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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «ООП»

ТЕМА: ДОБАВЛЕНИЕ ВРАГОВ

Студент гр. 9304 Прокофьев М.Д.

Преподаватель Шевская Н.В.

Санкт-Петербург

2020

Цель работы.

Обучение работе с классами, конструкторами на языке в C++.

Задание.

Создать шаблонный класс врага. Параметр шаблона должен определять поведение врага (параметров шаблона может быть несколько, например отдельный параметр для политики передвижения и для политики атаки). Класс врага должен препятствовать игроку. Класс игрока должен иметь возможность взаимодействовать с врагом и наоборот.

Обязательные требования:

-Создан шаблонный класс врага

-Создано не менее 3 типа поведения врагов

-Взаимодействие происходит через перегруженный оператор

Дополнительные требования:

-Передача хода между игроком и врагами происходит с использованием паттерна Состояния в классе игры

Основные теоретические положения.

Класс - это пользовательский тип данных.

Конструктор копирования используется для инициализации класса путем создания копии необходимого объекта.

Оператор присваивания копированием (или «копирующее присваивание») используется для копирования одного класса в другой (существующий) класс.

Конструкторы перемещения принимают ссылку на значение объекта класса и используются для реализации передачи владения ресурсами объекта параметра.

Метод в объектно-ориентированном программировании — это функция или процедура, принадлежащая какому-то классу или объекту. Как и процедура в процедурном программировании, метод состоит из некоторого количества

операторов для выполнения какого-то действия и имеет набор входных аргументов.

Интерфейс — программная/синтаксическая структура, определяющая отношение между объектами, которые разделяют определённое множество и не связаны никак иначе. При проектировании классов, разработка интерфейса тождественна разработке спецификации (множества методов, которые должен реализовывать каждый класс, использующий интерфейс).

Перегрузка операторов позволяет определить действия, которые будет выполнять оператор. Перегрузка подразумевает создание функции, название которой содержит слово operator и символ перегружаемого оператора. Функция оператора может быть определена как член класса, либо вне класса.

Инвариант объекта представляет собой конструкцию программирования, состоящую из набора инвариантных свойств. Это гарантирует, что объект всегда будет соответствовать предопределенным условиям, и поэтому методы могут всегда ссылаться на объект без риска сделать неточные презумпции. Определение инвариантов классов может помочь программистам и тестировщикам обнаружить больше ошибок при тестировании программного обеспечения.

Паттерн «Фасад» предоставляет унифицированный интерфейс вместо набора интерфейсов некоторой подсистемы. Фасад определяет интерфейс более высокого уровня, который упрощает использование подсистемы.

Шаблоны— средство языка C++, предназначенное для кодирования обобщённых алгоритмов, без привязки к некоторым параметрам (например, типам данных, размерам буферов, значениям по умолчанию).

Выполнение работы.

