**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: Шаблонные классы**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3383 |  | Земерова С.Н. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург 2025

**Цель работы**

Создать шаблонный класс управления игрой, который получается команду для выполнения и вызывает метод класса игры. Также необходимо создать шаблонный класс отображения игры, производящий отрисовку. Реализовать возможность сопоставлять клавише команду в файле с помощью класса, считывающего ввод из терминала. Сделать класс отрисовки поля.

**Задание**

1. Создать шаблонный класс управления игрой. Данный класс должен содержать ссылку на игру. В качестве параметра шаблона должен указываться класс, который определяет способ ввода команда, и переводящий введенную информацию в команду. Класс управления игрой, должен получать команду для выполнения, и вызывать соответствующий метод класса игры.
2. Создать шаблонный класс отображения игры. Данный класс реагирует на изменения в игре, и производит отрисовку игры. То, как происходит отрисовка игры определяется классом переданном в качестве параметра шаблона.
3. Реализовать класс считывающий ввод пользователя из терминала и преобразующий ввод в команду. Соответствие команды введенному символу должно задаваться из файла. Если невозможно считать из файла, то управление задается по умолчанию.
4. Реализовать класс, отвечающий за отрисовку поля.

**Примечание:**

* Класс отслеживания и класс отрисовки рекомендуется делать отдельными сущностями. Таким образом, класс отслеживания инициализирует отрисовку, и при необходимости можно заменить отрисовку (например, на GUI) без изменения самого отслеживания
* 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 считывании 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.

**Описание архитектурных решений и классов**

В ходе данной лабораторной работы был создан класс управления игрой (GameController), класс, считывающий ввод (InputHandler), класс отображения игры (GameRenderer), класс отрисовки поля (FieldRenderer).

**Класс GameController**

Шаблонный класс, отвечает за управление процессом игры. Связывает игровой процесс с пользовательским вводом и отображением, обеспечивая обработку команд игрока и обновление состояния игры.

Шаблонные параметры:

*InputHandler* — класс, отвечающий за обработку пользовательского ввода.

Этот объект предоставляет команды для управления игрой.

*RenderType* — способ визуализации игрового процесса. Поля класса:

*Game& game* — ссылка на объект Game, представляющий основную игровую логику.

*GameRenderer<RenderType>& gameRenderer* — ссылка на рендерер, который используется для отображения игры.

*InputHandler inputHandler* — объект обработчика пользовательского ввода. Методы класса:

*GameController(Game& game, GameRenderer<RenderType>& gameRenderer)-* конструктор класса. Инициализирует ссылки на объект игры и рендерера, создаёт обработчик ввода.

*int processInput()* – метод управления. Обрабатывает пользовательский ввод, полученный от *inputHandler*, и вызывает соответствующие методы игры или рендерера. Возвращает 1, если игра продолжается, 0 в противном случае. С помощью метода *getCommand* класса *InputHandler* получает команду. Обрабатывает её с помощью оператора switch.

**Класс InputHandler**

Отвечает за обработку пользовательского ввода. Позволяет связывать нажатия клавиш с игровыми командами, выводить доступные команды и загружать пользовательский файл с сопоставлением клавиш и команд.

Поля класса:

*std::unordered\_map<char, CommandType> commandsMapping –* неупорядоченный словарь, содержащий сопоставленную пару: клавиша (char) – команда (CommandType).

Методы класса:

*InputHandler() –* конструктор класса. Загружает клавиши и команды из файла key\_mapping.txt. Если файл не удается открыть, устанавливается управление по умолчанию методом defaultKeyMap.

*CommandType getCommand() -* метод ожидает пользовательский ввод клавиши и возвращает соответствующую команду. Если введенная клавиша не имеет пары, возвращает CommandType::UNKNOWN.

*void defaultKeyMap()* – устанавливает стандартное сопоставление клавиш с командами. (s – SAVE, l – LOAD, e – ENEMY\_FIELD, u – USER\_FIELD, c –

CONTINUE, q – QUIT, h – HELP). В конце выводит установленные пары.

*void loadKeyMap(const std::string& filename) –* загружает пользовательское сопоставление клавиш из файла. Отрывает файл читает пары ключ-команда из файла. Проверяет, что ключ не используется для другой команды, каждая команда имеет уникальный ключ (клавишу), для каждой команды установлена клавиша. Если все условия выполнены устанавливает пользовательское управление, иначе – вызвается метод defaultKeyMap.

*CommandType stringToCommand(const std::string& command)* – преобразует строковое представление команды в CommandType.

*std::string commandToString(CommandType type)* – преобразует CommandType в строку.

*void help()* - выводит список доступных команд и соответствующих им клавиш.

