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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Объектно ориентированное программирование»

Тема: **СОЗДАНИЕ КЛАССОВ, КОНСТРУКТОРОВ И МЕТОДОВ**

Студент гр. 1304 _____ Шаврин А.П.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы.

Изучить основы ООП в языке C++ и применить полученные знания в реализации игры. Научиться создавать классы, их конструкторы и методы, а также выстраивать архитектуру проекта.

Задание.

Реализовать прямоугольное игровое поле, состоящее из клеток. Клетка - элемент поля, которая может быть проходима или нет (определяет, куда может стать игрок), а также содержит какое-либо событие, которое срабатывает, когда игрок становится на клетку. Для игрового поля при создании должна быть возможность установить размер (количество клеток по вертикали и горизонтали). Игровое поле должно быть зациклено по вертикали и горизонтали, то есть если игрок находится на правой границе и идет вправо, то он оказывается на левой границе (аналогично для всех краев поля).

Реализовать класс игрока. Игрок - сущность контролируемая пользователем. Игрок должен иметь свой набор характеристик и различный набор действий (например, разные способы перемещения, попытка избежать событие, и так далее).

Требования:

- Реализован класс игрового поля
- Для игрового поля реализован конструктор с возможностью задать размер и конструктор по умолчанию (то есть конструктор, который можно вызвать без аргументов)
- Реализован класс интерфейс события (в данной лабораторной это может быть пустой абстрактный класс)
- Реализован класс клетки с конструктором, позволяющим задать ей начальные параметры.
- Для клетки реализованы методы реагирования на то, что игрок перешел на клетку.
- Для клетки реализованы методы, позволяющие заменять событие. (То есть клетка в ходе игры может динамически меняться)
- Реализованы конструкторы копирования и перемещения, и соответствующие им операторы присваивания для игрового поля и при необходимости клетки
- Реализован класс игрока минимум с 3 характеристиками. И соответствующие ему конструкторы.
- Реализовано перемещение игрока по полю с проверкой допустимости на переход по клеткам.

Примечания:

- При написании конструкторов учитывайте, что события должны храниться по указателю для соблюдения полиморфизма
- Для управления игроком можно использовать медиатор, команду, цепочку обязанностей

Выполнение работы.

