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

ОТЧЕТ №4

по лабораторной работе

по дисциплине «Объектно-ориентированное программирование»

ТЕМА: УРОВНИ АБСТРАКЦИИ, УПРАВЛЕНИЕ ИГРОКОМ

Студент гр. 1381 _____ Дудко М.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы.

Ознакомиться с работой основных принципов ООП. Написать примитивную игру на языке C++.

Задание.

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.

Выполнение работы.

В интерфейсе Command_Interfac определен один полностью виртуальный метод command_read. Который в качестве аргументом принимает клавиши управления, а так же ссылку на направление движение, которое он должен переопределить.

Реализация интерфейса (смотри Рис.1) происходит в классе Control_Console, в котором данный метод был переопределен. В данном случае при помощи функций _khibit() и _getch() символы считываются с консоли.

```
#include "Control_Console.h"
void Control_Console::command_read(std::map<char, char>& setting, char& direction){
    if (_kbhit()) {
        char button = _getch();
        if(setting.count( x: button)!=0)  direction = setting[button];
    }
}
```

Рисунок 1 - Реализация Интерфейса

В классе Command_File (смотри Рис.2) создан конструктор (открывающий файл), а так же деструктор(закрывающий его). В методе read_str считывается строка, записанная в файл. Метод get_char возвращает элемент строки с заданным индексом. Метод def_commands предотвращает неправильную запись в файл с командами, и в случае не верного формата записи использует команды по умолчанию.

```
void Command_File::read_str(){
    if(file.is_open()){
        getline( & file,str);
    }
}

void Command_File::def_commands(){
    for(int i = 0; i < str.length(); i++){
        for(int j = i+1; j < str.length(); j++){
            if(str[i] == str[j]) str = ":wsad";
        }
    }
    if(str.length() != 5){
        str = ":wsad";
    }

}

char  Command_File::get_char(int n){
    return str[n];
}
```

Рисунок 2 Считывание управления из файла

В классе Command_Reader (смотри Рис.3) в методе read_commands с помощью метода get_char из класса Command file записываются определенные в файле клавиши управления. В методе player_handler вызывается метод класса Control_Console command_read при помощи указателя на объект интерфейса. В это методе определяется переменная direction.

```
void Command_Reader::player_handler(Command_File& command_file) {
    // U - UP D - DOWN L - LEFT R - RIGHT
    command_read->command_read(setting, direction);
}

void Command_Reader::read_commands(Command_File& command_file){
    command_file.read_str();
    command_file.def_commands();
    setting[command_file.get_char( n: 1)] = 'U';
    setting[command_file.get_char( n: 2)] = 'D';
    setting[command_file.get_char( n: 3)] = 'L';
    setting[command_file.get_char( n: 4)] = 'R';
}
```

Рисунок 3 - Определение направления движения

Класс Controller получает направление (direction) из метода get_direction класса Command_Reader и перемещает игрока на одну клетку в данном направлении.

UML диаграмма (смотри Рис.4)

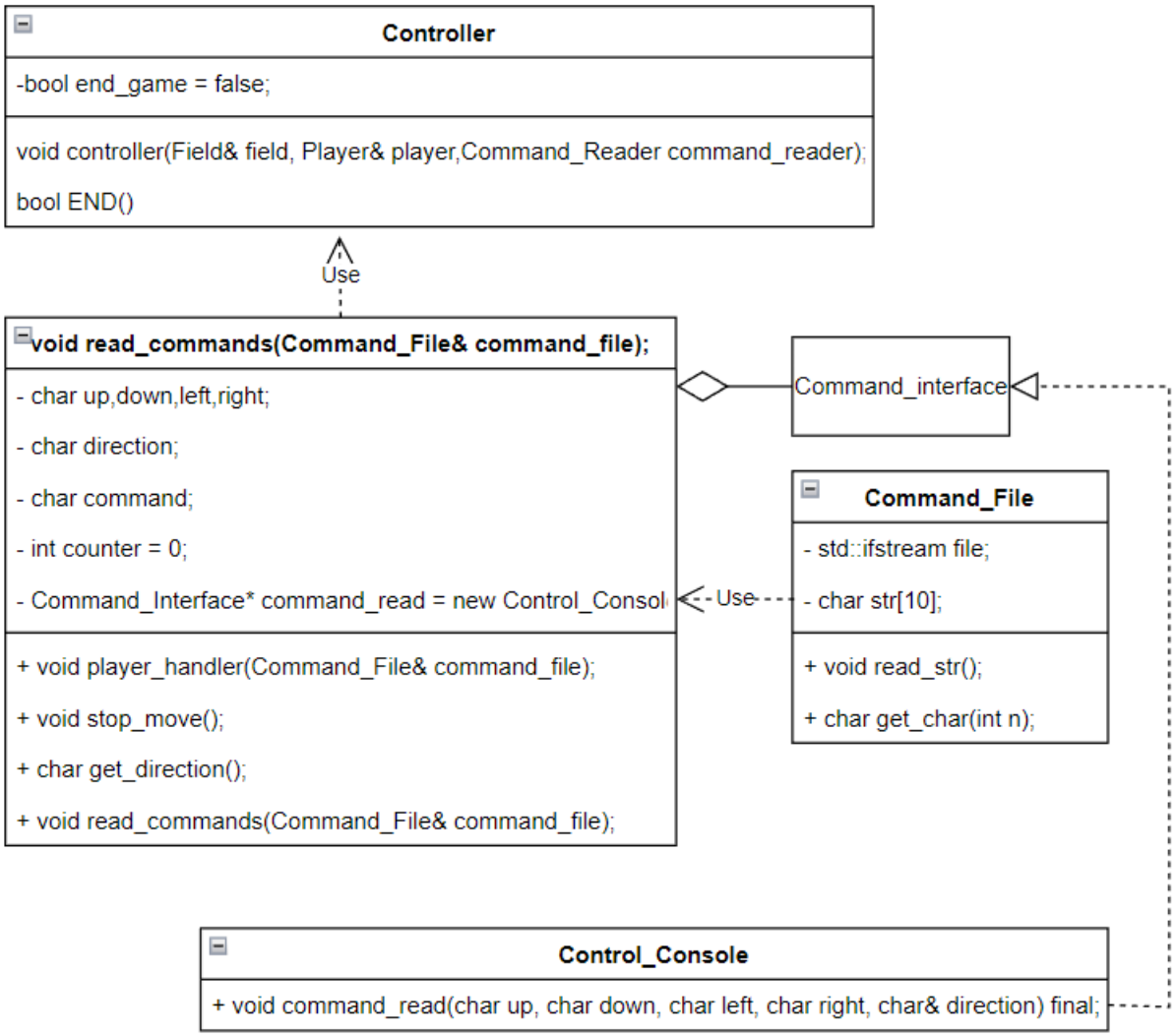

Рисунок 4 - UML

Вывод

Изучены принципы вводы и вывода данных, а так же механизмы работы с файлами, изучены уровни абстракции, постигнут дзен, достигнута нирвана, познана вселенная.