**Класс GameRenderer**

Шаблонный класс, отвечает за визуализацию состояния игры. Использует переданный тип рендера (RenderType) для отображения игрового поля. Таким образом, позволяет использовать различные типы рендеринга.

Поля класса:

*const GameState& gameState –* ссылка на объект GameState, представляющий текущее состояние игры. Используется для доступа к данным, которые будут отображаться.

*RenderType rendererType* - экземпляр шаблонного типа RenderType, отвечающего за метод визуализации. Предоставляет метод render для отображения.

Методы класса:

*explicit GameRenderer(const GameState& gameState)* - конструктор класса. Принимает ссылка на объект текущего состояния игры. Сохраняет ссылку на состояние для последующей визуализации.

*void render(bool isUserField)* – выполняет визуализацию игры. Переменная *isUserField –* флаг, указывающий какое поле будет отображаться (true – пользователя, false – врага). Вызывает метод render у объекта rendererType, передает состояние игры и флаг.

**Класс FieldRenderer**

Отвечает за визуализацию игровых полей на основе текущего состояния

игры.

Поля класса:

*Enum class CellState*

Перечисление, представляющее состояния клетки на игровом поле. Каждое

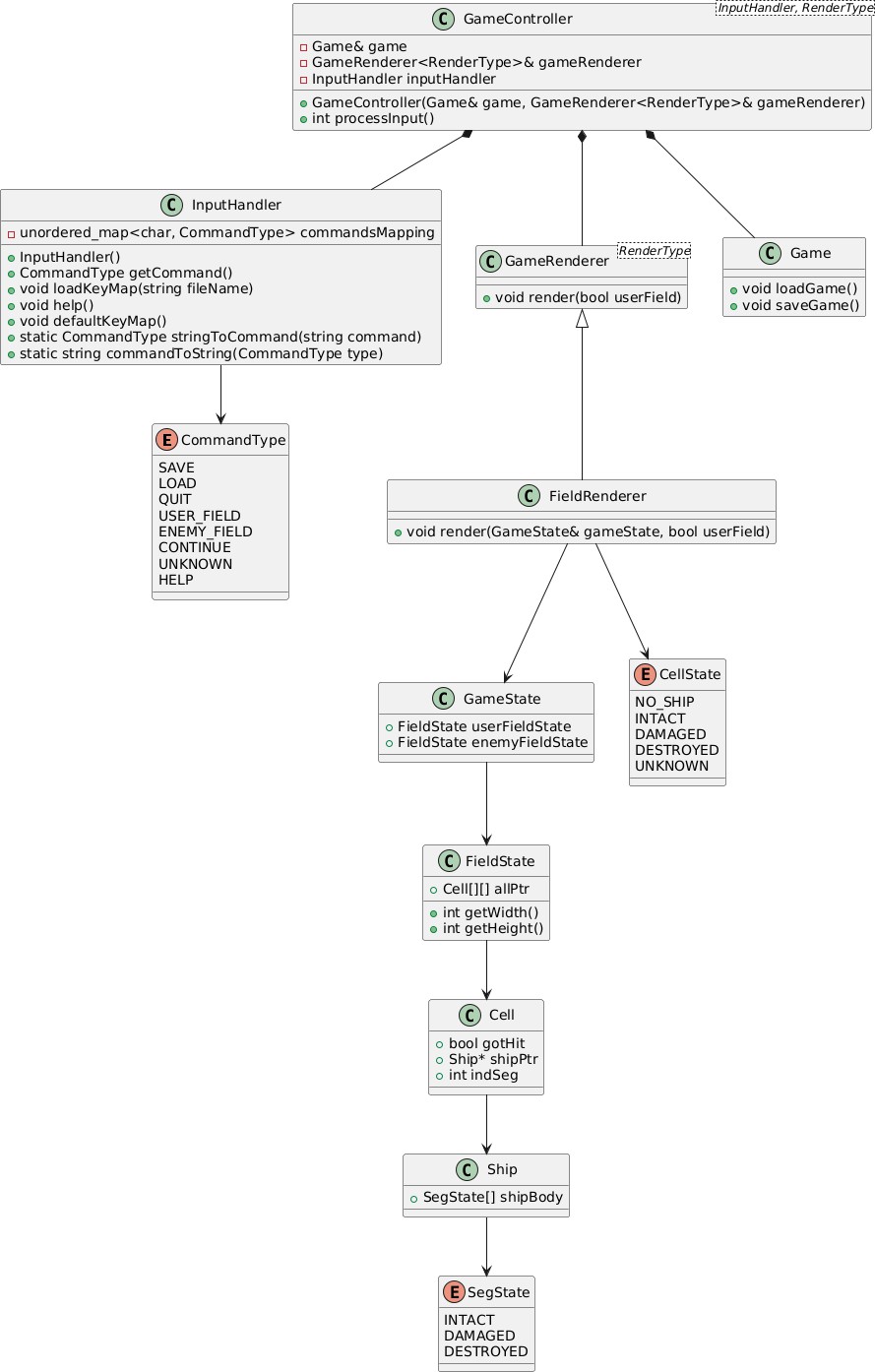
состояние отображается соответствующим символом:

* NO\_SHIP (~) — клетка без корабля.
* DESTROYED (0) — разрушенная клетка.
* DAMAGED (1) — поврежденный сегмент корабля.
* INTACT (2) — целый сегмент корабля.
* UNKNOWN (?) — неизвестное состояние (используется для поля противника).

