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: Уровни абстракции, управление игроком

Студент гр. 1304 _____ Павлов Д.Р.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы.

Реализовать набор классов отвечающих за считывание команд пользователя, обрабатывающих их и изменяющих состояния программы (начать новую игру, завершить игру, сохраниться, управление игроком, и.т.д.). Команды/клавиши определяющие управление должны считываться из файла.

Требования.

Реализован класс/набор классов обрабатывающие команды Управление задается из файла

Реализованные классы позволяют добавить новый способ ввода команд без изменения существующего кода (например, получать команды из файла или по сети)

Из метода считывающего команду не должно быть “прямого” управления игроком

Описание архитектурных решений и классов.

Описание классов:

1) ***Command_Reader*** — Класс Интерфейс Обработчика команд Юзера. Имеет **чисто виртуальный** метод get_character(). От него наследуются ***File_Reader*** и ***Input_Stream_Reader,*** и он определяет их поведение. В Классах-Наследниках идет перегрузка метода get_character(). Интересные факты о данном классе: 1) — Класс занимает всего 4 строчки кода; 2) — Метод

get_character() возвращает char, поскольку идея названия метода была взята с небезизвестного макроса getchar(), который считывает символ со стандартного потока ввода, и после — возвращает его; 3) — Данный Класс был создан 8 октября. В этот день Республика Намибия отмечает день посадки деревьев.

2) ***File_Reader*** — Наследник Класса Command_Reader. Отвечает за считывания команд из файла (File_Controller.txt), с помощью переопределения метода get_character. Считывание происходит за счет открытия файла и вызова у него метода get(), который считывает символ в файле, далее возвращает этот символ. Так же вызывается функция sleep, которая замедляет действия игрока на 1 сек. Это сделано для того, что без sleep'а действия выполняются слишком быстро.

3) ***Input_Stream_Reader*** — Наследник Класса Command_Reader. Отвечает за считывание команд со стандартного потока ввода. Возвращает getchar().

4) ***Request_Stream*** — Класс, отвечающий за выбор пользователем потока ввода при помощь перемещения кнопок YES/NO. Возвращает тип считывания (перечисление) для определения в классах фрейма выше уровень считывания.

5) ***Request_Stream_Dispaly*** — Отвечает за визуализацию панели выбора пользователем потока ввода. Панель отрисовывает: «Would you like to control from a file?», что в переводе на русский означает «Вы хотите управлять с файла?»; ответов юзера, которые переключаются «Yes» / «No» (в переводе на русский «Да» / «Нет»); и отрисованного Алукарда. Было решено нарисовать именно Алукарда, поскольку название игры связано с Дьяволом «Devil May

Cry 6», в которой мы управляем **Вампиром** (Как известно, Алукард тоже было вампиром). Так же цвет всех символов, отрисовыющих панель — КРАСНЫЙ. Это решение создает мрачную атмосферу с самого запуска игры.

6) Command_Wrapper — Класс-обертка для всех классов выше. Имеет поле структуры Control_Setting, внутри которой хранятся файлы с путями к конкретным биндам, а так же Commands, commands, которая отвечает за сохранение символов-комманд. Данный класс так же с помощью методов создает файловый или стандартный ввод, а так же конвертирует символ пользователя в перечисление.

Демонстрация работы программы и тестирование.

