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. 3342 _____ Бобовский А. К.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2024

Цель работы

Научиться связывать классы.

Задание

a. Создать класс игры, который реализует следующий игровой цикл:
   i. Начало игры
   ii. Раунд, в котором чередуются ходы пользователя и компьютерного врага. В свой ход пользователь может применить способность и выполняет атаку. Компьютерный враг только наносит атаку.
   iii. В случае проигрыша пользователь начинает новую игру
   iv. 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.

b. 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

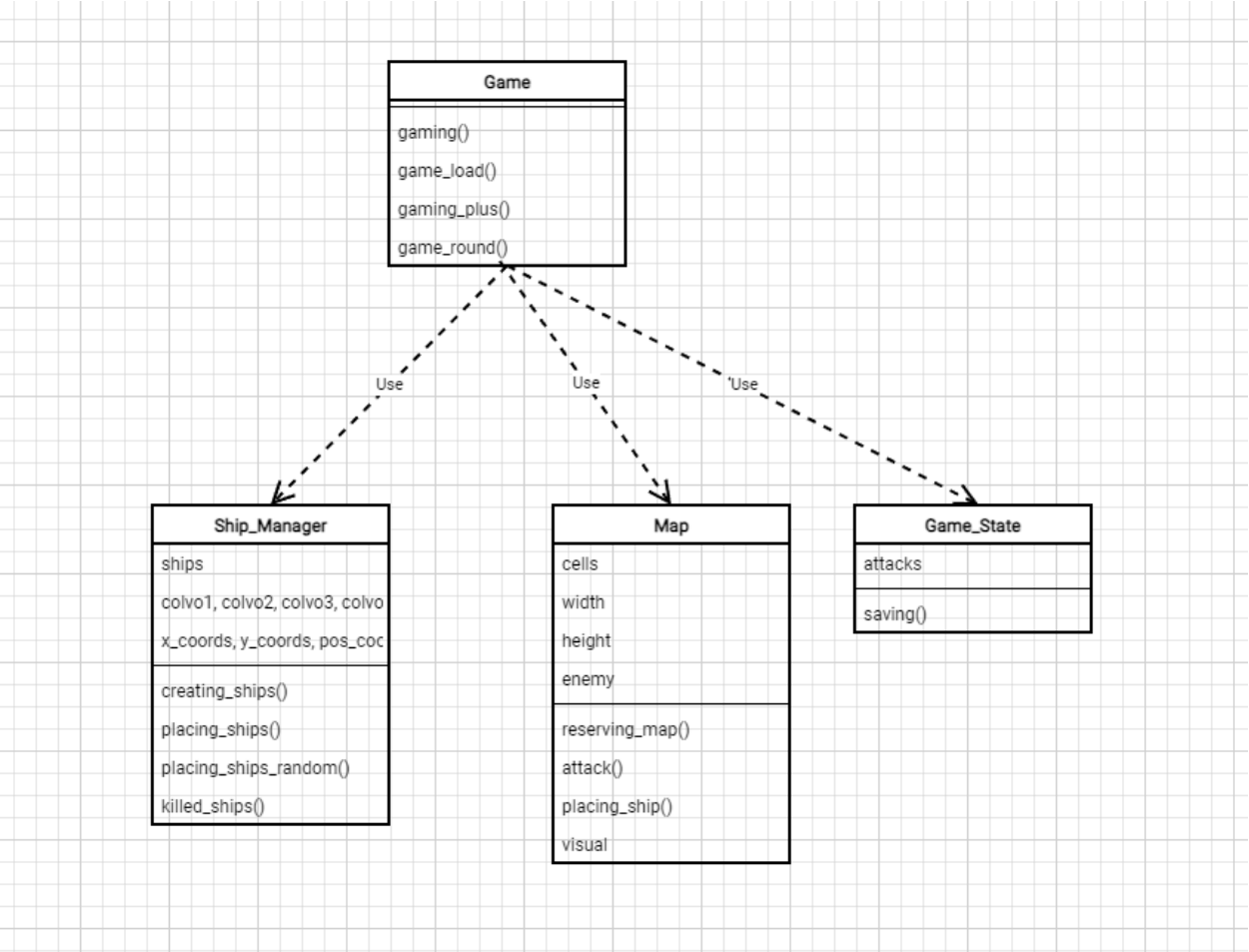
При выполнении работы были созданы:

1)Класс Game. В классе Game используются методы gaming(), gaming_plus(), gaming_load() и game_round(). Этот класс отвечает за воспроизведение игрового цикла. Метод gaming() воспроизводит игру как обычно с самого начала, метод с самого начала, метод gaming_plus() позволяет начать новую игру плюс после победы, сохранив текущее состояние своего поля, но получив нового противника. Метод gaming_load() нужен, чтобы загружать сохранённую в сейв-файле информацию и начинать игру с того момента.

2) Класс Game_State. Метод saving этого класса сохраняет важную информацию об игре в сейв-файл.

3) Во многие старые классы во время работы тоже были внесены изменения, чтобы сделать программу из испытательного полигона прототипом игры. В класс ship_manager нужно было добавить метод случайного заполнения поля кораблями, а в класс map метод visual, который позволит видеть игровое поле в командной строке.

UML-ДИАГРАММА



Вывод

В процессе выполнения лабораторной работы было изучено связывание классов. Морской бой впервые оформился как настоящая игра, в которую можно поиграть от начала до конца.