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

ОТЧЕТ

по лабораторной работе № 4

по дисциплине «Объектно-ориентированное программирование»

Тема: Шаблонные Классы. Управление.

Студент гр. 0303 _____ Болкунов В.О.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург 2021

Цель работы.

Создать набор классов — параметризуемых правил, параметризуемый класс игры.

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.

Основные теоретические положения.

Шабло́ны (англ. template) — средство языка C++, предназначенное для кодирования обобщённых алгоритмов, без привязки к некоторым параметрам (например, типам данных, размерам буферов, значениям по умолчанию). В C++ возможно создание шаблонов функций и классов. Шаблоны позволяют создавать параметризованные классы и функции. Параметром может быть любой тип или значение одного из допустимых типов (целое число, enum, указатель на любой объект с глобально доступным именем, ссылка). Хотя

шаблоны предоставляют краткую форму записи участка кода, на самом деле их использование не сокращает исполняемый код, так как для каждого набора параметров компилятор создаёт отдельный экземпляр функции или класса. Как следствие, исчезает возможность совместного использования скомпилированного кода в рамках разделяемых библиотек.

Выполнение работы.

Класс BaseController — базовый контроллер, то есть класс управления объектом расположенным на поле (так как сам объект на поле про себя ничего не должен знать и сам двигаться не может, аналогия из жизни — шахматные фигуры).

- шаблонный параметр T определяет тип управляемого объекта (T должен наследоваться от класса Entity)*
- содержит метод put для установки объекта (если тот не был поставлен на поле)
- метод move для относительного перемещения на заданный вектор.
- moveAbs — для абсолютного перемещения по полю.

(*) Здесь и далее требование наследования типа T от типа Base (в UML - «*extends*») реализовано с помощью использования конструкции

```
template<...
    typename std::enable_if<std::is_base_of<Base, T>::value, void *>::type * = nullptr,
...>
```

которая осуществляет проверку типа на этапе компиляции.

Класс PlayerController — контроллер для объекта игрока.

Класс EnemyController — контроллер для объекта врага.

Класс BotLogic — интерфейс алгоритма управления врагом. (паттерн — стратегия)

Класс Predator — алгоритм для преследования игрока на задаваемом расстоянии (наследник BotLogic)

Класс Bot — осуществляет автоматическое управление объектом врага в соответствии с заданным алгоритмом. Параметризуется алгоритмом управления.

- Метод spin — двигает объект врага на значение вектора полученного из алгоритма.

Класс Rule — абстрактное правило окончание игры (производит проверку условия виртуального метода condition и позиции игрока).

Класс ObjectsCounter — правило для уничтожения (сбора) объектов, параметризуется количеством (отрицательное число = все) уничтоженных (собранных) объектов и типом объектов T (T должно наследоваться от Entity).

- Принимает вектор weak_ptr : разделяемый указатель хранится только у самой клетки поля, поэтому количество уничтоженных объектов определяется с помощью метода expired у weak_ptr.

Класс Collector — правило для сбора предметов, параметризуется количеством собранных предметов.

