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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «ООП»

Тема: Управление, разделение на уровни абстракций.

Студент гр. 0383 _____ Козлов Т.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Необходимо организовать управление игрой (номинально через CLI). При управлении игрой с клавиатуры должна считываться нажатая клавиша, после чего происходит перемещение игрок или его взаимодействия с другими элементами поля.

Требования:

- Реализовать управление игрой. Считывание нажатий клавиш не должно происходить в классе игры, а должно происходить в отдельном наборе классов.
- Клавиши управления не должны жестко определяться в коде. Например, это можно определить в отдельном классе.
- Классы управления игрой не должны напрямую взаимодействовать с элементами игры (поле, клетки, элементы на клетках)
- Игру можно запустить и пройти.

Потенциальные паттерны проектирования, которые можно использовать:

- *Команда (Command) - передача команд с информацией через единый интерфейс. помещение команд в очередь*
- *Посредник (Mediator) - организация взаимодействия различных модулей*

Ход работы:

Для реализации управления игроком был создан класс *Controller* – от которого будут наследоваться конкретные реализации управления.

Класс контроллер имеет 4 поля кнопки (значение *int*) – отвечающие за «номер» кнопки, соответствующие 4 возможным направлениям передвижения *(_keyForwardDirection, _keyBackDirection, _keyRightDirection, _keyLeftDirection*) а так же четыре парных значения, отвечающие за вектор перемещения *(_forwardDiraction, _backDiraction, _rightDiraction, _leftDiraction*). Все вышеперечисленные атрибуты являются константами, и либо инициализируются при создании экземпляра класса, либо имеют значения по умолчанию.

Так же класс имеет поле Медиатора, класса, отвечающего за посредничество между командами управления и бизнес-логикой.

По функционалу класс *Controller* имеет две чистые функии: *KeyIsPressed()* – которая принимает в качестве аргумента номер нажатой клавиши и *Action()* – в котором происходит «уведомление» игрока о перемещении, и в котором передается вектор соответствующий перемещению.

На момент сдачи лабораторной работы реализован всего один наследник класса *Controller* – *QtController*, использующий в своей реализации коды символов перечеслений Qt (и так же подразумевает возможность расширения функционала класса, используя функции и библиотеки фрейморка Qt).

QtController содержит перегрузку чистых виртуальных функций класса Controller, а так же в его конструкторе указаны значения по умолчанию для клавиш (используя перечисления пространства имен Qt).

UML-диаграмма данного модуля представлена на рис «Controller».

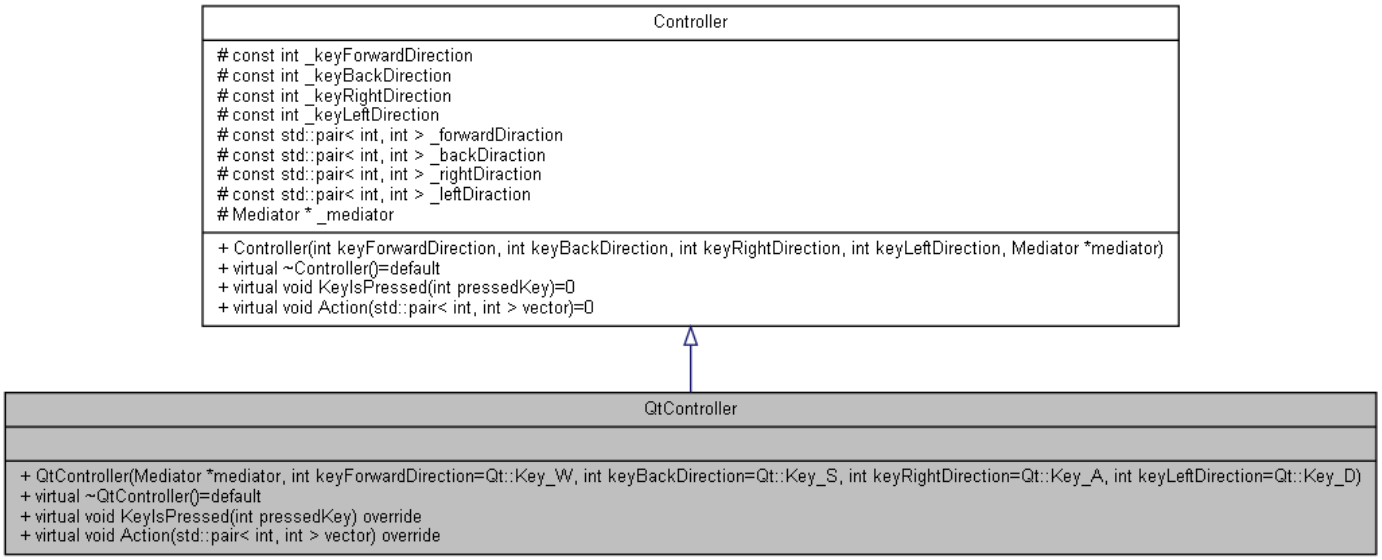

Рис. «Controller»

Считывание нажатой клавиши происходит в классе MainWindow, используя функцию *keyPressEvent()* фрейморка Qt, и получение из события номера нажатой клавиши и передача его через Медиатор в Controller (для этого Mediator был дополнен полем _controller – ссылкой на Controller, а так же функцией *notifyContoller()* – в которой и вызывается метод *KeyPressEvent()*)

Связи между Медиатором, главным окном GUI и главным классом игры приведены на рис «Game, Mediator, Controller»

