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

ОТЧЕТ

по лабораторной работу №6

по дисциплине «Объектно-ориентированное программирование»

Тема: Сериализация, исключения

Студентка гр. 0382 _____ Деткова А.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Ознакомиться с сериализацией и исключениями. Продолжить писать игру.

Задание.

Сериализация - это сохранение в определенном виде состоянии программы с возможностью последующего его восстановления даже после закрытия программы. В рамках игры, это сохранения и загрузка игры.

Требования:

Реализовать сохранения всех необходимых состояний игры в файл

Реализовать загрузку файла сохранения и восстановления состояния игры

Должны быть возможность сохранить и загрузить игру в любой момент

При запуске игры должна быть возможность загрузить нужный файл

Написать набор исключений, который срабатывают если файл с сохранением некорректный

Исключения должны сохранять транзакционность. Если не удалось сделать загрузку, то программа должна находится в том состоянии, которое было до загрузки. То есть, состояние игры не должно загружаться частично

Потенциальные паттерны проектирования, которые можно использовать:

Снимок (Memento) - получение и восстановления состояния объектов при сохранении и загрузке

Выполнение работы.

В ходе работы был расширен класс игры и логики движения врагов, игрока и логики предметов.

У всех классов Logic (классы, реализующие логику; имя папки с классами) появились методы load(значения для заполнения полей) и save(), первый загружает данные в поля объекта класса (например, у всех объектов есть поле позиции, этот метод записывает это поле и многие другие поля) при загрузке сохранения, второй — сохраняет состояние и возвращает его в качестве строки. Эти данные заносятся в файл с сохранением.

Также подобные методы есть у классов Entities (классы, реализующие сущностей), они также записывают состояние или заполняют поля, полученными значениями, которые были сохранены.

У класса поля Field был реализованы методы save() и load(), save() сохраняет размеры поля, а также состояния всех клеток на поле в одну строку, load() получает на вход размеры поля и строку с состояниями, заполняет клетки соответствующими состояниями, поле объекта на клетке пустое, оно будет заполнено при загрузке состояний объектов.

Класс Save — выполняет сохранение и загрузку состояний объектов.

Методы: save(Field& field, PlayerLogic& player, EntityLogic** enemies, int enemies_quantity, ArmorLogic& armor, HealLogic& heal, WeaponLogic& weapon, KeyLogic* keys, bool* key_put, bool step) — получает поле, врагов, их количество, броню, оружие, лечение, ключи, игрока, булевые переменные положены ли на поле ключи и булевая переменная шага (чей сейчас ход). Получает состояния объектов в виде строк с помощью методов save() и записывает информацию в файл SaveFiel.txt. Информация файле: 1 строка — размер поля (высота и ширина), 2 строка — состояния клеток поля, 3 строка — информация об игроке: здоровье, сила удара, броня, количество ключей, количество убитых игроков, позиция на поле (x, y), жив ли игрок, находится

ли на поле. 3-5 строки — ключи: характеристика, позиция (x, y), существует ли объект, лежит ли на поле, 6-8 строки — аналогичные характеристики лечения, оружия и брони, 9 строка — количество врагов (число n), следующие n строк — информация о врагах — тип, здоровье, броня, позиция (x, y), жив ли, находится ли на поле, направление движения и текущий шаг (нужно, чтобы враг продолжал идти по намеченной траектории. Предпоследняя строка — 3 переменные, показывают положен ли ключ на поле (если 0, то нет, если 1 — да), последняя строка — очередь хода. В методе в виде строки получаем состояния объектов всех, что указаны в файле и записываем в файл. Метод load(Field &field, PlayerLogic &player, EntityLogic **enemies, int& number, ArmorLogic &armor, HealLogic &heal, WeaponLogic &weapon, KeyLogic *keys, bool *key_put, bool& step) соответственно считывает эту информацию с файла и заносит ее в поля объектов методами load(…). Также в методе размещаются объекты на поле в зависимости от того, существуют они или нет, подобраны игроком или нет, убиты ли враги. Метод tryLoad() - пытается считать файл, если в файле больше 17 строк, то файл может читаться, также проверят возможно ли открыть файл, и не пустой ли файл.

Чтобы сохранить состояние можно нажать enter в любой момент игры. Чтобы загрузить, можно нажать esc в любой момент игры, а если вначале, то ввести 1 после соответствующего приглашения от игры.

Выводы.

Была разработана следующая часть игры на языке C++.