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

ОТЧЕТ

по лабораторной работе №6

по дисциплине «ООП»

ТЕМА: СОХРАНЕНИЕ И ЗАГРУЗКА / НАПИСАНИЕ ИСКЛЮЧЕНИЙ

Студент гр. 9304 _____ Прокофьев М.Д.

Преподаватель _____ Шевская Н.В.

Санкт-Петербург

2020

Цель работы.

Обучение работе с классами, конструкторами на языке в C++.

Задание.

Создать классы, которые позволяют сохранить игру, а потом загрузить ее. Также, написать набор исключений, которые как минимум позволяют контролировать процесс сохранения и загрузки

Обязательные требования:

-Игру можно сохранить в файл

-Игру можно загрузить из файла

-Взаимодействие с файлами по идиоме RAII

-Добавлена проверка файлов на корректность

-Написаны исключения, которые обеспечивают транзакционность

Дополнительные требования:

-Для получения состояния программы используется паттерн Снимок

Основные теоретические положения.

Класс - это пользовательский тип данных.

Конструктор копирования используется для инициализации класса путем создания копии необходимого объекта.

Оператор присваивания копированием (или «копирующее присваивание») используется для копирования одного класса в другой (существующий) класс.

Конструкторы перемещения принимают ссылку на значение объекта класса и используются для реализации передачи владения ресурсами объекта параметра.

Метод в объектно-ориентированном программировании — это функция или процедура, принадлежащая какому-то классу или объекту. Как и процедура в процедурном программировании, метод состоит из некоторого количества операторов для выполнения какого-то действия и имеет набор входных аргументов.

Интерфейс — программная/синтаксическая структура, определяющая отношение между объектами, которые разделяют определённое множество и не связаны никак иначе. При проектировании классов, разработка интерфейса тождественна разработке спецификации (множества методов, которые должен реализовывать каждый класс, использующий интерфейс).

Перегрузка операторов позволяет определить действия, которые будет выполнять оператор. Перегрузка подразумевает создание функции, название которой содержит слово operator и символ перегружаемого оператора. Функция оператора может быть определена как член класса, либо вне класса.

Инвариант объекта представляет собой конструкцию программирования, состоящую из набора инвариантных свойств. Это гарантирует, что объект всегда будет соответствовать предопределенным условиям, и поэтому методы могут всегда ссылаться на объект без риска сделать неточные презумпции. Определение инвариантов классов может помочь программистам и тестировщикам обнаружить больше ошибок при тестировании программного обеспечения.

Шаблоны— средство языка C++, предназначенное для кодирования обобщённых алгоритмов, без привязки к некоторым параметрам (например, типам данных, размерам буферов, значениям по умолчанию).

RAII — программная идиома объектно-ориентированного программирования, смысл которой заключается в том, что с помощью тех или иных программных механизмов получение некоторого ресурса неразрывно совмещается с инициализацией, а освобождение — с уничтожением объекта.

Транзакция в программировании - группа последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными. Транзакция может быть выполнена либо целиком и успешно, соблюдая целостность данных и независимо от параллельно идущих других транзакций, либо не выполнена вообще, и тогда она не должна произвести никакого эффекта.

Выполнение работы.

