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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Объектно-ориентированное программирование»

Тема: Шаблонные классы, генерация карты

Студент гр. 1304 _____ Павлов Д.Р.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы.

Реализовать шаблонный класс генерирующий игровое поле. Данный класс должен параметризироваться правилами генерации (расстановка непроходимых клеток, как и в каком количестве размещаются события, расположение стартовой позиции игрока и выхода, условия победы, и.т.д.). Также реализовать набор шаблонных правил (например, событие встречи с врагом размещается случайно в заданном в шаблоне параметре, отвечающим за количество событий)

Требования.

Реализован шаблонный класс генератор поля. Данный класс должен поддерживать любое количество правил, то есть должен быть variadic template.

Реализовано не менее 6 шаблонных классов правил

При запуске программы есть возможность выбрать уровень (не менее 2) из заранее заготовленных шаблонов

Описание архитектурных решений и классов.

В данной лабораторной работе использовался паттерн ***Strategy.***

1) ***Level_Generator*** — Шаблонный класс генератора уровня, поддерживающий любое количество правил (другими словами, имеет переменное количество параметров). Вместе с этим Класс имеет 3 (+2 в 8 лабе) конкретных параметра шаблона. Они отвечают за уникальные игровые правила спавна (Например: Спавн героя, спавн победной клетки). Было решено сделать их отдельными параметрами шаблона, поскольку по своей логике каждый параметр — какое-то правило; но с уникальными правилами работать куда проще, если они будут независимы от других правил. Данный класс имеет 4 (+2 в 8 лабе) метода: ***hero_spawn(****int x, int y, T1* hero, Field* field****)*** — спавнит героя на поле; ***win_cell_spawn(****int x, int y, T2* win_cell, Field* field****)*** — спавнит победную клетку на поле; ***walls_spawn(****int magic_number, T3* walls, Field* field****)*** — спавнит стены на поле использую алгоритм с неким параметром *magic_number*, значение которого нам неважно, он нужен исключительно для расстановки позиций клеток-стен; ***set_rules(****Types*... args, int magic_number, int amount, Field* field****)*** — осуществляет распаковку переменных параметров типа *Types**, по своей логике, метод, смотря по параметрам magic_number (делает то же что и с *walls*) и *amount* (количество) расставляет на поле Ивенты. Поскольку данный класс Шаблонный, мы не знаем точно какого типа данные имеют некоторые классы (why?), поэтому мы будем перегружать операторы вызова функции у каждых классов-правил, даб.

Правила:

2) ***Rule_Hero_Spawn*** — Шаблонный Класс (Функтор) Правила, задающий точку спавна Героя на поле, путем переопределения оператора вызова функции.

3) ***Rule_Win_Cell_Spawn*** - Шаблонный Класс (Функтор) Правила, задающий позицию победной клетки, путем переопределения оператора вызова функции.

4) ***Rule_Walls_Spawn*** - Шаблонный Класс (Функтор) Правила, создающий на карте стены, путем переопределения оператора вызова функции.

5) ***Rule_Heals_Spawn*** - Шаблонный Класс (Функтор) Правила, создающий на карте хилки, путем переопределения оператора вызова функции.

6) ***Rule_XPs_Spawn*** - Шаблонный Класс (Функтор) Правила, создающий на карте клетки опыта, путем переопределения оператора вызова функции.

7) ***Rule_Enemies_Spawn*** - Шаблонный Класс (Функтор) Правила, создающий на карте клетки врагов, путем переопределения оператора вызова функции.

8) ***Rule_Teleports_Spawn*** - Шаблонный Класс (Функтор) Правила, создающий на карте клетки телепортов, путем переопределения оператора вызова функции.

9) ***Rule_Refresher_Spawn*** - Шаблонный Класс (Функтор) Правила, создающий на карте грибы, путем переопределения оператора вызова функции.

*В Оператор Вызова Функций у Правил от ***Rule_Walls_Spawn*** до ***Rule_Refresher_Spawn*** принимается некий аргумент *magic_number*, который псевдорандомно задает положение клеток.

Strategy:

*В данном контексте, Стратегия — Алгоритм создания уровней по определенным правилам.

10) ***Level_Strategy*** — Интерфейс Стратегии. Интерфейс Стратегии объявляет операции, общие для всех поддерживаемых версий некоторого алгоритма. Контекст использует этот интерфейс для вызова алгоритма, определённого Конкретными Стратегиями.

11) ***Level_Context*** — Класс, который работает с нужной стратегией, и, при возможности, меняет ее. Контекст определяет интерфейс, представляющий интерес для клиентов. Контекст хранит ссылку на один из объектов Стратегии. Контекст не знает конкретного класса стратегии. Он должен работать со всеми стратегиями через интерфейс Стратегии.

