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. 3385 Баринов М.А

Преподаватель _____ Жангиров Т.Р.

_____

Санкт-Петербург

2024

Цель работы

Разработать простой проект для игры в Морской бой.

Задание

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.

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.

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.

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 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.

Выполнение работы

Класс **GameController**

Этот класс является контроллером для управления игрой. Он использует шаблоны для обработки ввода и стратегии рендеринга.

Методы:

GameController(Game& game): Конструктор, который принимает ссылку на объект Game и инициализирует его.

void startGame(int height, int width, std::vector<int> lens): Метод для запуска новой игры. Он устанавливает параметры игрока (высота, ширина, длины кораблей) и запускает игру с использованием переданных шаблонов InputHandler и RenderStrategy.

Класс **InputHandler**

Этот класс отвечает за обработку ввода пользователя. Он загружает сопоставления клавиш с командами из файла и предоставляет метод для получения ввода.

Методы:

InputHandler(): Конструктор, который загружает сопоставления клавиш из файла a.txt.

std::string getInput(): Метод для получения ввода от пользователя. Он считывает символ с клавиатуры и возвращает соответствующую команду из словаря keyBindings.

void loadKeyBindings(const std::string& filename): Метод для загрузки сопоставлений клавиш из файла. Если файл не открывается или содержит ошибки, используются настройки по умолчанию.

void defaultKeyBindings(): Метод для установки настроек по умолчанию, если файл с сопоставлениями не найден или содержит ошибки.

Класс **Render**

Этот класс отвечает за рендеринг игры в консоль.

Методы:

void clearConsole() const: Метод для очистки консоли. Используется system("cls") для Windows и system("clear") для других платформ.

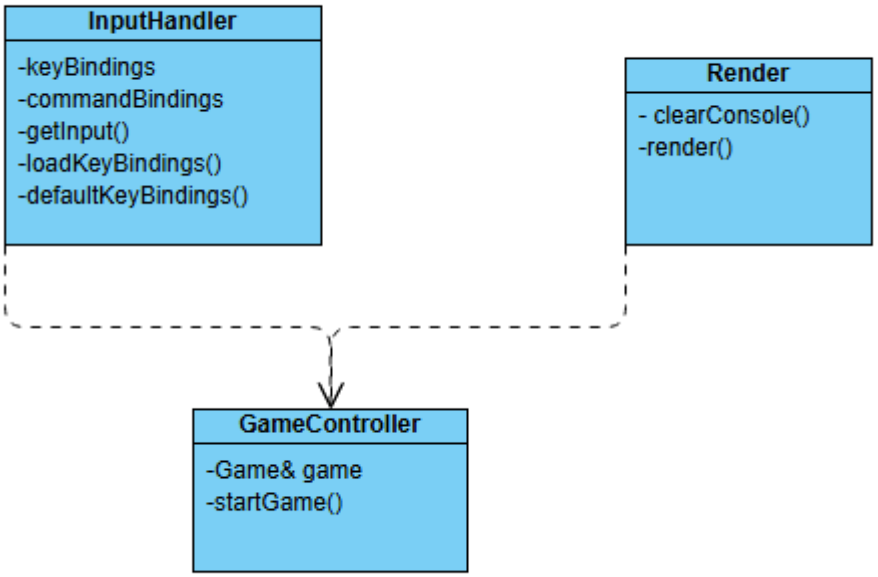
void render(Game &game): Метод для рендеринга текущего состояния игры. Он очищает консоль, выводит статус игрока и врага, а также список доступных команд.

Основная функция main()

Это точка входа в программу. Она создаёт объект Game, инициализирует GameController с использованием InputHandler и Render, а затем запускает игру с заданными параметрами.

Для каждого класса были созданы заголовочные файлы.

UML схема классов:



Выводы.

В ходе выполнения лабораторной работы была разработана архитектура игры, включающая основные компоненты: контроллер игры, обработчик ввода, рендерер, а также классы, представляющие игрока, врага и саму игру.