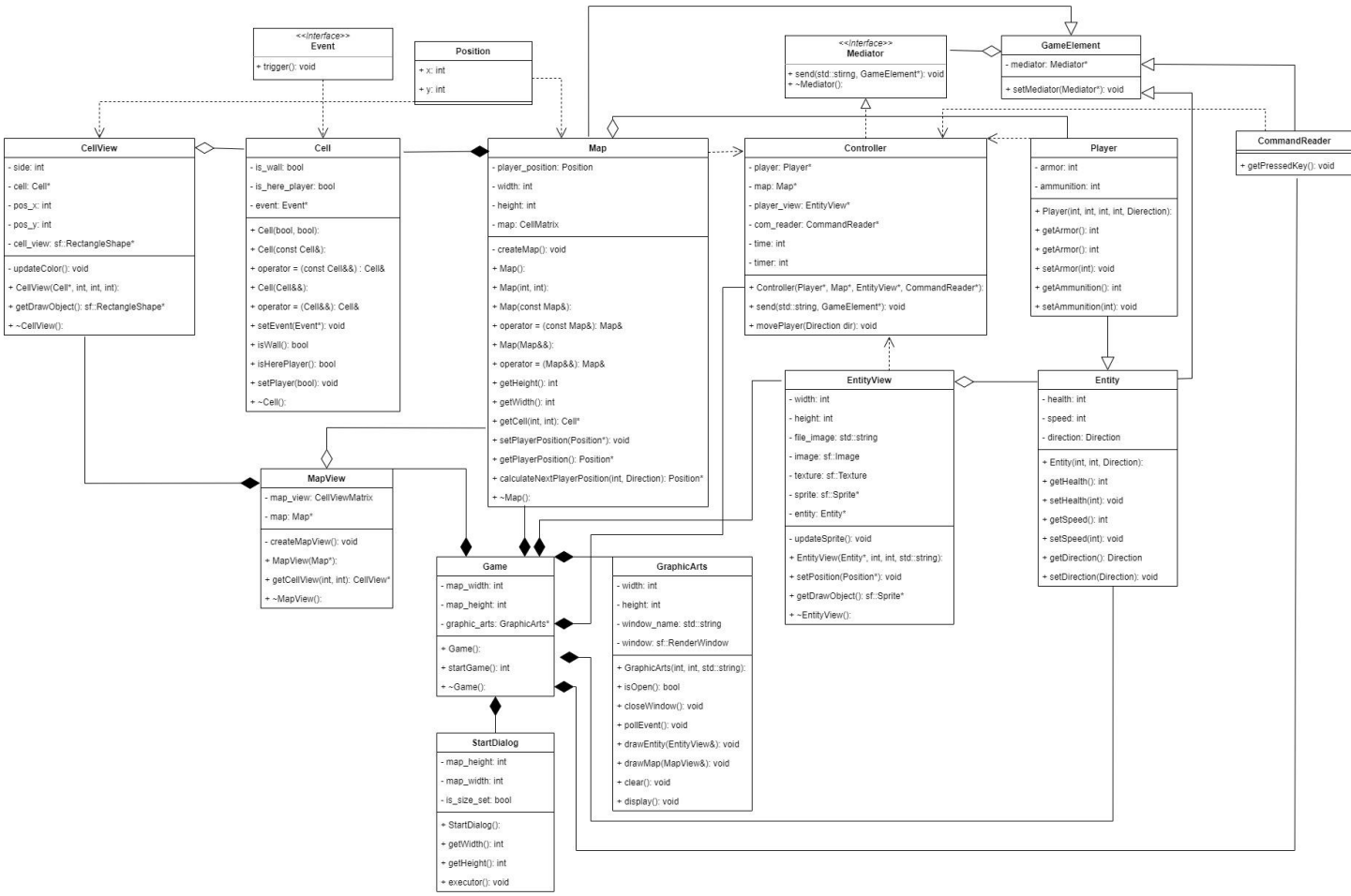
1. Был создан класс Position, имеющий 2 координаты x и y типа int.
2. Был создан класс StartDialog для первого диалога с пользователем. В методе executor выводится информация о возможности задачи таких параметров, как ширина и высота поля (в клетках). Если пользователь соглашается задать стартовые данные, то они записываются в соответствующие им поля класса, в случае отказа устанавливаются параметры по умолчанию.
3. Создан абстрактный класс Mediator, имеющий виртуальную функцию send, отвечающую за передачу сообщения.
4. Создан класс GameElement, хранящий в себе указатель на медиатор и метод позволяющий установить медиатор.
5. Создан класс CommandReader, унаследованный от GameElement, отвечающий за считывание клавиш и передающий через медиатор сообщение, содержащее задачу, которую должен выполнить контроллер.
6. Был создан абстрактный класс Event, отвечающий за события, что будут в дальнейшем происходить в ходе взаимодействия игрока, карты и прочих сущностей.
7. Затем создан класс Cell хранящий в себе поля, отвечающие за возможность проходимости и наличие на ней игрока, а также указатель на событие.
8. Затем создан класс CellView хранящий в себе поля, отвечающие за размер клетки, визуальное отображение и координаты левого верхнего угла в пикселях для отрисовки, а также указатель на логическую клетку, за чье отображение он отвечает. В методе updateColor происходит обновление цвета клетки, если игрок находится на этой клетке или клетка проходима или нет.

9. После создан класс Map, унаследованный от GameElement, для возможности работы с контроллером, хранящий в себе свои размеры, координаты игрока и двумерный массив клеток. В методе calculateNextPlayerPosition происходит получение новых координат игрока, а в методе setPlayerPosition происходит изменение позиции игрока с учетом зацикливания поля и возможности пройти на клетку. Также там меняются характеристики клеток, отвечающие за нахождение на них игрока.
10. После создан класс MapView, хранящий в себе размеры поля, двумерный массив визуальных клеток и указатель на логическую карту. Метод getCellView возвращает визуальное отображение клетки с заданными координатами.
11. Создан класс Entity, унаследованный от класса GameElement, имеющий поля характерные для всех существ (здоровье, скорость, направление)
12. Затем создан класс EntityView, имеющий поля отвечающие за визуальное отображение существа (высота и ширина, изображение, текстура изображения, спрайт существа, а также указатель на существо, для доступа к направлению). Метод getDrawObject возвращает спрайт существа, для графического класса.
13. Был создан класс Player, унаследованный от класса Entity, хранящий в себе поля броня и боеприпасы (и все поля класса Entity)
14. Создан класс Controller, унаследованный от класса Mediator, осуществляющий выполнение действий запрашиваемых через медиатор от разных классов (унаследованных от GameElement). Он имеет в своих полях указатели на классы Player, Map, CommandReader and EntityView. В переопределенном методе send выполняется проверка от кого через медиатор пришло сообщение и в зависимости от содержимого сообщения выполняется то или иное действие.
15. Был создан класс GraphicArts, класс отвечающий за графику. В себе он хранит размеры отрисовываемого окна и обьект класса sfml, который

производит всю отрисовку. Для отрисовки карты и существа были реализованы соответствующие методы.

