**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

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

Тема: **Уровни абстракции**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2303 |  | Волков И.С. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2023

**Цель работы.**

Продумывание общей структуры игры для лучшей маштабируемости, создание классов, берущих на себя взаимодействие с пользователем – прослойки между игровой логикой и человеком.

**Задание.**

а) Создать класс игры. Класс игры отвечает за запуск игры (в начале и во время текущей сессии), выход из игры, выбор уровня, а также инициализирующего перемещение игрока. Также класс должен проводить проверку проигрыша или выигрыша игрока и давать возможность начать новую игру или завершить работу программы.

б) Создать класс (или набор классов) считывания ввода пользователя. Данный класс(ы) должен сопоставить ввод пользователя с командой, которую необходимо выполнить. И передать эту информацию в класс игры. Клавиши управления (то на какую клавишу назначено определенное действия) должны считываться из файла. Класс(ы) должен быть разработан так, чтобы:

* потенциально можно было масштабировать управление с минимальным количеством изменений в коде
* можно было заменить схему управления, например, заменить ввод команду через терминал на чтение команд из файла или из сети с другого устройства

Примечания:

* После считывания клавиши, считанный символ должен сразу обрабатываться, и далее работа должна проводить с сущностью, которая представляет команду.
* Для представления команды можно разработать системы классов или использовать перечисление enum.
* Хорошей практикой является создание “прослойки” между считыванием/обработкой команды и классом игры, которая сопоставляет команду и вызываемым методом игры. Существуют альтернативные решения без явной “прослойки”
* При считывания управления необходимо делать проверку, что на все команды назначена клавиша, что на одну клавишу не назначено две команды, что на одну команду не назначено две клавиши.
* При работе с файлом используйте идиому RAII

**Выполнение задания.**

Для выполнения задания реализованы классы: Game, Menu, lvl, I\_ PlayerInput, PlayerInput, Timer.

Game отвечает за запуск игры, выбор уровня, создание уровня (объекта lvl), удаление уровня и выход из игры. В классе только один метод – StartGame() – выполняет всё вышеперечисленное.

Menu реализует взаимодействие с игроком в меню – это выбор уровня и выход из игры. В классе только один метод – std::string MainMenu() – возвращает путь к папке с данными об уровне или пустую строку, если игрок решил закрыть игру.

Класс lvl – создаёт уровень, запускает игру, отслеживает изменения, в данной работе изменения – это ввод пользователем каких-либо команд, исполнение их и вызов отрисовки.

Поля:

* Player player – объект игрока
* Field field – объект игрового поля
* Controller contr – объект контроллера
* I\_PlyerInput pi – объект класса пользовательского ввода
* I\_Visualise v – объект класса визуализации поля
* std::map<int, Controlls> crls – словарь с командой и её кодовым представлением
* std::map<Symbols, char> visuals – словарь с символами на понятном программе языке и их версией для отрисовки для пользователя
* long long frameRate – время, которое должно пройти между 2-мя шагами (2-мя кадрами отрисовки)

Методы:

* lvl(std::string path, long long frameRate) – при инициализации на вход подаётся положение папки с файлами уровня и предполагаемое время кадра в миллисекундах, здесь же файлы читаются и данные записываются в поля
* void start() – запускает уровень (отслеживает пользовательский ввод, запускает отрисовку), проверяет, продолжается ли игра, выводит сообщения для пользователя
* ~lvl() – удаляет данные из полей

I\_PlayerInput – интерфейс пользовательского ввода, имеет метод action – возвращает команду, которую ввёл пользователь.

PlayerInput – реализация интерфейса I\_PlayerInput считывает пользовательский ввод посредством опроса о состоянии соответствующих клавиш, возвращает тип команды, которую ввёл пользователь в течении текущего кадра.

Поля:

* std::map<int, Controlls> crls – словарь команд и их кодов
* Timer ti – таймер, для контроля времени кадра
* long long frameRate – время кадра в миллисекундах
* Controlls c – команда, которую вернёт класс после считывания ввода

Методы:

* PlyerInput(std::map<int, Controlls>& crls, long long frameRate) – конструктор, принимает на вход обязательные поля
* Controlls action() – считывает и возвращает введённую пользователем команду

Timer отсчитывает время от момента вызова метода start().

