*МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВВГУ»)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ОТЧЕТ
ОТЧЕТ ПО ФИНАЛЬНОЙ РАБОТЕ
ТЕКСТОВАЯ RPG-ИГРА

по дисциплине
«Информатика и основы программирования»

Студент
гр. БИС-25-3 _____ В.В. Челпан

Ассистент
преподавателя _____ М.В. Водяницкий

Владивосток 2026

Задание

Разработать программный прототип консольной текстовой RPG-игры.

Вы работаете программистом в небольшой игровой компании. Вам поручено создать текстовую RPG-игру, работающую в консоли и демонстрирующую основные игровые механики: создание персонажа, бои, прокачку, инвентарь и случайные события.

1 Общая идея программы

Программа представляет собой консольную текстовую RPG-игру, в которой игрок:

- создаёт персонажа с выбором расы;
- получает случайные характеристики;
- исследует подземелье, состоящее из комнат;
- сражается с врагами;
- собирает предметы и золото;
- повышает уровень и улучшает характеристики.

Игра работает в пошаговом режиме и управляется вводом команд с клавиатуры.

2 Персонаж

В начале игры пользователь выбирает расу персонажа:

- Человек;
- Эльф;
- Дворф.

Каждая раса задаёт диапазоны генерации характеристик.

Характеристики персонажа:

- здоровье (HP);
- сила атаки;
- защита;
- ловкость;
- рост;
- вес.

Рост и вес дополнительно влияют на ловкость персонажа.

3 Прокачка и опыт

Персонаж получает опыт за победу над врагами. При достижении необходимого количества опыта повышается уровень.

Каждый новый уровень даёт очки прокачки, которые можно распределять между характеристиками персонажа в комнатах отдыха.

4 Подземелье и комнаты

Подземелье состоит из случайно генерируемых комнат. После каждой комнаты игрок выбирает направление движения.

Типы комнат:

- боевая комната;
- комната отдыха;
- комната с сундуком.

Информация о содержимом комнаты может быть как известна, так и скрыта.

5 Боевая система

Бой происходит в пошаговом режиме. Игрок может:

- атаковать;
- использовать предметы;
- попытаться уклониться от атаки.

В бою учитываются характеристики персонажа, экипировка и случайные факторы.

6 Инвентарь и предметы

Игрок имеет инвентарь, в котором хранятся:

- зелья;
- оружие;
- броня;
- золото.

Предметы можно использовать, выбрасывать или экипировать.

7 Ограничения и требования

- программа является консольной;
- управление осуществляется через текстовое меню;
- язык программирования – Python;
- код должен быть структурирован и читаем.

Содержание

Введение

Современные программные системы всё чаще ориентированы на интерактивное взаимодействие с пользователем и моделирование сложных процессов в упрощённой форме. Одним из наглядных примеров таких систем являются компьютерные игры, в частности текстовые RPG-игры, которые позволяют реализовать широкий спектр алгоритмических и объектно-ориентированных решений в рамках консольных приложений.

Текстовые RPG-игры представляют собой удобную среду для изучения основ программирования, так как включают в себя работу с пользовательским вводом, условными операторами, циклами, структурами данных, функциями и классами. Кроме того, разра- ботка подобной игры позволяет ознакомиться с принципами проектирования программных систем, такими как модульность, инкапсуляция и повторное использование кода.

Целью данной работы является разработка консольной текстовой RPG-игры, моделирующей процесс исследования подземелья, пошаговых боёв, развития персонажа и управления инвентарём. В рамках работы была поставлена задача реализовать про- граммный прототип, демонстрирующий основные игровые механики и обеспечивающий устойчивую работу при различных сценариях взаимодействия пользователя с системой.

В процессе выполнения работы были реализованы механизмы создания персонажа с выбором расы и генерацией характеристик, пошаговая боевая система, система опыта и уровней, генерация случайных комнат подземелья, а также управление инвентарём и экипировкой персонажа.

Программа разработана на языке Python и предназначена для работы в консольном режиме. Выбор данного языка программирования обусловлен его простотой, наглядностью синтаксиса и широкими возможностями для разработки прототипов программных систем.

1 Разбор функциональности программы

Разработанная программа представляет собой консольную текстовую RPG-игру, реализованную на языке программирования Python. Программа построена по модульному принципу и использует классы и функции для разделения логики игрового процесса на независимые функциональные блоки.

Архитектура программы ориентирована на поэтапное прохождение игры и последовательную обработку действий пользователя, что упрощает расширение и сопровождение кода.

