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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Объектно-ориентированное программирование»

Тема: Логирование, перегрузка операций

Студент гр. 0303 Парамонов В.В.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Реализовать возможность логирования состояния и изменения различных объектов игры в ходе работы с возможностью вывода в файл или консоль, а также изменения формата логов.

Задание.

Необходимо проводить логирование того, что происходит во время игры.

Требования:

- Реализован класс логгера, который будет получать объект, который необходимо отслеживать, и при изменении его состоянии записывать данную информацию.
- Должна быть возможность записывания логов в файл, в консоль или одновременно в файл и консоль.
- Должна быть возможность выбрать типа вывода логов
- Все объекты должны логироваться через перегруженный оператор вывода в поток.
- Должна соблюдаться идиома RAII.

Основные теоретические положения.

Потенциальные паттерны проектирования, которые можно использовать:

- Адаптер (*Adapter*) - преобразование данных к нужному формату логирования.
- Декоратор (*Decorator*) - форматирование текста для логирования.
- Мост (*Bridge*) - переключение между логированием в файл/консоль.
- Наблюдатель (*Observer*) - отслеживание объектов, которые необходимо логировать.
- Синглтон (*Singleton*) - гарантия логирования в одно место через одну сущность.
- Заместитель (*Proxy*) - подстановка и выбор необходимого логирования.

Выполнение работы.

1) Логирование:

- Класс *Log* – обертка для сообщения, содержащий само сообщение в *log_message_* и информацию о выбранном стиле (выбранных стилях) в *log_style_*. Имеет два геттера *getMessage* и *getStyle* для получения сообщения или стиля соответственно.
- Классы *LogConsole* и *LogFile* – наследники класса , необходимы для определения внутри класса-логгера в какой поток следует выводить данное сообщение (в консоль или в файл соответственно).
- Перед рассмотрением двух основных классов *Logger* и *Loggable* остановимся на двух вспомогательных перечислениях: *LogMode*, который нужен для выбора вывода в консоль или в файл (*kConsole* и *kFile*), и *LogStyle*, позволяющий выбрать различные оформления логов. Оба перечисления созданы таким образом, чтобы через оператор “или” была возможность выбрать несколько вариантов. Таким образом можно выбрать вывод в файл и в консоль одновременно или несколько стилей.
- Теперь рассмотрим класс *Logger* – сам логгер. Он создан с использованием Паттерна “одиночка” и при создании объекта открывает поток в файл по стандартному пути (*log_path_ = "log.txt"*). Изменить путь можно с помощью метода *setLogPath*. Вывод логов осуществляется через 2 перегруженных оператора вывода: Logger& operator << (LogFile& log_info) и Logger& operator << (LogConsole& log_info). В зависимости от вида класса логов используется первую (для вывода в файл) или вторую (для вывода в консоль) перегрузку. Перед выводом логи стилизуются в соответствии с выбранными в них *LogStyle* методом *stylize_log*.
- И, наконец, последнее звено механизма логирования - класс *Loggable*, позволяющий логировать унаследованные от него

классы. Логирование включается с помощью метода *LogActive,* который ставит внутренний флаг *logging_active_* в значение *true*, выключается с помощью *LogDisable,* который ставит *logging_active_* в значение *false*. Сама информация в логгер передается с помощью метода *LogInfo*, который принимает: куда мы выводим логи (*LogMode*), стиль логов (*LogStyle*) и само сообщение *message,* после этого в зависимости от выбранного вывода, переданное сообщение или стиль оборачивается в класс *LogConsole* или *LogFile*, эта обертка передается логгеру через оператор вывода.

2) Бизнес-логика:
   - Класс *GameLogic,* олицетворяющий связь бизнес-логики игры, стал наследоваться от *Loggable* для поддержки логирования. Здесь логируются события создания комнаты с выводом её размера, события создания всех объектов комнаты и их характеристики, взаимодействия объектов друг с другом в ходе игры.
3) GUI:
   - Класс *GameView* – основа графического интерфейса, стал наследоваться от *Loggable* для поддержки его логирования. В целях теста информация из этого класса логируется только про изменение размера экрана приложения.

Разработанный программный код см. в приложении А.

Тестирование см. в приложении Б.

UML-диаграмму см. в приложении В.

Выводы.

Был реализован вывод логов во время работы приложения в консоль и в файл, для логирования объекта достаточно унаследовать его от *Loggable*, добавить, где необходимо получать информацию, *LogInfo*, и установить объект логируемым через метод *LogActive*.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <QApplication>

#include "Gui/GameView.h"
#include "Logic/Room/Room.h"
#include "Logic/Room/Pos.h"
#include "Logic/GameLogic.h"
#include "Logic/Controller/Controller.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    GameLogic logic;
    logic.LogActive();
    GameView game_view(&logic);
    game_view.LogActive();
    logic.CreateRoom();
    Controller control(&logic, &game_view);
    game_view.show();

    return a.exec();
}
```

Название файла: Log.h

```
#ifndef LOG_H
#define LOG_H

#include "string"

#include "../UtilityEnums/LogStyle.h"

class Log {
private:
    std::string log_message;
    int log_style;
public:
    Log(std::string message, LogStyle style);

    std::string getMessage();
    int getStyle();
};

#endif // LOG_H
```

Название файла: LogConsole.h

```
#ifndef LOGCONSOLE_H
#define LOGCONSOLE_H

#include "Log.h"

class LogConsole: public Log {
public:
```

```
    LogConsole(std::string message, LogStyle style);
};

#endif // LOGCONSOLE_H
```

Название файла: LogFile.h

```
#ifndef LOGFILE_H
#define LOGFILE_H

#include "Log.h"

class LogFile: public Log {
public:
    LogFile(std::string message, LogStyle style);
};

#endif // LOGFILE_H
```

Название файла: Loggable.h

```
#ifndef LOGGABLE_H
#define LOGGABLE_H

#include "string"

#include "../UtilityEnums/LogMode.h"
#include "../UtilityEnums/LogStyle.h"
#include "../Logger/Logger.h"
#include "../Log/LogFile.h"
#include "../Log/LogConsole.h"

class Loggable {
    bool logging_active = false;
public:
    void LogActive();
    void LogDisable();
    void LogInfo(int mode, int style, std::string message);
};

#endif // LOGGABLE_H
```

Название файла: Logger.h

```
#ifndef LOGGER_H
#define LOGGER_H

#include <time.h>

#include <iostream>
#include <stdexcept>
#include <fstream>
#include <string>

#include "../Log/LogFile.h"
#include "../Log/LogConsole.h"
```

```
#include "../UtilityEnums/LogStyle.h"

class Logger {
private:
    static Logger* logger_inst;
    std::ofstream file_stream;
    std::string log_path;

    long long int log_count;

