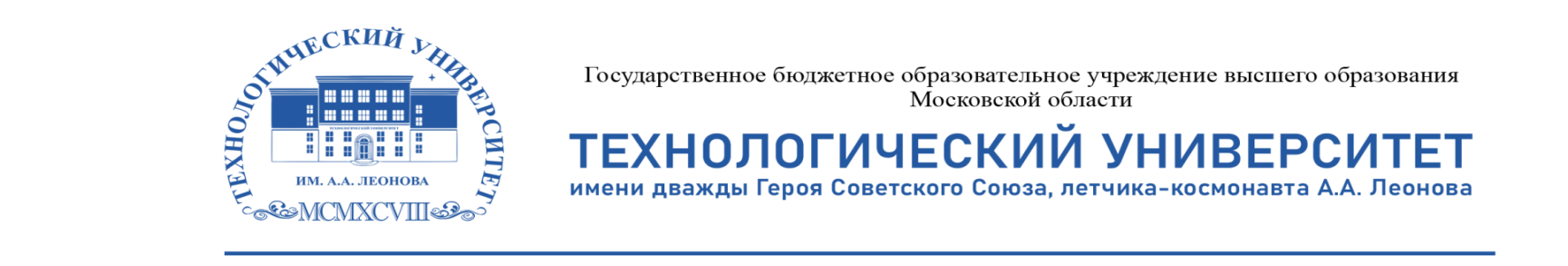
****

Колледж космического машиностроения и технологии

**Курсовой проект**

**Разработка модуля интерфейса для приложения «Ролевая игра «Подземелье Короля»»**

Пояснительная записка

КП.09.02.03.22.15ПЗ

Обучающийся группы П2-19 Пицхелаури В.Г

Руководитель курсового проекта Гусятинер Л.Б.

Результат защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Королев, 2022 г.

Оглавление

[Введение 3](#_Toc106793815)

[1 Теоретическая часть 4](#_Toc106793816)

[1.1 Описание предметной области 4](#_Toc106793817)

[1.2 Описание существующих разработок 7](#_Toc106793818)

[1.2.1 Choice of Robots[4] 7](#_Toc106793819)

[1.2.2 The Life and Suffering of Sir Brante[5] 8](#_Toc106793820)

[2. Проектная часть 10](#_Toc106793821)

[2.1 Построение диаграммы прецедентов 10](#_Toc106793822)

[2.2 Выбор инструментов 11](#_Toc106793823)

[2.3 Проектирование сценария 12](#_Toc106793824)

[2.4 Диаграмма классов 13](#_Toc106793825)

[2.5 Описание главного модуля 15](#_Toc106793826)

[2.6 Описание спецификаций к модулям 21](#_Toc106793827)

[2.7 Описание модулей 23](#_Toc106793828)

[2.8 Описание тестовых наборов модулей 25](#_Toc106793829)

[2.9 Описание применения средств отладки 27](#_Toc106793830)

[3. Эксплуатационная часть 30](#_Toc106793831)

[3.1 Руководство оператора 30](#_Toc106793832)

[3.2 To-Do лист 33](#_Toc106793833)

[Заключение 34](#_Toc106793834)

[Список использованной литературы и Интернет-ресурсов 35](#_Toc106793835)

[Приложение 1. Интерфейс tracker 36](#_Toc106793836)

[Приложение 2. Боевая система BattleFunction 42](#_Toc106793837)

[Приложение 3. Система торговли Traderf 46](#_Toc106793838)

# **Введение**

Данный курсовой проект заключается в разработке текстовой ролевой игры со спрайтами. Разработка игры будет на Python, PyQt. Данный курсовой проект был выбран потому что, он позволит получить опыт в области разработки игр.

В первой части будут рассмотрены предметная область и существующие продукты по данной теме.

Во второй части будут описаны инструменты разработки программы и модули программы.

В третьей части описаны взаимодействие пользователя с программой и сообщения оператору.

В заключительной части будут приведены общие выводы по проекту.

# **1. Теоретическая часть**

## **Описание предметной области**

**Текстовая ролевая игра** - это ролевая игра, в которую играют на компьютере, используя исключительно текстовый интерфейс.

**Функции игры**

Текстовая ролевая игры выполняет несколько функций, развлечь пользователя, подарить ему интересный опыт и позволить игроку интересно провести время.

Развлекательная функция – это основная функция игры – развлечь. Играя в нашу игру, игрок должен получать удовольствие от интересного приключения и новых ощущений.

Интересный опыт, жанр нашей игры «текстовая ролевая», игрок примеряет на себя образ персонажа, цель которого спасти своё королевство. В реальной жизни подобный опыт получить очень сложно, дорого и опасно, поэтому наша игра хорошая альтернатива.

Время препровождение, часть времени в которой человек отдыхает и получает удовольствие. Наша игра создана для этого. Игрок проведёт некоторое время в нашей игре и сможет отдохнуть.

**Компьютерная ролевая игра** – жанр компьютерных игр, основанный на элементах игрового процесса традиционных настольных ролевых игр. В ролевой игре игрок управляет одним или несколькими персонажами, каждый из которых описан набором численных характеристик, списком способностей и умений; примерами таких характеристик могут быть очки здоровья (англ. *hit points, HP*), показатели силы, ловкости, интеллекта, защиты, уклонения, уровень развития того или иного навыка и т. п.

**Виды компьютерных ролевых игр:**

* **RPG-повествование** (англ. *Narrative RPG* или [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *story-driven RPG*) построено вокруг некоего повествования; ожидается, что игрок будет увлечен сюжетом игры, персонажами и декорациями и будет следовать основной сюжетной линии ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *main quest*), вокруг которой строится весь остальной игровой процесс. В качестве примеров таких игр называются [Mass Effect](https://ru.wikipedia.org/wiki/Mass_Effect), [Dragon Age: Origins](https://ru.wikipedia.org/wiki/Dragon_Age:_Origins).
* **RPG-песочница** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *sandbox RPG*) помещает игрового персонажа в [открытый мир](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D0%BC%D0%B8%D1%80_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B5_%D0%B8%D0%B3%D1%80%D1%8B)), предоставляя игроку делать то, что ему заблагорассудится. В таких играх вместо основной сюжетной линии игроку предлагается множество независимых заданий, мест для посещения и тому подобного. В качестве примеров таких игр называются серия [The Elder Scrolls](https://ru.wikipedia.org/wiki/The_Elder_Scrolls) и [Fallout](https://ru.wikipedia.org/wiki/Fallout).
* **Зачистка подземелий** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Dungeon crawler*) ставит во главу угла развитие персонажа, превращая повышение характеристик и приобретение новой, более ценной экипировки в самоцель. Сюжеты и декорации таких игр очень просты и служат не более чем предлогом для истребления множества противников и поиска сокровищ. Примерами таких игр могут служить [серия Diablo](https://ru.wikipedia.org/wiki/Diablo_(%D1%81%D0%B5%D1%80%D0%B8%D1%8F_%D0%B8%D0%B3%D1%80)), [Torchlight](https://ru.wikipedia.org/wiki/Torchlight), [Dungeon Siege](https://ru.wikipedia.org/wiki/Dungeon_Siege).

Так же ролевые игры разделяются на две «школы»:

- “Западная школа”

- “Японская школа”

**Преимущества и недостатки**

По сравнению с другими ролевыми играми, текстовые ролевые игры свои преимущества и недостатки.

Преимущества:

**Простой внешний вид**. В отличие от других игр в данном жанре текстовые ролевые игры отличаются внешней простотой. Всё что нужно игроку, это текст на экране, сделать интерфейс под такую игру крайне легко, благодаря этому мы можем углубиться в другие составляющие игры, такие как игровая механика и сюжет.

**Проработка игровых механик**. Текстовые ролевые игры зачастую куда более проработаны по сравнению с аналогами (если смотреть среди малых «индии» разработчиков).

Недостатки:

**Малое количество визуальных образов.** Т.к. жанр нашей игры «текстовый» вся информация подаётся через текст, из-за этого в игре мало картинок.

**Монотонность игрового процесса.** В нашей игре нет красивых заставок и боевой системы, озвучки, только текст и картинки, поэтому большей части современных игроков попросту будет не интересно в это играть.

**Игры в образовании.**

Как известно, учеба далеко не всем дается легко, кому-то не интересно, а кто-то не может усидеть на месте. Игры давно являются не только интерактивным развлечением, но и полноценным видом искусства, но также их можно использовать и во обучении.

Некоторые игры в интересной форме рассказывают истории не хуже литературы, а головоломки развивают мышление и логическое мышление.

“Но если говорить об играх и образовании одновременно, то нельзя не упомянуть великий и могучий Minecraft Education Edition или Образовательное Издание!

На самом деле, в ретроспективе создание такой штуки кажется чем-то до ужаса очевидным. Всмысле, Minecraft — это игра, в которой можно построить буквально что угодно. Начиная с любого здания в реальную величину и заканчивая моделью работающего компьютера! Мягко говоря, такая вещь обладает образовательным потенциалом уже сама по себе! А если к ней прикрутить ещё и несколько интересных обновлений, или совместимость со всяким софтом... то получится Education Edition, да.”[6]

## **Описание существующих разработок**

В этом разделе рассмотрены некоторые уже имеющиеся игры, а также цели таких программ и требования.

Интерфейс таких игр чаще всего простой и понятный для обычных пользователей.

Цели использования таких игр:

1. Отдых игрока от жизненных трудностей.
2. Развлечение игрока.
3. Помочь игроку испытать интересный, необычный опыт.

Основные требования:

1. Сделать красивый интерфейс.
2. Разработка интересной механики.

### **1.2.1 Choice of Robots[4]**

Игра Choice of Robots - лучше всего подходит под формат «текстовой», ведь это интерактивный научно-фантастический роман. На экране вы не увидите ничего, кроме страниц книги, однако именно вы и определите ее сюжет.

Достоинства, которые подтверждают отзывы:

1. Большая свобода выбора и вариативность.
2. Хорошо написанная история.

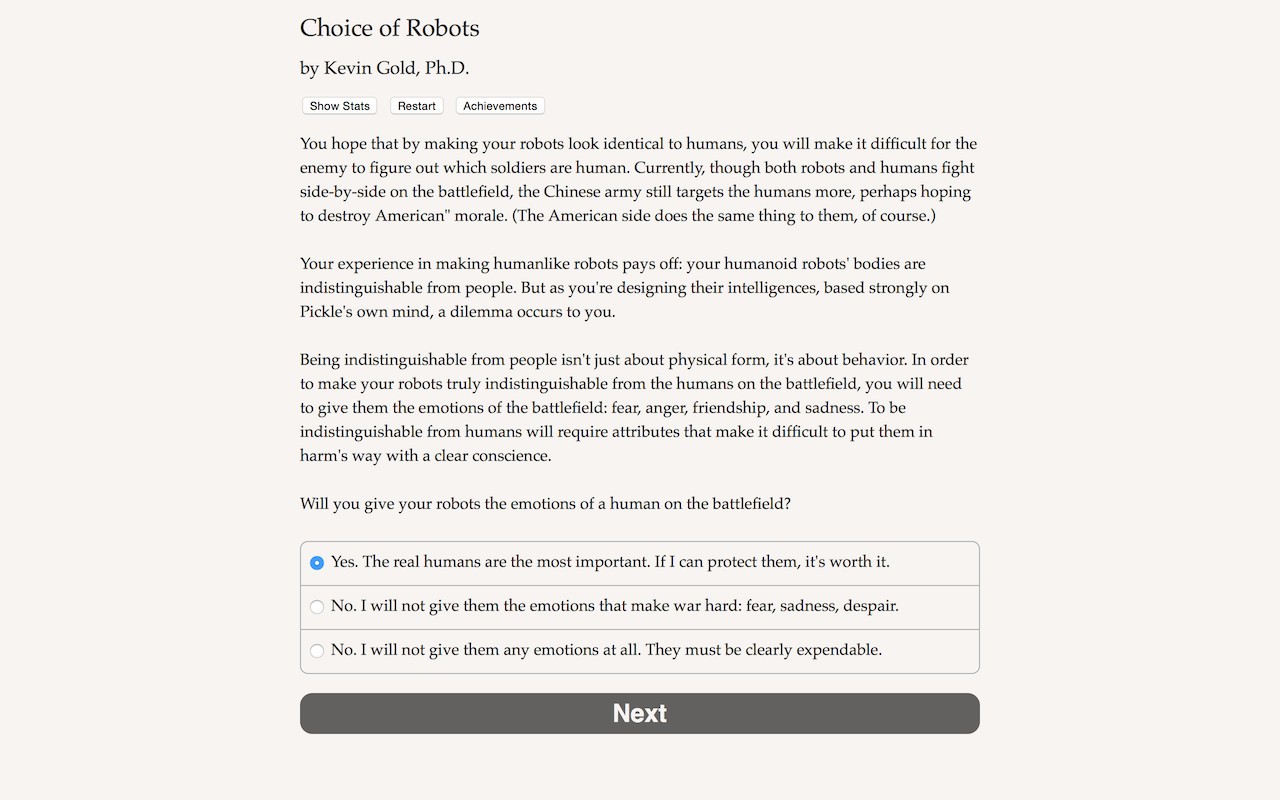


Рисунок 1. Интерфейс Choice of Robots

Данная игра состоит из основного окна, на котором происходит действие, а уже на нём находится следующие:

1. Название игры и имя автора.
2. Кнопка «Show Status», которая выводит статистику по игре.
3. Кнопка «Restart», которая начинает игру сначала.
4. Кнопка «Achievement»
5. Текст, основной источник информации.
6. Панель с выбором вариантов(ответов), которые влияют на развитие сюжета.
7. Кнопка «Next», которая выводит следующий пласт текста.