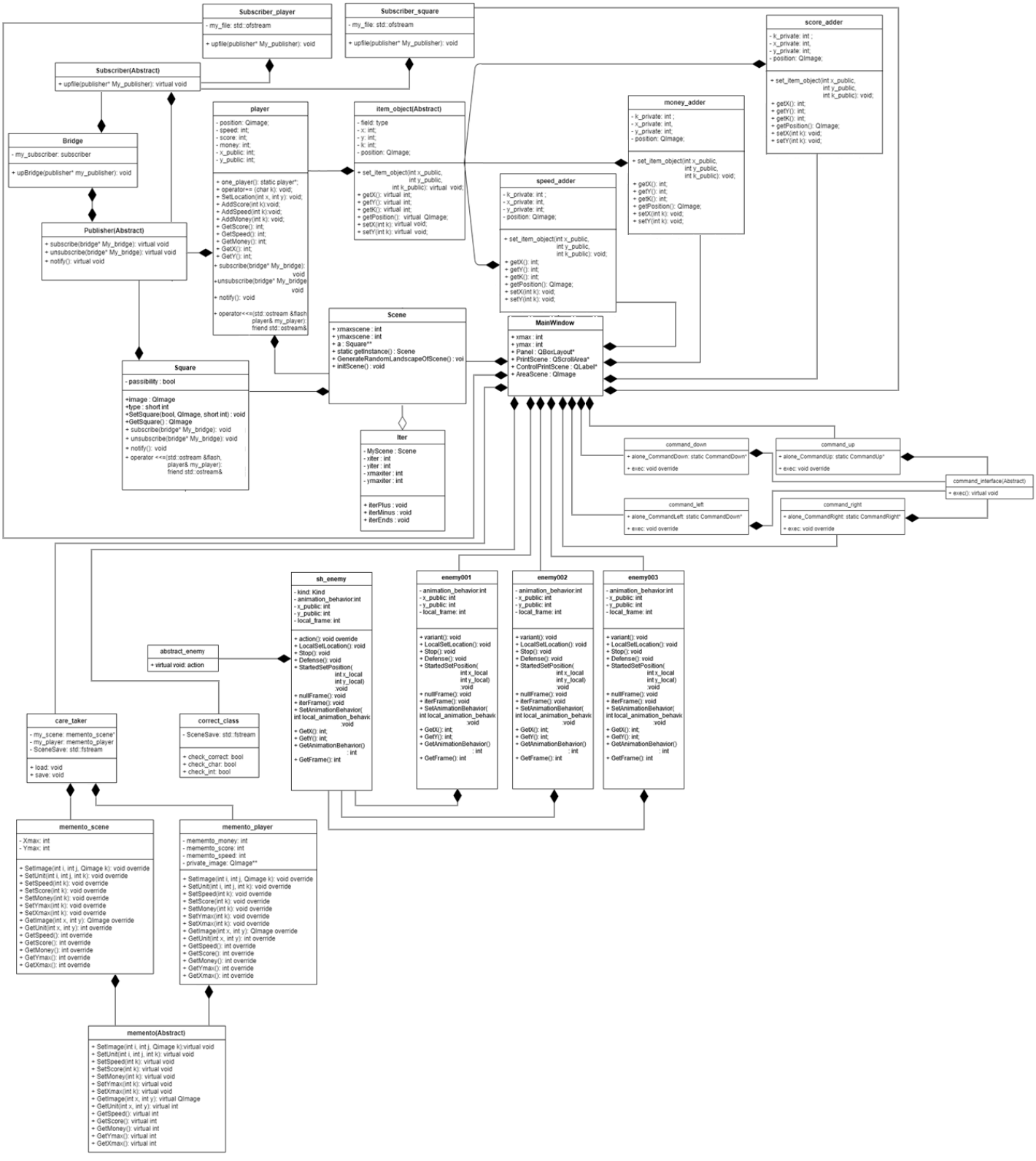

Рисунок 1 – UML-схема

Для выполнения работы создан абстрактный класс memento. Этот класс содержит методы, исполняемые классами memento_player и memento_scene. Эти классы служат для хранения элементов соответственно игрока и сцены. Запись элементов в эти классы происходят с обращением соответственно к классам player и scene, поэтому они туда включаются.

Для самой записи в файл/считывания с файла реализован класс care_taker(). Он включает в себя такие методы, как load() и save(), соответственно загрузка и сохранение. В конструкторе этого класса открывается файл SaveScene, а в деструкторе закрывается, соответственно, взаимодействие идет по идиоме RAII.

В Функции save() идет последовательная запись данных в файл SaveScene.txt. Запись данных идет с обращением на классы memento_player и memento_scene. Сама запись идёт по следующему порядку: изначально записывается размер сцены (ее ширина и высота, соответственно – Xmax, Ymax), после чего записываются данные о игроке (Монеты, скорость и очки, соответственно – Money, Speed и Score), и затем записываются данные всех ячеек сцены в определенном порядке.

При вызове load() идет считывание и изменение сначала данных размера, потом игрока и затем ячеек.

