Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Направление подготовки: 09.04.04 – Программная инженерия

Профиль: Разработка программно-информационных систем

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Разработка решения для визуализации сценарного прототипа инструментами генерации раскадровок

Студент 2 курса

Группы 11-821

«___»_____ 2020 г. Сахибгареева Г.Ф.

Научный руководитель

к.т.н., доцент кафедры программной инженерии

«___»_____ 2020 г. Кугуракова В.В.

Директор Высшей школы ИТИС, к.т.н.

«___»_____ 2020 г. Абрамский М.М.

Казань – 2020 г.

Содержание

Введение

В игровой разработке все чаще говорят о повествовании. Появляются научные работы, в которых не только анализируют игры, но и предлагают подходы к разработке интерактивного нарратива.

Пока компьютерная графика была на низком уровне, истории в играх были условными. Нужно было пофантазировать, чтобы представить, что набор желтых пикселей – это персонаж игрока, а набор красных – это враг, от которого надо бежать. Тогда никто не говорил о драматургическом потенциале игр.

Реалистичная графика не позволяет оставлять истории на уровне условности. Сегодня игры до сих пор не соперники кинематографу по мастерству сторителлинга.

Отсутствие практик разработки интерактивных историй приводит к ошибкам. Реализм происходящего обостряет конфликт между игровым действом и повествованием и разрушает погружение. Знания, которые позволяют создавать гармоничное повествование в играх, передаются от разработчика к разработчику. До сих пор нет достаточной литературы и методологий для разработки грамотного игрового повествования.

Другой вопрос – это отсутствие налаженной связи между разработчиками и учеными [1]. На сегодняшний день игровые компании и научные сообщества существуют раздельно. Отсутствие коммуникаций приводит к тому, что существующие методы не задокументированы в научных источниках и не получили признания. В то же время научные знания не учитываются при разработке игр.

Исключительные случаи есть. Например, Hellblade: Senua’s Sacrifice (здесь и далее упоминаемые игры и сериалы представлены в Прил. Е). Разработка этой игры не была бы такой успешной, если бы не сотрудничество с

психотерапевтами. Совместная работа специалистов привела к созданию уникального игрового опыта.

Изолированность науки от разработки привела к тому, что такой подход, как сценарный прототип, до сих пор не освещен в научной среде. Эффективность данного подхода не подтверждена.

Интерактивный сторителлинг – это мощный инструмент, который начинает использовать и кинематограф. Один из примеров интерактивного кино – серия сериала Черное зеркало – Брандашмыг.

Один из способов повышения качества интерактивного повествования – это тестирование. В случае с повествованием – это тестирование упомянутого сценарного прототипа.

Прототипирование в ИТ – это возможность сократить риски. В жизненном цикле программного обеспечения, в основе которого есть интерактивное взаимодействие с пользователем и компьютерная графика, прототипирование – это возможность проработать гипотезы еще до начала дорогостоящей разработки. Так же и в игровой разработке, как направлении ИТ, важно прототипирование.

Еще одно актуальное направление развития ИТ – это автоматизация. Автоматизация процесса прототипирования – это генерация прототипов.

Автоматизация в области разработки сценарного прототипа – это генерация сценарного прототипа. В данной работе входными данными для генерации выбран текст на естественном языке.

Реализация инструмента генерации сценарного прототипа из текста – это комплекс задач: извлечение сущностей и связей между ними из текста, извлечение разветвленной структуры истории, визуализация разветвленной структуры, автоматическая сборка интерактивного прототипа, генерация или автоматический подбор визуализации извлеченных сущностей.

В данной работе приводится анализ существующих решений визуализации текстовых описаний, анализируется эффективность разработки инструмента генерации сценарного прототипа, формулируются требования к компоненту визуализации инструмента, а также приводится результат экспериментальной разработки компонента визуализации.

Объект исследования в работе – прототипирование интерактивного повествования.

Предмет исследования – визуализация сгенерированного сценарного прототипа игры.

В первой главе приведен анализ предметной области. В главе приводится объяснение того, что такое сценарий прототип и какие к нему предъявляются требования. На основании трудов в сфере ИТ и в сфере разработки игр анализируется актуальность инструмента генерации сценарного прототипа. В главе анализируется текущий инструментарий разработчиков повествования. Приводится обзор существующих инструментов визуализации текста. Приведены существующие разветвленные структуры.

Во второй главе сформулированы требования к инструменту генерации сценарного прототипа и, в частности, требования к компоненту визуализации инструмента. Разработана концепция пилотного решения. Приведена постановка задачи, описана гипотеза, сформулированы задачи для проведения эксперимента.

Третья глава посвящена подробностям технической реализации эксперимента. Описана архитектура инструмента, в рамках которой должен работать компонент визуализации. Приведен анализ средства реализации эксперимента. Приведено описание процесса обработки текста и создания визуализации.

Четвертая глава содержит результаты эксперимента, анализ эффективности системы, описание ограничений системы и планы развития.

В заключении приведены общие результаты работы.

Глава 1. Анализ предметной области

1.1 Актуальность инструмента генерации сценарного прототипа

1.1.1 Сценарный прототип

Прежде чем привести актуальность инструмента генерации сценарного прототипа, необходимо дать определение сценарного прототипа и проанализировать его свойства.

Сценарный прототип – это интерактивный прототип, который позволяет пройти всю историю игры в виде, в котором она должна быть представлена в игре. По этой причине сценарный прототип может содержать в себе повествовательные механики, которые не включены в рамках данной работы, но подразумеваются в определении. Кроме повествовательных механик, повествование в игре может транслировать через визуализацию, текст и звук. В данной работе акцент сделан на визуализации.

Сценарный прототип фокусируется на повествовании. В нем может не быть геймплейных механик в чистом виде. Вместо них в сценарном прототипе допустимо оставить заглушки. Каждый раз, когда необходимо показать, что происходит геймплейное событие, в качестве заглушки можно в случайном порядке воспроизвести одно из подходящих событий. Например, можно сгенерировать исход боя: смерть или победа, сколько наград, какие потери. Это необходимо для симуляции полноценного игрового опыта.

Повествовательные же механики лучше прототипировать в рамках сценарного прототипа, т.к. именно они определяют как подается история. Например, через диалоги, взаимодействие с интерактивными объектами: книгами, алтарями, паззлами и т.д..

Необходимость в интерактивном взаимодействии в сценарном прототипе обусловлена тем, что интерактивность – это главное свойство игры.

От степени интерактивности зависит уровень погружения в мир повествования. А еще уровень вовлечения в игровой процесс. Такую степень интерактивности, степень возможности взаимодействия с виртуальным миром игры называют агентивностью. Агентивность – это обязательное условие для любого интерактивного медиа. Где-то игроку дозволено идти по уровню, продвигаясь при этом по сюжету, а где-то ему предоставлена возможность творить собственный мир. Сценарный прототип должен реализовывать агентивность с точки зрения повествования. Это значит, игрок должен иметь возможность направлять события, выбирать реплики диалогов, выбирать с кем общаться, что делать и какой стратегии следовать.

Еще одно требование к сценарному прототипу – соответствие жанру. Если прототипируется текстовая игра, тогда сценарный прототип может быть текстовым. Но если прототипируется игра другого жанра, создавать сценарный прототип в отрыве от визуализации нельзя по двум причинам: визуализация – это тот же способ трансляции повествования; визуализация может быть интерактивной. Таким образом, пусть визуализация будет простой, шаблонной, но она должна присутствовать для тестирования. Другая причина соответствия жанру – возможность для итеративной разработки. Проще нарастить функционал сценарного прототипа, чем оставить его без применения.

Другое обязательное условие для сценарного прототипа, чтобы он был визуализирован.

Воспринимать информацию с иллюстрациями и анимацией проще, чем читать [2]. Действовать самостоятельно в интерактивной среде лучше, чем наблюдать (см., например, [3; 4]).

Визуальность сценарного прототипа воплощается в том, как выглядит сцена, в которой происходят действия, как выглядят персонажи. Зачастую это зависит от того, какой стиль был выбран разработчиками, для передачи определенной атмосферы и эмоций. Визуальный канал восприятия

чувствителен, эстетическая красота привлекает внимание. При отсутствии визуализации необходимо самостоятельно придумывать то, как выглядят описываемые события.

Практическая эффективность использования сценарного прототипа не доказана. Чаще советуют создавать бумажные прототипы, чем интерактивные цифровые (см., например, [5; 6; 7; 8]). Поэтому сперва целесообразно проанализировать эффективность прототипирования при разработке повествования игр.

Эффективность прототипирования в ИТ проанализирована в работе 2018 года [9]. В противовес четырем минусам в ней приведены девять плюсов. Ниже приведено обобщение и интерпретация плюсов применительно к разработке игр:

1. Когда на этапе ранней разработки происходит тестирование прототипов, игроки чувствуют свое участие в проекте и радуются этому. Более того, они превращаются в лояльную аудиторию и образуют комьюнити еще до назначения даты релиза и начала маркетинговой кампании игры. В таком формате работают инди-проекты, которые воспользовались краудфандингом, например, на Kickstarter[1].
2. Фидбек игроков, которые составляют целевую аудиторию, конкретизирует ожидания и превращает их в требования. От этого игра становится лучше. Иногда случается, что в сессиях тестирования прототипов выясняются неожиданные специфические требования.
3. Прототипирование – однозначно эффективный и надежный способ сэкономить силы, нервы и деньги.

