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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Объектно-ориентированное программирование»

Тема: Связывание классов

Студент гр. 3385 _____ Комаренко Т.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2024

Цель работы

Цель задания — создать класс игры, который реализует игровой цикл с чередующимися ходами игрока и компьютерного врага. Игрок может использовать способности и атаковать, а враг только атакует. При поражении игрока начинается новая игра, а при победе продолжается следующий раунд с сохранением состояния поля и способностей. Класс должен включать методы для управления игрой, начала новой игры, выполнения ходов и сохранения/загрузки игры. Также необходимо переопределить операторы ввода/вывода для состояния игры и реализовать сохранение, которое можно загрузить после перезапуска программы, используя идиому RAII для работы с файлами.

Задание

Создать класс игры, который реализует следующий игровой цикл:

Начало игры

Раунд, в котором чередуются ходы пользователя и компьютерного врага. В свой ход пользователь может применить способность и выполняет атаку. Компьютерный враг только наносит атаку.

В случае проигрыша пользователь начинает новую игру

В случае победы в раунде, начинается следующий раунд, причем состояние поля и способностей пользователя переносятся.

Класс игры должен содержать методы управления игрой, начало новой игры, выполнить ход, и т.д., чтобы в следующей лаб. работе можно было выполнять управление исходя из ввода игрока.

Реализовать класс состояния игры, и переопределить операторы ввода и вывода в поток для состояния игры. Реализовать сохранение и загрузку игры. Сохраняться и загружаться можно в любой момент, когда у пользователя приоритет в игре. Должна быть возможность загружать сохранение после перезапуска всей программы.

Примечание:

Класс игры может знать о игровых сущностях, но не наоборот

Игровые сущности не должны сами порождать объекты состояния

Для управления самое игрой можно использовать обертки над командами

При работе с файлом используйте идиому RAII.

Выполнение работы

Класс Game

Класс Game представляет собой основной класс для управления игровой логикой, включая чередование ходов игрока и компьютерного врага, а также сохранение и загрузку состояния игры. Он взаимодействует с игровыми полями, менеджерами кораблей и менеджером способностей с помощью объекта класса состояния игры.

Поля класса Game:

- gameState: указатель на объект класса GameState, который хранит текущее состояние игры, включая поля, менеджеры кораблей, способности.

Методы класса Game:

- Game(GameState* gameState): Конструктор, инициализирующий поле gameState.
- void startNewGame(): Запускает цикл игры где чередуются ход игрока и ход противника.
- void playRound(): метод определяющий цикл игры.
- void userTurn(): Выполняет ход игрока.
- void enemyTurn(): выполняет ход противника, атакуя случайную клетку поля игрока.
- bool isRoundOver(): Проверяет закончилась ли игра := одно из полей более не содержит “живых” кораблей.
- void collocationEnemyShips(): распределяет в случайным образом корабли на поле противника.
- void save(): сохраняет текущее состояние игры в файл.
- void load(): загружает состояние игры из файла.

- void createEnemiesItems (): определяет игровые сущности противника.
- void createNewShipsManager(): создает менеджер кораблей противника.

Класс GameState

Класс GameState представляет собой структуру данных, которая хранит все ключевые элементы состояния игры, включая поля, менеджер кораблей, менеджер способностей. Этот класс используется для управления состоянием игры, его сохранения в файл и инициализации из файла.

Поля класса GameState:

- userManager: указатель на объект менеджера игрока (тип Manager*).
- enemyManager: указатель на объект менеджера противника (тип Manager*).
- userfield: указатель на объект поля игрока (тип GameMap*).
- enemyfield: указатель на объект поля противника (тип GameMap*).
- abilManager: указатель на объект класса менеджера способностей(тип AbilityManager*).
- userShipsCoord: вектор пар хранящий координаты на которых стоят корабли игрока.
- enemyShipsCoord: вектор пар хранящий координаты на которых стоят корабли противника.

Методы класса GameState:

- GameState(): Конструктор по умолчанию.

- void loadAll(std::string data): метод который из полученной строки с данными инициализирует новые игровые сущности.
- std::string saveAll(): Метод который записывает необходимые для сохранения данные в строку и возвращает ее.
- void clearFile(const std::string& filename): Метод для отчистки файла.
- unsigned int calculateChecksum(const std::string& data): Метод для составления чек суммы по строке с данными.
- void setUserField(GameMap* userField): Метод устанавливающий новой обьект поля игрока.
- GameMap* getUserField(): метод возвращающий указатель на поле игрока.(далее подобным образом определены методы для получения и установки игровых сущностей, таких как: enemyFiled, enemyManager, userManager, abilManager).
- void addUserCoord(int x, int y): метод добавляющий новые координаты для корабля игрока.
- void addEnemyCoord(int x, int y): метод добавляющий новые координаты для корабля противника.
- std::pair<int,int> getUserCoord(int idx): метод возвращающий координаты корабля игрока по индексу.
- std::pair<int,int> getEnemyCoord(int idx): метод возвращающий координаты корабля противника по индексу.
- void cleanUserCoord(): метод очистки вектора пар с координатами кораблей игрока.
- void cleanEnemyCoord(): метод очистки вектора пар с координатами кораблей противника.