Обе функции вызываются в классе mainwindow(). Перед загрузкой файла идет его проверка на корректность, реализовано это в классе correct_class(), классе исключений.

correct_class(), аналогично классу care_taker() реализован по идиоме RAII: в конструкторе происходит открытие файла, в деструкторе – закрытие. Класс имеет функции check_correct(), check_char() и check_int(), каждая из которых являются логическими переменными. Ключевой функцией этого класса является check_correct(). При выполнении этой функции изначально проверяется, не пуст ли файл, если пуст – возвращается false. Если файл эту проверку прошел, дальше проверяется его содержимое с помощью функций check_int() и check_char(). С помощью check_int() проверяется, является ли получаемая строка целочисленным числом, если нет - возвращается false. С помощью check_char(),

проверяется, валидна ли получаемая строка данным ячейки, если нет, также возвращается false. Соответственно при работе функции check_correct() подаются последовательно строки из файла SceneSave, подобно функции load() в классе care_taker(). И где, например подается ширина поля, идет проверка функцией check_int(), а где подается данные ячейки – проверяется функцией check_char(). После этого, идет проверка на то, закончен ли файл или нет, если файл закончен, по итогу выдается true, иначе – false.

Тестирование.

1) Запуск программы, на сцене произошли какие-то действия, после чего, игрок сохранил игру.