[1] Kickstarter – веб-ресурс для привлечения денег на разработку проекта в виде краудфандинга. Режим доступа: https://www.kickstarter.com/ (дата обращения 5.05.2020)

Обоснование эффективности сценарного прототипа можно привести с другой стороны. В работе 2007 года был проведен эксперимент, в котором для прототипирования игр использовали и бумажный прототип, и цифровой [8].

Плюсы цифрового прототипа оказались в том, что интерактивность не всегда поддается тестированию на бумаге. Заведомо компьютерное приложение лучше тестировать на целевой платформе: на ПК, веб или мобильной платформе и т.д..

Плюс бумажного прототипирования в том, что для этого не нужно скачивать и разворачивать среду для прототипирования – достаточно взять ручку и бумагу.

Таким образом формулируется одно из требований к сценарному прототипу: он должен быть разработан для целевой платформы.

Одно из главных преимуществ сценарного прототипа, соответствующего всем перечисленным требованиям состоит в том, что его можно отправить на тестирование целевой аудитории. Чаще всего сценарный прототип – это легковесный проект, который не требует высокой мощности устройства. Это расширяет аудиторию тестирования. Более того, если развернуть сценарный прототип на веб-платформе, это решит проблему отправки актуальной версии и не будет занимать память устройства.

Таким образом сформированы главные свойства сценарного прототипа:

1. Он должен быть интерактивным.
2. Он должен содержать визуализацию.
3. Он должен соответствовать жанру.
4. Он должен быть реализован для целевой платформы.
5. Он должен быть цифровым.

1.1.2 Генерация сценарного прототипа

За разработку повествования игры в команде проекта отвечают два вида специалистов: игровой сценарист и нарративный дизайнер[2]. Сценаристы и писатели часто приходят не из ИТ-сферы, а из журналистики или кинематографа. Такие специалисты редко имеют опыт программирования. Чаще они ограничиваются текстовыми редакторами.

Для выходцев из ИТ изучить новое ПО легче. Поэтому такие специалисты могут работать с целевым движком и в других редакторах, что положительно сказывается на процессе разработки.

Создавать разветвленную структуру и не прибегать при этом к программированию – сложно. Знание алгоритмов и структур повышают качество работы разработчиков повествования.

Даже применение Twine [11] требует простейших знаний разметки. В случае с RenPy [12] полезно знать как программировать на python.

Упрощение процесса тестирования с применением генератора сценарного прототипа решает две задачи: разгружает других специалистов, которых приходится привлекать из-за отсутствия опыта программирования у сценаристов и нарративных дизайнеров; высвобождает время для обучения основам программирования у сценаристов и нарративных дизайнеров.

Противоположное мнение состоит в том, что так или иначе в будущем всем придется обучиться программированию. Первые шаги в программировании сценарного прототипа могут стать толчком в развитии специалистов.

Некоторые считают прототипирование вредным. Обоснование этого мнения в том, что если прототипирование занимает много времени, то в какой-то момент прототипирование может неоправданно сорвать сроки. Сама идея о генерации сценарного прототипа устраняет этот риск.

---

[2] Нарративный дизайнер – специализация гейм-дизайнера, дизайнер по внедрению истории в игру.

В работе с сравнением бумажного прототипа и цифрового был обозначен минус цифрового прототипа – прежде чем начать разработку, необходимо развернуть среду программирования [8]. Генератор сценарных прототипов устраняет этот вопрос: инструмент надо лишь скачать, загрузить в него текстовые документы и визуальные зарисовки и запустить процесс генерации прототипа.

Говоря о прототипировании в разработке игр, нужно упомянуть труд Джесси Шелла – Искусство геймдизайна [5].

Призма 16 в этой книге называется Призмой оптимизации рисков. Как известно, риски производства дешевле устранить в начале разработки, поэтому следом Джесси Шелл приводит десять советов для прототипирования игр.

Если применить данные советы к инструменту генерации сценарного прототипа, то получаются следующие выводы:

1. При тестирования прототипов формируются ответы на различные вопросы, которые лучше решить на раннем этапе. Примеры вопросов для сценарного прототипа:
    a. история создает нужную атмосферу?
    b. она резонирует с аудиторией?
    c. в ней соблюдены правила драматургии?
    d. она имеет достаточный потенциал для интерактива?
    e. соблюдается ли темпоритм[3]?
    f. персонажи и события интересные?
    g. сколько визуального контента и геймплея понадобится?
    h. история и визуализация будут сочетаться?
2. Сгенерированный прототип выглядит аккуратно.
3. Для генерации сценарного прототипа не нужно быть художником или программистом.

[3] Темпоритм – частота и плотность подачи истории

4. Отказаться от сгенерированного прототипа проще, чем он прототипа, созданного своими руками.
5. Удавшийся прототип можно итерировать.
6. Время, которое экономит генерация, можно использовать для прототипирования других компонентов, например, геймплея и визуализации.
7. Генератор позволяет забыть об анализе и выборе игрового движка на ранних стадиях разработки.

Данные выводы показывают, что гипотетически генерация сценарного прототипа улучшает качество повествования игры и показывает уязвимые места сценария на ранних стадиях.

На начальных этапах разработки игры повествование – это серия текстовых документов, таблиц, заметок и схем. Объединение и анализ этих документов дает источник знания для инструмента, способного генерировать варианты сценарного прототипа.

Система, которая сможет генерировать и распространять сценарные прототипы между разработчиками и будущими игроками – ускорит и упростит процесс разработки повествования игр.

1.2 Существующие форматы документирования игр

Для генерации сценарного прототипа необходимы входные данные в виде текстов. Тексты для описания повествовательной составляющей игры документируются в следующих форматах: текстовый документ, таблица и wiki-разметка.

1.2.1 Текстовый документ

История фильма укладывается в единственный документ, который называют сценарием. Для оформления сценария существуют международный стандарт. Учитывая, что первые игровые сценаристы были сценаристами кино,

журналистами и писателями, этот подход применили и к разработке повествования игр. Практика показывает, что для разветвленного повествования не нужен текстовый сценарий.

В текстовом формате можно хранить некоторую информацию об игре:

- описания локаций,
- описания персонажей,
- внутриигровые события,
- реплики диалогов,
- статичные тексты, с которыми игрок встретится во внутриигровом мире.

Структура истории игры отличается тем, что ней заложена возможность выбора собственного пути развития событий. Это может быть выбор реплик в диалогах, последовательности прохождения игры, стиля игры, внутриигровой роли. Любой из этих выборов порождает изменения, которые сложно зафиксировать в линейной структуре текстового документа.

1.2.2 Wiki-разметка

Текст с гиперссылками позволяет поддерживать определенную нелинейность. Формат Wiki-разметки подходит для хранения знаний о мире игры. В такой структуре каждый документ содержит в себе часть истории игры с ссылками на сущности, используемые в тексте. Например, в описании развития событий квеста могут содержаться ссылки на страницы с описаниями персонажей, обязательных интерактивных предметов, а также ссылки на предыдущие и последующие квесты. Кроме этого можно создать отдельные страницы для описания развития событий каждого действия.

Часто Wiki-ресурсы появляются после создания игр, над ними работают фанаты. Но в процессе разработки такой формат незаменим для поддержания внутриигровой истории в актуальном виде. Более того, доступ к Wiki-разметке всех участников разработки поможет ввести в курс дела и напоминать о доработке и обновлениях.

1.2.3 Таблица

Табличное представление применяется в процессе разработки игр. В таком виде проще всего представить игру для программистов и художников. В каждой строке такой таблицы содержится информация о каждом элементе игрового процесса. Чаще всего такие форматы применимы для игр с низкой степенью ветвления структуры, а также для таких жанров как мобильные игры.

Табличное представление эффективнее текстового, но оно также более специализированное.

В качестве примера – таблица для разработки квеста. У таблицы могут быть следующие поля: номер квеста, название, квестгивер[4], этапы, список пропсов[5], локация, аудиосопровождение и другие элементы дизайна (Таблица 1).

Таблицы 1 – Пример таблицы для мобильного квеста

№	Название квеста	1 этап	2 этап	3 этап
1	Спасти котика	Найти лестницу	Поставить лестницу к дереву	Залезть на лестницу и спасти котика

1.3 Обзор литературы и существующего инструментария

Кроме создания привычной бумажной и электронной документации, в разработке применяются альтернативные способы прототипирования, например, бумажное прототипирование и раскадровки.

1.3.1 Бумажный прототип

Изначально бумажное прототипирование советуют при проектировании игровых механик (см., например, [5; 6]). Благодаря развитию теории интерактивного повествования данный вид прототипирования стали применять и в разработке нарратива (см., например, [7; 8]).

[4] Квестгивер – англ. quest giver, персонаж, который дает задание, квест.

[5] Пропсы – сокр. от англ. properties, props, бутафория, наполнение уровня, элементы антуража.

Джесси Шелл приводит примеры бумажного прототипирования головоломки Тетрис и шутера Halo, что говорит о том, что бумажный прототип применим к любому жанру [5].

В работа Хартмута Кеница приводится пример бумажных карточек для работы над повествованием игры [6]. Такой вариант способствует креативности и не отвлекает на рутину – заготовку карточек.

В одном из выступлений на GDC Vault была представлена инструкция о бумажном прототипировании повествования игр [7]. Способ показывает свою эффективность, т.к. применяется в ряде компаний.

Преимущество бумажного прототипа в том, что это дешевое решение не требует много времени. Все ключевые механики игры в том или ином виде можно реализовать в виде партии в настольную игру. Причем составляющие этой игры могут быть в готовом виде: карточки, карты, кубики, фишки. Все, что остается сделать разработчикам, это научить игроков правилам игры и провести сессию тестирования. Данный способ полагается на воображение.