    Logger();

public:
    static Logger& getInstance();
    Logger(const Logger& from) = delete;
    Logger& operator =(const Logger& from) = delete;

    void setLogPath(std::string new_path);

    Logger& operator << (LogFile& log_info);
    Logger& operator << (LogConsole& log_info);

    std::string stylize_log(Log& log_info);

    ~Logger();
};

#endif // LOGGER_H
```

Название файла: LogMode.h

```
#ifndef LOGMODE_H
#define LOGMODE_H

enum LogMode {
    kConsole = 0b01,
    kFile = 0b10
};

#endif // LOGMODE_H
```

Название файла: LogStyle.h

```
#ifndef LOGSTYLE_H
#define LOGSTYLE_H

enum LogStyle {
    kCapitalise = 0b001,
    kTime =       0b010,
    kNumbered =   0b100
};

#endif // LOGSTYLE_H
```

Название файла: GameLogic.h

```
#ifndef GAMELOGIC_H
#define GAMELOGIC_H

#include <vector>

#include <QFile>

#include "../Gui/View/Observable.h"
#include"../Logging/Loggable/Loggable.h"
#include "Room/Room.h"
#include "Room/Pos.h"
#include "PlaceableInCell/Player/Player.h"
#include "PlaceableInCell/Enemies/DamageableEnemy.h"
#include "PlaceableInCell/Enemies/LimitedLifeEnemy.h"
#include "PlaceableInCell/Enemies/ImmortalEnemy.h"
#include "PlaceableInCell/Items/SanityAffectItem.h"
#include "PlaceableInCell/Items/DamageToEnemiesItem.h"
#include "PlaceableInCell/Items/KeyItem.h"
#include "PlaceableInCell/Enemies/MoveStrategies/StrategiesEnum.h"

class GameLogic: public Observable, public Loggable {
private:
    Room* curr_room_;
    std::vector<PlaceableInCell*> room_objects_;

    std::map<MoveStrategies, MoveStrategy*> strategy_list;
public:
    GameLogic();

    void CreateRoom(int id = 1, int is_new_or_load = 0);
    void Start();

    Room& getRoom();
    std::vector<PlaceableInCell*>& getRoomObjects();

    void LoadCellObjects(QFile& obj_file);
    void MakeTurn(Pos player_pos_change);

    ~GameLogic();
};

#endif // GAMELOGIC_H
```

Название файла: GameView.h

```
#ifndef GAMEVIEW_H
#define GAMEVIEW_H

#include <vector>
#include <typeinfo>

#include <QSizePolicy>
#include <QResizeEvent>
#include <QGraphicsScene>
#include <QGraphicsPixmapItem>
#include <QPropertyAnimation>
#include <QLabel>
```

```
#include <QWidget>

#include "View/Observer.h"
#include "../Logging/Loggable/Loggable.h"
#include "../Logic/GameLogic.h"
#include "DrawObject/DrawableObject.h"

namespace Ui {
class GameView;
}

class GameView : public QWidget, public Observer, public Loggable {
private:
    Q_OBJECT

    Ui::GameView *ui;
    QGraphicsScene* room_scene;
    GameLogic* curr_logic_;
    std::vector<DrawableObject*> draw_objects_;

    int floor_cell_size_ = 500;
    int top_padding_scene_ = 500;

    virtual void resizeEvent(QResizeEvent* event);
    virtual void keyPressEvent(QKeyEvent *event);

    QLabel* test_info;
public:
    explicit GameView(GameLogic* new_logic, QWidget *parent = nullptr);

    void UpdateRoomChange();
    void UpdateRoomObjectsChange();
    void UpdateObjectDestroy(int i);
    void UpdateTurnComplete();
    void UpdateWin();
    void UpdateDefeat();

    void TestShowPlayerConditions();
    void CenterViewOnPlayerPos();

    void ClearObjects();
    void DeleteRoom();

    ~GameView();
signals:
    void PlayerPressMove(Pos);
};

#endif // GAMEVIEW_H
```

Название файла: Log.cpp

```
#include "Log.h"

Log::Log(std::string message, LogStyle style) {
    log_message = message;
    log_style = style;
```

```
}

std::string Log::getMessage() {
    return log_message;
}

int Log::getStyle() {
    return log_style;
}
```

Название файла: LogConsole.cpp

```
#include "LogConsole.h"

LogConsole::LogConsole(std::string message, LogStyle style): Log(message,
style) {}
```

Название файла: LogFile.cpp

```
#include "LogFile.h"

LogFile::LogFile(std::string message, LogStyle style): Log(message,
style) {}
```

Название файла: Loggable.cpp

```
#include "Loggable.h"

void Loggable::LogActive() {
    logging_active = true;
}

void Loggable::LogDisable() {
    logging_active = false;
}

void Loggable::LogInfo(int mode, int style, std::string message) {
    if (logging_active) {
        if (mode & LogMode::kFile) {
            LogFile log_file = LogFile(message, stat-
ic_cast<LogStyle>(style));
            Logger::getInstance() << log_file;
        }
        if (mode & LogMode::kConsole) {
            LogConsole log_console = LogConsole(message, stat-
ic_cast<LogStyle>(style));
            Logger::getInstance() << log_console;
        }
    }
}
```

Название файла: Logger.cpp

```
#include "Logger.h"

Logger* Logger::logger_inst = nullptr;
```

```
Logger::Logger() {
    log_path = "log.txt";
    file_stream.open(log_path);
    if (!file_stream.is_open()) {
        std::string error_msg = "Can't open file with path: " + log_path;
        throw new std::invalid_argument(error_msg);
    }

    log_count = 0;
}

Logger& Logger::getInstance() {
    if (logger_inst == nullptr) {
        logger_inst = new Logger();
    }
    return *logger_inst;
}

void Logger::setLogPath(std::string new_path) {
    log_path = new_path;
}

Logger& Logger::operator << (LogFile& log_info) {
    if (log_info.getStyle() & LogStyle::kNumbered)
        log_count++;
    file_stream << stylize_log(log_info);
    file_stream.flush();
    return *logger_inst;
}

