**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: Управление, разделение на уровни абстракции**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0303 |  | Калмак Д.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2021

**Цель работы.**

Реализовать управление игрой.

**Задание.**

Необходимо организовать управление игрой (номинально через CLI). При управлении игрой с клавиатуры должна считываться нажатая клавиша, после чего происходит перемещение игрок или его взаимодействия с другими элементами поля.

Требования:

* Реализовать управление игрой. Считывание нажатий клавиш не должно происходить в классе игры, а должно происходить в отдельном наборе классов.
* Клавиши управления не должны жестко определяться в коде. Например, это можно определить в отдельном классе.
* Классы управления игрой не должны напрямую взаимодействовать с элементами игры (поле, клетки, элементы на клетках)
* Игру можно запустить и пройти.

Потенциальные паттерны проектирования, которые можно использовать:

* Команда (Command) - передача команд с информацией через единый интерфейс. помещение команд в очередь
* Посредник (Mediator) - организация взаимодействия различных модулей

**Выполнение работы.**

Класс Commands – класс команд для управления игроком. Для удобства использовано перечисление enum – Command: UP, DOWN, LEFT, RIGHT. Для соответствия клавиш для управления и команд использован контейнер map – поле Commands\_l. Commands\_l = {{'w', UP}, {'s', DOWN}, {'a', LEFT}, {'d', RIGHT}}. Поле command для команды для игрока.

Класс ConsoleController – класс для считывания и обработки команд. Метод get\_command() – метод считывает символ для управления и возвращает команду, соответствующую этому символу.

В классе Game в методе start() в переменные xp, yp записываются координаты игрока. В первый раз они совпадают с координатами входа. С помощью цикла определяются координаты выхода. Это нужно для того чтобы игрок, достигнув выхода, прошел игру. Создается объект класса Commands control и объект класса ConsoleController controller. Control будет содержать команды для управления игрока, которые будет передавать controller. Запускается цикл и работает, пока игрок не дойдет до выхода или не потеряет все хп. Выводится состояние поля. Control получает команду от controller с помощью метода get\_command() и записывает в поле command. В зависимости от команды игрок пойдет вверх, вниз, влево, вправо. Осуществляется сравнение команды с существующими командами и выполняется движение. За рамки поля игрок выйти не может. Выполняется движение врагов.

Игрок взаимодействует с предметами, врагами. Игрок может дойти до выхода и пройти игру.

Разработанный код см. в Приложении А.

Разработанную диаграмму классов UML см. в Приложении Б.

**Выводы.**

Таким образом, был создан класс контроллера и класс команд для управления игрока. Игру можно пройти.

**ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ**

**Файл Commands.h:**

#ifndef COMMANDS\_H

#define COMMANDS\_H

#include <map>

using namespace std;

#include <iostream>

class *Commands*{

    public:

        enum *Command*{

            UP, DOWN, LEFT, RIGHT

        };

*Command* command;

        map<char, *Command*> Commands\_l = {

            {'w', UP}, {'s', DOWN}, {'a', LEFT}, {'d', RIGHT}

        };

};

#endif

**Файл ConsoleController.h:**

#ifndef CONSOLECONTROLLER\_H

#define CONSOLECONTROLLER\_H

#include "Commands.h"

class *ConsoleController*: public *Commands*{

    public:

*Command* get\_command(){

            char c;

            cin >> c;

            return Commands\_l[c];

        }

};

#endif

**ПРИЛОЖЕНИЕ Б UML ДИАГРАММА КЛАССОВ**