12) ***Level_One*** — Класс-наследник от ***Level_Strategy***. Определяет алгоритм создания первого уровня.

13) ***Level_Two*** — Класс-наследник от ***Level_Strategy***. Определяет алгоритм создания второго уровня.

Демонстрация работы программы и тестирование.

Рисунок 1 — Второй Уровень.

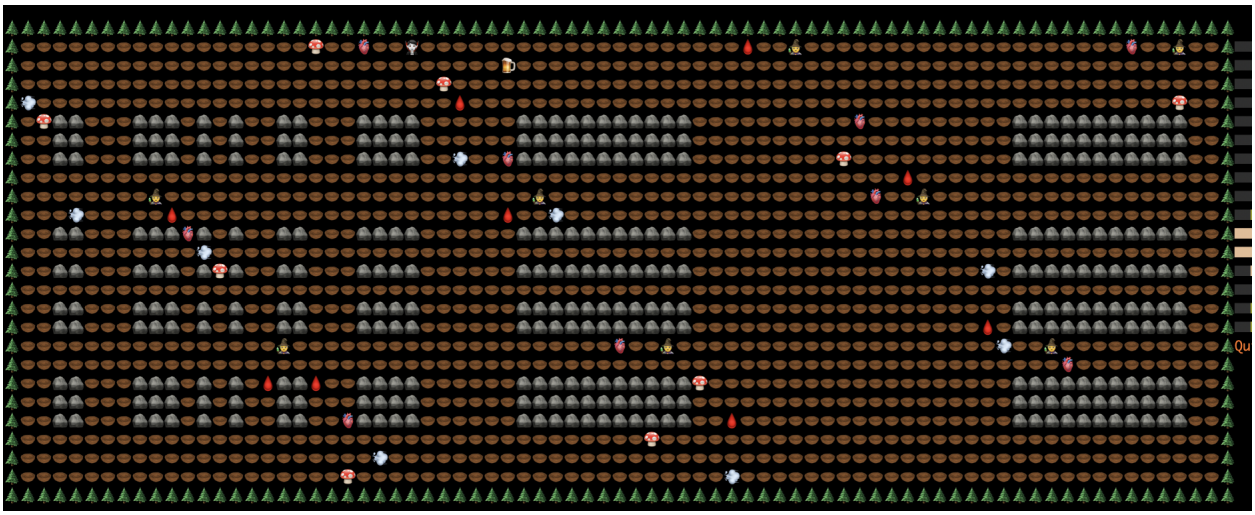


Рисунок 2 — Первый уровень.

По Рисунку 1 и 2 видно различие уровней.