С другой стороны, исследования подтвердили, что бумажные прототипы далеки от специфичности платформы: ПК, мобильных устройств, VR устройств и другого оборудования [8]. Отсутствие мануального взаимодействия с устройствами ввода допустимо только с этой оговоркой. В остальном, бумажный прототип гораздо эффективнее для тестирования, чем чтение и редактирование технической документации.

1.3.2 Раскадровка

Раскадровка – это способ прототипирования, который пришел в игровую индустрию из кинематографа. Раскадровку применяют при разработке компьютерной графики. Также она применима для визуализации сценария игры. Однако создавать раскадровку должен сценарист, который умеет рисовать. Либо необходимо задействовать художника.

Если рассматривать раскадровку как одну из итераций создания компьютерной графики, то допустимо ее тестирование. Тестирование сценария, представленного в виде серии картинок с подписями или устными комментариями сценариста, эффективнее, чем чтение сценарного документа.

Создание раскадровки включают в виде одного из обязательных этапов разработки игр компания Disney [13].

1.3.3 Профессиональное ПО сценаристов

Кроме бумажного и электронного документирования, разработчики используют программные решения, которые упрощают разработку.

Среди инструментов для создания текстовых документов распространены стандартные Microsoft Word и Google Docs. Кроме этого игровые сценаристы используют в работе специализированные приложения для создания сценариев кино: Scrivener[6], Final Draft[7], КИТ Сценарист[8]. Тот же самый софт используют сценаристы игр, для поддержания документации в едином формате. Однако, как уже ни раз повторялось, есть серьезное но, против хранения истории игры исключительно в текстовом виде: разветвленную структуру тяжело поддерживать в виде текста.

Для визуализации и поддержания разветвленного повествования на помощью приходят простые в изучении движки. Выбор обусловлен тем, что чаще всего игровые сценаристы приходят в индустрию из гуманитарных направлений.

Примеры таких инструментов – текстовые движки вроде Twine и движки для создания визуальных новелл, например RenRy. Кроме движков игровые сценаристы могут воспользоваться таким специализированным программным обеспечением как Articy:draft [14].

---

[6] Scrivener – программа-ассистент литератора. Режим доступа: https://www.literatureandlatte.com/scrivener/overview (дата обращения 5.05.2020)

[7] Final Draft – программа для написания и форматирования сценариев. Режим доступа: https://www.finaldraft.com/ (дата обращения 5.05.2020)

[8] КИТ Сценарист – программа для написания сценариев. Режим доступа: https://kitscenarist.ru/ (дата обращения 5.05.2020)

Twine

Twine – open-source движок для создания текстовых игр (рис. 1). Инструмент снижает порог вхождения в игровую индустрию благодаря трем ключевым преимуществам:

1. в Twine построение историй происходит с помощью карточек. Карточки формируют граф, который наглядно показывает структуру игры;
2. чтобы работать в Twine не нужно знать программирование. Внутри каждой карточки пользователь вводит текст с применением разметки. Кроме этого доступны настройки стиля текста, макросов, добавление условий, переменных и картинок.
3. в Twine доступен экспорт проекта в виде html-файла. Его можно опубликовать в таким сервисе как itch.io[9]. Затраты на публикацию при этом минимальные.

Кроме этого Twine применятся для создания диалоговой системы. Результат работы можно, например, экспортировать в Unity с помощью ассета Crandle [15].

С точки зрения создания сценарного прототипа Twine хорош для создания текстовых игр и диалогов. Но кроме этого его применение лишено смысла, т.к. главное требование к сценарному прототипу – это его разработка в соответствующем жанре.

[9] itch.io – веб-ресурс для загрузки и продажи игр. Режим доступа: https://itch.io/ (дата обращения 5.05.2020)

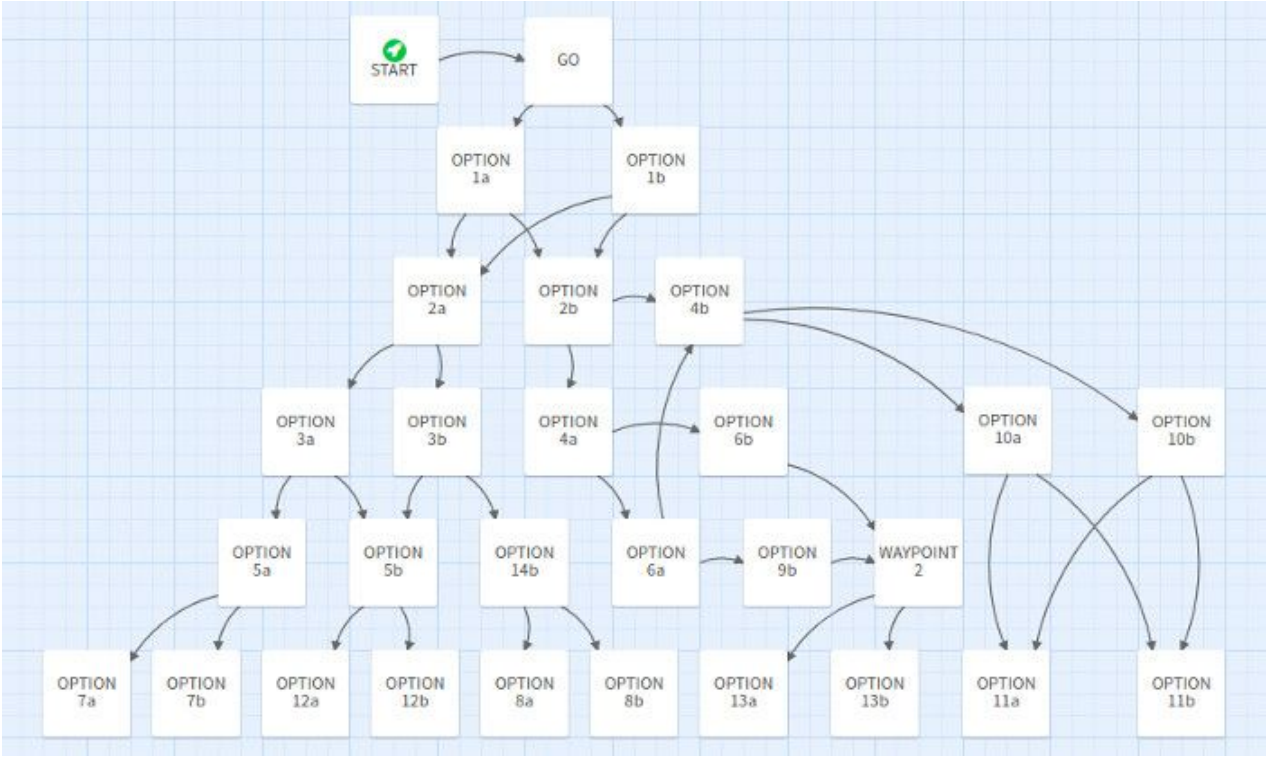


Рисунок 1 – Пример игры в Twine

<u>RenPy</u>

RenPy – бесплатный open-source движок для разработки визуальных новелл.

Три ключевых преимущества RenPy:

1. редактор кода в сама движке;
2. язык программирования – python;
3. встроенный гайд для быстрого освоения инструмента.

В проект в RenPy можно добавлять стандартные для новелл компоненты: текст, фоновое изображение, персонажи, музыка, звуки, анимации.

В качестве следующей итерации проработки сюжета визуальная новелла эффективнее текстовой игры, т.к.включает в себя визуализацию. Однако, как и в случае с Twine, RenPy можно и нужно создавать сценарные прототипы визуальных новелл. В случае с играми других жанров RenPy может оказаться пустой тратой времени.

Articy:draft

Articy:draft – приложение для визуализации и организации игровых компонентов (рис. 2). Разработчики относят его к приложениям для разработки point-and-click[10] адвенчур, но она применима при разработке игр других жанров. По сути Articy:draft позволяет систематизировать информацию, которая постоянно нужна в течении разработки игры в одном хранилище в стандартной форме. Систему хранения сущностей в Articy:draft можно сравнить с диаграммой связей или с системой wiki-разметки.