Logger& Logger::operator << (LogConsole& log_info) {
    if (log_info.getStyle() & LogStyle::kNumbered)
        log_count++;
    std::cout << stylize_log(log_info);
    fflush(stdout);
    return *logger_inst;
}

std::string Logger::stylize_log(Log& log_info) {
    std::string message = "";

    if (log_info.getStyle() & LogStyle::kCapitalise) {
        message = log_info.getMessage();
        for (auto it = message.begin(); it < message.end(); it++) {
            *it = toupper(*it);
        }
    }  else {
        message = log_info.getMessage();
    }
    if (log_info.getStyle() & LogStyle::kTime) {
        time_t curr_sec = time(nullptr);
        std::string curr_time = asctime(localtime(&curr_sec));
        curr_time.pop_back();
        message = "(" + curr_time + ") " + message;
    }

    if (log_info.getStyle() & LogStyle::kNumbered) {
```

```
        message = "[" + std::to_string(log_count) + "] " + message;
    }

    message += "\n";

    return message;
}

Logger::~Logger() {
    file_stream.close();
}
```

Название файла: GameLogic.cpp

```
#include "GameLogic.h"

GameLogic::GameLogic() {
    strategy_list[kStandart] = new MoveStrategyStandart();
    strategy_list[kChase] = new MoveStrategyChase();
    strategy_list[kConfusion] = new MoveStrategyConfusion();
}

void GameLogic::CreateRoom(int id, int is_new_or_load) {
    // 0 - new; not 0 - load
    if (!is_new_or_load) {
        QFile
room_file(QStringLiteral(":/Rooms/Data/Rooms/%1_room.txt").arg(id));
        curr_room_ = new Room(id, room_file);
        room_file.close();

        LogInfo(LogMode::kFile, LogStyle::kTime,
                "Room created!\n|-->Width: " + std::to_string(curr_room_-
>getWidth()) +
                "\n|-->Height: " + std::to_string(curr_room_-
>getHeight()));

        QFile
objects_file(QStringLiteral(":/RoomObjects/Data/RoomObjects/%1_objects.tx
t").arg(id));
        LoadCellObjects(objects_file);
        objects_file.close();
    }
    NotifyRoomChange();
    NotifyRoomObjectsChange();
}

Room& GameLogic::getRoom() {
    return *curr_room_;
}

std::vector<PlaceableInCell*>& GameLogic::getRoomObjects() {
    return room_objects_;
}

void GameLogic::LoadCellObjects(QFile& obj_file) {
    if (obj_file.exists() && obj_file.open(QIODevice::ReadOnly)) {
        QString curr_line = obj_file.readLine();
```

```
        QStringList list_param;
        int num_objects = curr_line.toInt();
        PlaceableInCell* curr_obj;
        for (int i=0; i < num_objects; i++) {
            curr_line = obj_file.readLine();
            list_param = curr_line.split(" ");
            switch(list_param[0].toInt()) {
            case(1):
                curr_room_->getRoomCell(list_param[1].toInt(),
list_param[2].toInt()).setObject(\
                        &Player::getInstance());
                Player::getInstance().Move(Pos{list_param[1].toInt(),
list_param[2].toInt()});
                Player::getInstance().setSanity(list_param[3].toInt());
                Player::getInstance().damage_to_enemy_ =
list_param[4].toInt();
                room_objects_.push_back(&Player::getInstance());

                LogInfo(LogMode::kFile, LogStyle::kNumbered |
LogStyle::kTime,
                        "Player added: \n|-->Position: (" +
std::to_string(list_param[1].toInt())
                        + ", " + std::to_string(list_param[2].toInt()) +
")" +
                        "\n|-->Sanity: " +
std::to_string(list_param[3].toInt()) +
                        "\n|-->Damage to enemies: " +
std::to_string(list_param[4].toInt()));

                break;
            case(2):
                switch(list_param[1].toInt()) {
                case(1):
                    curr_obj = new DamageableEnemy(list_param[2].toInt(),
Pos{list_param[3].toInt(), list_param[4].toInt()},
                                list_param[5].toInt(),
list_param[6].toInt(),
strategy_list[(MoveStrategies)list_param[7].toInt()]);

                    LogInfo(LogMode::kFile, LogStyle::kNumbered |
LogStyle::kTime,
                            "DamageableEnemy added: \n|-->Position: (" +
std::to_string(list_param[3].toInt())
                            + ", " +
std::to_string(list_param[4].toInt()) + ")" +
                            "\n|-->Image id: " +
std::to_string(list_param[2].toInt()) +
                            "\n|-->Attack: " +
std::to_string(list_param[5].toInt()) +
                            "\n|-->Life: " +
std::to_string(list_param[6].toInt()));

                    break;
                case(2):
                    curr_obj = new
LimitedLifeEnemy(list_param[2].toInt(), Pos{list_param[3].toInt(),
list_param[4].toInt()},
```

```
                                    list_param[5].toInt(),
list_param[6].toInt(),
strategy_list[(MoveStrategies)list_param[7].toInt()]);
                    LogInfo(LogMode::kFile, LogStyle::kNumbered |
LogStyle::kTime,
                                "LimitedLifeEnemy added: \n|-->Position: (" +
std::to_string(list_param[3].toInt())
                                + ", " +
std::to_string(list_param[4].toInt()) + ")" +
                                "\n|-->Image id: " +
std::to_string(list_param[2].toInt()) +
                                "\n|-->Attack: " +
std::to_string(list_param[5].toInt()) +
                                "\n|-->Life: " +
std::to_string(list_param[6].toInt()));
                    break;
                case(3):
                    curr_obj = new ImmortalEnemy(list_param[2].toInt(),
Pos{list_param[3].toInt(), list_param[4].toInt()},
                                    list_param[5].toInt(),
strategy_list[(MoveStrategies)list_param[6].toInt()]);

                    LogInfo(LogMode::kFile, LogStyle::kNumbered |
LogStyle::kTime,
                                "ImmortalEnemy added: \n|-->Position: (" +
std::to_string(list_param[3].toInt())
                                + ", " +
std::to_string(list_param[4].toInt()) + ")" +
                                "\n|-->Image id: " +
std::to_string(list_param[2].toInt()) +
                                "\n|-->Attack: " +
std::to_string(list_param[5].toInt()) +
                                "\n|-->Life: Inf");

                    break;
                }
                break;
            case(3):
                switch(list_param[1].toInt()) {
                case(1):
                    curr_obj = new
SanityAffectItem(list_param[2].toInt(), Pos{list_param[3].toInt(),
list_param[4].toInt()},
                                    list_param[5].toInt());
                    LogInfo(LogMode::kFile, LogStyle::kNumbered |
LogStyle::kTime,
                                "SanityAffectItem added: \n|-->Position: (" +
std::to_string(list_param[3].toInt())
                                + ", " +
std::to_string(list_param[4].toInt()) + ")" +
                                "\n|-->Image id: " +
std::to_string(list_param[2].toInt()) +
                                "\n|-->Sanity heal: " +
std::to_string(list_param[5].toInt()));
                    break;
                case(2):
```