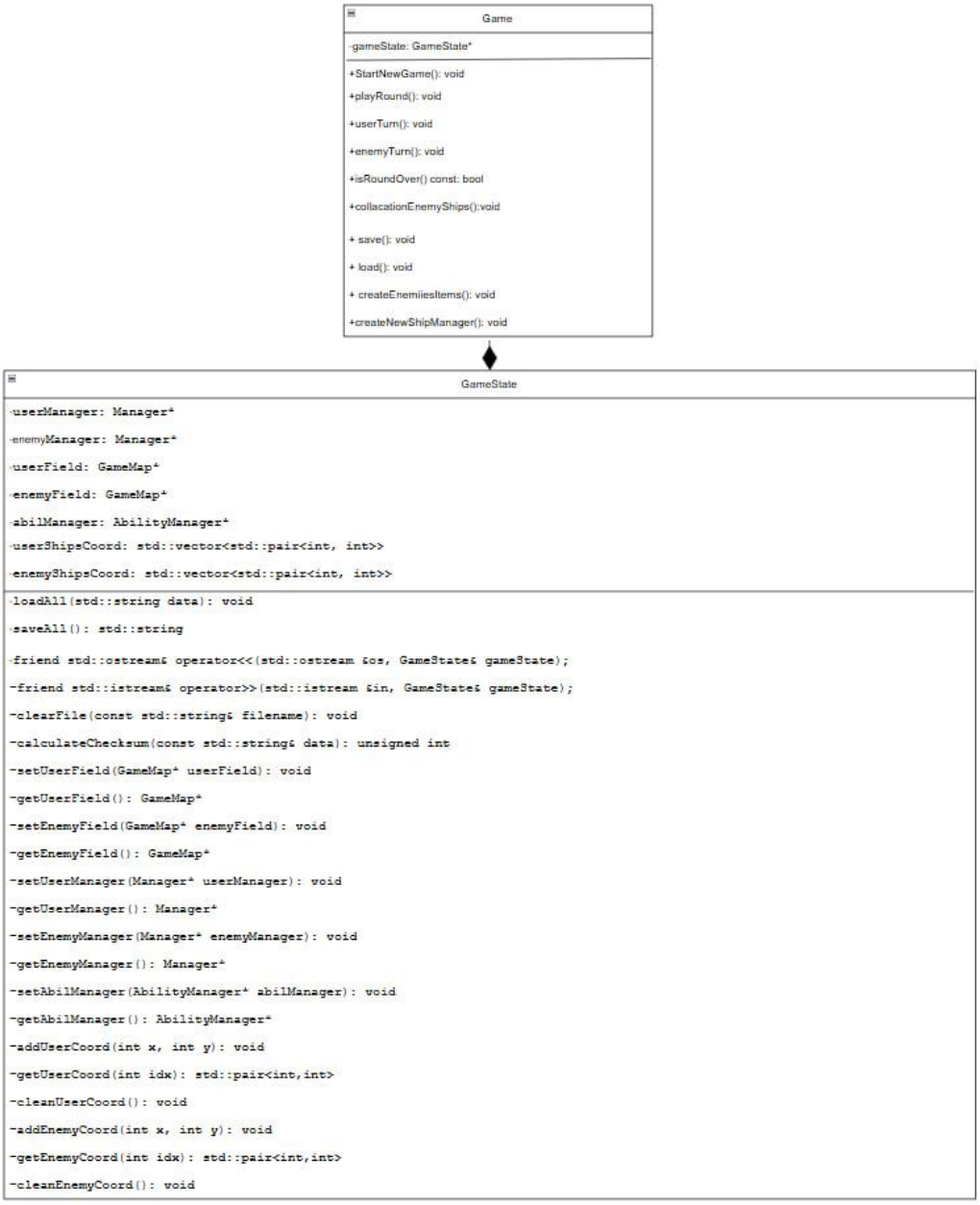
Операторы ввода/вывода:

- friend std::ostream& operator<<(std::ostream &os, GameState& gameState): переопределенный оператор вывода.
- friend std::ostream& operator>>(std::ostream &in, GameState& gameState): переопределенный оператор ввода.

main()

В программе создается игра, в которой игрок размещает свои корабли на поле, а затем поочередно делают ходы. Для каждого игрока создаются менеджеры кораблей и поля боя. Центральным элементом игры является класс Game, который организует игровой процесс, выполняет ходы игрока и противника, а также обеспечивает возможность сохранения и загрузки текущего состояния игры.

UML-диаграмма классов



Выводы

Была создана система управления игрой "Морской бой", реализующая чередующийся игровой процесс с ходами игрока и противника. Игрок может использовать различные способности и атаковать, тогда как враг ограничен исключительно атакующими действиями. В случае поражения игрока игра перезапускается, а при победе продолжается новый раунд с сохранением состояния поля и доступных способностей.

Класс игры предоставляет методы для управления ходами, запуска новой игры, сохранения и загрузки текущего состояния. Для упрощения сохранения и восстановления игрового процесса реализованы операторы ввода и вывода. Использование идиомы RAII при работе с файлами гарантирует правильное управление ресурсами. Такой подход обеспечивает надежность, гибкость игрового процесса, удобство для пользователей и возможность дальнейшего расширения функционала системы.