Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет Школа разработки видеоигр

Образовательная программа Технологии разработки компьютерных игр

Направление подготовки (специальность) Прикладная информатика

О Т Ч Е Т

о преддипломной практике

Тема задания: Разработка многоканальной диалоговой системы для группы неигровых персонажей с возможностью прерывания и переключения между участниками диалога без потери информативности

Обучающийся Иванов Вадим Вячеславович, J4221

Согласовано:
Руководитель практики от университета: Карсаков Андрей Сергеевич, доцент факультета цифровых трансформаций

Практика пройдена с оценкой _____

Дата _____

Санкт-Петербург
2024

СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

RPG (Role-playing game) – жанр компьютерных игр, берущий за основу элементы игрового процесса традиционные для настольных ролевых игр. В играх этого жанра игроку предоставляется возможность отыгрывать одного или нескольких персонажей с большей свободой действий, чем в других жанрах. Игрок может определять, как развивать управляемых им персонажей в рамках заданной игровыми дизайнерами системе.

NPC (Non player character) – неигровой персонаж в компьютерной игре.

Нейронная сеть – алгоритм, разновидность машинного обучения, заточенный под выполнения конкретных задач.

ВВЕДЕНИЕ

Фундаментальная проблема игр, в которых происходит взаимодействие с окружающим миром, заключается в том, что игрок теряет погружение в игровой мир, когда игрок бездействует. Мир и NPC живут и работают только тогда, когда игрок с ними взаимодействует. Если игрок прекращает взаимодействие, то мир буквально замирает. Многофункциональные диалоги могут помочь несколько оживить игровой мир. Чем больше проработано игровое окружение, тем больше игрок погружается в него и начинает больше верить в происходящее [1]. Факт искусственности данного мира становится очевиден в те моменты, когда какой-то элемент симуляции работает не так, как должен, что по итогу выбивает из погружения. Взаимодействие игрока с неигровыми персонажами как раз один из таких элементов. Зачастую диалог с NPC представляет собой обычный набор информации для какого-либо задания в игре, по окончания которого неигровой персонаж перестает представлять для игрока какой-либо интерес в силу отсутствия возможности дать новую информацию. Особенно видно несовершенство диалога в играх в том случае, когда в процессе общения участвует больше двух персонажей. В играх, когда игрок с кем-то начинает разговор, остальные персонажи тихо ждут своей очереди в диалоге, пока общение с первым NPC не закончится. Мир будто замирает (в некоторых играх, таких как "Pokemon" и "The Legend of Zelda" он буквально "замирает" – все персонажи перестают передвигаться и все точки взаимодействия становятся неактивными), и никто не будет влезать в разговор с игроком, пока тот сам не прекратит общение, и так происходит абсолютно с каждым персонажем. В связи с этим отсутствует возможность появления настоящего живого общения в группе людей, что моментально выбивает из погружения.

Зачем вообще пытаться вовлечь игрока в происходящие в игре события? Согласно исследованию университета Иордании науки и технологии [2], игры способны влиять на наше поведение и в должной мере заставляют испытывать нас как негативные, так и позитивные эмоции в зависимости от разных

условий. В первую с это все нужно, чтобы развлечь игрока и заставить его отвлечься от рутины и проблем, с которыми он сталкивается в обычной жизни. Погружение часто рассматривают, как ключевой показатель в получении удовольствия от игры, вовлеченность — это результат хорошего игрового опыта [1]. Чем больше различных элементов игры проработано и сделано правдоподобно, тем сильнее эффект погружения усиливается. Взаимодействие с неигровыми персонажами (NPC) является как раз одним из таких элементов.

Многие разработчики игр пытались внедрить различные системы взаимодействия с персонажами, прибегая к разным реализациям, например, "Radiant AI" или "Nemesis", но они в первую очередь затрагивают непосредственно индивидуальное общение игрок с одним персонажем, не затрагивая при этом общение с группой NPC, что является неотъемлемой частью повседневного общения в обычной жизни.

Целью данной работы является создание системы многоканального взаимодействия с группой NPC с возможностью в любой момент начать общение с каждым из участников компании без потери информации с другими. Для осуществления данной цели следует выполнить следующие задачи:

1) Определить необходимость исследования.

2) Проанализировать уже готовые диалоговые системы из выпушенных игр.

3) Проанализировать работы, посвященные данной тематике и выявить плюс и минусы представленных реализаций.

4) Выявить какие системы можно использовать, как основу для дальнейшей реализации своего решения.

5) Реализовать систему на основе ранее проведенных исследований.

6) Провести тестирование для выявления проблем и недостатков реализации и устранить недочеты.

Эта система поможет решить проблему погружения игрока в игровой

процесс, поскольку в играх практически не представлено общение групповое общение, а оно является неотъемлемой частью повседневной жизни. Реализация такого взаимодействия с персонажами поможет сократить количество игровых условностей, которые часто выбивают из погружения.

В данном разделе будет описан процесс доработки ранее созданной системы взаимодействия персонажей. А, так же, будут проведены тестирования работоспособности модели для поиска уязвимых мест.

1 ОБЗОР

1.1 Необходимость исследования

В 2017-ом году участники ежегодной конференции "Project Horseshoe", посвященной играм и геймдизайну, размышляли на тему видеоигровых диалоговых систем и рассматривали с критической стороны одни из наиболее популярных реализаций [3]. На конференции они старались разобрать взаимодействие с NPC по частям с целью найти неудачные или устаревшие аспекты и в последствии предоставить свои варианты того, как стоило бы улучшить диалоговые системы. В своих рассуждениях многие участники сходились на том, что диалоговые деревья – одна из наиболее широких реализаций диалогов, имеет много недостатков и ограничений во взаимодействии: "…в такой системе у NPC нет никакой возможности повлиять никак на ситуацию: они отвечают именно так, как их запрограммировали". В такой реализации практически невозможно создать ощущение настоящего диалога между игроком и NPC. Неигровой персонаж скорее выступает в роли таблички, на которой написано, что игроку нужно делать. Такая реализация диалоговых ветвей вполне работала, когда RPG зарождались, и не было достаточно ресурсов и технологий, чтобы в полной мере сделать общение с персонажами. Однако в наше время уже есть возможности расширить роль собеседника от обычного набора информации для продвижения по сюжету до полноценной личности, с которой можно общаться на разные темы. Но насколько это нужно? Есть ли в целом необходимость в развитии взаимодействия с NPC? Студенты из Сасакачеванского университета из Канады Мартин Дешант, Робин Велшч, Джулиан Фроммел и Реган Мындрык [4] провели исследование с помощью которого выяснили, что негативные эмоции, которые выражает NPC по отношению к игроку, непосредственно влияют на самого игрока и на его эмоционально состояние – чем неигровой персонаж более ярко и живо выражает свои эмоции, тем больший эффект это накладывает на игрока. Исследователи создали игровой протитип, в котором пользователь общался с

разного рода персонажами, которые относились к игроку либо нейтрально, либо негативно. После чего исследователи проводили анализ того, как участники эксперимента ведут себя с разными персонажами и записывали свои результаты.

Различные анализы и рассуждения показывают, что NPC оказывают эмоциональное влияние на игрока во время их общения, вызывая у того положительные или негативные эмоции, что в конечном итоге приводит к тому, что игрок больше погружается в мир игры. За время развития индустрии разные разработчики уже пытались создать различные вариации продвинутого общения с NPC, поэтому в дальнейшем будут разобраны некоторые из этих реализаций.

Ниже, в описаны разные проблемы, с которыми сталкиваются игроки в реализованных диалоговых системах, и способы решения данных недостатков и недочетов. Анализ примеров из игр лучше поможет определить проблемы с выбиванием из погружения.

Игроки сталкиваются с различными проблемами, которые мешают погружению в уже реализованных диалоговых системах, ниже будут описаны эти недостатки и решения, которые предлагают разные проекты.

Общение только с одним персонажем в одном диалоге без возможности игрока или постороннего NPC присоединиться к нему. Игра "Façade" решает эту проблему через возможность игрока вклиниться в диалог и вести общение параллельно с двумя собеседниками. Проблема заключается в том, что игрок может произнести абсолютно любую фразу, многие из которых ломают поведение ИИ. Решить же этот недостаток можно с помощью ограничения свободы игрока выбором заранее заготовленных реплик. Игры серии "The Sims" решают эту проблему общением сразу с несколькими персонажами. Недостатком этого решения является то, что все диалоги представлены в игре абстрактно и не могут в полной мере передать суть. Исправить это можно, сделав игрока непосредственным участником событий.

Пошаговые диалоги – общение с персонажем выглядит, как бой в старых

RPG или настольных играх вроде шахмат, где игрок и соперник атакуют по очереди. В игре "Façade" персонажи время от времени могут перебить друг друга, игрок в том числе, стараясь сменить ход диалога. Проблема заключается в том, что перебивания со стороны игрока по большей части являются иллюзией. Поскольку перебивание в неподходящий момент заставят остальных участников замолчать на пару секунд, после чего, не зависимо от реакции со стороны игрока, диалог вернется либо в начало обсуждения, либо перейдет к следующей теме. Таким образом общение является ничем иным, как иллюзией выбора. Решить это можно путем создания такой системы, где перебивания диалога игроком на разных этапах общения приводили бы к разным ответам, таким образом создавая новое русло общения, отличное от первоначального. В играх серии "The walking dead" общение с персонажами идет в реальном времени, игрок должен вовремя реагировать на действия вокруг одним из нескольких вариантов ответа. Он может промолчать, что приведет к альтернативному развитию диалогу. Но при этом, большая часть выборов в игре является иллюзией, которые приводят к одной-двумя дополнительным репликам, после чего развитие диалога возвращается в привычное русло. Решить это можно через создание систему, где молчание и разные выборы в диалогах ведут к новым темам в общении и уникальным ситуациям.

Эмоциональная не вовлеченность пользователя, игрок пропускает диалоги в виду незаинтересованности. В играх "Mass Effect" диалоговые выборы игрока влияют на отношение персонажей к нему и отражаются на игровом процессе, что повышает интерес игрока к происходящему. Основная проблема сводится к тому, что игрок выбирает один из трех вариантов ответа: "хороший", "нейтральный", "злой", что может наскучить при слишком частой репрезентации. При этом не всегда понятно, какой из выборов ответа к какому. Исправить ситуацию можно, добавив больше вариантов ответов, основанных на отношении персонажа к игроку и информации, полученной ранее в ходе других диалогов.

Таким образом, в классической системе ведения общения с персонажами существует много проблем с погружением и вовлечением игрока в происходящие события. Целью данной работы является решение этих проблем, путем создания новой системы взаимодействия с NPC, с учетом выше перечисленных ситуациях. Представленная в работе система будет в первую очередь решать проблему вовлеченности игрока в происходящие события, путем усовершенствования непосредственно самого опыта общения с использованием возможностей вести диалог с несколькими персонажами одновременно. В разрабатываемой системе будет опция перебивания диалога и возможности вести его в другое русло независимо от момента перебивания, используя заранее заготовленные фразы и информацию, которая была получена в ходе общения.

В играх диалоги создаются в первую очередь для игрока и его продвижению по сюжету, от чего внимание NPC переключается на игрока в момент его появления, не зависимо от того, чем тот занимался. Например, если два персонажа вели диалог и игрок взаимодействовал с одним из них, то второй будет ждать окончания диалога игрока с NPC, что не похоже на настоящее общение. Невозможность посторонних собеседников реагировать на слова игрока или участвующего в диалоге NPC разрушает погружение в игровой мир. Разрабатываемая в данной работе система будет представлять собой решение выше представленных проблем, где можно было бы общаться с несколькими персонажами в рамках одной диалоговой цепи с возможностью прерывания и переключения на любого NPC в группе в реальном времени без потери информации от других и выбора ответов, влияющих на дальнейшее общение.

1.2 Анализ существующих решений в играх

Существует несколько систем, пытающихся усовершенствовать поведение NPC, чтобы те чувствовались живее. В данной работе будут рассмотрены эти реализации с целью выявления наиболее близкой системы к тематике работы, чтобы в дальнейшем использовать полученные данные для

реализации своего решения.

1.2.1 Middle-earth: Shadow of war – Nemesis

В системе "Nemesis" из "Shadow of war" игра запоминает действия игрока, такие как побег от врага, после чего при следующей встрече с ним, орк прокомментирует действие игрока. Или же, если игрок убивает важного для конкретного орка NPC, то он так же прокомментирует данное действие при следующей встрече. Для формирования орков и их отличительных реакций на действия игрока, используется продвинутая система процедурной генерации: определяются характеристики орка и под них уже подстраивается внешний вид персонажа.

Схожую систему взаимодействия можно использовать и в реализации общения с NPC, адаптируя все основные концепции и механики под диалоги. В активном диалоге с двумя или более NPC можно заранее выстроить взаимоотношение персонажей друг с другом. Например, игрок общается с группой из трех персонажей, где первый недолюбливает второго, второй хорошо относится к третьему, а третий не доверяет первому. Игрок общается с первым NPC, в то время как двое других слышат, диалог и в случае, если игрок выстраивает негативные отношения с первым, то у третьего повышаются очки доверия к игроку и это дает новые ветки в общении с ним.

Такой пример работы мультиканальных диалогов хорошо работает в небольшой замкнутой системе, однако практически невозможен в реализации в рамках крупной местности, такой как отдельно взятый город. Для такой реализации общения нужно учитывать слишком много переменных, поскольку для абсолютно каждого NPC нужно выстраивать иерархию взаимоотношений со всеми другими. Когда персонажа всего 3, то каждому нужно прописывать всего 2 вида отношения с отдельным счетчиком, когда же персонажей 50, то для каждого нужно выстраивать 49 уникальных взаимоотношений для каждого из 50 людей! Плюс у каждого NPC должны быть свои уникальные ветки диалогов, построенные на полученной новой информации.

1.2.2 The elder scrolls 4: OBLIVION – Radiant AI

В поисках решения можно обратиться к Radiant AI из игр серии TES, где у каждого NPC есть свой распорядок дня, которому он следует, однако в процессе "жизни" персонаж может столкнуться с другим NPC и начать вести с ним диалог. На этом моменте как раз и начинает действовать диалог, где NPC может узнать новую для него информацию. На момент 2005 года, система искусственного интеллекта, созданная для данной игры, являлась настолько революционной, что даже сами разработчики не всегда понимали, как она работает. Все персонажи имели свои цели жизни и стремились к их выполнению, путешествуя по всему миру. Однако это могло вести к непредвиденным ситуациям, когда NPC мог просто умереть по ходу игры, из-за чего систему пришлось ограничить таким образом, что большинство персонажей стали привязаны к конкретным территориям. Пусть взаимодействие NPC друг с другом происходило в реальном времени, но общение с игроком все еще шло один на один. В это же время, мир вокруг героя замирал, то есть никакой другой персонаж не мог присоединиться к общению.

1.2.3 F.E.A.R

F.E.A.R имеет продвинутую систему ИИ врагов, которые стараются анализировать обстановку и используют окружение, чтобы как можно удачней добраться к игроку и устранить его, эта система носит название “Goal-Oriented Action Planning, GOAP” [5]. Система эта работает на расставлении приоритетов и последовательности действий для NPC, происходит анализ окружающей среды, после чего записываются активные точки взаимодействии и по итогу выстраивается порядок действии в соответствии того, насколько близко активные точки находятся к персонажу. Сама по себе эта система строится вокруг игрока, поскольку F.E.A.R является активно ориентированной игрой в замкнутом помещении, то и вся логика искусственного интеллекта построена вокруг устранения игрока. Логика персонажей состоит из трех состояний:

1) Перемещение в точку.

2) Проигрывание анимаций.

3) Взаимодействие с активным объектом.

Перемещение происходит с помощью усовершенствованного алгоритма A*, который отвечает за поиск самого оптимального варианта последовательности действий. Под каждое действие определяется одно из трех состояний.

Однако похожую реализацию искусственного интелекта можно использовать не только в активной игре, а применить ее для общения с игроком, поскольку анализ окружения можно подстроить и под обычные общение, например, во время общения диалога с игроком NPC может захотеть попить воды, в такой момент как раз и может включиться система цели ориентирования.

1.2.4 The Sims

В играх серии "The Sims" игрок выступает в роли бога, который следит за жизнью одного или нескольких персонажей. Пусть пользователь не может напрямую общаться с NPC, но он может наблюдать за тем, как ведут беседу и строят отношения непосредственно сами неигровые персонажи. Персонажи могут вести диалог, как и сами, так и игрок может выбирать опции в общении. На рисунке 1 показано, как выглядит общение персонажей в игре.

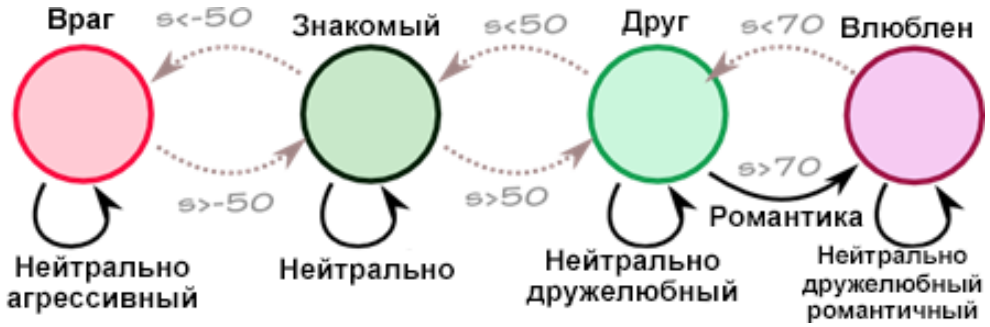

Рисунок 1 – Поведение персонажей в Sims, источник: https://team.inria.fr/imagine/files/2014/10/sims-slides.pdf

Как можно заметить из рисунка, взаимоотношения представлены, как последовательность блоков, открывающихся или закрывающихся в зависимости от показателя очков. На каждом этапе отношений открываются

новые опции взаимодействия, они могут быть как положительными, так и отрицательными. На получение или потерю очков влияют следующие факторы:

1) Настроение.
2) Персональные качества.
3) Случайные факторы.

Совокупность этих показателей определяет, как распределяются очки после диалога. Система социального взаимодействия в "The sims" работает на довольно простом уровне, но эта реализация может хорошо сочетаться с другими формами взаимодействий, например, с системой Nemesis, описанной ранее. Обе системы нацелены на построении отношений между различными персонажами, поэтому связь была бы уместной.

1.2.5 Façade

Взаимодействие с персонажами в игре "Façade" состоит из двух самостоятельных систем, которые работают в симбиозе, стараясь создать правдоподобный опыт общения игрока с NPC [6]. Первая – менеджер драмы (Drama manager), ее суть заключается в том, что она следит за продвижением сюжета по ходу игры и старается вовлечь игрока в ход "пьесы". Сюжет состоит из нескольких развилок, которые система комбинирует на основе действий игрока, создавая целостную историю. Менеджер старается следить кому игрок больше симпатизирует по ходу игры и на основе этих данных открывает и закрывает различные развилки. В случае если развилка ведет в тупик, например, когда игрок резко испортил отношения с персонажем, с которым он их выстраивал на протяжении игры, то система способна возвращаться обратно до тех пор, пока не будет найдена другая развилка.

Вторая система – это язык действия и поведения (Action behavior language). Она определяет то, как именно происходит интеракция игрока с персонажами. Пользователь вводит все слова в текстовой строке, что способствует большему погружению, однако может и привести к поломки искусственного интеллекта, если была введена фраза, которую игра не

понимает. Для этого в игре есть обширный список тем, на которые персонажи стараются реагировать, когда игрок вводит их. Эта же система определяет и интеракцию с окружением, будь это один из NPC или игрок. Персонажи реагируют на различные действия игрока, будь то интеракция с картиной или фигуркой. В такие моменты начинается симбиоз систем, поскольку за реакцию на окружение отвечает система действия и поведения, а уже за последующий диалог отвечает менеджер драмы, старающийся привести игрока к следующей развилке.

Данная реализация взаимодействия с NPC пусть является довольно продвинутой по сравнению со многими другими, однако она слишком заточена под конкретные, созданные разработчиками сценарии, которые могут сломаться если игрок не будет следовать им. У пользователя слишком много инструментов, чтобы сломать игровой процесс. Решением данной проблемы могло бы быть некое ограничение в взаимодействии, созданное таким образом, чтобы больше управлять игроком, однако при этом чтобы у самого пользователя все еще создавалось ощущение контроля над ситуацией. Например, сделать повествование более линейным, но обогатить его различными необязательными деталями, углубляющих историю, которые игрок бы получал в качестве вознаграждения за то, что принимает активное участие в диалоге. Это увеличивало бы заинтересованность в изучении игры, что в свою очередь увеличило бы погружение в игровой процесс. В качестве дополнительного улучшения можно было бы использовать ограничение возможности ответа со стороны игрока, чтобы он не писал текст вручную, а выбирал ответ из списка тем. На первый взгляд ограничение свободы может привести к потере погружения, но на самом деле, как раз слишком большая свобода может привести к выбиванию из вовлеченности в игру. Искусственный интеллект не всегда может корректно понимать запросы игрока, что приводит к ответу, который никак не будет связан с ходом диалога.

1.3 Анализ работ, посвященных данной тематике

1.3.1 Improving video game conversation with trop-informed design

В попытках улучшить взаимодействие игрока с неигровыми персонажами было проведено несколько исследований, в которых пытались выделить ключевые особенности в поведении людей при общении и перенести эти черты в рами игровых миров. В работе “Improving video game conversation with trop-informed design” [7] Стефани Ренник и School of Humanities рассматривают различные тропы взаимодействия с персонажами во время диалога, которые могут как усилить, так и ослабить погружение. Они показывают на различных примерах, как и почему современное общение с NPC рушит вовлеченность игрока, после чего предлагают несколько решений. Например, они затрагивают то, что в обычном разговоре люди никогда не стоят молча и не ждут своей очереди сказать фразу, они всегда поддакивают собеседнику, давая знак, что понимают его или знают то, о чем он говорит. В играх многие NPC выполняют роль посредника информации, такое происходит и в жизни, но это не единственное что происходит. Зачастую речь состоит из различных переходных конструкций, которые сами по себе не несут информации, но показывают отношение человека к чему-то или выражают желания. Например, фраза “Душновато здесь…” показывает то, что человек хочет открыть окно. Авторы статьи предлагают использовать такие конструкции в общении с NPC, чтобы подчеркивать их характерные особенности личности и делать общение более живым.

Другим не менее важным затронутым аспектом является время, которое дается игроку на выбор варианта диалога. В большинстве современных играх общение реализовано пошагово, то есть один персонаж что-то сказал и ждет ответа от второго, и так по кругу. В жизни общение выглядит совсем не так. Диалог идет в потоке, где один дополняет другого, в данной работе была предложена идея реализации такого общения. Диалог всегда идет в реальном времени, и пока персонаж что-то говорит у игрока появляются облачка с вариантами ответов, которые со временем пропадут, если игрок не выберет их.

Это похоже на то, как наши мысли формируются во время обычного общения в жизни.

В исследовании также затрагивается необходимое для нашей работы прерывание диалога. Рассматриваются способы того, как можно интегрировать этот аспект в игры и то, как это действие влияло на отношение NPC к игроку. В современных играх прерывание зачастую не несет никакие последствия и выступает скорее инструментом для перехода к другим действиям, чем полноценным способом интеракции. В частности, в работе предлагается две опции для данного действия:

1) Использование знака, показывающего, что игрок хочет перейти к следующему этапу.

2) Произношение таких звуков, как “угу”, “ага”.

По своей сути эти действия имеют тот же смысл, что нажатие кнопки для пропуска сцены, однако они интегрированы в игровой процесс и их можно использовать, как механики взаимодействия, влияющие на отношения, как в положительном, так и в отрицательном ключе.

Многие из представленных в работе тропы по усовершенствованию погружения можно удачно имплементировать в систему много потокового общения, хотя в исследовании не уточняется, рассматривались ли эти решения для взаимодействия одновременно с несколькими персонажами или один на один, но полученные результаты все еще могут быть полезными. Также имплементация всех представленных способов общения может чрезмерно усложнить восприятие игрока и произвести эффект обратный от задуманного, а именно у игрока банально пропадет желание вести диалог с различными NPC. В играх очень важен контекст происходящего, нередко игроку как раз нужно всего лишь узнать небольшую информацию, а обильное количество игровых механик общения в десятки раз усложнит то, что того не требует. Поэтому если и вводить описанные в работе механики, то лучше либо ограничиться самыми важными из выше представленных, либо тщательно балансируя их, не выкидывая все разом на игрока, а вводя постепенно и

уместно.

1.3.2 A culture Model for Non-Player Characters' Behaviors in Role-Playing Games

В работе "A culture Model for Non-Player Characters' Behaviors in Role-Playing Games" [8] за авторством Луиса Фернандо Бикальо, Бруно Фейхо и Аугусто Баффа представления реализация поведения NPC, основанная не только на эмоциях и персональных чертах, но и на его происхождении и культуры. То откуда он родом кардинальным образом влияет на то, как он будет общаться с игроком. Поскольку то, что в рамках обычая для одного может оказаться неприемлемым для другого. Все его предпочтения, интересы и в какой-то мере характер определяются в симбиозе с его культурным происхождением. В исследовании представлена эмоциональная модель, использующая 6 различных показателей, которые определяют характер персонажа:

1) Время – определяет то, как быстро персонаж перемещается и как быстро реагирует.

2) Благо состоятельность – определяет, как персонаж будет реагировать на такие действия, как взятка или воровство.

3) Титул – определяет то, как игрок ведет себя рядом с ним, персонаж с большим показателем титула будет недоволен, если игрок подойдет слишком близко для разговора и нарушит его личное пространство.

4) Учтивость – определяет то, как персонаж будет реагировать на грубое общение.

5) Коллективизм – определяет то, насколько персонажу небезразличные другие NPC, когда они рядом с ним.

6) Рациональность – определяет насколько яркие будут его эмоции.

В симбиозе с другими моделями, определяющие характер и эмоции, получается новая модель поведения персонажей в игровом мире. На рисунке 2 показано, как все поведенческие модели комбинируются в персонаже.

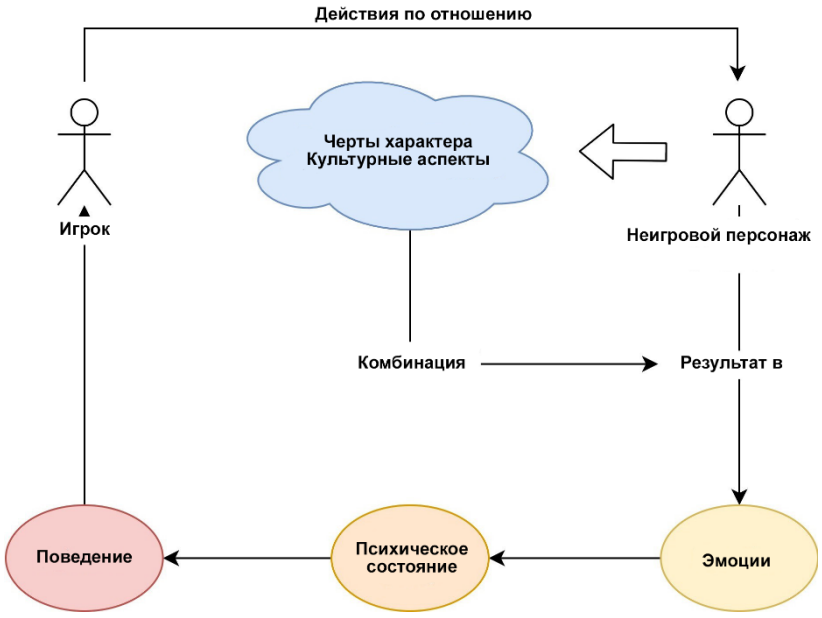

Рисунок 2 – Диаграмма взаимодействия NPC, источник: https://www.sbgames.org/proceedings2020/ComputacaoFull/210071.pdf

После того, как игрок начал общение с персонажем, NPC обращается к своим переменным, которые были определены с помощью модулей поведения, после чего результат определяет эмоциональное и ментальное состояния, после чего дает ответную реакцию для игрока.

Данная модель поведения выглядит уникально, и ее имплементация или имплементация некоторых ее аспектом могут улучшить опыт взаимодействия с NPC.

1.3.3 Asynchronous Dialogue System

В работе “Asynchronous Dialogue System” [9] Alfred Andersson и Keman Nguyen описывают процесс создания асинхронного диалога с NPC. Суть их работы заключается в создании более правдоподобной системы взаимодействия с игровыми персонажами, где игрок должен активно участвовать в общении с персонажем, реагируя вовремя на его действия. Отличительной чертой данной системы является то, что в отличии от большинства игр, где общение представлено пошагово, (сначала NPC говорит фразу, потом игрок, потом снова NPC и т.д) в их системе диалог происходит в реальном времени, и игроку всегда нужно понимать, когда нужно промолчать и дать персонажу договорить фразу или, когда лучше перебить его и дать

понять, что ты вовлечен в разговор. Оба варианта в разные этапы разговора могут привести к разным исходам, в одном случае перебивания может улучшить отношения собеседника к игроку, а в другом ухудшить, нужно всегда понимать, в какой момент отвечать. Такая система общения во многом похожа на настоящий разговор.

В своей работе авторы, создавая систему, в первую очередь ориентировались на визуальные новеллы, от чего все взаимодействие строится на том, что общение происходит на отельной сцене, где присутствует только два действующих лица – игрок и его собеседник.

Эта система работает хорошо, когда дело касается перебивания собеседника, однако она не затрагивает другие аспекты более живого взаимодействия. Например, вербальное общение, когда люди кивают друг другу, которое было описано в одной из выше перечисленных работ.

Данную систему можно использовать как основу для реализации многопотокового общения, дополняя ее иными ранее описанными тропами, лучше погружающими в общения и модифицирующие его для общения сразу с несколькими персонажами.

2 ИССЛЕДОВАНИЕ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Для дальнейшей работы над реализации системы с ведением диалога в группе NPC, нужно более подробно изучить принцип работы систем, на которых будет базироваться данная реализация. Тщательный анализ пошаговой работы других систем, поможет определить их слабые места, которые можно будет исправить при имплементации этих решений в нашу разработку. Кроме того, понимание структуры работы других алгоритмов общения с неигровыми персонажами, поможет более четко представить финальную схему того, как именно будет происходить процесс общения с персонажами в нашем решении.

2.1 Система взаимодействия с NPC

2.1.1 Анализ структуры

Система взаимодействия между персонажами реализована через несколько этапов, далее по тексту эти этапы будут расписаны.

Существует игрок, который обладает механикой перебивания собеседника. В любой момент диалога он может воспользоваться данной опцией, что приведет к изменению курса диалога и повлияет на отношение персонажей к игроку.

Есть неигровые персонажи, которые ведут диалог по заранее заготовленному сценарию, активно пытаясь вовлечь игрока. Каждый такой персонаж обладает одним из представленным ниже эмоциональным параметром:

1) Знакомство.
2) Уважение.
3) Злость.
4) Нейтральность.

Каждый из этих четырех параметров напрямую влияет на показатель пятого параметра – дружелюбие. От значения переменной дружелюбия зависит то, как именно будет идти диалог между персонажем и игроком. Числовые показатели данной характеристики открывают или закрывают

активные опции для игрока, это могут быть как некоторые характеризующие фразы, так и целые диалоговые ветки. В конечном итоге, именно этот параметр отвечает за то, чем закончится общение между персонажами.

Каждое действие игрока отражается на показателе дружелюбия, будто активная диалоговая опция или молчание, оба будут иметь конечный вес. Действия игрока либо повышают, либо понижают показатель доверия собеседника. Все опции пользователя напрямую связаны с эмоциональными параметрами, которые по ходу общения могут и будут изменяться у неигрового персонажа. Одно и то же действие со стороны игрока будет иметь разный конечный вес в зависимости от эмоционального параметра. Предположим, что игрок решил перебить своего собеседника во время его речи, в любом случае это действие будет иметь негативное последствие на отношения с этим персонажем. Однако значимость этого действия будет варьироваться от того, какую эмоциональную связь имеет собеседник. Если персонажи едва знакомы, то отношение значительно ухудшатся, а некоторые диалоговые ветки могут быть заблокированы в виду недоверия персонажей друг к другу. А если же отношения имеют дружественный характер, и беседчик имеет уважительное отношение, то действие перебивания все еще повлечет негативный оттенок. Но, все же, урон по доверию будет получен не столь значительный, что приведет к закрытию каких-либо диалоговых веток.

Структурно вся система взаимодействия состоит из нескольких логических блоков, в которых хранится и между которыми передается, какая-либо информация. Описание этих блоков представлено ниже:

Диалоговый блок – содержит в себя диалоговый текст, представленный персонажем, а также различные действия, которые игрок может выбрать для продвижения диалога. Активные действия связаны с другими блоками и игрок может переместиться к ним, выбрав соответствующее действие. Весь диалог с персонажем разбит на несколько таких диалоговых блоков, каждый из которых имеет свои параметры.

Блок свойства – он содержит в себе эмоциональное состояние

конкретного персонажа. Именно здесь хранится параметр дружественности. Этот блок обновляется в реальном времени и используется для регулирования отношений между игроком и NPC.

Базовый блок – он хранит в себе значение, которое может быть integer, float или bool, преимущественно используется для блока состояния.

Блок состояния – сравнивает показатели блока свойства и базового блока. Сравнение идет по показателям равно, больше, меньше, не равно и так далее. Если сравнение соответствуют условию, то диалог переходит к следующему блоку, закрепленному к true, иначе же false. Этот блок закрепляется к действиям игрока, чтобы открывать или же закрывать диалоговое опции в соответствии с показателем взаимоотношений.

Блок действия – содержит действие, которое игрок может выбрать в диалоге с персонажем. Действия, содержащиеся в данном блоке, отличаются от активных действий в диалоговом блоке тем, что они могут быть вызваны много раз в разных диалоговых ситуациях, если это потребуется по игре. В то время, как действия из блока диалога привязаны к конкретному блоку и повторно не вызовутся.

Блок перемешивания – создает новый диалоговый блок еще до того, как текущий закончится, в результате в игре идут две фразы от разных людей параллельно. Он включается, когда игрок выбирает действие во время фразы персонажа.

Блок конфигурации – используется между переходами от одного диалогового блока к другому. Регулирует параметры дружелюбия и эмоциональности.

Блок входа – Стартовая точка, с которой начинается цепочка диалога.

Маркер – блок, который дает доступ к внешнему коду с целью активировать другие дополнительные события.

На рисунке 3 представлена схема того, как реализован алгоритм работы диалога с перебиванием в рассматриваемой работе.

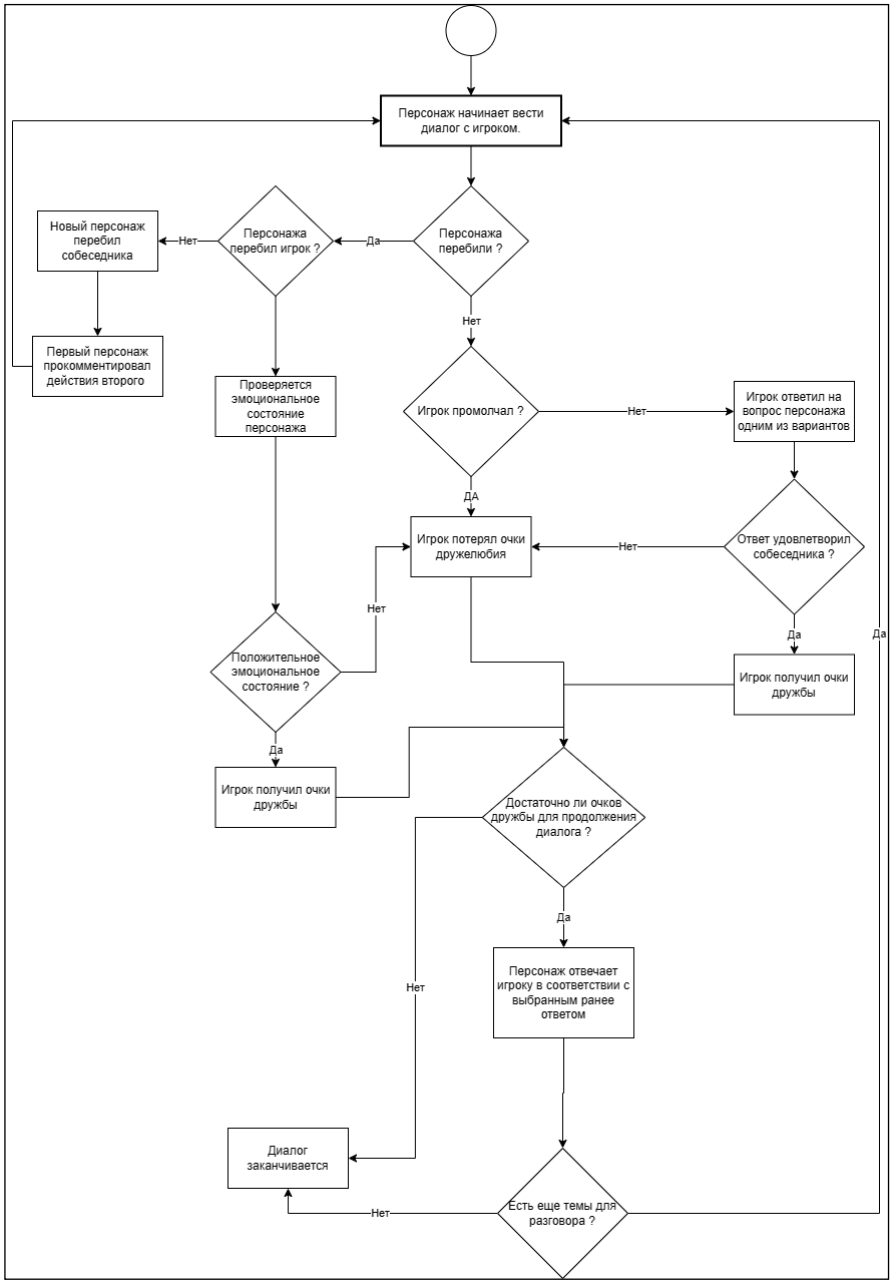

Рисунок 3 – Схема диалога в игре

В процессе ведения диалога, разные собеседники могут его перебить, перетянув одеяло на себя. Это может сделать, как игрок, так и сторонний персонаж. Диалог идет в реальном времени и продвигается по своей структуре без пауз даже во время выбора реплик со стороны игрока. Игрок решает сам хочет ли он вести беседу или молчать, однако это сказывается на его отношении с этим персонажем. Сами по себе опции для действий игрока появляются по ходу диалога, то есть в любой момент перебить собеседника нельзя. Все диалоговые опции строго предопределены сюжетной структурой.

Перебивание может оказать, как положительный, так и отрицательный эффект. Все зависит от показателя эмоционального состояния, склонен ли собеседник к перебиванию или же нет. А эмоциональное состояния определяется по контексту диалога, если персонажа постоянно перебивает другой неигровой персонаж, то он будет раздраженным и аналогичное действие со стороны игрока приведет к ухудшению отношений. Если же общение хорошо складывается, то игрок может перебить собеседника, закончив предложение за него. Это покажет собеседнику вовлеченность игрока в разговор, что улучшит отношения между персонажами.

Игровой персонаж не будет ждать вечно, даже если игрок не выбирает ни одно из действий. Рано или поздно диалог дойдет до той точки, когда игровой персонаж сам произнесет фразу, чтобы как-то закончить диалог или продвинуть сюжет.

Разговор идет до тех пор, пока отношение между собеседниками не испортятся, что приведет к закрытию дальнейших реплик. Или же до тех пор, пока все темы для разговора не закончатся. На рисунке 4 проиллюстрирован пример того, как могут проходить диалоги в этой системе.

Рисунок 4 – Пример диалогового дерева в этой системе

Каждое из действий игрока ведет к одной из трех разных веток с разными исходами. Даже если игрок и вовсе ничего не выбирал.

2.1.2 Выявление проблем

Посмотрев на анализ работы выше представленной системы, можно выделить ряд ее недостатков, которые можно было исправить. Ниже по тексту

описаны эти аспекты.

Первым делом стоит обратить внимание на то, что абсолютно все перебивания строго построены на скриптах. Программисту или сценаристу вручную нужно прописывать все эти сценки с четким осознанием того, когда можно перебивать, а когда нельзя. Игрока по сути водят за ручку, давая лишь иллюзию того, что он может перебить собеседника и изменить ход повествования.

По своей сути, перебивание и молчание являются классическими вариантами ответа на диалог собеседника, как во всех визуальных новеллах или RPG играх. Существенным изменением является то, что эти варианты замаскированы и появляются не в конце диалога с выбором, а по ходу диалога. Как такого инструментария здесь не представлено.

Перебивания со стороны второго неигрового персонажа не являются как таковой механикой, а просто немного другая визуальная интерпретация диалога с несколькими персонажами. Игрок никак не может взаимодействовать со вторым персонажем, как-то прокомментировать его слова или переключить диалог на него. Сам этот персонаж просто создает иллюзию группового диалога, когда по факту практически не влияет на события. Он выступает, как инструмент для игрока, показывающий эмоциональное состояние первого персонажа, с которым непосредственно идет диалог.

Малое количество участников в диалоге. Вовлечение большего количества участников сделает общение живее, однако они все должны быть полноценными участниками со своими отличительными чертами.

В своей сути эта диалоговая система старается сделать общение с персонажами более живым, проецируя некоторые реальные ощущения от диалога на виртуальное общение. К этим вещам относятся: временные рамки, когда можно сказать фразу, периодические перебивания со стороны и возможность самому перебить собеседника. Однако, даже эти ощущения довольно ограничены. Довольно большая часть ощущения от группового

общения из реальной жизни все еще теряется. В этой реализации не предусмотрен вариант того, что второй собеседник переключит внимание на себя в какой-то момент и начнет вести свой собственный диалог. В то же время, как и игрок не сможет переключиться на другого персонажа и поддержать его разговор.

В жизни диалог очень часто отклоняется от первоначальной темы общения, постоянно происходят прерывания и переключения на другие рассуждения, а спустя какое-то время общение снова может вернуться изначальной тематике. Здесь же второй персонаж представляет декоративный характер

По итогу, ряд аспектов взаимодействия с персонажами в представленной системе можно изменить или улучшить, ниже по тексту будут описаны эти аспекты.

Первым делом, можно улучшить возможность переключения на другого NPC по действую игрока с последующим диалогом с новым неигровым персонажем.

NPC может перетягивать диалог на себя в моменты, когда поднимается тема, на которую ему есть что сказать.

В момент, когда общение снова возвращается к первому персонажу, он может прокомментировать, сказанное ранее другим участником разговора. Так общение с несколькими персонажами будет ощущаться целостным групповым диалогом, а не отдельными не связанными друг с другом кусками, будто общение идет один на один.

Добавить больше собеседников в диалог, с возможностью каждого комментировать. А для увеличения количества разнообразных комментариев, добавить больше характеристик, отвечающих за индивидуальность каждого персонажа.

Сделать систему более автономной. Она не должна полностью зависеть от заранее подготовленных диалогов, ведь игрок может собрать совершенно разную группу собеседников в любой случайный промежуток времени, а все

такие ситуации невозможно заранее просчитать разработчику.

Чтобы не изобретать дополнительные системы для реализации некоторых из описанных улучшений, можно воспользоваться уже существующем инструментарием. Для создания большей индивидуальности персонажам, можно воспользоваться системой, описанной в работе Луиса Фернандо Бикальо, Бруно Фейхо и Аугусто Баффа “A culture Model for Non-Player Characters” Behaviors in Role-Playing Games” [8], ранее рассмотренной в теоретической части.

2.2 Система поведения

Использование идей из культурной модели поможет довольно сильно разнообразить поведение персонажей на ситуации, возникающие внутри диалога. На одно и то же действие персонажи с разными характером и происхождением обязаны действовать не похоже. Один персонаж может спокойно выслушать рассказ собеседника, а после прокомментировать какую-то из ранее поднятых тем, а другой может прервать историю и начать рассказывать свою. Именно аспекты характера и происхождения влияют на поведение в общении, поэтому имплементация системы, использующей такие метрики, сыграет на руку нашей модели общения.

Для того, чтобы внедрить культурную систему, нужно сначала проанализировать, как она работает в деталях и выделить ключевые особенности:

В исследуемой структуре игровой персонаж может совершать ряд действий по отношению к NPC, одна часть из которых могут оказать позитивный на отношения персонажей, а другая, напротив, только испортят взаимопонимание. На то, как какие действия игрока влияют на доверия окружающих к нему, определяется персональными и культурными характеристиками.

Для начала стоит разобрать, какие именно действия может совершать игрок по отношению к другим:

1) Приближение к NPC: если игрок не соблюдает личное

пространство любого из NPC, то это может испортить отношения с этими персонажами.

2) Стрельба из оружия: если игрок начнет стрелять, то все рядом находящиеся персонажи отреагируют либо позитивно, либо негативно на это действие в зависимости от значения коллективизма.

3) Толкание NPC: позитивная либо негативная реакции в зависимости от характеристик.

4) Общение в вежливом или невежливом тоне: зависит от уровня вежливости персонажей.

5) Отдать или украсть деньги: влияет положительно или отрицательно в зависимости от показателя достоинства NPC.

6) Предрассудки: показатель, который зависит от расы или места рождения персонажа.

У самих же неигровых персонажей, в свою очередь, есть числовые характеристики, которые и определяют, какую именно реакцию получит игрок при взаимодействии с ними:

1) Время.
2) Благосостояние.
3) Достоинство.
4) Учтивость.
5) Коллективизм.
6) Рациональность.

Каждый этот показатель варьируется от 0.1 до 1.0. Чем больше какой-либо из этих показателей, тем более резко персонаж отреагирует на действия со стороны игрока. Если, например, у NPC показатель достоинства 0.9, то на нарушение личного пространства со стороны игрока, персонаж отреагирует с большой эмоциональной интенсивностью. Если же показатель достоинства будет 0.1, то он отреагирует с незначительной интенсивностью. Действия игрока также влияют на эмоциональное состояние персонажа.

Всего в системе имеется восемь эмоциональных характеристик: четыре

положительных и четыре отрицательных. В таблицах 1 и 2 представлены эти эмоции.

Таблица 1 – Положительные эмоции и их значения

Легкая эмоция (0.2)	**Стандартная эмоция (0.5)**	**Сильная эмоция (1.0)**
Умиротворенность	радость	экстаз
Принятие	Доверие	Восхищение
Предчувствие	Страх	Террор
Отвлечение	Сюрприз	Изумление

Таблица 2 – Отрицательные эмоции и их значения

Легкая эмоция (-0.2)	**Стандартная эмоция (-0.5)**	**Сильная эмоция (-1.0)**
Горе	Печаль	Задумчивость
Скука	Отвращение	Ненависть
Раздражение	Гнев	Ярость
Интерес	Ожидание	бдительность

В среднем столбце обеих таблиц представлены стандартные эмоции, а слева и справа представлены менее яркие и более яркие эмоции соответственно. За то, какая именно эмоция будет у персонажа отвечают культурные характеристики. Для каждой эмоции закреплено свое конкретное значение в соответствии со столбцом.

Эмоции имеют прямое влияние на коммуникацию и поведение. В культурной модели на каждое действие игрока есть список результирующих эмоций. Связь между действиями игрока, полученной эмоции и финальной реакции можно увидеть в таблице 3

Таблица 3 – Действия игрока и реакция NPC на них

Действия игрока	**Культурные параметры**	**Список эмоций на культурный параметр**
Атака	Достоинство	Ярость, возмущение, отчаяние, ужас
Стрельба	Коллективизм	Раздражение, пессимизм, неодобрение, опасение
Вред	Коллективизм	Злость, презрение, недоверие страх
Ранение	Коллективизм	Гнев, неодобрение, гордость, радость
Отдать предмет	Благосостояние	Радость, оптимизм, надежда, доверие
Украсть предмет	Благосостояние	Печаль, стыд, раскаяние, отвращение
Отдать деньги	Благосостояние	Восхищение, любовь, сентиментальность, экстаз
Украсть деньги	Благосостояние	Горе, доминирование, благоговение, ненависть
Социальная дистанция	Достоинство	Смятение, тревога, восторг, интерес
Личная дистанция	Достоинство	Предвкушение, цинизм, любопытство, удивление
Интимная дистанция	Достоинство	Бдительность, агрессивность, представление, изумление
Вежливое общение	Учтивость	Скука, зависть, гордость спокойствие
Невежливое общение	Учтивость	Задумчивость, вина, одобрение, болезненность

Каждая из четырех представленных эмоций имеет диапазон ассоциации с параметром культуры, располагаются диапазоны следующим образом: [0, 20), [20, 50), [50, 70) и [70, 100)

Для того, чтобы определить, какая эмоция будет выбрана, используется формула:

$$CF = \sqrt{CD_{Value} * (1 - R_{VALUE})}), \quad (1)$$

где CF– культурный фактор; CD_{Value} – значение параметра культуры, который отреагировал на действия игрока; R_{VALUE} – значение параметра рациональности у персонажа.

Финальное значение получается в диапазоне от 0.1 до 1.0 и по его результату мы определяем в какой диапазон из четырех эмоций попало число. Например, если игрок атаковал, то в качестве культурного параметра выбирается достоинство и подсчитывается значение эмоции, если значение выше 0.7, то будет выбран ужас, если меньше 0.7, но больше 0.5, то будет отчаяние, если меньше 0.5, но больше 0.2, то возмущение, в противном же случае выберется ярость.

У каждого персонажа есть уровень доверия к игроку, который вычисляется на основе действий игрока, и который влияет на то, как персонаж будет общаться. Вычисление эмоции нужно как раз для определения уровня доверия. Для определения доверия используется следующая формула:

$$tLvl = tLvl + pLvl * mentalFactor \quad (2)$$

Где $tLvl$– уровень доверия; $pLvl$ – параметр предрассудков, который задается среди характеристик персонажа; $mentalFactor$ – численное значение эмоции, которая была выбрана.

2.3 Реализация

После анализа системы, использующейся в качестве основ для реализации нашей модели, и дополнительных инструментариев, способных дополнить и улучшить опыт от общения с NPC, можно переходить к описанию внедрения этих инструментариев в изначальную систему для реализации своей модели.

2.3.1 Улучшение поведения персонажей

Модель поведения персонажей на основе культуры имеет ряд полезных решений, которые могли бы пригодиться для улучшения опыта от асинхронной модели. В первую очередь можно внедрить культурные характеристики для персонажей, которые поспособствуют увеличению разнообразия в общении, а также большему вовлечению дополнительных участников диалога. На основе этих характеристик можно будет прописать точки интереса для каждого из персонажей, на которые он будет реагировать, в случае если услышит их. Тогда он сможет вникать в диалог и даже перебивать собеседника, что обсудить точку интереса. Решение перебивать или не перебивать будет приниматься на основе культурных характеристик, которые в нем заранее заданы разработчиком.

Если в культурологической модели все эмоции и реакции вызывались по отношению игроку, то в нашей реализации эмоциональные реакции можно использовать для определения решения перебивать или дослушать собеседника. Все основные культурные характеристики можно оставить без изменений и просто вписать их в реализацию персонажа. А вот эмоциональные реакции нужно несколько изменить и адаптировать под обычное общение на разные темы. Их список можно упростить и свести до:

1) Нейтральная заинтересованность (0).
2) Низкая заинтересованность (-1.0).
3) Высокая заинтересованность (+1.0).

В случае Высокой заинтересованности, персонаж будет настроен комментировать слова собеседника, в случае нейтральной реакции перехода на пункт перебивания не будет, и он продолжит слушать дальше, после чего прокомментирует. Если заинтересованность низкая, то персонаж не будет обсуждать эту тему, пока его не спросить о ней.

Когда персонаж слышит интересующую его тему, система запускает вычисление реакции по формуле (1) на основе внесенных культурных показателей. В зависимости от полученной эмоций системой принимается

решение перебить или не перебить собеседника.

Поскольку количество выходных эмоций на каждое действие сократилось до трех, то и диапазон ассоциаций теперь высчитывается немного иначе: [0, 20) в случае низкой заинтересованности, [20, 60) в случае нейтральной и [60, 100) в случае положительной.

Алгоритм того, как происходит оценивание поднимаемой темы показан на рисунке 5.

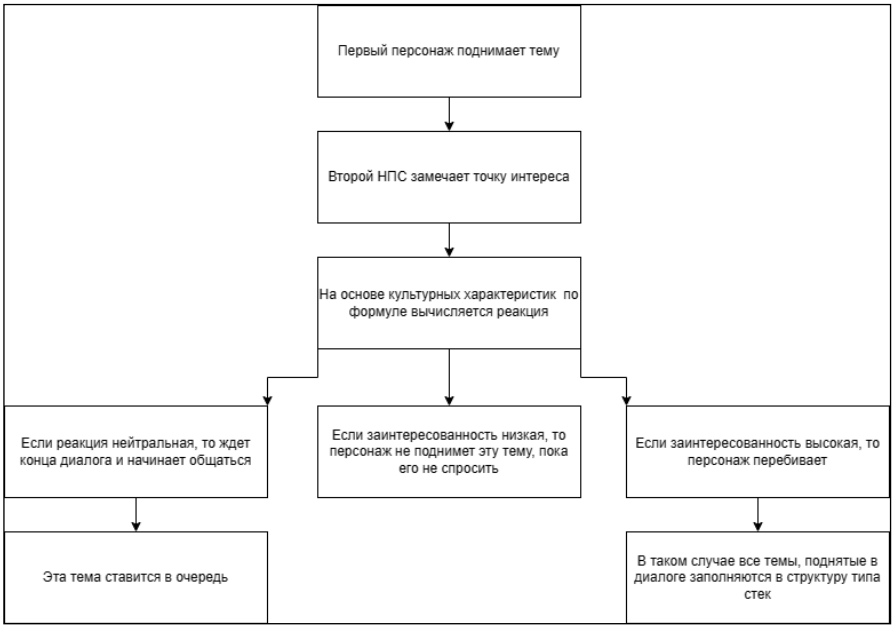

Рисунок 5 – Алгоритм оценки темы

По итогу анализа диалога, система может прийти к одному из трех разных вариантов. В двух из них алгоритм создаст структуру порядка обсуждения, если эта структура еще не была создана, в противном же случае, тема просто добавится в существующий список.

Точки интереса для персонажа, на которые он будет реагировать, можно реализовать через связку ключевых слов. У каждого персонажа в его памяти хранится наборе тегов, на которые он может среагировать. В случае если одна из интересующих его тем поднимается в диалоге, он, в зависимости от своих характеристик, может прокомментировать реплику, сказанную другим персонажем, после чего сведет диалог к этой теме. На каждый историю персонажа, которую он готов рассказать игроку и другим NPC, хранится некоторое количество тегов, которые и активизируют историю. Причем,

рассказ персонажа может быть представлен таким образом, что в нем будут находиться точки интересов для других персонажей. В таком случае уже другие персонажи будут реагировать на свои теги, замеченные в рассказе.

В случае, если персонаж решил не перебивать диалог на свою тему, он все еще фиксирует тему, совпадающую с точкой интереса, и в момент, когда игрок переключит общение на него, он начнет свой диалог с упоминания этой темы от прошлого собеседника. При этом, персонаж запоминает рассказ собеседника на какой-то промежуток времени, определяемый параметром “время” внутри характеристик. По окончанию времени разговори стирается из памяти и в случае интеракции игроком с ним, он не будет уже комментировать прошлый диалог.

Структура, содержащая в себе информацию об истории конкретного персонажа вместе с его тегами, представлена следующим образом.

Таблица 4 – Структура хранения тегов в персонаже

ID	**Тема**	**Закрепленная культурная характеристика**	**Теги**	**История**
22	Хороший улов	Благосостояние	Рыба, золотая, рыбалка	На днях я выловил огромную золотую рыбу, представляешь!
35	Странная ситуация	Коллективизм	Дерево, человек, странная ситуация	Шел я как-то по лесу и увидел дерево, на верхушке которого сидел человек, во дает!
12	Еле успел!	Достоинство	Повозка, смерть, выжил	Меня вчера чуть не задавила повозка!

ID – отвечает за идентификатор истории, по нему можно определить чей и о чем рассказ, когда истории начинают храниться в очереди или стеке.

Тема – отображается в диалоговом окне персонажа в случае если игрок решил пообщаться

Закрепленная культурная характеристика – отвечает за то, какая именно характеристика будет использоваться при подсчете эмоции в формуле

Теги – ключевые слова по которым персонаж определяет интересна ли ему история и готов ли он её рассказать

История – сама история, которую NPC будет рассказывать в случае если перебьет собеседника или с ним заговорит игрок. Здесь могут храниться как обычные предложения, так и ветки диалогов, на которые игрок будет отвечать. Все истории заранее прописаны разработчиками в структуру. Вручную написанные сюжеты приводят к меньшему количеству неожиданных и неблагоприятных исходов. В таких играх, как Façade [6] вся история генерируется по ходу игры, что довольно часто привод к поломке алгоритма, поскольку игрок не следует заложенной игрой схеме. Во избежание поломок работы нашей модели, было принято решение сами сюжеты реализовать вручную, а генерацию использовать в более конкретных и контролируемых аспектах.

У каждого персонажа хранится такая отдельная структура со своими уникальными тегами, историями, темами идентификаторами и показателями культурных характеристик.

2.3.2 Ответная реакция

Каждый раз, когда персонажа перебивают на его рассказе, модель запоминает точку остановки в диалоге на последнем абзаце, с которого продолжит после возвращения диалога к NPC-1. Для этого у каждого абзаца в диалоге стоит числовой маркер, который игра запоминает в случае перебивания, система хранит это значение и после того, как диалог второго персонажа закончится, модель находит нужную историю через сохраненный

идентификатор, после чего пробегается по всем маркерам этой истории и находит нужный.

Однако, перед тем, как продолжить диалог с точки остановки, персонаж скажет предложение, которое послужит мостиком между темой NPC-2 и его старой темой, на которой его прервали. Он может сделать переход заранее заготовленной фразой, например, такой как: “Что касается моей темы…”, а может и прокомментировать тему собеседника, фразой, которая будет сгенерирована на основе его характеристик.

Для генерации фразы можно использовать ChatGPT[10], который активно используется в генерации сюжетов для разных игр. В работе Judith van Stegeren и Jakub Mysliwiec “Fine-tuning GPT-2 on annotated RPG quests for NPC dialogue generation” [11] эта нейросеть используется для генерации сюжетных заданий для RPG игр. А в исследовании Mika Hämäläinen и Khalid Alnajjar “RPG-GPT” [12] рассматриваются пути и игровые механики, в которых можно использовать GPT для улучшения получаемого опыта.

Работу алгоритма генерации ответа можно реализовать следующим образом:

1) NPC-2 перебивает NPC-1 на основе точек интереса.

2) NPC-1 анализирует обсуждаемую тему и через идентификатор находит культурный параметр, который повлиял на перебивание и находит числовое значение этого параметра у себя.

3) По формуле (1) высчитывается какая эмоция будет вызвана, после чего она записывается в string переменную.

4) У NPC-1 запускается таймер, по истечению которого персонаж потеряет интерес к комментированию истории.

5) Вся история NPC-2 записывается в string переменную, которая потом будет передана в ChatGPT.

6) После окончания рассказа NPC-2 и возврата и перед возвратом к NPC-1, в ChatGPT передается две string переменные: одна с историей, и вторая с эмоциональной реакцией.

7) В нейросеть передается запрос по созданию комментария из string переменной истории с эмоциональным окрасом string переменной эмоции.

8) Сгенерированный текст отправляется NPC-1 для озвучивания.

9) После произношения комментария, NPC-1 находит маркер, на котором остановился и продолжает свой рассказ.

На рисунке 6 показан весь алгоритм работы взаимодействий между NPC.

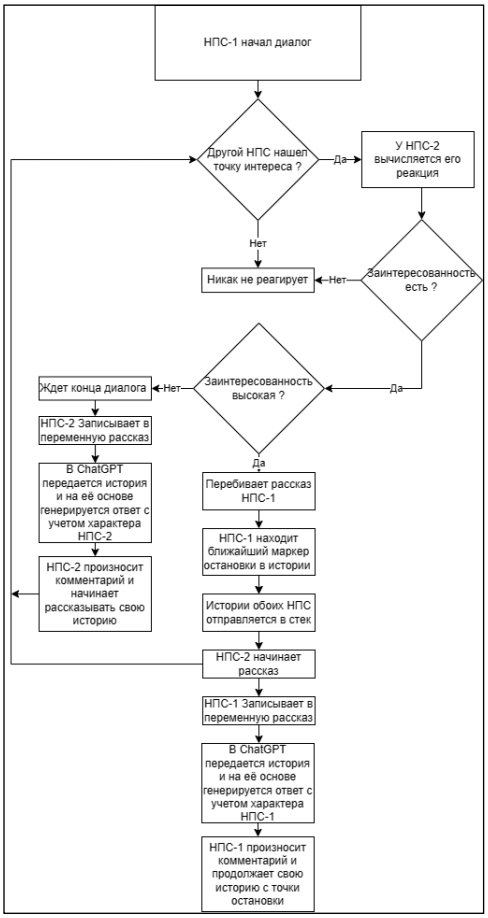

Рисунок 6 – Взаимодействие между NPC

2.3.3 Взаимодействие игрока

Взаимодействие игрока с NPC происходит следующим образом:

1) Игрок подходит к NPC и начинает диалог.

2) Он выбирает одну из тем для диалога из выпавшего списка.

3) Игрок может дослушать диалог или прервать общение и переключиться на рядом стоящего персонажа. В случае прерывания диалога, запоминается ближайший маркер у NPC. По формуле (2) пересчитывается отношение персонажа к игроку.

4) Если NPC-1 слышит диалог между игроком и NPC-2, то он начинает запоминать его и генерировать ответ, как по аналогии с общением между двумя NPC.

5) Если NPC-1 услышит точки интереса, то включается алгоритм оценивания интереса.

На рисунке 7 показан алгоритм взаимодействия игрока с NPC

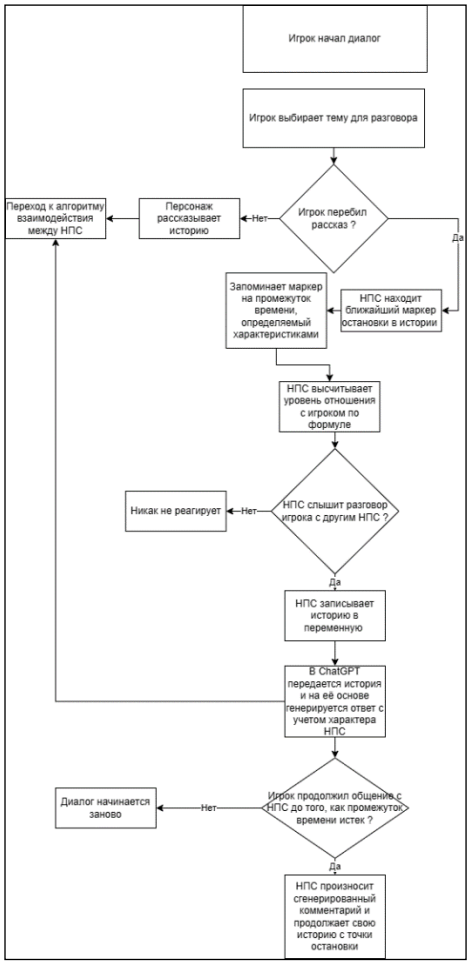

Рисунок 7 – Взаимодействие игрока с NPC

Реакция не перебивание со стороны игрока зависит от темы, на которой он перебил. А заинтересованность темы определяется через показатель культурных характеристик. Если произошло перебивание, то персонаж отреагирует соответствующей эмоцией (либо нейтральной, либо отрицательной). После чего системой будет вычислено по формуле (2), как изменилось отношение персонажа к игроку. Уровень отношений отвечает за то, на какие темы будет говорить персонаж с игроком.

Если игрок перебил NPC -1 и переключил общение на рядом стоящего

NPC-2, то система начнет записывать диалог в память первого персонажа. После окончания общения с NPC -2 информация о диалоге хранится некоторое время в памяти NPC-1 (время хранения в памяти определяется культурной характеристикой “Время” внутри персонажа), после чего стирается из его памяти. Это запоминание нужно для того, чтобы NPC -1 прокомментировал диалог между игроком и NPC-2 перед тем, как продолжить общение на тему, ранее поднимаемую с игроком. Эта механика позволит сделать ощущение от общения между персонажами живее, что увеличит погружение в игровой процесс. Комментарий на хранящуюся внутри персонажа информацию происходит за счет генерации сообщения на основе характеристик персонажа. Генерация происходит через использование ChatGPT, встроенного в систему. Генерация работает аналогичным образом, как и при общении двух NPC друг с другом.

Такую систему можно использовать в большом количестве разных игровых ситуаций. Например, игрок знает, что у персонажа есть информация, которая ему нужна, но он не хочет её говорить игроку, в виду личных причин. Чтобы заставить его сказать эту информацию, можно подстроить групповой диалог, таким образом, что он не выдержит и сам все расскажет в порыве эмоций, потому что эта довольно эмоциональна для него. Можно начать вести общение в группе из 3 персонажей: игрок, тот от кого нужна информация и третий собеседник, через которого диалог будет идти. Игрок будет общаться на разные темы с третьим собеседником и в какой-то момент будет поднята тема, которая заденет второго участника, тогда он не выдержит и случайно расскажет нужную нам информацию.

3 РЕШЕНИЕ

3.1 Структура модели

Сама система взаимодействия игрока с персонажами состоит из трех основных элементов:

1) Персонажи и их персональные характеристики.
2) Чат-бот обработчик.
3) Диалоговая структура.

Через диалоговую структуру происходит весь основной процесс общения, ей передаются все ранее обработанные параметры, где диалоговое окно воспроизводит и отображает полученный результат, через эту структуру игрок взаимодействует с другими персонажами, получает от них и передает им данные. Через эту структуру игрок напрямую взаимодействует с персонажами в игровом пространстве.

Персонажи хранят в себе заранее заготовленные характеристики, показатели которых влияют на то, как они будут реагировать на тему, которую обсуждает игрок с другим NPC в данный момент. Эти параметры определяют финальный запрос, который отправляется в нейросеть на обработку. Составление запроса происходит в тот момент, когда персонаж слышит разговор другого NPC, он анализирует услышанное и решает насколько он заинтересован в разговоре.

Чат-бот составляет сообщение на основе полученного предложения. Финальный результат зависит от нескольких факторов:

1) Времени.
2) Заданной температуры.
3) Количества токенов на вход.
4) Сложности запроса.

После составления предложения, оно отправляется на хранение в один из параметров персонажа, а тот в свою очередь передает его в диалоговую структуру, где уже выводится сообщение и тема, которую персонаж готов обсудить.

В следующих разделах подробнее о том, как работают элементы модели.

3.2 Структура NPC

Неигровые персонажи имеют три основных состояния:

1) Покой.
2) Анализ окружающих диалогов.
3) Участие в диалоге.

В первом состоянии NPC бездействует или занимается заранее прописанным действием. Второе состояние включается в тот момент, когда персонаж входит в зону слышимости диалога другим NPC, тогда он начинает генерировать ответ, в случае заинтересованности. После генерации ответа если показатель заинтересованности достаточно высок, он приближается к активному диалогу и присоединяется к нему, становясь в очередь. Когда цепочка доходит до него, он начинает общаться с игроком и группой других персонажей по средствам диалоговой структуры, тогда он и переходит из второго состояния в третье. Третье состояние так же активируется в случае, если игрок начинает общаться с NPC, в этом случае он просто начнет обсуждать заготовленные темы.

3.2.1 Личные характеристики

Для придания большей живости, каждый персонаж обладает рядом характеристик, уникальных только для него. Это сделано для того, чтобы увеличить вовлеченность игрока в игровой мир и для введения большего разнообразия при общении. Для воплощения личностных качеств была взята одна из известных моделей, описывающих личность человека – Большая пятерка “The Big-Five Trait Taxonomy: History, Measurement, and Theoretical Perspectives” за авторством Оливера П. Джона и Санджай Шриваставы [13]. Она состоит из 5 черт, характеризующих личность человека:

1) Открытость к опыту.
2) Добросовестность.
3) Экстраверсия.
4) Доброжелательность.

5) Нейротизм.

Все эти пять характеристик представлены в алгоритме поведения NPC и отвечают за то, как персонаж поведет себя при реакции на диалог. Данная модель личности была имплементирована в диалоговую систему, поскольку она является одной из наиболее общепризнанных моделей описания характеристик человека. Более того, данная модель личности ранее уже была успешно реализована в других игровых работах, посвященных анализу поведения и взаимодействия неигровых персонажей друг с другом, упомянутых ранее в данной работе [8].

В данной системе характеристики представлены в виде переменных, хранящих в себе число с плавающей запятой. Значение каждой из переменной может варьироваться от 0.0 до 1.0.

Персональные характеристики применяются в том случае, когда персонаж анализирует рассказ другого NPC неподалеку. Он смотрит на то, какое личностное качество используется в тексте и сравнивает его со своим.

3.2.2 Темы для разговоров

У каждого NPC есть переменная, которая хранит все темы, на которые он может провести разговор, игрок, начиная общение, может выбрать одну из заранее заготовленных тем в качестве общения, после чего персонаж начнет рассказывать истории. Сами темы имеют свою собственную структуру по который неигровых персонажи могут анализировать рассказы друг друга. Структура имеет следующий вид:

1) Айди.
2) Тема.
3) Заданная черта.
4) Теги.
5) Диалоговые линии.
6) Полный текст диалога.

Айди – идентификатор, он хранит в себе числовое значение с номером данной темы, на случай если надо будет найти конкретную тему среди всего

списка

Тема – название темы о которой будет идти речь, эта переменная нужна для того, чтобы кратко передать суть того, о чем весь разговор. Это параметр является частью составного запроса, который формируется для чат-бота.

Заданная черта – параметр, который может хранить в себе одну из индивидуальных характеристик. Он показывает, какая именно черта характера задействуется в текущем рассказе действующего лица.

Теги – группа параметров, которая может хранить в себе несколько ключевых слов, по которым данная тема анализируется другими персонажами. Эта группа параметров в первую очередь проверяется иными участниками диалога на тему интереса, если находятся совпадения с один из тегов другого персонажа, то это значит, что имеется схожая тема.

Диалоговые линии – группа параметров, в которые записываются сам диалог. Каждый из параметров передается диалоговой системе для отображения текста.

Полный текст диалога – содержит в себе весь текст из параметров dialogue lines в рамках одной переменной, на случай если весь диалог понадобится передать в чат-бот.

Все параметры этой структуры заполняются вручную дизайнером и в последствии передаются в другие элементы структуры модели, такие как, чат-бот или диалоговая система.

Таблица 5 – Структура диалога

ID	**Тема**
1	2
Theme	I caught golden fish

Продолжение таблицы 5

1	2
Fixed traits	Extraversion
Tags	fish, gold, fishing
Dialogue lines	You won't believe what happened with me recently!
	I was fishing as usual. When suddenly my road started shaking
	I immediately began to pull it and when I caught the fish I was shocked
	It was most rare fish ever known. The gold one!

В таблице 5 представлен пример одной из диалоговых тем, которая была использована при тестировании модели.

3.2.3 Слышимость

Каждый персонаж обладает компонентом слышимости, который отвечает за то, насколько NPC громко говорит. Он обладает свой собственной коллизией с радиусом, который можно регулировать. Все персонажи, которые попадают в пределы действия компонента, добавляются в группу параметров, отображающих количество персонажей, находящихся в радиусе действия. Как только NPC попадает в зону видимости компонента, то он сразу же включает алгоритм, который анализирует теги из темы вошедшего персонажа с тегами темы которая в данный момент активна.

Данный компонент становится активен только тот момент, когда персонаж начинает общение на одну из своих заданных тем и становится неактивным тогда, когда общение заканчивается. Наличие объекта,

отвечающего за распространение звука очень важно, так как звуковой дизайн отлично усиливает эффект погружения игрока в игровой мир. В исследовании за авторством Патрика Нг и Кейт Несбитт “Informative Sound Design in Video Games” [14] рассказывается о важности звуков в игровом пространстве.

3.3 Структура чат-бота

В данной модели в качестве чат-бота использует Chat-GPT3, однако, в будущих версиях планируется ввести возможность заменять разные модели чат-ботов.

В этой работе нейросеть используется для того, чтобы комментировать темы разных собеседников перед тем, как переходить к собственному рассказу. Нейронные технологии все более активно применяются в различных сферах в компьютерных играх[12] и написание диалогов не исключение.

Структурно чат-бот состоит следующим образом:

1) Чат менеджер.
2) ChatGPT.
3) Чат контроллер.
4) Хранилище чата.

Это менеджер, который хранит в себе экземпляр чат-бота, через который любой другой объект может получить доступ к нейросети. Сам же чат менеджер прикреплен к объекту на игровой сцене, доступ к которому может получить любой объект.

Это класс, который занимается обработкой полученного запроса, генерируя на его основе ответ с учетом того опыта, что модель обработчик уже получила. Сам этот класс состоит из следующих элементов:

1) Конечная точка запроса.
2) Токен доступа.
3) Ключ доступа.
4) Модель.
5) Экземпляр контролера.

Конечная точка запроса – адрес по которому обработчик получается

доступ к ресурсам, требующимся для обработки запроса

Токен доступа – адрес с доступом к базе данных токенов для лучшего составления запросов

Ключ доступа – токен безопасного доступа, он позволяет использовать расширенные функции языковой обработки.

Модель – натренированная модель нейросити для обработки запросов. Для разных ситуаций используются разные натренированные модели. В нашем случае это ChatGPT-3.

Экземпляр контролера – экземпляр чата контроллера, данные и алгоритмы которого используются для настроек работы чат-бота и хранения полученных данных.

Чат контроллер – структура, которая хранит данные, обработанные чат-ботом, и параметры которой используются для настроек нейросети. Имеет следующую структуру:

1) Чат контролер.
2) Контроллер персонажа.

Чат контролер – основной класс-конструктор, который хранит в себе все базовые алгоритмы хранения и передачи готового запроса. Через него создаются все экземпляры контроллера персонажа. Именно через него отправляются данные в чат хранилище после финальной обработки. Он является главных связующим звеном между обработчиком запроса и хранилищем настроек для чат-бота.

Контроллер персонажа – группа дочерних классов чата контролера. Они могут изменять и подстраивать под себя принцип работы базовых алгоритмов родительского класса с обработанным запросом для разных нужд. В нашем случае они используются для отправки данных NPC в чат-бот.

Используется для хранение отправленных запросов в нейросеть и полученных выходных ответов. Через эту структуру задаются настройки для чат-бота для генерации ответа. Имеет следующую структуру:

1) Очистка истории на старте.

2) Максимальное количество отправлений.

3) Температура.

4) Максимальный токен.

5) Тренировки.

6) Сообщения.

Очистка истории на старте – параметр стирающий прошлые запросы после каждого повторного запуска программы. Он может быть выключен или включен.

Максимальное количество отправлений – количество запросов и ответов, которое будет храниться в хранилище.

Температура – параметр, отвечающий за то, насколько точно нейросеть будет стараться подстроиться под запрос. Число с плавающей запятой, варьируется от 0.0 до 1.0.

Максимальный токен – Максимальное количество токенов, которое может сгенерировать нейросеть.

Тренировки – заранее заданный текстовый шаблон, в рамках которого чат-бот будет стараться отвечать.

Сообщения – хранилище для запросов от персонажа и ответов от нейросети.

В таблице 6 представлен пример того, как выглядит структура хранилища. Для каждого персонажа создан свой собственный экземпляр.

Таблица 6 – Пример хранилища чата для одного из персонажей

Clear history on start	Yes
Max send count	10
Temperature	0.7
Max tokens	1
Trains	System
	You are farmer in fantasy world
Role	user
Content	React to this text 'I caught golden fish' in 10 words
Role	assistant
Content	That's an extraordinary catch! The treasure of the sea awaits!

3.4 Диалоговая структура

Диалоговая структура является главным менеджером, который связывает всех персонажей и чат-бота. Именно сюда передаются все диалоги от NPC и сгенерированный комментарий нейросети. Сама структура состоит из двух элементов:

1) Диалоговое окно.
2) Игровой персонаж.

Диалоговое окно – интерфейс, который воспроизводит текст и комментарий от NPC. При включении он анализирует какие данные были ему

переданы для воспроизведения. После чего он задает параметры того, как и в каком количестве будет отображаться текст для игрока. С заранее заданным параметром скорости он отображает полученный на вывод текст до тех пор, пока заданное количество панелей не будет отображено или пока диалог не будет прерван по инициативе игрока. В этом объекте определены алгоритмы прерывания общения или добавления новых тем в список проигрываемых диалогов.

Игровой персонаж – объект, управляемый игроком, через который происходит взаимодействие с NPC, он является начало в цепочки обработки диалоговой структуры. Когда объект игрока оказывается в зоне видимости других персонажей, он может взаимодействовать с этими лицами. В этом случае он вызовет алгоритм из диалогового окна, который задает количество панелей, беря текст из параметров NPC, а у персонажа включится объект, который отвечает за слышимость. Здесь же игрок может прервать общение, отойдя на достаточно большое расстояние или же нажав на соответствующую кнопку. Тогда NPC начнут запоминать точку, на которой остановился их диалог. Если игрок вернется к общению спустя небольшой промежуток времени, то персонажи продолжат рассказ с той точки на которой они остановились. Различные исследования на тему взаимодействия игрока с NPC[13] показывают, что чем больше у игрока есть способов проявлять активность по отношению к персонажам, тем большую вовлеченность он испытывает. Соответственно, добавление большей интерактивности и свободы действия со стороны игрока в вышеописанную систему положительно скажется на погружение в процесс.

3.5 Принцип работы модели

Поскольку теперь весь ход диалога не создан заранее дизайнером, а происходит в реальном времени и общение теперь затрагивает случайных прохожих, стоит подробно продемонстрировать, как все выше описанные структуры работают вместе друг с другом, представляя полную картину работы итоговой модели. Всю модель можно разбить на две составляющие:

как NPC взаимодействуют с другими персонажами и как игрок взаимодействует с персонажами. Основная концепция взаимодействия двух и большее персонажей друг с другом построена на анализе интересов каждого из персонажей в совокупности с личными характеристиками. Взаимодействие же игрока с NPC построено на различных возможностях, которые доступны управляемым персонажем, и на конечных ответных реакциях со стороны неигрового персонажа. Ниже на рисунке 8 представлена схема того, как NPC реагируют на действия друг друга во время диалога.

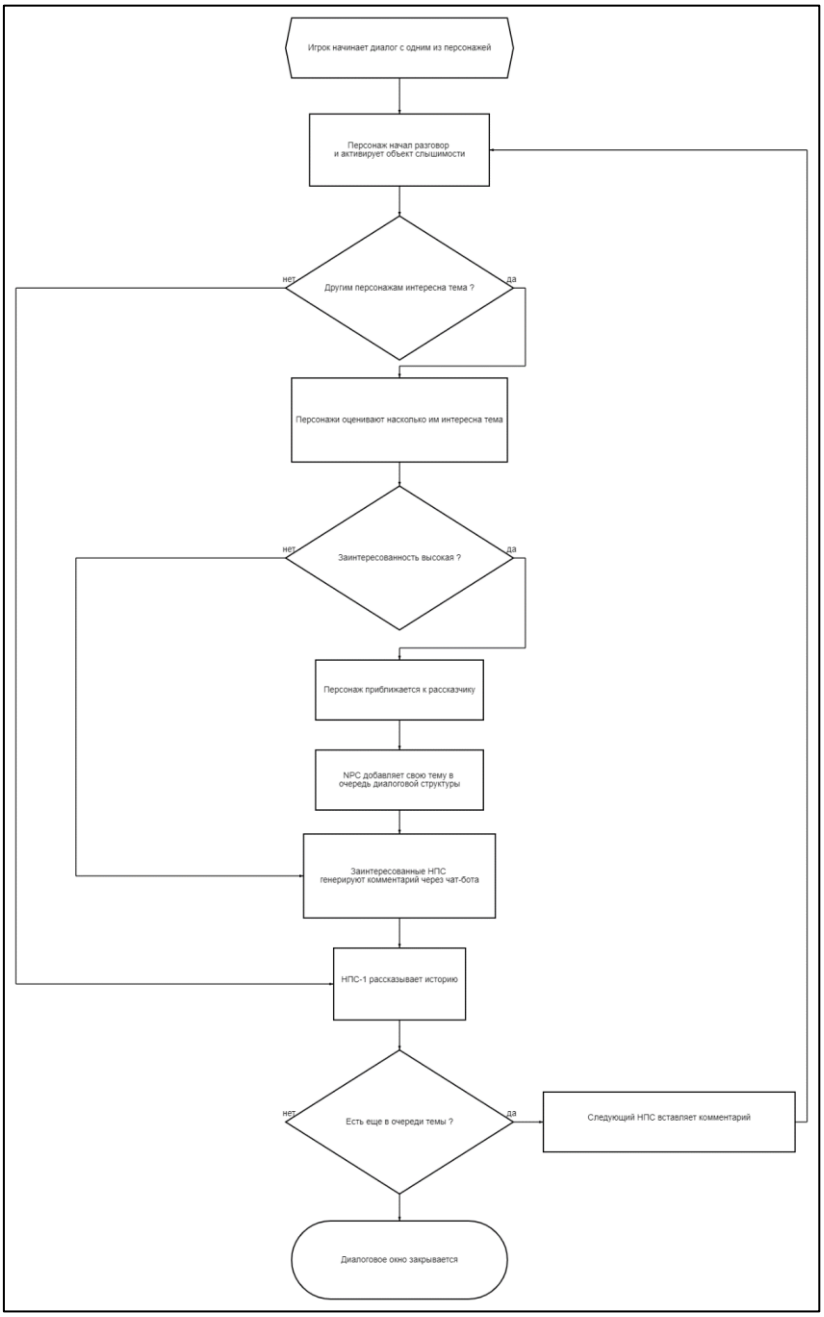

Рисунок 8 – Взаимодействие NPC

Далее ниже на рисунке 9 представлена схема непосредственного взаимодействия игрока с различными NPC

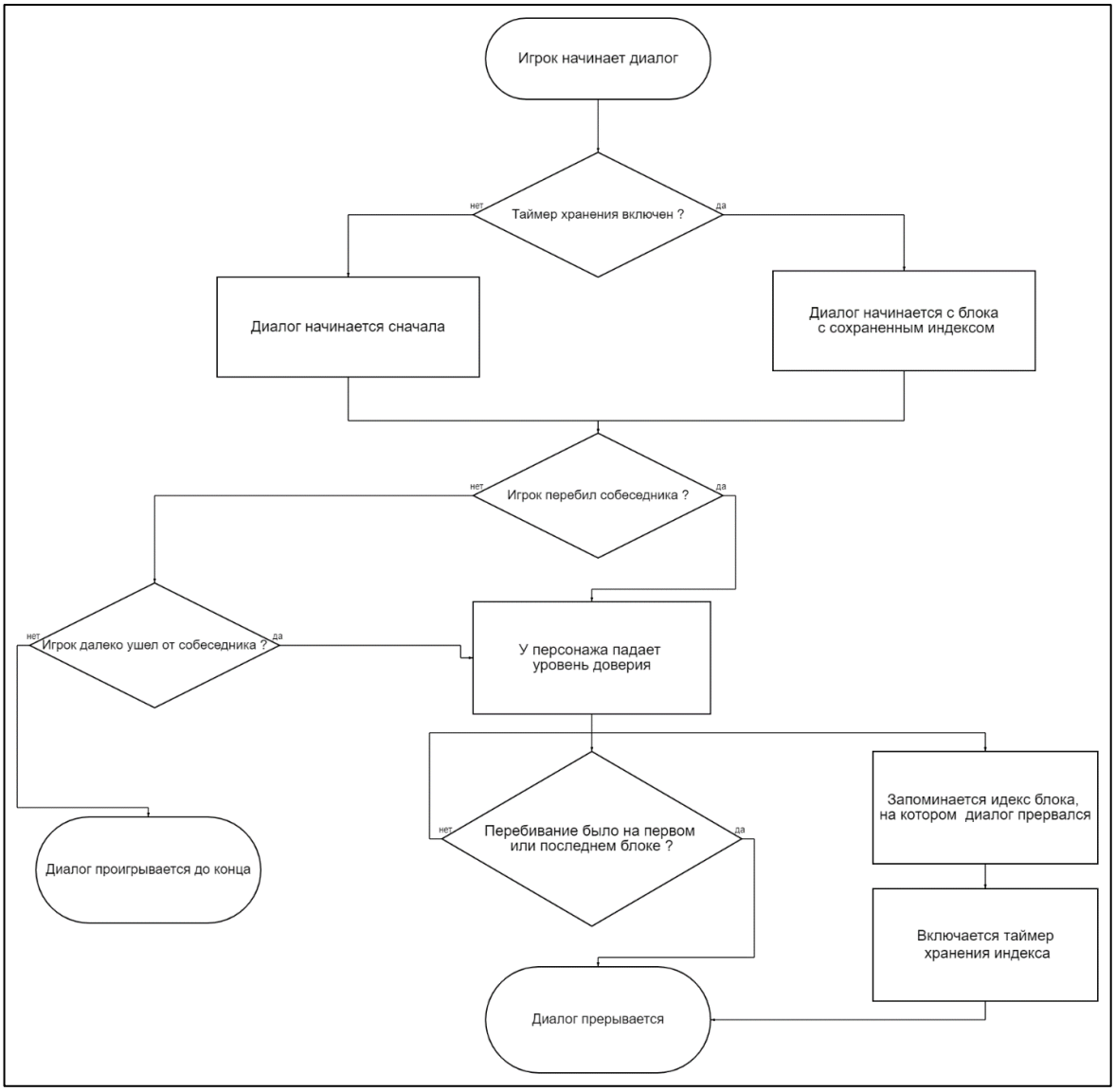

Рисунок 9 – Взаимодействие игрока с NPC

Реализация модели осуществлялось с помощью языка программирования C# и игрового движка Unity, поскольку он является одним из самых успешных и перспективных движков на рынке в данный момент [15]

3.6 Реализация NPC

3.6.1 Реализация структуры диалогов

Для начала было принять заняться реализацией тем для разговоров, поскольку именно на точках интереса в них в первую очередь завязано взаимодействие между персонажами. Класс, хранящий в себе диалоги должен иметь в себе все переменные и параметры необходимые для анализа и

сравнения тем друг с другом. Среди этих параметров должны быть теги, по которым персонажи определяют интересен ли им этот диалог или нет. Помимо ключевых слов должен быть еще и параметр, отвечающий за персональный показатель, который показывал бы интенсивность заинтересованности. Этот параметр должен в себе на выбор одну из пяти черт, характеризующих человека. Такой класс должен иметь свой собственный id номер, по которому можно будет найти нужную диалоговую тему среди сотен разных диалогов. Сам диалог, заранее записанный дизайнером и название темы.

Поскольку в классе есть переменная, обозначающая какая именно черта характера задействована при воспроизведении диалога, то в таком случае должен быть и алгоритм, позволяющий NPC извне узнать, эту черту и сопоставить ее со числовым показателем аналогичной черты внутри своего класса:

```
public float ChoseTrait(TraitsType fixedTrait, float choosenTrait, float
oppennes, float conscientiousness, float extraversion, float agreeableness,
float neuroticism)
    {
        switch (fixedTrait)
        {
            case TraitsType.Oppennes:
                choosenTrait = oppennes;
                return choosenTrait;

            case TraitsType.Conscientiousness:
                choosenTrait = conscientiousness;
                return choosenTrait;

            case TraitsType.Extraversion:
                choosenTrait = extraversion;
                return choosenTrait;
                ;
            case TraitsType.Agreeableness:
                choosenTrait = agreeableness;
                return choosenTrait;

            case TraitsType.Neuroticism:
                choosenTrait = neuroticism;
                return choosenTrait;

            default:
                return choosenTrait;

        }
          }
```

В алгоритм посылается одна внутренняя переменная с определением

используемой черты и шесть внешних переменных от стороннего NPC. Пять из них являются всеми его персональными чертами и шестая переменная, которая принимает в себя значения одной из черт характера. Через внутреннюю переменную определяется какая именно черта задействована в диалоге, и потом значение этой переменной от стороннего NPC передается в бутылку этого персонажа для дальнейшей обработки. После того, как нужная черта была найдена, алгоритм прекращает свою работу.

Окончательная структура диалогов имеет вид, представленный на рисунке 10.

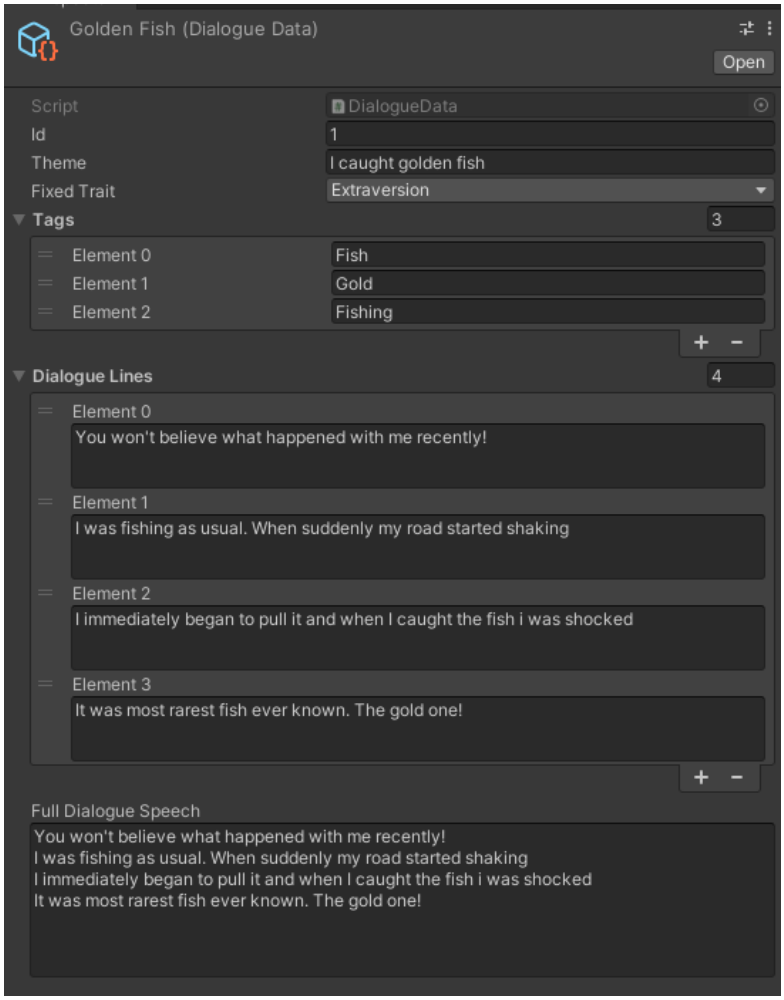

Рисунок 10 – Структура диалога

3.6.2 Реализация NPC

После реализации структуры диалогов, можно преступить к непосредственному созданию алгоритма, который будет считать отношение персонажа к услышанному диалогу[16]. Этот алгоритм имеет следующий вид:

```
public void FindsInterestingThemes(NpcHearing npcsVoice)
    {
        string npcTheme;

        if(dialogue == null)
        {
            dialogue = Dialogue.instance;
        }

        Npc npc = npcsVoice.transform.parent.GetComponent<Npc>();
        DialogueData npcData = npc.listOfDialogues[0];

        for(int i = 0; i < listOfDialogues[0].tags.Length; i++)
        {
            for(int y = 0; y<npcData.tags.Length; y++)
            {
                if (listOfDialogues[0].tags[i] == npcData.tags[y])
                {
                   choosenTrait = npcData.ChoseTrait(npcData.fixedTrait,
choosenTrait, opennesToExperience,
                      conscientiousness, extraversion, agreeableness,
neuroticism);

                    if(choosenTrait == 1)
                    {
                        npcToFollowPosition =
npcsVoice.transform.parent.position;
                        dialogue.queueToPlay.Add(this);
                        goToOtherNPC = true;
                        npcTheme = npcData.theme;
                        reactionToTheme = "React to this text " + "'" +
npcTheme + "'" + " in 10 words";
                        GenerateGPTDialogue(reactionToTheme);
                        return;
                    }
                    else
                    {
                        npcTheme = npcData.theme;
                        reactionToTheme = "Tell you have heard a story about:
" + "'" + npcTheme + "'" + " in 10 words";
                        GenerateGPTDialogue(reactionToTheme);
                        hasComment = true;
                        return;
                    }
```

Алгоритм работает следующим образом: сначала через звуки диалога определяется место положение его источника. После этого, через данный источник мы получаем доступ к списку, хранящему все темы, на которые персонаж способен разговаривать. Находим в данный момент активный диалог. После чего в цикле проходимся по каждой хранящейся у NPC, который услышал диалог, теме и сравниваем все его теги со всеми тегами активного диалога. Если находится хотя бы одно пересечение, то значит есть общие

точки интереса. В этом случае мы через активный диалог активируем алгоритм нахождения использующейся черты характера, который был представлен на рисунке 3. Полученное значение мы проверяем на диапазон. Если больше половины, то значит персонаж сильно заинтересован в диалоге, и мы даем ему указание идти в сторону рассказа, после чего NPC составляет запрос для чат-бота и отправляет его в алгоритм генерации комментария. В противном случае персонаж просто запоминает услышанную речь, составляет запрос для комментария в нейронной сети, и хранит ее в соответствии со значением, хранящемся в переменной “время”.

Каждый NPC в модели обладает объектом “Слышимость”, который обладает своим радиусом действия, и через который другие персонажи получают доступ к активному диалогу. Это объект обладает своим собственным алгоритмом, который вступает в действие, когда какое-либо другое активное лицо вошло в его радиус. В нем проверяется участвует ли уже этот персонаж в диалоге и если нет, то как раз через этот алгоритм вызывается анализ интересных точек пересечения. Таким образом, данный объект выступает посредником между двумя разными персонажами:

```
if (other.gameObject.tag == "NPC")
        {
            if (other is CapsuleCollider)
            {
                if (other.GetComponent<Npc>().goToOtherNPC == false)
                {
                    other.GetComponent<Npc>().FindInterestingWords(this);
                }
            }
          }
```

3.7 Реализация нейронной сети

Поскольку в алгоритме поиска точек интереса персонаж составляет запрос для чат-бота нейронной сети, то нам так же необходимо определить алгоритм работы чат-бота. В данной модели в качестве API используется Chat-GPT3, поскольку она является одной из самых быстро растущих моделей на рынке. Однако, реализуемая диалоговая модель предполагает возможность заменять различные решения и модели обработки текста, на случай если использующееся в данный момент API не удовлетворяет запросам.

Нейросеть была настроена таким образом, что каждый персонаж на сцене обладает своими уникальными настройками под запросы нейронной сети. Это сделано для придания большей уникальности, поскольку неоднократные исследования на тему улучшения восприятия от игрового мира показывают, что большее количество уникальных ситуаций ведет к большему интересу от получаемого процесса, что в свою очередь больше погружает в игровое пространство. Каждый раз, когда-то кто-то отправляет запрос на обработку в чат-бота, тот берет эти индивидуальные настройки и просматривает их для более лучшего контекста. Более того, если персонаж раньше уже отправлял запросы в нейронную сеть, то эти запросы и выходящие ответы сохраняются в этих же параметрах и учитываются при будущих ответах, таким образом позволяя создавать ощущение, что персонаж запоминает прошлое общение. Пример таких уникальных настроек представлен на рисунке 11.

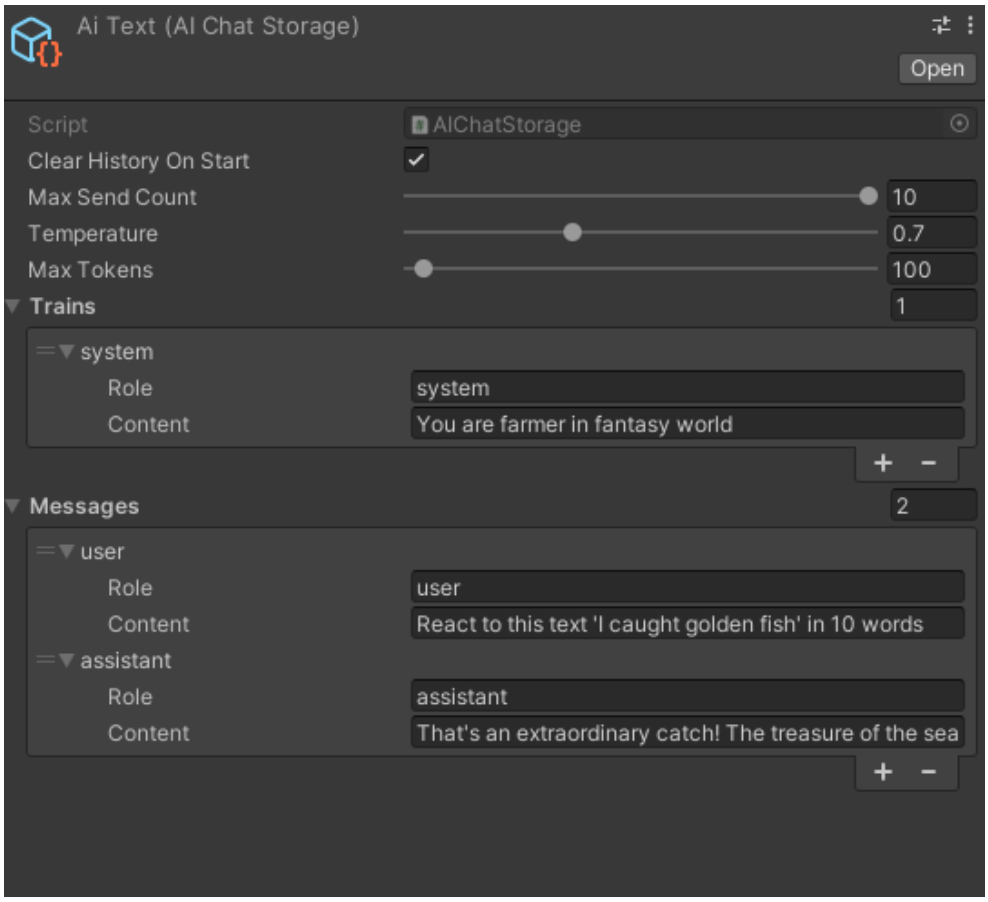

Рисунок 11 – Структура хранения настроек для нейронной сети

Сама же нейронная сеть берет эти данные через персонажа, который отправляет запрос, и задает в своих настройках:

```
var sendMessages = new List<AIMessage>(controller.chatStorage.trains);
            sendMessages.AddRange(

controller.chatStorage.messages.TakeLast(controller.chatStorage.maxSendCount)
.ToList());
            var requestBody = new AIRequestBody
            {
                model = GetModelName(egptModel),
                messages = sendMessages,
                temperature = controller.chatStorage.temperature,
                max_tokens = controller.chatStorage.maxTokens
                  };
```

В зависимости от этих настроек финальный комментарий может получить разный текст при одинаковом запросе на рисунках 12 и 13 показаны примеры ответа от персонажа на один и тот же запрос, но с двумя разными настройками.

Рисунок 12 – Пример комментария при одной настройке

В поле "Content" задается условие по которому бот будет генерировать ответы. При одной настройке он задает ответ, показанный на рисунке выше.

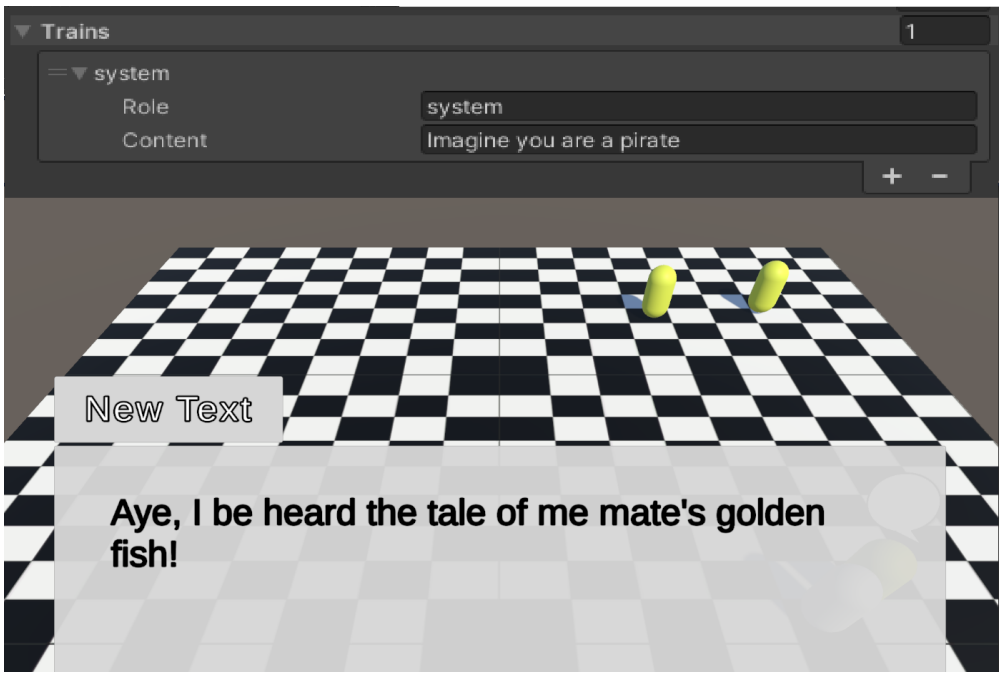

Рисунок 13 – Пример комментария с другими настройками

При иначе заданным параметром "Content" персонаж уже комментирует совсем иначе.

3.8 Реализация диалога NPC

Следующим этапом в работе модели является непосредственная интеракция игрока с окружением и отображение этого взаимодействия. Необходим алгоритм, который смог бы принимать от персонажей их заранее заготовленные фраза и отображать их будто они в настоящее время произносят слова [17]. Для этого нужен объект, который вызывался бы при наступлении конкретного события общения с NPC. Сам алгоритм бы вызывался при включении этого объекта и принимал бы содержание и количество фраз, отображая их при этом с конкретной скорость. Ниже представлен алгоритм того, как задаются начальные настройки у окна отображения и как он показывает текст в реальном времени, создавая иллюзию общения:

```
npcComments =
queueToPlay[0].GetComponent<AITestController>().finalGeneratedPhrase;
textComponent.text = string.Empty;
index = tempIndex - 1;
inDialogue = true;

StartCoroutine(TypeComment(npcComments));
```

```
      foreach (char c in lines[index].ToCharArray())
              {
                  textComponent.text += c;
                  yield return new WaitForSeconds(textSpeed);
              }
```

Так же нужен алгоритм, который будет понимать какой текст конкретно он должен показывать, какого он должен быть размера и с какой точки отображать. Для этого ему нужно взять данные у активного NPC и сохранить в свои временные переменны до тех пор, пока текст не отобразился, после этого он может очищать это параметры и принимать в них новые диалоговые данные:

```
public void SetDialogue(Dialogue dialogueWindow, Npc npcParameters)
    {
        DialogueData npcDataOfDialogues;

        if(npcParameters.GetComponent<Npc>().triggeredDialogue > -1)
        {
            npcDataOfDialogues =
npcParameters.GetComponent<Npc>().listOfDialogues[npcParameters.GetComponent<
Npc>().triggeredDialogue];
        }
        else
        {
            npcDataOfDialogues =
npcParameters.GetComponent<Npc>().listOfDialogues[0];// 0 replace with
correct index in future
        }

        dialogueWindow.GetComponent<Dialogue>().lines = new
string[npcDataOfDialogues.dialogueLines.Length];

        for (int i = 0; i < npcDataOfDialogues.dialogueLines.Length; i++)
        {
            dialogueWindow.GetComponent<Dialogue>().lines[i] =
npcDataOfDialogues.dialogueLines[i];
        }

        if (npcParameters.GetComponent<Npc>().tempIndex != 0)
        {
            dialogueWindow.GetComponent<Dialogue>().tempIndex =
npcParameters.GetComponent<Npc>().tempIndex;
        }

        if (npcParameters.gameObject.GetComponent<ResetPos>() != null)
        {
            npcParameters.gameObject.GetComponent<ResetPos>().speed = 0;
        }

        npcParameters.GetComponent<Npc>().dialogueBubble.SetActive(true);
        npcParameters.voiceSpeech.SetActive(true);

npcParameters.GetComponent<Npc>().dialogueInteractionIcon.SetActive(false);
        currentlyActiveNPC = npcParameters;
          }
```

3.9 Реализация взаимодействия игрока

Последней составляющей модели является игровой персонаж, без которого сама система не будет работать. Он должен обладать возможностью взаимодействовать с персонажами, тем самым давая знать диалоговому окну, что оно должно взять данные из NPC, с котором игрок взаимодействует. Для более лучше понимания с кем конкретного идет диалог в данный момент, каждый персонаж имеет объект, обозначающий речь. Он включается в тот момент, когда игрок начал взаимодействовать с персонажами. При нахождении в нужном радиусе от персонажа и при нажатии на конкретную заданную кнопку, игрок активирует диалог с NPC который по цепочке за собой активирует все последующие алгоритмы системы:

```
   if (_canSpeak)
        {
            if (Input.GetKeyDown(KeyCode.X) && npcParameters != null &&
dialogueWindow.GetComponent<Dialogue>().inDialogue == false)
            {

dialogueWindow.GetComponent<Dialogue>().SetDialogue(dialogueWindow.GetCompone
nt<Dialogue>(), npcParameters);


dialogueWindow.GetComponent<Dialogue>().queueToPlay.Add(npcParameters);
                dialogueWindow.SetActive(true);
            }

            if (Input.GetKeyDown(KeyCode.Z))
            {
                if (dialogueWindow.GetComponent<Dialogue>().inDialogue ==
true)
                {

dialogueWindow.GetComponent<Dialogue>().InterruptDialogue(npcParameters.GetCo
mponent<Npc>());
                }
            }

            if (dialogueWindow.GetComponent<Dialogue>().inDialogue == false)
            {

npcParameters.GetComponent<Npc>().dialogueBubble.SetActive(false);

npcParameters.GetComponent<Npc>().dialogueInteractionIcon.SetActive(true);
            }

              }
```

Кроме начала диалога, игрок может еще и прерывать резко общение и переходить к другому собеседнику. В такие моменты диалоговое окно запоминает точку остановки в диалоге, чтобы продолжить в будущем общение

с этого места. А персонаж включает таймер, на протяжении которого он помнит точку диалога, но по истечению, которого он забывает об этом общении и при следующей встрече начнет общение по этой теме с нуля. Это алгоритм активен только в тот момент, когда прерывание произошло не на первой фразе и не на второй, в противном случае запоминания не происходит, поскольку собеседник либо еще ничего не успел рассказать, либо уже поведал всю историю:

```
public void InterruptDialogue(Npc npc)
    {
        if (index != 0 && index != lines.Length - 1)
        {
            npc.trustLevel--;
            npc.tempIndex = index;
            npc.isRememberLine = true;
        }

        inDialogue = false;
        npc.voiceSpeech.SetActive(false);

        afterCommentQueue.Clear();

        foreach (var character in queueToPlay)
        {
            character.dialogueBubble.SetActive(false);
            character.isFoundInterestingWord = false;

            if (character.GetComponent<ResetPos>() != null)
            {
                character.GetComponent<ResetPos>().speed =
character.GetComponent<ResetPos>().defaultSpeed;
            }
        }
        queueToPlay.Clear();

        gameObject.SetActive(false);
          }
```

4 ТЕСТИРОВАНИЕ

4.1 Улучшение модели

После создания модели и первичного её тестирования, были выявлены аспекты, требующие дополнительную доработку, такие как: более глубокий анализ текста собеседника в реальном времени, прерывание диалога в цепочке из нескольких персонажей с последующим возвратом к исходной теме. Отключение возможностей чат-бот и создание комментариев, путем ручного алгоритма, создание дополнительного поведения NPC, создание более одной темы для обсуждения с персонажами, улучшение интерфейса и многое другое.

Первым делом была проведена работа по улучшению анализа текста со стороны NPC. В изначальной модели, персонажи, находящиеся в радиусе слышимости, получали для анализа на вход сразу весь текст, который планировал проговаривать собеседник. Это приводило к тому, что точки интереса могли быть найдены задолго до того, как персонаж произнесет их в слух. Это могло приводить к ситуациям, когда персонаж присоединялся в группу общения ровно в ту же секунду, как собеседник только начал общение, что выглядело слишком неестественно. Помимо этого, такое поведение со стороны NPC давало четко понять игроку, что этот персонаж точно как-то будет комментировать текущий диалог, что выбивало из погружения, поскольку паттерны поведения выглядели слишком отчетливо. Поэтому алгоритм был частично переработан, чтобы персонажи анализировали текст в реальном времени и находили точки пересечения ровна в те моменты, когда их произносит NPC. Теперь персонажу в переменную посылаются слова, которые в данный момент отображаются в диалоговом окне, после чего они сверяются с тегами у каждой из закрепленных тем. Таким образом, если присутствует пересечение между темами, то это определяется в момент того, как эта фраза была произнесена и только после этого персонаж присоединится к диалогу.

Обновленный алгоритм имеет следующий вид:

```
public void FindInterestingWords(NpcHearing npcsVoice)
    {
        if(isFoundInterestingWord == false)
        {
            var found = 0;
            if (dialogue == null)
            {
                dialogue = Dialogue.instance;
            }

            Npc npc = npcsVoice.transform.parent.GetComponent<Npc>();
            DialogueData npcData = npc.listOfDialogues[0];

            for(int x = 0; x < listOfDialogues.Length; x++)
            {
                for (int i = 0; i < listOfDialogues[x].tags.Length; i++)
                {
                    for (int y = 0; y < dialogueText.Length; y++)
                    {
                        found =
dialogueText.IndexOf(listOfDialogues[x].tags[i]);
                    }
                    Debug.Log(found);

                    if (found > 0)
                    {
                        triggeredDialogue = x;
                        Debug.Log("Found word: " +
listOfDialogues[x].tags[i]);
                        isFoundInterestingWord = true;

                        if (gameObject.GetComponent<MoveNpc>() != null)
                        {
                            gameObject.GetComponent<MoveNpc>().enabled =
false;
                            agent.enabled = true;
                        }

                        if(gameObject.GetComponent<ResetPos>() != null)
                        {
                            gameObject.GetComponent<ResetPos>().speed = 0;
                        }
                        DefineBehaviour(npcData, npcsVoice);
                        found = 0;
                    }
                }
            }
        }
    }
```

Далее была дополнена и улучшена система перебивания собеседника. В прошлой итерации перебивать собеседника мог только персонаж, управляемый игроком, что выглядело несколько неестественно. В данной вариации все заинтересованные NPC взаимодействуют со порядковым списком проигрывания реплик и могут добавлять себя выше собеседника, перебивая его, или же ниже, произнося свои реплики уже после

первоначального диалога. Работа данного алгоритма выглядит следующим образом:

```
    public void OtherNPCInterruptDialogue(Npc interruping, Npc interruped)
{
    if (index != 0)
    {
        interruped.tempIndex = index;
        interruped.isRememberLine = true;
    }

    inDialogue = false;

    foreach (var character in queueToPlay)
    {
        character.dialogueBubble.SetActive(false);
        character.voiceSpeech.SetActive(false);
    }

    interruping.goToOtherNPC = false;
    interruping.isHearOtherNpc = false;
    SetDialogue(this, interruping);
    StartComment();
}
```

Персонаж, которого перебили запоминает индекс фразы, чтобы продолжить с этого же места, когда разговор вернется к нему. В это же время для того, кто перебил задаются данные для начала диалога.

Помимо этого, была реализована возможность перебивать собеседников персонажу, с которого диалог был начат. Раньше изначальный собеседник игрока мог рассказывать только одну свою историю, в то время как другие участники спокойно имели возможность перебивать и рассказывать что-то свое. Теперь же если первого участника диалога перебить, то он сможет перебить в ответ в случае, когда он найдет для себя интересную тему для обсуждения. При этом, первая поднятая тема никуда не денется и ее продолжат обсуждать, когда все перебивания закончатся.

Была реализована возможность составлять комментарий на диалог без использования нейронной сети. Сделано это было для тех случаев, когда результаты чат-бота не устраивают разработчиков по тем или иным причинам, а также для большего контроля со стороны дизайнеров игры. Для этого у каждой диалоговой темы был добавлен параметр “Comment phrase” в который дизайнер может написать фразу. Эта фраза будет произнесена при перебивании собеседника или при комментировании его темы для придания

чувства более плавного перехода между темами.

Помимо этого, был добавлен параметр “Gap phrase”, предназначенный для плавного возвращения к перебитой теме. Чтобы диалог ощущался более естественным персонажи используют перебивки между двумя разными фразами, возвращая таким образом контекст ранее поднятой темы. Дизайнером задается в этом поле текст, который в последствии будет использоваться в диалоге. На рисунке 14 показан пример обновленной диалоговой структуры

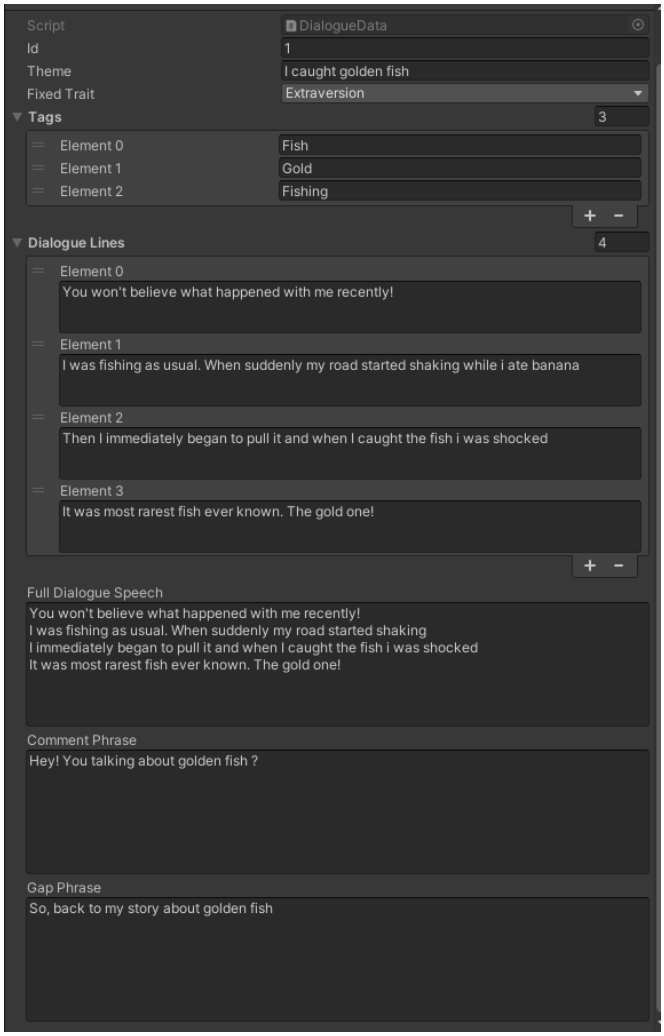

Рисунок 14 – Обновленная диалоговая система

Была дополнена система с хранением разных тем у персонажей. Ранее каждый персонаж мог комментировать только одну тему, в дополненной же системе количество тем для поиска точек интереса не ограничено. Более того, теперь NPC обладают скрытыми темами для разговоров, которые будут

подняты только в те моменты, когда они услышат для себя пересекающиеся точки интересов.

Улучшен интерфейс системы. Теперь наглядно отображается какой конкретно персонаж сейчас ведет общение с помощью значка в виде облака над головой NPC. А при приближении игрока к какому-либо из персонажей над ними будет отображаться значок взаимодействия для начала диалога, чтобы было понятно пользователю какие персонажи активны для разговора, а какие нет. Пример улучшенного интерфейса представлен на рисунках 15 и 16.

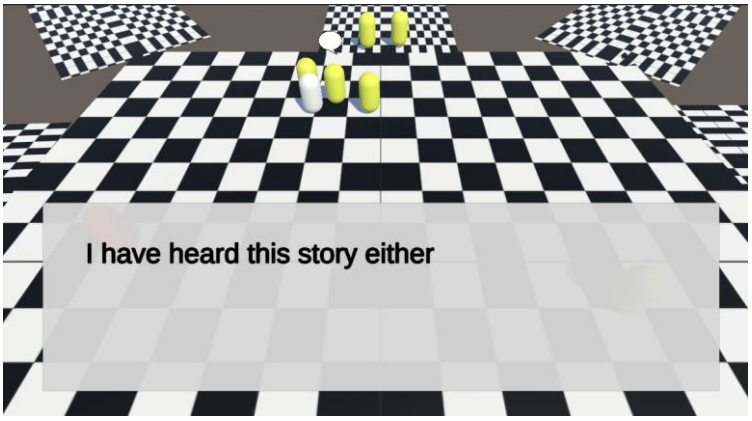

Рисунок 15 – Интерфейс активного в диалоге персонажа

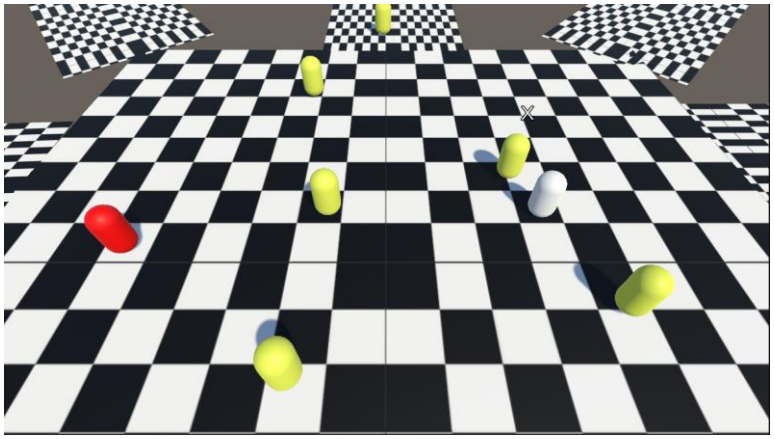

Рисунок 16 – Интерфейс взаимодействия с NPC

4.2 Проведение тестов

В рамках работы было проведено тестирование работоспособности системы. Была подготовлена сцена с некоторым количеством NPC на ней. Для каждого персонажа было задано минимум три диалоговых темы. Для некоторых персонажей было задано поведение, например, охрана точки на сцене или перемещение по локации по нескольким точкам. Также, был задан

контекст событий для игрока: ему нужно узнать в каком направлении нужно идти. Однако, персонажи просто так не хотят сообщать ему нужную информацию, поэтому игроку путем общения с группами NPC на разные темы нужно в разговоре выпутать информацию. После получения нужной информации, игроку нужно проследовать по направлению, полученному из диалогов, где он завершит “. На рисунке 17 показана сцена для тестирования.

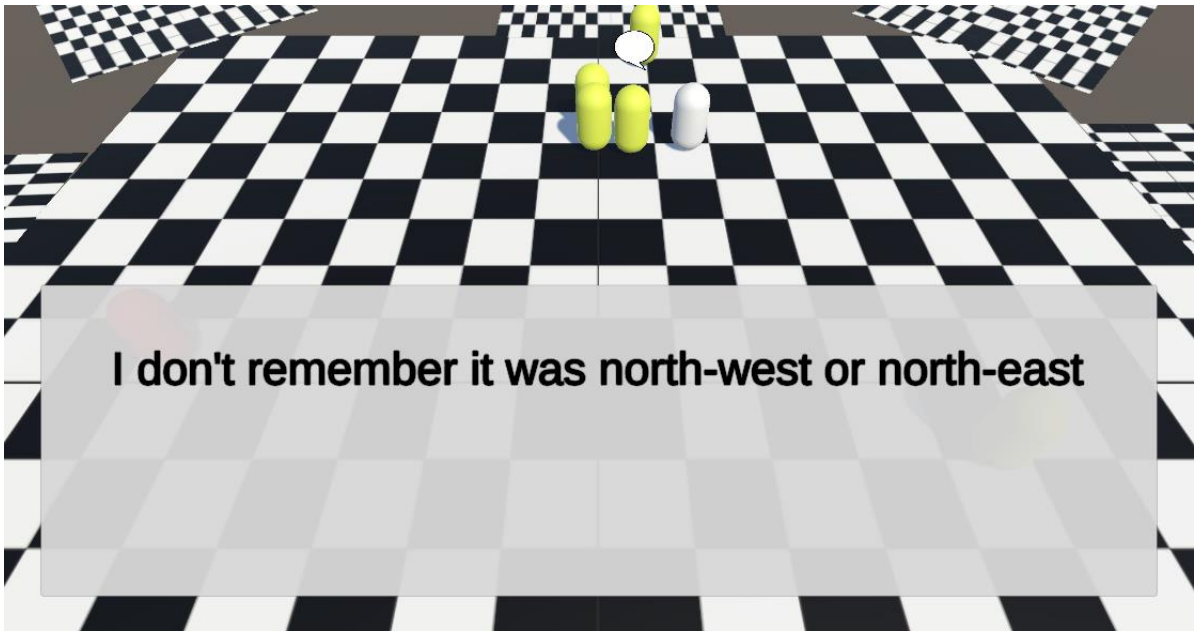

Рисунок 17 – Пример тестовой сцены

После прохождения прототипа, испытуемым предлагалось ответить на ряд вопросов, связанных с проверяемой системой.

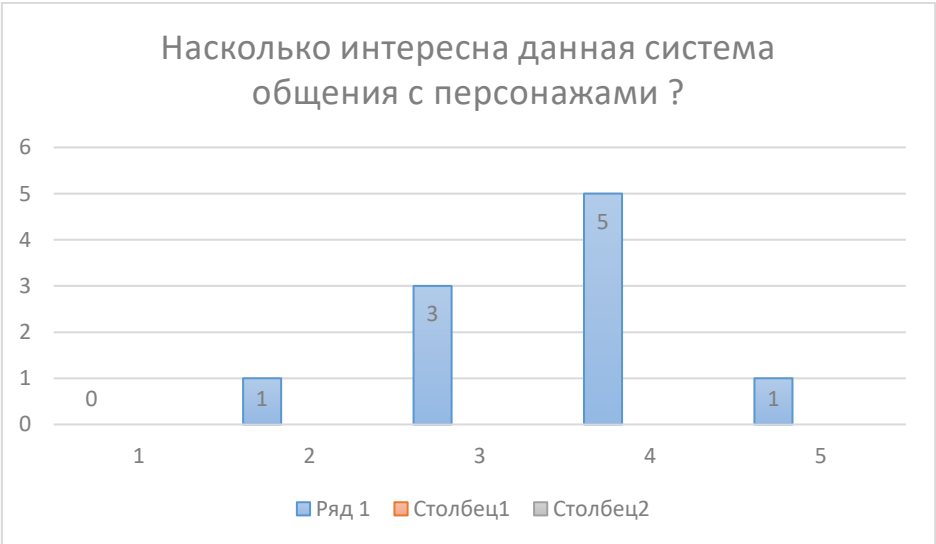

Рисунок 18 – Интерес к системе

На рисунке 18 по результатам таблицы видно, что сама концепция

системы в целом интересна большей части испытуемым.

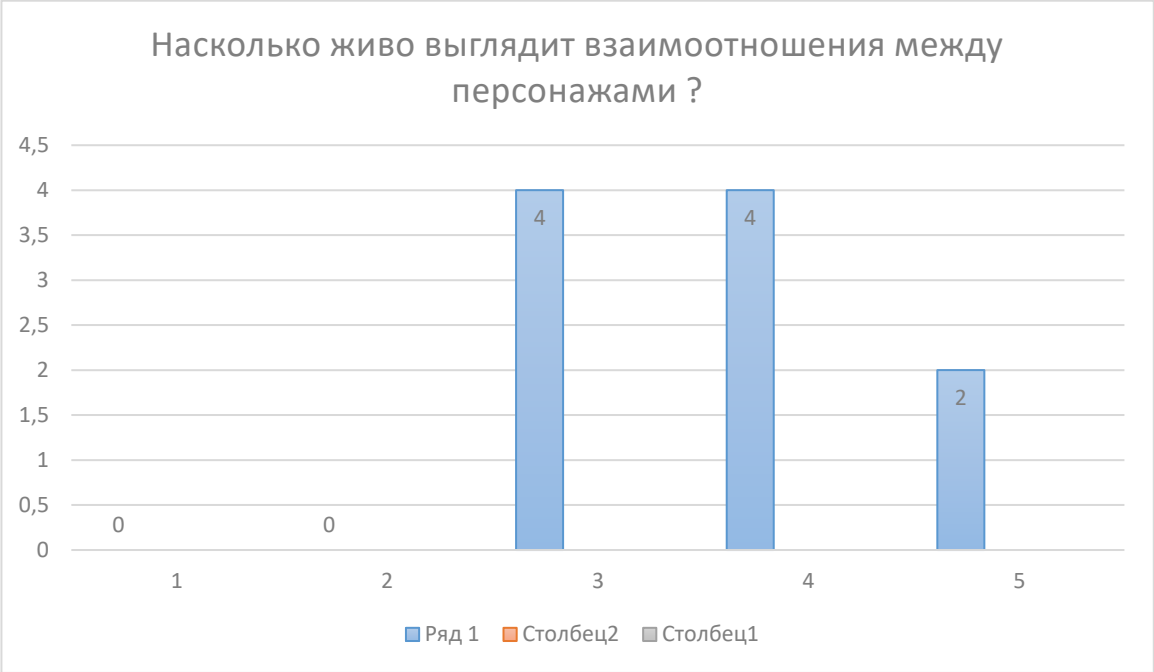

Рисунок 19 – Живость отношений между персонажами

На рисунке 19 видно, что впечатления от живости взаимодействия NPC оказались смешанными. Основная претензия заключается в том, что тем для общения слишком мало и из-за этого слишком много персонажей общаются об одном и том же.

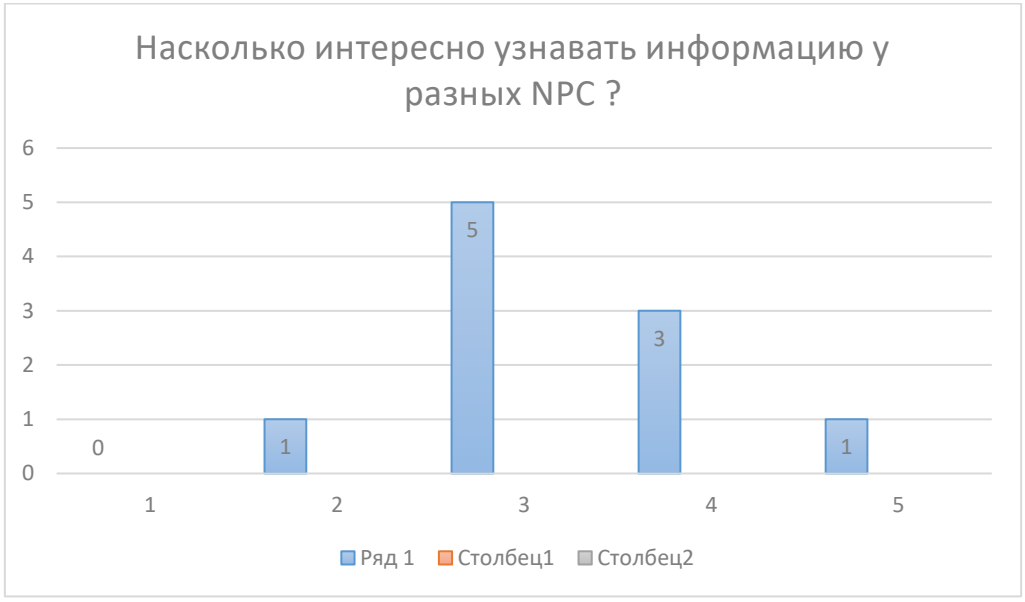

Рисунок 20 – Насколько интересно узнавать информацию

Рисунок 20 показывает, что основная претензия в получении новой информации заключается в том, что персонажи никак не выделяются друг от

друга, отчего игрок начинает путаться в том, кто что сказал. Даже при условии, что над активным персонажем есть облако диалога, когда стоит толпа других персонажей рядом, то все превращается в кашу.

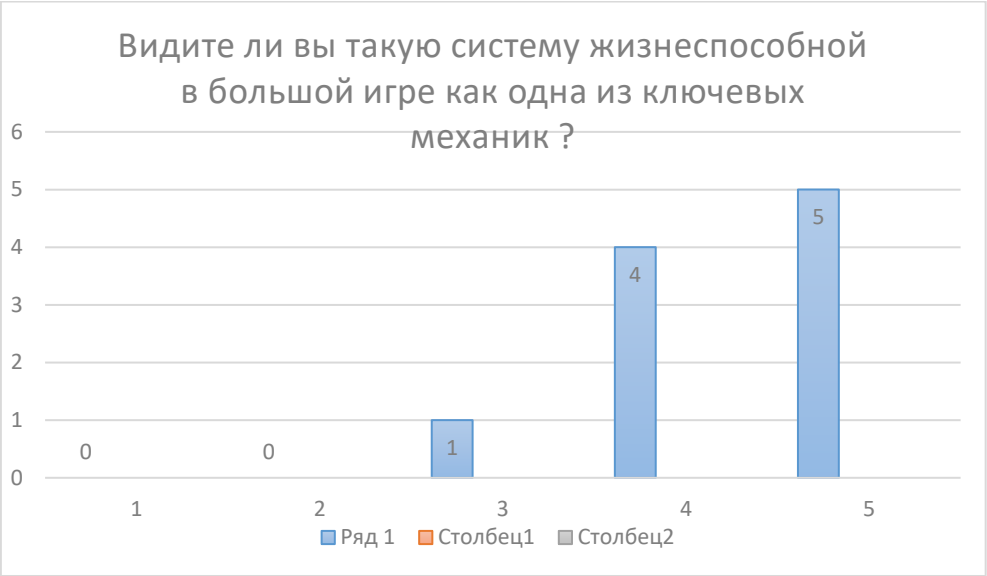

Рисунок 21 – Жизнеспособность системы

По рисунку 21 видно, что испытуемые в основном видят систему жизнеспособной в большой игре. Многие выделяют то, что в играх жанра RPG эта система смотрелась бы очень уместной в городах и в местах большого скопления разных персонажей. Часть опрашиваемых выделило, что эта система хорошо будет работать в маленьких локациях. Другие выделяют, что модель будет работать хорошо, как дополнительная система при общении с массовкой в игровом мире.

По итогам тестирования можно сделать вывод, что система рабочая и может на какой-то промежуток времени предоставлять интерес для игрока. Все испытуемые сообщили, что видят возможность реализации такой системы общения в больших полноценных играх, но расходятся в реализации модели. Одни считают, что система будет хорошо работать на больших толпах, другие же наоборот в маленьких группах. Однако, такие аспекты, как: интерфейс, разнообразие тем, наличие отличительных черт у персонажей нуждаются в доработке.

Результаты тестирования показывают, что система еще нуждается в

дополнительной полировке перед последующими проверками работоспособностями. В первую очередь нужно обратить внимание на удобство получения информации от собеседника.

4.3 Повторное тестирование

После первичных тестов было принято решение по улучшению взаимодействия пользователей с игрой. Первым делом было принято решение сделать процесс перебивания персонажей друг друга более явным для визуального считывания. Теперь, когда персонаж слышит интересующие его темы, над его головой высвечивается фраза, заинтересовавшая его.

Все персонажи на сцене были выделены разными цветом для придания большей уникальности и более легкого запоминания для игрока. Помимо этого, теперь каждый из них обладает уникальным именем, который отображается при проигрывании диалога в диалоговом окне.

Персонажи теперь могут со временем приобретать новые темы для разговоров, для этого части из них был имплементирован скрипт движения для создания имитации оживленного города, где люди время от времени возвращаются в старые места, но уже с новой информацией для разговоров. Это приводит к ситуациям, когда один и тот же NPC по-разному реагирует на один и тот же диалог игрока с другим персонажем. Было увеличено общее количество тем для общения у каждого из персонажей, а также было добавлено несколько новых.

Наконец, цвет диалогового окна подстраивается под цвет того персонажа, который в данный момент ведет диалог с игровым персонажем. Реализовано это для того, чтобы игрок мог явно определить, когда и какой персонаж перебивает разговор и влезает со своей темой.

Все эти изменения продемонстрированы на рисунке 22.

Рисунок 22 – Улучшение прототипа

После проведения работ по улучшению взаимодействия игрока и модели, было проведено повторное тестирование для наглядного сравнения с прошлой итерацией. Была отобрана группа, частично состоящая из уже ранее испытуемых и частично из новых людей. Сделано это было для проверки того, насколько приятнее стала для взаимодействия система по сравнению с прошлой версией. В то же время для того, чтобы посмотреть, как воспринимается вторая итерация теми людьми, которые ранее не сталкивались с ней. После прохождения всем испытуемым были заданы вопросы по восприятию системы, некоторые вопросы были заданы только группе, ранее игравших в первую итерацию, поскольку они были нацелены на сравнение, другие же вопросы были заданы всем тестирующим. Далее будут приведены результаты опросов на рисунках 23, 24 и 25.

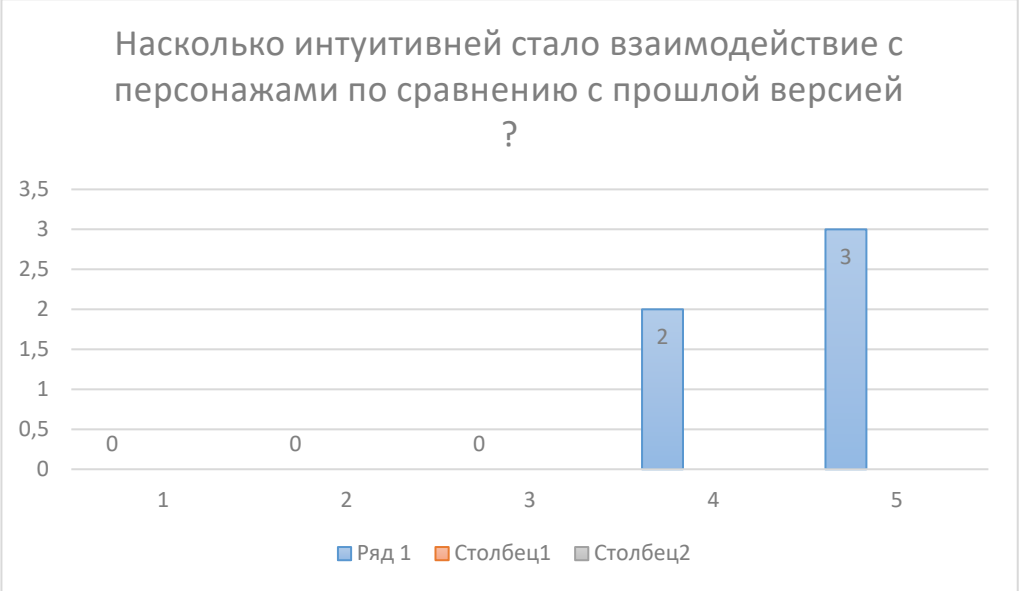

Рисунок 23 – Удобство новой итерации

Результаты первой таблице относятся только к тем тестируемым, которые участвовали в первой итерации. Как можно увидеть, все испытуемые были довольны изменениям. Из недостатков было выявлено, что в больших толпах перебивания все еще смотрятся сумбурно и не так ярко выделяются, как могли бы.

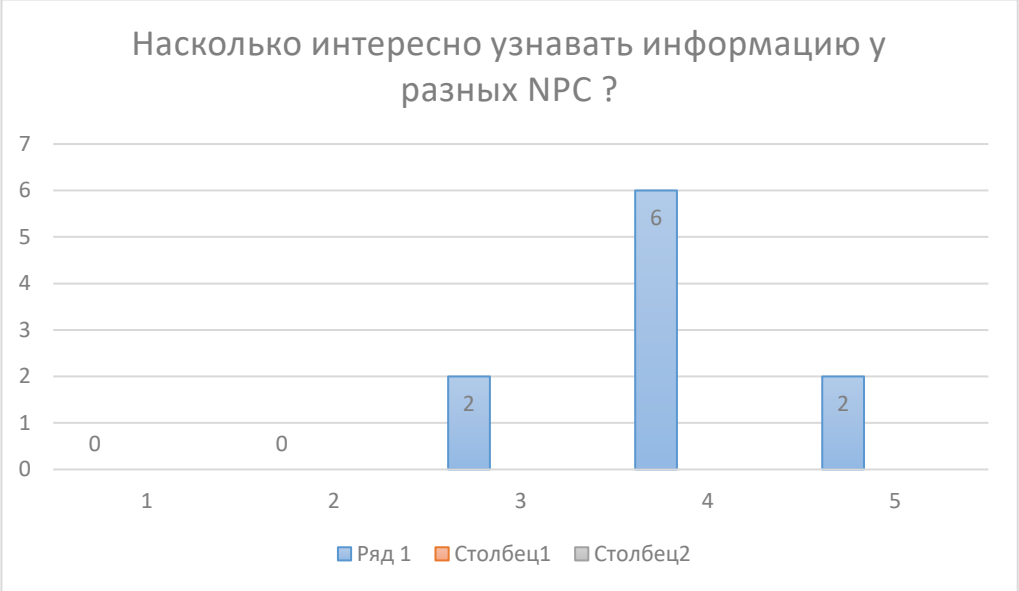

Рисунок 24 – Насколько интересно узнавать информацию

По сравнению с прошлыми показателями в аналогичном вопросе во второй итерации результаты выросли в положительную сторону. Старые испытуемые высказали мнение, что общение стало понятнее, а новые интуитивно понимали, как узнавать новую информацию через групповое

общение. Из минусов было выделено то, что переход от темы к теме местами странно смотрится и не всегда чувствуется связь между собеседниками.

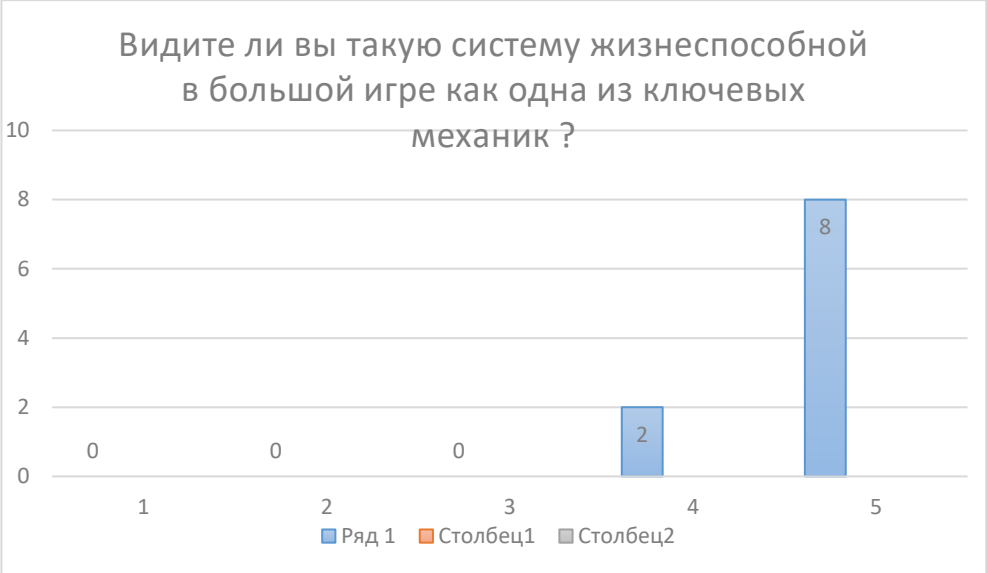

Рисунок 25 – Жизнеспособность системы

По результатам второй итерации все испытуемые сказали, что видят эту систему рабочей в полноценной игре. Опаску у некоторых тестирующих вызывало то, насколько гармонично будут смотреться разные темы в цепочке диалогов. Этот аспект сильно зависит от того, насколько грамотно написаны диалоги сами по себе.

Некоторые испытуемые выразили мнение о том, что испытывали трудности в запоминании большого количества персонажей, и предложили идею того, что персонажей можно было бы окрашивать в разные цвета уже после того, как игрок с ними общался. А до этого момента оставлять всех персонажей в одном общем цвете, например, белом. Эта деталь позволит лучше считывать окружение и понимать с кем еще стоит поговорить.

По результатам двух тестирований можно прийти к выводу, что игрокам интересно взаимодействовать с персонажами по средствам данной системы. Она усиливает вовлечение и погружение в игровой процесс.

4.4 Сравнение с подходами в других проектах

Для понимания работоспособности системы также не будет лишним сравнить реализации взаимодействия м NPC в проектах других компаний для выявления отличительных сильных особенностей. Сравнение систем

предлагается по следующим критериям:

1) Взаимодействие NPC в группе.
2) Взаимодействие группы NPC с игроком.
3) Количество вариантов поведения NPC в диалоге.

Отобранные критерии сравниваются исключительно в аспектах диалогового общения между персонажами и игрока, поскольку созданная в работе система изучает механики коммуникации NPC и игрока с окружающим их миром.

В качестве сравниваемых игр были выбраны проекты, вышедшие не более, чем 4 года назад. Это было сделано для большей объективности сравнения. Были отобраны следующие игры:

1) Baldur's gate 3.
2) Starfield.
3) Cyberpunk 2077.

Данные проекты были выбраны по причине того, что это игры с большим количеством NPC, поэтому потенциальные возможности для группового общения в этих играх довольно высоки.

Baldur's gate 3. Взаимодействие NPC в группе в данной игре можно разделить на две составляющие: коммуникация компаньонов игрока и коммуникация персонажей, населяющий игровой мир. Вариативность компаньонов в группе четко привязано к сюжетным сценам и заранее созданным сценариям. Персонажи будут общаться между собой только в созданных заранее дизайнерами сценах, где они обменяются несколькими репликами, но при этом никогда не будут комментировать темы друг друга. Каждый из компаньонов отрыве от заданного сценария никак не реагируют друг на друга и не обмениваются информацией. Групповое общение между NPC населяющими мир как таковое отсутствует. Персонажи могут стоять кучкой, но никак не будут общаться друг с другом или распространять информацию.

Взаимодействие группового общения компаньонов и персонажей,

населяющих мир в аспекте общения с игроком схоже. Интеракция с игроком фактически отсутствует, каждая интеракция игрока с NPC представляет из себя отдельную сцену, где игрок выбирает варианты для диалога с одним персонажем, в то время, как другие просто не слышат.

Варианты поведения NPC в данной игре сильно ограничены: компаньоны следуют за игроком и участвуют в диалогах с ним один на один. С другими NPC или же между собой обмениваться информацией они не могут. Если игрок начнет играть на музыкальном инструменте, то они могут бросить несколько монет, однако в будущих диалогах этот момент никак обговариваться между ними не будет. Населяющие мир персонажи обладают тем же поведением, за исключением того, что они не следуют за игроком. На рисунке 26 показан пример диалога в Baldur's gate 3.

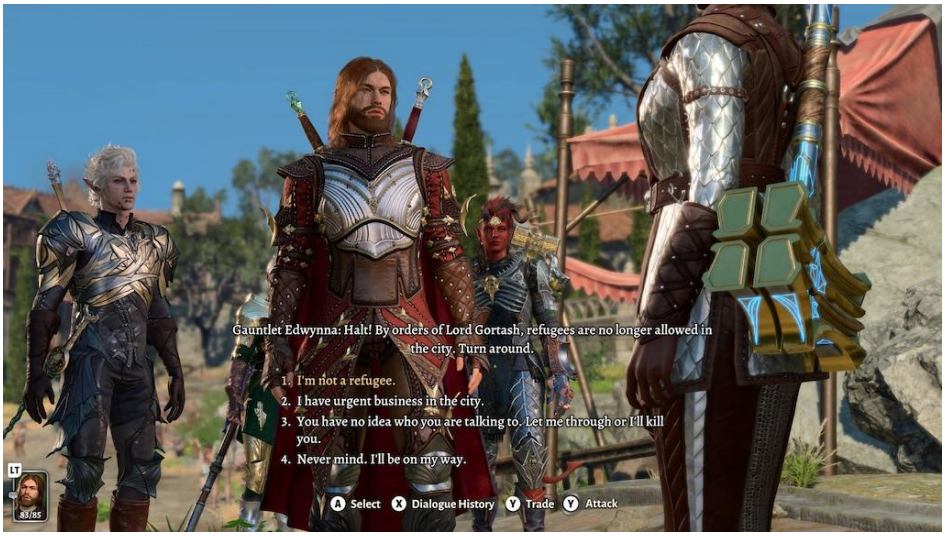

Рисунок 26 – Пример диалога в Baldur's gate 3

Starfield. В данной игре NPC между друг другом могут обмениваться различными слухами, которые распространяются на близлежащих территориях, а также небольшими новостями. Однако, такие диалоги происходят исключительно один на один, а рядом находящиеся персонажи никак не реагируют на диалог.

Персонажи в игре всегда общаются с игроком один на один.

Персонажи могут с каким-то шансом начать обсуждать слухи друг с другом, причем слухи могут распространятся от одного NPC к другому. NPC

сами могут заговорить с игроком если у них есть тема для разговора, однако групповое общение отсутствует. На рисунке 27 показан пример диалога Starfield

Рисунок 27 – Пример диалога в Starfield

Cyberpunk 2077. В этой игре общение персонажей друг с другом всегда заранее созданы дизайнерами. Случайные прохожие на улице всегда будут проходить мимо друг друга, а если они начинают общение, то оно всегда создано заранее в одном конкретном месте. Если игрок будет возвращаться в это место, то диалог будет повторяться.

Интеракция игрока с NPC сильно ограничена, за раз можно общаться всегда только с одним персонажем.

Поведение в диалогах всегда заранее прописано разработчиками. Между собой общение только в заранее созданных сценах, а с игроком в тех случаях, когда это необходимо сюжету. Пример диалога из cyberpunk 2077 показан на рисунке 28.

Рисунок 28 – Пример диалога в cyberpunk 2077

ЗАКЛЮЧЕНИЕ

По итогу выполненной работы были разобраны актуальные системы взаимодействия неигровых персонажей в игровом пространстве. Были подняты вопросы несовершенства общения в группе из нескольких NPC. Рассмотрены существующие системы, предлагающие решить эту проблему теми или иными способами.

Была спроектирована модель общения с группой разных NPC, основанная на ранее проведенных исследованиях и берущая во внимание некоторые аспекты других реализаций с целью улучшить и дополнить их. В последствии модель была несколько подкорректирована и доработана для проведения тестирований с целью оценки усиления погружения в игровой мир по сравнению с другими системами.

По итогу тестирования были подведены результаты и были выявлены аспекты, которые стоит в будущем учесть и доработать с целью еще большего увеличения вовлечения и погружения в игровое пространство игроком. Однако уже на данном этапе испытуемые подтверждают, что такая система общения имеет место на существование и она погружает игрока в процесс, подталкивая его изучать различные темы для диалога.

Систему можно использовать как вспомогательную механику в играх с большим количеством персонажей и широким миром для исследования, так и строить различные задания с поиском нужной информации на ее основе.

Было проведено сравнение с другими актуальными на момент написания работы играми. По итогу было выявлено, что крупные современные проекты не очень активно затрагивают данную проблему.

Созданная в данной работе системе может увеличить вовлеченность и погружение пользователей в игровой процесс за счет группового общения с группой неигровых персонажей и возможностью прерывания и переключения с одного персонажа на другого без потери информативности. Разработанное решение опубликовано в репозитории GitHub и размещено по следующей ссылке: https://github.com/Zengard/master-dissertation.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Charlene Jennetta, Anna L.Coxa, Paul Cairnsb, Samira Dhopareec, Andrew Eppsc, Tim Tijsd, Alison Waltond Measuring and defining the experience of immersion in games // ScienceDirect [Electronic resource]. - 2008. - URL: https://www.sciencedirect.com/science/article/abs/pii/S1071581908000499 (дата обращения (09.01.2023)

2. Muhanna Quwaider, Abdullah Alabed, Rehab Duwairi The Impact of Video Games on the Players Behaviors: A Survey // Procedia Computer Science [Electronic resource]. - 2019. – URL: https://www.sciencedirect.com/science/article/pii/S1877050919305393 (дата обращения 15.12.2022)

3. Ian Schreiber Better than Dialogue Trees // Project Horseshoe [Electronic resource]. - 2017. - URL: https://www.projecthorseshoe.com/reports/featured/ph17r5.htm (дата обращения 02.12.2022)

4. Martin J. Dechant, Robin Welsch, Julian Frommel, Regan L. Mandryk (Don’t) stand by me: How trait psychopathy and NPC emotion influence player perceptions, verbal responses, and movement behaviours in a gaming task // Digital library [Electronic resource]. – 2022. - URL: https://dl.acm.org/doi/pdf/10.1145/3491102.3502014 (дата обращения 26.12.2022)

5. Building the AI of F.E.A.R. with Goal Oriented Action Planning // AIandGames [Electronic resource]. – 2020. – URL:https://www.aiandgames.com/2020/05/06/ai-101-goap-fear/ (дата обращения 20.12.2022)

6. Tommy Thompson The Story of Facade: The AI-Powered Interactive Drama // Game developer [Electronic resource]. – 2020. – URL:https://www.gamedeveloper.com/design/the-story-of-facade-the-ai-powered-interactive-drama (дата обращения 25.12.2022)

7. Stephanie Rennick, School of Humanities Improving video game

conversations with trope-informed // University of Glasgow [Electronic resource]. - 2021. – URL: https://eprints.gla.ac.uk/252546/2/252546.pdf (дата обращения 06.12.2022)

8. Lu´ıs Fernando Bicalho, Bruno Feijo, Augusto Baffa A Culture Model for Non-Player Characters' Behaviors in Role-Playing Games // Proceedings of SBGames [Electronic resource]. - 2020. – URL: https://www.sbgames.org/proceedings2020/ComputacaoFull/210071.pdf (дата обращения 03.12.2022)

9. Alfred Andersson, Keman Nguyen Asynchronous Dialogue System // Malmö University [Electronic resource]. - 2022. - URL:https://www.diva-portal.org/smash/get/diva2:1676057/FULLTEXT02.pdf (дата обращения 07.12.2022)

10. Ian Schreiber Role of ChatGPT in Gaming: According to ChatGPT // University of Tennessee [Electronic resource]. - 2023. - URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4375510 (дата обращения 02.06.2022)

11. Judith van Stegeren, Jakub Mysliwiec Fine-tuning GPT-2 on annotated RPG quests for NPC dialogue generation // AIandGames [Electronic resource]. – 2020. – URL:https://dl.acm.org/doi/pdf/10.1145/3472538.3472595 (дата обращения 20.05.2023)

12. Mika Hämäläinen, Khalid Alnajjar RPG-GPT // Master's Programme in Linguistic Diversity and Digital Humanities [Electronic resource]. – 2023. - URL: https://helda.helsinki.fi/bitstream/handle/10138/358208/Poyhonen_Teemu_thesis_2023.pdf?sequence=2&isAllowed=y (дата обращения 26.05.2022)

13. Oliver P. John and Sanjay Srivastava The Big-Five Trait Taxonomy: History, Measurement, and Theoretical Perspectives // University of California at Berkeley [Electronic resource]. – 1999. - URL:https://pages.uoregon.edu/sanjay/pubs/bigfive.pdf

14. Informative Sound Design in Video Games Patrick Ng и Keith Nesbitt – URL:https://www.researchgate.net/profile/Keith-

Nesbitt/publication/262215104_Informative_Sound_Design_in_Video_Games/links/62d5e3b8daa38132ab5983eb/Informative-Sound-Design-in-Video-Games.pdf

15. Unity Game Development Engine: A Technical Survey – URL:https://www.researchgate.net/profile/Faizan-Hussain-2/publication/348917348_Unity_Game_Development_Engine_A_Technical_Survey/links/60167daf45851517ef2b2c0d/Unity-Game-Development-Engine-A-Technical-Survey.pdf

16. Nuno Afonso, Rui Prada Agents that Relate: Improving the Social Believability of Non-Player Characters in Role-Playing Games // Technical University of Lisbon [Electronic resource]. - 2009. - URL: https://www. gaips.inesc-id.pt/component/gaips/publications/showPublicationPdf?pid=73&format=raw

17. Yongcheng Liu The Emotional Experience of Interaction with NPCs in RPG Games // Game Developer [Electronic resource]. – 2022. - URL: https://www.gamedeveloper.com/blogs/the-emotional-experience-of-interaction-with-npcs-in-rpg-games

18. Joni Heikkonen Interaction in video game dialogue. Exploring the effects of ludic context on fictional dialogue // University of Helsinki [Electronic resource]. - 2018. – URL:https://helda.helsinki.fi/bitstream/handle/10138/289774/Heikkonen_Joni_Pro_gradu_2018.pdf?isAllowed=y&sequence=2 (дата обращения 12.12.2022)

19. Marina Krcmara, Kirstie Farrarb, Rory McGloinb The effect of video game realism on attention, retention and aggressive outcomes // ScienceDirect [Electronic resource]. - 2010. – URL: https://www.sciencedirect.com/science/article/abs/pii/S0747563210002803 (дата обращения 07.01.2023)

20. Nikolai O.Veselov, Arthur A.Chubarov, Mikhail N.Roshchin, Lev M.Marder, Stanislav M.Segal, Alexei V.Samsonovich - ScienceDirect Emotional BICA for non-player characters: New empirical data // // ScienceDirect[Electronic resource]. - 2020. - URL: https://www.sciencedirect.com/science/article/pii/S187705092030362 8?via%3Dih

ub (дата обращения 04.01.2023)

21. Yongcheng Liu The Emotional Experience of Interaction with NPCs in RPG Games // Game Developer [Electronic resource]. – 2022. - URL: https://www.gamedeveloper.com/blogs/the-emotional-experience-of-interaction-with-npcs-in-rpg-games (дата обращения 08.12.2022)

22. Phyllis Hillwig What Video Games Can Teach Us About Engagement in Education // Eurekii [Electronic resource]. - 2021. - URL: https://eurekii.com/what-video-games-can-teach-us-about-engagement-in-education/ (03.12.2022)

23. Jeffrey Georgeson, Christopher Child NPCS AS PEOPLE, TOO: THE EXTREME AI PERSONALITY ENGINE // Arxiv [Electronic resource]. – 2016. - URL: https://arxiv.org/ftp/arxiv/papers/1609/1609.04879.pdf (дата обращения 08.1.2023)

24. Daniil A.Azarnov, Artur A.Chubarov, Alexei V.Samsonovich Virtual Actor with Social-Emotional Intelligence // ScienceDirect [Electronic resource]. - 2018. – URL: https://www.sciencedirect.com/science/article/pii/S1877050918300140 (09.1.2023)

25. Salma Hamdy, David King AFFECT AND BELIEVABILITY IN GAME CHARACTERS – A REVIEW OF THE USE OF AFFECTIVE COMPUTING IN GAMES // Research Gate [Electronic resource]. - 2017. - URL:https://rke.abertay.ac.uk/ws/portalfiles/portal/9263201/ElSayed_King_AffectAndBelievabilityInGameCharacters_Published_2017.pdf (дата обращения 19.12.2022)

26. Sara Bögels Neural correlates of turn-taking in the wild: Response planning starts early in free interviews // ScienceDirect [Electronic resource]. - 2020. - URL:https://www.sciencedirect.com/science/article/pii/S0010027720301669 (дата обращения 05.1.2023)

27. Alfred Andersson, Keman Nguyen Asynchronous Dialogue System // Malmö University [Electronic resource]. - 2022. - URL:https://www.diva-portal.org/smash/get/diva2:1676057/FULLTEXT02.pdf (дата обращения 07.12.2022)

28. Kohji Dohsaka, Ryota Asai Effects of Conversational Agents on Human Communication in Thought-Evoking Multi-Party Dialogues // The 10th Annual Meeting of the Special Interest Group in Discourse and Dialogue [Electronic resource]. - 2009. - URL:https://aclanthology.org/W09-3932.pdf (дата обращения 07.12.2022)

18. Kimberly Voll Less is more: Designing Awesome AI for Games // GDC [Electronic resource]. - 2015. – URL:https://www.youtube.com/watch?v=1xWg54mdQos (дата обращения 05.01.2023)

29. Artificial Intelligence in The Sims series // Yoann Bourse [Electronic resource] – 2014. - URL:https://team.inria.fr/imagine/files/2014/10/sims-slides.pdf (дата обращения 03.01.2023)

30. Chris Hoge Helping players hate (or love) their nemesis // GDC [Electronic resource]. - 2018. – URL:https://www.youtube.com/watch?v=p3ShGfJkLcU&t=221s (дата обращения 14.12.2022)

31. Augusto Baffa, Pedro Sampaio, Bruno Feijo´, Mauricio Lana Dealing with the emotions of Non Player Characters // Proceedings of SBGames [Electronic resource]. - 2017. - URL: https://www.sbgames.org/sbgames2017/papers/ComputacaoFull/175357.pdf (04.01.2023)

32. Domsch Sebastian Dialogue across Media // Dialogue in Video Games. – 2017. P.251 – 271.

33. James Ryan A Lightweight Videogame Dialogue Manager // Research Gate [Electronic resource]. – 2016. URL:https://www.researchgate.net/publication/303329650_A_Lightweight_Videogame_Dialogue_Manager (дата обращения 08.01.2023)