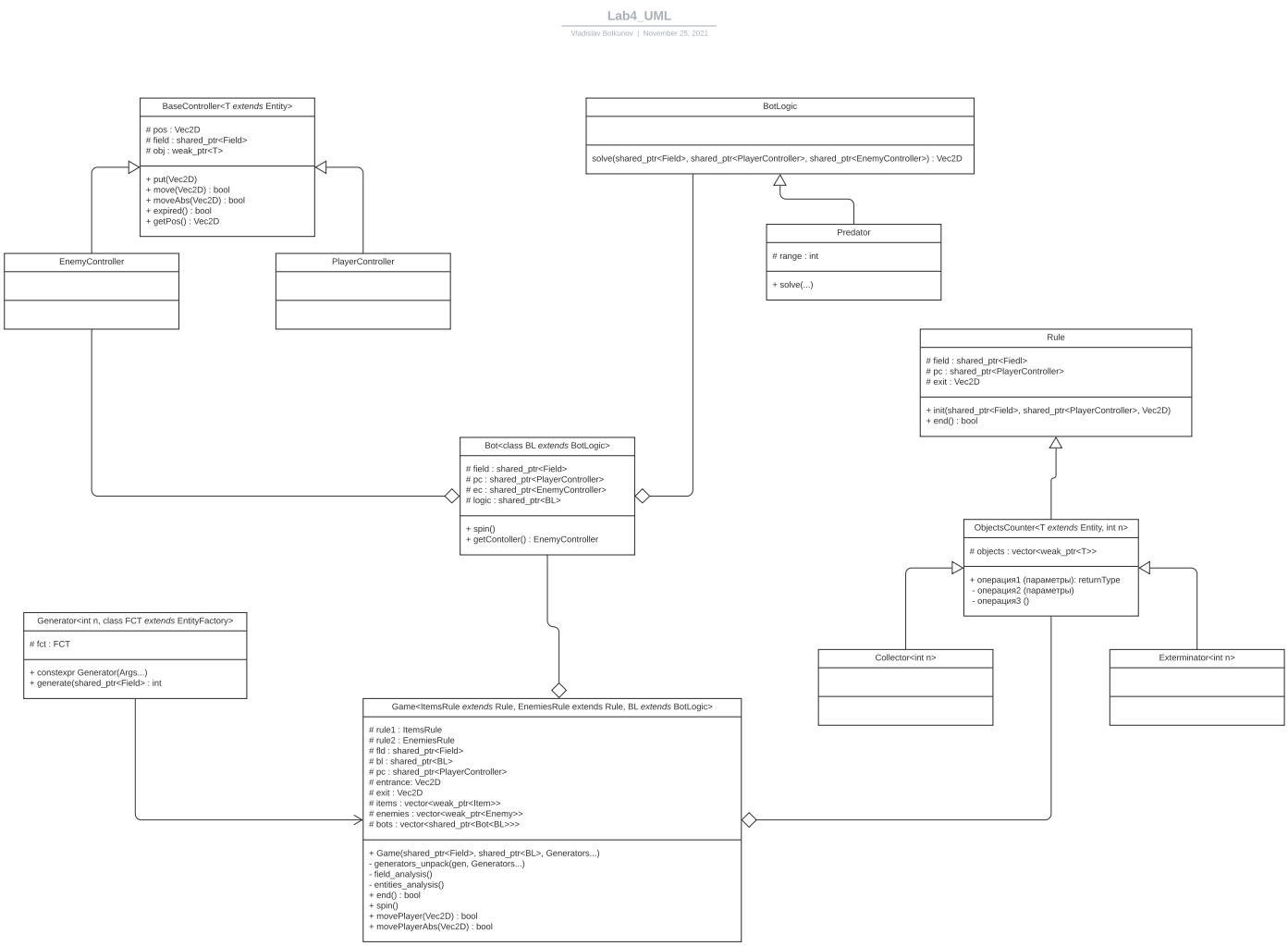
Класс Exterminator — правило для убийства вражеских объектов, параметризуется количеством.

Класс Generator — генератор объектов на игровом поле, параметризуется количеством генерируемых объектов и типом фабрики для генерации.

- Конструктор сделан с помощью вариативного шаблона (задаваемая фабрика инициализируется переданными в генератор аргументами).
- Метод generate производит попытки генерации объектов, возвращает количество успешно созданных объектов.

Класс Game — класс игры, параметризуется двумя типами правил и классом алгоритма для ботов. Одному правилу передаётся список указателей на предметы на поле, второму список указателей на врагов.

- Принимает игровое поле, алгоритм управления врагами, и вариативный шаблон, то есть произвольный список генераторов.
- Все генераторы будут вызваны в рекурсивном методе unpack_generators.
- При создании производит анализ поля: находит координату входа, выхода, предметы и врагов.
- Игрок ставится на найденную клетку входа.
- Методы movePlayer и movePlayerAbs для перемещения игрока относительно и абсолютно соответственно.
- Метод spin — производит ход роботов.
- Метод end — проверка соответствия правилам окончания.



Разработанный программный код см. в директории /*game_lib u /game*.

Разработанную диаграмму классов UML см. в Lab4_UML.pdf .

Тестирование.

Разработанные тесты см. в директории /tests.

Выводы.

Были реализованы классы для управления врагами, игроком, шаблонные классы задающие параметры и правила игры и сам класс игры, предоставляющий функции управления игрой.