Игра Choice of Robots – является отличным образчиком игр такого жанра, интересная история не даст заскучать. Из серьёзных минусов: отсутствие локализации и отсутствие визуальных образов.

### **1.2.2 The Life and Suffering of Sir Brante[5]**

The Life and Suffering of Sir Brante – игра от отечественного разработчиков, повествующая о жизни и страдании главного героя.

Достоинства, которые подтверждают отзывы:

1. Интересный и захватывающий сюжет.
2. Огромная вариативность. Большое количество сюжетных развилок.
3. Красивый интерфейс.
4. Интересные персонажи.



Рисунок 2. Интерфейс The Life and Suffering of Sir Brante

Данная игра состоит из основного окна, на котором происходит действие, и на экране находится следующие:

1. Текст, источник информации.
2. Картинка, показывающая происходящие в тексте.
3. Кнопки

3.1. Стилизованные кнопки, которые двигают текст либо вперёд, либо назад.

3.2. Кнопка настройки.

3.3. Прочие

The Life and Suffering of Sir Brante- пример того, во что может развиться наш проект. Отличный визуальный стиль. Интересное переосмысление старых механик и интересный сюжет, делает эту игру одной из лучших в жанре.

# **2. Проектная часть**

## **Построение диаграммы прецедентов**

На рисунке 3 представлена диаграмма прецедентов взаимодействия игрока с системой. Как видно на рисунке, игрок, нажимая на кнопки отдает команды системе, та в свою очередь выполняет весь функционал программы, от восстановления здоровья до столкновения с противником.

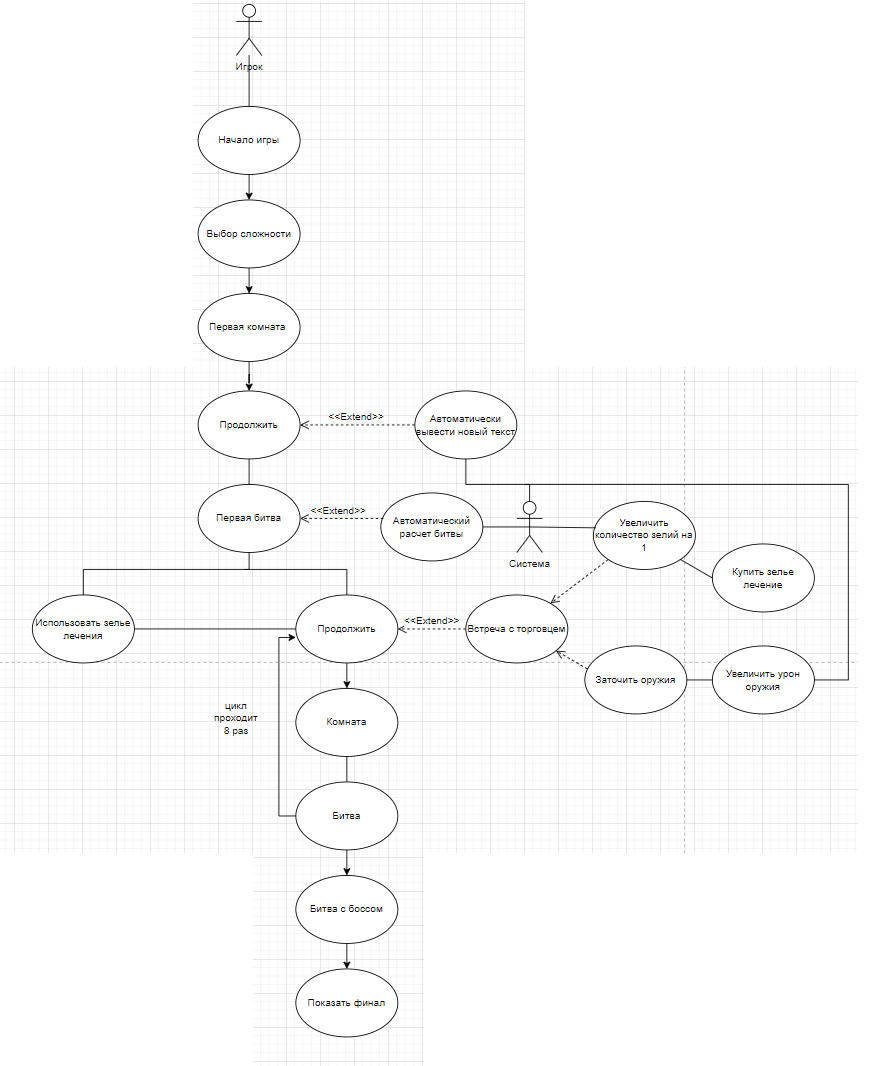
****

Рисунок 3. Диаграмма прецедентов

## **Выбор инструментов**

При выборе инструментов было проведено сравнение по критериям:

1. Простота использования.
2. Наличие нужных библиотек.
3. Возможность разделения проекта на модули.
4. Наличие документации на русском языке.

Таблица 1

Оценка языков программирования.

|  |  |  |
| --- | --- | --- |
| Критерии языка программирования | Kotlin | Python |
| Простота использования. | 7 | 10 |
| Наличие нужных библиотек. | 7 | 10 |
| Возможность разделения проекта на модули. | 6 | 10 |
| Наличие документации на русском языке | 8 | 9 |
| Скорость разработки | 6 | 10 |
| Итого баллов | 34 | 49 |

По выше перечисленным критериям был выбран язык Python. Т.к. в нём присутствуют нужные нам библиотеки, так же на нём легко создавать отдельные модули, нами он был изучен больше чем Kotlin.

## **Проектирование** **сценария**

В данном разделе приведен сценарий использования игры.

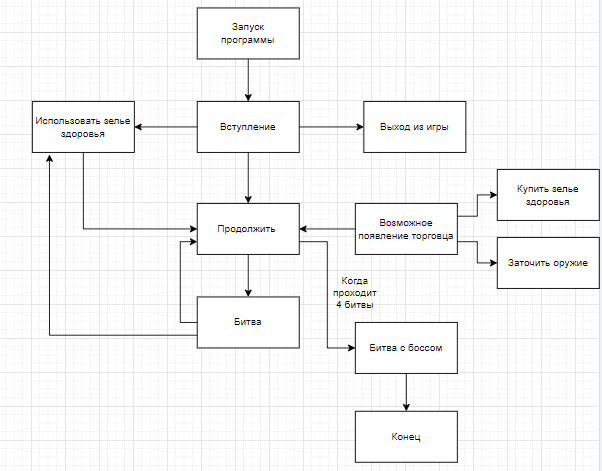


Рисунок 4. Сценарий использования

Пользователь после запуска программы может выполнить 3 действия: продолжить, использовать зелье здоровья, выход из игры. Все эти действия показаны на рисунке 4.

При выборе “выход из игры” программа закрывается, при выборе “продолжить” текст переключается дальше, при выборе “использовать зелье здоровья” показатель здоровья (в единицах) повышается на 20. Так же при выборе кнопки “продолжить”, рандомно может появиться торговец, у него можно “купить зелье здоровья” или повысить урон оружия. Когда проходит 4 битвы, перед главным героем появляется босс, после битвы с ним игра заканчивается.

## **Диаграмма классов**

В данном разделе представлены все классы, использующиеся в проекте, а также их отношения между собой. Для более удобного понимания была сделана диаграмма классов (рис 5).

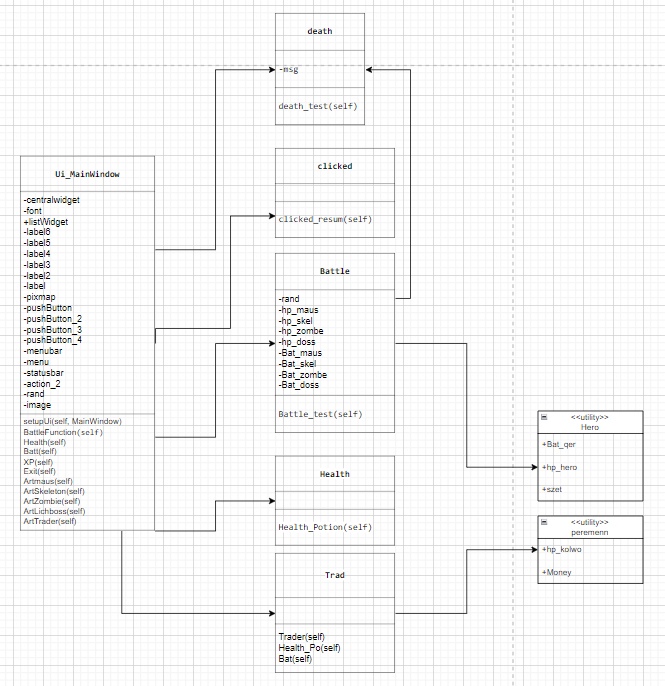
****

Рисунок 5. Диаграмма классов для проекта

Класс «Ui\_MainWindow», содержит публичные методы setupUi (self, MainWindow), создает главное меню со всем интерфейсом, BattleFunction(self) создает и убирает текст, спрайты, кнопки, Health(self) запускает функцию покупки зелий из класса Trad, Batt(self) запускает функцию усиления урона из класса Trad, XP(self) запускает функцию восстановления здоровья из класса Health, Exit(self) закрывает программу. Artmaus(self), ArtSkeleton(self), ArtZombie(self), ArtLichboss(self), ArtTrader(self) создают спрайты персонажей.

Класс «Hero» содержит публичные поля Bat\_qer (урон героя), hp\_hero (здоровье героя) и szet (счетчик боев).

Класс «peremenn» содержит публичные поля hp\_kolwo, Money.

Класс «Battle» содержит публичный метод Battle\_test(self), функция симулирует битву, вызывает спрайты из класса «Ui\_MainWindow».

Класс «death» содержит публичный метод death\_test(self), делает проверку на гибель персонажа и выводит текст.

Класс «clicked» содержит публичный метод clicked\_resum(self), который вызывает функции из классов Battle и death.

Класс «Health» содержит публичный метод Health\_Potion(self), функция которая восстанавливает здоровье главного героя.

Класс «Trad» содержит публичный метод Trader(self), данная функция выводит текст при появлении торговца, Health\_Po(self) функция покупки зелий, Bat(self) функция увеличивает урон персонажа.

## **Описание главного модуля**

В главный модуль входит класс Ui\_MainWindow который отвечает за функционал программы.

В главный модуль также импортируется еще 2 модуля, один из которых отвечает за игровую механику, а за функционал торговца.

Также в главный модуль входит код, который запускает саму программу при ее запуске. Полный код главного модуля находится в приложении 1.

В классе Ui\_MainWindow реализованы следующие методы:

**Листинг 1. Обработка всех событий в программе.**

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

MainWindow = QMainWindow()

ui = Ui\_MainWindow()

ui.setupUi(MainWindow)

app.setStyleSheet(qss)

MainWindow.show()

app.exec()

Данный метод создает окно с интерфейсом и задним фоном.

