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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Объектно-ориентированное программирование»

Тема: Логирование, перегрузка операций.

Студент гр. 1304 _____ Павлов Д.Р.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы.

Реализовать класс/набор классов отслеживающих изменения состояний в программе. Отслеживание должно быть 3-х уровней:

1. Изменения состояния игрока и поля, а также срабатывание событий

2. Состояние игры (игра начата, завершена, сохранена, и.т.д.)

3. Отслеживание критических состояний и ошибок (поле инициализировано с

отрицательными размерами, игрок попытался перейти на непроходимую клетку,

и.т.д.)

Реализованы классы для вывода информации разных уровней для в консоль и в файл с перегруженным оператором вывода в поток.

Требования.

Разработан класс/набор классов отслеживающий изменения разных уровней

Разработаны классы для вывода в консоль и файл с соблюдением идиомы RAII

и перегруженным оператором вывода в поток.

Разработанные классы спроектированы таким образом, чтобы можно было добавить новый формат вывода без изменения старого кода (например, добавить возможность отправки логов по сети)

Выбор отслеживаемых уровней логирования должен происходить в runtime

В runtime должен выбираться способ вывода логов (нет логирования, в консоль,

в файл, в консоль и файл)

Описание архитектурных решений и классов.

В данной лабораторной работе **был использован паттерн *Observer***.

Upd: так же было замечено, что программа удовлетворяет архитектурному паттерну MVC.

Описание классов:

1) ***Iobserver*** — Абстрактный Класс, описывающий поведение Класса-Наблюдателя. Данный класс имеет виртуальный метод *update*, при помощи которого идет обновление данных у Классов-Наблюдателей.

2) ***Isubject*** — Абстрактный Класс Объекта Наблюдения, описывающий поведение Класса-Объекта. Данный класс имеет три виртуальных метода: *Attach* (Подписывает субъект на наблюдение), *Detach* (Отписывает субъект от наблюдения), *Notify* (Оповестить всех наблюдателей о действиях субъекта); а так же хранит указатель на *Message* и вектор указателей на *Iobserver* (наблюдатели).

3) ***Message*** — Класс, отвечающий за передачу наблюдателям конкретные данные. Имеет метод виртуальный метод *get_message()*.

Message Childs — классы-наследники от ***Message***, отвечающие за передачу конкретного сообщения логам. Каждый потомок от ***Message*** отвечает конкретному уровню логирования (Например: ***Event_Message*** отвечает за логирования конкретных ивентов; ***Warning_Message*** — логирование ошибок) и перегружает метод родителя.

Список наблюдателей (Логгеров):

4) ***Console_Logger*** — Класс-Наследник от Iobserver'а, хранящий в себе строки с сообщениями логов.

5) ***File_Logger*** — Класс-Наследник от Iobserver'а, записывающий логи в текстовый файл Logs.txt.

Список Объектов Наблюдения:

-

~~*Objects_Generator*~~*, Hero_Moves, Request_Input_Stream*.

Демонстрация работы программы и тестирование.

(Изначально поток логирования стоит «File Only»)

Ходы Юзера:
1) Переключить поток логирования на «Console + File»
2) Собрать Опыт
3) Собрать ХП
4) Убить Врага (На поле только один враг)
5) Переключить поток логирования на «File Only»
6) Собрать ХП
7) Собрать опыт до нового уровня
8) Попробовать пройти через стену
9) Дойти до Победной Клетки