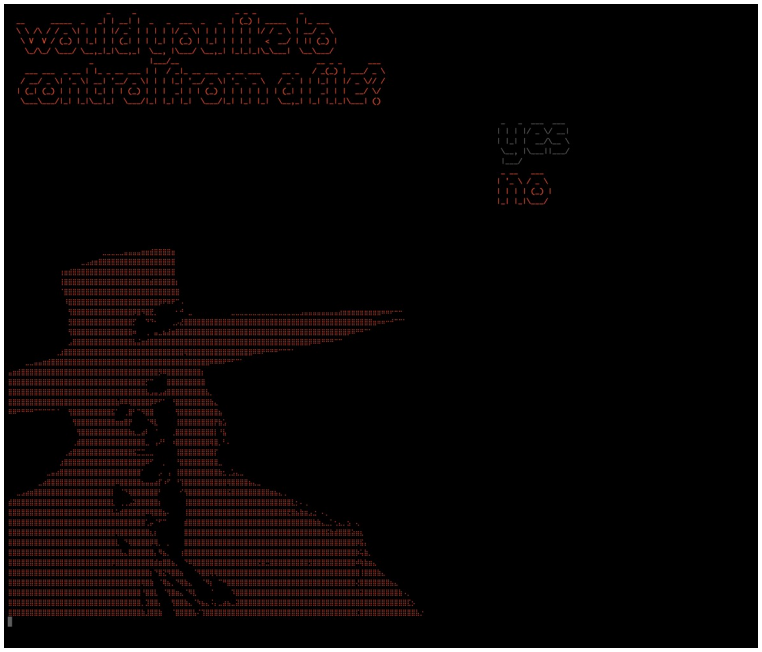

Рисунок 1 — Панель выбора потока ввода.

Для перемещения между yes/no следует нажимать W/S (ВВЕРХ/ВНИЗ).

Если Юзер ответит «YES», то далее он должен открыть файл File_Controller.txt, находящийся в корневой директории, и в нем вводить символ, который отвечает за дальнейшие действия(Например, чтобы герой пошел налево, вводится символ «d»). Если же «NO», то идет классическое управление (со стандартного потока ввода).