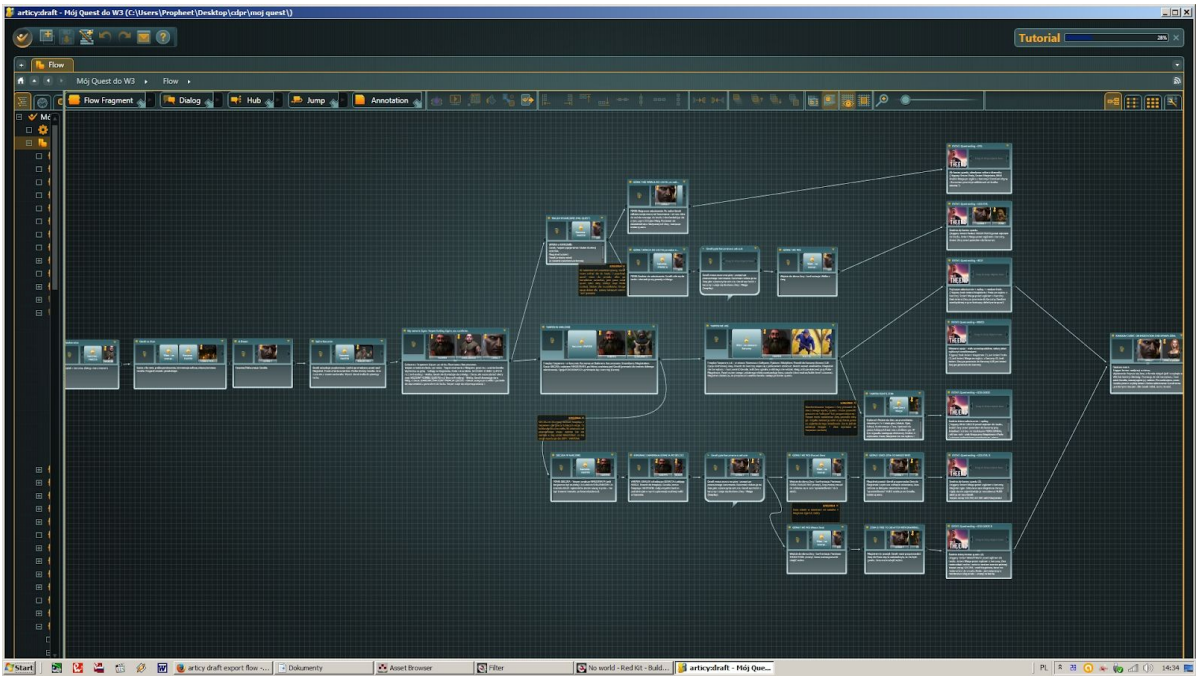


Рисунок 2 – Пример игры в Articy:draft

С точки зрения разработки сценарного прототипа у Articy:draft есть следующие три преимущества:

1. Экспорт данных доступен в формате json, что делает инструмент универсальным.

[10] Point-and-click – жанр игр. Игровой процесс заключается в поиске интерактивных объектов визуализации и клике по ним мышкой. Кроме этого в них встречаются диалоги.

2. В Articy:draft можно хранить разнообразную информацию: персонажей, диалоги, квесты, инвентарь, достижения, игровые фичи и других игровые сущности.
3. Принцип визуального программирования снижает порог вхождения для начинающих специалистов.

Минус Articy:draft в его стоимости. Она доступна для AAA-компаний. В случае с инди-разработчиками использование Articy:draft может оказаться неоправданным.

Функционал Articy:draft позволяет назвать его универсальным инструментом благодаря встроенным шаблонам и готовым элементам UI. Articy:draft позволяет разработать быстрые прототипы, но лишь для жанра point-and-click адвенчур или для системы диалогов.

1.3.4 Визуализация текста

На сегодняшний день задача визуализации повествования на основе текста на естественном языке предпринималась ни раз.

В обзоре 2016 года [10] выделены три группы инструментов преобразования текста в визуализацию: текст в картинки, текст в сцену, текст в анимацию. Исходя из нее можно сформулировать выводы по каждой группе инструментов. После обзора дополнительных источников (см. [16; 17]), был сделан вывод, что существует четвертая группа инструментов, применяющих нейросети (см. Табл. 2).

Таблица 2 – Группы инструментов визуализации текста

Визуализация текста в картинки - Story Picturing Engine - Text-to-Picture Synthesis System	Визуализация текста в сцену - Storyboarder
Визуализация текста в анимацию - ScriptViz - SceneMaker	Нейросети в обработке текста - CRAFT - RivetAI

Визуализация текста в картинки

Навыки рисования нужно развивать годами. Использование ПО для создания компьютерной графики не упрощает процесс – нужно разбираться в технологических процессах. А применение программ генерации визуализации текста порой не достаточно успешно из-за двусмысленности исходного текста.

Опыт движка Story Picturing Engine для подбора иллюстрации к тексту показывает, что если извлекать информацию из всей истории целиком – временные и пространственные отношения теряются после анализа [18].

По опыту системы Text-to-Picture Synthesis System синтеза текста в изображение можно сказать, что генерация изображения на каждую фразу дробит историю на картинки, которые выступают в роли хештегов, но делят историю на атомарные части [19]. При этом вновь теряется смысл.

Визуализация текста в сцену

Следующая форма визуализации – генерация по текстовому описанию статичной сцены. Создание сцены включает в себя генерацию окружения, генерацию персонажей, постановку кадра. Один из примеров подобных инструментов – Storyboarder [20].

Storyboarder – инструмент для создания раскадровки с многочисленными настройками. В Storyboarder доступны шесть кистей, настройки кадра и освещения, широкий выбор анимаций, возможность ввести субтитры и заметки к сценам. Кадры можно редактировать в Photoshop. Раскадровку можно экспортировать в формате картинок, pdf и анимации.

Визуализация текста в анимацию

В качестве примеров визуализации текста в анимацию интересны два инструмента: ScriptViz [21] и SceneMaker [22].

ScriptViz разработан в 2005 году. Инструмент способен генерировать трехмерную анимацию в сцене в режиме реального времени. Это значит, что анимация генерируется прямо во время ввода текста.

Стоит отметить, что на вход ScriptViz принимает только грамотный и однозначный текст. Растолковывать двусмысленности он не умеет.

Система состоит из трех модулей: модуль понимая текста, планировщик высокого уровня и генератор сцен.

Модуль понимания текста использует анализатор Apple Pie [23] для получения синтаксической структуры входного текста. Анализатор делит текст на предложения в соответствии с союзами, извлекает глаголы и распознает существительные. Связывание глаголов и существительных происходит с помощью наивного байесовского классификатора.

Модуль планирования высокого уровня генерирует план действий на основе информации, которую он получает от Модуля понимания.

Генератор сцены назначает полученный из Модуля планирования PAR соответствующему агенту, обновляет состояния и визуализирует сцену в режиме реального времени.

ScritpVis написан на Java и использует OpenGL.

Для семантического анализа текста в ScriptViz используется наивное сравнение (naive matching).

Опираясь на разработки предшественников, SceneMaker должен был стать мощным инструментом предварительной разработки фильмов. В нем возможна генерация анимированных трехмерных сцен, персонажи в которой разыгрывают эмоции и настроение, заложенные в сценарие. Сгенерированную анимацию можно отредактировать.

Обработка и генерация осуществляется тремя модулями: модуль понимания, модуль рассуждений и модуль визуализации.

Модуль понимания использует CONFUCIS [24].

Модуль рассуждений использует WordNetAffect [25] для интерпретации контекста, определения эмоций и планирования действий.

Модуль визуализации выбирает трехмерные модели и музыку из базы, генерирует речь, настраивает камеру и освещение.

К сожалению, в научных работах о SceneMaker не представлено ни одного скриншота, чтобы можно было оценить результат проделанной работы.

<u>Нейросети в обработке текста</u>

При использовании технологий искусственного интеллекта получаются более функциональные инструменты. Например, коммерческий инструмент RivetAI [16] для кинематографа и нейросеть для создания мультфильмов CRAFT [17].

RivetAI – анализирует сценарий, генерирует раскадровку и список кадров, применяет методологию Agile для оптимизации подготовки к съемкам, генерирует смету.

Последние новости об инструменте были опубликованы в 2018 году.

Демо инструмента не доступно для изучения.

Composition, Retrieval and Fusion Network CRAFT – модель, обученная более чем на 25000 видео. Модель умеет создавать новые ролики на основе субтитр. Она может составлять сложные сцены, на которых может присутствовать несколько персонажей.

1.3.5 Визуализация разветвленной структуры

В рамках сценарного прототипа автор считает уместным визуализировать не только графический компонент игры, но и извлеченную структуру повествования.

Разветвленная структура – это структура, в которой возможен выбор. Выбор определяет то, по какой ветке продолжится развитие событий.

Структура, которая не имеет ответвлений называется линейной. Это классическая структура для литературы и кино.

В интерактивной литературе жанра «Выбери свое приключение» структура уже разветвленная. В такой книге можно выбрать следующий отрывок. В интерактивном кино разветвленная структура чаще бинарная.

Выбор не обязан порождать большое количество концовок. Чаще выбор влияет не на сиюминутные события, а на будущие последствия. В качестве примера, что игрок выберет на завтрак, мюсли или хлопья? В этом выборе не заложен сюжетообразующий ход. Но игрок почувствует свое участие. Он становится режиссером своей собственной истории.

В играх разного жанра встречаются как простейшие так и сложные структуры. Все зависит от того, какой у проекта бюджет.

В случае с инди-проектами[11] разветвленность небольшая, мнимая. Такая иллюзия создается за счет нескольких приемов: альтернативные или эмоциональные реплики в диалогах, кастомизация. В первом случае отыгрыш не имеет последствий, события будут развиваться линейно. Во втором случае игрок проявит свою индивидуальность, но это также не повлияет на сюжет.

В более дорогостоящих проектах каждый выбор может влиять на появление новой концовки. Большое количество концовок характерно для игр студии Quantic Dream[12]. Heavy Rain, Beyond: Two Souls и Detroit: Become Human имеют большое количество концовок, на разработку каждой из которых было потрачено немало ресурсов.

Чаще всего разработчики используют средние по сложности структуры, которые содержат как выбор, так и иллюзию выбора.

На данный момент существуют различные решения для визуализации разветвленной структуры. У них разные свойства и функциональность.

---

[11] Инди-проект – проект независимой студии разработки.

[12] Quantic Dream – студия разработки игр. Режим доступа: https://www.quanticdream.com/ru (дата обращения 5.05.2020)

Всего их несколько: граф, пряжа и визуальное программирование.

Граф – это самая распространенная визуализация. Такая структура поддерживает любую сложность выбора. Примеры графов представлены в пунктах 5.2–5.12.