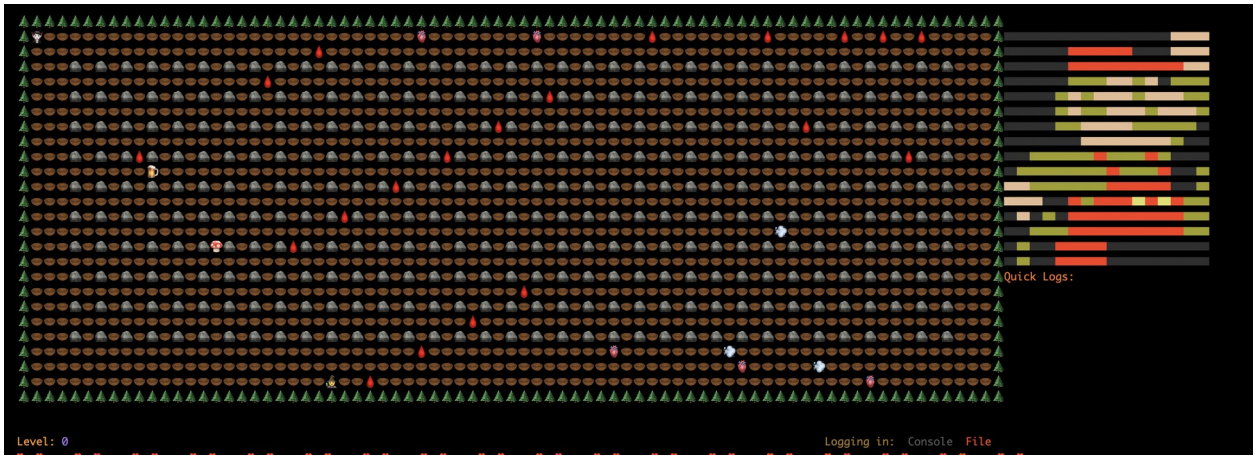


Рисунок 1 — Начало игры.

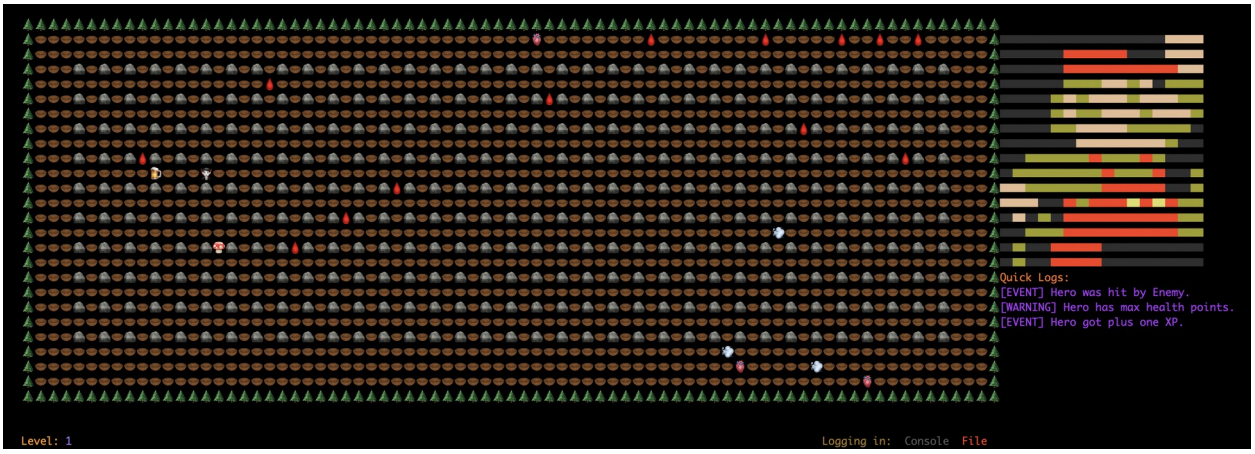


Рисунок 2 — Конец игры. (Герою осталось дойти до победной клетки)

