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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «ООП»

Тема: шаблонные классы, управление

Студент гр. 0382 _____ Азаров М.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Изучить такой механизм языка C++ как шаблоны.

Задание.

Необходимо определить набор правил для игры в виде классов (например, какие задачи необходимо выполнить, чтобы он мог выйти с поля; какое кол-во врагов и вещей должно быть на поле, и.т.д.). Затем определить класс игры, которое параметризуется правилами. Класс игры должен быть прослойком между бизнес-логикой и командами управления, то есть непосредственное изменение состояния игрой должно проходить через этот класс.

Требование:

Созданы шаблонные классы правил игры. В данном случае параметр шаблона должен определить конкретные значения в правилах.

Создан шаблонный класс игры, который параметризуется конкретными правилами. Класс игры должен проводить управление врагами, передачей хода, передавать информацию куда переместить игрока, и.т.д.

Потенциальные паттерны проектирования, которые можно использовать:

- Компоновщик (Composite) - выстраивание иерархии правил
- Фасад (Facade) - предоставления единого интерфейса игры для команд управления
- Цепочка обязанностей (Chain of Responsibility) - обработка поступающих команд управления
- Состояние (State) - отслеживание состояние хода / передача хода от игрока к врагам

- Посредник (Mediator) - организация взаимодействия элементов бизнес-логики

Выполнение работы.

Для реализации и выполнения задания были созданы следующие классы :

Класс ***Game*** :

Шаблонный класс модели игры. Во-первых отвечает за перемещение игрока, те является прослойкой между контролером и игроком. Также принимает при создании набор правил которые задают карту игры , расположение объектов на ней и режим игры. Из за того что класс шаблонный пришлось реализацию класса перенести в файл ***Game.inl*** и подключить после объявления в ***Game.h***. С помощью передаваемого правила ***GenLevel*** класс создает поле , с помощью правила ***RSetLocatedObj*** класс размещает объекты игры на поле, и с помощью правила ***RWinAndLose*** задает когда игрок побеждает а когда проигрывает.

Класс ***GenerationLevel*** :

Шаблонный класс правила создания уровня. При создании создает строителя поля, который потом в методе ***generationField()*** строит конкретный уровень. Уровень задается передаваемым ***int*** значением non-type в параметры шаблона. Есть реализация по умолчанию , которая создает простое квадратное поле без стен . Есть спецификация этого шаблона для значения 1 non-type, там прописано построение поля первого уровня.

Класс ***RuleSetLocatedGameObj*** :

Шаблонный класс правила распределения игровых объектов по уровню. Шаблон в параметрах шаблона принимает два значения non-type первое - уровень для которого нужно расположить объекты , второе - вариация расположения для этого уровня. Метод ***getSetEnemy()*** - создает существ на поле и возвращает вектор этих существ для дальнейшего обновления этих существ во время игры. Метод ***setItem()*** -создает предметы на поле. Метод ***getPlayer()*** - создает игрока на поле и возвращает умный указатель на игрока. Есть реализация по умолчанию , которая не создает существ и предметов и только ставит игрока на клетку (1,1). Есть спецификация этого шаблона для уровня 1 , для 1-го расположения .

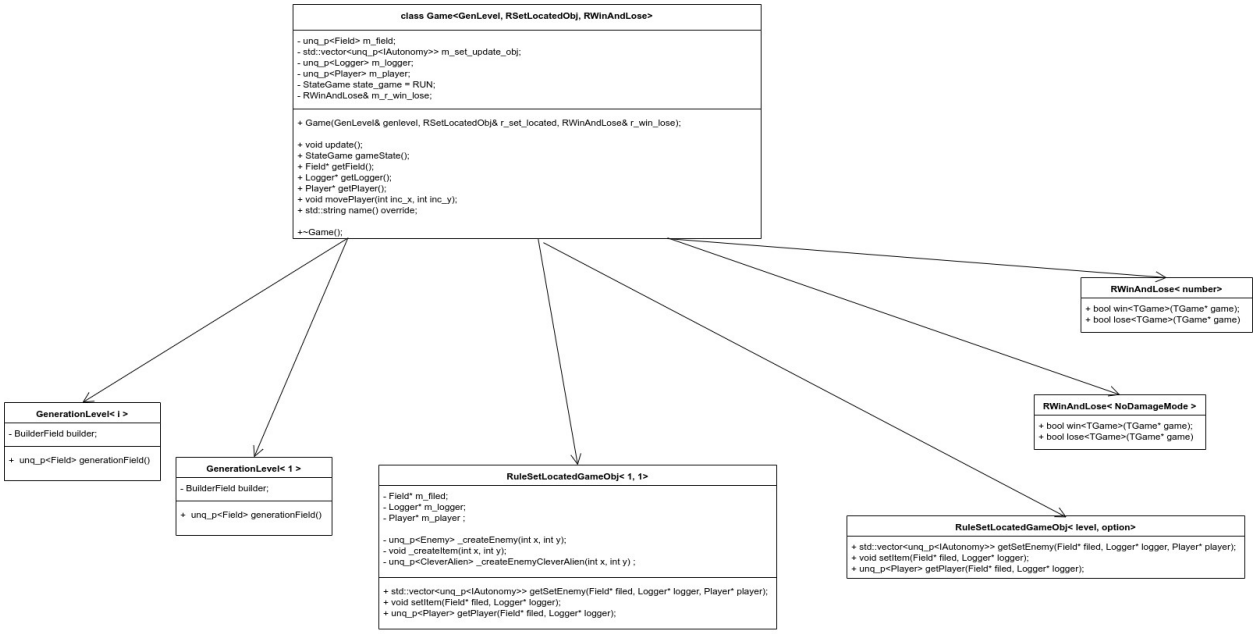
Класс ***RWinAndLose***:

Шаблонный класс правила выигрыша и проигрыша. Задает условия при которых игрок может выиграть и может проиграть. Шаблон в параметрах шаблона принимает какой режим нужно реализовать (передается с помощью enam ***modeGame***). Есть реализация по умолчанию для ***NormalMode***, по ней игрок проигрывает когда умирает , выигрывает когда достигает финишной клетки. Есть реализация режима ***NoDamageMode***, условия победы такие же , проигрывает когда получает урон. Класс имеет метод ***win()*** , который принимает саму модель игры и проверяет выполнены ли условия для победы или нет. Если выполнены возвращает ***true***, если нет возвращает ***false***. Метод ***lose()*** аналогичен только для условия проигрыша.

Мелкие изменения в уже созданных классах:

Так как класс ***Game*** стал шаблонным , то все класы которые принимали как параметр объект этого класса тоже пришлось сделать шаблонными. (Например ***Controler*** и ***ViewGame***).

UML диаграмма классов:



Выводы.

Был изучен механизм работы шаблонов и получен опыт в из пременении на практике . Разработана программа, выполняющая поставленное задание.