**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

|  |  |  |
| --- | --- | --- |
| Студент гр. 3383 |  | Галкин А.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2024

**Цель работы.**

Изучение работы с шаблонами и шаблонными классами в языке программирования С++ на примере создания шаблонного класса-контроллера и класса-отрисовщика игры «Морской бой». Создать на основе изученного материала требуемые классы и тем самым реализовать полноценную логику игры «Морской бой» с наличием управления и отрисовкой.

**Задание на лабораторную работу**

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.

**Выполнение работы.**

Во время выполнения лабораторной работы реализованы шаблонные классы GameController и GameRenderer, реализующие управление игрой и ее отрисовку, соответственно. Также, реализованы классы Tracker, выполняющий роль отслеживания и реагирующий на изменения в игре, TerminalField – класс, который может быть подан в качестве шаблона для GameRenderer, тем самым реализуя отрисовку игры в консоли и класс TerminalHandler, который может быть подан в качестве шаблона класса GameController, реализуя управление игрой в соответствии с вводом пользователя в консоли.

Класс GameController. Класс управления игрой GameController имеет конструктор, принимающий на вход ссылку на игру Game и инициализирующий полученной ссылкой хранящуюся в приватном поле ссылку на игру game\_. Класс GameController имеет единственный метод – Start(), содержащий в себе бесконечный игровой цикл до момента, пока игрок не нажмет кнопку, отвечающую за команду выхода из игры. В данном цикле метод реагирует на ввод пользователя, получая введенную команду при помощи метода GetCommand() приватного поля InputType input. InputType – шаблонный тип, подаваемый в объект класса при его создании. Для осуществления контроля игрой реализован класс перечисления enum ComandType, содержащий 15 различных команд.

Класс TerminalInputHandler. Класс обработки вводимых пользователем команд (нажимаемых клавиш на клавиатуре) имеет конструктор, в котором происходит попытка считывания управления из файла при помощи метода ReadCommands(). В случае, если считать управление не удалось (либо не на все команды назначены клавиши, либо на две клавиши назначена одна команда, либо на одну клавишу назначены две команды), управление задается по умолчанию. Метод ReadCommands() открывает файл с управлением при помощи класса FileHandler и считывает содержимое построчно, проверяя при помощи множества std::set, была ли уже использована очередная полученная команда, также проверяющая при помощи std::map, была ли использована клавиша, а после – проверяющая, что количество клавиш управления совпадает с количеством команд (15).

Метод GetCommand() класса TerminalInputHandler ожидает нажатия клавиши в цикле while с временем «сна» 50 мс. В случае, если клавиша была нажата, из метода возвращается соответствующая данной клавише команда из ассоциативного массива commands\_.

Класс Tracker. Данный класс реализует контроль над происходящими изменениями в игре и является прослойкой между классом GameController, совершающим действия согласно введенному от пользователя, и шаблонным классом отрисовки GameRenderer. Класс Tracker имеет систему триггеров, поднимающих определенные флаги и вызывающих методы отрисовки, а также хранит информацию о том, какое действие было совершено. Другими словами, класс Tracker – полноценный класс отслеживания и соответствующего реагирования на все производимые в игре действия. Класс содержит соответствующие триггерам флаги в качестве приватных полей, к которым можно получить доступ при помощи методов-сеттеров, а также – координаты местоположения курсора пользователя в определенный момент времени, вводимые им размеры поля и кораблей для корректной отрисовки. Существующие триггеры: триггер начала игры, триггер победы, триггер поражения, триггер начала раунда, триггер создания поля, триггеры изменения координаты, триггеры изменения ориентации корабля, триггеры изменения размера корабля, триггеры совершения ходов игроком и компьютером, триггер загрузки из файла (необходим для получения считанного поля и его отрисовки), триггер постановки корабля и триггер того, что пользователю необходимо вывести информацию (например, ошибку).

Класс GameRenderer. Шаблонный класс отрисовки игры принимает в качестве параметра шаблона метод отрисовки RenderType. Класс имеет приватное поле отрисовщика RenderType render\_. Имеется набор методов для реагирования на события игры и соответствующей отрисовки. Данные методы вызываются из описанного выше класса Tracker. Сами методы вызывают соответствующие методы у самого отрисовщика – приватного поля render\_ типа RenderType.

Класс TerminalField. Реализует отрисовку игры в консоли. Подается в качестве параметра шаблона для класса GameRenderer и содержит все методы, необходимые при отрисовке: метод полной отрисовки поля игрока/компьютера, метод отрисовки ставящегося корабля, метод зачистки (удаления из консоли) корабля при его передвижении или смене ориентации, метод отрисовки совершенной атаки в зависимости от ее результата, метод отрисовки полей при загрузке из файла (принимает на вход считанное из файла поле в виде сериализованного объекта поля – массива строк для отрисовки).

Результаты тестирования и UML-диаграмма реализованных классов представлены в Приложении А и Приложении Б, соответственно.

**Выводы.**

В ходе выполнения лабораторной работы, была изучена работа с шаблонными классами в языке программирования С++. Реализованы два шаблонных класса управления игрой и отрисовки игры, принимающие как параметры шаблонов классы отрисовки и ввода команд. Также реализованы классы считывания ввода пользователем и отрисовки в консоли, которые подаются в качестве параметров шаблонов соответствующих классов.

**ПРИЛОЖЕНИЕ А**

**РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ**

Проверка корректности построения поля

|  |  |  |
| --- | --- | --- |
| Ввод | Вывод программы | Комментарий |
|  |  | ОК |

Проверка корректности отрисовки после загрузки сохранения

|  |  |  |
| --- | --- | --- |
| Ввод | Вывод программы | Комментарий |
|  | - сохраненное состояние  - произведено несколько ходов  - загружено сохранение | ОК |

Проверка корректности отрисовки начала нового раунда

|  |  |  |
| --- | --- | --- |
| Ввод | Вывод программы | Комментарий |
|  | - состояние перед победой (остался один ход игроку)  - после победы начался новый раунд и поле противника покрылось «туманом войны» | ОК |

**ПРИЛОЖЕНИЕ Б**

**UML-ДИАГРАММА**

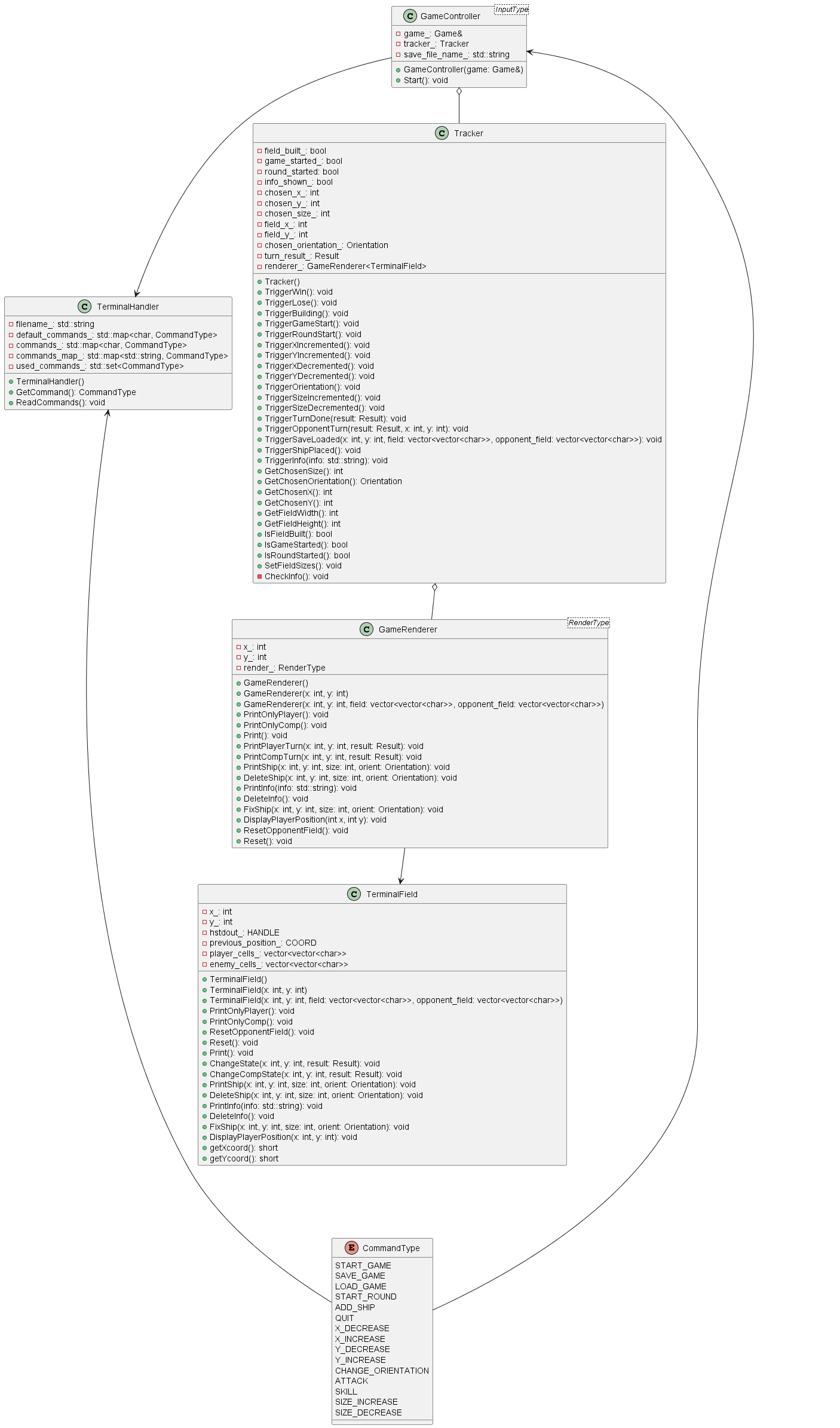


Рисунок 1 - UML-диаграмма реализованных классов