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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «ООП»

ТЕМА: ДОБАВЛЕНИЕ КЛАССА УПРАВЛЕНИЯ ИГРОЙ

Студент гр. 9304 Прокофьев М.Д.

Преподаватель Размочаева Н.В.

Санкт-Петербург
2020

Цель работы.

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

Задание.

Создать класс игры, через который пользователь взаимодействует с игрой. Управление игроком, начало новой игры, завершение игры. Могут быть созданы дополнительные необходимые классы, которые отвечают отдельно за перемещение, создание игры и.т.д. Но пользователь должен взаимодействовать через интерфейс одного класса.

Обязательные требования:

-Создан класс управления игрой

-Взаимодействие сохраняет инвариант

Дополнительные требования:

-Пользователь взаимодействует с использованием паттерна Команды

-Взаимодействие с компонентами происходит через паттерн Фасад

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.

Паттерн «Фасад» предоставляет унифицированный интерфейс вместо набора интерфейсов некоторой подсистемы. Фасад определяет интерфейс более высокого уровня, который упрощает использование подсистемы.

Выполнение работы.

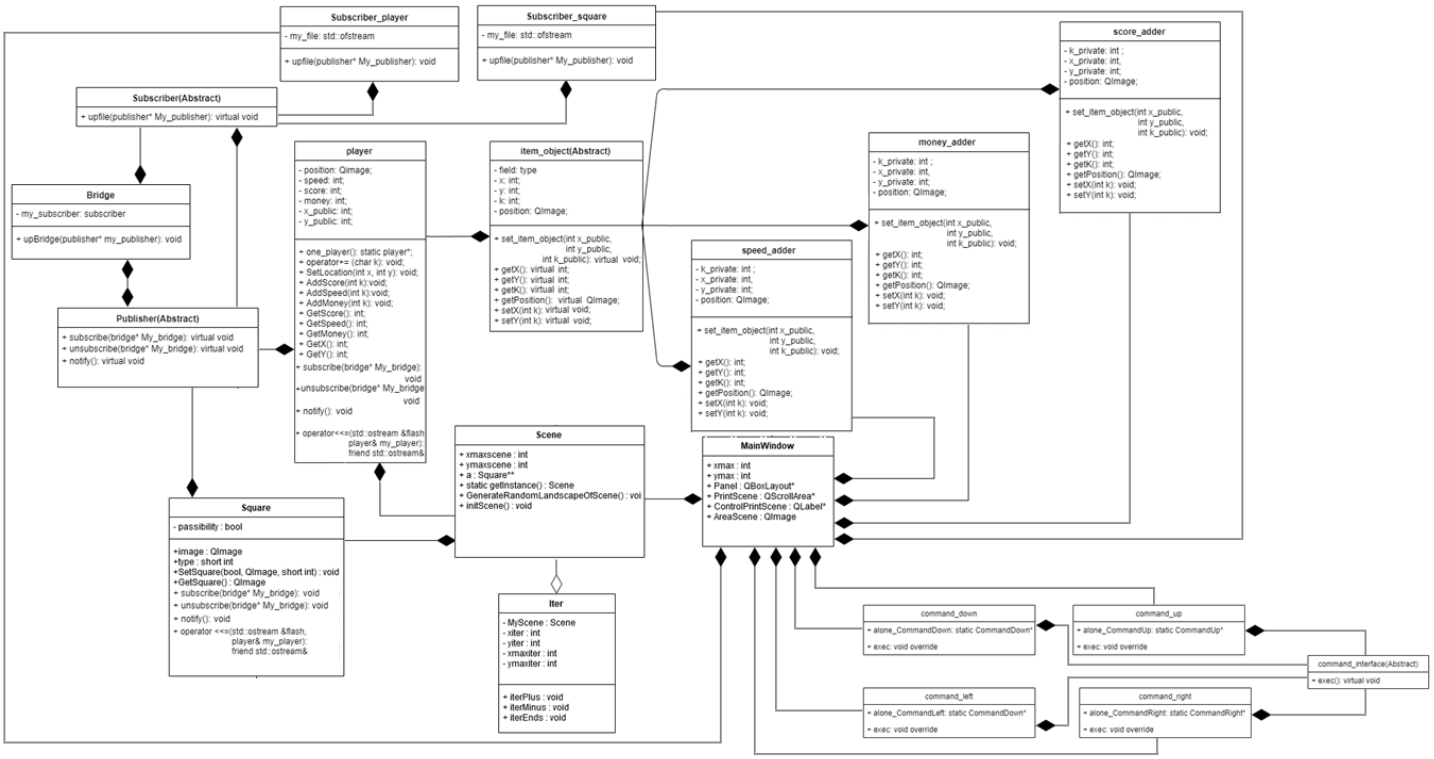

Рисунок 1 – UML-схема

В программе классом управления игрой является *mainwindow*. В нем происходят все необходимые настройки игры, такие как размер экрана, прорисовка элементов сцены, методы-события, а также этот класс отвечает за начало и конец игры.