Методы класса:

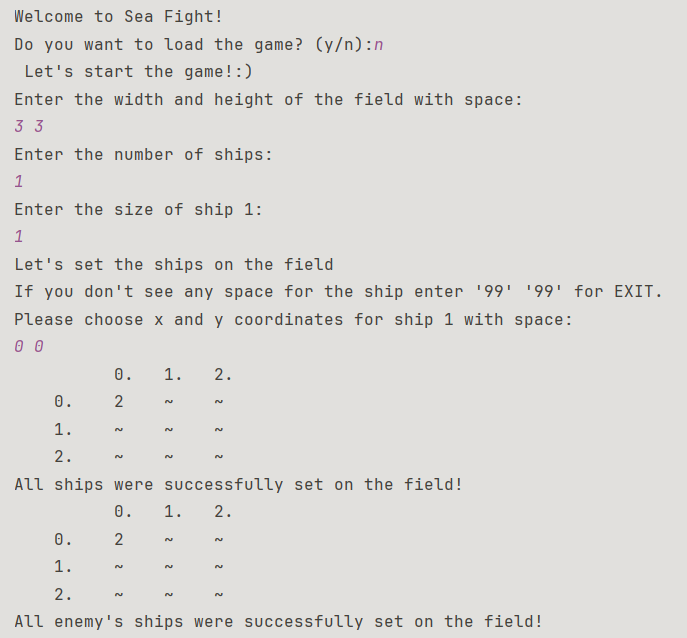
*void render(const GameState& gameState, bool userField)* - статический метод, который выводит игровое поле в консоль. Отображает строки с номерами колонок и строк, начиная с 0. Проходит по каждой клетке поля и выводит соответствующий символ enum класса CellState. Использует std::setw(5) для выравнивания столбцов.

**UML – диаграмма классов**

****

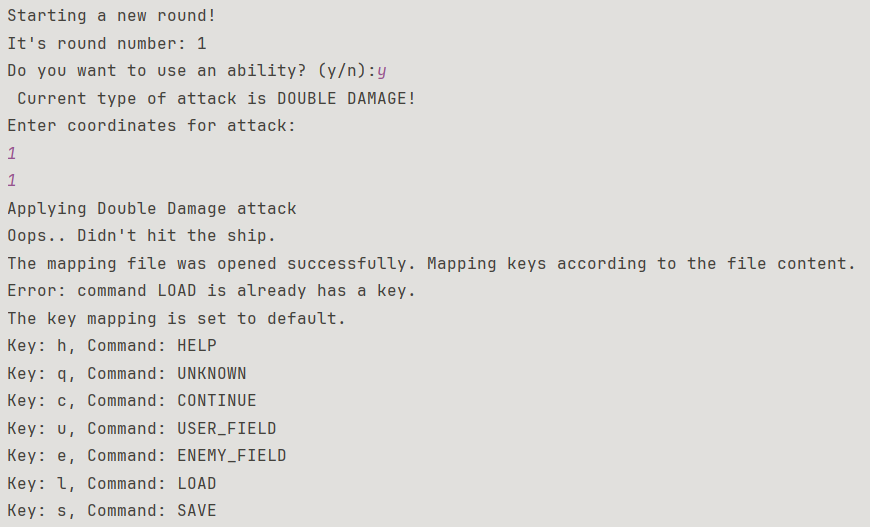
**Тестирование**

Демонстрация поля пользователя во время расстановки кораблей.

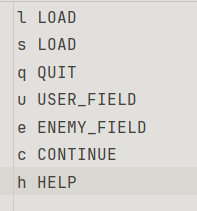


Начало первого раунда и ход пользователя. Применение способности и

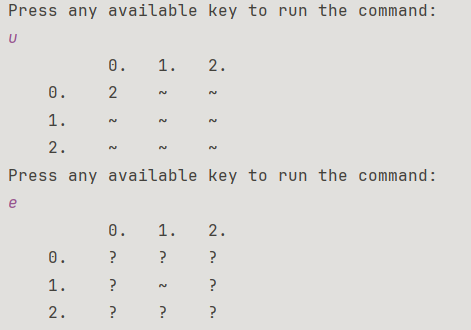
последующая атака. Вывод доступных команд и соответствующих клавиш.