```
                    curr_obj = new
DamageToEnemiesItem(list_param[2].toInt(), Pos{list_param[3].toInt(),
list_param[4].toInt()},
                                        list_param[5].toInt());

                    LogInfo(LogMode::kFile, LogStyle::kNumbered |
LogStyle::kTime,
                                    "DamageToEnemiesItem added: \n|-->Position:
(" + std::to_string(list_param[3].toInt())
                                    + ", " +
std::to_string(list_param[4].toInt()) + ")" +
                                    "\n|-->Image id: " +
std::to_string(list_param[2].toInt()) +
                                    "\n|-->Damage up: " +
std::to_string(list_param[5].toInt()));

                    break;
                case(3):
                    curr_obj = new KeyItem(list_param[2].toInt(),
Pos{list_param[3].toInt(), list_param[4].toInt()});

                    LogInfo(LogMode::kFile, LogStyle::kNumbered |
LogStyle::kTime,
                                    "KeyItem added: \n|-->Position: (" +
std::to_string(list_param[3].toInt())
                                    + ", " +
std::to_string(list_param[4].toInt()) + ")" +
                                    "\n|-->Id: " +
std::to_string(list_param[2].toInt()));

                    break;
                }
                break;
            }
            if (list_param[0].toInt() != 1) {
                room_objects_.push_back(curr_obj);
                curr_room_->getRoomCell(list_param[3].toInt(),
list_param[4].toInt()).setObject(curr_obj);
            }
        }
    }
}

void GameLogic::MakeTurn(Pos player_pos_change) {
    Pos last_pos, curr_pos;
    for (int i=0; i < room_objects_.size(); i++) {
        last_pos = room_objects_[i]->getPos();
        if (room_objects_[i] == &Player::getInstance()) {
            room_objects_[i]->Move(Pos{last_pos.x + player_pos_change.x,
last_pos.y + player_pos_change.y});
        } else {
            room_objects_[i]->Move();
        }
        curr_pos = room_objects_[i]->getPos();
        room_objects_[i]->setPos(last_pos);

        if (curr_pos.x == last_pos.x && curr_pos.y == last_pos.y) {
            continue;
```

```
        }

        if (curr_pos.x < curr_room_->getWidth() && curr_pos.x >= 0 && \
                curr_pos.y < curr_room_->getHeight() && curr_pos.y >= 0)
{
            if (curr_room_->getRoomCell(curr_pos).getObject() != nullptr)
{
                PlaceableInCell* interact_obj = curr_room_-
>getRoomCell(curr_pos).getObject();
                Interactions interaction = room_objects_[i]-
>Interact(interact_obj);

                curr_room_->getRoomCell(room_objects_[i]-
>getPos()).setObject(room_objects_[i]);
                curr_room_->getRoomCell(interact_obj-
>getPos()).setObject(interact_obj);

                if (interaction == kI_interact_P) {
                    curr_room_->getRoomCell(last_pos).setObject(nullptr);
                    delete room_objects_[i];
                    room_objects_.erase(room_objects_.begin() + i);
                    NotifyObjectDestroy(i);

                    LogInfo(LogMode::kFile, LogStyle::kTime,
                            "Player interact with item at position: (" +
std::to_string(last_pos.x) +
                            ", " + std::to_string(last_pos.y) + ")");

                } else if (interaction == kP_interact_I) {
                    auto interact_obj_pos =
std::find(room_objects_.begin(), room_objects_.end(), interact_obj);
                    curr_room_->getRoomCell(last_pos).setObject(nullptr);
                    curr_room_->getRoomCell(room_objects_[i]-
>getPos()).setObject(room_objects_[i]);
                    delete interact_obj;
                    room_objects_.erase(interact_obj_pos);
                    NotifyObjectDestroy(interact_obj_pos -
room_objects_.begin());

                    LogInfo(LogMode::kFile, LogStyle::kTime,
                            "Player interact with item at position: (" +
std::to_string(room_objects_[i]->getPos().x) +
                            ", " + std::to_string(room_objects_[i]-
>getPos().y) + ")");

                } else {

                    LogInfo(LogMode::kFile, LogStyle::kTime,
                            "Object interact another object, positions:
(" + std::to_string(room_objects_[i]->getPos().x) +
                            ", " + std::to_string(room_objects_[i]-
>getPos().y) + ") <--> (" +
                            std::to_string(interact_obj->getPos().x) + ",
" + std::to_string(interact_obj->getPos().y) + ")");

                }
            } else {
                if (room_objects_[i] == &Player::getInstance()) {
```

```
                    bool end = curr_room_-
>getRoomCell(curr_pos).PlayerInteract(Player::getInstance());
                    if (Player::getInstance().getPos() == curr_pos) {
                        room_objects_[i]->setPos(curr_pos);
                        curr_room_-
>getRoomCell(curr_pos).setObject(room_objects_[i]);
                        curr_room_-
>getRoomCell(last_pos).setObject(nullptr);
                    }
                    if (end) {
                        NotifyWin();
                    }
                } else {
                    room_objects_[i]->setPos(curr_pos);
                    curr_room_-
>getRoomCell(curr_pos).setObject(room_objects_[i]);
                    curr_room_->getRoomCell(last_pos).setObject(nullptr);
                }
            }
        }
    }
    NotifyTurnComplete();
}

GameLogic::~GameLogic() {
    auto it = strategy_list.begin();
    while (it != strategy_list.end()) {
        delete it->second;
        it = strategy_list.erase(it);
    }
    delete curr_room_;

    delete &Logger::getInstance();
}
```

Название файла: GameView.cpp

```
#include "GameView.h"
#include "ui_GameView.h"

GameView::GameView(GameLogic* new_logic, QWidget *parent) :
    QWidget(parent), ui(new Ui::GameView) {
    ui->setupUi(this);
    curr_logic_ = new_logic;
    curr_logic_->setObserver(this);

    room_scene = new QGraphicsScene(this);
    ui->room_view->setScene(room_scene);
    ui->room_view->scale(0.4, 0.4);
    ui->room_view->setFocusPolicy(Qt::NoFocus);
    ui->room_view->setRenderHint(QPainter::Antialiasing);
    ui->room_view->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);
    ui->room_view->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);

    test_info = new QLabel(this);
    test_info->setGeometry(0, 0, 500, 700);
    test_info->setStyleSheet("QLabel {color: white;}");
```