Пряжа, yarn – это структура в авторской разработке Extended Story Flow [26]. Эта структура учитывает нелинейность повествования, временные параметры и персонажей, участвующих в событиях (рис. 3). События сценария располагаются в структуре в хронологическом порядке слева направо. Пересечение линий персонажей означает их встречи. Структура позволяет поддерживать повествование с воспоминаниями (флешбеками) и будущими событиями (флешфорвордами).

Рисунок 3 – Визуализация нелинейной структуры The Witcher 3

Визуальное программирование – это способ создания программ с помощью визуальных элементов. Такая структура естественно образует структуру повествования во время разработки. Примеры программ с визуальным программированием повествования: Twine, Articy:draft, Orange [27].

Orange – инструмент с открытым исходных кодом. Визуальное программирование в Orange позволяет выстраивать процессы анализа данных. Полученный результат инструмент визуализирует. Инструмент содержит методы машинного обучения и интеллектуального анализа.

Чтобы определить, какие именно визуализации повествования могут быть, проанализированы существующие задокументированные структуры (см., например, [28; 29]).

Time Cave

Самая дорогая в разработке структура Time Cave (рис. 4) содержит в себе большое количество ответвлений, каждое из которых заканчивается отдельной концовкой игры.

Такая структура порождает огромное количество контента. Если игра не реиграбельна, то смысл в такой структуре теряется.

Примеры игр со структурой Time Cave: Heavy Rain, Beyond: Two Souls, Detroit: Become Human.

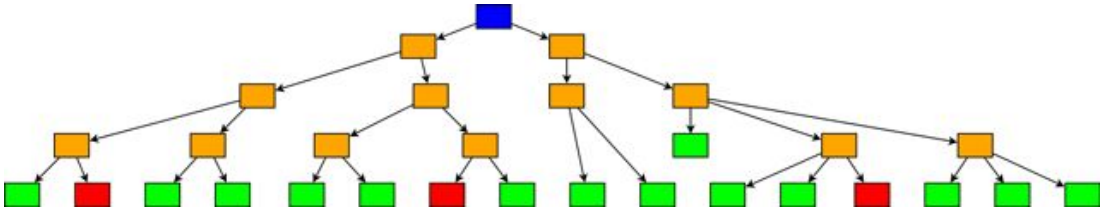


Рисунок 4 – Time Cave

Алмазная структура

У алмазной структуры есть другое название – бутылочное горлышко (рис. 5). Суть структуры в том, что сколько бы она не ветвилась, в конце концов все варианты приведут к одному узкому бутылочному горушку. Нужно это для того, чтобы снизить сложность и стоимость разработки.

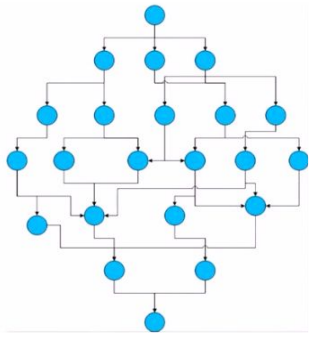


Рисунок 5 – Алмазная структура

<u>Жемчужная нить</u>

После бутылочного горлышка может следовать новая череда разветвления сюжета. Серия алмазных структур образует жемчужную нить (рис. 6).

Жемчужная нить может использоваться в диалогах, квестах и других структурах игры.

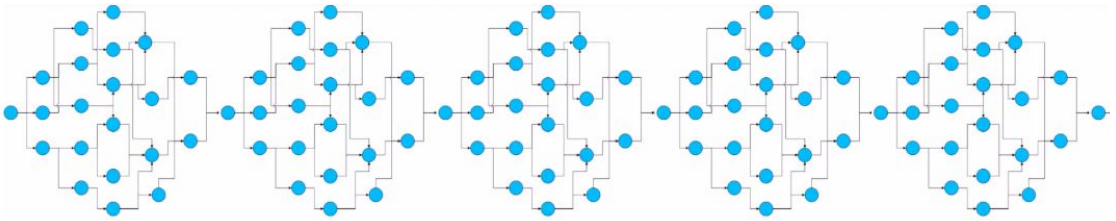


Рисунок 6 – Жемчужная нить

<u>Sorting Hat</u>

В структуре Sorting Hat есть начальный и конечный узел (рис. 7). В начальном узле есть выбор, который определяет, какую из веток пройдет герой до конечного узла.

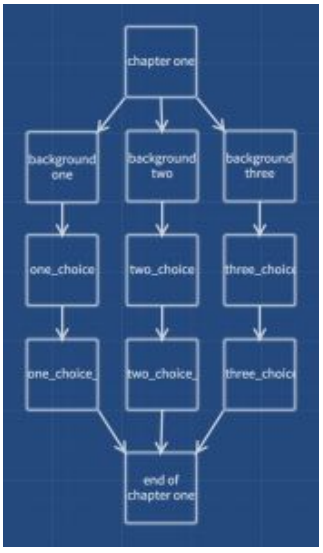


Рисунок 7 – Sorting Hat

Такую структуру использовали, например, в Dragon Age: Origins. В зависимости от выбора персонажа игрок проходит уникальное обучение. В

сумме шесть предысторий, каждая из которых относится к определенной расе и классу персонажа.

Gauntlet

В структуре Gauntlet (англ. перчатка или рукавица) проходить побочные ветки необязательно (рис. 8). В таких разветвлениях может быть уточняющая информация о мире игры или пасхалки, которые делают игру живой, но не влияют на прохождение.

Структура может использоваться в дизайне диалогов, уровней и игры в целом.

Структура названа Gauntlet из-за того, что ответвления похожи на пальцы перчатки.

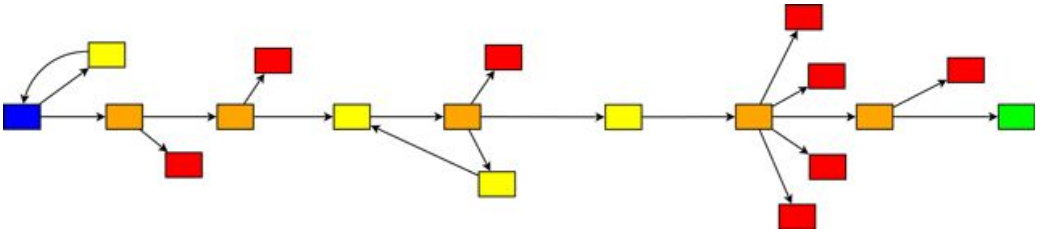


Рисунок 8 – Gauntlet

Квест

Структура Квест – это структура одноименного жанра (рис. 9). В квесте есть цель и шаги к ее достижению. Обычно в играх жанра RPG[13] есть персонаж квестгивер, который дает игроку квест. Чтобы сдать квест, нужно вернуться к этому же персонажу.

Серия квестов может образовывать основную сюжетную ветку игры.

[13] RGP – role-playing game, жанр ролевая видеоигра.

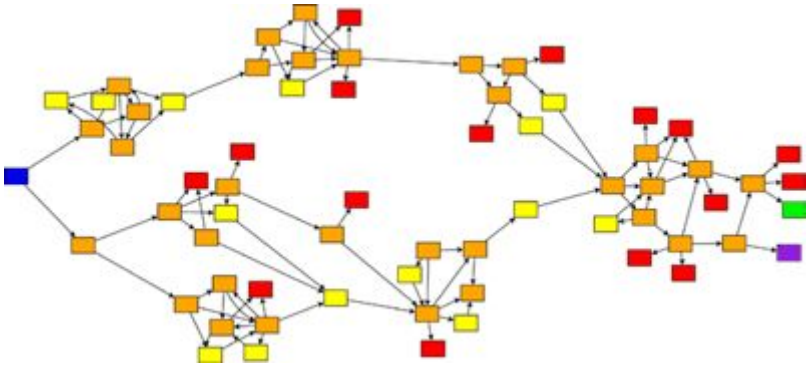


Рисунок 9 – Квест

<u>Открытый мир</u>

Структура открытого мира предполагает начало и конец истории, но не имеет строгой основной сюжетной ветки (рис. 10). Игры вроде The Witcher 3 или TES: Skyrim, имеют открытый мир, но при этом и основную сюжетную ветку. Из-за этого в игре есть определенный конфликт: сюжет торопит игрока участвовать в истории, а открытый мир с множеством возможностей, останавливает его и привлекает внимание своими квестами и возможностями для активности.

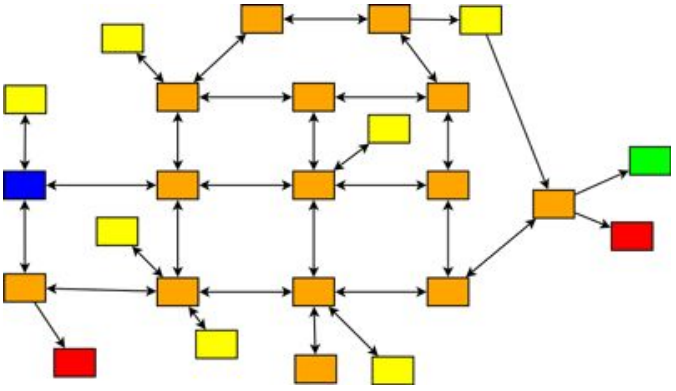


Рисунок 10 – Открытый мир

<u>Re-enterable Conversation Node</u>

Данная структура применяется в диалогах (рис. 11). Структура дает игроку возможность посмотреть все доступные варианты или воспользоваться единственным, в зависимости от контекста.

Такие диалоги часто встречаются в RPG серии Divinity или Dragon Age. Для дизайнеров это возможность обогатить мир дополнительной информацией, которая улучшить погружение в мир. В то же время, игрок свободен в собственном решении о том, как много он хочет узнать из диалога, а какой информации будет достаточно.

