**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Программирование»**

Тема: Полиморфизм

|  |  |  |
| --- | --- | --- |
| Студент гр. 2381 |  | Рыжиков И.С. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2023

## Цель работы

Целью работы является изучение полиморфизма и взаимосвязей между классами, которые реализуют один и тот же интерфейс в языке C++.

## Задание

а) Создать интерфейс игрового события. Интерфейс должен обеспечивать срабатывание события когда игрок наступает на клетку.

б) Реализовать интерфейс игрового события тремя конкретными событиями. Одно событие должно положительно влиять на характеристики игрока, второе должно негативно влиять на характеристики игрока, третье изменять координаты игрока на поле. При желании можно реализовать больше событий и/или события меняющие само поле (например, делать из непроходимой клетки проходимую).

в) В классе управления игроком добавить проверку на наличие события на клетке, если событие присутствует, то оно должно сработать. Срабатывание должно происходить через интерфейс события, и не должно быть никаких проверок на тип события (реализация через динамический полиморфизм)

г) Создать класс создающий поле. Предусмотреть возможность создания 2 разных уровней. По желанию можно сделать случайную генерацию уровней. Должно гарантироваться, что игрок может дойти от входа до выхода.

Примечания:

* События должны быть такими, чтобы был сценарий проигрыша игрока.
* В событиях и клетках не должно быть полей сообщающих информацию о типе события

## Выполнение работы

### IEvent

Добавлен интерфейс игрового события IEvent, в котором определены следующие виртуальные функции:

* handle(EventTarget &) — вызвать (обработать) событие. Принимает как аргумент структуру EventTarget, в которой хранятся ссылки на контроллер игрока PlayerController, игрока Player, поле Field и координаты клетки с которой был вызван ивент.
* clone() — метод, который возвращает указатель на IEvent. Данный метод используется для создания копии события, ввиду того что конструкция new IEvent(\*other\_ptr\_to\_event).

### HealEvent

Имплементирует интерфейс IEvent.

При вызове события восстанавливает здоровье игрока. Может использоваться ограниченное количество раз, после чего удаляется с поля.

### TeleportEvent

Имплементирует интерфейс IEvent.

При вызове события handle() телепортирует игрока на указанную координату.

### TrapEvent

Имплементирует интерфейс IEvent.

При вызове события наносит урон игроку. После чего событие удаляется.

### EventFactory

Создаёт игровые события. Методы возвращают указатели на IEvent.

* Метод createHealEvent создает экземпляр HealEvent;
* Метод createTeleportEvent создает TeleportEvent;
* Метод createTrapEvent создает TrapEvent;

### FieldCell

В клетку добавлена возможность хранить указатель на IEvent. И методы setEvent и getEvent.

### FieldCreator

Добавлен класс FieldCreator, который создаёт поле для определенного уровня. Заполняет его событиями и устанавливает проходимость клеток.

Имеет метод createFieldForLevel(int level), которому можно указать какой уровень нужно создать. Если передать номер уровня, генерация которого ещё не создана будет возвращен уровень по умолчанию.

### FieldView

Добавлен класс FieldView, который при вызове метода renderField создаёт строковое представления поля и игрока на нём. На текущий момент показывается проходимость клетки.

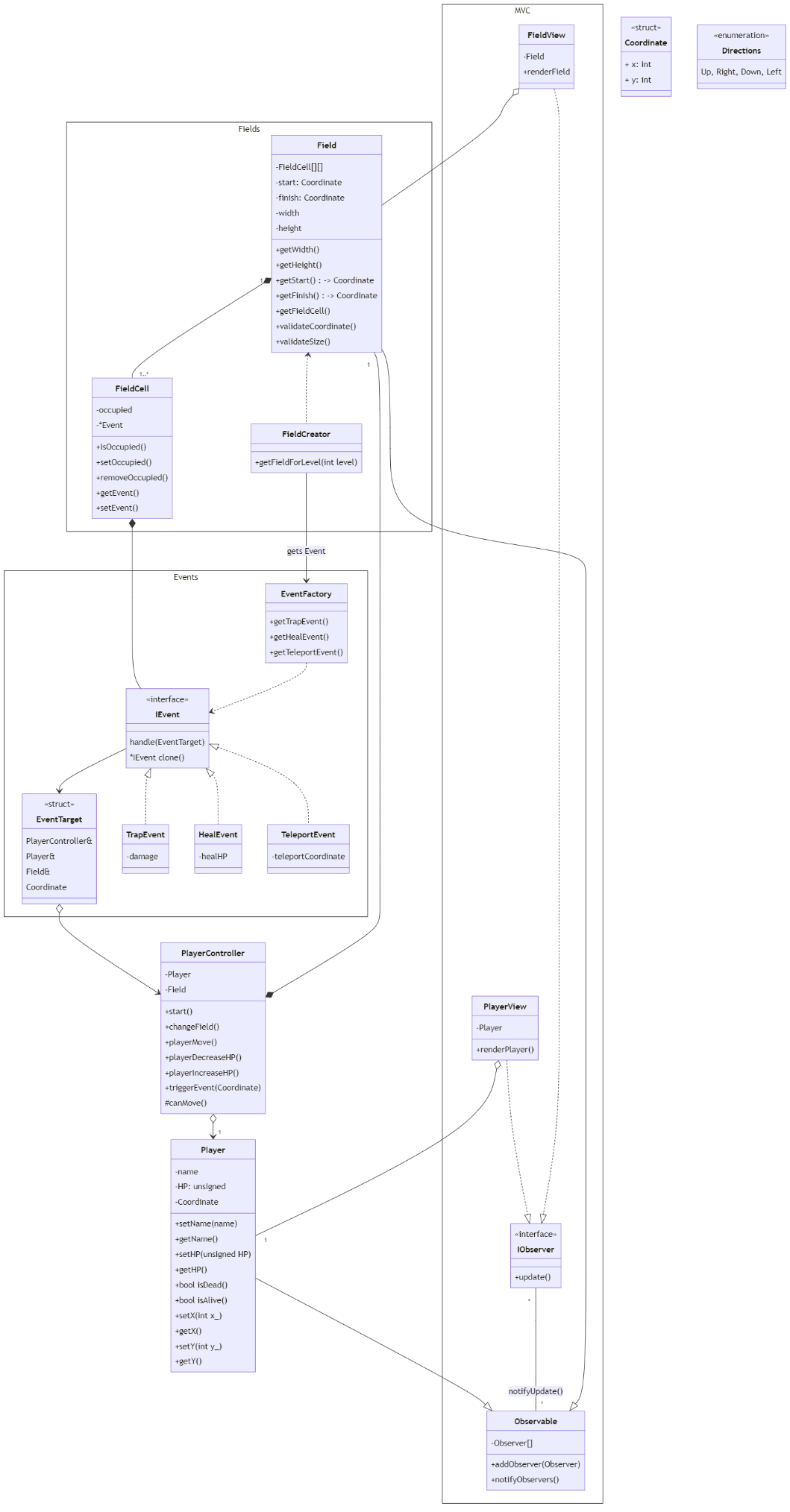
В будущем будет показаны различные события на клетке.