1.1 Создание персонажа

Функция `create_character()` реализует начальный этап игры и отвечает за создание игрового персонажа пользователя. На рисунке 1 показана эта функция.

```
119  def create_character():
120      character = Character()
121
122      print("=== СОЗДАНИЕ ПЕРСОНАЖА ===")
123      print("Выберите расу:")
124      print("1. Человек (сбалансированный)")
125      print("2. Эльф (ловкий, но хрупкий)")
126      print("3. Дворф (крепкий, но медленный)")
127
128      while character.race is None:
129          try:
130              choice = int(input("Ваш выбор: "))
131              if choice == 1:
132                  character.race = "Человек"
133                  # Диапазоны для человека
134                  character.height = random.randint(165, 185)
135                  character.weight = random.randint(65, 85)
136                  character.max_hp = random.randint(80, 100)
137                  character.hp = character.max_hp
138                  character.attack = random.randint(8, 12)
139                  character.defense = random.randint(5, 8)
140                  character.agility = random.randint(10, 14)
141
142              elif choice == 2:
143                  character.race = "Эльф"
144                  # Диапазоны для эльфа
145                  character.height = random.randint(175, 195)
146                  character.weight = random.randint(55, 75)
147                  character.max_hp = random.randint(70, 90)
148                  character.hp = character.max_hp
149                  character.attack = random.randint(9, 13)
150                  character.defense = random.randint(3, 6)
151                  character.agility = random.randint(14, 18)
152
153              elif choice == 3:
154                  character.race = "Дворф"
```

Рисунок 1 - Функция create_character()

В начале выполнения функции пользователю выводится меню выбора расы персонажа. Выбор осуществляется из трёх вариантов:

- человек – сбалансированный персонаж;

- эльф – персонаж с повышенной ловкостью;

- дворф – персонаж с повышенной выносливостью и защитой.

Выбор пользователя обрабатывается с проверкой корректности ввода, что исключает возможность возникновения ошибок выполнения программы.

После выбора расы выполняется генерация стартовых характеристик персонажа. Характеристики формируются случайным образом в пределах диапазонов, заданных для каждой расы, что обеспечивает разнообразие игровых сценариев.

Генерируются следующие параметры:

- максимальное и текущее здоровье;
- сила атаки;
- защита;
- ловкость;
- рост;
- вес.

Дополнительно в функции реализована логика влияния физических параметров персонажа на его игровые характеристики. Рассчитывается индекс массы тела, который может увеличивать или уменьшать ловкость персонажа. Данный механизм добавляет элемент реализма и вариативности в игровой процесс.

По завершении создания персонажа пользователю выводится подробная информация о всех сгенерированных характеристиках.

1.2 Класс персонажа

Класс `Character` предназначен для хранения и управления состоянием игрового персонажа. Данный класс представлен на рисунке 2.

```
3   # Класс персонажа
4   class Character:
5       def __init__(self):
6           self.race = None
7           self.level = 1
8           self.exp = 0
9           self.exp_to_next = 100
10          self.skill_points = 0
11          self.hp = 0
12          self.max_hp = 0
13          self.attack = 0
14          self.defense = 0
15          self.agility = 0
16          self.height = 0
17          self.weight = 0
18          self.inventory = []
19          self.equipped = {"weapon": None, "armor": None}
20          self.coins = 0
```

Рисунок 2 – Класс `Character`

В данном классе содержатся следующие группы данных и реализованы методы:

- основные характеристики персонажа;
- информация об уровне и опыте;
- количество очков прокачки;
- инвентарь и экипировка;
- количество внутриигровых монет.
- получения урона с учётом защиты;
- восстановления здоровья;
- получения опыта;
- повышения уровня;
- распределения очков прокачки;
- отображения текущих характеристик.

Инкапсуляция логики персонажа в отдельный класс позволяет упростить управление состоянием игрока и повысить читаемость программного кода.

1.3 Генерация подземелья и комнат

Функция generate_room() отвечает за генерацию игровых комнат подземелья. Тип комнаты определяется случайным образом, что позволяет создать непредсказуе-

мый игровой процесс. Для генерации используются взвешенные вероятности, в результате боевые комнаты встречаются чаще. Данная функция представлена на рисунке 3

```
180  def generate_room(room_type=None):
181      if room_type is None:
182          room_type = random.choice(["combat", "rest", "treasure", "combat", "rest"])
183
184      rooms = {
185          "combat": "Боевая комната",
186          "rest": "Комната отдыха",
187          "treasure": "Комната с сундуком"
188      }
189
190      return {
191          "type": room_type,
192          "name": rooms[room_type],
193          "visited": False
194      }
```

Рисунок 3 – Функция generate_room()

Каждая комната описывается структурой данных, содержащей:

- тип комнаты;
- название комнаты;
- признак посещения.

Перед выбором направления движения игроку может быть показана либо скрыта информация о типе следующей комнаты, что добавляет элемент случайности и принятия решений.

