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: Создание классов, конструкторов и методов

Студент гр. 1304 _____ Павлов Д.Р.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы.

Реализовать прямоугольное игровое поле, состоящее из клеток. Клетка - элемент поля, которая может быть проходима или нет (определяет, куда может стать игрок), а также содержит какое-либо событие, которое срабатывает, когда игрок становится на клетку. Для игрового поля при создании должна быть возможность установить размер (количество клеток по вертикали и горизонтали). Игровое поле должно быть зациклено по вертикали и горизонтали, то есть если игрок находится на правой границе и идет вправо, то он оказывается на левой границе (аналогично для всех краев поля).

Реализовать класс игрока. Игрок – сущность, контролируемая пользователем. Игрок должен иметь свой набор характеристик и различный набор действий (например, разные способы перемещения, попытка избежать событие, и так далее).

Требования.

Реализован класс игрового поля

Для игрового поля реализован конструктор с возможностью задать размер и конструктор по умолчанию (то есть конструктор, который можно вызвать без аргументов).

Реализован класс интерфейс события (в данной лабораторной это может быть пустой абстрактный класс).

Реализован класс клетки с конструктором, позволяющим задать ей начальные параметры.

Для клетки реализованы методы реагирования на то, что игрок перешел на клетку.

Для клетки реализованы методы, позволяющие заменять событие. (То есть клетка в ходе игры может динамически меняться).

Реализованы конструкторы копирования и перемещения, и соответствующие им операторы присваивания для игрового поля и при необходимости клетки.

Реализован класс игрока минимум с 3 характеристиками. И соответствующие ему конструкторы.

Реализовано перемещение игрока по полю с проверкой допустимости на переход по клеткам.

Описание архитектурных решений и классов.

В данной лабораторной работе были ***реализовано три паттерна***: Singleton, Facade, Mediator.

Описание классов:

1) ***Facade_Game*** — Класс Фасада. Фасад - это структурный паттерн, который предоставляет простой интерфейс к сложной системе объектов, библиотеке или фреймворку.

Директория Singleton (Хранит .cpp и .h файлы Класса Героя-Одиночки)**:**

2) ***Singleton_Hero*** — Класс Одиночки, в роли которого выступает Герой, управляемый Юзером. Герой так же хранит в себе 3 атрибута: хп, опыт, уровень. Было решено сделать именно этот класс Singleton'ом, так как статичность Героя позволит проще взаимодействовать с другими классами (Например: Классы Событий, напрямую изменяющие некоторые атрибуты Героя). Данный Класс включает в себя набор методов, меняющих и возвращающих его текущие состояния, а так же характерный для Паттерна Singleton метод getInstance, который возвращает самого героя. Данный паттерн гарантирует наличие только одного экземпляра класса.

Директория Mediator (отвечает за Класс медиатора и других классов, которых использует медиатор)**:**

3) ***Mediator*** — Класс Медиатора. Выполняет функцию взаимосвязи классов, отвечающих за действия Юзера.

4) ***Hero_Moves*** — Класс, отвечающий за движения Героя на игровом поле.

Директория Objects Generator (отвечает за создание объектов)**:**

5) ***Objects_Generator*** — Класс, отвечающий за создание объектов на всем игровом поле.

Директория Field (отвечает за данные игрового поля)**:**

6) ***Field*** — Класс всего поля, хранящий в себе значения длины и ширины, которое опционально может быть задано Юзером, а так же 2Д вектор, из классов Cell.

7) ***Cell*** — Класс клетки. Хранит в себе текущее состояние клетки (Клетка-Герой, Пустая Клетка, Клетка-Стена и т.д.) и тип события: NO_EVENT — клетка хранит в себе либо Героя, либо Стену, либо является Пустой; POSITIVE_EVENT — клетка будет генерировать положительное событие; и так далее…

8) ***Space_Request*** — Класс, отвечающий за запрос Юзеру указать значения длины и ширины игрового поля. При ввода некорректных значений (Например: Символы, Отрицательное значение …), длина или ширина задается дефолтной.