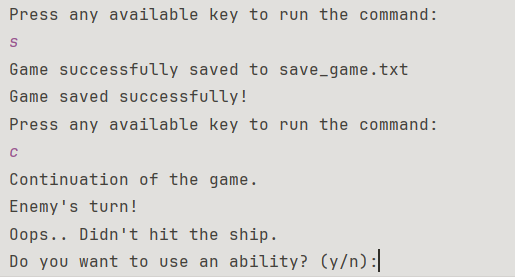
Содержимое файла key\_mapping.txt



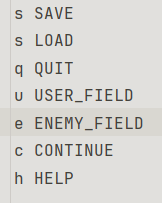
Вывод пользовательского и вражеского поля.

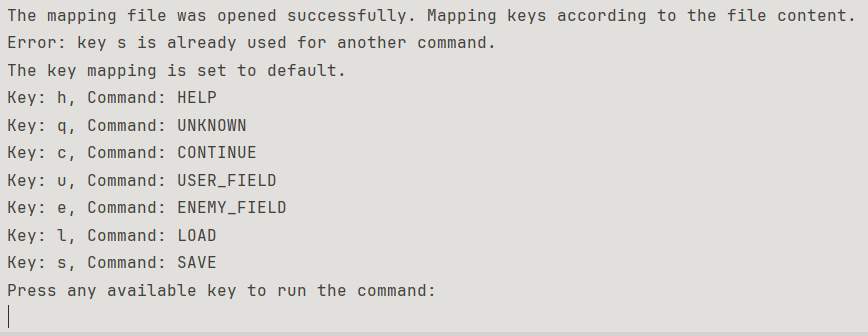


Сохранение и её продолжение.

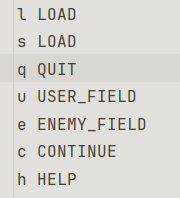


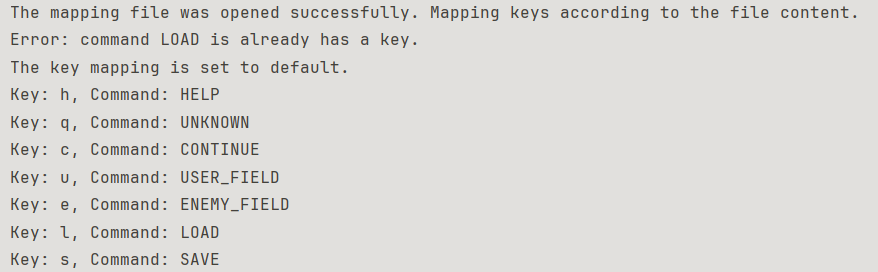
Некорректный файл key\_mapping.txt (одна клавиша назначена для двух команд)



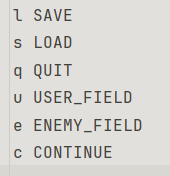
Работа программы в данном случае.

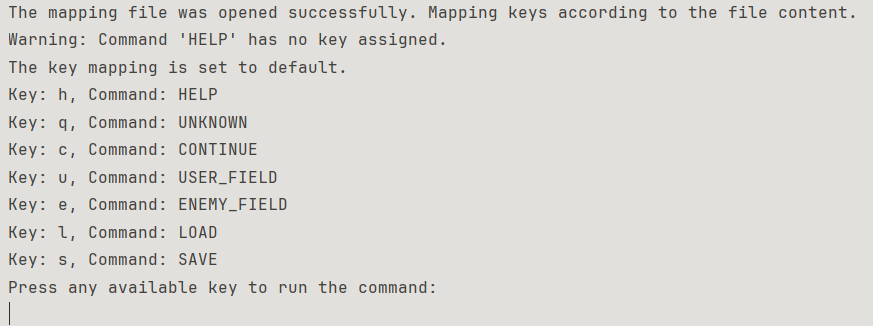
Некорректный файл key\_mapping.txt (для команды назначено несколько клавиш)



Работа программы в данном случае.

Некорректный файл key\_mapping.txt (клавиши назначены не для всех команд)



Работа программы в данном случае.

**Выводы**

В ходе выполнения лабораторной работы был создан шаблонный класс управления игрой, шаблонный класс отображения игры, класс, считывающий ввод из терминала, класс отрисовки поля. Класс управления игрой поддерживает различные типы визуализации без изменения основной логики благодаря использованию шаблона. Была реализована возможность загрузки пользовательского сопоставления клавиш с командами из файла. Класс рисовки игры также можно адаптировать для различных интерфейсов, предоставив другую реализацию RenderType.