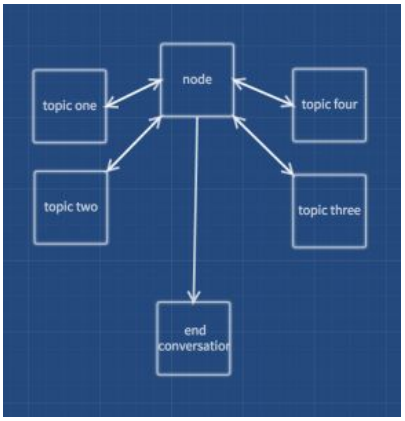


Рисунок 11 – Re-enterable Conversation Node

<u>Loop and Grow</u>

Циклическая структура используется в симуляторах жизни, например, The Sims, в котором каждый день сим должен выполнять ежедневную рутину (рис. 12).

Иногда циклическая структура подается в неожиданном виде. например в The Stanley Parable зациклено не только повествование, но и уровень. Каждый раз, когда игрок совершает действие, отличное от запланированного нарратором, он приходит к неожиданным последствиям.

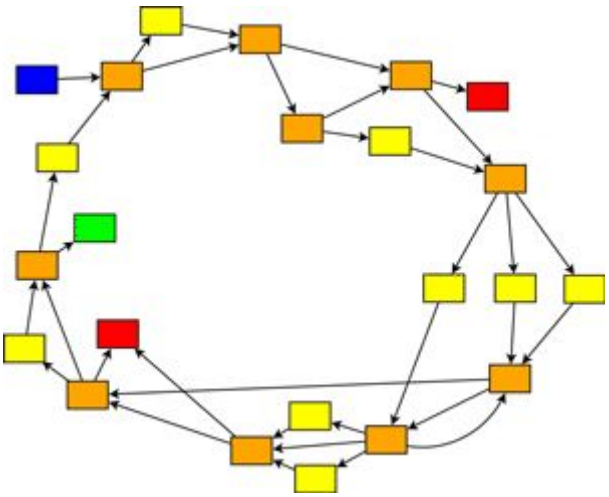


Рисунок 12 – Loop and Grow

<u>Confirmation-required Choice</u>

Иногда, чтобы убедиться в том, что игрок совершает осознанное действие, необходимо создать условия, в которых он должен подтвердить это (рис. 13).

Бросать игрока в небезопасные для его текущего уровня условия так же плохо, как и давать бесконечную свободу. Данная структура спасает от таких ситуаций, а также от ошибок и неосторожных решений игрока.

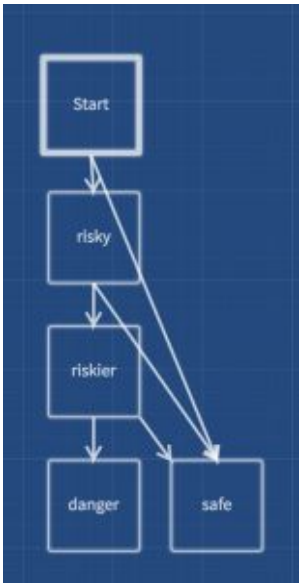


Рисунок 13 – Confirmation-required Choice

Track Switching Choice

Игры позволяют игрокам почувствовать себя в роли кого-то, кем они не могут быть в реальности. Более того, игры по сюжету ставят игрока в положение, в котором он должен выбрать сторону. Чтобы сделать прохождение игры уникальной для каждой категории игроков, можно вести подсчет параметров.

Один из вариантов параметров – выбор реплик в диалогах (рис. 14). Такое решение реализовано в игре Mass Effect. Каждая реплика увеличивает один из параметров и игрок переходит в роль, в которой число больше всего. Игрок может менять сторону в течении игры, что увеличивает погружение и создает возможность почувствовать себя в разных ролях.

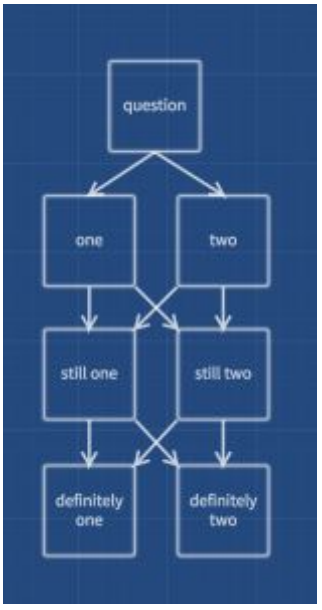


Рисунок 14 – Track Switching Choice

Глава 2. Проектирование технического решения

2.1 Требования к инструменту генерации сценарного прототипа

Работа с повествованием с применением инструмента генерации выглядит следующим образом:

- создание сценария игры в формате технической документации – текст, таблицы, wiki-разметка;
- генерация сценарного прототипа по технической документации;
- обсуждение результатов генерации внутри команды разработки;
- тестирование;
- редактирование сценарного прототипа или генерация нового.

Для разработки инструмента, который способен охватить весь процесс генерацией сценарного прототипа, был выделен ряд задач. Алгоритмы обработки текста подразумевают готовые решения или разработку собственных. Под данными на вход инструменту подразумевается документация:

- извлечение сущностей и связей между ними: персонажи, локации, артефакты, реплики, события;
- извлечение разветвленной структуры повествования;
- извлечение информации о постановке персонажей и их анимациях;
- извлечение информации о составе локаций;
- извлечение информации о постановке кадра;
- извлечение информации о настройках освещения в сцене;
- визуализация разветвленной структуры повествования;
- автоматическая сборка интерактивного сценарного прототипа в зависимости от жанра игры;
- экспорт готового сценарного прототипа в игровые движки;

- разработка набора моделей женского и мужского рода разного возраста: младенец, ребенок, подросток, молодой, взрослый, старый.
- разработка минимального набора анимаций для моделей: стоять, идти, бежать, сидеть, лежать, разговаривать.
- разработка примитивов[14] разных форм и размеров: кубы, параллелепипеды, сферы и т.п..
- набор стандартных артефактов или алгоритм поиска готовых моделей. Во втором случае необходим алгоритм, который упрощает геометрию[15] модели и создает стандартную текстуру[16].
- разработка стандартного UI: отображение реплики, выбор реплик и действий, подсветка для интерактивных объектов, подписи, сноски.

Еще один возможный компонент инструмента – алгоритм генерации диаграммы баланса Machination [30]. Данный компонент считается уместным в случае, если прототип генерируется вместе с игровыми механиками.

Разработка инструмента представляет собой серьезный проект, на который уйдет много времени. На данный момент части этого проекта реализованы в ряде работ. Подробнее они описаны в Гл.3.

2.1.1 Требования к документации

Алгоритмы, от которых сильно зависит результат генерации, это алгоритмы обработки текста. Добиться удовлетворительного качества анализа текста можно с помощью продвинутых алгоритмов обработки с одной стороны и с помощью хорошо отредактированного текста с другой.

<u>Алгоритмы обработки текста</u>

По результатам исследования инструментов визуализации текста 2016 года выяснилось, что кроме существующих проблем понимания естественного

[14] Геометрические примитивы – простейшие объекты, из которых возможно построение более сложных форм и моделей.

[15] Геометрия модели – набор точек, которые составляют в пространстве трехмерный объект.

[16] Текстура модели – изображение, которое покрывает форму модели, придает ей цвет.

языка, которые ограничивают качество визуализации, существуют препятствия, которые инициируют авторы систем [10]. Оказалось, что большинство авторов не используют существующие ресурсы и доступные методы, которые могли бы улучшить результат. В число доступных ресурсов входят, например, лексические и семантические сети. Среди примеров актуальных методов активное обучение, поверхностный семантический анализ и алгоритмы сквозного обучения.

При попытках построения системы для визуализации событий, которые построены по разветвленной структуре, появляются вопросы, специфичные для этой области.

Среди них вопросы, связанные с анализом игровой документации:

- стандартного способа документирования в игровой разработке и в сфере разработки интерактивных приложений нет. Каждая компания формулирует свои требования к документам.
- для анализа документации методами машинного обучения, необходимы аннотированные корпуса. На данный момент таких корпусов не существует. Создание корпусов ограничено тем, что не так много документации находится в свободном доступе.
- представить разветвленную структуру в виде текста сложно. Поэтому кроме текста необходимо анализировать таблицы, wiki-разметку, файлы в форматах xml, json, html и даже python код.

Обработка текста документации, также, как и любого текста на естественном языке, включает в себя последовательное проведение следующих процессов:

- графематический анализ – выделение предложений и токенов,
- морфологический анализ – выделение основ слов,
- синтаксический анализ – выделение грамматических отношений в предложении,

- поверхностный семантический анализ – выделение семантических отношений в предложении,
- глубинный семантический анализ – выделение семантических отношений между приложениями,
- прагматический анализ – выделение смысла всего текста.

В игровой документации одной из главных задачи является выделение именованных сущностей. Это могут быть названия географических объектов, имена персонажей, названия артефактов, названия исторических внутриигровых событий, время и даты и т.п..

Кроме извлечения таких именованных сущностей, как персонажей, нужны данные о том, какие действия они осуществляют.

Для анализа текста игровой документации так же важно построение структуры всего повествования. Это могут быть связи между локациями, персонажами, событиями и интерактивными объектами.

Главное требование к алгоритмам обработки документов – сохранение сути. Алгоритмы должны устранять неоднозначность, связывать происходящие действия в сцены, учитывать пространственно-временные связи, учитывать монтаж и склейки, ракурс, темп.

