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

ОТЧЕТ

по лабораторной работе №8

по дисциплине «Объектно-ориентированное программирование»

Тема: Разработка динамической системы инвентаря игрока

Студент гр. 1304 _____ Павлов Д.Р.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы.

Разработать систему инвентаря игрока и его экипировки (вещей). Класс игрока должен содержать инвентарь, который содержит список экипировки, который у него есть. Инвентарь ограничен суммарным весом вещей. В случае, если игрок пытается получить новую вещь, и веса не хватает, то он не может получить вещь в инвентарь. Разработать систему классов, реализующих систему вещей. Все вещи могут храниться в инвентаре. Вещи делятся на две категории: можно надеть на игрока и можно использовать. Вещи, которые можно надеть, можно снять или заменить на другую. Вещи, которые используются, после использования исчезают.. При использовании вещи, которая используется, игрок тратит ход и никуда не двигается (например, использование зелья лечения), при надевании вещи, ход не должен тратиться. Вещи, которые надеваются, каким-то образом улучшают характеристики игрока пока надеты. Предусмотреть систему получения игроком экипировки (в ходе взаимодействия событий или NPC).

Требования.

Разработан интерфейс экипировки

Реализовано минимум 2 класса экипировки, которую можно надеть (например,

оружие и броня)

Реализовано минимум 2 класса вещей, которые можно использовать

Реализована система инвентаря игрока с возможность надевать/использовать

вещи

Реализована система получения экипировки игроком с учетом ограничений

инвентаря

Описание архитектурных решений и классов.

Описание Классов:

Интерфейсы:

1) Iconsumable — интерфейс класса-расходника. Имеет три абстрактных метода: *void use()* - использовать расходник; *std::string get_icon()* - вернуть иконку расходника; *bool is_empty()* - возвращает protected поле *empty,* которая означает, что расходник что-то делает / не делает (другими словами, пуст/не пуст); *int get_weight()* - возвращает вес айтема.

2) Iequipment — интерфейс класса-экипировки. Имеет три абстрактных метода: *void put_on()* - надеть на игрока; *std::string get_icon()* - вернуть иконку экипировки; *bool is_empty()* - вернуть protected поле *empty* (нужен для отличия каких-либо расходников от класса, говорящий, что расходника нет); *int get_weight()* - возвращает вес айтема.

3) IMask — интерфейс класса-маски. Наследуется от экипировки.

4) IBoot — интерфейс класса-ботинка. Наследуется от экипировки.

Конкретные Расходники:

5) Drug — наследник интерфейса раходника. Представляет грибной расходник. При использовании Герой ловит галлюцинации. Поле *empty = false*. Занимает 5 веса.

6) Heal_Potion — наследник интерфейса расходника. Самое обычное зелье здоровья. При использовании Герой хилится на 1. Поле *empty = false*. Занимает 10 веса.

7) No_Consumable — наследник интерфейса расходника. «Ничего». При использовании ничего не делает. Данный класс служит для заполнения пустых слотов «мешка» с расходниками в инвентаре. Поле *empty = true*. Занимает 0 веса.

Конкретная Экипировка:

8) Ghost_Head — наследник интерфейса маски. При надевании баффает героя на проход сквозь стены. Сам бафф представлен в виде булевого поля в классе Героя. Поле *empty = false*. Занимает 50 веса.

9) Pumpkin_Head — наследник интерфейса маски. При надевании баффает героя на получение дополнительного опыта с клеток с опытом. Сам бафф представлен в виде булевого поля в классе Героя. Поле *empty = false*. Занимает 90 веса.

10) No_Equipmnet — наследник интерфейса маски. При надевании меняет характеристики Героя на дефолтные. Служит для заполнения слотов в инвентаре, если в контексте игры они пустые. Поле *empty = true*. Занимает 0 веса.

11) Slippers - наследник интерфейса ботинка. При надевании герой получает +1 опыт с клеток с опытом. Поле *empty = false*. Занимает 10 веса.