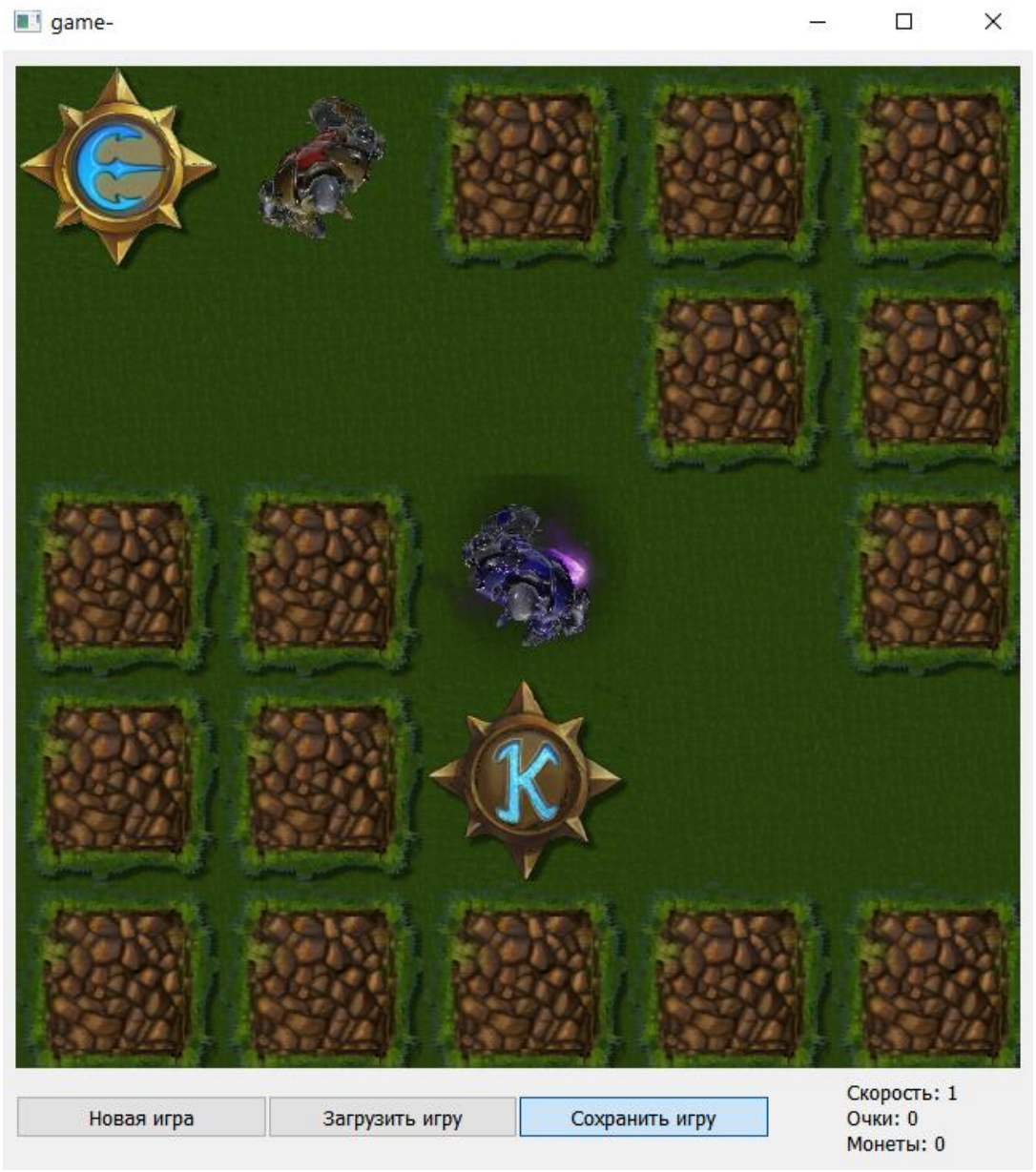

Рисунок 2 – Запуск и сохранение

2) В папке с игрой появился файл с данными сцены.

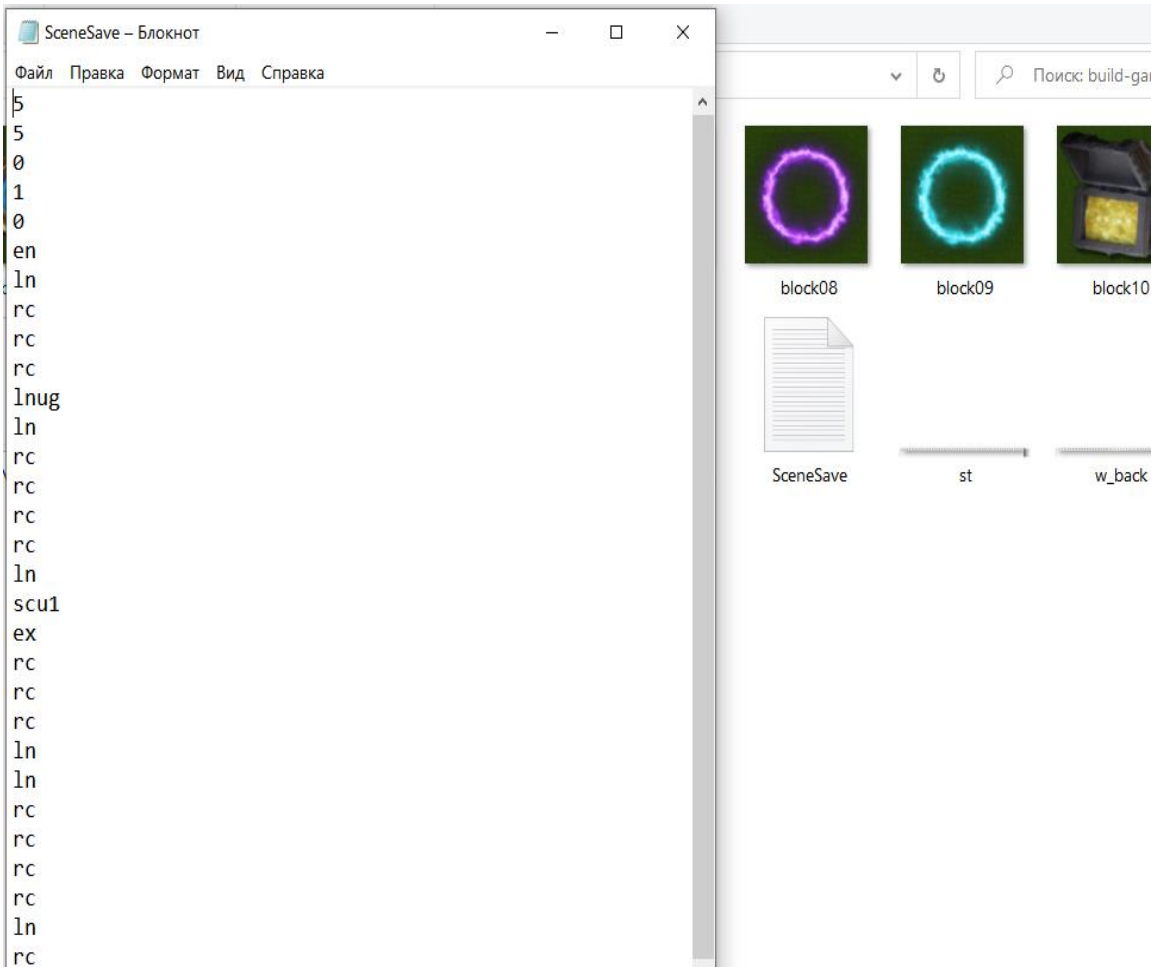

Рисунок 3 – Сохраненный файл сцены

3) Игрок начинает игру заново, создавая сцену:

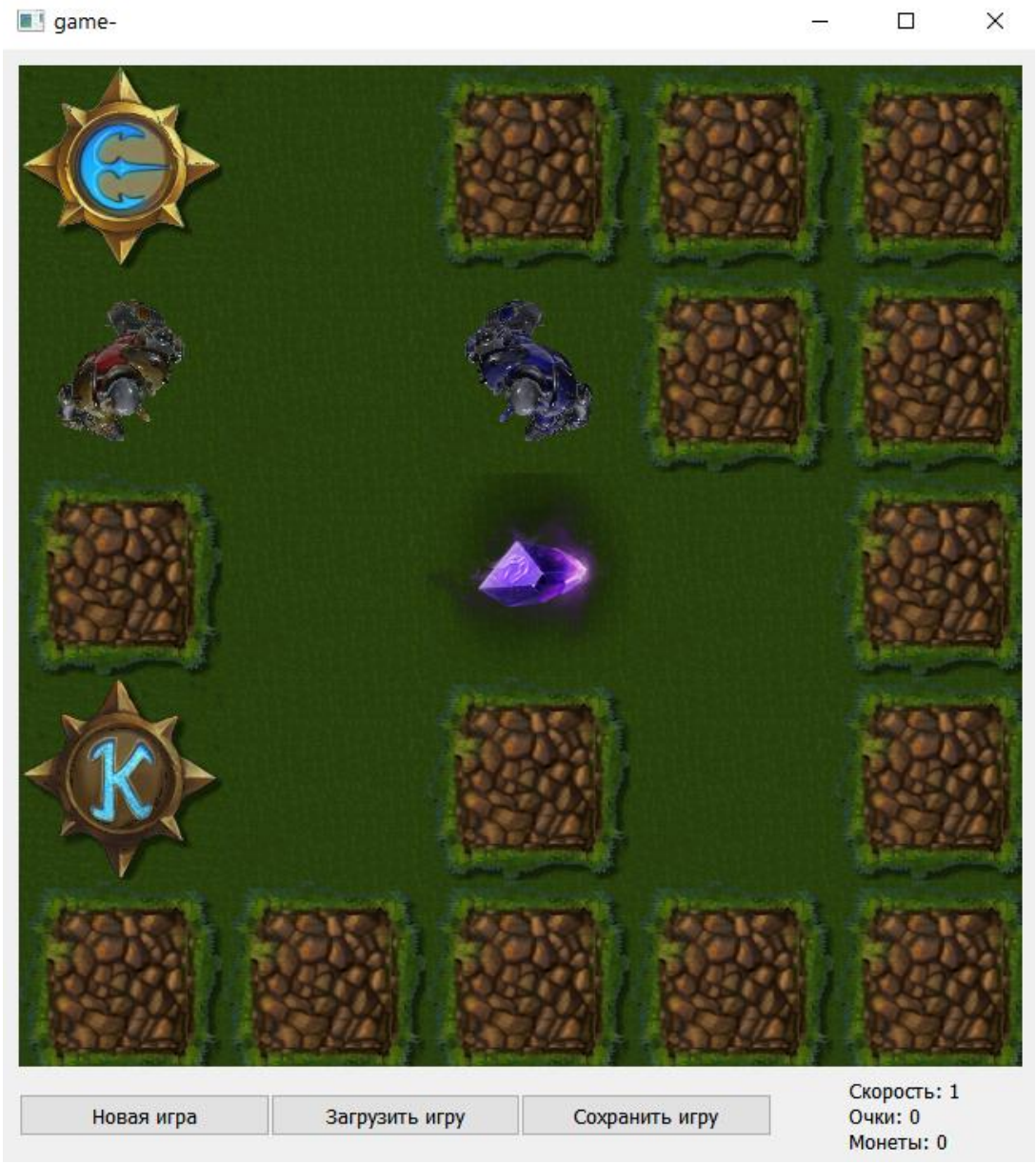

Рисунок 4 – Игрок начал игру заново.