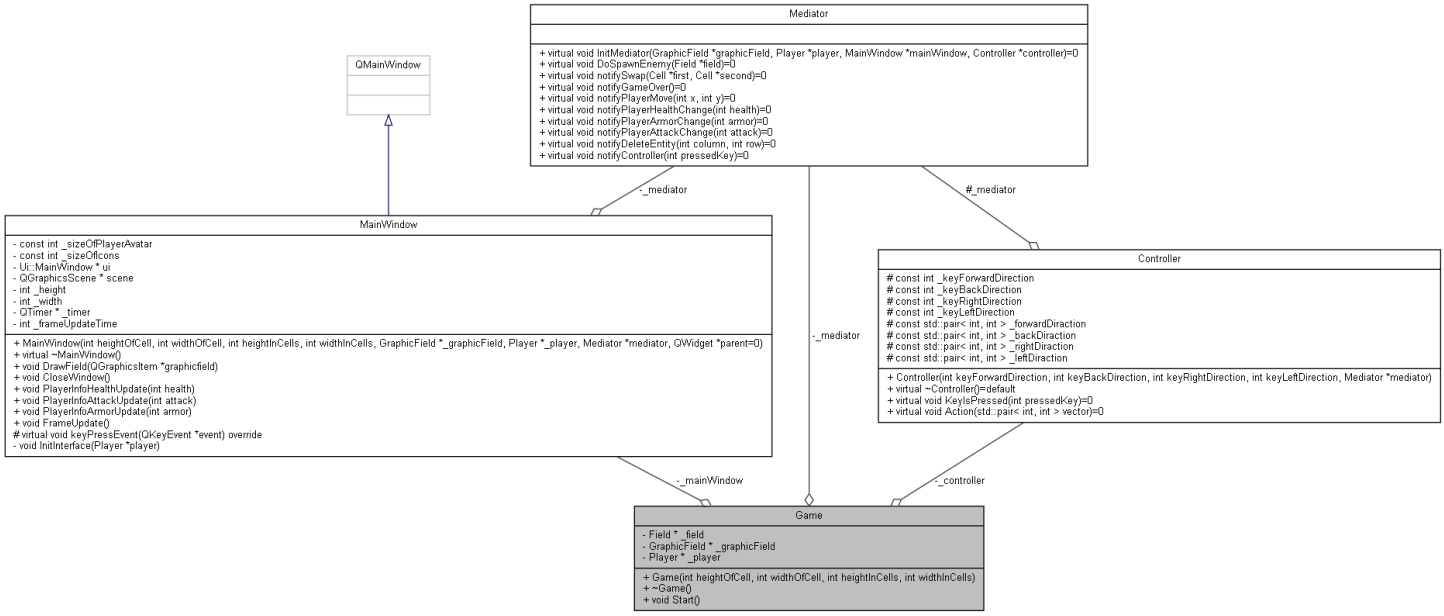

Рис.«Game, Mediator, Controller»

Разделение на уровни абстракций (отделение бизнес-логики от GUI, а так же добавленный в этой лабораторной работе класс Controller осуществляющий управление) – в большей степени было произведено еще на момент сдачи 2 л.р. Взаимодействие бизнес-логики, GUI и командами управления происходит через класс Медиатор.

Демонстрация роботоспособности и возможности завершить игру представлено на рис. «Демонстрация игры»

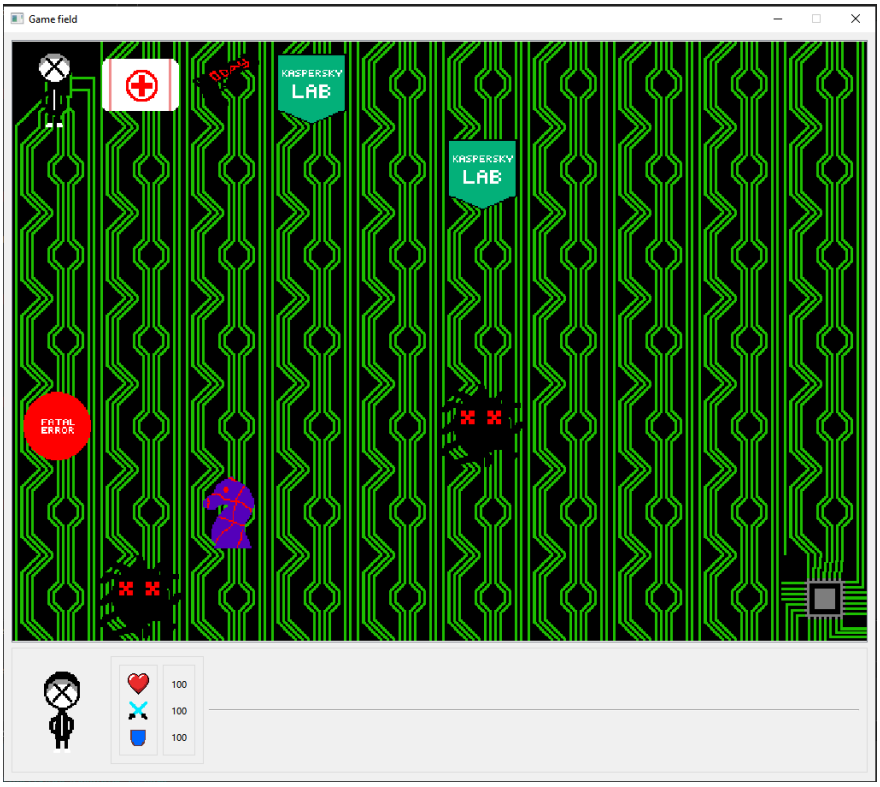

Рис. «Демонстрация игры»

Выводы:

В ходе выполнения лабораторной работы был реализован класс управления игрой (считывание клавиш происходит не в классе игры, а в классе главного графического окна, после чего информация о нажатой клавише передается через медиатор в класс управления игроком). В связи с внесенными изменениями был дополнен класс Mediator. Игру можно запустить и пройти. По созданному модулю создана UML-диаграмма.