```
}

void GameView::resizeEvent(QResizeEvent* event) {
    QWidget::resizeEvent(event);

    LogInfo(LogMode::kFile, LogStyle::kTime | LogStyle::kCapitalise,
            "Window resized");
}

void GameView::keyPressEvent(QKeyEvent* event) {
    // пока что жестко закреплены в коде, потом будут изменены
    if (event->key() == Qt::Key_Up) {
        emit PlayerPressMove(Pos{0, -1});
    } else if (event->key() == Qt::Key_Down) {
        emit PlayerPressMove(Pos{0, 1});
    } else if (event->key() == Qt::Key_Left) {
        emit PlayerPressMove(Pos{-1, 0});
    } else if (event->key() == Qt::Key_Right) {
        emit PlayerPressMove(Pos{1, 0});
    }
}

void GameView::UpdateRoomChange() {
    DeleteRoom();
    Room& curr_room = curr_logic_->getRoom();
    int width_room = curr_room.getWidth();
    int height_room = curr_room.getHeight();

    QPixmap
floor_img(QStringLiteral(":/Img/Data/Img/Floor/%1_floor.jpg").arg(curr_ro
om.getId()));
    QPixmap ex-
it_img(QStringLiteral(":/Img/Data/Img/Floor/%1_exit.jpg").arg(curr_room.g
etId()));
    QPixmap en-
trance_img(QStringLiteral(":/Img/Data/Img/Floor/%1_entrance.jpg").arg(cur
r_room.getId()));

    floor_img = floor_img.scaled(floor_cell_size_, floor_cell_size_,
Qt::KeepAspectRatio);
    exit_img = exit_img.scaled(floor_cell_size_, floor_cell_size_,
Qt::KeepAspectRatio);
    entrance_img = entrance_img.scaled(floor_cell_size_,
floor_cell_size_, Qt::KeepAspectRatio);

    for (int x = 0; x < width_room; x++) {
        for (int y = 0; y < height_room; y++) {
            QPixmap chosen_img;
            if (typeid(curr_room.getRoomCell(x, y)) == typeid(ExitCell))
{
                chosen_img = exit_img;
            } else if (typeid(curr_room.getRoomCell(x, y)) ==
typeid(EntranceCell)) {
                chosen_img = entrance_img;
            } else {
                chosen_img = floor_img;
            }
```

```
            QGraphicsPixmapItem* floor_item = room_scene-
>addPixmap(chosen_img);
            floor_item->setPos(x*floor_cell_size_, y*floor_cell_size_ +
top_padding_scene_);
        }
    }
}

void GameView::UpdateRoomObjectsChange() {
    ClearObjects();
    auto room_obj = curr_logic_->getRoomObjects();
    for (int i=0; i < room_obj.size(); i++) {
        draw_objects_.push_back(new DrawableObject(room_scene,
room_obj[i]->getId(), room_obj[i]->getPos(), floor_cell_size_,
top_padding_scene_));
    }
    CenterViewOnPlayerPos();
}

void GameView::UpdateObjectDestroy(int i) {
    room_scene->removeItem(draw_objects_[i]);
    delete draw_objects_[i];
    draw_objects_.erase(draw_objects_.begin() + i);
}

void GameView::UpdateTurnComplete() {
    Room& curr_room = curr_logic_->getRoom();
    int width_room = curr_room.getWidth();
    int height_room = curr_room.getHeight();

    for (int x = 0; x < width_room; x++) {
        for (int y = 0; y < height_room; y++) {
            if (curr_room.getRoomCell(x, y).getObject()) {
                int i = std::find_if(curr_logic_-
>getRoomObjects().begin(), curr_logic_->getRoomObjects().end(),
                [x, y](PlaceableInCell* curr_obj) {
                    return curr_obj->getPos().x == x && curr_obj-
>getPos().y == y;
                }) - curr_logic_->getRoomObjects().begin();

                QPropertyAnimation* anim = new QPropertyAnima-
tion(draw_objects_[i], "getScenePos");

                anim->setDuration(200);
                anim->setEndValue(draw_objects_[i]-
>RoomPosToScenePos(curr_logic_->getRoomObjects()[i]->getPos()));
                anim->start(QAbstractAnimation::DeleteWhenStopped);
            }
        }
    }
    CenterViewOnPlayerPos();
    TestShowPlayerConditions();
}

void GameView::UpdateWin() {
    QLabel* win = new QLabel("You win!");
    win->show();
}
```

```
void GameView::UpdateDefeat() {
    QLabel* defeat = new QLabel("You lose... your sanity ;)");
    defeat->show();
}

void GameView::TestShowPlayerConditions() {
    QString info = QStringLiteral("Player: "\
    "Sanity: %1\nDamage_to_enemies: %2\nNum_key_items:
%3\n\n").arg(Player::getInstance().getSanity()

).arg(Player::getInstance().damage_to_enemy_).arg(Player::getInstance().g
etNumItems());

    test_info->setText(info);
}

void GameView::CenterViewOnPlayerPos() {
    Pos player_pos = Player::getInstance().getPos();
    ui->room_view->centerOn(floor_cell_size_* player_pos.x,
floor_cell_size_* player_pos.y + floor_cell_size_);
}

void GameView::DeleteRoom() {
    ClearObjects();
    room_scene->clear();
}

void GameView::ClearObjects() {
    for (int i=0; i < draw_objects_.size(); i++) {
        room_scene->removeItem(draw_objects_[i]);
        delete draw_objects_[i];
    }

    draw_objects_.clear();
}

GameView::~GameView() {
    DeleteRoom();
    delete room_scene;
    delete ui;
}
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