12) Socks - наследник интерфейса ботинка. При надевании герой, убивая врагов, получает +2 опыта. Поле *empty = false*. Занимает 10 веса.

13) No_Boot - наследник интерфейса ботинка. При надевании меняет характеристики Героя на дефолтные. Служит для заполнения слотов в инвентаре, если в контексте игры они пустые. Поле *empty = true*. Занимает 0 веса.

Фабрики:

9) Consumable_Factory — фабрика по созданию конкретных расходников. Имеет отдельный метод для создания каждого конкретного расходника.

10) Equipment_Factory — фабрика по созданию конкретной экипировки. Имеет отдельный метод для создания каждой конкретной экипировки.

11) Inventory — Класс Инвентарь. Хранит векторы расходников и экипировки (векторы указателей на интерфейсы) — это и есть наборы слотов, доступные Герою. Так же есть поля *int consumbler_switcher* и *int equipment_switcher*, которые отвечают за выбор пользователем конкретного расходника и экипировки. Естественно, инвентарь может выкидывать или добавлять предметы из наборов слотов, а так же инвентарь делегирует использование расходника (метод *use()*). К тому же инвентарь проверяет количество использованных слотов, если все слоты расходников/экипировки используются, то далее нельзя взять еще — надо выкинуть либо же использовать.

Демонстрация работы программы и тестирование.

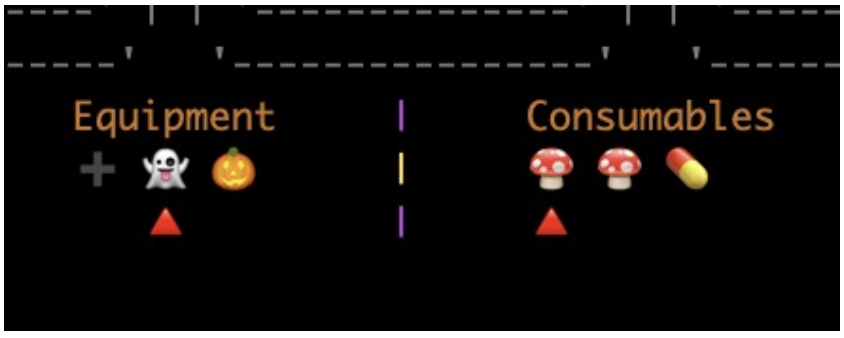

Рисунок 1 — Инвентарь.

На Рисунке 1 представлен инвентарь герой. Слева — Экипировка, а справа — Расходники.

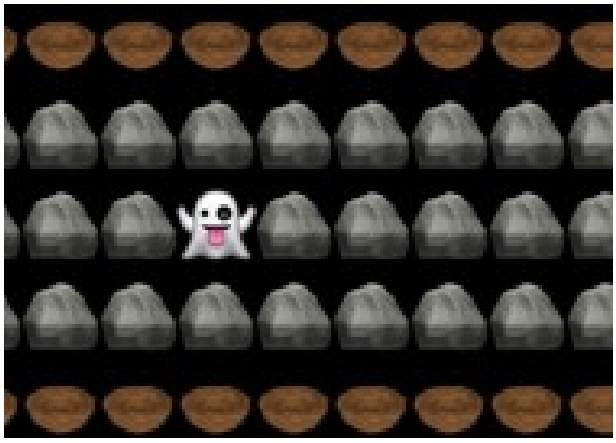

Рисунок 2 — Герой ходит через стены.

На Рисунке 2 видно, как герой, надев экипировку призрака, начал ходить через стены.