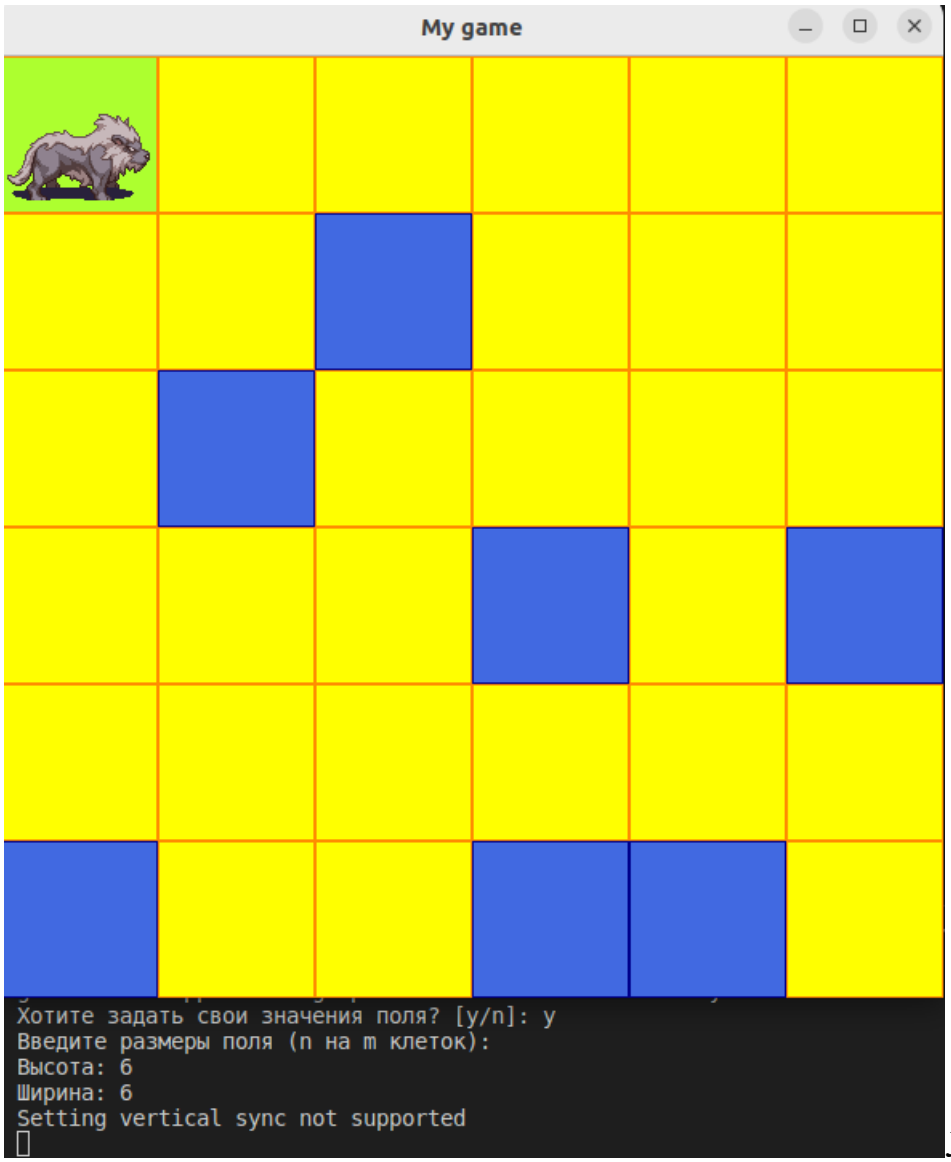
16. В конце был создан класс Game, хранящий в себе размеры поля, для передачи их при создании карты и графичкеского класса, а также указатель на GraphicArts. В конструкторе происходит создание обьекта класса StartDialog и от него передаются значеничя ширины и высоты поля в собственные поля. В методе startGame происходит создание всех необходимых обьектов классов и осуществляется основная логика игры.

Ниже приведена UML диаграмма зависимости классов со всеми их полями и методами:



Тестирование.

Вывод результата работы диалогового окна и пример успешного создания поля.



Левый верхний угол имеет координаты (0, 0).

Синий блок не проходим игроком и является стенкой.

Выводы.

Были изучены основы ООП в языке C++ и применены в реализации игры. Были созданы классы, их конструкторы и методы, а также выстроена архитектура проекта.