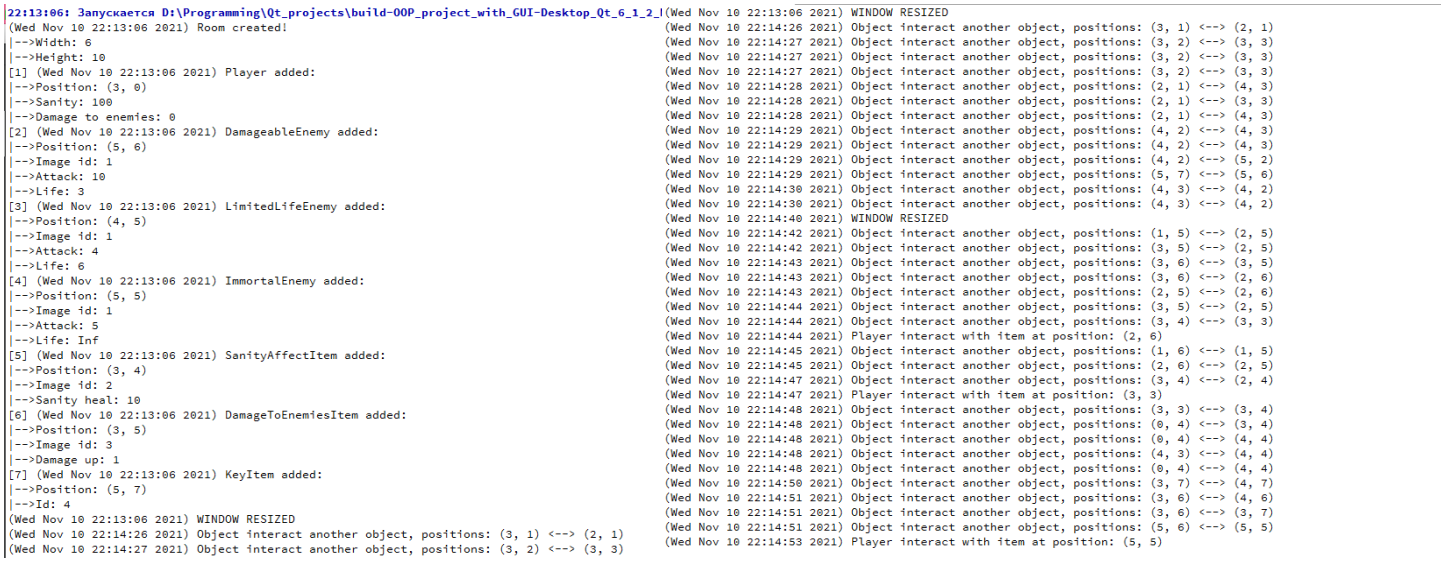
Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Выполняемые операции	Результат	Комментарии
1.	Логирование в консоль в начале игры [логируются *GameLogic* и *GameView*].	См. рис. 1	Верно
2.	Логирование в консоль, несколько ходов прошло [логируются *GameLogic* и *GameView*].	См. рис. 2	Верно
3.	Логирование в файл (открыт после завершения игры) [логируются *GameLogic* и *GameView*].	См. рис. 3	Верно
4.	Логирование в файл и в консоль всей информации [логируются *GameLogic* и *GameView*].	См. рис. 4	Верно
5.	Смена формата логов для события изменения экрана приложения + прекращение отслеживания GameLogic изменений [логируется *GameView*].	См. рис. 5	Верно
6.	Смена всех форматов для GameLogic + прекращение отслеживания *GameView* изменений [логируется GameLogic].	См. рис. 6	Верно

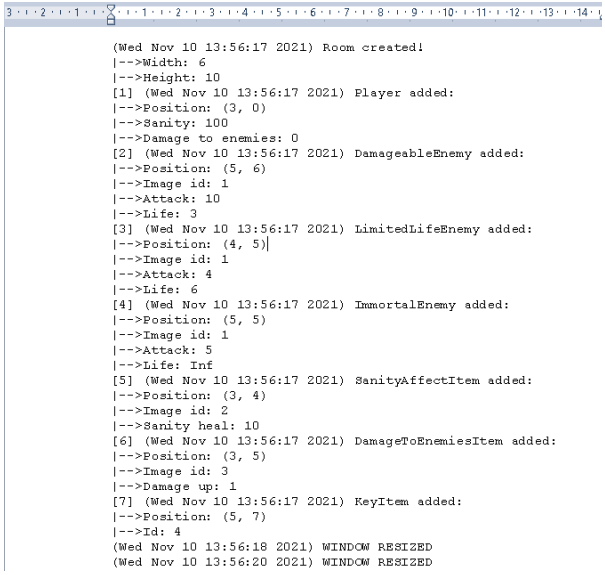
```
22:13:06: Запускается D:\Programming\Qt_projects\build-OOP_project_with_GUI-Desktop_Qt_6_1_2_MinGW_64_bit-Debug\debug\OOP_project_with_GUI.exe ...
(Wed Nov 10 22:13:06 2021) Room created!
|-->Width: 6
|-->Height: 10
[1] (Wed Nov 10 22:13:06 2021) Player added:
|-->Position: (3, 0)
|-->Sanity: 100
|-->Damage to enemies: 0
[2] (Wed Nov 10 22:13:06 2021) DamageableEnemy added:
|-->Position: (5, 6)
|-->Image id: 1
|-->Attack: 10
|-->Life: 3
[3] (Wed Nov 10 22:13:06 2021) LimitedLifeEnemy added:
|-->Position: (4, 5)
|-->Image id: 1
|-->Attack: 4
|-->Life: 6
[4] (Wed Nov 10 22:13:06 2021) ImmortalEnemy added:
|-->Position: (5, 5)
|-->Image id: 1
|-->Attack: 5
|-->Life: Inf
[5] (Wed Nov 10 22:13:06 2021) SanityAffectItem added:
|-->Position: (3, 4)
|-->Image id: 2
|-->Sanity heal: 10
[6] (Wed Nov 10 22:13:06 2021) DamageToEnemiesItem added:
|-->Position: (3, 5)
|-->Image id: 3
|-->Damage up: 1
[7] (Wed Nov 10 22:13:06 2021) KeyItem added:
|-->Position: (5, 7)
|-->Id: 4
(Wed Nov 10 22:13:06 2021) WINDOW RESIZED
```