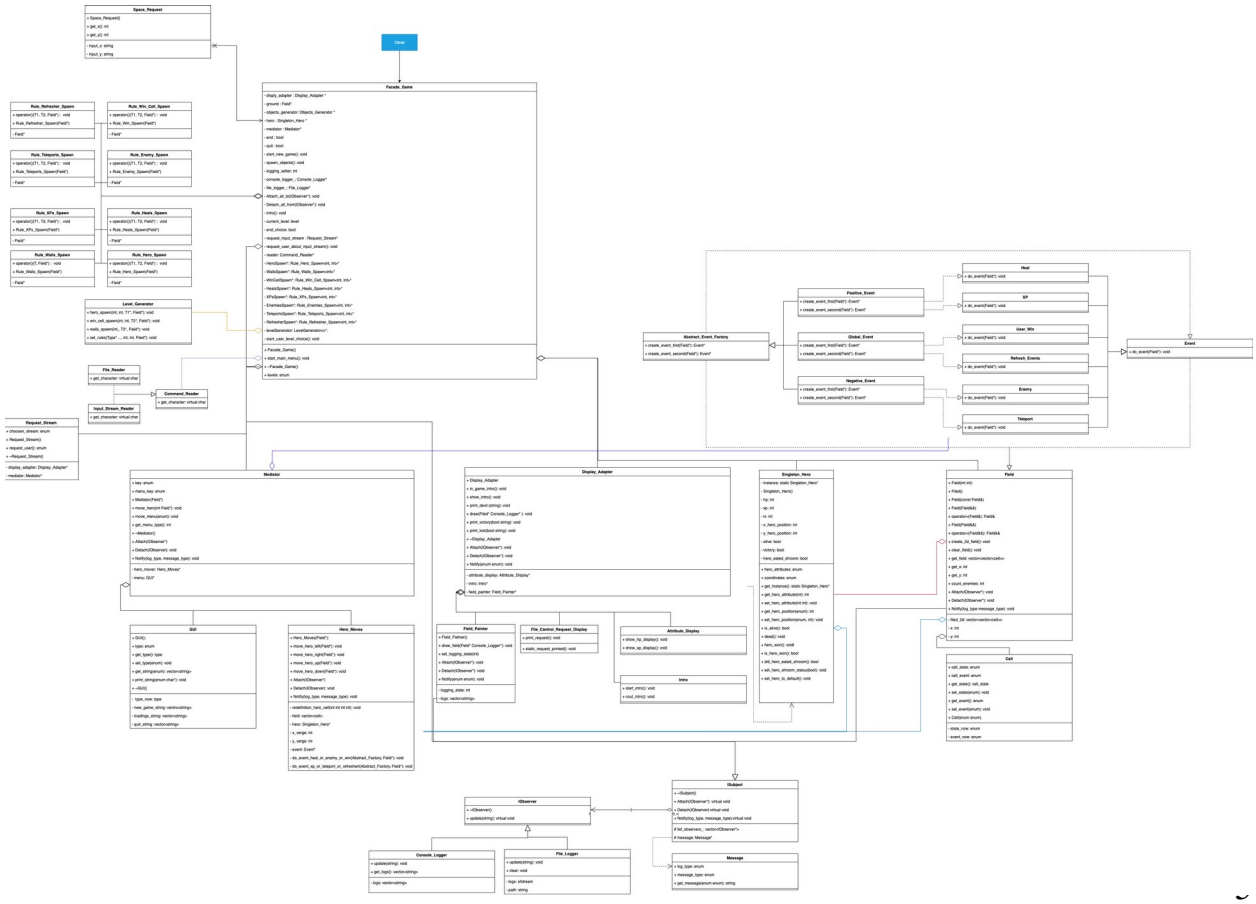

Рисунок 2. - UML Диаграмма всего проекта

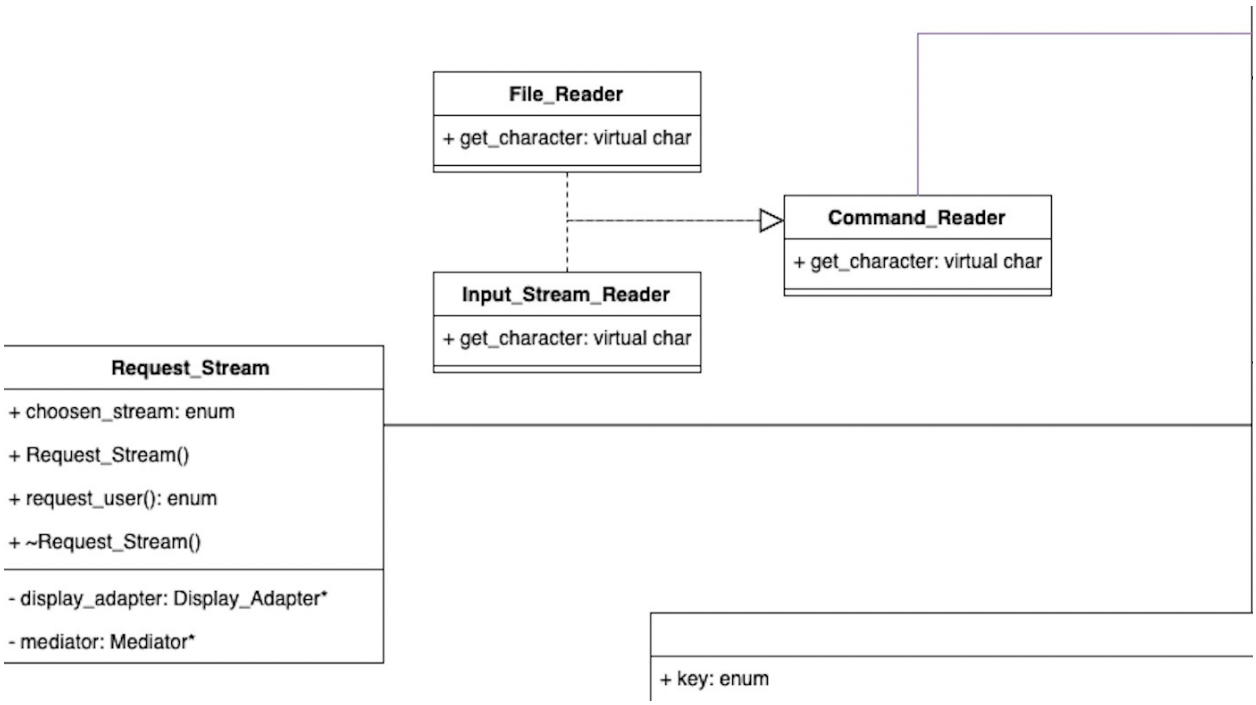

Рисунок 3. - UML Диаграмма лабораторной работы.

Вывод.

Реализованы классы, обрабатывающие команды пользователя, которые позволяют управлять как из файла, так и из стандартного потока ввода.

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