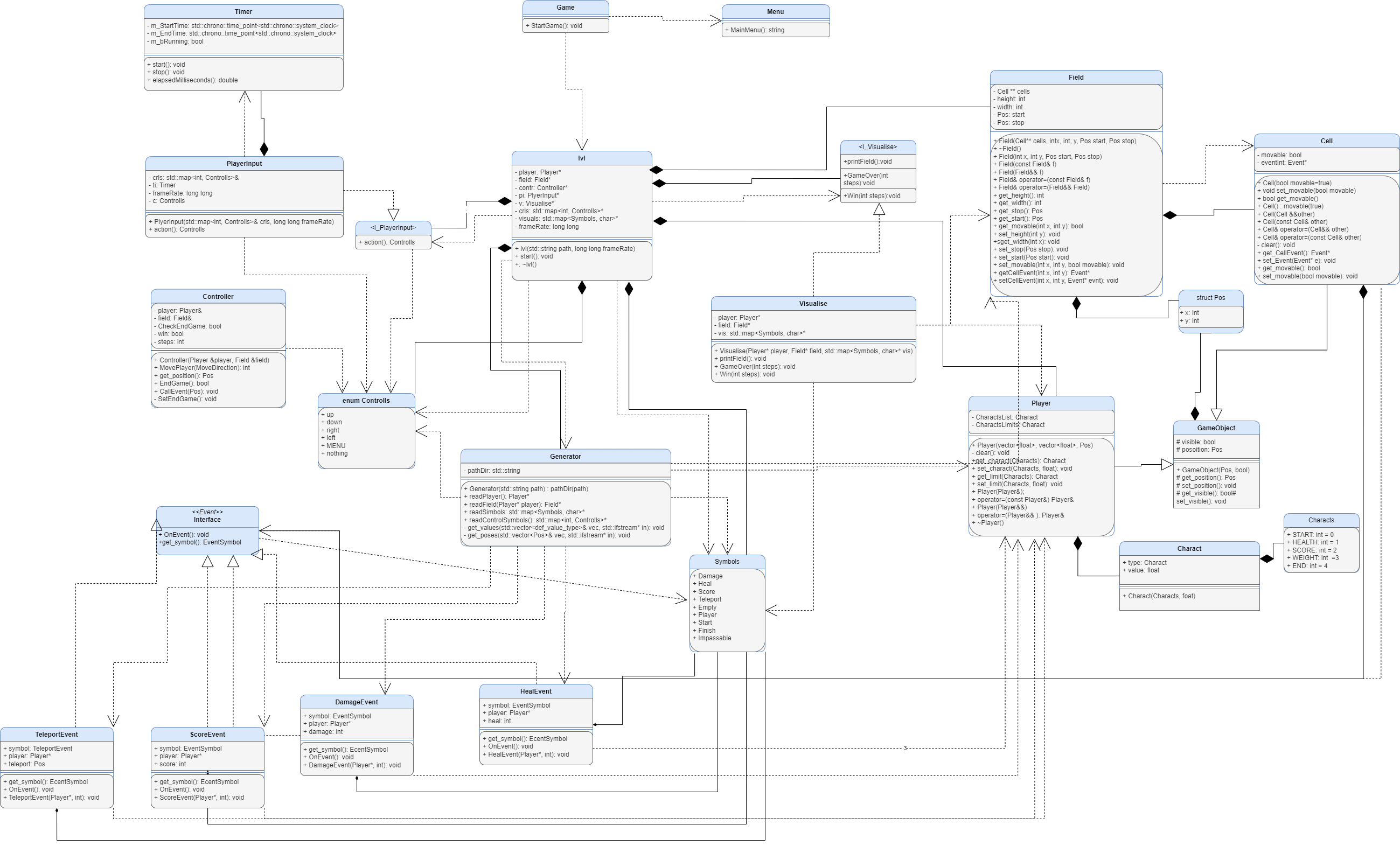
Поля:

* bool m\_bRunning – маркер, что таймер запущен
* std::chrono::time\_point<std::chrono::system\_clock> m\_StartTime – момент начала отсчёта
* std::chrono::time\_point<std::chrono::system\_clock> m\_EndTime – момент окончания отсчёта

Методы:

* void start() – запускает таймер – ставит маркер на true и записывает время начала
* void stop() – останавливает таймер – ставит маркер на false и записывает время остановки
* double elapsedMilliseconds() – возвращает количество миллисекунд, прошедшее с начала отсчёта

**UML**



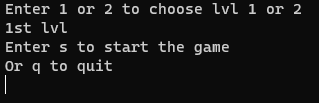
**Выводы.**

Реализованы необходимые классы с конструкторами, деструкторами и функционалом.

**Тестирование.**

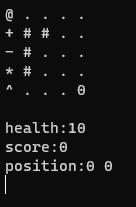
Проверка работы классов:

Здесь запущена программа, ввод – 1, выбран 1-ый уровень.



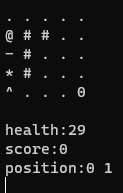
ввод – s

Запущен уровень, он отрисовывается корректно, программа ждёт ввода игрока

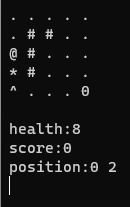


Сразу после ввода перерисовывается игровое поле, для ввода достаточно просто ввести символ, без подтверждения (нажатия enter).

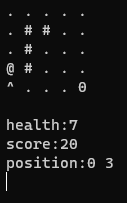
ввод – s



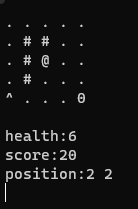
ввод – s



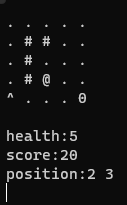
ввод – s



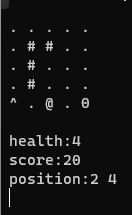
ввод – s



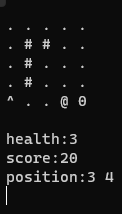
ввод – s



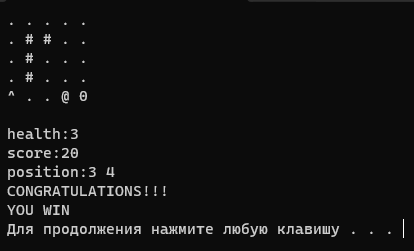
ввод – s



ввод – d



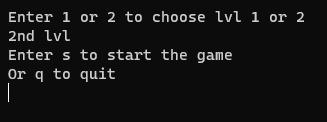
ввод – d



Все функции ввода работают корректно, отрисовка тоже, класс игры выполяет свою задачу.

Игра запущена второй раз, это 2-ой уровень.

ввод – 2



Ввод - s



Правила те же, что и в первом уровне (правила были перечислены на прошлой лабораторной работе). Сам уровень больше, загружается из файла.