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(ка) гр. 1381 _____ Денисова О.К.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы

Изучить шаблоны, создать шаблонный класс, генерирующий карту

Общая формулировка задачи

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.
- Класс генератор создает поле, а не принимает его.
- Класс генератор не должен принимать объекты классов правил в каком-либо методе, а должен сам создавать (в зависимости от реализации) объекты правил из шаблона.
- Реализовано не менее 6 шаблонных классов правил
- Классы правила должны быть независимыми и не иметь общего класса-интерфейса
- При запуске программы есть возможность выбрать уровень (не менее 2) из заранее заготовленных шаблонов
- Классы правила не должны быть только “хранилищем” для данных.
- Так как используются шаблонные классы, то в генераторе не должны быть dynamic_cast

Выполнение работы

В ходе выполнения лабораторной работы был реализован шаблонный класс-генератор поля FieldGenerator. Генератор принимает любое количество параметров, т.е. 0 и более, и, таким образом, является variadic template. Класс генерирует поле и возвращает его в методе generate(): создает поле через конструктор по умолчанию, с помощью метода add применяет к нему правила, переданные в качестве параметров шаблона, применяя распаковку, и возвращает указатель на него. Метод add является шаблонным (с параметром шаблона Rule), он создает объект класса Rule и вызывает у него метод fill, реализующий заполнение поле в зависимости от конкретного правила.

Шаблонные классы-правила имеют метод fill, принимающий указатель на поле, каждый класс по-своему реализует заполнение:

1) PatencyRule: принимает в параметре шаблона size_t count, делает непроходимыми клетки в количестве count, клетки выбираются рандомно с использованием random library c++11.
2) unitPositionRule: принимает в параметрах шаблона size_t i, size_t j, и устанавливает юнита в позицию (i, j). В случае, если i и/или j указаны больше размеров поля, юнит будет установлен в позицию, взятую от (i, j) по модулю от размеров поля.
3) WinRule: принимает в параметрах шаблона size_t x, size_t y, создает событие победы и устанавливает его в позицию (y, x). Если x и/или y указаны больше размеров поля, победа будет установлена в позицию, взятую от (y, x) по модулю размеров поля.
4) WolfRule: принимает в параметре шаблона size_t number, проходит в цикле number раз, на каждом проходе создавая новое событие Wolf() и устанавливая его в рандомно сгенерированные клетки.

5) MoneyRule: принимает в качестве параметров шаблона size_t money_count и size_t count. count отвечает за количество событий Money, которые будут на поле, money_count – за количество монет, которые будет получать игрок, в каждом из них. События размещаются рандомно аналогично предыдущим правилам.
6) MerchantRule: принимает в качестве параметра шаблона size_t number, где number – количество событий типа Merchant, которые будут сгенерированы на поле. События размещаются рандомно аналогично предыдущим правилам.

Также существуют два заранее созданных уровня игры: генератор, специализированный двумя разными способами. Создание поля при помощи генератора происходит в классе Game из предыдущих лаб.работ. Для класса Game реализованы методы: public chooseLevel и private setLevel. В конструкторе класса Game вызывается метод chooseLevel, запрашивающий у пользователя уровень игры: пользователь вводит 1 или 2, этот параметр передается в приватный метод setLevel, где и происходит создание карты. Сначала создается генератор нужного уровня (выбор происходит при помощи switch), затем у него вызывается метод generate, который создает поле и возвращает указатель на него, указатель присваивается в приватное поле field_, с которым дальше и происходит взаимодействие. Если в методе chooseLevel был выбран некорректный уровень (не 1 и не 2), тогда в методе setLevel будет по умолчанию установлен первый уровень игры, таким образом обеспечивается защита от ошибочного ввода.

Очищение памяти от поля происходит в деструкторе Game, очищение от всех добавленных событий, как и раньше, в деструкторе клетки.

UML-диаграмма

UML-диаграмма лабораторной работы представлена на рисунке 1.

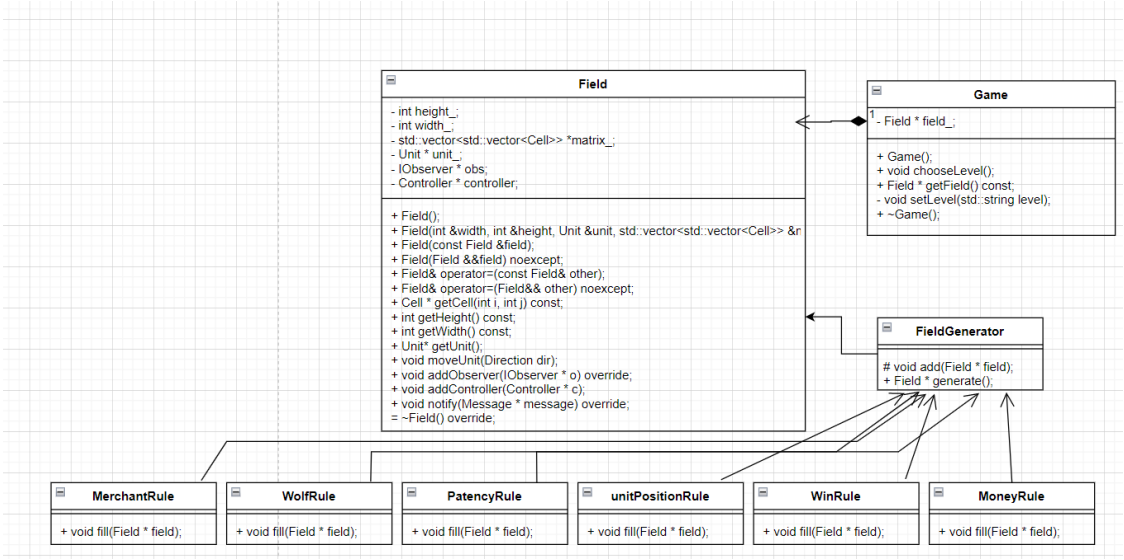

Рисунок 1. UML-диаграмма лабораторной работы №5

Выводы

В ходе выполнения лабораторной работы были приобретены знания о шаблонах в C++, и приобретен опыт написания шаблонов: реализован шаблонный класс-генератор, являющийся variadic template, и набор шаблонных классов-правил.