МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ

КАФЕДРА ПРОГРАММНАЯ ИНЖЕНЕРИЯ

ОТЧЁТ О ПРОДЕЛАННОЙ РАБОТЕ

«информатика и основы программирования»

Создание текстовой RPG-игры

Студент

гр. БИН-25-1_____Д.В.Потылицин

Преподаватель_____А.Д.Шнейдер

Владивосток 2025

Содержание

Введение

Целью данного проекта является разработка полностью функциональной текстовой ролевой игры (RPG) на языке программирования Python. Работа направлена на практическое применение фундаментальных принципов объектно-ориентированного программирования (ООП) и демонстрацию навыков создания интерактивного консольного приложения. Игровой процесс построен на классических для жанра механиках, которые включают в себя создание и кастомизацию персонажа, исследование случайно генерируемого мира, пошаговые сражения с разнообразными противниками, сбор и использование предметов, а также систему развития персонажа через накопление опыта и повышение уровня.

Актуальность проекта заключается в его учебно-практической ценности. Разработка игры позволяет комплексно освоить ключевые аспекты Python: работу с классами и объектами, организацию программной логики с помощью условных операторов и циклов, обработку пользовательского ввода, реализацию случайных событий, а также структурирование кода для обеспечения читаемости и расширяемости. Кроме того, проект затрагивает важные для начинающего разработчика темы, такие как балансировка игровых параметров и проектирование взаимодействия между компонентами системы.

В результате планируется получить законченное игровое приложение, которое не только выполняет развлекательную функцию, но и служит наглядным примером реализации типичных задач программирования. Игра предусматривает четко определённую конечную цель — победу над одним из финальных боссов, что придаёт игровому процессу целенаправленность и мотивацию. Данный отчёт подробно описывает концепцию игры, её основные механики, процесс разработки и структуру программного кода.

1 Описание концепции

1.1 Игровой мир и сюжет

Действие игры происходит в фэнтезийном мире, наполненном опасностями и приключениями. Игрок принимает роль безымянного героя, который путешествует по различным локациям, сражается с врагами, собирает предметы и повышает свой уровень. Хотя игра не имеет развернутого сюжета, её геймплей построен вокруг цикла "исследование-бой-прокачка", который мотивирует игрока к дальнейшему прохождению.

Конечная цель – достижение 5-го уровня и победа над одним из финальных боссов: «Старостой Давидом» или «Фенриром». Каждый босс имеет уникальные характеристики, требующей от игрока стратегического подхода и правильного использования накопленных ресурсов. Например, староста Давид имеет всего 5 единиц здоровья, но 99% защиту.

1.2 Основные механики

Игрок начинает игру с создания персонажа, основные характеристики которого — здоровье, урон, броня и уклонение — генерируются случайным образом в заданных пределах. Это позволяет каждому новому прохождению быть уникальным. При неудовлетворённости начальными значениями игрок имеет возможность перераспределить их повторно до начала приключения.

Основной игровой цикл строится вокруг исследования мира, в ходе которого случайным образом происходят различные события. Игрок может встретить врага, вступить с ним в бой, найти сундук с ценными предметами или же безопасную локацию для отдыха, восстанавливающую здоровье. Система боя реализована пошагово: в свой ход игрок выбирает одно из доступных действий — атаковать противника, использовать предмет из инвентаря или попытаться уклониться от следующей вражеской атаки. Важно отметить, что использование предмета является самостоятельным действием и отменяет возможность атаки в текущем ходу, что добавляет стратегической глубины принятию решений.

После успешной победы над врагом персонаж не только получает опыт, но и частично восстанавливает здоровье. Накопление опыта ведёт к повышению уровня игрока, что является ключевым элементом прогрессии. Инвентарь персонажа может содержать разнообразные предметы, такие как оружие, боеприпасы, лечебные средства и специальные артефакты. Их использование предоставляет временные бонусы к характеристикам или накладывает постоянные эффекты, напрямую влияющие на игровой процесс и открывающие дополнительные тактические возможности в бою.

1.3 Баланс

Важнейшим аспектом игрового дизайна в данном проекте является система баланса, обеспечивающая устойчивый и увлекательный игровой процесс. Баланс достигается за счёт продуманного взаимодействия нескольких ключевых механизмов, которые создают динамичное чередование сложности и возможностей для восстановления.

Во-первых, сложность противников прогрессивно возрастает с повышением уровня игрока. Однако это не приводит к непреодолимым трудностям благодаря компенсирующим элементам. Основным из них является система инвентаря, который по своей сути неограничен и может пополняться ценными предметами (такими как лечебные зелья, усиливающие артефакты или наступательные гранаты) в процессе исследования.

Во-вторых, центральным балансирующим элементом выступает уникальная механика выбора событий после каждого боя или в начале игры. Следующая локация определяется абсолютно случайным образом из заранее заданного пула возможностей. Игроку с равной вероятностью может встретиться как новый, потенциально более сильный враг, так и сундук с полезными предметами или безопасная зона для отдыха, полностью восстанавливающая здоровье. Эта непредсказуемость не позволяет игроку заранее подготовиться к следующему шагу и делает каждый игровой сеанс уникальным, предотвращая монотонность.