## Выводы

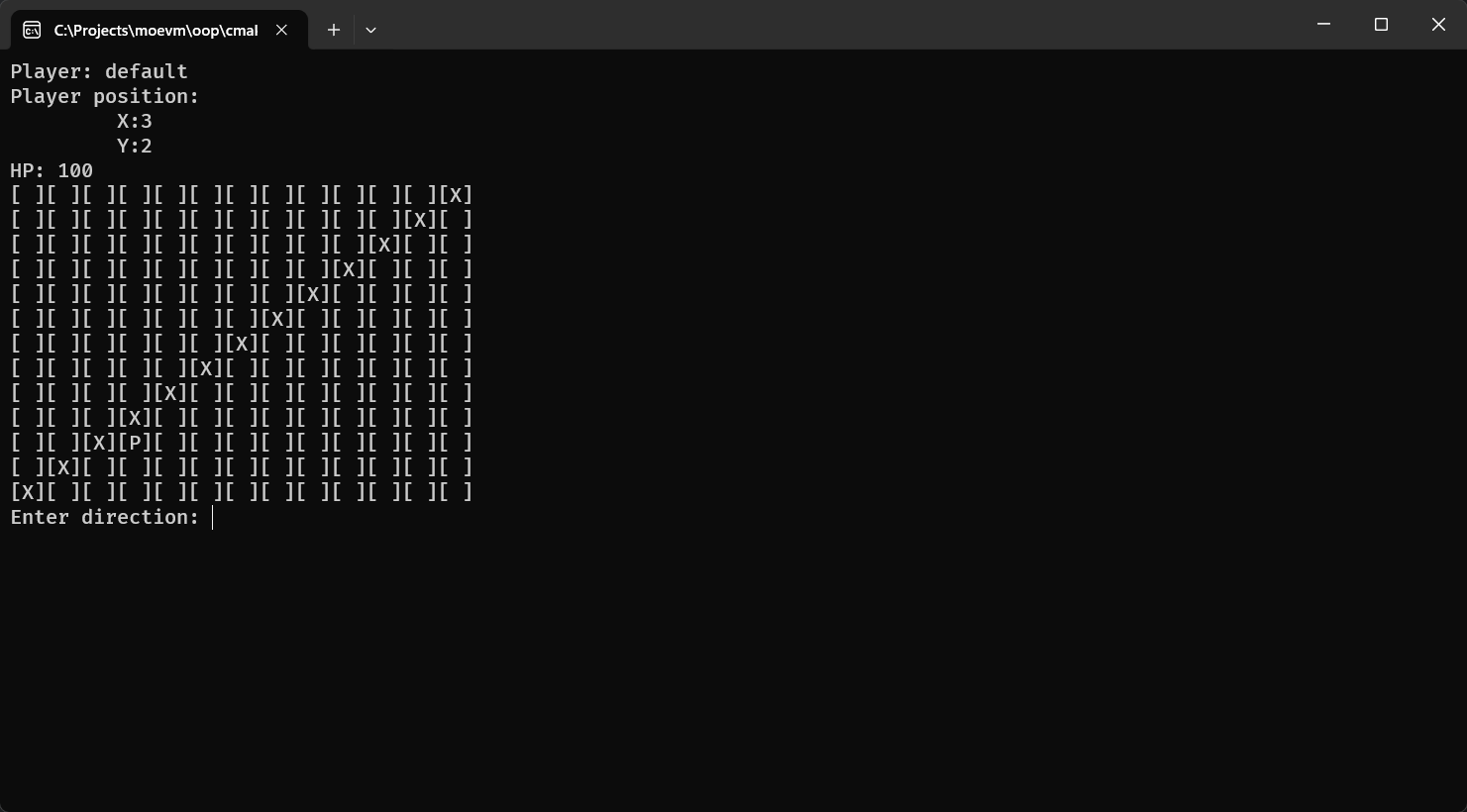
Создан интерфейс игрового события и отдельные классы, которые реализуют данный интерфейс. Тем самым обеспечивается динамический полиморфизм.

Пример работы программы (рис. 1) и диаграмма классов (рис. 2) представлена в Приложении.

# Приложение



1. Диаграмма классов



1. Пример работы программы