**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №2**

**по дисциплине «Объектно-ориентированное программирование»**

**Тема: Интерфейсы, динамический полиморфизм**

| Студент гр. 1303 |  | Чубан Д.В. |
| --- | --- | --- |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2022

## **Цель работы.**

* 1. Реализовать систему событий. Событие - сущность, которая срабатывает при взаимодействии с игроком. Должен быть разработан класс интерфейс общий для всех событий, поддерживающий взаимодействие с игроком. Необходимо создать несколько групп разных событий реализуя унаследованные от интерфейса события (например, враг, который проверяет условие, будет ли воздействовать на игрока или нет; ловушка, которая безусловно воздействует на игрока; событие , которое меняет карту; и.т.д.). Для каждой группы реализовать конкретные события, которые по разному воздействуют на игрока (например, какое-то событие заставляет передвинуться игрока в определенную сторону, а другое меняет характеристики игрока). Также, необходимо предусмотреть событие “Победа/Выход”, которое срабатывает при соблюдении определенного набора условий.
  2. Реализовать ситуацию проигрыша (например, потери всего здоровья игрока) и выигрыша игрока (добрался и активировал событие “Победа/Выход”)

**Требования.**

* Разработан интерфейс события с необходимым описанием методов
* Реализовано минимум 2 группы событий (2 абстрактных класса наследников события)
* Для каждой группы реализовано минимум 2 конкретных события (наследники от группы события)
* Реализовано минимум одно условное и безусловное событие (условное - проверяет выполнение условий, безусловное - не проверяет).
* Реализовано минимум одно событие, которое меняет карту (меняет события на клетках или открывает расположение выхода или делает какие-то клетки проходимыми (на них необходимо добавить события) или не непроходимыми
* Игрок в гарантированно имеет возможность дойти до выхода

## **Описание архитектурных решений и классов.**

Был реализован изначальный абстрактный класс *Event*, который имеет только набор виртуальных методов для последующих наследуемых классов (*virtual event clone(), virtual void eventProc(Field& field)*). Далее, от *Event* наследуются абстрактные классы *Map\_Events и Player\_Events,* которые отвечают за события, влияющие на карту и на игрока соответственно.

От класса *Player\_Events* наследуются уже не абстрактные классы *Trap, Overseer, Win* и *Enemy.*

Когда игрок переходит в клетку с событием класса *Trap*, вызывается переопределенный метод *eventProc()*, в зависимости от характеристик игрока и уровня события, он получает урон. Характеристика события - поле *int lvl.*

Класс *Enemy* олицетворяет противника на клетке. Он имеет характеристики уровня, здоровья и атаки. Когда игрок переходит на клетку с противником, также вызывается метод *eventProc()*, в котором происходит “бой” игрока с противником без прямого участия игрока. В случае победы, характеристики игрока увеличатся в зависимости от уровня противника (поле *lvl*), а также добавляются очки счета.

Если здоровье игрока опускается до нуля, поле игрока *winState* становится *LOSE(*переопределение значения -1) и игра считается проигранной, программа завершается.

При переходе на клетку с событием класса *Overseer* на карте при выводе на экран клетки с событияминачинают отображаться не как [?](неизвестное событие), а будут иметь свое собственное отображение.

При переходе на клетку с событием класса *Win* игроку добавляется заданное при создании события количество очков.

От класса *Map\_Events* наследуются неабстрактные классы *Sledge* и *Earthquake.*

При переходе на клетку с событием класса *Sledge* все непроходимые клетки на поле меняют свой статус на проходимые.

При переходе на клетку с событием класса *Earthquake* клетки слева, справа и снизу игрока становятся непроходимыми, изменяя карту.

**Вывод.**

Были созданы классы различных событий, влияющие либо на карту, либо на игрока с проверками характеристик и без них. UML-диаграмма была дополнена новыми объектами.