Logs.txt

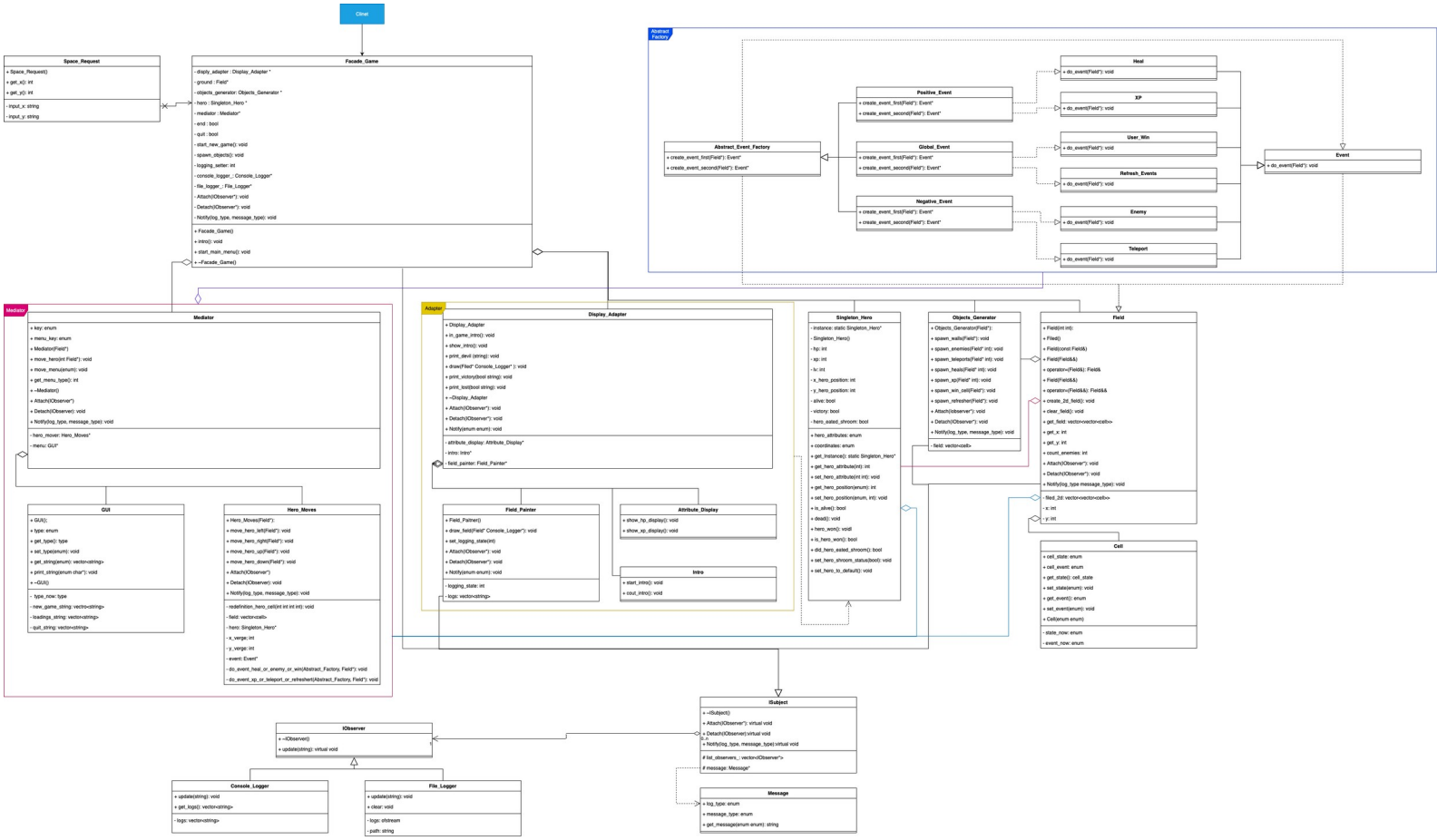
```
19:27:48  [GLOBAL] Intro was shown.
19:27:48  [SYSTEM] Have A Nice Life - Bloodhail is playing...
19:27:48  [GLOBAL] GUI Started.
19:27:52  [GLOBAL] New game is starting...
19:27:52  [SYSTEM] Music was switched off.
19:27:52  [SYSTEM] Joy Division - Disorder is playing...
19:27:58  [EVENT] Hero got plus one XP.
19:27:59  [WARNING] Hero has max health points.
19:28:2  [EVENT] Hero was hit by Enemy.
19:28:21  [EVENT] Hero was healed by Healing.
19:28:28  [EVENT] Hero got plus one XP.
19:28:31  [EVENT] Hero got plus one XP.
19:28:32  [EVENT] Hero got plus one XP.
19:28:33  [EVENT] Hero got plus one XP.
19:28:39  [EVENT] Hero got plus one XP.
19:28:42  [HERO] Hero reached New Level.
19:28:42  [EVENT] Hero got plus one XP.
19:28:47  [EVENT] Hero moved to the victory cell.
19:28:47  [HERO] Hero wins.
19:28:47  [SYSTEM] Music was switched off.
19:28:47  [SYSTEM] Have A Nice Life - Bloodhail is playing...
19:28:47  [GENERATOR] Walls generated...
19:28:47  [GENERATOR] Enemies spawned...
19:28:47  [GENERATOR] Teleports generated...
19:28:47  [GENERATOR] Heals generated...
19:28:47  [GENERATOR] Xps generated...
19:28:47  [GENERATOR] Refresher generated...
19:28:47  [GENERATOR] Victory Cell generated...
19:28:49  [GLOBAL] Bye!
```

Рисунок 3 — Файл Логов

Рисунок 4 — Панель Потока Логирования.

Рисунок 2 и 3 — явная демонстрация переключения Потоков Логирования.

UML Диаграмма Проекта



Вывод.

Реализована система логирования игрового процесса, в основе которой лежит паттерн Observer

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