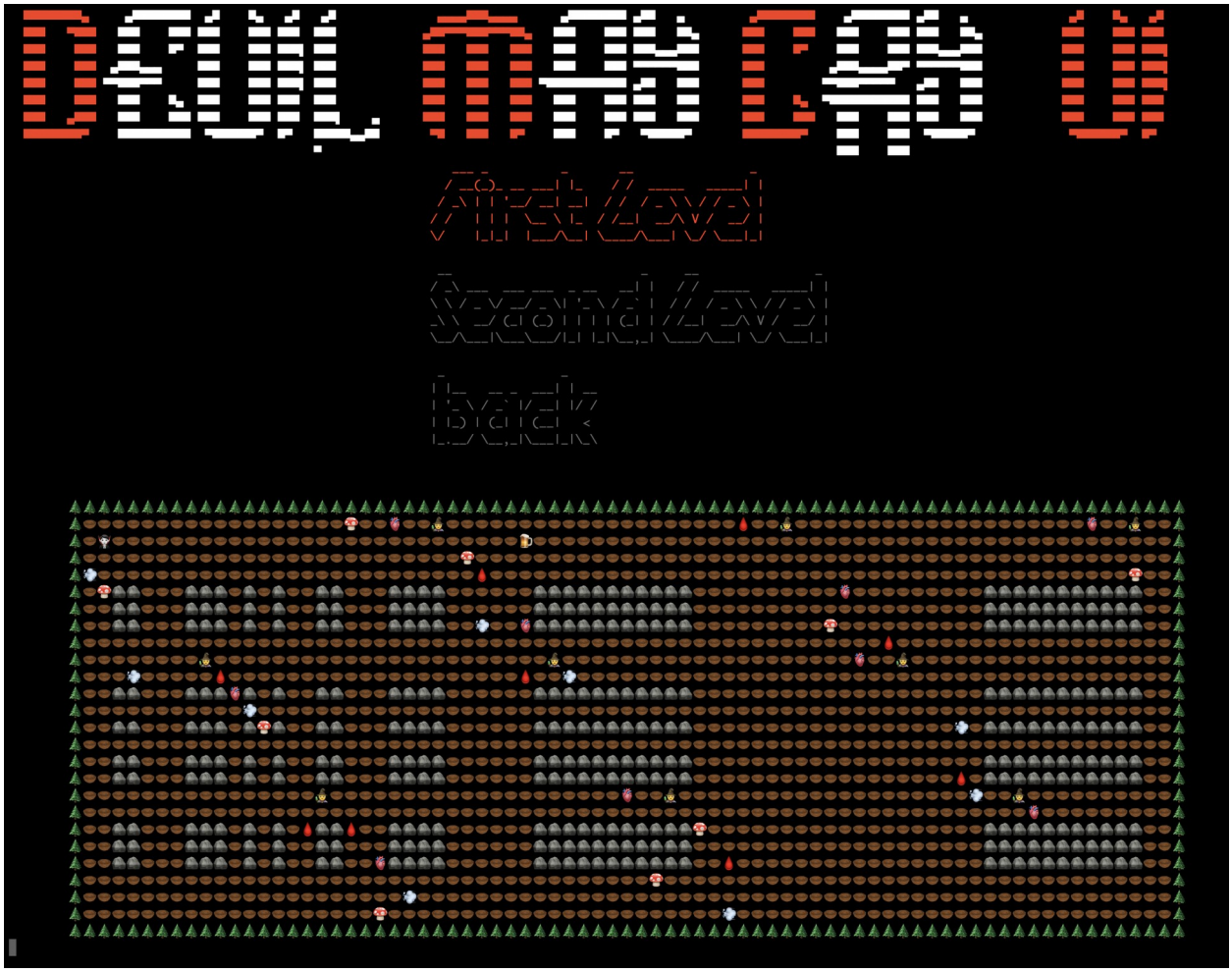
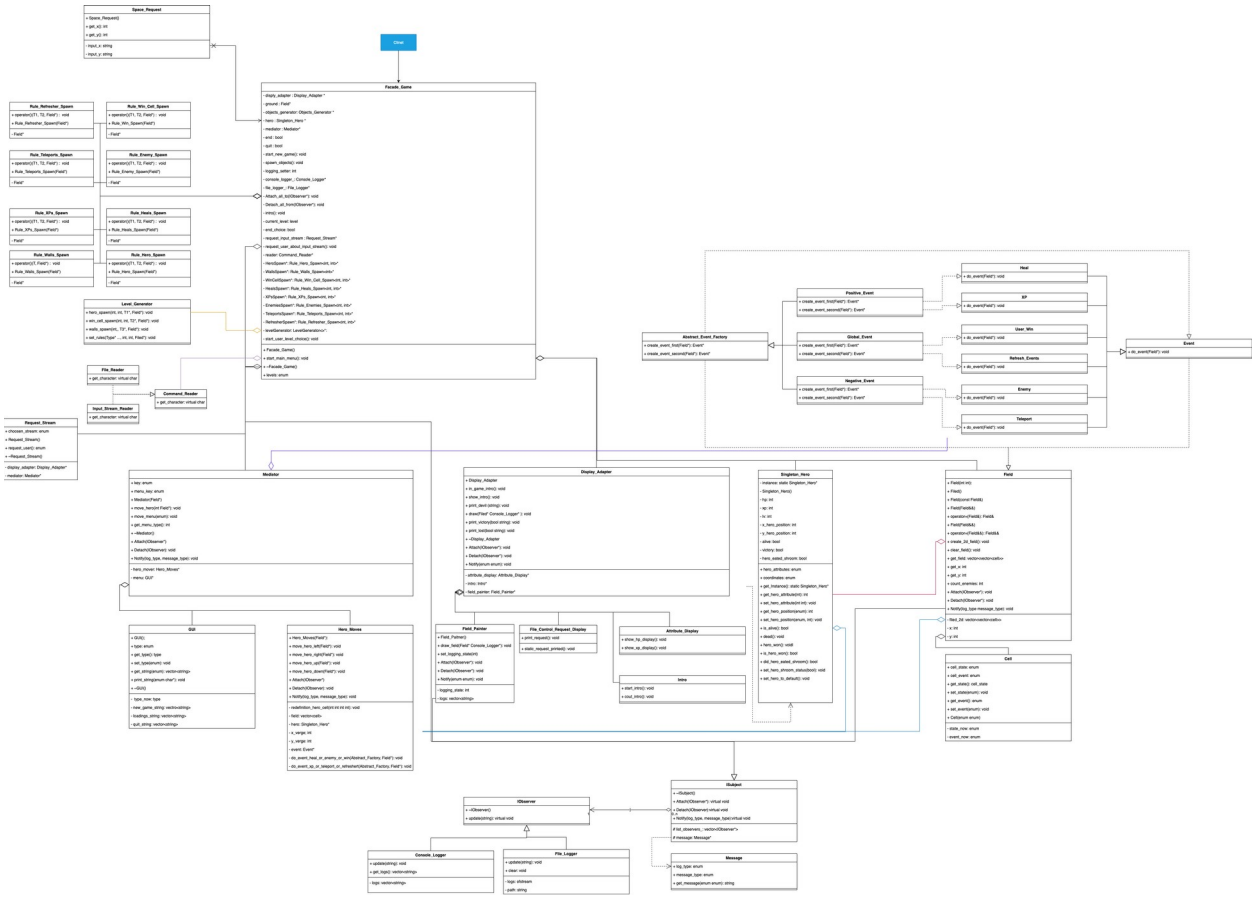