Рисунок 1 - Лог, консоль, старт игры



```
22:13:06: Запускается D:\Programming\Qt_projects\build-OOP_project_with_GUI-Desktop_Qt_6_1_2_
(Wed Nov 10 22:13:06 2021) Room created!
|-->Width: 6
|-->Height: 10
[1] (Wed Nov 10 22:13:06 2021) Player added:
|-->Position: (3, 0)
|-->Sanity: 100
|-->Damage to enemies: 0
[2] (Wed Nov 10 22:13:06 2021) DamageableEnemy added:
|-->Position: (5, 6)
|-->Image id: 1
|-->Attack: 10
|-->Life: 3
[3] (Wed Nov 10 22:13:06 2021) LimitedLifeEnemy added:
|-->Position: (4, 5)
|-->Image id: 1
|-->Attack: 4
|-->Life: 6
[4] (Wed Nov 10 22:13:06 2021) ImmortalEnemy added:
|-->Position: (5, 5)
|-->Image id: 1
|-->Attack: 5
|-->Life: Inf
[5] (Wed Nov 10 22:13:06 2021) SanityAffectItem added:
|-->Position: (3, 4)
|-->Image id: 2
|-->Sanity heal: 10
[6] (Wed Nov 10 22:13:06 2021) DamageToEnemiesItem added:
|-->Position: (3, 5)
|-->Image id: 3
|-->Damage up: 1
[7] (Wed Nov 10 22:13:06 2021) KeyItem added:
|-->Position: (5, 7)
|-->Id: 4
(Wed Nov 10 22:13:06 2021) WINDOW RESIZED
(Wed Nov 10 22:14:26 2021) Object interact another object, positions: (3, 1) <--> (2, 1)
(Wed Nov 10 22:14:27 2021) Object interact another object, positions: (3, 2) <--> (3, 3)
(Wed Nov 10 22:13:06 2021) WINDOW RESIZED
(Wed Nov 10 22:14:26 2021) Object interact another object, positions: (3, 1) <--> (2, 1)
(Wed Nov 10 22:14:27 2021) Object interact another object, positions: (3, 2) <--> (3, 3)
(Wed Nov 10 22:14:27 2021) Object interact another object, positions: (3, 2) <--> (3, 3)
(Wed Nov 10 22:14:27 2021) Object interact another object, positions: (3, 2) <--> (3, 3)
(Wed Nov 10 22:14:28 2021) Object interact another object, positions: (2, 1) <--> (4, 3)
(Wed Nov 10 22:14:28 2021) Object interact another object, positions: (2, 1) <--> (3, 3)
(Wed Nov 10 22:14:28 2021) Object interact another object, positions: (2, 1) <--> (4, 3)
(Wed Nov 10 22:14:29 2021) Object interact another object, positions: (4, 2) <--> (4, 3)
(Wed Nov 10 22:14:29 2021) Object interact another object, positions: (4, 2) <--> (4, 3)
(Wed Nov 10 22:14:29 2021) Object interact another object, positions: (4, 2) <--> (5, 2)
(Wed Nov 10 22:14:29 2021) Object interact another object, positions: (5, 7) <--> (5, 6)
(Wed Nov 10 22:14:30 2021) Object interact another object, positions: (4, 3) <--> (4, 2)
(Wed Nov 10 22:14:30 2021) Object interact another object, positions: (4, 3) <--> (4, 2)
(Wed Nov 10 22:14:40 2021) WINDOW RESIZED
(Wed Nov 10 22:14:42 2021) Object interact another object, positions: (1, 5) <--> (2, 5)
(Wed Nov 10 22:14:42 2021) Object interact another object, positions: (3, 5) <--> (2, 5)
(Wed Nov 10 22:14:43 2021) Object interact another object, positions: (3, 6) <--> (3, 5)
(Wed Nov 10 22:14:43 2021) Object interact another object, positions: (3, 6) <--> (2, 6)
(Wed Nov 10 22:14:43 2021) Object interact another object, positions: (2, 5) <--> (2, 6)
(Wed Nov 10 22:14:44 2021) Object interact another object, positions: (3, 5) <--> (2, 5)
(Wed Nov 10 22:14:44 2021) Object interact another object, positions: (3, 4) <--> (3, 3)
(Wed Nov 10 22:14:44 2021) Player interact with item at position: (2, 6)
(Wed Nov 10 22:14:45 2021) Object interact another object, positions: (1, 6) <--> (1, 5)
(Wed Nov 10 22:14:45 2021) Object interact another object, positions: (2, 6) <--> (2, 5)
(Wed Nov 10 22:14:47 2021) Object interact another object, positions: (3, 4) <--> (2, 4)
(Wed Nov 10 22:14:47 2021) Player interact with item at position: (3, 3)
(Wed Nov 10 22:14:48 2021) Object interact another object, positions: (3, 3) <--> (3, 4)
(Wed Nov 10 22:14:48 2021) Object interact another object, positions: (0, 4) <--> (3, 4)
(Wed Nov 10 22:14:48 2021) Object interact another object, positions: (0, 4) <--> (4, 4)
(Wed Nov 10 22:14:48 2021) Object interact another object, positions: (4, 3) <--> (4, 4)
(Wed Nov 10 22:14:48 2021) Object interact another object, positions: (0, 4) <--> (4, 4)
(Wed Nov 10 22:14:50 2021) Object interact another object, positions: (3, 7) <--> (4, 7)
(Wed Nov 10 22:14:51 2021) Object interact another object, positions: (3, 6) <--> (4, 6)
(Wed Nov 10 22:14:51 2021) Object interact another object, positions: (3, 6) <--> (3, 7)
(Wed Nov 10 22:14:51 2021) Object interact another object, positions: (5, 6) <--> (5, 5)
(Wed Nov 10 22:14:53 2021) Player interact with item at position: (5, 5)
```

Рисунок 2 - Лог, консоль, продолжение игры



```
(Wed Nov 10 13:56:17 2021) Room created!
|-->Width: 6
|-->Height: 10
[1] (Wed Nov 10 13:56:17 2021) Player added:
|-->Position: (3, 0)
|-->Sanity: 100
|-->Damage to enemies: 0
[2] (Wed Nov 10 13:56:17 2021) DamageableEnemy added:
|-->Position: (5, 6)
|-->Image id: 1
|-->Attack: 10
|-->Life: 3
[3] (Wed Nov 10 13:56:17 2021) LimitedLifeEnemy added:
|-->Position: (4, 5)
|-->Image id: 1
|-->Attack: 4
|-->Life: 6
[4] (Wed Nov 10 13:56:17 2021) ImmortalEnemy added:
|-->Position: (5, 5)
|-->Image id: 1
|-->Attack: 5
|-->Life: Inf
[5] (Wed Nov 10 13:56:17 2021) SanityAffectItem added:
|-->Position: (3, 4)
|-->Image id: 2
|-->Sanity heal: 10
[6] (Wed Nov 10 13:56:17 2021) DamageToEnemiesItem added:
|-->Position: (3, 5)
|-->Image id: 3
|-->Damage up: 1
[7] (Wed Nov 10 13:56:17 2021) KeyItem added:
|-->Position: (5, 7)
|-->Id: 4
(Wed Nov 10 13:56:18 2021) WINDOW RESIZED
(Wed Nov 10 13:56:20 2021) WINDOW RESIZED
(Wed Nov 10 13:56:21 2021) Object interact another object,
positions: (3, 5) <--> (4, 5)
(Wed Nov 10 13:56:22 2021) Object interact another object,
positions: (3, 4) <--> (3, 5)
(Wed Nov 10 13:56:24 2021) Object interact another object,
positions: (1, 4) <--> (2, 4)
(Wed Nov 10 13:56:25 2021) Player interact with item at
position: (3, 5)
(Wed Nov 10 13:56:25 2021) Player interact with item at
position: (4, 5)
(Wed Nov 10 13:56:26 2021) Object interact another object,
positions: (5, 4) <--> (4, 4)
(Wed Nov 10 13:56:26 2021) Object interact another object,
positions: (5, 4) <--> (4, 4)
(Wed Nov 10 13:56:27 2021) Player interact with item at
position: (5, 7)
(Wed Nov 10 13:56:31 2021) Object interact another object,
positions: (4, 3) <--> (3, 3)
(Wed Nov 10 13:56:32 2021) Object interact another object,
positions: (3, 3) <--> (2, 3)
(Wed Nov 10 13:56:32 2021) Object interact another object,
positions: (3, 3) <--> (2, 3)
(Wed Nov 10 13:56:36 2021) Object interact another object,
positions: (2, 5) <--> (3, 7)
(Wed Nov 10 13:56:39 2021) Object interact another object,
positions: (3, 3) <--> (4, 5)
(Wed Nov 10 13:56:40 2021) Object interact another object,
positions: (5, 5) <--> (5, 4)
```