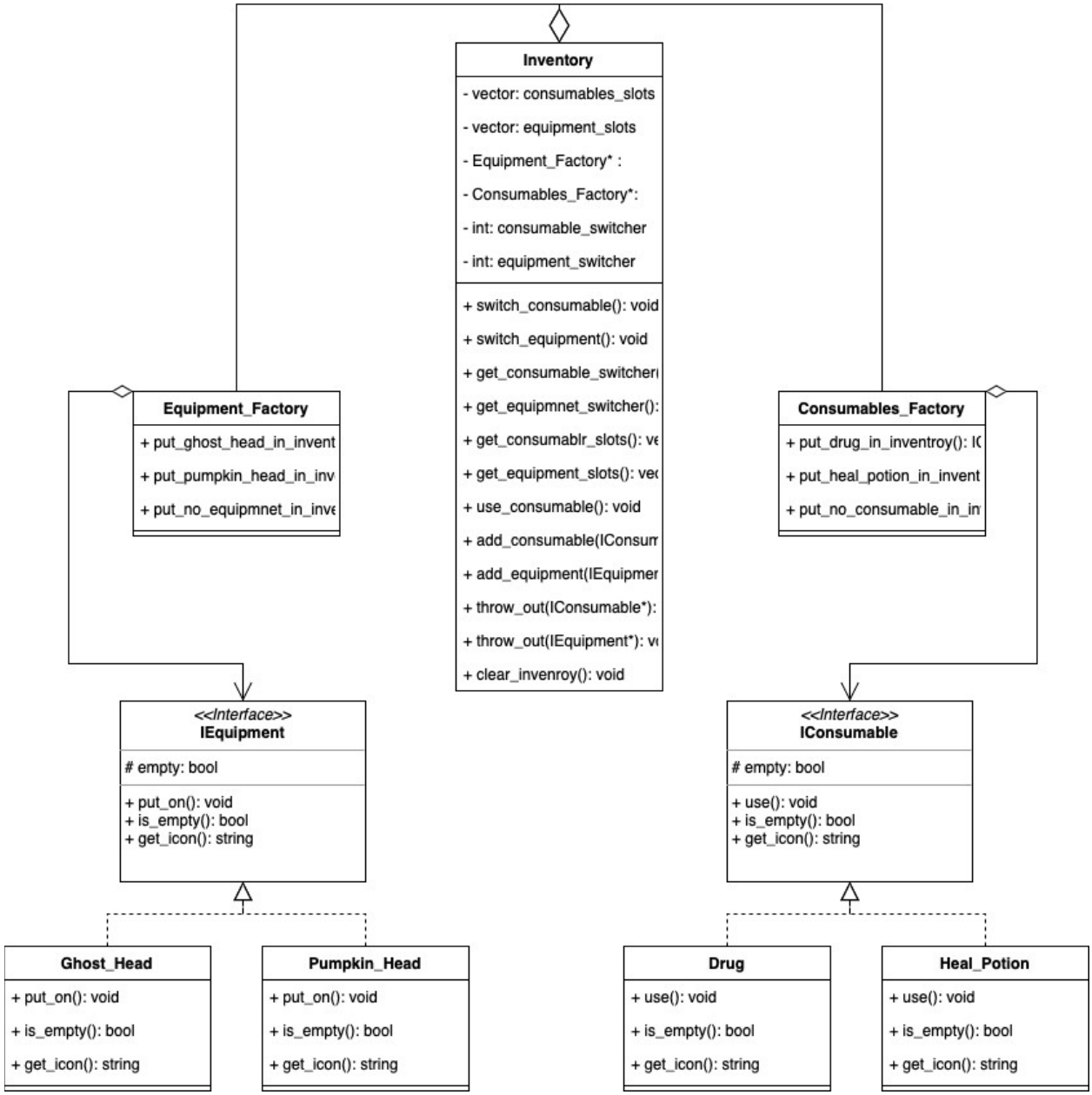

Рисунок 3 — uml диаграмма работы.

Вывод.

Реализована динамическая система инвентаря героя, а так же предметы вида: снаряжение и расходник.

Была изучена работа с классами на языке C++, паттерны проектирования, основы составления UML-диаграмм.