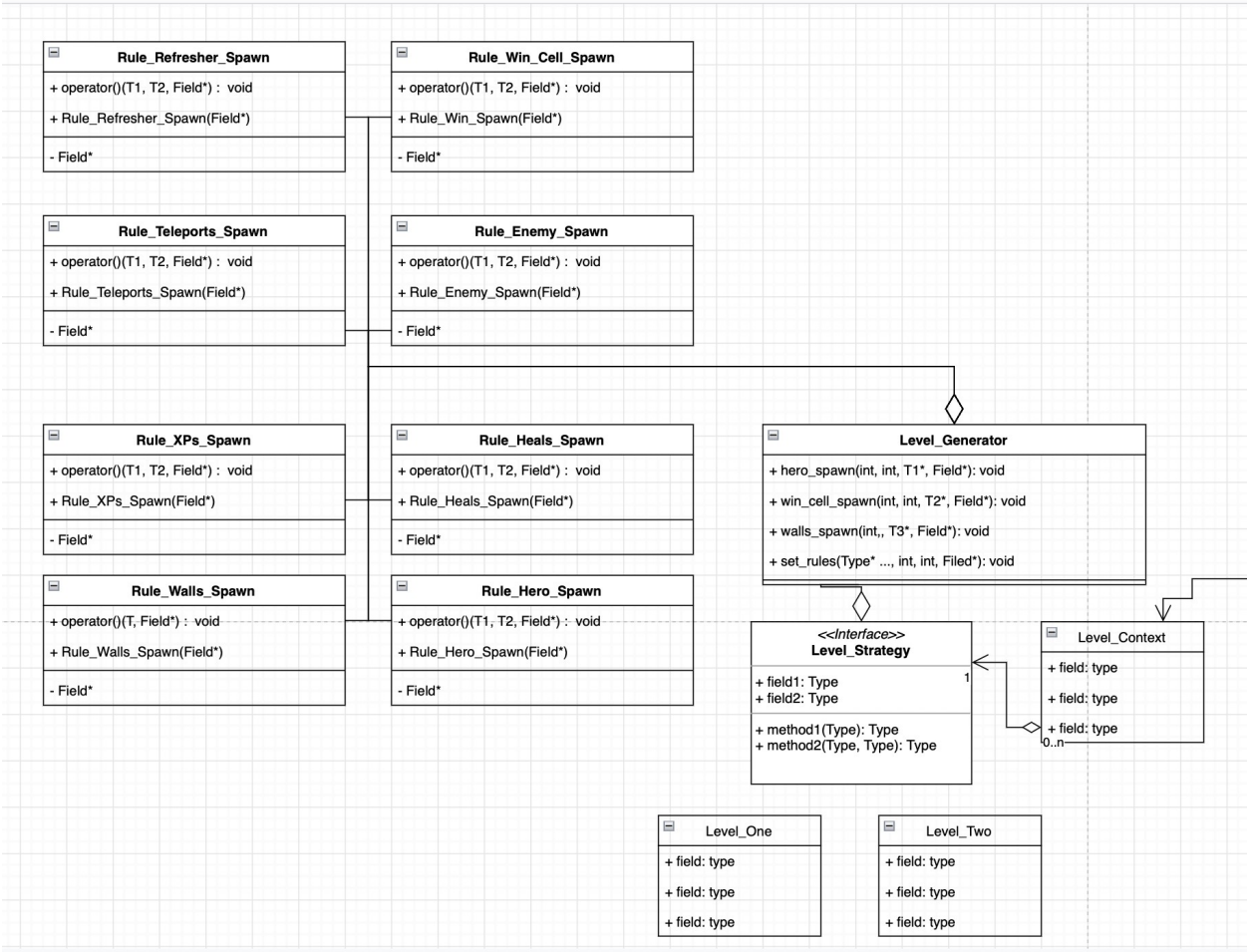
Рисунок 3 — Панель выбора уровня.

Так же в GUI была добавлена панель выбора уровня с превью самих уровней.

UML Диаграмма проекта



UML Диаграмма лабораторной работы



Вывод:

Реализован шаблонный класс генерации уровней, а так же шаблонные классы правил для конкретных уровней, а так же Стратегия, позволяющая сгенерировать конкретный уровень.

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