Директория Display (отвечает за визуальное отображение игры)**:**

9) ***Attribute_Display*** — Класс, отвечающий за отрисовку Атрибутов Героя.

10) ***Intro*** — Класс, отвечающий за отрисовку Интро Игры.

11) ***Main_Menu*** — Класс, отвечающий за отрисовку Главного Меню.

12) ***Field_Painter*** — Класс, отвечающий за отрисовку игрового поля

Демонстрация работы программы и тестирование.

При запуске программы запускается отрисовка интро с названием самой игры. (Рисунок 1)

Рисунок 1. - Отрисовка интро.

Далее перед Юзером открывается Главное Меню, в котором можно выбрать: Новая Игра, Загрузки (Лаб. Работа 5-6), Выход. Для перемещения следует нажимать клавиши вверх(w) и вниз(s), а для выбора — Enter. На Рисунке 2 продемонстрировано отображение Главного Меню.

Рисунок 2. - Главное Меню.

На данном этапе разработки, работают только кнопки «New Game» и «Quit» При нажатии на «New Game», генерируется Игровое Поле. Игровое поле с дефолтными значениями продемонстрировано на Рисунке 3.

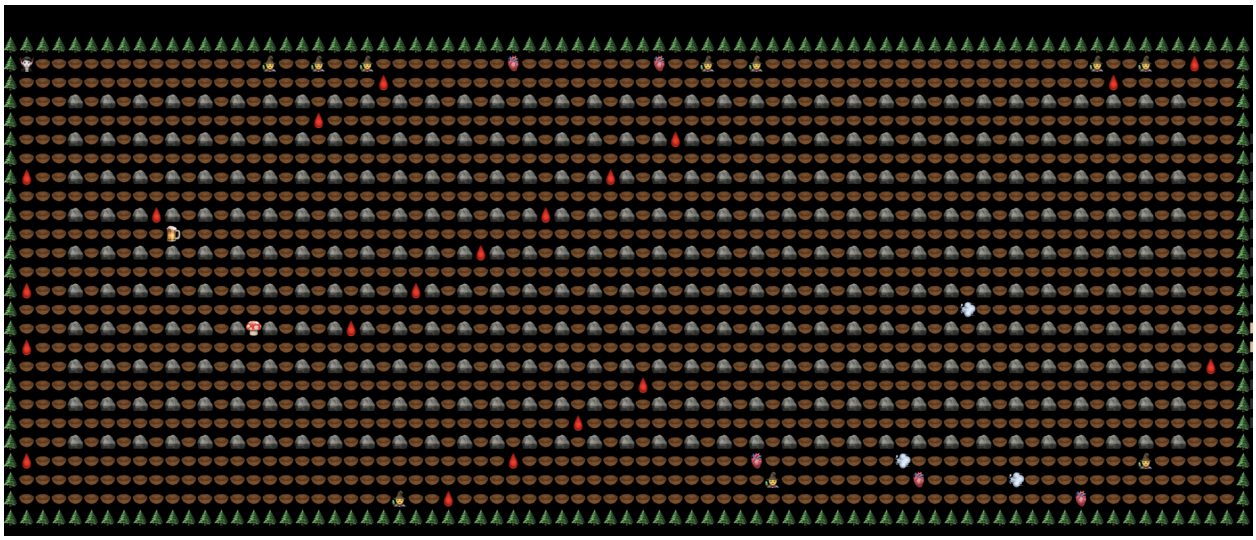


Рисунок 3. - Игровое Поле.

Для передвижения Героя используется классическая раскладка WASD. При движении игрока на любую грань происходит перестановка Героя на край параллельной грани. (Рисунок 4)

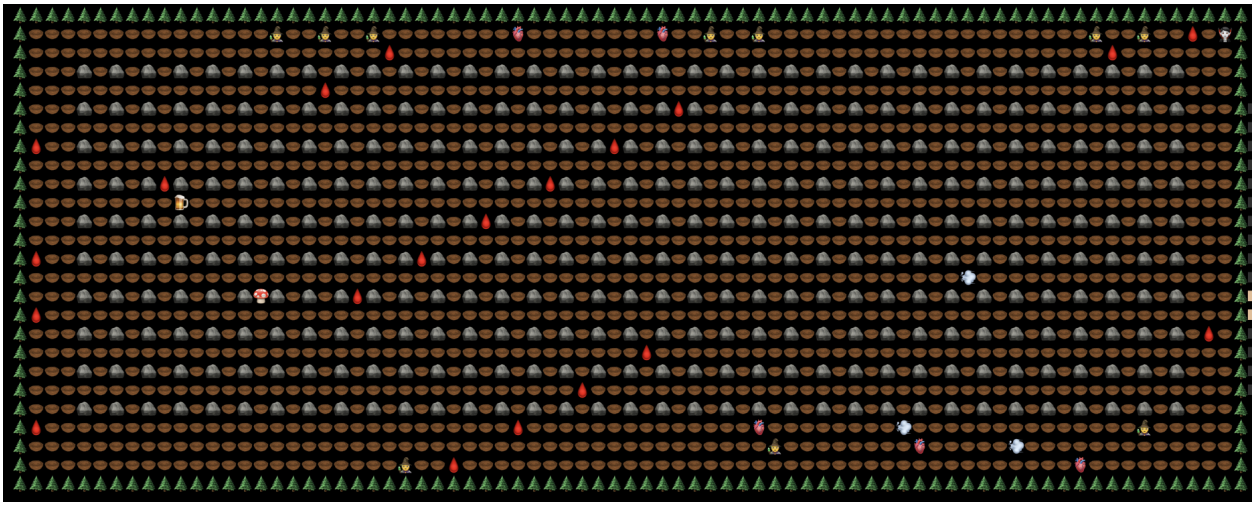


Рисунок 4. - Юзер приказал Герою перейти на Левую Границу, тем самым Герой переместился на край Правой Границы.

На Игровом Поле нельзя наступать на «Стены», нарисованные камнями.

Так же рисуются и сами атрибуты Героя. (Рисунок 5)

