**Министерство образования и науки Российской Федерации**

Федеральное государственное автономное образовательное учреждение

высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники

Направление: 09.03.02 «Информационные системы и технологии»

Отделение «Информационных технологий»

**Игровой бот с алгоритмическим ИИ**

Отчет по дисциплине

Учебно-исследовательская работа студентов

Выполнили студенты гр. 8И6А \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Голушков А.Н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Видякин Г.И.

(Подпись) (Ф.И.О.)

Дата сдачи отчета преподавателю \_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

Руководитель Ассистент ОИТ Мыцко Е.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Оценка руководителя) (Подпись)

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_г.

(Дата проверки)

Учебно-исследовательскую работу студенты защитили с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Члены комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_г.

(дата защиты)

Содержание

[Введение 3](#_Toc533426223)

[Цель работы 4](#_Toc533426224)

[1. Задачи 4](#_Toc533426225)

[2. Выбор языка для реализации приложения 4](#_Toc533426226)

[3. Изучение API приложения Telegram 4](#_Toc533426227)

[4. Инициализация бота в приложении Telegram 5](#_Toc533426228)

[5. Разработка бота 5](#_Toc533426229)

[5.1. Характеристики персонажа 5](#_Toc533426230)

[5.2. Предметы в игре 6](#_Toc533426231)

[5.3. Магазин алхимика 8](#_Toc533426232)

[5.4. Открытие двери 9](#_Toc533426233)

[5.5. Место молитвы 13](#_Toc533426234)

[5.6. Таблица рекордов 13](#_Toc533426235)

[5.7. Алгоритмический ИИ 14](#_Toc533426236)

[5.8. Принятие базовых решений 15](#_Toc533426237)

[5.9. Принятие сложных решений 16](#_Toc533426238)

[Заключение 19](#_Toc533426239)

[Список литературы 21](#_Toc533426240)

# Введение

Текстовый квест — это разновидность [компьютерной игр](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_%D0%B8%D0%B3%D1%80%D0%B0)ы, в которой взаимодействие с игроком осуществляется посредством текстовой информации. Существуют два вида интерфейса:

* интерфейс с вводом текста с клавиатуры;
* интерфейс в виде меню, где игрок выбирает действие из нескольких предложенных

Было решено сделать текстовую игру с помощью бота Telegram.

На первый взгляд, Telegram — это просто очередной мессенджер. Реклама гласит, что он быстр, защищен, в нем нет рекламы. Но самая важная для нас его функция — боты!

Бот — это не просто «автоответчик». Его правильнее считать автоматизированным помощником. Пользователи могут взаимодействовать с ботами при помощи сообщений, отправляемых через обычные или групповые чаты.

Telegram позволяет пользователям создавать своих ботов.

Логика бота контролируется при помощи HTTPS запросов к [API для ботов](https://tlgrm.ru/docs/bots/api).

В текстовой игре должен присутствовать алгоритмический искусственный интеллект.

# Цель работы

Целью данного проекта является разработка игрового Telegram бота с алгоритмическим ИИ на языке программирования Python.

# Задачи

* + - 1. Изучение API Telegram;
      2. Изучение Python;
      3. Создание бота;
      4. Изучение алгоритмического ИИ.

# Выбор языка для реализации приложения

Перед тем, как приступить к работе над кодом программы перед нами стал выбор средств программирования, в которые входят язык программирования, платформа, среда программирования.

Для разработки программы был выбран язык программирования Python. Python - очень простой язык программирования, он имеет лаконичный и в то же время довольно простой и понятный синтаксис. Данный язык обладает автоматическим управлением памяти и динамической типизацией. Python позволяет стартовать разработку быстрее, а это позволяет быстрее получить прототип решения. Скорость разработки на Python на начальных этапах проекта значительно выше по сравнению, например, с С++. Таким образом, в коротких малобюджетных проектах Phyton будет иметь преимущество по скорости разработки. К тому же, большинство документации по API Telegram создано именно для языка Python. Это и было ключевым моментом в выборе языка программирования.

# Изучение API приложения Telegram

Telegram предоставляет документацию по-своему API на своем сайте в свободном доступе. Для создания программы-бота необходимо подключить пакет “Telegram Bot API” в среде разработки с помощью консоли. Из документации известно, что Telegram API присылает приложению-боту массив в формате JSON, который содержит следующую информацию:

* Id чата
* Id пользователя
* Текст сообщения
* Прикрепленные фотографии или файлы
* Местоположение пользователя
* Карточка-контакт из его телефона

# Инициализация бота в приложении Telegram

Прежде чем начинать разработку, бота необходимо зарегистрировать и получить его уникальный id, являющийся одновременно и токеном. Для этого в Telegram существует специальный бот — @BotFather. В беседе с этим ботом была отправлена команда /newbot, и задано имя бота – @UirsBot. После выполнения этих шагов был получен токен нашего бота.

# Разработка бота

Основная идея игры заключается в том, что игрок находится в подземелье, выход из которого нужно отыскать. Перед игроком огромное количество дверей. За дверью может оказать подземелье с драконом, чубакой или другим монстром, хотя с такой же вероятностью за дверью может быть и озеро мороженого.

## Характеристики персонажа

У персонажа имеются следующие характеристики(рис.1 ):



Рисунок 1 - характеристики игрока

Это характеристики, доступные игроку. Здесь “Сила” - это характеристика, значение которого наносит урон персонажу; “Защита” - сколько урона будет игнорировано при получении урона; “Харизма” - необходима для успешного прохождения некоторых уровней (может быть повышена, например, солнцезащитными очками); “Интеллект” - характеристика, аналогичная “Харизме”, от которой зависит прохождение того или иного уровня. Так же Интеллект влияет на возможность надеть ту или иную вещь. “Магический урон” - отвечает за урон от “Магического удара” (также повышается некоторыми предметами); “HP” - уровень здоровья, который снижается с получением урона (его можно восстанавливать, съев, например, яблоко или выпив эликсир). При получении такого урона, что “HP” < 0, игра начинается сначала. “MP” - количество маны, расходуется на магический удар. “Gold” - внутриигровая валюта, которую можно тратить на различные предметы в магазине или у некоторых персонажей в процессе игры.

## Предметы в игре

Предметы в игре играют ключевую роль. Без них невозможно пройти игру целиком. Они могут как улучшать, так и ухудшать характеристики героя. При наличии или отсутствии определенных предметов сюжет игры может принимать различные обороты. Каждый предмет имеет свою стоимость. Предметы можно получать различными способами: за одобрение или гнев богов, купить в магазине алхимика или за успешное прохождение комнат.

Например, положительный предмет “Шапка из фольги” увеличивает защиту игрока на 20 единиц. А вот специальный предмет “Магический свиток” (рис.2 ) позволяет во время боя с вероятностью 0,75 превратить противника в лягушку, если конечно у вас хватит ресурса “Мана”.

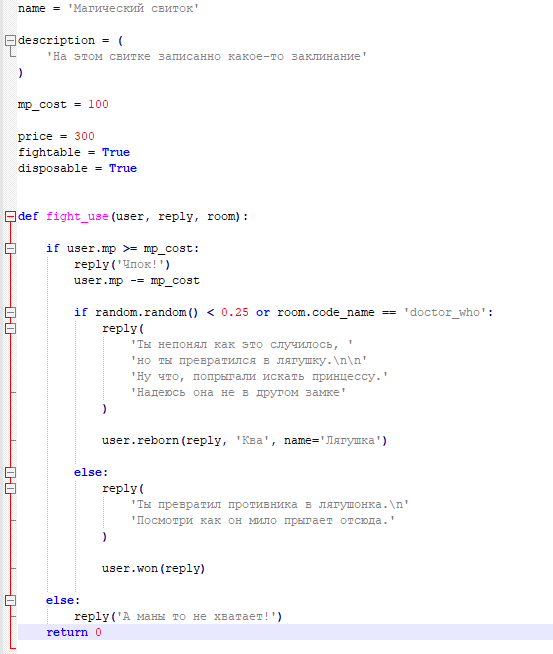


Рисунок 2 - Исходный код предмета Магический свиток

## Магазин алхимика

В игре имеется магазин алхимика. Он может предложить всегда три различных предмета. Никто не знает, помогут ли игроку эти предметы или деньги будут потрачены впустую. Некоторые предметы могут даже убить игрока. Рассмотрим пример:

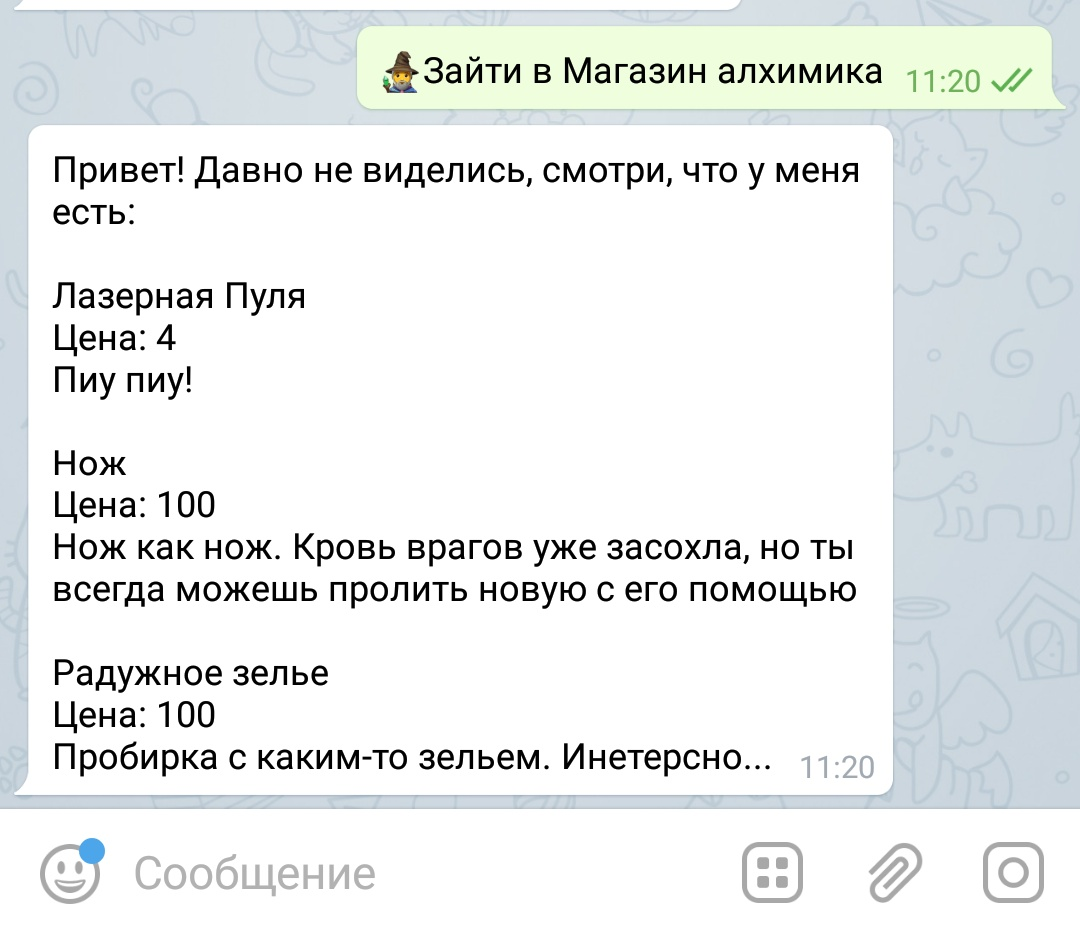


Рисунок 3 - Магазин алхимика

В данном случае, пулю целесообразно покупать, если в инвентаре имеется пистолет. Нож добавит к характеристике “Сила” некоторое количество единиц. Зелья в игре не соответствуют логике “яркий цвет - хороший эффект”, эффект может быть, как и положительным, так и отрицательным. Например, “Радужное зелье” превратит игрока в крысу, а “Черное зелье” восстановит здоровье.

## Открытие двери

При открытии очередной двери игрок встречается с одним из персонажей и начинается диалог рассказчика с игроком, например, «Вы открываете дверь, а за ней…<monsterName>, Ваши действия?». Далее, игрок выбирает что сделать дальше из списка вариантов. Данный список зависит от типа двери, в которую заходит игрок (под типом двери подразумевается тип персонажа, который прячется за дверью). Персонажи могут быть сюжетные, нейтральные, положительные и отрицательные.

При прохождении сюжетных персонажей, они могут давать полезные подсказки или просто пропустить дальше при выполнении определенных действий. Всего сюжетных комнаты 3.

Например, первая сюжетная комната (рис.4) - это “Старик”.



Рисунок 4 - Исходный код первой сюжетной комнаты

При входе в эту комнату (функция enter()) игрок получает пистолет, пулю, очки и карту, которые помогут игроку в дальнейшем. С ним не надо вступать в бой, есть лишь один вариант развития событий - уйти с полученными вещами.

Нейтральные персонажи в зависимости от диалога игрока с этими персонажами могут реагировать по-разному: нанести урон, дать полезный предмет или ничего не сделать.

Например, комната “Отдел кадров” (рис. 5). Эта комната ничего не делает, она создана исключительно целях повышения эмоционального фона игрока и увеличения мотивации играть в игру дальше.

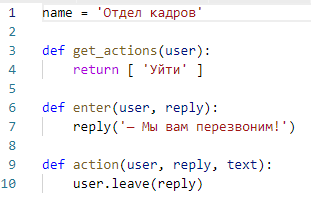


Рисунок 5 - Исходный код комнаты Отдел кадров

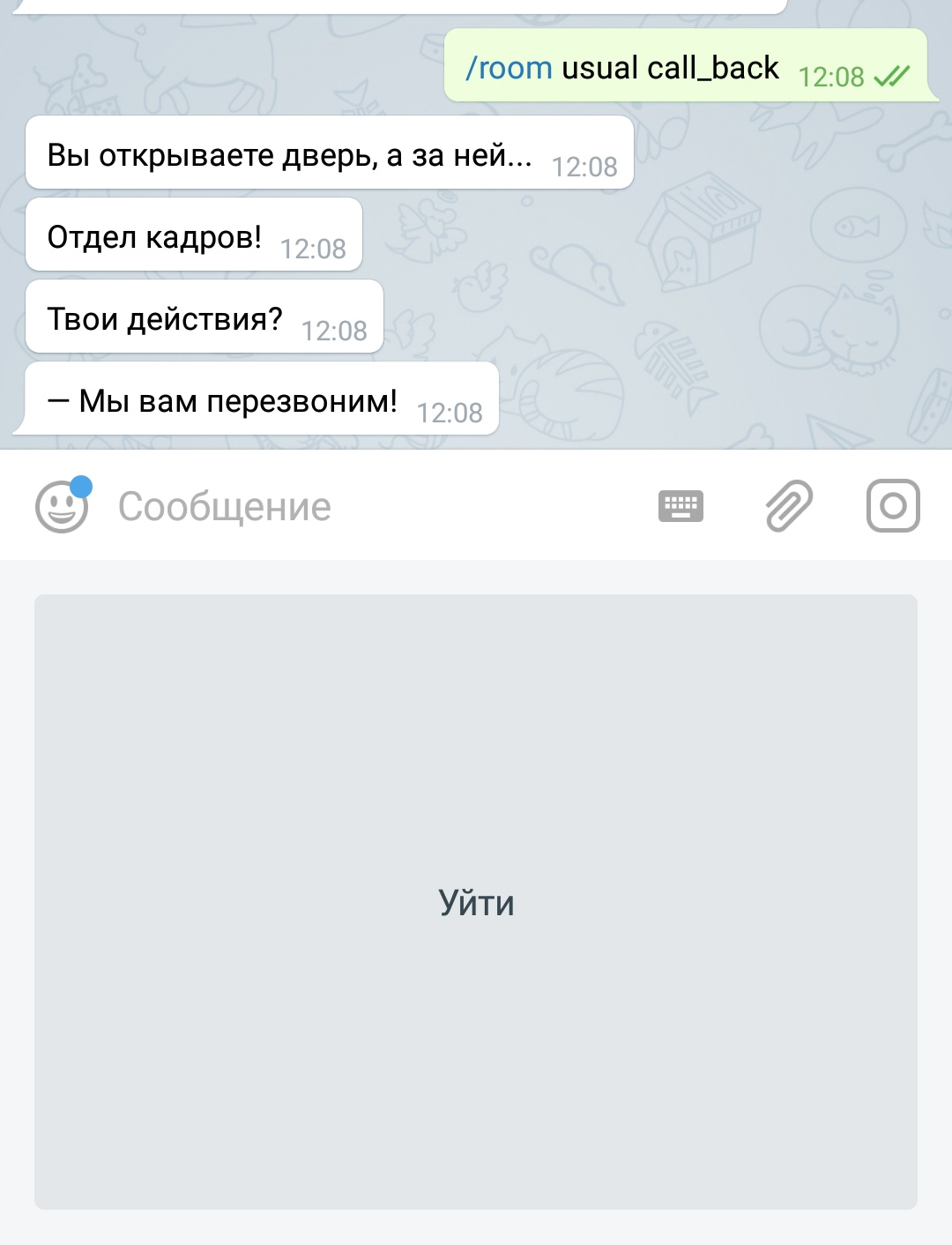


Рисунок - 6

Следующий код описывает работу комнаты с сундуком (рис. 7). Игрок может либо открыть сундук, либо уйти. При открытии сундука игроку может выпасть совершенно случайная вещь. С вероятность 0.33 в сундуке не будет ничего, с вероятностью 0.33 в сундуке окажется немного монет. И с вероятностью 0.34 в сундуке будет лежать предмет (положительный, отрицательный, или нейтральный).

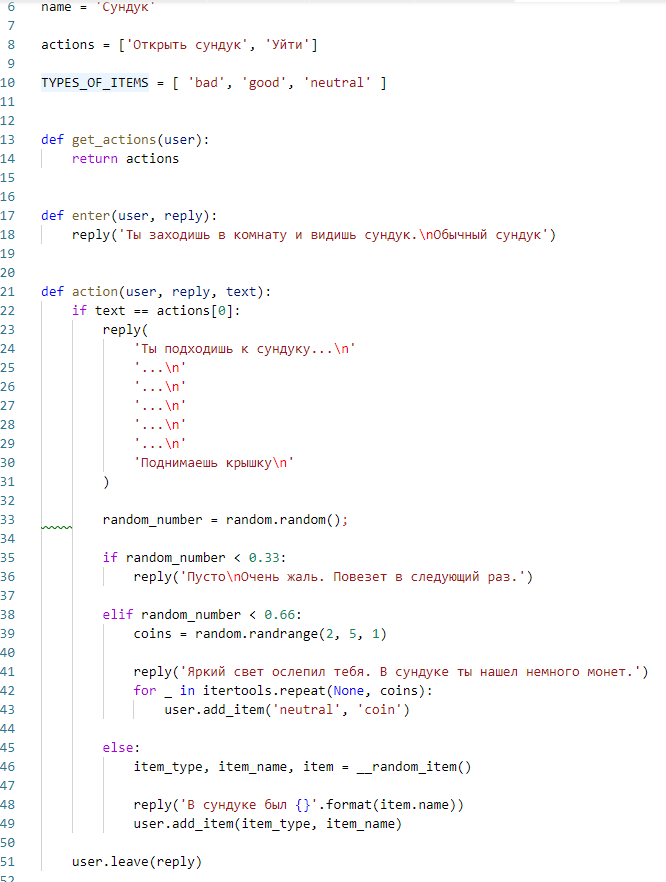


Рисунок 7- Исходный код комнаты “Сундук”

Отрицательные персонажи в любом случае попытаются атаковать игрока, однако, нанесенный урон будет так же зависеть от характеристик игрока и от его действий.

Например, “Рыцарь” (рис.8). Рыцарь является единственным персонажем, на котором может закончится игра выигрышем игрока. Для того чтобы можно было с ним сражаться, необходимо пройти все 3 сюжетные комнаты. Рыцарь может нанести урон от 30 до 40 единиц и имеет броню 170 единиц.

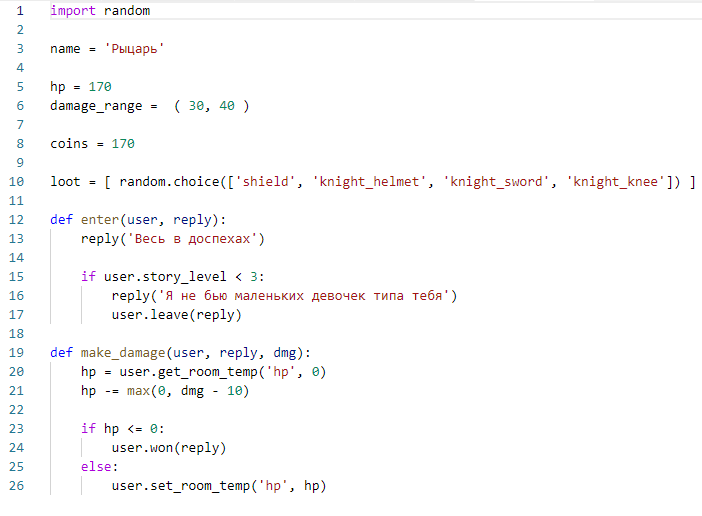


Рисунок 8 - исходный код комнаты “Рыцарь”

Поведение персонажей определяется искусственным интеллектом.

## Место молитвы

В игре существует специальная локация, где игроку дается шанс помолиться одному из нескольких присутствующих богов. Сейчас в игре их 4: Будда, Иисус, Аллах, Рассказчик.

При мольбе каждому из богов повышается уровень доверия бога к нашему персонажу. При достижении определенного уровня бог дарит игроку либо уникальную способность, либо дает предмет с хорошими характеристиками, либо может полностью восстановить его здоровье.

Но каждый из богов не одобряет, когда игрок молится другим богам. Если пользователь начал молиться одному из богов, то при переходе к другому богу, первый бог его накажет. Это опять-таки может быть предмет, но уже с отрицательными характеристиками, от которого игрок не может избавиться или же может быть разовое ухудшение характеристик.

При прохождении некоторых комнат уровень взаимодействия с богом может влиять на сюжет, как в положительную, так и в отрицательную сторону.

## Таблица рекордов

При вызове команды /leaderboard происходит отображение текущей таблицы рекордов (рис. 9). База данных таблицы рекордов хранится в формате json. Для работы с базой был создан модуль databasemanager.py, который содержит функции для добавления и получения записей из таблицы. На данный момент таблица рекордов строится только по количеству пройденных комнат игроком, но в дальнейшем будет добавлено еще несколько параметров, по которым будут строиться таблицы рекордов.

При получении от пользователя команды происходит вызов функции leaderboard в модуле main.py. Там формируется сообщение при помощи вызова функции get\_leaderboard в модуле databasemanager.py, которая возвращает кортежи таблицы.

При смерти игрока происходит вызов функции update\_leaderbord в модуле user.py, которая добавляет в таблицу новый кортеж данных.

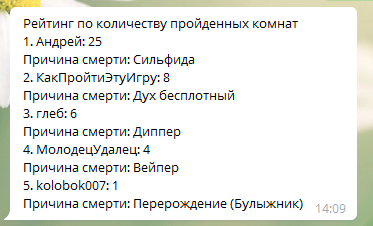


Рисунок 9 - Отображение рейтинга игроков

# Алгоритмический ИИ

Игровой ИИ сосредоточен на том, какие действия должен выполнять объект, исходя из условий, в которых находится. Обычно это называют управлением «интеллектуальными агентами», где агент является игровым персонажем, транспортным средством, ботом, а иногда и чем-то более абстрактным: целой группой сущностей или даже цивилизацией. В каждом случае это вещь, которая должна видеть свое окружение, принимать на его основе решения и действовать в соответствии с ними. Это называется циклом Sense/Think/Act (Чувствовать/Мыслить/Действовать):

Восприятие среды в текстовой игре – это ответы игрока и его характеристики, в зависимости от передаваемых алгоритму параметров он должен сделать цепочку решений и выдать ответ. Так на рисунке 10 указан перечень параметров, которые может оценивать игра. Это и предыдущая комната, и текущее HP игрока, и уровень доверия богов, и состояние инвентаря героя.

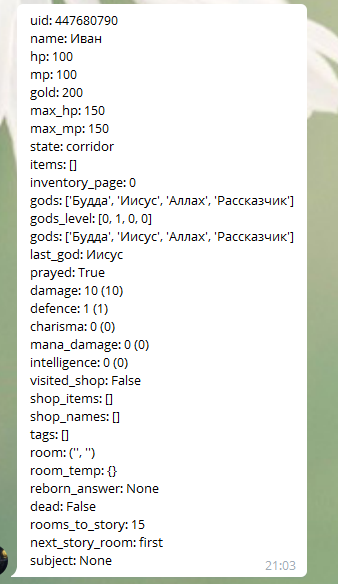


Рисунок 10 - параметры восприятия среды

# Принятие базовых решений

Рассмотрим комнату, полностью наполненную водой. Пользователю при входе предлагается сделать выбор доплыть до сундука или закрыть дверь и уйти. При этом стартует таймер времени. Если пользователь принимает решение быстрее, чем некая константа (5 секунд), то алгоритм идет дальше, предлагаю кинуть кость игроку. Если выпадает значение превышающее среднее значение возможного выпадения очков, то игрок доплывает до сундука и достает из него полезный предмет. Если это значение меньше среднего, то вода смывает игрока в коридор и наносит ему некоторый урон. То есть состояние игрока и решение ИИ зависит от двух факторов: скорости реакции и эффекта случайности. Была создана блок-схема дерева решений для алгоритма данной комнаты (рис. 11).

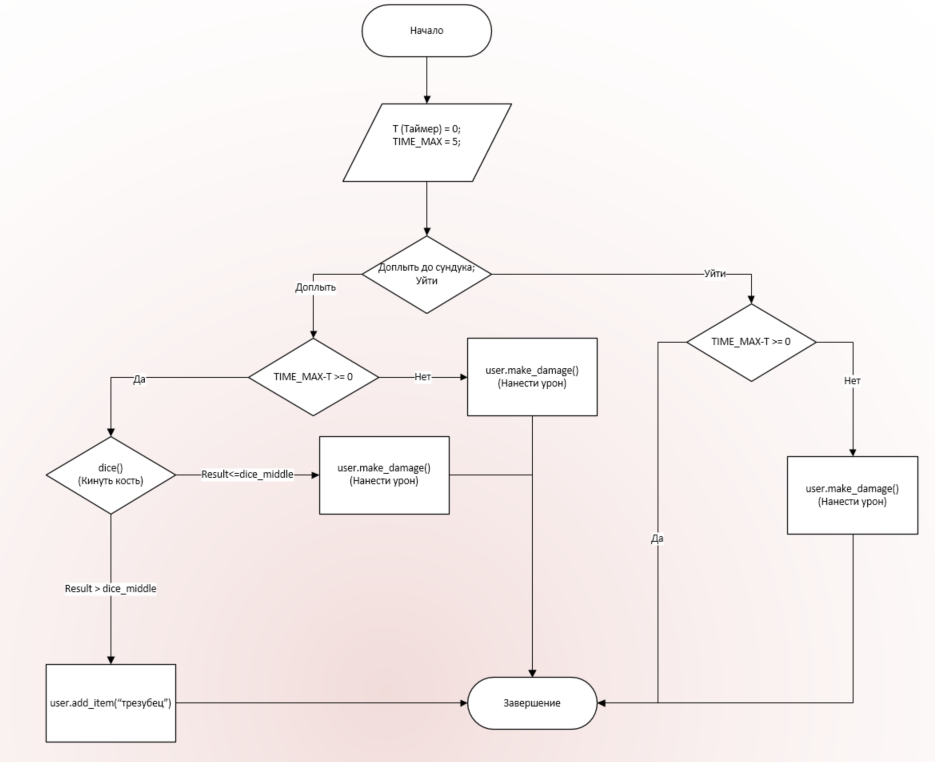


Рисунок 11 - Дерево решений

# Принятие сложных решений

Хоть простые системы реакций очень действенны, бывает много ситуаций, когда их недостаточно. Иногда нужно принимать различные решения, основанные на том, что агент делает в настоящий момент, но представлять это за условие тяжело. Иногда существует слишком много условий, чтобы эффективно представить их в дереве решений или скрипте. Иногда нужно заранее оценивать, как изменится ситуация, прежде чем принимать решение о следующем шаге. Для решения этих проблем нужны более сложные подходы.

Для примера возьмем стража с такими состояниями:

* Патрулирующий (Patrolling).
* Атакующий (Attacking).
* Убегающий (Fleeing).

И такими условиями для изменения его состояния:

* Если страж видит противника, он атакует.
* Если страж атакует, но больше не видит противника, он возвращается к патрулированию.
* Если страж атакует, но сильно ранен, он убегает.

Также можно написать if-операторы с переменной-состоянием стража и различные проверки: есть ли поблизости враг, какой уровень здоровья NPC и т. д. Добавим еще несколько состояний:

* Бездействие (Idling) — между патрулями.
* Поиск (Searching) — когда замеченный враг скрылся.
* Просить о помощи (Finding Help) — когда враг замечен, но слишком силен, чтобы сражаться с ним в одиночку.

Выбор для каждого из них ограничен — например, страж не пойдет искать скрывшегося врага, если у него низкое здоровье.

В конце концов огромный список «если <x и y, но не z>, то <p>», может стать слишком громоздким, поэтому следует формализовать метод, который позволит нам держать в уме состояния и переходы между состояниями. Чтобы это сделать, примем во внимание все состояния, и под каждым состоянием запишем в список все переходы в другие состояния, вместе с необходимыми для них условиями.

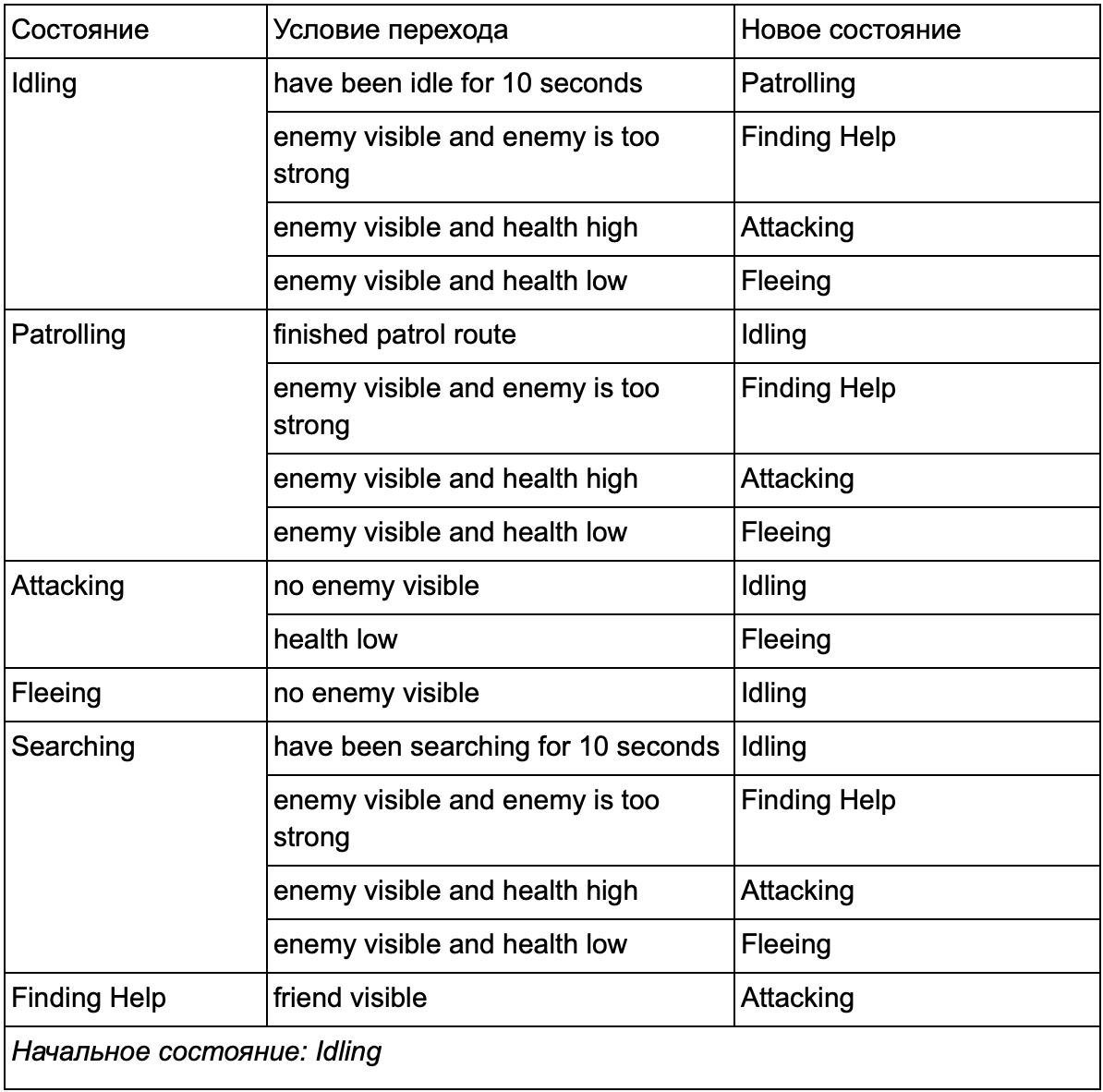


Рисунок 12 - Таблица состояний

Нарисуем диаграмму и получим полный обзор того, как меняется поведение NPC.

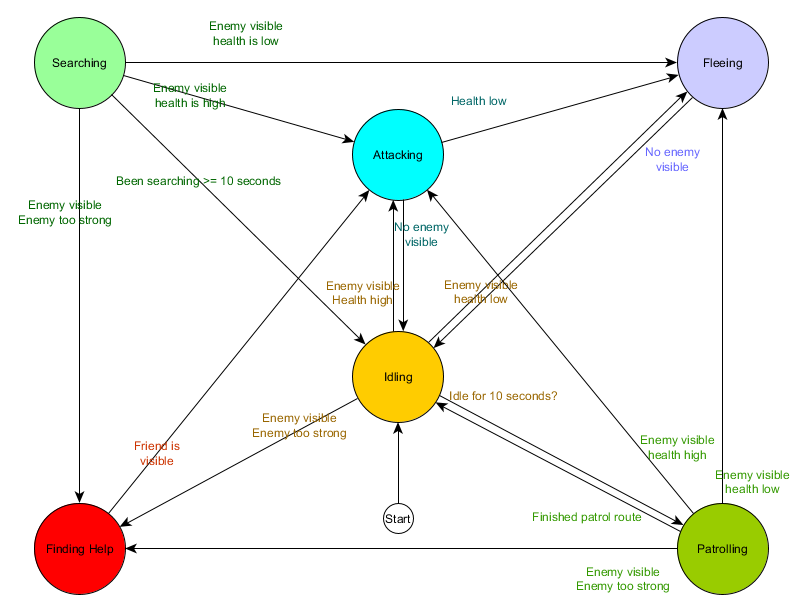


Рисунок 13 - Диаграмма состояний

Диаграмма отражает суть принятия решений для этого агента на основе текущей ситуации. Причем каждая стрелка показывает переход между состояниями, если условие рядом с ней истинно.

Заключение

Результатом учебно-исследовательской работы является Телеграм бот с применением простейшего алгоритмического игрового ии. Для достижения этой цели были изучены основы языка Python, принципы работы API, непосредственно сам Telegram API, устройство и система работы ботов, их регистрация и настройка, а также запуск собственного сервера. В ходе работы был изучен принцип построения игрового ИИ для принятия сложных решений. Планируется продолжать работу над приложением, внедряя изученный ИИ для создания более сложных игровых персонажей.

Список литературы