**Листинг 2. Создание интерфейса.**

def setupUi(self, MainWindow):

"""Создание фона игры"""

MainWindow.setObjectName("MainWindow")

MainWindow.setStyleSheet("#MainWindow{border-image:url(Background.png)}")

"""Полноэкранный режим"""

MainWindow.setWindowState((MainWindow.windowState() &

~(QtCore.Qt.WindowMinimized | QtCore.Qt.WindowMaximized))

| QtCore.Qt.WindowFullScreen)

"""Создание текстового поля"""

self.centralwidget = QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

font = QtGui.QFont()

font.setPointSize(36)

self.listWidget = QListWidget(self.centralwidget)

self.listWidget.setGeometry(QRect(420, 440, 515, 200))

self.listWidget.setObjectName("listWidget")

###############################################################################

"""Раздел создающий лейблы для изображений и размещение спрайтов"""

self.label6 = QLabel(self.centralwidget)

self.label6.move(429, 10)

self.label6.show()

self.label5 = QLabel(self.centralwidget)

self.label5.move(429, 10)

self.label5.show()

self.label4 = QLabel(self.centralwidget)

self.label4.move(429, 10)

self.label4.show()

self.label3 = QLabel(self.centralwidget)

self.label3.move(429, 10)

self.label3.show()

self.label2 = QLabel(self.centralwidget)

self.label2.move(429, 10)

self.label2.show()

self.label = QLabel(self.centralwidget)

self.label.move(429, 10)

self.pixmap = QPixmap('Lady.jpg')

self.pixmap = self.pixmap.scaled(500, 400)

self.pixmap = self.pixmap.scaled(self.pixmap.width(), self.pixmap.height())

self.label.setPixmap(self.pixmap)

#######################################################################################

self.listWidget.addItem(

"С древних врёмен, когда боги ходили по земле, существовала страна под названием\nАльбион!\n"

"Жили в ней храбрые рыцари, сталью и кровью они служили своей Вечной Королёве и \nбыли защитниками своей страны.\n"

"Но злой Король Лич напал на великое королевство.\n"

"Убив Лорда защитника и похитив его дочь он скрылся в своём древнем подземелье.\n"

"Большинство рыцарей либо пали, либо в далёких землях, о храбрый рыцарь,\n"

"тебе предстоит спасти принцессу и защитить свою родину! \n"

"Но помни, Лич поднял мертвых на свою сторону, сможешь ли ты победить?")

"""Создание вводного текста"""

#################################Создание кнопок#################################################################

self.pushButton = QPushButton("Продолжить", self.centralwidget)

self.pushButton.setGeometry(QtCore.QRect(540, 640, 101, 41))

self.pushButton\_2 = QPushButton("Зелье здоровья", self.centralwidget)

self.pushButton\_2.setGeometry(QRect(720, 640, 101, 41))

self.pushButton\_2.setObjectName("pushButton\_2")

font = QtGui.QFont()

font.setFamily("Rage Italic")

font.setPointSize(12)

self.pushButton.setFont(font)

self.pushButton.setObjectName("pushButton")

##################################################################################################

"""Создание меню"""

MainWindow.setCentralWidget(self.centralwidget)

self.menubar = QMenuBar(MainWindow)

self.menubar.setGeometry(QRect(0, 0, 800, 21))

self.menubar.setObjectName("menubar")

self.menu = QMenu("Меню", self.menubar)

self.menu.setObjectName("menu")

MainWindow.setMenuBar(self.menubar)

self.statusbar = QStatusBar(MainWindow)

self.statusbar.setObjectName("statusbar")

MainWindow.setStatusBar(self.statusbar)

self.action\_2 = QAction("Выход")

self.action\_2.setObjectName("action\_2")

self.menu.addAction(self.action\_2)

self.menubar.addAction(self.menu.menuAction())

QtCore.QMetaObject.connectSlotsByName(MainWindow)

MainWindow.setWindowTitle("Игра")

self.action\_2.triggered.connect(self.Exit)

self.pushButton.clicked.connect(self.BattleFunction) # чтобы вызвать функцию битвы

self.pushButton\_2.clicked.connect(self.XP)

"""Создание кнопок торговца"""

self.pushButton\_3 = QPushButton("Купить зелье\n здоровья", self.centralwidget)

self.pushButton\_3.setGeometry(QRect(950, 500, 101, 41))

self.pushButton\_3.setObjectName("pushButton\_3")

self.pushButton\_4 = QPushButton("Заточить меч", self.centralwidget)

self.pushButton\_4.setGeometry(QRect(950, 550, 101, 41))

self.pushButton\_4.setObjectName("pushButton\_4")

self.pushButton\_3.hide()

self.pushButton\_4.hide()

Данный метод создаёт главное окно и внутренний интерфейс.

**Листинг 3. Удаление спрайтов, кнопок и вызов функций.**

def BattleFunction(self):

"""Уберание всех изобращений с экрана"""

self.label6.hide()

self.label5.hide()

self.label4.hide()

self.label3.hide()

self.label2.hide()

self.label.hide()

"""Уберание кнопок торговца"""

self.pushButton\_3.hide()

self.pushButton\_4.hide()

self.listWidget.clear()

rand = randint(1, 100)

"""Определение рандомных моментов"""

if rand <= 20:

"""Появление кнопкок и вызов модуля торговца"""

self.pushButton\_3.show()

self.pushButton\_4.show()

self.pushButton\_3.clicked.connect(self.Health)

self.pushButton\_4.clicked.connect(self.Batt)

Traderf.Trad.Trader(self) # функция для торговца

else:

BattleFunction.clicked.clicked\_resum(self) # вызов модуля битвы

**Листинг 4. Покупка зелий и увеличения их количества.**

def Health(self):

"""Функция покупки зелий"""

Traderf.Trad.Health\_Po(self)

Данный метод увеличивает кол-во лечебных зелий и уменьшает кол-во денег.

**Листинг 5. Усиления персонажа.**

def Batt(self):

"""Функция уселения персонажа"""

Traderf.Trad.Bat(self)

Данный метод повышает урон персонажа и меняет цифры в списке урона.

**Листинг 6. Восстановление здоровья главного героя.**

def XP(self):

"""Функция выстоновление здоровья"""

BattleFunction.Health.Health\_Potion(self)

Данный метод увеличивает здоровья персонажа и уменьшает кол-во целебных зелий на один.

**Листинг 7. Выход.**

def Exit(self):

"""Функция выхода из игры"""

self.close

При срабатывания данного метода программа закрывается.

**Листинг 8. Вывод спрайта.**

def Artmaus(self):

image = 'Art/maus.jpg'

self.pixmap = QPixmap(image)

self.pixmap = self.pixmap.scaled(500, 400)

self.pixmap = self.pixmap.scaled(self.pixmap.width() , self.pixmap.height () )

self.label2.setPixmap(self.pixmap)

self.label2.adjustSize()

self.label2.show()

Данный метод вызывает спрайт мыши. Такими же методами вызываются остальные враги (См. приложение 1).

## **Описание спецификаций к модулям**

В данном разделе описаны публичные члены модулей курсового проекта.

В главном модуле (tracker.py) содержатся публичные методы класса Ui\_MainWindow. Так как все эти методы были описаны в разделе 2.5, в данном разделе описание данных методов опущено. В данном модуле также есть несколько объектов:

1. app – объект, который создает системное окно и объект самого приложения.
2. MainWindow – объект интерфейса, представленный в виде стека окон.
3. ui – объект который представляет собой сам интерфейс.

В модуле, который реализует интерфейс (tracker.py) так же содержатся публичные методы класса Ui\_MainWindow, которые будут описаны ниже в разделе 2.7, в данном разделе они описаны не будут. В данном классе так же есть публичные объекты, все они будут перечислены и описаны в этом разделе.

Объекты класса Ui\_MainWindow:

1. centralwidget – объект класса qwidget который создает главное окно.
2. font – шрифт.
3. listWidget – создание текстового поля.
4. label6 – поле для размещения спрайта.
5. label5 – поле для размещения спрайта.
6. label4 – поле для размещения спрайта.
7. label3 – поле для размещения спрайта.
8. label2 – поле для размещения спрайта.
9. label – поле для размещения спрайта.
10. Pixmap – вывод спрайта.
11. pushButton – кнопка продолжить.
12. pushButton\_2 – кнопка зелье здоровье.
13. menubar – для создания верхней полосы для настроек.
14. menu – кнопка для меню.
15. statusbar – статус строки.
16. action\_2 – кнопка для выхода.
17. pushButton\_3 - кнопка для покупки зелья.
18. pushButton\_4 – кнопка для увеличения урона.
19. image – ссылка на спрайт.

## **Описание модулей**

Кроме главного модуля программа содержит еще два модуля которые реализует игровую механику (BattleFunction.py, Traderf. py).

Полный код модулей приведен в приложениях 2 и 3. В этом разделе приведена блок-схема (рис.6) все методы, которые содержаться в классах Hero, peremenn, Battle, death, clicked, Health из модуля BattleFunction.py и Trad из модуля Traderf. py.

**Листинг 9. Метод класса Battle.**

def Battle\_test(self):

"""Функция запускает саму битву"""

**Листинг 10. Метод класса death.**

def death\_test(self):

"""функция, проверка на показатель хп"""

**Листинг 11. Метод класса clicked.**

def clicked\_resum(self):  
 *""" данная функция вызывает другие функции Battle\_test, death\_test"""*

**Листинг 12. Метод класса** **Health.**

def Health\_Potion(self): *"""Функция восстанавливающая здоровье главного героя"""*

**Листинг 13. Методы класса Trad.**

*def Trader(self):*

*"""Функция выводящая текст торговца"""*

def Health\_Po(self):  
 *""" функция покупка зелий здоровья у торговца, если не хватает денег, об этом сообщается"""*

def Bat(self):  
 *""" функция заточка оружия у торговца, если не хватает денег, об этом сообщается"""*

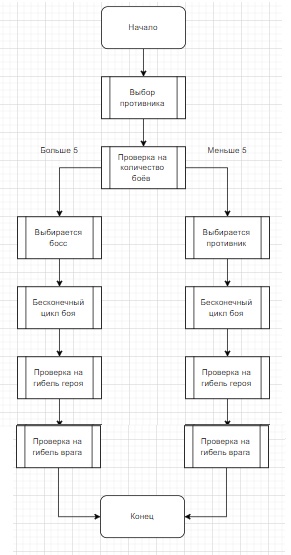


Рисунок 6. Блок-схема метода Battle.

## **Описание тестовых наборов модулей**

Проведём тестирование. Начнем игру, проведем бой, поторгуем с торговцем и используем зелье лечения.

1. Запустим нашу игру. Мы видим, что спрайт выводится вместе с текстом, значит все работает правильно.

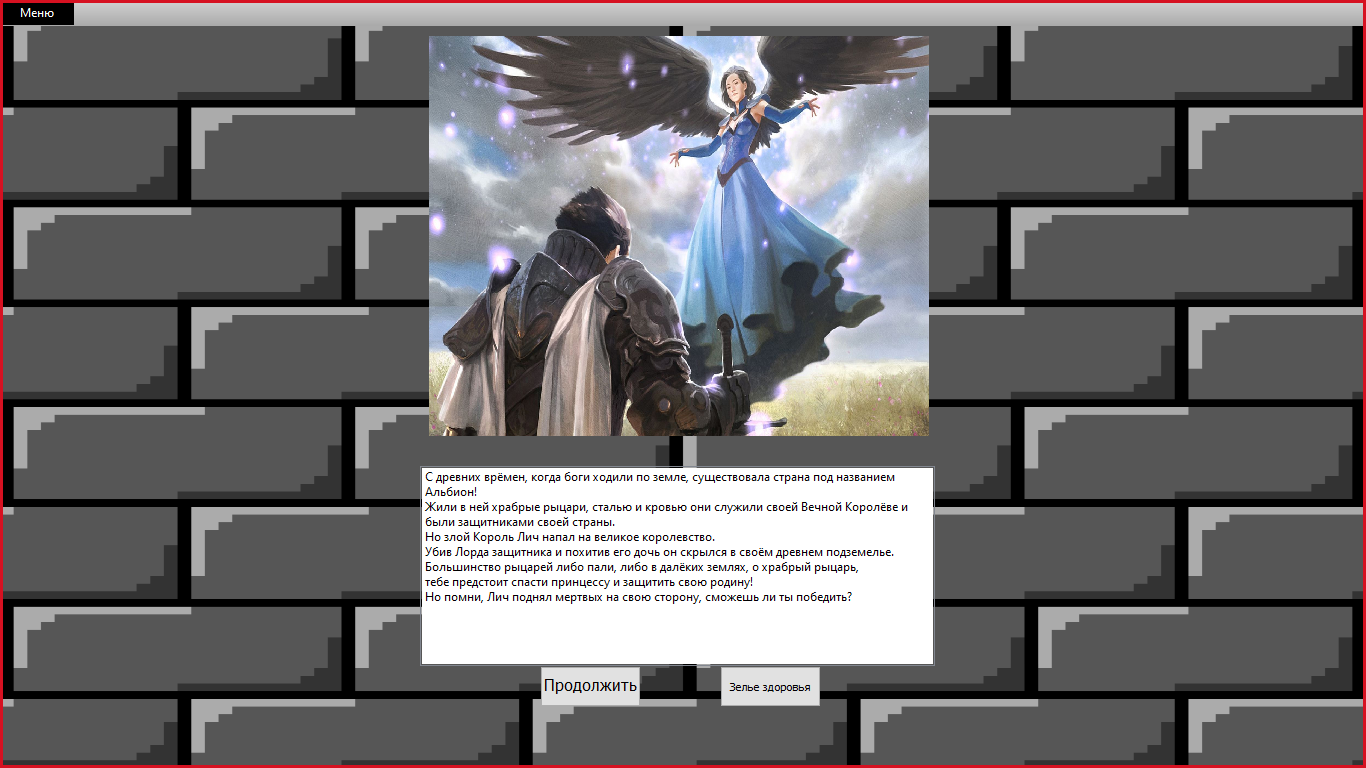


Рисунок 7 Запуск программы.