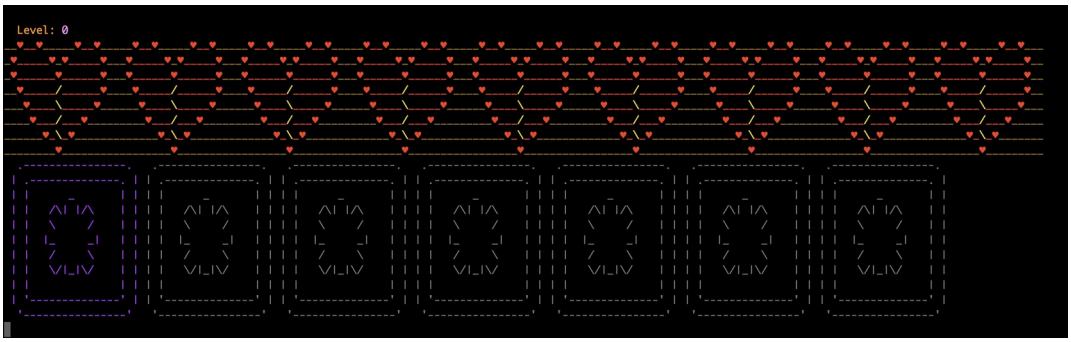
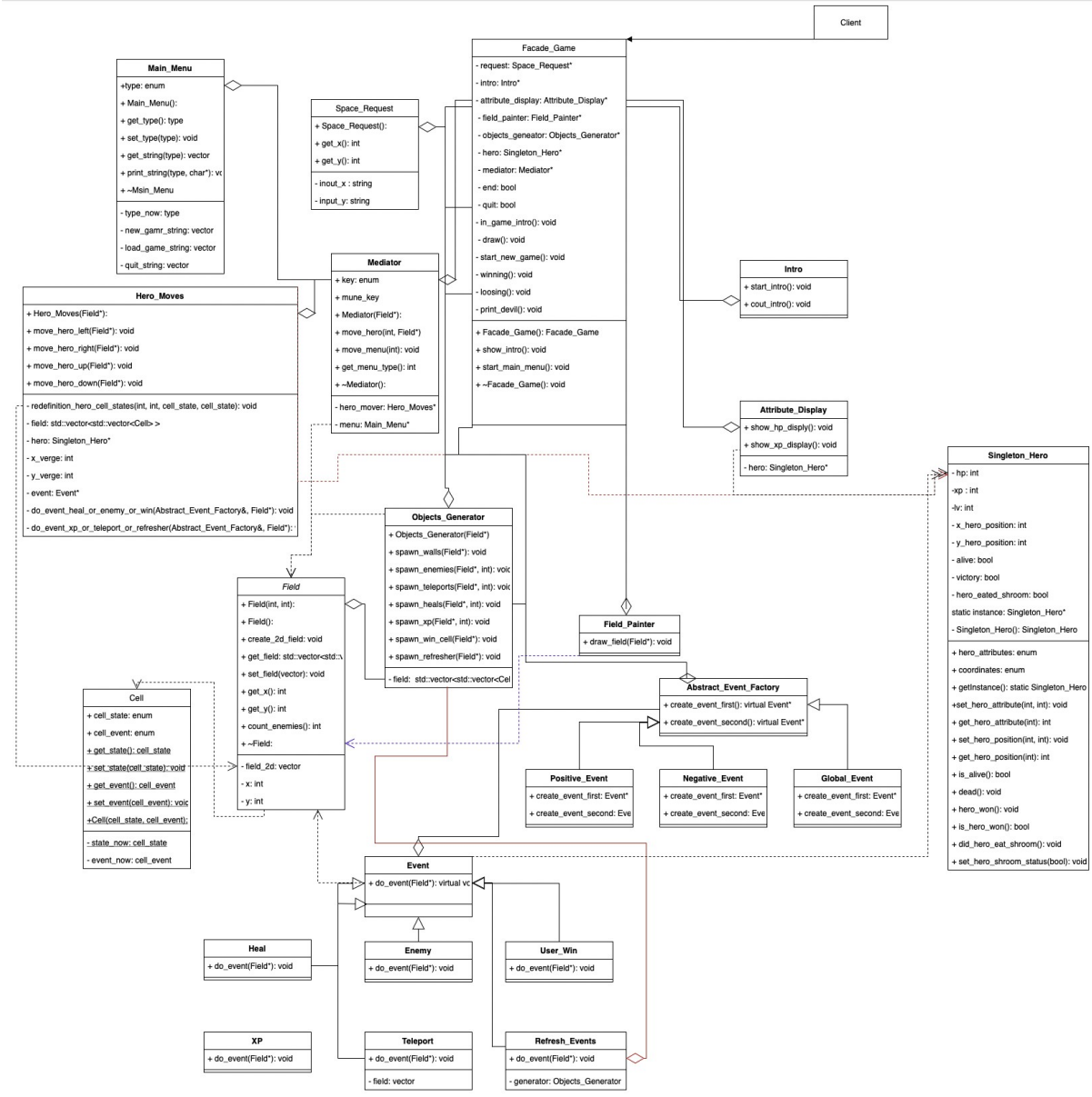


Рисунок 5. - Панель Атрибутов Героя. (В самом низу опыт Героя).

UML диаграмма проекта:



Вывод.

Реализовано прямоугольное игровое поле, состоящее из клеток. Игровое поле зациклено, по нему может передвигаться игрок с помощью клавиш-стрелочек. Клетки имеют свойства проходимости, проверку на наличие игрока на них.

Была изучена работа с классами на языке C++, паттерны проектирования, основы составления UML-диаграмм.