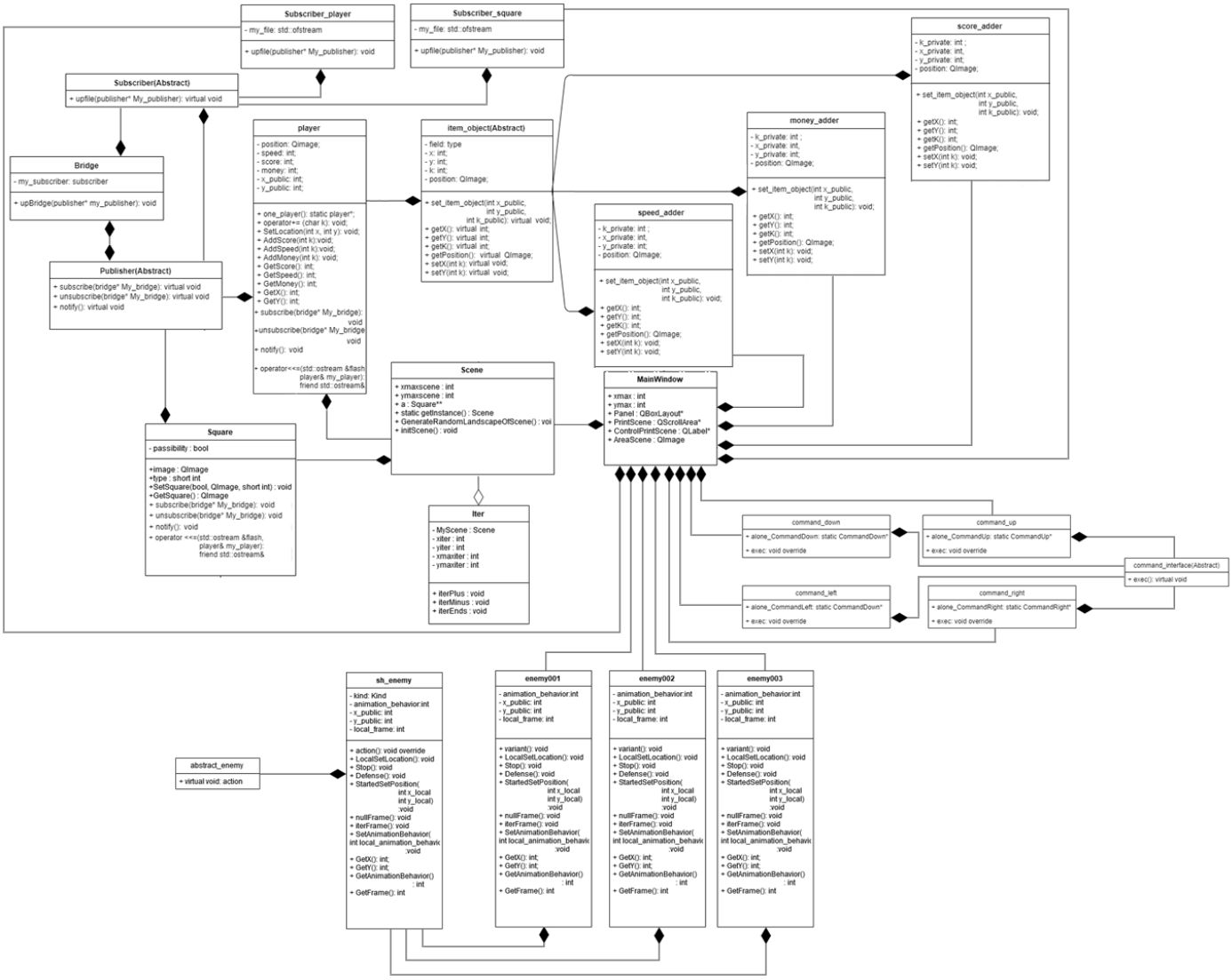

Рисунок 1 – UML-схема

Для выполнения работы создан абстрактный класс abstract_enemy, содержащий виртуальную функцию action.

Кроме того создан шаблонный класс sh_enemy, где “неопределенным” типом является тип Kind.

Созданы три класса enemy001, enemy002, enemy003, каждый из которых отвечает за линию поведения врага. В игре враг, в зависимости от того, какой он при себе имеет один из трех типов линии поведения, делает определенные действия. При enemy001 враг может сделать одно из трех действий: пропустить

ход, сходить на клетку ниже, защититься. Какое из трех действий сделает враг зависит от rand, который используется в функции variant(). Действия enemy002 почти аналогичны за исключением варианта перемещения: здесь враг пойдет не в клетку ниже, а в клетку слева. С enemy003 схожая ситуация: при enemy003 враг пойдёт вправо.

Кроме того, сделано взаимодействие между игроком и врагом. Взаимодействие реализовано через перегруженный оператор “-“, при столкновении игрока и врага, игрок проигрывает.

Реализована пошаговая система: каждый раз после того, как походит игрок, ходит враг.

Тестирование.

1) Запуск программы(проверка с поведением enemy001):

Рисунок 2 – Запуск

2) Игрок походил (сделал шаг вниз), после чего враг сделал одно из трех действий, в данном случае походил вниз:

Рисунок 3 – Враг сделал передвижения

3) Игрок походил (сделал шаг вправо), после чего враг вновь выполнил одно из трех действий, в данном случае, встал в режим обороны:

Рисунок 4 – Враг выполнил одно из действий

4) Игрок походил (сделал шаг вправо), после чего враг выполнил одно из действий, а именно – пас (не предпринял никакого хода):