Такой подход формирует динамический уровень сложности, который колеблется от относительно лёгких периодов (когда после боя следуют несколько сундуков или отдых подряд) до напряжённых испытаний (при последовательных встречах с врагами). Это требует от игрока стратегического планирования и адаптивного использования накопленных ресурсов: применять ли усиливающий предмет в текущем бою или сохранить его для более грозного противника; рисковать ли, вступая в новый бой с неполным здоровьем, в надежде получить ресурсы для его пополнения. Таким образом, баланс строится не на жёстком математическом равенстве сил, а на создании рискованных ситуаций выбора и управляемой случайности, что поддерживает интерес и вовлечённость на протяжении всей игры.

2 Разработка

2.1 Система классов

Игра изначально разрабатывалась с объектно-ориентированным программированием (классы). Это сделано для того, чтобы было проще обращаться к переменным в разных частях кода. В игре всего 2 класса: это *Player* (Игрок) и *Enemy* (Враг). Класс *Player* используется как для хранения данных об игроке, так и для основы игры. Класс *Enemy* используется для хранения информации о враге.

2.2 Класс *Player*

Класс Player включает следующие функции:

- *__init__* - инициализация игрока с заданием характеристик и начальных предметов;
- *show_status* – вывод текущих параметров персонажа; *walk* – перемещение по случайным локациям с обработкой событий (сундук, враг, отдых);
- battle – управление пошаговым боем с выбором действия;
- *attack* – атака врага с учётом брони и эффектов;
- *use_item* – использование предметов из инвентаря с применением соответствующих эффектов;
- avoid – попытка уклонения от атаки врага;
- *get_damage* – расчёт итогового урона с учётом всех модификаторов;
- *death* – обработка поражения игрока;
- *victory* – завершение игры при победе над финальным боссом.

2.3 Класс *Enemy*

Класс Enemy представляет игрового противника и отвечает за хранение его характеристик. При создании объекта класса задаётся имя врага, на основе которого определяются его основные параметры: здоровье, урон, броня, опыт за победу и особые эффекты.

2.4 Внеклассовая часть кода

Помимо классов в игре содержатся следующие части кода:

- импорт библиотеки для определения случайных чисел или выбора;
- константы с названиями локаций, предметов и врагов на разных уровнях;
- функция *bar*, помогающая сделать красиво оформленную шкалу (для опыта, здоровье и т.д.);
- запуск самой игры путём вызова основного класса *Player.*

Импорт библиотеки осуществляется в самом начале файла и включает в себя функции *randint* и *choice* из модуля *random*. Эти функции обеспечивают генерацию случайных характеристик персонажа и случайный выбор событий, локаций и врагов, что является основой динамичного и непредсказуемого геймплея.

Константы с названиями локаций, предметов и врагов представляют собой кортежи, которые хранят все возможные варианты игровых объектов. Например, константа *LOCATIONS* содержит названия областей игрового мира, а *ENEMIES_1_LVL* – названия врагов первого уровня. Такая организация данных позволяет легко управлять содержимым игры и балансировать сложность.

Функция *bar* принимает числовые параметры, такие как текущее и максимальное значение здоровья или опыта, а также длину шкалы. Она формирует строковое представление шкалы, где заполненная часть обозначается символами «=» (равно), а незаполненная — символами «-» (тире). Этот визуальный элемент делает интерфейс боя более информативным и наглядным.

Запуск игры осуществляется в самом конце кода с помощью создания экземпляра класса *Player*. Это действие инициирует выполнение метода *init*, который запрашивает имя игрока, генерирует начальные характеристики и запускает основной цикл игры через метод *walk*. Таким образом, вся игровая механика приводится в движение единой командой.

Заключение

В ходе выполнения данного учебного проекта была успешно разработана и реализована текстовая ролевая игра (RPG) на языке Python, что подтверждает достижение поставленной цели. Основным результатом работы стало создание полностью функционального консольного приложения, которое включает в себя ключевые элементы классического игрового жанра: систему создания и кастомизации персонажа, процедурную генерацию событий, пошаговую боевую систему, инвентарь с разнообразными предметами и механику прокачки через получение опыта.

Проект демонстрирует практическое применение принципов объектно-ориентированного программирования. Весь игровой процесс инкапсулирован в двух основных классах – *Player* и *Enemy*, что обеспечивает чёткое разделение ответственности, структурированность кода и его относительную лёгкость в поддержке и расширении. Реализованная система баланса, основанная на случайном выборе событий и неограниченном инвентаре, создаёт динамичный и увлекательный игровой цикл, требующий от пользователя стратегического планирования и адаптивного использования ресурсов.

Работа имеет значительную учебно-практическую ценность. В процессе разработки были закреплены навыки работы с основными конструкциями Python, включая классы, условные операторы, циклы, работу со списками и кортежами, обработку пользовательского ввода, а также организацию логики программы. Кроме того, был получен опыт в области базового игрового дизайна, в частности, в вопросах балансировки игровых параметров и создания вовлекающего пользовательского опыта.