1.4 Боевая система

Боевая система реализована в функции `combat_room()` и является одной из ключевых частей игрового процесса.

При входе в боевую комнату создаётся противник класса `Enemy`. Характеристики врага зависят от текущего уровня персонажа, что обеспечивает постепенное увеличение сложности игры. Код функции `combat_room()` представлен на рисунке 4.

```
196  def combat_room(player):
197      print("\n=== БОЕВАЯ КОМНАТА ===")
198      enemy = Enemy(player.level)
199      print(f"На вас напал {enemy.name}!")
200
201      while enemy.hp > 0 and player.hp > 0:
202          print("\n" + "="*30)
203          enemy.show_stats()
204          print(f"Ваше HP: {player.hp}/{player.max_hp}")
205          print("\nДоступные действия:")
206          print("1. Атаковать")
207          print("2. Использовать зелье")
208          print("3. Попытаться уклониться")
209
210          try:
211              choice = int(input("Выберите действие: "))
212
213              if choice == 1:
214                  # Атака игрока
215                  player_damage = max(1, player.attack + random.randint(-2, 3))
216                  enemy.hp -= player_damage
217                  print(f"Вы нанесли {player_damage} урона!")
218
219                  # Контратака врага, если жив
220                  if enemy.hp > 0:
221                      if random.random() < player.agility/100:  # Шанс уклонения
222                          print("Вы уклонились от атаки врага!")
223                      else:
224                          enemy_damage = max(1, enemy.attack - player.defense)
225                          player.hp -= enemy_damage
226                          print(f"{enemy.name} нанес вам {enemy_damage} урона!")
227
228              elif choice == 2:
```

Рисунок 4 – код функции `combat_room()`

В процессе боя игроку доступны следующие действия:

- стандартная атака;
- использование предметов лечения;
- попытка уклонения от атаки противника.

После победы над врагом игрок получает опыт, монеты, а также имеет вероятность получить случайный предмет. В случае поражения игровой процесс завершается.

1.5 Комнаты с наградами

Функция `treasure_room()` реализует механику получения наград. Эта функция представлена на рисунке 5.

```
275  def treasure_room(player):
276      print("\n=== КОМНАТА С СУНДУКОМ ===")
277      print("Вы нашли сундук!")
278
279      treasure_type = random.choice(["coins", "item", "both"])
280
281      if treasure_type in ["coins", "both"]:
282          coins = random.randint(20, 50) + player.level * 10
283          player.coins += coins
284          print(f"Найдено монет: {coins}")
285
286      if treasure_type in ["item", "both"]:
287          items = ["Зелье здоровья", "Зелье силы", "Стальной меч",
288                   "Кожаный доспех", "Эльфийский лук", "Доспех гномов"]
289          item = random.choice(items)
290          player.inventory.append(item)
291          print(f"Найдено: {item}")
292
293      print(f"Ваши монеты: {player.coins}")
```

Рисунок 5 - Функция `treasure_room()`

В комнате с сундуком случайным образом определяется тип награды:

- только монеты;
- только предмет;
- комбинация монет и предметов.

Размер награды масштабируется в зависимости от уровня персонажа.

1.6 Комнаты отдыха

Функция `rest_room()` реализует безопасные зоны подземелья.

В данных комнатах персонаж восстанавливает часть здоровья и может распределить накопленные очки прокачки, улучшая выбранные характеристики.

Данная функция представлена на рисунке 6

```
295  def rest_room(player):
296      print("\n=== КОМНАТА ОТДЫХА ===")
297      print("Здесь безопасно. Вы можете отдохнуть.")
298
299      rest_amount = player.max_hp * 0.3
300      player.heal(int(rest_amount))
301      print(f"Вы восстановили {int(rest_amount)} HP")
302
303      if player.skill_points > 0:
304          use = input("Хотите использовать очки прокачки? (да/нет): ").lower()
305          if use == "да":
306              player.use_skill_points()
307
```

Рисунок 6 – Функция `rest_room()`

1.7 Инвентарь и экипировка

Функция `manage_inventory()` реализует систему управления инвентарём и использования в нем предметов. Код данной функции представлен на рисунке 7.

```
308 def manage_inventory(player):
309     while True:
310         print("\n=== ИНВЕНТАРЬ ===")
311         print(f"Монеты: {player.coins}")
312         print("Предметы:")
313 
314         if not player.inventory:
315             print("Инвентарь пуст")
316         else:
317             for i, item in enumerate(player.inventory, 1):
318                 print(f"{i}. {item}")
319 
320         print("\nЭкипировано:")
321         print(f"Оружие: {player.equipped['weapon'] or 'Нет'}")
322         print(f"Броня: {player.equipped['armor'] or 'Нет'}")
323 
324         print("\n1. Использовать предмет")
325         print("2. Выбросить предмет")
326         print("3. Экипировать предмет")
327         print("4. Выйти")
328 
329         try:
330             choice = int(input("Выберите действие: "))
331 
332             if choice == 1:
333                 if not player.inventory:
334                     print("Инвентарь пуст!")
335                     continue
```

Рисунок 7 – код функции `manage_inventory()`

В инвентаре игрок может:

- просматривать список предметов;
- использовать зелья;
- выбрасывать предметы;
- экипировать оружие и броню.