Форматирование текста

Тексты на естественном языке не всегда подчиняются формальным правилам. Понимание такого текста в NLP относит к AI-полной задаче[17]. Это значит, что на сегодняшний день решить эту задачу с максимальной точностью невозможно.

Создание сценарных прототипов связано с применением игровых движков.

[17] AI-полная задача – задача, которую может решить только сильный ИИ.

Для однозначной интерпретации повествования в игровых движках используется разметка и языки программирования. В случае с Twine – это html-разметка, RenPy – python, Articy:draft – визуальное программирование. В движках не нужны алгоритмы интеллектуального понимания текста, т.к. в широком смысле текст написан по правилам.

Таким образом, создание повествования в игровой разработке сводится к программированию. В таком случае справедливо следующее: текст, написанный по правилам, обязательно будет понят внутренним алгоритмом обработки игрового движка; текст с ошибками придется отлаживать разработчику. Второй вывод ведет к тому, что творческий процесс превращается в рутину. Написание текста обязательно включает в себя этап редактирования, но этап форматирования вводит дополнительную нагрузку на писателя, сценариста или их коллег-программистов.

Если не вносить в процесс написания дополнительных этапов, то выходит, что хорошим текстом на вход программы обработки текста, это текст, который хорошо отредактирован. Такие правила можно найти в учебниках для редакторов. Один из них – Пиши, сокращай [31].

В данной книге вводится понятие инфостиля. Его главный принцип: текст должен быть написан с уважением к читателю. Данное правило приводит к тому, что текст приходит в вид, который уважителен не только к читателю, но и к алгоритмам, которые могут обработать этот текст с большим успехом.

Одно из правил инфостиля, которое подтверждает эту теорию, заключается в следующем: в одном предложении должна быть одна мысль, а в одном абзаце – одна тема. Согласно этому правилу за редким исключением в предложении не должно быть больше трех конструкций подлежащее-сказуемое.

Конструкция подлежащее-сказуемое транслирует информацию о том, кто является действующим лицом и какое действие он совершает. Данная информация является ключевой для генерации визуализации.

Таким образом формулируется вывод о том, что, возможно, не стоит вводить ограничения на форматирование текста. В противовес этому достаточно хорошей редактуры, которая приведет текст в вид, понятный и людям, и алгоритмам.

2.1.2 Извлекаемые данные

В результате анализа текста для генерации сценарного прототипа необходимо получить следующие сущности: персонажи, отношения между персонажами, локации, наполнение локаций объектами и их пространственными позициями, артефакты, реплики, события.

У персонажей должны быть известны пол, имя и характеризующие его черты: раса, сверхспособность, цель, характер, предпочтения и др..

У локации должно быть известно название, ключевые точки, относительное расположение, наполнение.

У артефактов должны быть известны название, предназначение, местоположение и первое появление в истории.

Реплики и события формируют структуру истории. Необходимо знать, кто и когда их инициирует, сколько они длятся по времени, какая у них последовательность подачи.

Кроме этого их текста можно извлечь информацию о времени, погоде и другие подробности, которые помогут сформулировать требования к освещению, постановке кадра, настройках монтажа, как это происходит, например, в SceneMaker.

Информация для визуализации может поступать в виде структур. Например, графов. Это могут быть карты локаций, дерево диалогов, структура

сюжета. Такие структуры помогут при построении сценария визуализации. Такие структуры также можно визуализировать в виде, удобном для просмотра.

2.2 Требования к компоненту визуализации

Визуализация – это один из процессов, который происходит в инструменте, который должен генерировать сценарный прототип из текста.

Перед визуализация происходит анализ текста. После того, как данные получены, происходит подбор из базы готовых моделей. Завершающий этап – сборка проекта и отрисовка результата.

Главное требование для сценарного прототипа – передать видение разработчиков целевой аудитории и получить от них фидбек. В такой системе отношений может возникнуть недопонимание, связанное с тем, что прототип – это всего лишь черновик.

При создании прототипа вручную видение разработчиков эволюционирует. Но сгенерированная визуализация не сможет передать авторский стиль, задумку, атмосферу с высокой точность. При генерации визуализации используются те данные, которые уже есть: аннотированные модели, фотографии и двумерные ассеты. Такой подход может привести к разочарованию целевой аудитории. Если игрокам не понравится графика, может сформироваться предвзятое отношение к повествованию и игровому процессу. Тестирование в таком случае возможно только при наличии наработок художников, на работу которых уйдет время и ресурсы.

Чтобы избежать проблемы генерации визуализации без учета авторского стиля можно использовать обезличивание. Когда визуализация исключительно функциональная, пользователям нужно вообразить себе антураж. Такой подход не сковывает разработчиков непредсказуемыми ожиданиями игроков.

Обезличивание применяется в левел-дизайне. Этот подход называется блочным дизайном[18]. Суть применения метода – тестирование функциональности уровня без учета его эстетичного вида.

Блочный дизайн – это примитивы без текстур. Это нагромождения серых кубов и параллелепипедов, которые образуют уровень.

Адаптировать данный подход для сценарного прототипа необходимо для следующих составных: локация, персонаж, артефакты.

Локации – аналог блочного дизайна – кубы и параллелепипеды, которые наполняют сгенерированную сцену в зависимости от описания. Например, лес – это неограниченная карта, на которой разбросаны деревья в виде параллелепипедов, пни и камни в виде кубов. Или учебный класс – это коробка, в которой есть выход, а также парты и стулья в виде примитивов.

Персонажи – в зависимости от графики – двумерные или трехмерные болванки. У них не должно быть мимики, но должны быть стандартные анимации. В качестве анимаций могут быть следующие действия: идти, бежать, сидеть, читать, разговаривать, прыгать и т.п.. Кроме анимаций действий у персонажей могут быть позы, которые характерны для действий. Болванки должны как минимум отражать возраст и пол персонажа. Чтобы различать болванки, можно окрасить их в стандартные цвета. Еще лучше указывать имена, которые персонажи имеют по сценарию при отображении их реплик.

Артефакты – самый сложный элемент визуализации. С одной стороны это могут быть конкретные предметы. Если в визуализации должен присутствовать меч, то в качестве артефакта должен отрисовываться стандартный меч без текстур. Но это усложняет процесс визуализации. Другая крайность – использование кубов и параллелепипедов и подписи. Такой способ может сбить игроков с толку или напротив способствует развитию их фантазии о происходящем.

[18] Designer block out – grey box, макет уровня для проведения тестирования игрового процесса.

На основании этого анализа можно прийти к выводу, что блочный дизайн хоть и не реализует видение, но уменьшает риски, связанные с ожиданием целевой аудитории.

Другое требование к визуализации связано с дизайном интерфейса. Articy:Draft предлагает свои шаблоны для элементов интерфейса. Они разработаны для воспроизведения диалогов на манер визуальных новелл. Данный подход не считается универсальным с точки зрения прототипирования. Отсутствие стилей и создание минималистичного дизайна стандартных элементов интерфейса устранит этот момент. В качестве элементов могут быть различные индикации, кнопки и диалоговое интерфейсы.

2.3 Концепция пилотного решения компонента визуализации

Основное требование, которое должно быть реализовано в разработке пилотного решения – генерация анимации персонажей по текстовому описанию.

В разработке не учитывается интерактивность структуры повествования и извлечение информации о постановке кадра. Ключевая цель – доказать возможность и эффективность генерации визуализации для сценарного прототипа.

Визуализация должна соответствовать блочному дизайну.

Результат должен быть аккуратным.

Должна быть возмозможность для экспорта полученного результата для последующей проработки.

2.4 Постановка задачи

Цель

Разработать решение для генерации визуализации сценарного прототипа игры, входные данные для которого извлекаются из повествовательного текста.

Задачи

- проанализировать актуальность инструмента генерации сценарного прототипа;
- проанализировать текущие решения визуализации повествования из текста;
- проанализировать разветвленные структуры повествования;
- проанализировать текущие решения визуализации разветвленной структуры повествования;
- сформулировать требования для инструмента генерации сценарного прототипа;
- сформулировать требования для компонента визуализации сценарного прототипа;
- сформулировать гипотезу эффективности генерации визуализации сценарного прототипа;
- реализовать пилотное решение компонента визуализации;
- проанализировать полученный результат.

2.5 Гипотеза и эксперимент

Гипотеза. Инструмент для генерации раскадровок является одним из эффективных инструментов для создания сценарного прототипа.

Эффективный, т.к. сокращает время разработки.

Эксперимент

- создать прототипное решение, который моделирует генерацию визуализации сценарного прототипа на основе текста;
- сгенерировать раскадровку на основе сценария;
- зафиксировать время на генерацию раскадровки;
- для сравнения создать раскадровку вручную;

- сравнить время на генерацию раскадровки и создание раскадровки вручную;
- сформулировать выводы, подтвердить или опровергнуть гипотезу;
- сформулировать требований к дальнейшей разработке.

Задачи

Для разработки прототипного решения был выбран инструмент Storyboarder. В связи с этим построен следующий план:

- провести реинжиниринг процесса генерации кадра по текстовому запросу в Storyboarder;
- подключиться к функционалу генерации кадров;
- написать алгоритм обработки текста с учетом ограничений Storyboarder;
- сформулировать тестовый запрос и проверить получившийся результат на соответствие.

Процесс генерации сценарного прототипа включает следующие задачи:

- создать сценарий;
- сгенерировать кадры для сценарного прототипа;
- собрать раскадровку;
- зафиксировать время на генерацию раскадровки.