Рисунок 3 - Лог, файл, продолжение игры

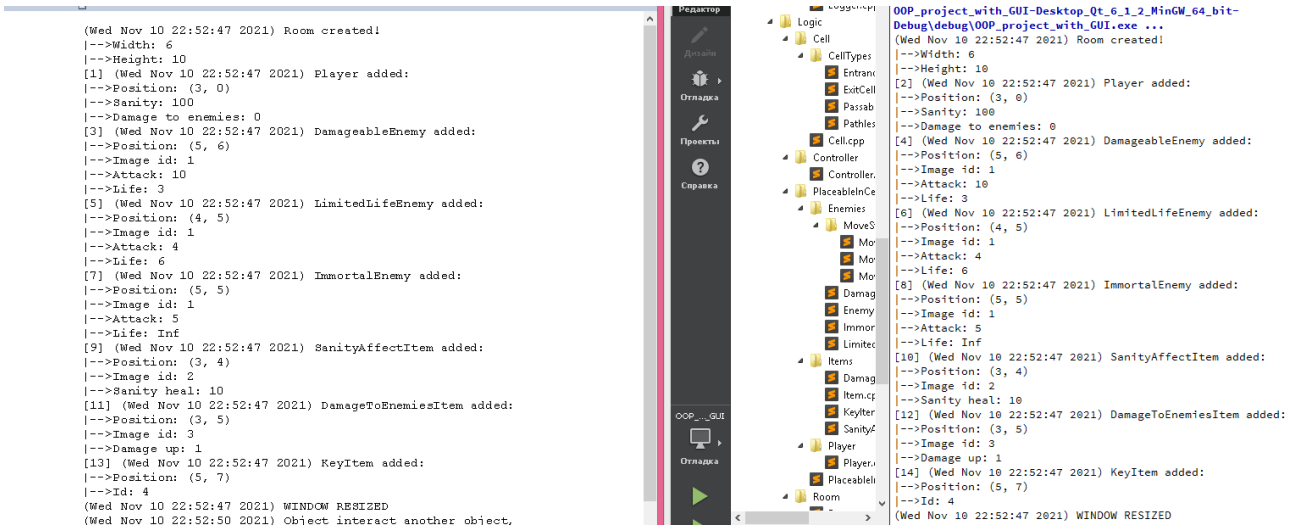


Рисунок 4 - Часть лога, файл и консоль

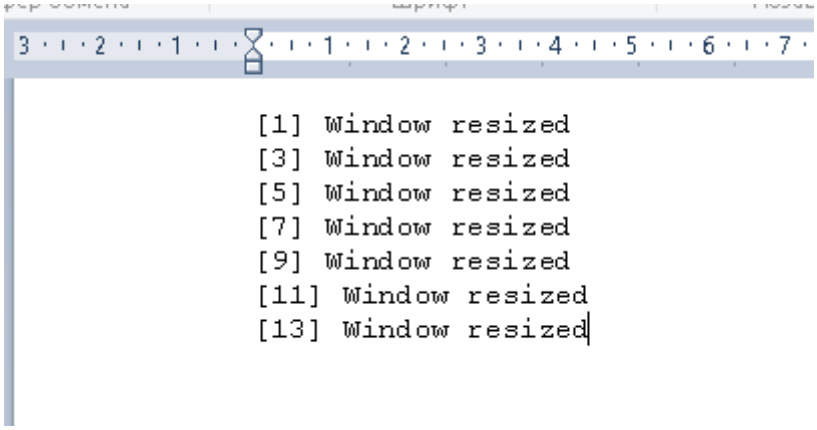


Рисунок 5 - Лог, отслеживание только GameView, файл

```
DamageableEnemy added:
|-->Position: (5, 6)
|-->Image id: 1
|-->Attack: 10
|-->Life: 3
LimitedLifeEnemy added:
|-->Position: (4, 5)
|-->Image id: 1
|-->Attack: 4
|-->Life: 6
ImmortalEnemy added:
|-->Position: (5, 5)
|-->Image id: 1
|-->Attack: 5
|-->Life: Inf
SanityAffectItem added:
|-->Position: (3, 4)
|-->Image id: 2
|-->Sanity heal: 10
DamageToEnemiesItem added:
|-->Position: (3, 5)
|-->Image id: 3
|-->Damage up: 1
KeyItem added:
|-->Position: (5, 7)
|-->Id: 4
Player interact with item at position: (3, 4)
Object interact another object, positions: (3, 5) <--> (3, 4)
Object interact another object, positions: (4, 5) <--> (3, 5)
Player interact with item at position: (3, 4)
Object interact another object, positions: (4, 4) <--> (5, 4)
Object interact another object, positions: (3, 4) <--> (3, 5)
Object interact another object, positions: (5, 7) <--> (5, 6)
Object interact another object, positions: (3, 3) <--> (3, 4)
Object interact another object, positions: (2, 4) <--> (3, 4)
Object interact another object, positions: (2, 4) <--> (3, 5)
Object interact another object, positions: (2, 4) <--> (3, 6)
Player interact with item at position: (5, 6)
```

Рисунок 6 - Лог, отслеживание только GameLogic, консоль

ПРИЛОЖЕНИЕ В
UML-ДИАГРАММА

UML-диаграмма классов представлена на рис. 5: