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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Объектно-ориентированное программирование»

Тема: «Создание классов, конструкторов и методов»

Студент гр. 1384 _____ Белокобыльский И. В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2022

Цель работы.

Научиться создавать классы, конструкторы и методы на языке C++, реализовывать межклассовые связи.

Задание.

Реализовать систему событий. Событие - сущность, которая срабатывает при взаимодействии с игроком. Должен быть разработан класс интерфейс общий для всех событий, поддерживающий взаимодействие с игроком. Необходимо создать несколько групп разных событий реализуя унаследованные от интерфейса события (например, враг, который проверяет условие, будет ли воздействовать на игрока или нет; ловушка, которая безусловно воздействует на игрока; событие , которое меняет карту; и.т.д.). Для каждой группы реализовать конкретные события, которые по разному воздействуют на игрока (например, какое-то событие заставляет передвинуться игрока в определенную сторону, а другое меняет характеристики игрока). Также, необходимо предусмотреть событие “Победа/Выход”, которое срабатывает при соблюдении определенного набора условий.

Реализовать ситуацию проигрыша (например, потери всего здоровья игрока) и выигрыша игрока (добрался и активировал событие “Победа/Выход”)

Требования:

Разработан интерфейс события с необходимым описанием методов

Реализовано минимум 2 группы событий (2 абстрактных класса наследников события)

Для каждой группы реализовано минимум 2 конкретных события (наследники от группы события)

Реализовано минимум одно условное и безусловное событие (условное - проверяет выполнение условий, безусловное - не проверяет).

Реализовано минимум одно событие, которое меняет карту (меняет события на клетках или открывает расположение выхода или делает какие-то

клетки проходимыми (на них необходимо добавить события) или не непроходимыми

Игрок в гарантированно имеет возможность дойти до выхода

Примечания:

Классы событий не должны хранить никакой информации о типе события (никаких переменных и функций дающие информации о типе события)

Для создания события можно применять абстрактную фабрику/прототип/строитель

Выполнение работы.

Класс GameMediator хранит в себе все необходимые контроллеры и реализует связи между ними, являясь посредников во взаимодействии ViewController и внутренней логики программы. В дальнейшем будет необходим для создания событий, загрузки и сохранения данных и прочее.

Класс CreatureManager хранит в себе существ (Creature) в структуре CreatureWrapper, которая в данный момент используется для указания направления существа, что в будущем будет необходимо для реализации атаки.

Класс Creature обобщает в себе все возможные существа, которые могут использоваться в программе. В данный момент единственным наследником данного класса является Player. В дальнейшем будут реализованы наследники в виде различных типов врагов.

В классе Field на основании ширины и высоты генерируется поле, состоящее из объектов класса Cell. Для него были реализованы конструкторы копирования и перемещения для возможности реализации загрузки карты из файла. Также он используется для определения перемещения по клеткам при помощи метода getMoving.

В классе ViewController создаются все необходимые объекты для визуализации данных из пространства имен engine, а также для обработки нажатий на кнопки клавиатуры. В нем запускается основной цикл SFML (while (window.open())), в котором используются созданные объекты.

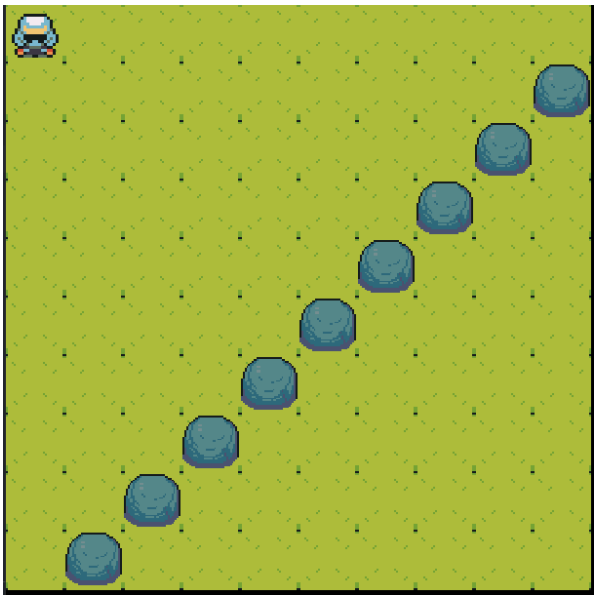
В классе KeyboardHandler происходит обработка нажатий на клавиатуру с дальнейшим переносом enum нажатия в GameMediator для перемещения игрока. Данный класс необходим для дальнейшей реализации загрузки клавиш из конфигурационных файлов.

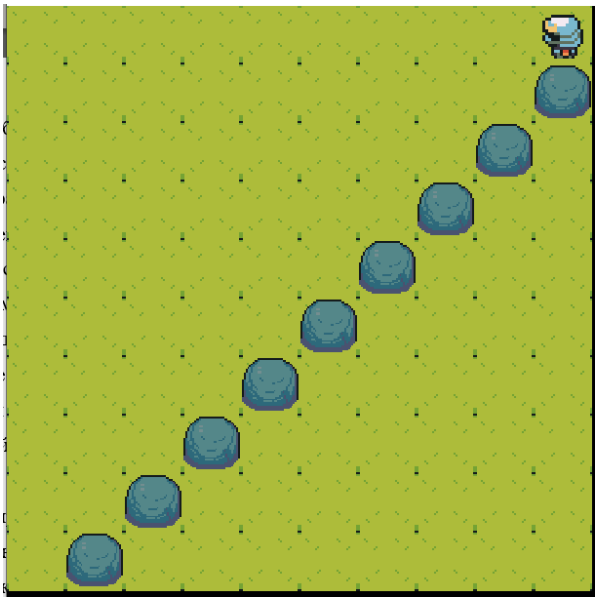
Класс DrawManager отвечает за отрисовку содержимого пространства имен engine. Он использует класс TextureManager для взятия текстур и спрайтов необходимых объектов.

При помощи класса TextureManager программе не приходится повторно загружать одни и те же изображения, так как они сохраняются в std::map для дальнейшего использования.

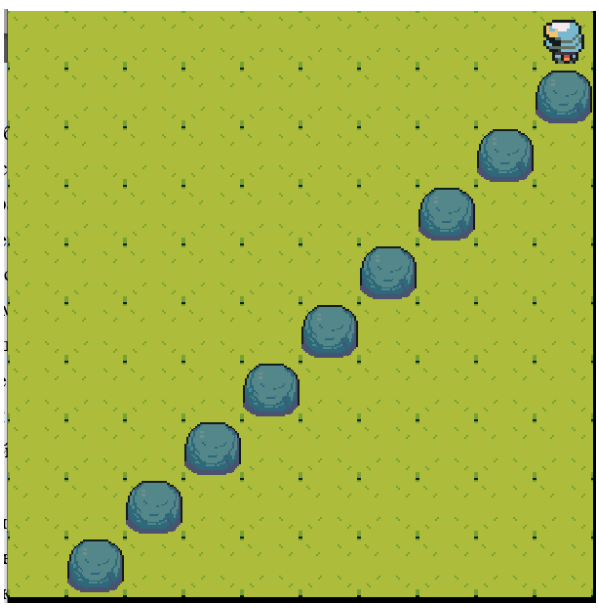
Тестирование программы.

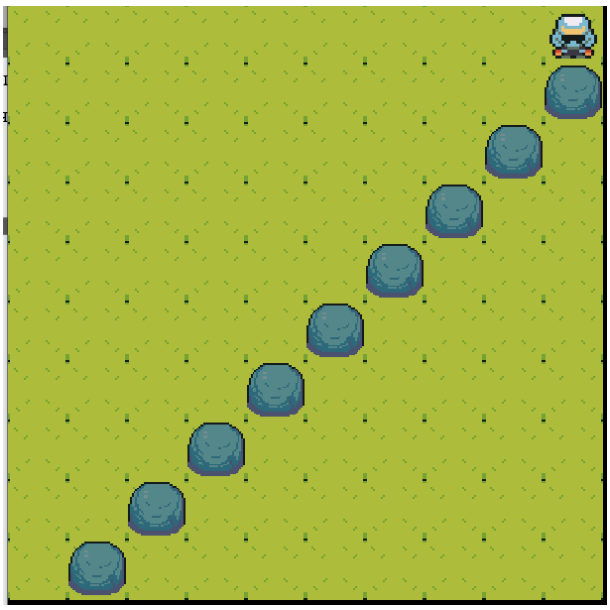
1. Проверка выхода за границу карты



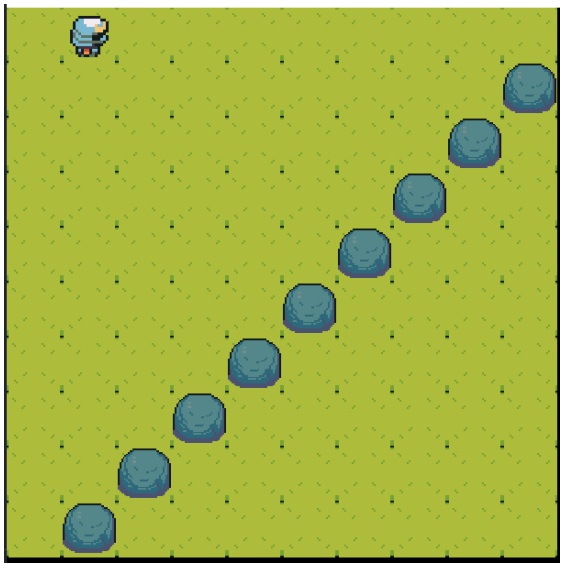


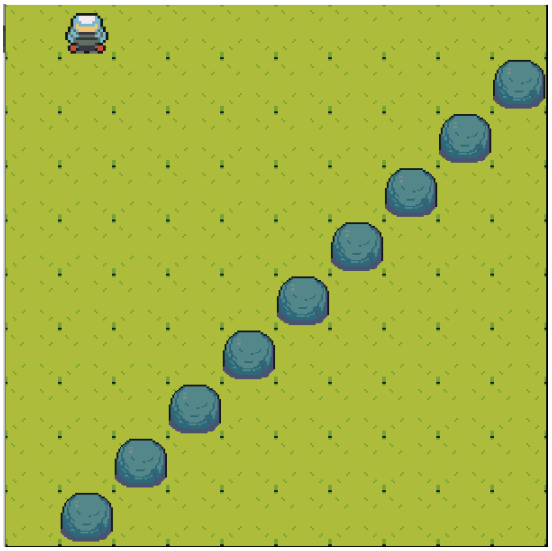
2. Проверка перехода в непроходимую клетку



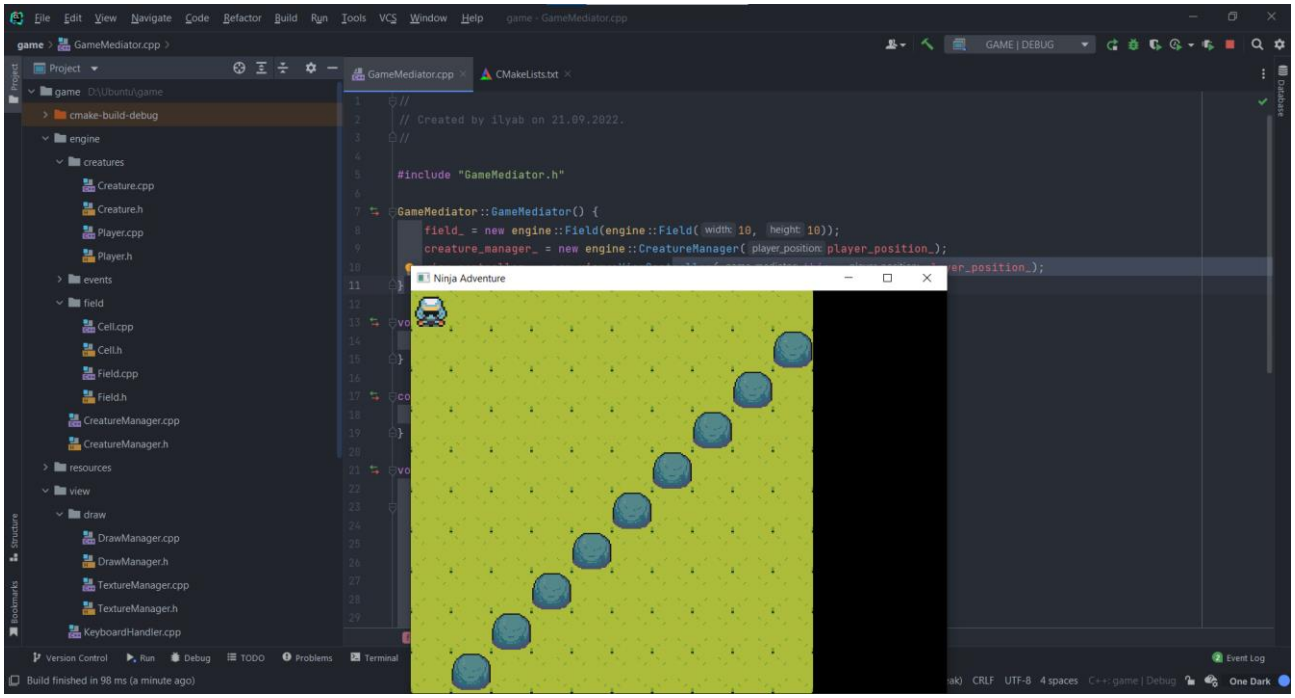


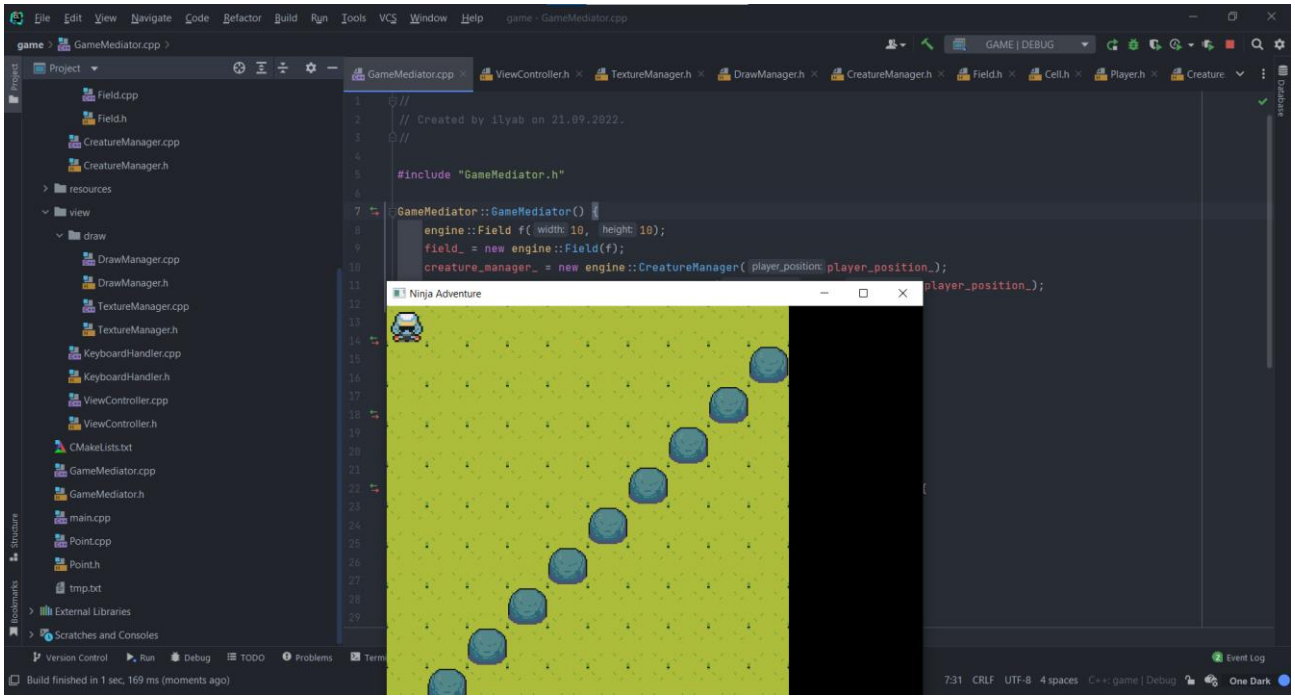
3. Проверка перехода в непроходимую клетку за границей карты





4. Проверка конструкторов копирования и перемещения для Field





Выводы.

В рамках данной лабораторной работы была разработана программа на языке C++ в объектно-ориентированной парадигме, реализующая перемещение сущности – игрока – по клеткам поля. Были изучены механизмы работы классов и их конструкторов.

ПРИЛОЖЕНИЕ 1

UML-диаграмма классов: