager отвечает за вывод сопутствующей информации (количество жизней, номер текущего хода, значение урона на данный момент, инструкцию по управлению), и за отрисовку поля.

Отрисовка поля происходит с помощью класса GField , который при каждом нажатии клавиши синхронизируется с логическим полем Field и отрисовывает клетки поля. Gfield не является наследником класса Field, а лишь использует его для синхронизации.

В GField находится двумерный массив типом класса GCell . Класс GCell является наследником класса Cell (ячейка поля). GCell использует данные Cell каждой ячейки логического поля для отрисовки соответствующего расположения игрока, врагов и непроходимых клеток.

Класс Cell — логическая ячейка поля. Для каждой ячейки необходимо знать, находится ли в ней сейчас игрок, враг или это непроходимая клетка. Эта информация используется для перемещения врагов и игрока.

В классе Field находится двумерный массив типом класса Cell. Это логическое игровое поле, по которому перемещаются игрок и враги. Класс Field имеет множество методов для получения информации о ячейках поля.

Класс Field, в свою очередь, является полем класса Manager - который управляет игрой. Он обрабатывает каждое нажатие клавиши и каждый ход перемещает врагов, игрока, увеличивает ходы. Главный игровой метод класса Manager — метод bool process(char c). Он возвращает значение false, если при очередном перемещении игрока или врагов игрок потерял свою последнюю жизнь. Игра продолжается, пока process возвращает значение true.

Классы Player и Enemy также являются полями класса Manager. Они оба являются наследниками класса Entity. Они отличаются установлением урона и количества жизней при создании. Также у класса Player есть поле
bool close_combat, который отображает, установлен сейчас ближний бой или дальний.
Класс Entity имеет поле Point coords, которое отображает текущие координаты сущности. Класс Entity имеет множество методов для работы с координатами.
Для удобного использования координат был создан класс Point, который хранит в себе два значения — координаты по оси X и по оси Y.

Каждый класс описан в своих файлах .cpp и .h. Файлы были разделены по папкам в соответствии с разделением классов на логическую и графическую часть, логическая часть в свою очередь была разделена на группы с сущностями и управлением. Итоговое файловое дерево проекта представлен на Рисунке 2.

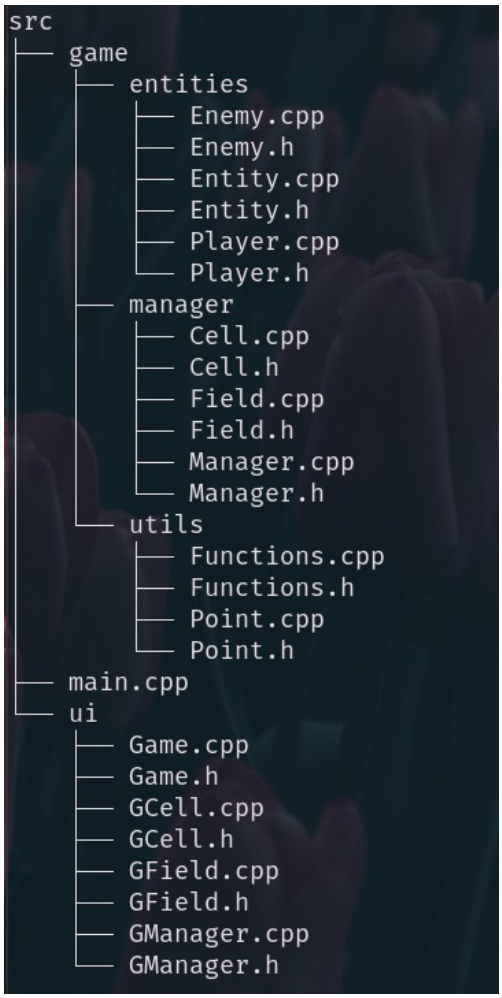

Рисунок 2.

Для сборки проекта был написан CMakeLists.txt.

Итоговый исполняемый файл называется my_game и находится в папке build. Чтобы поменять размеры игрового поля, необходимо поменять значения в вызове конструктора Game в файле main.cpp.

Непроходимые клетки генерируются псевдорандомной функцией rand, и их значения зависят от размеров поля. Так что при создании поля одного и того же размера непроходимые клетки будут находится на тех же местах.

На момент завершения Лабораторной работы №1 игровое приложение имеет следующий вид, представленный на Рисунке 3.

Рисунок 3.

Выводы:

Было разработано консольное игровое приложение на основе системы взаимодействующих классов для управления игровыми сущностями и игровым полем. Была построена UML-диаграмма, отображающая структуру классов, их взаимосвязи и архитектурные решения.