4) Игрок нажал на кнопку “Загрузить игру”, после чего загрузилась сохраненная сцена.

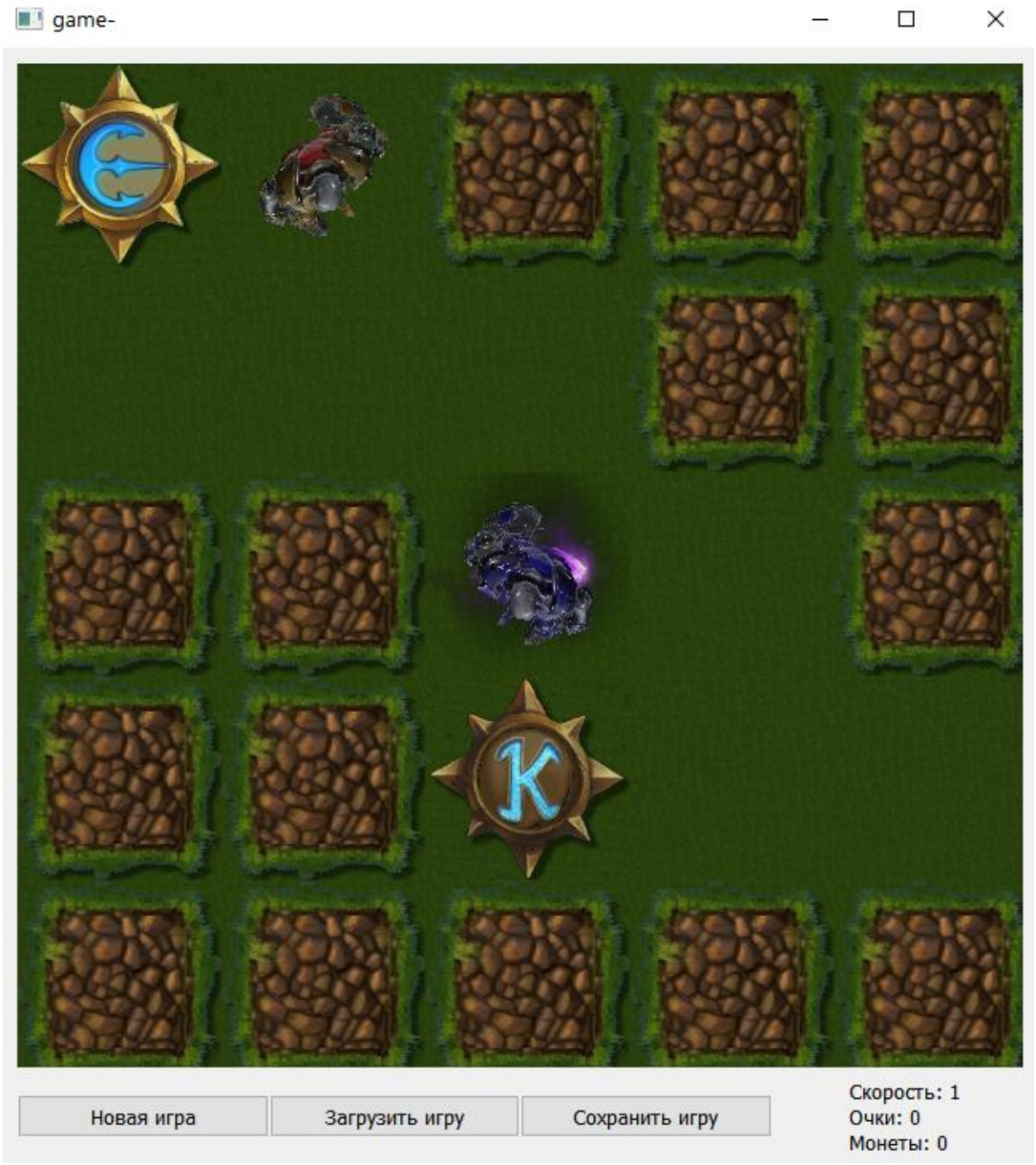

Рисунок 5 – Игрок загрузил сцену

5) Игрок решил написать что-то лишнее в файл сцены:

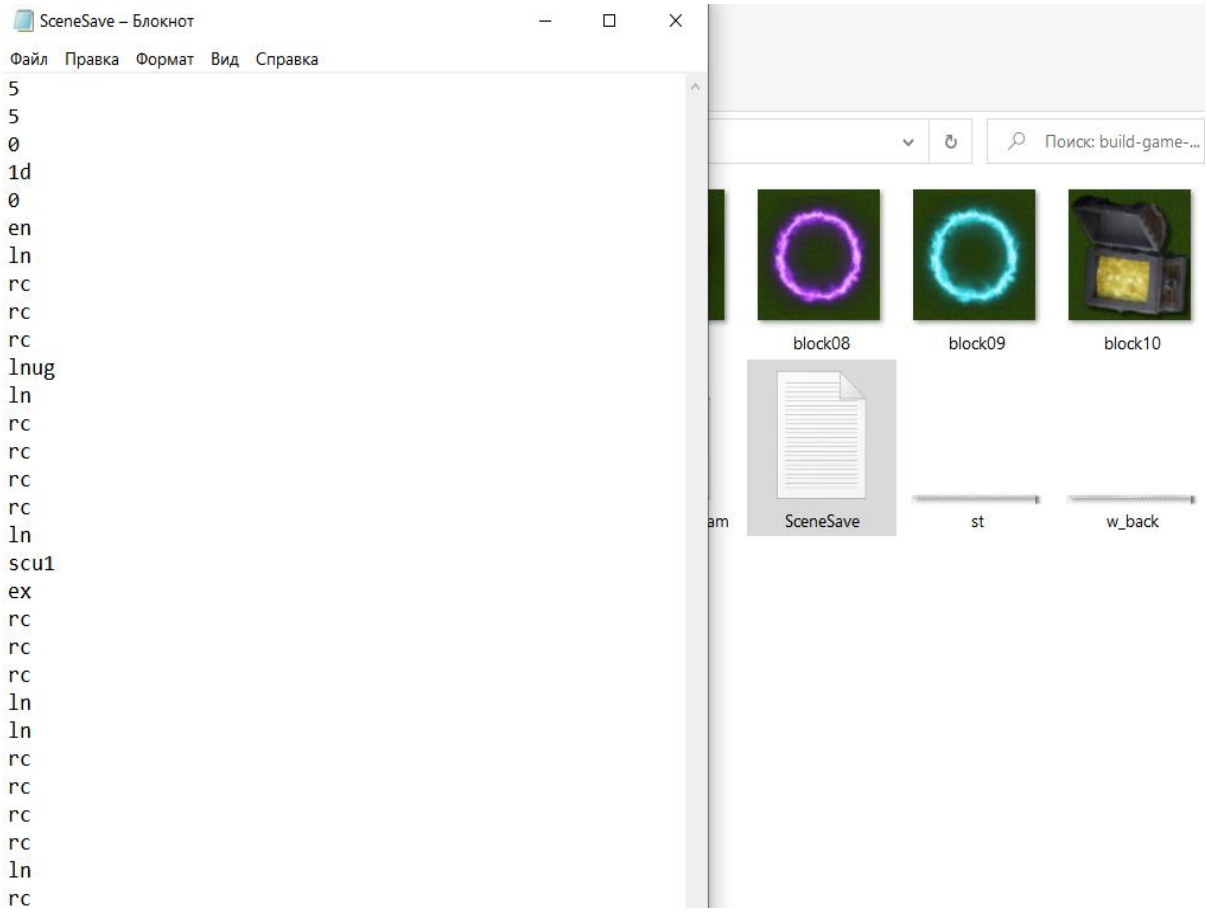

Рисунок 6 – Некорректное изменение файла сцены

6) После чего игрок пробует загрузить изменившийся файл сцены и получает ошибку:

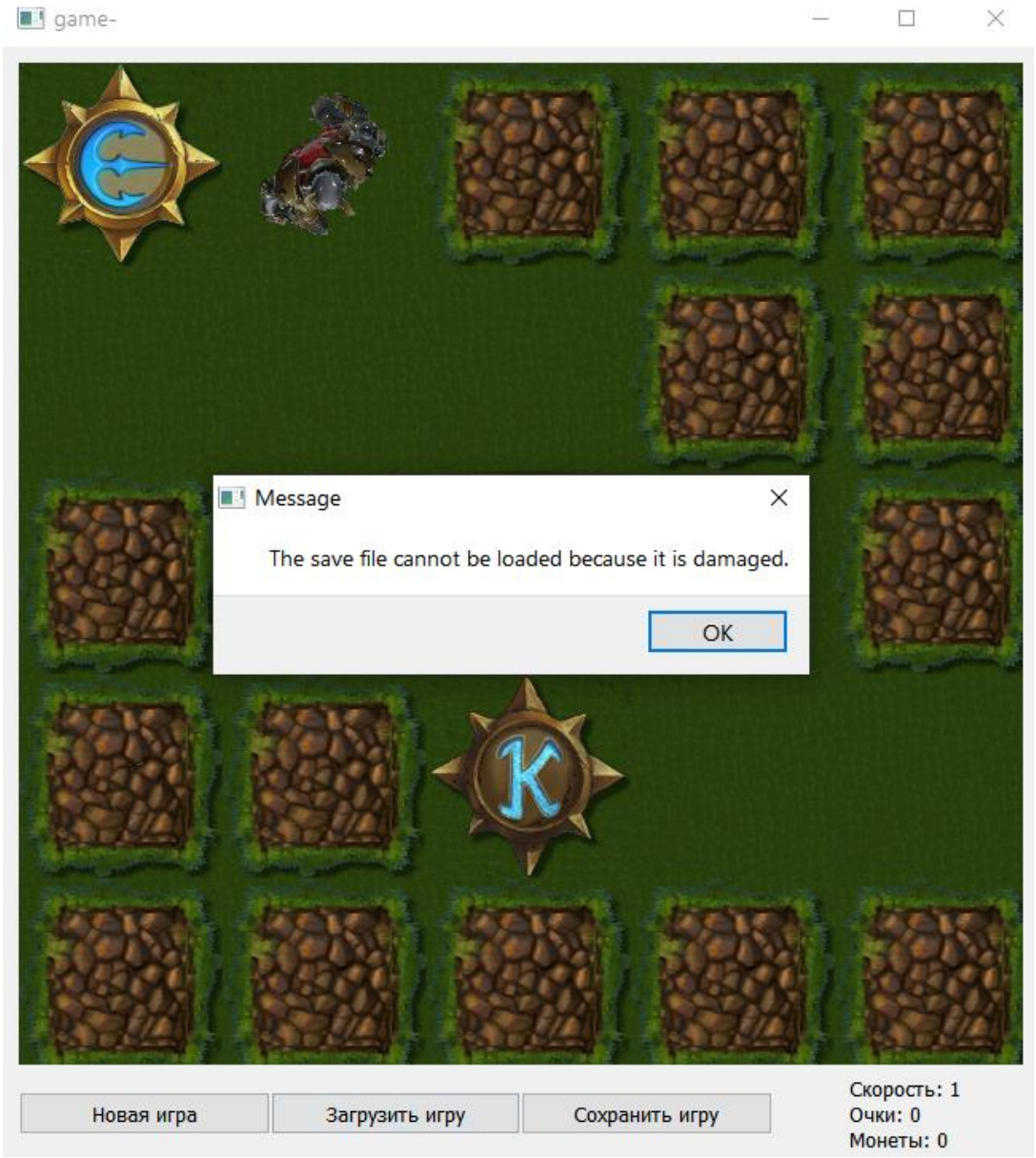

Рисунок 7 – Попытка загрузить некорректный файл сцены

Выводы

Были получены практические знания по использованию классов в программировании. Был создан абстрактный класс memento, классы memento_scene и memento_player, содержащие данные элементов сцены. Соблюдена идиома RAII: классы care_taker() и correct_class() в конструкторе открывают файл, в деструкторе закрывают. Проведена проверка на корректность файла. Написан класс исключений correct_class().