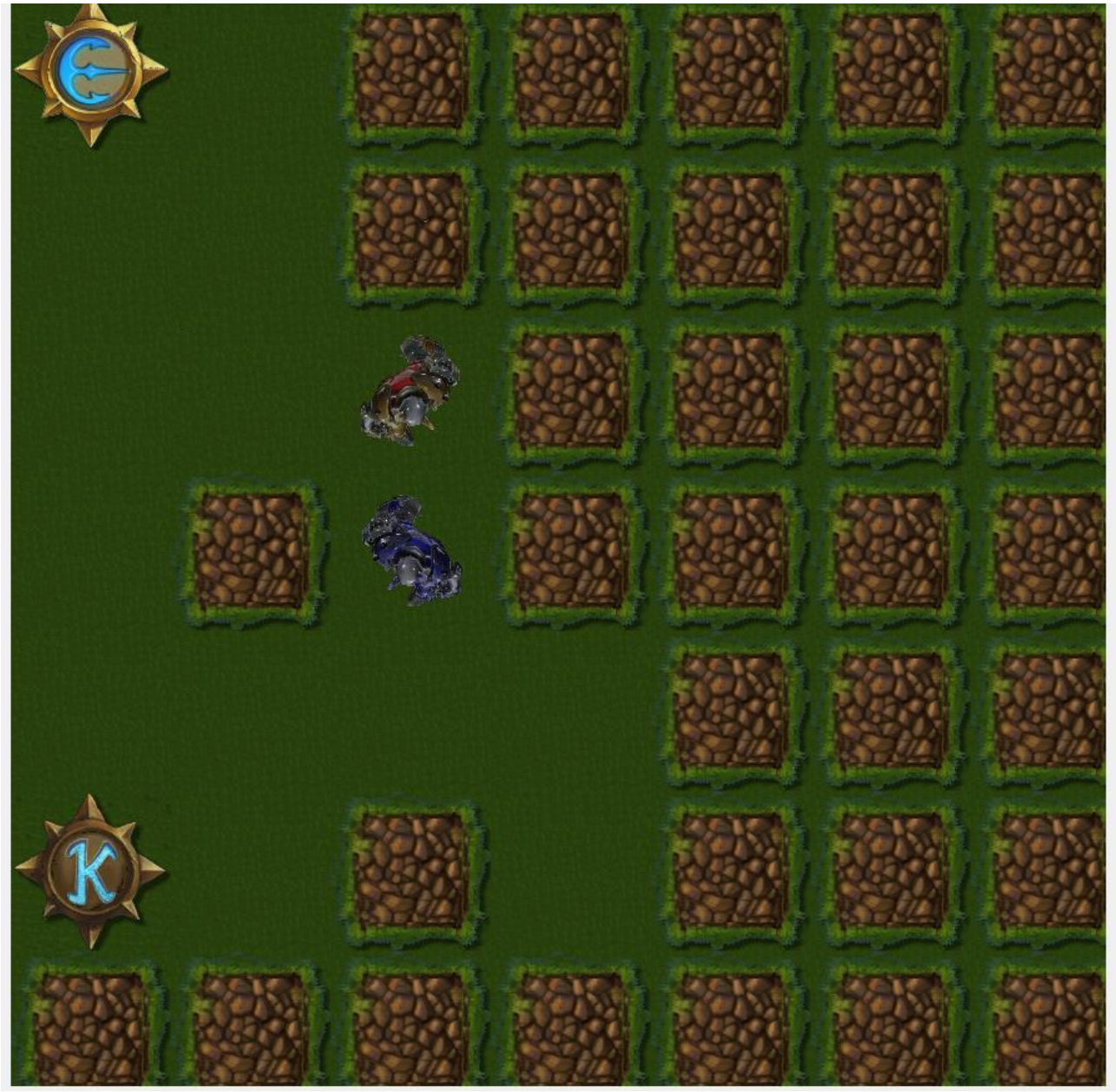

Рисунок 5 – Враг не походил

5) Игрок походил (сделал шаг вниз в клетку врага), после чего игрок был побежден.

Рисунок 6 – Игрок побежден

Выводы

Были получены практические знания по использованию классов в программировании. Был создан абстрактный класс abstract_enemy, к нему создан класс sh_enemy, представляющий из себя шаблон. Кроме того, было создано три класса линии поведения врага: enemy001, enemy002, enemy003. Каждой линии было создано три действия.