1. При нажатии на кнопку “Продолжить” спрайт меняется на врага, а в тексте идет описание боя.

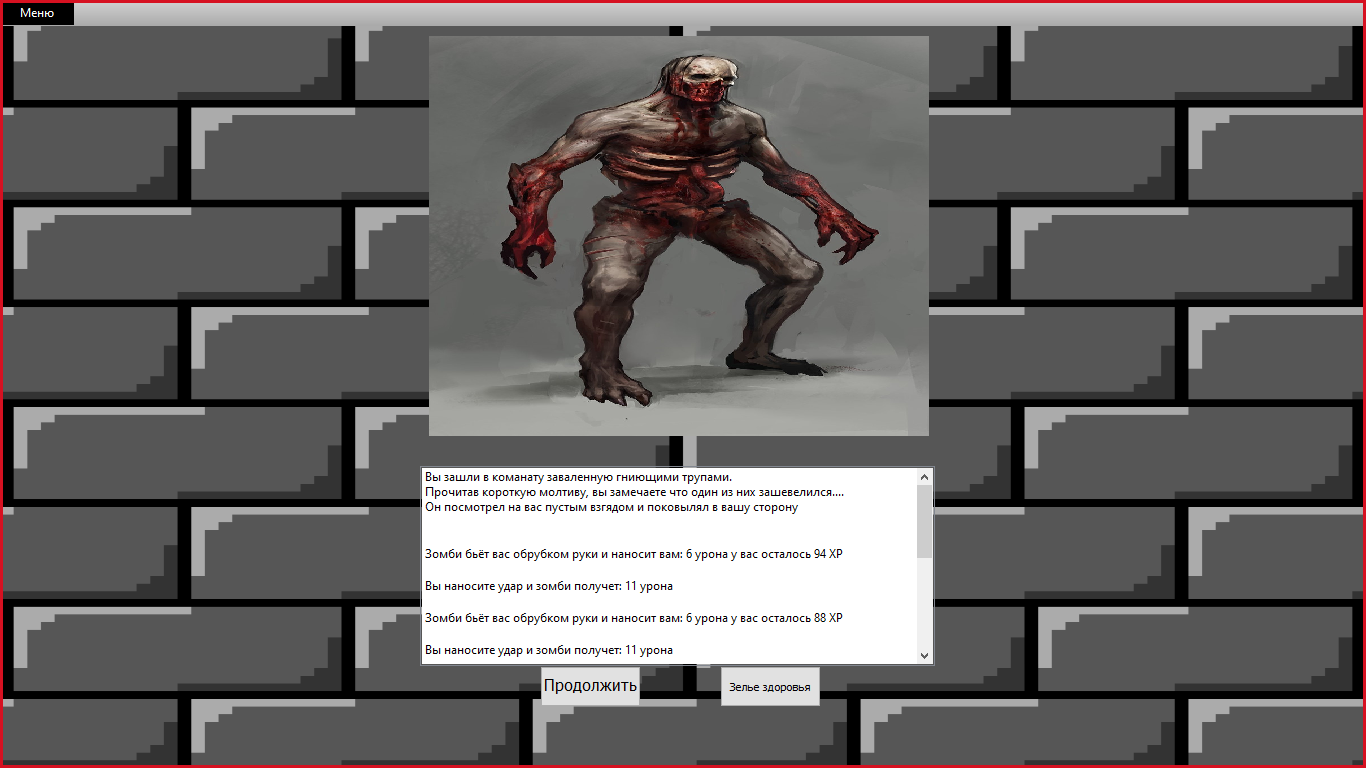


Рисунок 8. Бой.

1. После битвы мы можем восстановить здоровье героя, при нажатии на кнопку “Зелье здоровья” здоровье героя увеличиается.

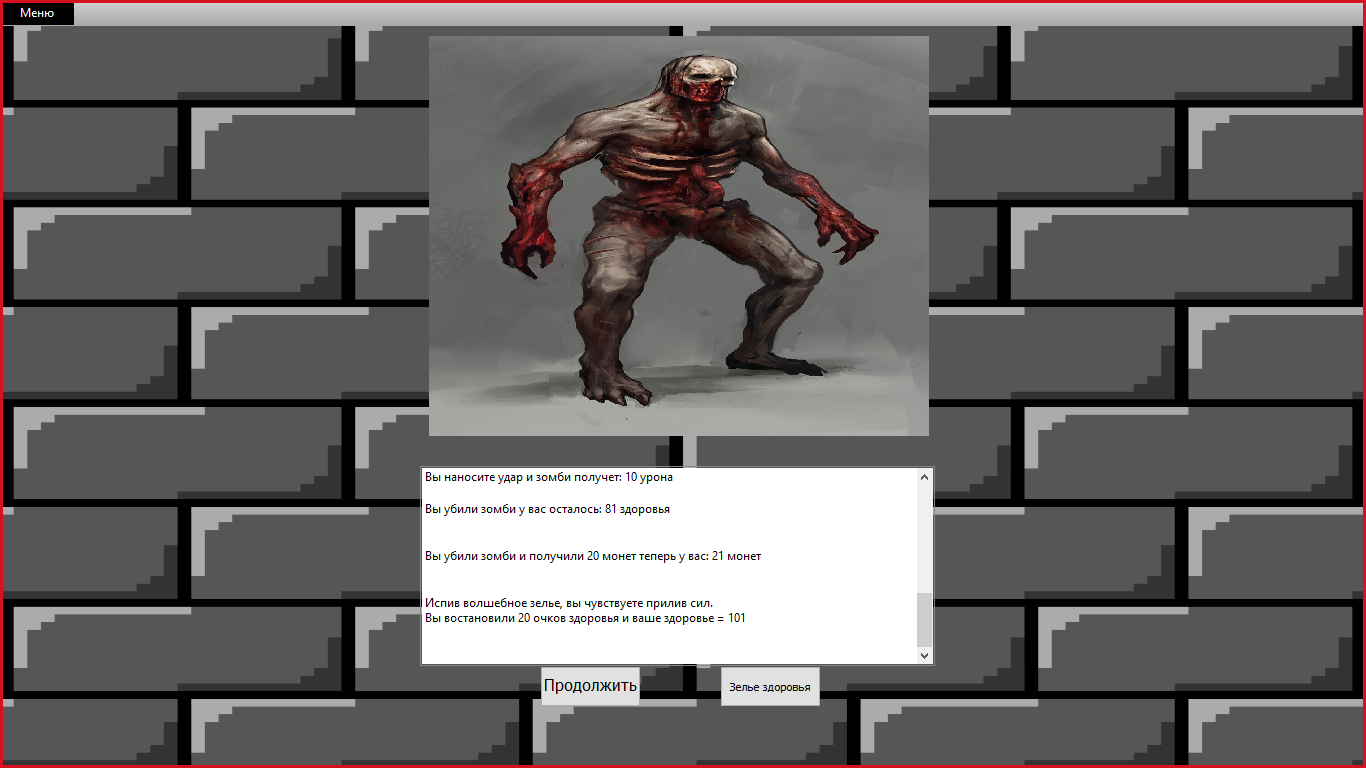


Рисунок 9. Восстановление здоровья.

1. После того как мы восстановили здоровье и продолжили путь появился торговец. После удачной торговли мы пошли дальше.

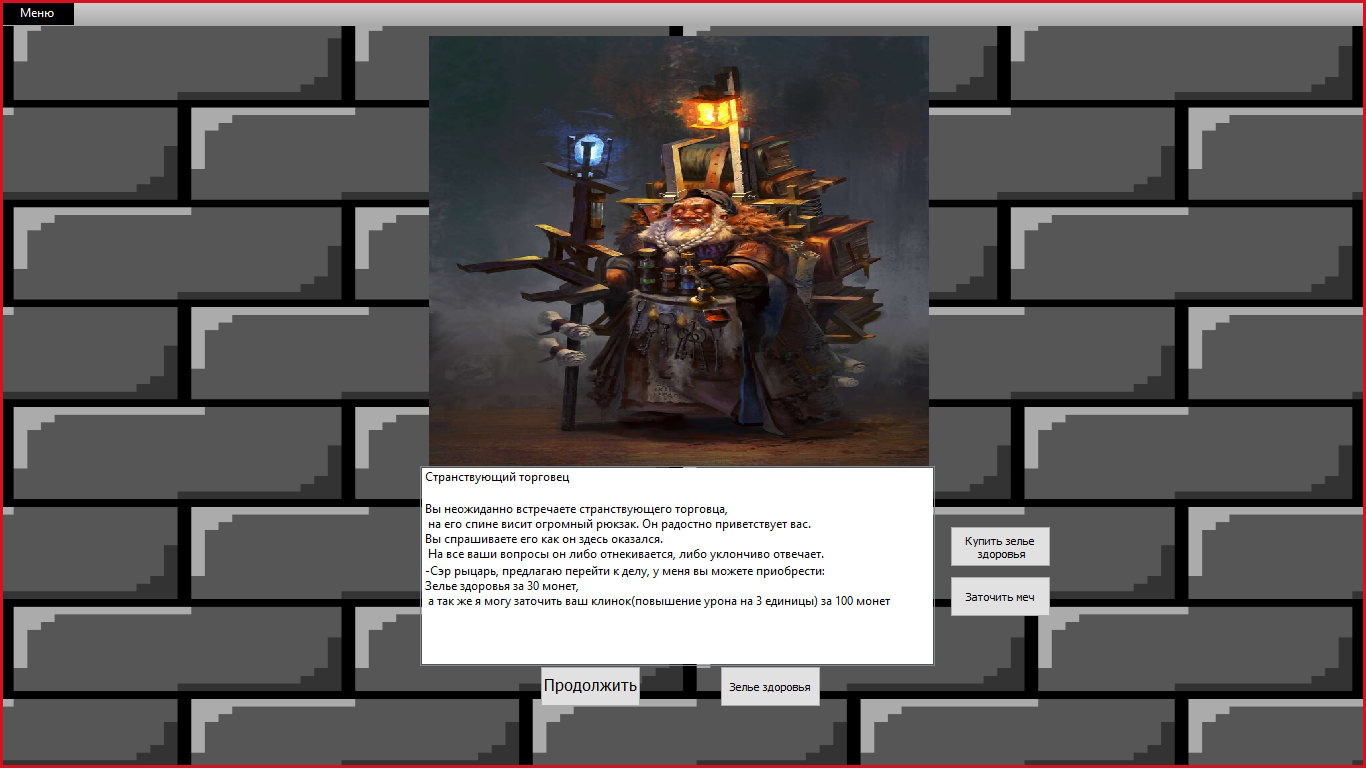


Рисунок 10. Торговец.

Завершив цикл игры, мы убедились, что все механики игры работают правильно.

## **Описание применения средств отладки**

В этом разделе показано умение применять средства отладки.

Откладка — это процесс поиска и исправления ошибок в коде программы.

Debug Everywhere

Конечно, PyCharm умеет отлаживать код, который выполняется на локальном компьютере в системном окружении, virtualenv, Anaconda или Conda. С PyCharm Professional Edition вы также сможете отлаживать код, который запускается в Docker-контейнере, на виртуальной машине или на удаленном хосте через SSH.

1. Перейдем в программе на интересующую нас строку, начиная с которой будет начала отладки.
2. В главном меню выбираем Run -> Toggle Line Breakpoint или Run -> Toggle Temporary Line Breakpoint:

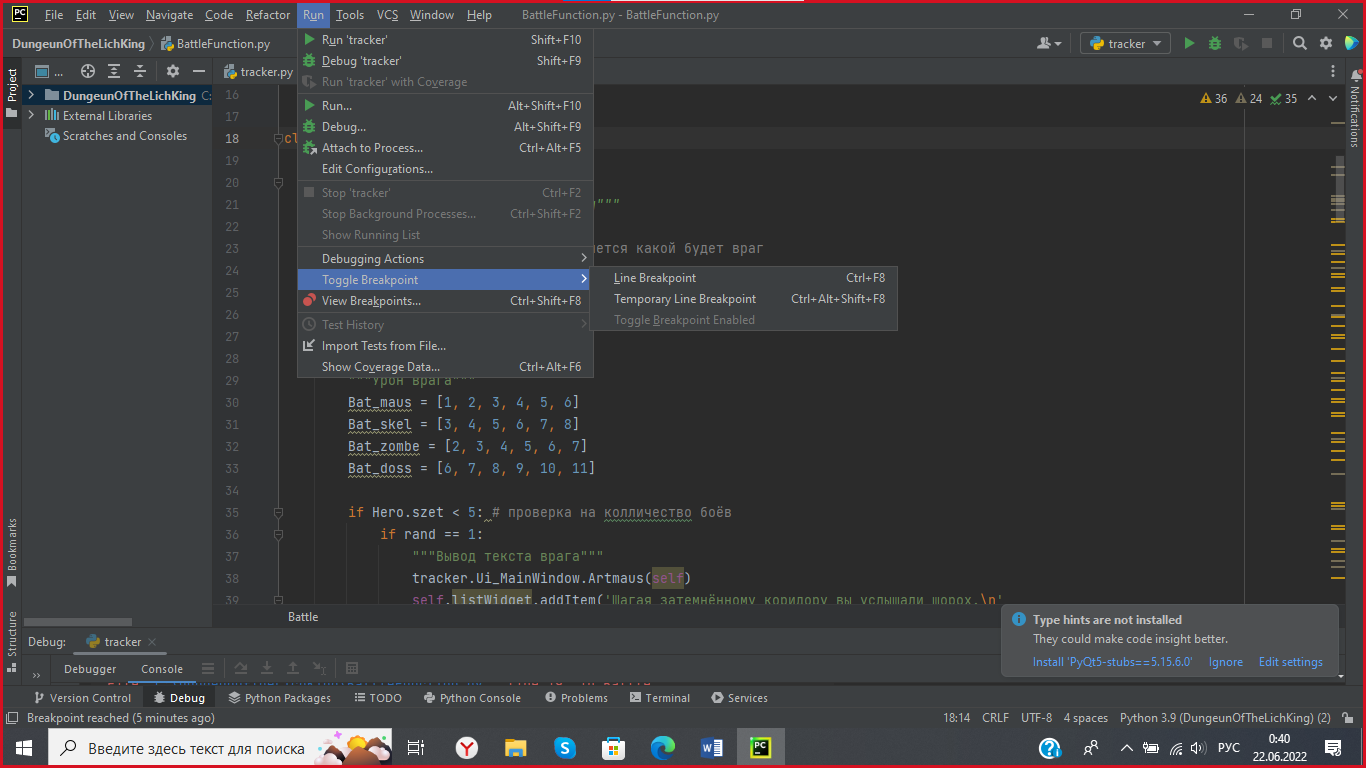


Рисунок 11. Отладка программы

1. Теперь рядом со строкой появилась красная жирная точка. Это точка останова. Теперь при выполнении программы в режиме отладки среда остановит её в этом месте, и можно будет узнать состояние программы.
2. Нажмем правой кнопкой на название файла с программой и выберем Debug:

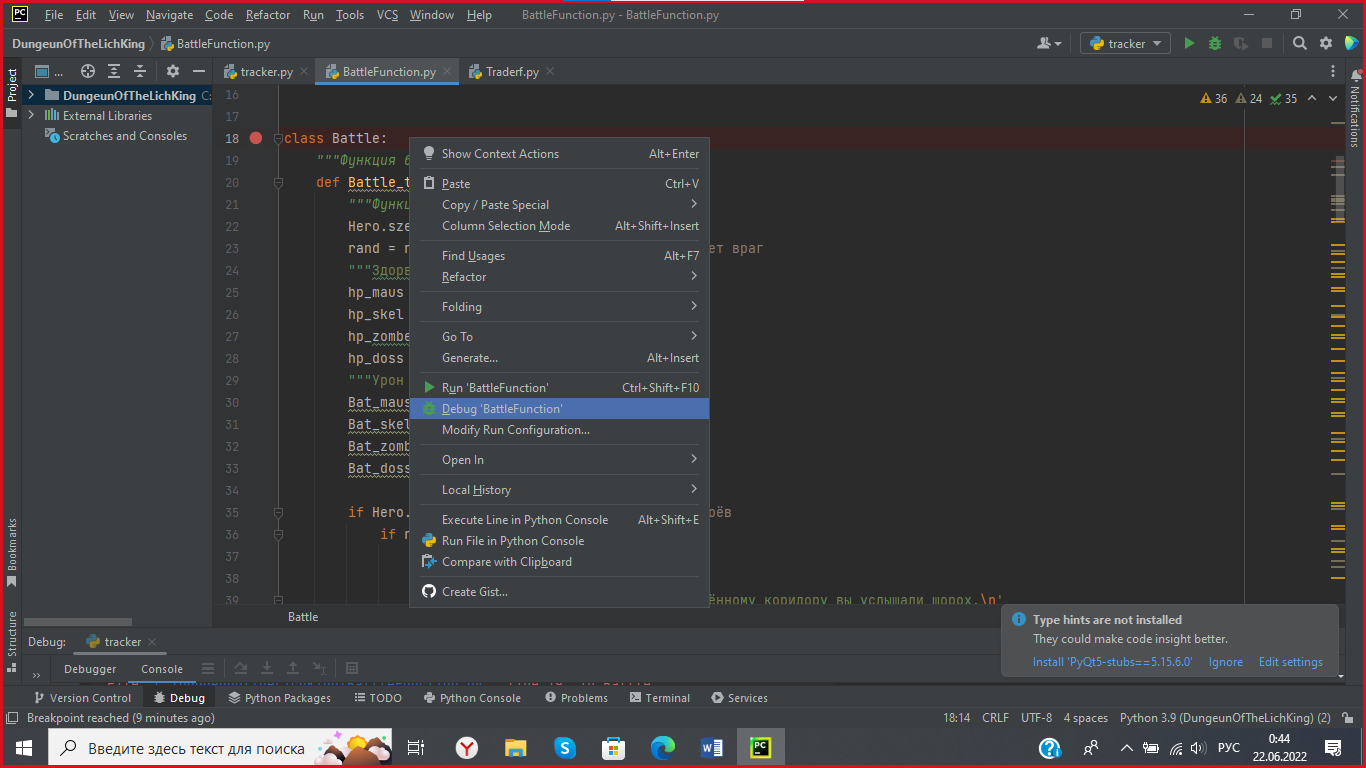


Рисунок 12. Использование функции Debug

1. Теперь программа запустилась и остановилась на указанной строке. Текущее положение интерпретатора Python в программе отмечается синей строкой. В нижней части экрана появилась вкладка отладки. Там виден список переменных, доступных из данной точки программы.
2. Для того, чтобы проследовать вслед за интерпретатором Python в класс (на строку 18), в главном меню выберем Run -> Debugging Actions-> Step Into, или нажмем клавишу F7.

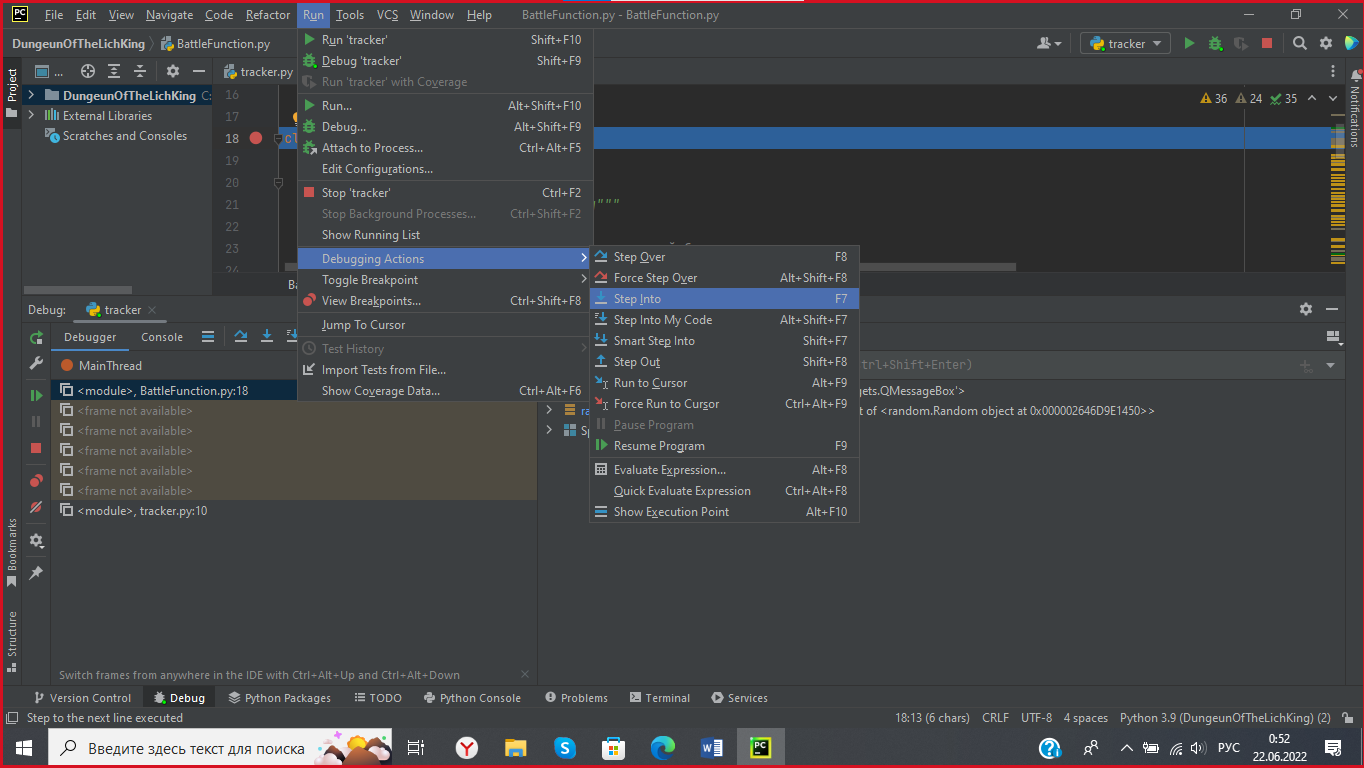


Рисунок 13. Использование функции Step Into

1. Как и ожидалось, интерпретатор переместил фокус своего внимания на строку 18, в классе Battle:
2. Чтобы сделать шаг вперёд, не заглядывая в используемые функции, используv команду Run -> Debugging Actions-> Step Over, или клавишу F8:

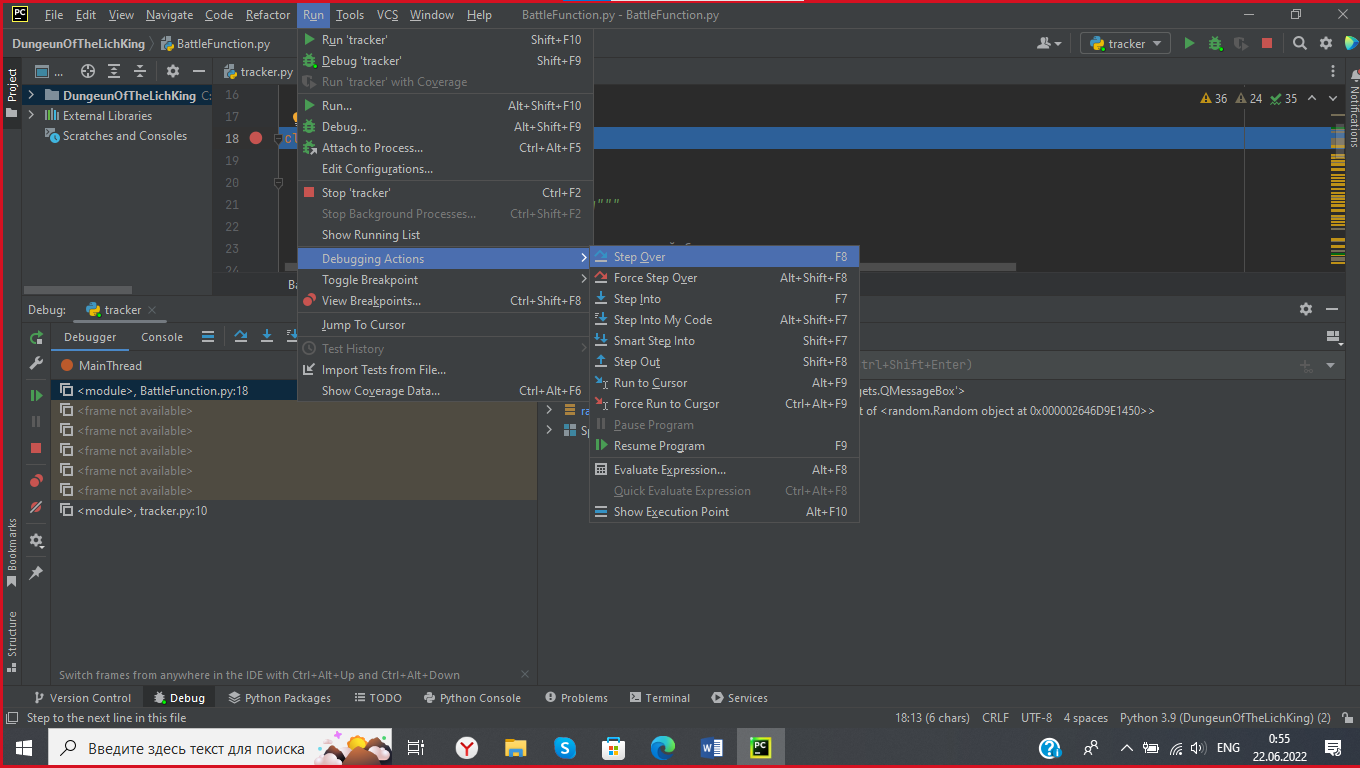


Рисунок 14. Использование функции Step Over

Таким образом можно легко узнавать, что происходит в каждой точке Вашей программы. Это часто необходимо для поиска ошибок, когда Вы не понимаете, почему программа выдаёт некоторый неверный ответ.

# **Эксплуатационная часть**

## **Руководство оператора**

**Аннотация**

В данном руководстве приведены инструкции по применению DungeunOfTheLichKing, которая служит для развлечения человека.

1. **Назначение программы**
2. **Функциональное назначение**

Пользователь использует данный продукт для развлечения.

1. **Эксплуатационное назначение**

Данный продукт могут играть люди от 12+ лет.

1. **Состав функций**
2. **Функция прохождения**

Данная функция позволяет проводить бой и идти дальше.

1. **Функция зелья здоровья**

Данная функция позволяет восстанавливать здоровья героя.

1. **Функция покупка зелий здоровья**

Данная функция позволяет покупать зелье здоровья, которые восстанавливают здоровье.

1. **Функция усиления урона**

Данная функция позволяет увеличить урон героя.

1. **Условия выполнения программы**
2. **Минимальные системные требования**

Операционная система: не ранее Windows 7.

ОЗУ: 200 МБ.

Место на диске: 200 МБ.

Разрешение экрана: не ниже 1024 на 768.

1. **Минимальный состав программных средств**

Наличие установленной PyCharm Community Edition.

1. **Требования к персоналу**

Отсутствует (нет персонала).

1. **Выполнение программы**
2. **Открытие, установка и запуск.**

Скачиваем файл переносим его на диск. Заходим в файл левой кнопкой нажимаем на файл tracker.py и выбираем “ Edit with PyCharm Community Edition”.

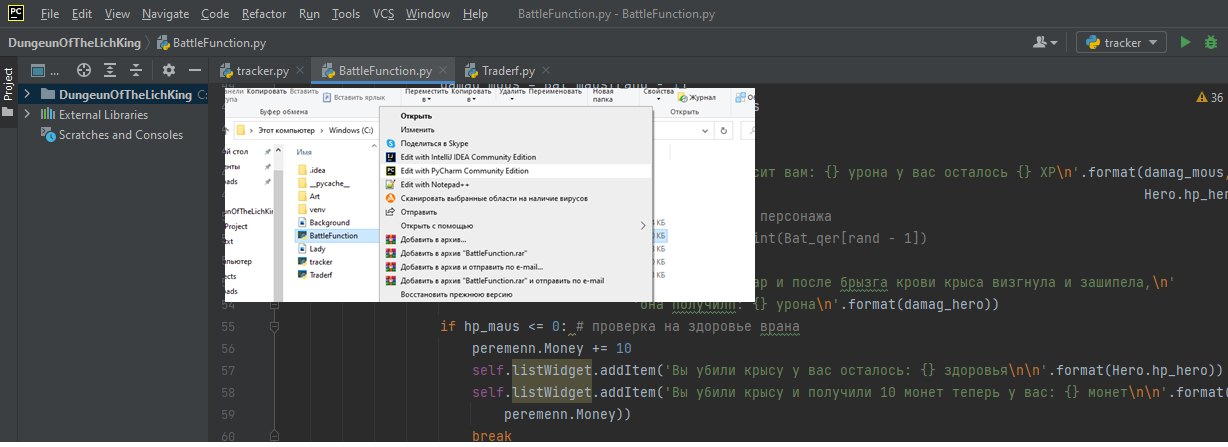


Рисунок 15. Открытие программы

Нажимаем на зеленую стрелочку, программа запускается.

1. **Продолжить**

При нажатии кнопки “продолжить” происходит битва.

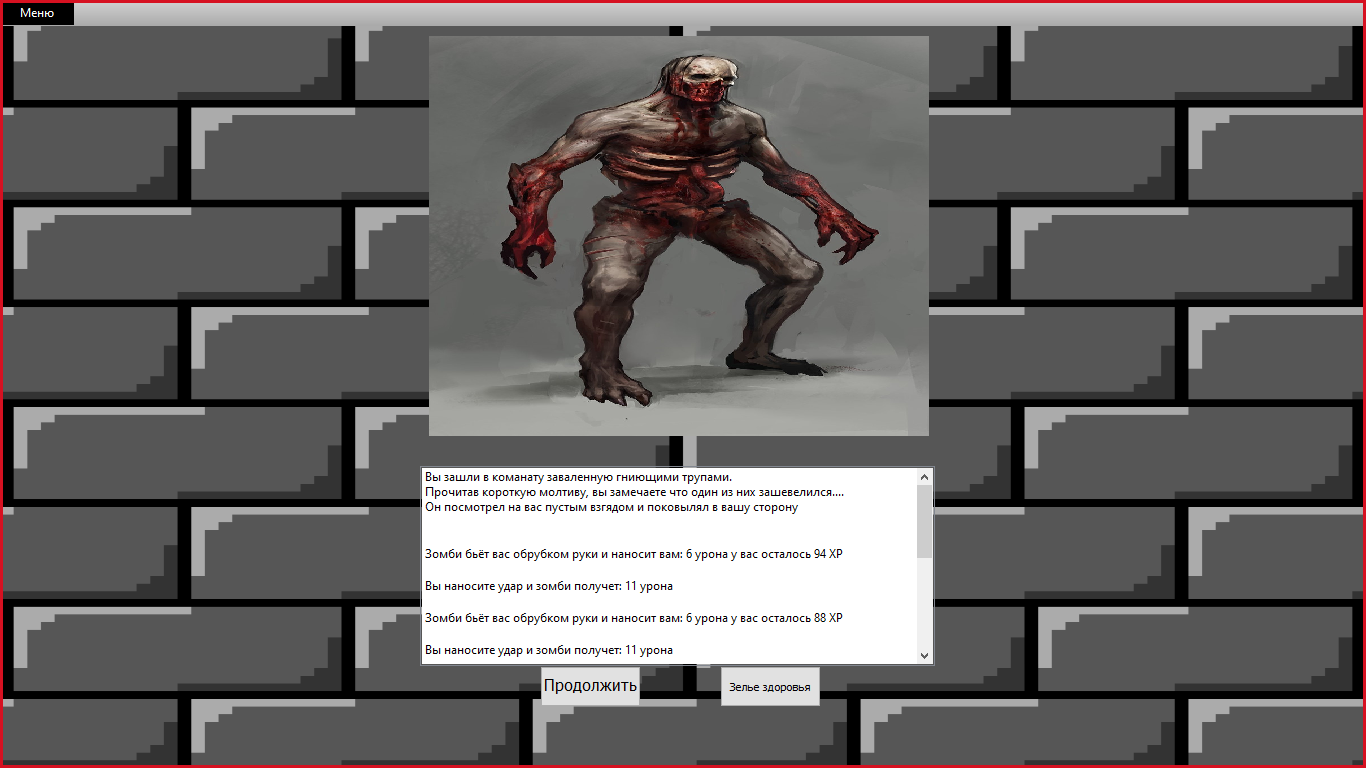


Рисунок 16. Бой.

1. **Зелье здоровья**

Нажатии кнопки “Зелье здоровья” происходит восстановление здоровья героя.

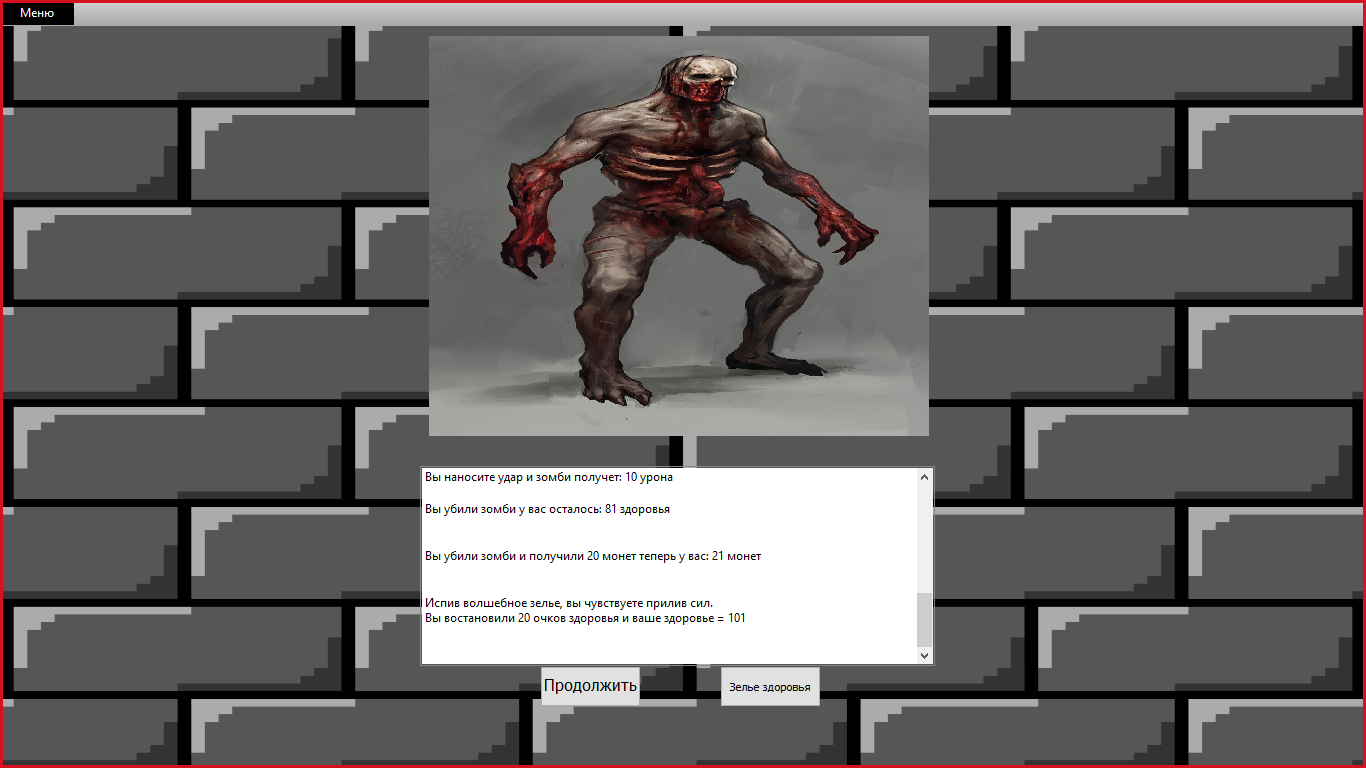


Рисунок 17. Восстановление здоровья.

1. **Кнопки торговца.**

У торговца присутствуют две кнопки “Купить зелье здоровья” и “Заточить меч”. При нажатии на кнопку “Купить зелье здоровья” увеличивается количество лечебных зелий, при нажатии на кнопку “Заточить меч” происходит увеличение урона героя.

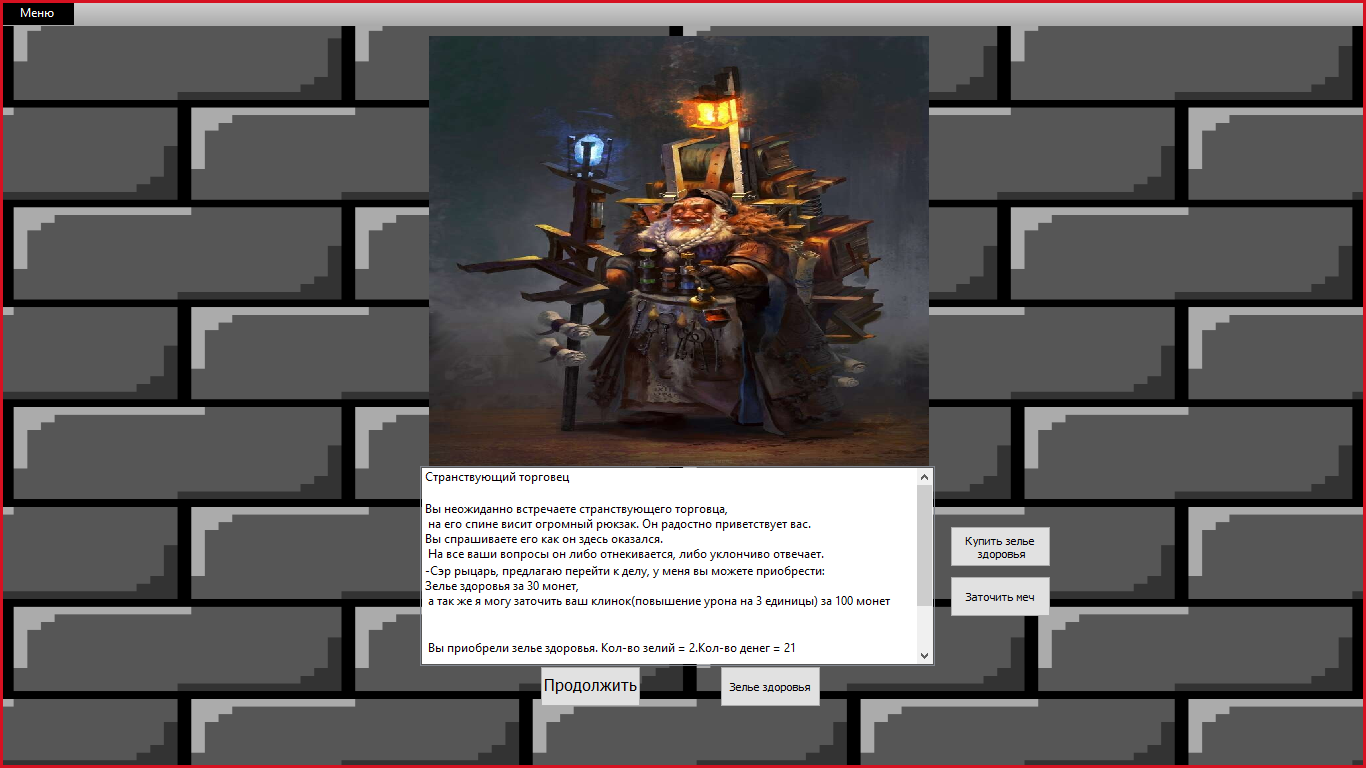


Рисунок 18. Кнопки торговца.

1. **Выход**

При нажатии кнопки “меню” выводится окно с кнопкой “Выход” и при нажатии на нее игра закрывается.

## **To-Do лист**

* Добавить полноценные комнаты.
* Добавить уровень сложности легкий, средний и сложный.
* Добавить новых противников.
* Увеличить продолжительность игры.
* Доработать и улучшить внешний вид программы.

# **Заключение**

Во время выполнения данного курсового проекта была исследована и написана игра Dungeun Of The Lich-King.

При написании программы была исследована среда разработки PyCharm Community Edition и работа с интерфейсом PyQt.

В результате данного исследования был сделан вывод, что процесс написания приложения, с использованием программ для создания интерфейсов и ранее неизвестными модулями, оказался очень трудоёмким. Подобные программы лучше всего писать группами из нескольких человек, с распределением обязанностей и областей программы.

Данный проект может использоваться в развлекательных целях. Он будет интересен людям, которым нравится ролевые игры и у которых не очень много свободного времени.

# **Список использованной литературы и Интернет-ресурсов**

1. Компьютерная ролевая игра.

[Компьютерная ролевая игра — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%80%D0%BE%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0)

1. Использование диаграммы классов UML при проектировании и документировании программного обеспечения.

[Использование диаграммы классов UML при проектировании и документировании программного обеспечения / Хабр (habr.com)](https://habr.com/ru/post/572234/)

1. Использование диаграммы вариантов использования UML при проектировании программного обеспечения.

[Использование диаграммы вариантов использования UML при проектировании программного обеспечения / Хабр (habr.com)](https://habr.com/ru/post/566218/)

1. Ссылка на игру Choice of Robots.

[Choice of Robots в Steam (steampowered.com)](https://store.steampowered.com/app/339350/Choice_of_Robots/)

1. Ссылка на игру The Life and Suffering of Sir Brante.

[The Life and Suffering of Sir Brante в Steam (steampowered.com)](https://store.steampowered.com/app/1272160/The_Life_and_Suffering_of_Sir_Brante/)

1. Видеоигры в учебной программе | Как игры используют в образовании

[Видеоигры в учебной программе | Как игры используют в образовании — Игры на DTF](https://dtf.ru/games/189651-videoigry-v-uchebnoy-programme-kak-igry-ispolzuyut-v-obrazovanii?ysclid=l4omi7968g128987136#2)

# **Приложение 1. Интерфейс tracker**

from random import randint  
from PyQt5 import QtCore, QtGui  
import sys  
  
from PyQt5.QtCore import QRect  
from PyQt5.QtGui import QPixmap  
from PyQt5.QtWidgets import QApplication, QMainWindow, QAction, QMenu, QStatusBar, QMenuBar, QPushButton, QListWidget, \  
 QWidget, QLabel  
  
import BattleFunction  
import Traderf  
  
  
class Ui\_MainWindow(object):  
  
  
 def setupUi(self, MainWindow):  
 *"""Создание фона игры"""* MainWindow.setObjectName("MainWindow")  
 MainWindow.setStyleSheet("#MainWindow{border-image:url(Background.png)}")  
 """Полноэкранный режим"""  
 MainWindow.setWindowState((MainWindow.windowState() &  
 ~(QtCore.Qt.WindowMinimized | QtCore.Qt.WindowMaximized))  
 | QtCore.Qt.WindowFullScreen)  
 """Создание текстового поля"""  
 self.centralwidget = QWidget(MainWindow)  
 self.centralwidget.setObjectName("centralwidget")  
  
 font = QtGui.QFont()  
 font.setPointSize(36)  
  
 self.listWidget = QListWidget(self.centralwidget)  
 self.listWidget.setGeometry(QRect(420, 440, 515, 200))  
 self.listWidget.setObjectName("listWidget")  
 ###############################################################################  
 """Раздел создающий лейблы для изображений и размещение спрайтов"""  
 self.label6 = QLabel(self.centralwidget)  
 self.label6.move(429, 10)  
 self.label6.show()  
  
 self.label5 = QLabel(self.centralwidget)  
 self.label5.move(429, 10)  
 self.label5.show()  
  
 self.label4 = QLabel(self.centralwidget)  
 self.label4.move(429, 10)  
 self.label4.show()  
  
 self.label3 = QLabel(self.centralwidget)  
 self.label3.move(429, 10)  
 self.label3.show()  
  
 self.label2 = QLabel(self.centralwidget)  
 self.label2.move(429, 10)  
 self.label2.show()  
  
  
  
 self.label = QLabel(self.centralwidget)  
 self.label.move(429, 10)  
 self.pixmap = QPixmap('Lady.jpg')  
 self.pixmap = self.pixmap.scaled(500, 400)  
 self.pixmap = self.pixmap.scaled(self.pixmap.width(), self.pixmap.height())  
 self.label.setPixmap(self.pixmap)  
 #######################################################################################  
  
 self.listWidget.addItem(  
 "С древних врёмен, когда боги ходили по земле, существовала страна под названием\nАльбион!\n"  
 "Жили в ней храбрые рыцари, сталью и кровью они служили своей Вечной Королёве и \nбыли защитниками своей страны.\n"  
  
 "Но злой Король Лич напал на великое королевство.\n"  
 "Убив Лорда защитника и похитив его дочь он скрылся в своём древнем подземелье.\n"  
  
 "Большинство рыцарей либо пали, либо в далёких землях, о храбрый рыцарь,\n"  
 "тебе предстоит спасти принцессу и защитить свою родину! \n"  
 "Но помни, Лич поднял мертвых на свою сторону, сможешь ли ты победить?")  
 """Создание вводного текста"""  
#################################Создание кнопок#################################################################  
 self.pushButton = QPushButton("Продолжить", self.centralwidget)  
 self.pushButton.setGeometry(QtCore.QRect(540, 640, 101, 41))  
  
 self.pushButton\_2 = QPushButton("Зелье здоровья", self.centralwidget)  
 self.pushButton\_2.setGeometry(QRect(720, 640, 101, 41))  
 self.pushButton\_2.setObjectName("pushButton\_2")  
  
 font = QtGui.QFont()  
 font.setFamily("Rage Italic")  
 font.setPointSize(12)  
 self.pushButton.setFont(font)  
 self.pushButton.setObjectName("pushButton")  
##################################################################################################  
 """Создание меню"""  
 MainWindow.setCentralWidget(self.centralwidget)  
 self.menubar = QMenuBar(MainWindow)  
 self.menubar.setGeometry(QRect(0, 0, 800, 21))  
 self.menubar.setObjectName("menubar")  
  
 self.menu = QMenu("Меню", self.menubar)  
 self.menu.setObjectName("menu")  
 MainWindow.setMenuBar(self.menubar)  
  
 self.statusbar = QStatusBar(MainWindow)  
 self.statusbar.setObjectName("statusbar")  
 MainWindow.setStatusBar(self.statusbar)  
  
 self.action\_2 = QAction("Выход")  
 self.action\_2.setObjectName("action\_2")  
 self.menu.addAction(self.action\_2)  
 self.menubar.addAction(self.menu.menuAction())  
  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 MainWindow.setWindowTitle("Игра")  
 self.action\_2.triggered.connect(self.Exit)  
 self.pushButton.clicked.connect(self.BattleFunction) # чтобы вызвать функцию битвы  
 self.pushButton\_2.clicked.connect(self.XP)  
 """Создание кнопок торговца"""  
 self.pushButton\_3 = QPushButton("Купить зелье\n здоровья", self.centralwidget)  
 self.pushButton\_3.setGeometry(QRect(950, 500, 101, 41))  
 self.pushButton\_3.setObjectName("pushButton\_3")  
  
 self.pushButton\_4 = QPushButton("Заточить меч", self.centralwidget)  
 self.pushButton\_4.setGeometry(QRect(950, 550, 101, 41))  
 self.pushButton\_4.setObjectName("pushButton\_4")  
 self.pushButton\_3.hide()  
 self.pushButton\_4.hide()  
  
  
 def BattleFunction(self):  
 *"""Уберание всех изобращений с экрана"""* self.label6.hide()  
 self.label5.hide()  
 self.label4.hide()  
 self.label3.hide()  
 self.label2.hide()  
 self.label.hide()  
 """Уберание кнопок торговца"""  
 self.pushButton\_3.hide()  
 self.pushButton\_4.hide()  
 self.listWidget.clear()  
 rand = randint(1, 100)  
 """Определение рандомных моментов"""  
 if rand <= 20:  
 """Появление кнопкок и вызов модуля торговца"""  
 self.pushButton\_3.show()  
 self.pushButton\_4.show()  
 self.pushButton\_3.clicked.connect(self.Health)  
 self.pushButton\_4.clicked.connect(self.Batt)  
 Traderf.Trad.Trader(self) # функция для торговца  
  
 else:  
 BattleFunction.clicked.clicked\_resum(self) # вызов модуля битвы  
  
 def Health(self):  
 *"""Функция покупки зелий"""* Traderf.Trad.Health\_Po(self)  
  
 def Batt(self):  
 *"""Функция уселения персонажа"""* Traderf.Trad.Bat(self)  
  
 def XP(self):  
 *"""Функция выстоновление здоровья"""* BattleFunction.Health.Health\_Potion(self)  
  
 def Exit(self):  
 *"""Функция выхода из игры"""* self.close  
######################################Появление спрайтов персонажей#####################################################  
 def Artmaus(self):  
  
 image = 'Art/maus.jpg'  
 self.pixmap = QPixmap(image)  
 self.pixmap = self.pixmap.scaled(500, 400)  
 self.pixmap = self.pixmap.scaled(self.pixmap.width(), self.pixmap.height())  
 self.label2.setPixmap(self.pixmap)  
 self.label2.adjustSize()  
 self.label2.show()  
  
 def ArtSkeleton(self):  
 image = 'Art/Skeleton.jpg'  
 self.pixmap = QPixmap(image)  
 self.pixmap = self.pixmap.scaled(500, 400)  
 self.pixmap = self.pixmap.scaled(self.pixmap.width(), self.pixmap.height())  
 self.label3.setPixmap(self.pixmap)  
 self.label3.adjustSize()  
 self.label3.show()  
  
 def ArtZombie(self):  
 image = 'Art/Zombie.jpg'  
 self.pixmap = QPixmap(image)  
 self.pixmap = self.pixmap.scaled(500, 400)  
 self.pixmap = self.pixmap.scaled(self.pixmap.width(), self.pixmap.height())  
 self.label4.setPixmap(self.pixmap)  
 self.label4.adjustSize()  
 self.label4.show()  
  
 def ArtLichboss(self):  
 image = 'Art/Lichboss.jpg'  
 self.pixmap = QPixmap(image)  
 self.pixmap = self.pixmap.scaled(500, 400)  
 self.pixmap = self.pixmap.scaled(self.pixmap.width(), self.pixmap.height())  
 self.label5.setPixmap(self.pixmap)  
 self.label5.adjustSize()  
 self.label5.show()  
  
 def ArtTrader(self):  
 image = 'Art/Trader.jpg'  
 self.pixmap = QPixmap(image)  
 self.pixmap = self.pixmap.scaled(500, 430)  
 self.pixmap = self.pixmap.scaled(self.pixmap.width(), self.pixmap.height())  
 self.label6.setPixmap(self.pixmap)  
 self.label6.adjustSize()  
 self.label6.show()  
########################################################################################################################  
  
  
"""Раскраска меню"""  
qss = """  
QMenuBar {  
 background-color: qlineargradient(x1:0, y1:0, x2:0, y2:1,  
 stop:0 lightgray, stop:1 darkgray);  
}  
QMenuBar::item {   
 padding: 5px 20px;  
 background-color: rgb(300,105,40);  
 color: rgb(255,255,255);   
}  
"""  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 MainWindow = QMainWindow()  
 ui = Ui\_MainWindow()  
 ui.setupUi(MainWindow)  
 app.setStyleSheet(qss)  
 MainWindow.show()  
 app.exec()

# **Приложение 2. Боевая система BattleFunction**

from PyQt5.QtWidgets import QMessageBox  
from random import randint  
import tracker  
  
class Hero:  
 *"""Класс героя"""* Bat\_qer = [7, 8, 9, 10, 11, 12]  
 hp\_hero = 100 # показатель героя  
 szet = 0 # счетчик боев  
  
  
# класс для счетчиков разных переменных  
class peremenn:  
 hp\_kolwo = 1 # счетчик для хилки их количество  
 Money = 1 # переменная для сбора монет  
  
  
class Battle:  
 *"""Функция боя"""* def Battle\_test(self):  
 *"""Функция запускает саму битву"""* Hero.szet += 1 # счетчик боев  
 rand = randint(1, 3) # определяется какой будет враг  
 """Здорвье врага"""  
 hp\_maus = 20  
 hp\_skel = 40  
 hp\_zombe = 30  
 hp\_doss = 50  
 """Урон врага"""  
 Bat\_maus = [1, 2, 3, 4, 5, 6]  
 Bat\_skel = [3, 4, 5, 6, 7, 8]  
 Bat\_zombe = [2, 3, 4, 5, 6, 7]  
 Bat\_doss = [6, 7, 8, 9, 10, 11]  
  
 if Hero.szet < 5: # проверка на колличество боёв  
 if rand == 1:  
 """Вывод текста врага"""  
 tracker.Ui\_MainWindow.Artmaus(self)  
 self.listWidget.addItem('Шагая затемнённому коридору вы услышали шорох.\n'  
 'Огромная, полугнила крыса с визгом наброслась на вас!\n\n')  
 while True:  
 """Проходит симуляция боя"""  
 rand = randint(1, 6)  
 damag\_mous = Bat\_maus[rand - 1]  
 Hero.hp\_hero = Hero.hp\_hero - damag\_mous  
  
 self.listWidget.addItem(  
 'Мышь царапает вас и наносит вам: {} урона у вас осталось {} XP\n'.format(damag\_mous,  
 Hero.hp\_hero))  
 death.death\_test(self) # тест на смерть персонажа  
 damag\_hero = Hero.Bat\_qer[rand - 1] # int(Bat\_qer[rand - 1])  
 hp\_maus = hp\_maus - damag\_hero  
 self.listWidget.addItem('Вы наносите удар и после брызга крови крыса визгнула и зашипела,\n'  
 'она получилп: {} урона\n'.format(damag\_hero))  
 if hp\_maus <= 0: # проверка на здоровье врана  
 peremenn.Money += 10  
 self.listWidget.addItem('Вы убили крысу у вас осталось: {} здоровья\n\n'.format(Hero.hp\_hero))  
 self.listWidget.addItem('Вы убили крысу и получили 10 монет теперь у вас: {} монет\n\n'.format(  
 peremenn.Money))  
 break  
 elif rand == 2:  
 tracker.Ui\_MainWindow.ArtSkeleton(self)  
 self.listWidget.addItem('Зайдя в комнату с каменными гробами, вам путь перегородил высокий скелет в латах!\n\n')  
 while True:  
 rand = randint(1, 6)  
 damag\_skel = Bat\_skel[rand - 1]  
 Hero.hp\_hero = Hero.hp\_hero - damag\_skel  
 self.listWidget.addItem(  
 'Скелет треща своими костями и громыхая своей бронёй\n'  
 'бьет вас ржавой алебардой и наносит вам: {} урона у вас осталось {} здоровья\n'.  
 format(damag\_skel, Hero.hp\_hero))  
 death.death\_test(self)  
 damag\_hero = Hero.Bat\_qer[rand - 1]  
 hp\_skel = hp\_skel - damag\_hero  
 self.listWidget.addItem('Вы ударили своим мечом и скелет затрещав получет: {} урона\n'.format(damag\_hero))  
 if hp\_skel <= 0:  
 peremenn.Money += 30  
 self.listWidget.addItem('Вы убили скелета у вас осталось: {} здоровья\n\n'.format(Hero.hp\_hero))  
 self.listWidget.addItem('Вы убили скелета и получили 30 монет теперь у вас: {} монет\n\n'.format(  
 peremenn.Money))  
 break  
 elif rand == 3:  
 self.listWidget.addItem('Вы зашли в команату заваленную гниющими трупами.\n'  
 'Прочитав короткую молтиву, вы замечаете что один из них зашевелился....\n'  
 'Он посмотрел на вас пустым взгядом и поковылял в вашу сторону\n\n')  
 while True:  
 tracker.Ui\_MainWindow.ArtZombie(self)  
 rand = randint(1, 6)  
 damag\_zombe = Bat\_zombe[rand - 1]  
 Hero.hp\_hero = Hero.hp\_hero - damag\_zombe  
 self.listWidget.addItem(  
 'Зомби бьёт вас обрубком руки и наносит вам: {} урона у вас осталось {} XP\n'.  
 format(damag\_zombe, Hero.hp\_hero))  
 death.death\_test(self)  
 damag\_hero = Hero.Bat\_qer[rand - 1]  
 hp\_zombe = hp\_zombe - damag\_hero  
 self.listWidget.addItem('Вы наносите удар и зомби получет: {} урона\n'.format(damag\_hero))  
 if hp\_zombe <= 0:  
 peremenn.Money += 20  
 self.listWidget.addItem('Вы убили зомби у вас осталось: {} здоровья\n\n'.format(Hero.hp\_hero))  
 self.listWidget.addItem('Вы убили зомби и получили 20 монет теперь у вас: {} монет\n\n'.format(  
 peremenn.Money))  
 break  
 elif Hero.szet == 5:  
 tracker.Ui\_MainWindow.ArtLichboss(self)  
 self.listWidget.addItem('Придя в древний тронный зал, вы видите ужасную фигуру на троне\n'  
 '-Ты уже проиграл, смертный червь, иди же и прими свою смерть!\n\n')  
 while True:  
 rand = randint(1, 6)  
 damag\_doss = Bat\_doss[rand - 1]  
 Hero.hp\_hero = Hero.hp\_hero - damag\_doss  
 self.listWidget.addItem(  
 'Лич прочитал заклинание, его огромный меч с черным как ночь клинком\n'  
 'направился к вам и наносит: {} урона у вас осталось {} XP\n'.format(damag\_doss,Hero.hp\_hero))  
 death.death\_test(self)  
 damag\_hero = Hero.Bat\_qer[rand - 1] # int(Bat\_qer[rand - 1])  
 hp\_doss = hp\_doss - damag\_hero  
 self.listWidget.addItem('С молитвой на устах и бьёте своим светящимся клинком и лич получет: {} урона\n'.  
 format(damag\_hero))  
 if hp\_doss <= 0:  
 self.listWidget.addItem('Вы уничтожили Короля Лича, принцесса спасена!')  
 break  
  
  
  
class death:  
 *"""функция, проверка на показатель хп"""* def death\_test(self):  
 if Hero.hp\_hero <= 0:  
 msg = QMessageBox()  
 msg.setWindowTitle("Вы умерли....")  
 msg.setText("Под напором врага вы не выдержали и пали! Начните сначала.")  
 msg.setIcon(QMessageBox.Warning)  
  
 msg.exec\_()  
 self.listWidget.show()  
 self.close  
 if Hero.szet == 6: # если все битвы пройдены герой победил  
 msg = QMessageBox()  
 msg.setWindowTitle("Вы победили")  
 msg.setText("Король Лич побеждён!")  
 msg.setIcon(QMessageBox.Warning)  
 msg.exec\_()  
 self.listWidget.show()  
 self.close  
  
  
  
class clicked:  
 *"""Класс привязанный к кнопке для запуска функций"""* def clicked\_resum(self):  
 Battle.Battle\_test(self)  
 death.death\_test(self)  
  
# функция зелья здоровья  
class Health:  
  
 def Health\_Potion(self):  
 *"""Функция выстанавливающая здорвье главного героя"""* if peremenn.hp\_kolwo != 0:  
 Hero.hp\_hero = Hero.hp\_hero + 20  
 self.listWidget.addItem('Испив волшебное зелье, вы чувствуете прилив сил.\n'  
 'Вы востановили 20 очков здоровья и ваше здоровье = {} \n\n'.format(Hero.hp\_hero))  
 peremenn.hp\_kolwo -= 1

**Приложение 3. Система торговли Traderf**

import BattleFunction  
import tracker  
  
class Trad:  
 *"""Класс торговца"""* def Trader(self):  
 tracker.Ui\_MainWindow.ArtTrader(self)  
 """Вывод текста торговца"""  
 self.listWidget.addItem('Странствующий торговец\n')  
 self.listWidget.addItem(  
 'Вы неожиданно встречаете странствующего торговца,\n на его спине висит огромный рюкзак. '  
 'Он радостно приветствует вас.\n'  
 'Вы спрашиваете его как он здесь оказался.\n На все ваши вопросы он либо отнекивается, '  
 'либо уклончиво отвечает.')  
 self.listWidget.addItem(  
 '-Сэр рыцарь, предлагаю перейти к делу, у меня вы можете приобрести:\n'  
 'Зелье здоровья за 30 монет,\n а так же я могу заточить ваш клинок'  
 '(повышение урона на 3 единицы) за 100 монет\n\n')  
  
 def Health\_Po(self):  
 *"""Покупка зелий здоровья у торговца, если не хватает денег, об этом сообщается"""* if BattleFunction.peremenn.Money < 30:  
 self.listWidget.addItem('У вас недостаточно денег их количество = {} \n\n'.  
 format(BattleFunction.peremenn.Money))  
 elif BattleFunction.peremenn.Money >= 30:  
 BattleFunction.peremenn.Money = BattleFunction.peremenn.Money - 30  
 BattleFunction.peremenn.hp\_kolwo += 1  
 self.listWidget.addItem(' Вы приобрели зелье здоровья. Кол-во зелий = {}.Кол-во денег = {} \n\n'.format(  
 BattleFunction.peremenn.hp\_kolwo, BattleFunction.peremenn.Money))  
  
 def Bat(self):  
 *"""Заточка оружия у торговца, если не хватает денег, об этом сообщается"""* if BattleFunction.peremenn.Money < 100:  
 self.listWidget.addItem('У вас недостаточно денег их количество = {} \n\n'.  
 format(BattleFunction.peremenn.Money))  
 elif BattleFunction.peremenn.Money >= 100:  
 BattleFunction.peremenn.Money = BattleFunction.peremenn.Money - 100  
 for i in range(len(BattleFunction.Hero.Bat\_qer)):  
 BattleFunction.Hero.Bat\_qer[i] += 3  
 self.listWidget.addItem('Торговец весело напевая себе под нос заточил вас клинок\n\n')