Экипированные предметы изменяют характеристики персонажа, что напрямую влияет на исход боёв.

1.8 Система опыта и повышения уровня

Система развития персонажа реализована функциями `add_exp()` и `level_up()`.Одна из функций представлена на рисунке 8.

```
48      def add_exp(self, amount):
49          self.exp += amount
50          print(f"Получено опыта: {amount}")
51          while self.exp >= self.exp_to_next:
52              self.level_up()
```

Рисунок 8 – функция `add_exp()`

При повышении уровня:

- увеличивается уровень персонажа;
- возрастает максимальное здоровье;
- выдаются очки прокачки;
- полностью восстанавливается здоровье.

1.9 Основной игровой цикл

Функция `main_game()` управляет всей логикой игры. Код данной функции представлен на рисунке 9.

```
394 def main_game():
395     print("=== ТЕКСТОВАЯ RPG ===")
396     print("Добро пожаловать в подземелье!\n")
397 
398     player = create_character()
399     current_floor = 1
400     rooms_cleared = 0
401 
402     # Начальные предметы
403     player.inventory.append("Малое зелье здоровья")
404     player.inventory.append("Малое зелье здоровья")
405 
406     while player.hp > 0:
407         print(f"\n=== ЭТАЖ {current_floor} ===")
408         print(f"Комнат пройдено: {rooms_cleared}")
409         player.show_stats()
410 
411         # Генерация комнат на развилке
412         left_room = generate_room()
413         right_room = generate_room()
```

Рисунок 9 – Код функции `main_game()`

В ней реализованы:

- инициализация игры;
- генерация развилок подземелья;
- обработка выбора игрока;
- переход между этажами;
- завершение игрового процесса.

Функция объединяет все модули программы в единый игровой процесс.

2 Тестирование программы

Тестирование программы проводилось в интерактивном режиме путём последовательного прохождения игровых сценариев и анализа реакции системы на действия пользователя.

Основной целью тестирования являлась проверка корректности работы всех реализованных функций и устойчивости программы к некорректному пользовательскому вводу.

В ходе тестирования были проверены следующие сценарии:

- запуск программы и корректность отображения стартового меню;
- создание персонажей всех доступных рас;
- генерация характеристик в допустимых диапазонах;
- влияние роста и веса на параметр ловкости;
- корректность пошаговой боевой системы;
- использование предметов в бою;
- обработка победы и поражения;
- начисление опыта и повышение уровня;
- распределение очков прокачки;
- генерация комнат различных типов;
- работа комнат отдыха и восстановление здоровья;
- получение наград в комнатах с сундуками;
- управление инвентарём и экипировкой;
- переход между этажами подземелья;
- корректное завершение игры.

Отдельное внимание уделялось обработке некорректного ввода пользователя. При вводе недопустимых значений программа корректно выводит сообщения об ошибках и не допускает аварийного завершения.

По результатам тестирования установлено, что программа корректно реализует заявленный функционал, устойчиво работает при различных сценариях использования и соответствует требованиям технического задания.

Заключение

В ходе выполнения данной работы был разработан программный проект, представляющий собой консольную текстовую RPG-игру, моделирующую процесс прохождения подземелья, включающий элементы боевой системы, развития персонажа и управления ресурсами.

В результате выполнения поставленных задач были реализованы основные игровые механики, такие как создание персонажа с выбором расы, генерация характеристик, пошаговые бои с противниками, система опыта и повышения уровня, работа с инвентарём и экипировкой, а также генерация случайных игровых событий.

Разработанная программа обладает логичной структурой, основанной на использовании классов и функций, что упрощает понимание кода и его дальнейшее сопровождение. Особое внимание было уделено обработке пользовательского ввода и предотвращению некорректных действий, что повышает устойчивость программы в процессе эксплуатации. Проведённое тестирование показало, что программа корректно функционирует при различных сценариях игры, устойчиво обрабатывает ошибки ввода и соответствует требованиям технического задания.

Разработанный проект может быть использован в качестве учебного примера для изучения основ программирования, объектно-ориентированного подхода и разработки консольных приложений. В дальнейшем программа может быть расширена за счёт добавления системы сохранений, новых типов врагов, предметов и игровых механик.