Также создан класс *command_interface*, это абстрактный класс для команд, которые вызывает игрок нажатием на клавиатуру.

Для этого класса было создано 4 класса наследника: *commandup*, *commandright*, *commandleft* и *commanddown*, которые обозначают сами команды, которыми игрок взаимодействует. Каждый из этих классов является синглтоном. Такое решение справедливо, поскольку каждая из этих команд может быть в игре только в единственном числе. Таким образом был реализован паттерн Команды для программы. Естественно, эти классы присоединены к классу управления игрой, классу *mainwindow*.

Также был создан класс *modifiers* для удобной работы с модификаторами, которые нужны для выполнения функции передвижения игрока.

Тестирование.

1) Запуск программы:

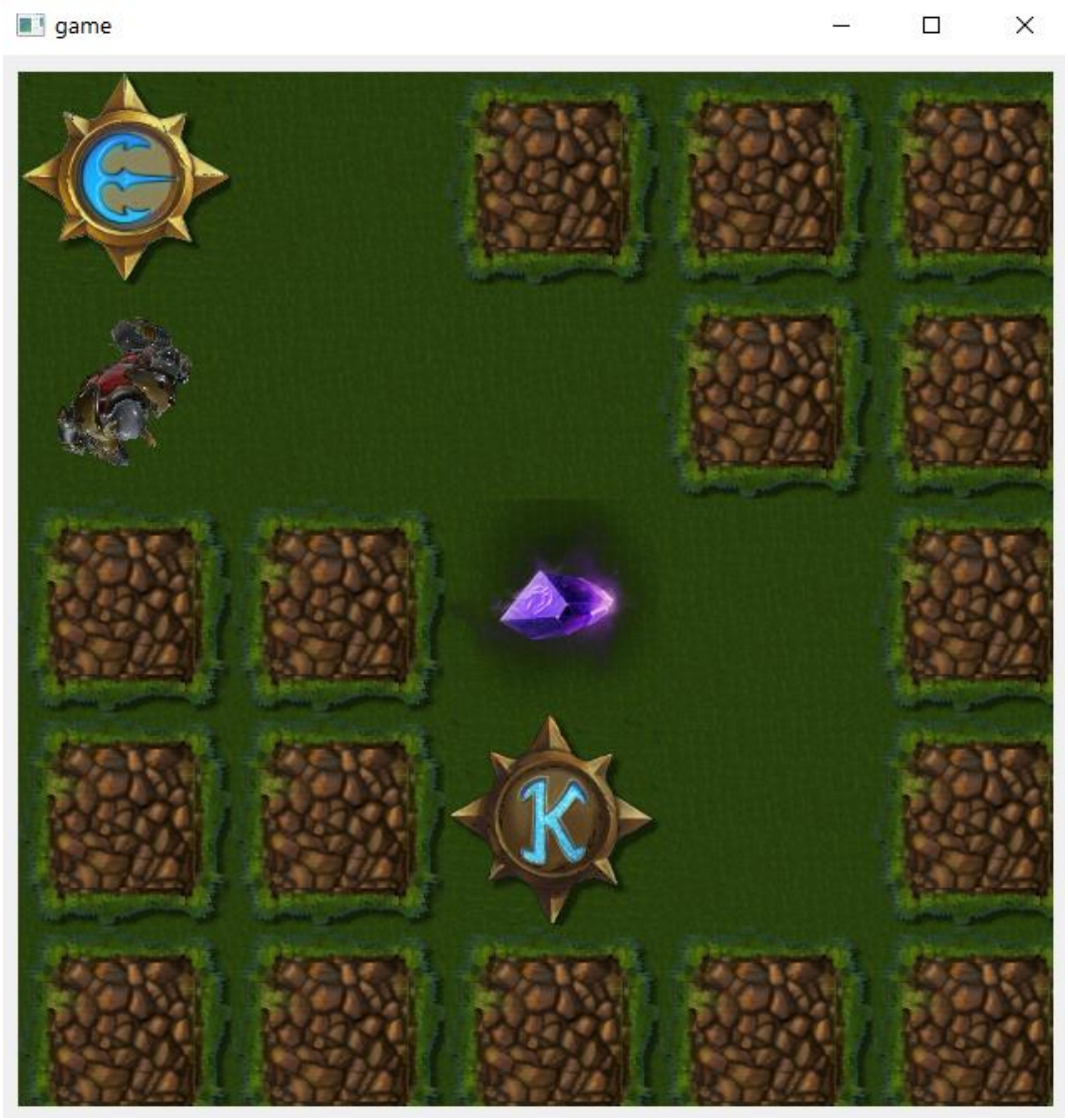

Рисунок 2 – Запуск

2) Игрок нажал на клавишу вправо:

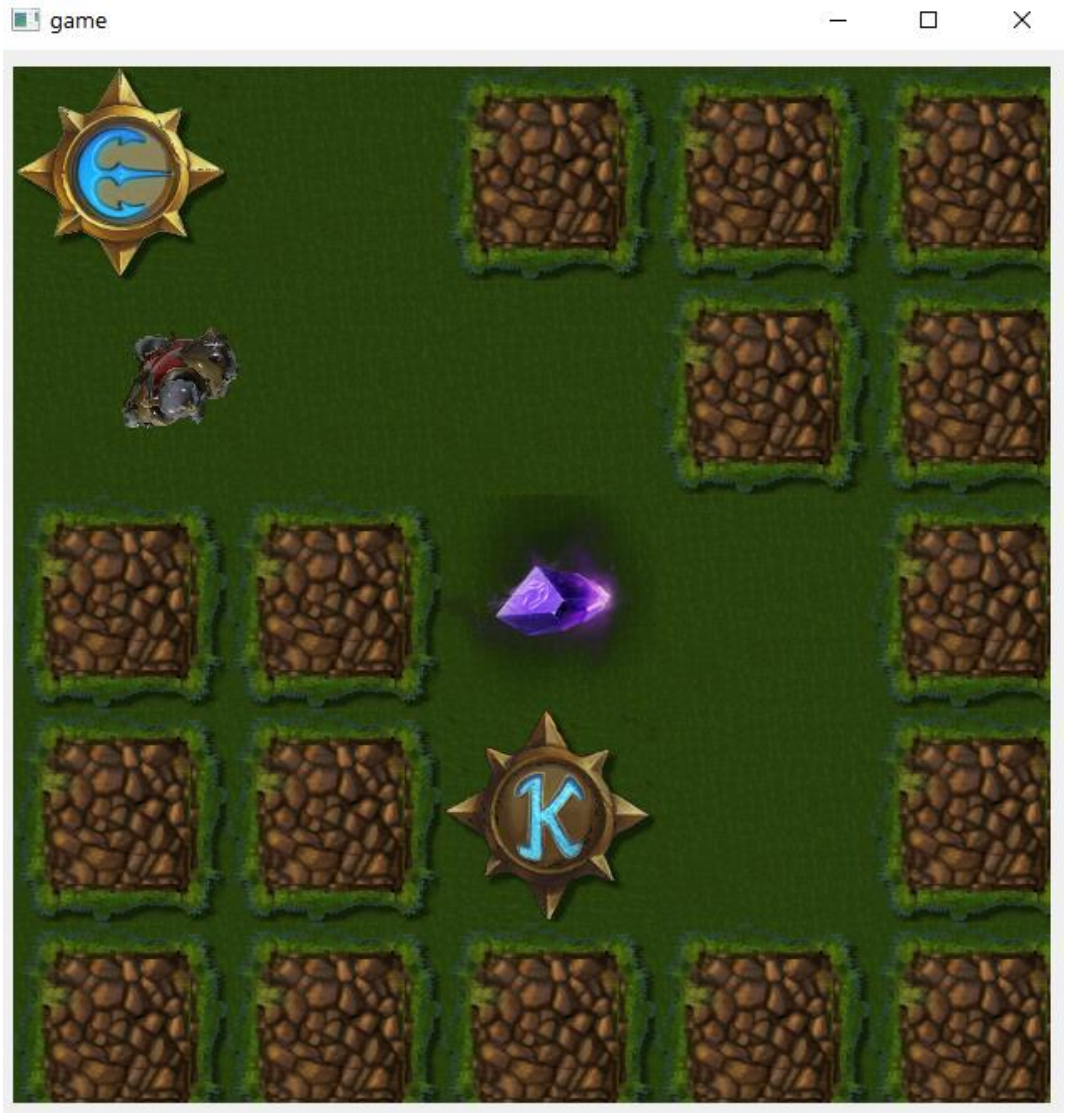

Рисунок 3 – Игрок сделал передвижения

3) Игрок дошел до финиша, конец игры:

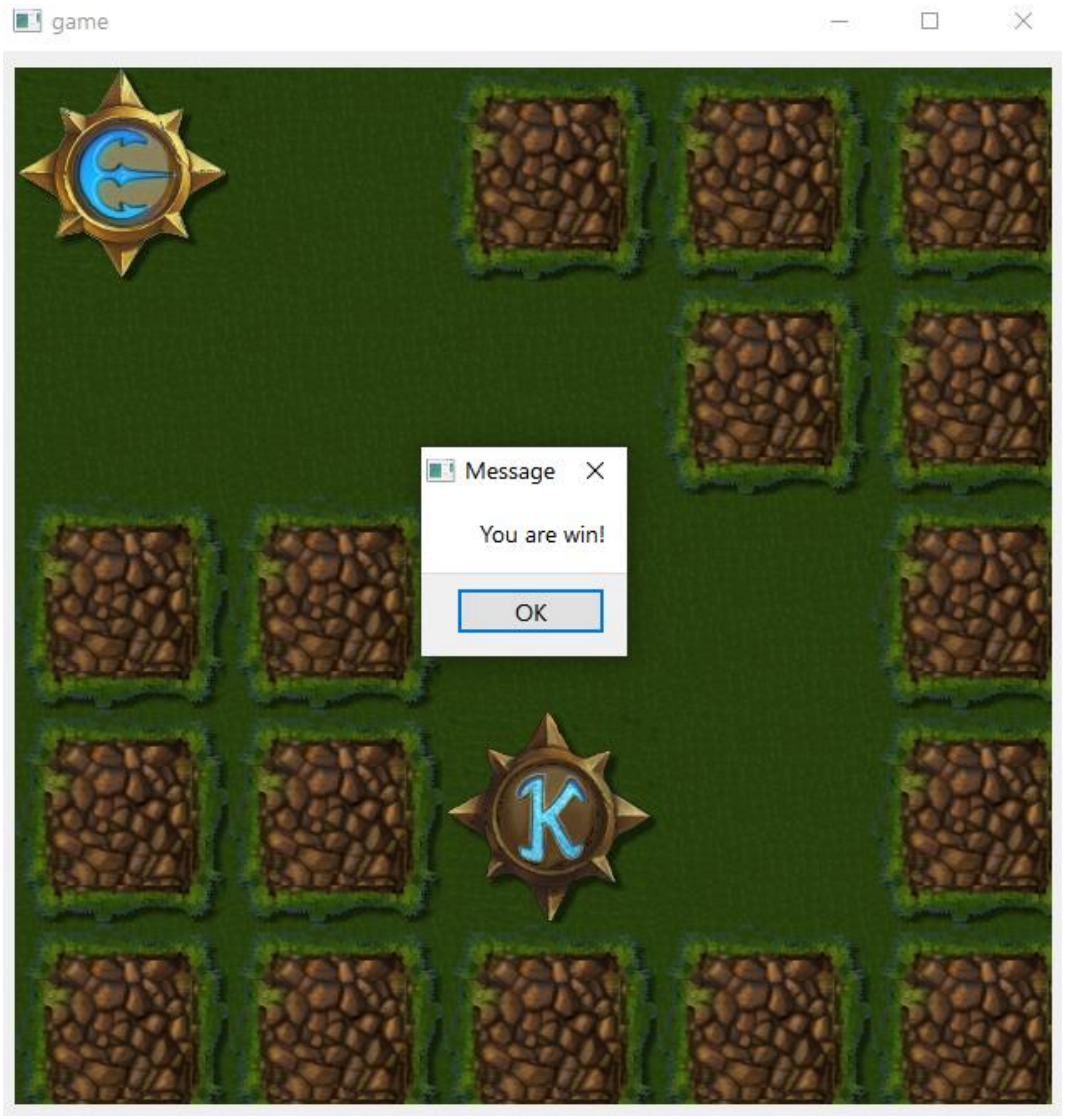

Рисунок 4 – Игрок дошел до финиша

Выводы

Были получены практические знания по использованию классов в программировании. Был создан класс mainwindow, как класс управления игрой. Кроме того был создан абстрактный класс command_interface, а также его наследники: *commandup*, *commandright*, *commandleft* и *commanddown*. Таким способом был реализован паттерн Команды.