Для проведения ручного прототипирования реализуются следующие задачи:

- создать кадры в Storyboarder вручную;
- зафиксировать время на создание раскадровки.

Кроме этого была сформулирована задача разработки требований к полноценному инструменту генерации сценарного прототипа.

Глава 3. Техническая реализация

3.1 Архитектура инструмента

Инструмент генерации сценарного прототипа – это комплекс модулей, каждый из которых решает свою задачу (рис. 15).

Рисунок 15 – Архитектура инструмента генерации сценарного прототипа

В рамках работы автора 2018 года реализованы следующие компоненты и функционал [32] (рис. 16):

- Модуль ввода.
- Модуль редактирования текста.
- Модуль распознавания сущностей и связей.
- Компонент категоризации в ручном режиме.
- Режим ручной сборки сценарного прототипа.

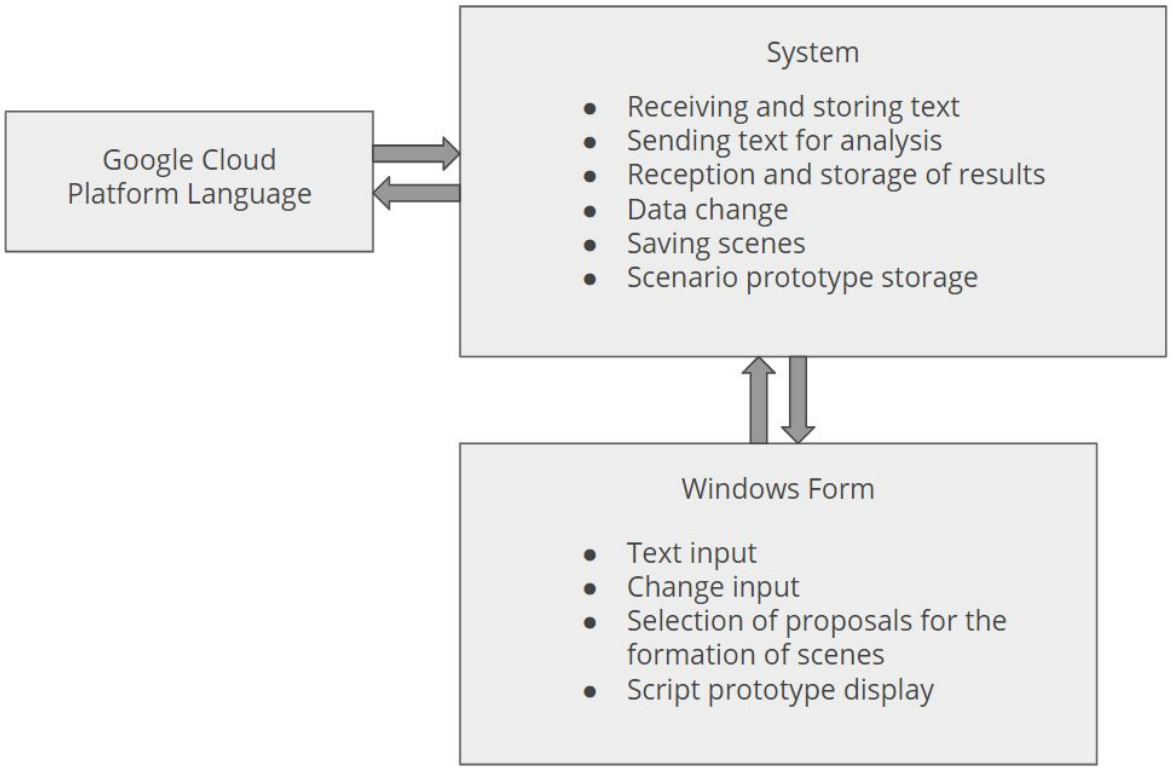


Рисунок 16 – Схема работы инструмента [32]

В рамках работы [33] 2019 года реализованы следующие компоненты и функционал (рис. 17):

- Модуль отображения структуры, сущностей, статистики.
- Модуль распознавания сущностей и связей.
- Модуль анализа и сборки структур кроме компонента построения структуры сценарного прототипа.
- Компонент визуализации структуры.

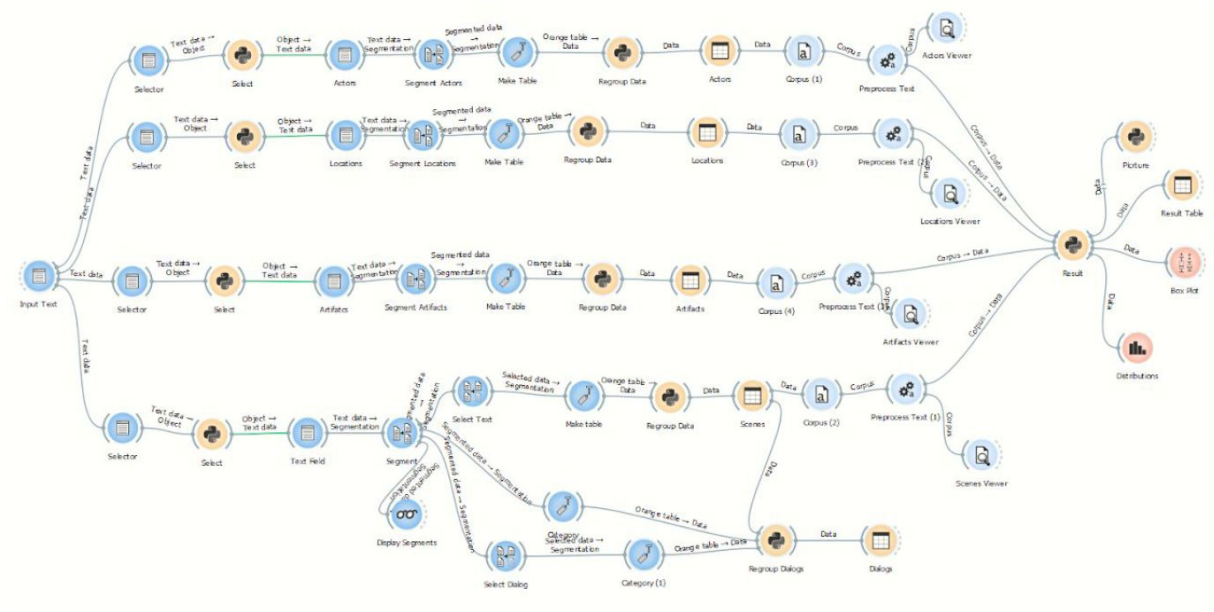


Рисунок 17 – Схема работы инструмента [33]

В рамках работы [34] 2020 года реализован компонент для приема тегов для постановки кадра.

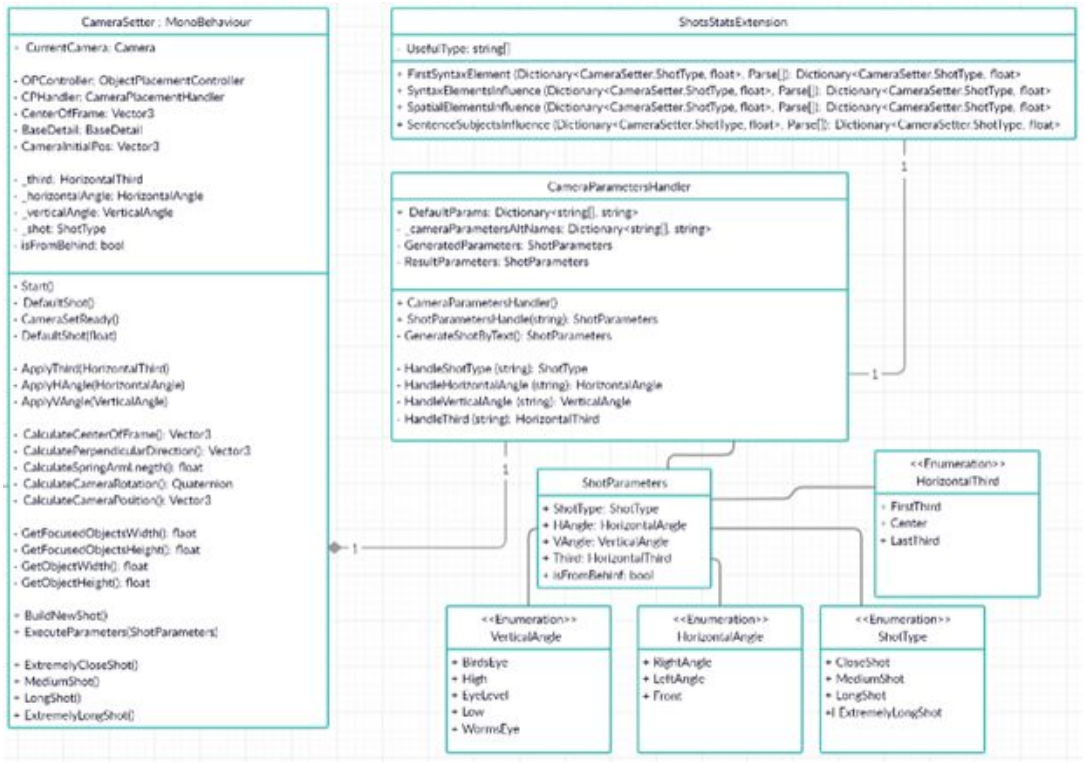


Рисунок 18 – Группа компонентов, отвечающих за тип съемки в [34]

В рамках работы [35] 2020 года реализован компонент генерации пространства.

Не прототипировались:

- Компонент редактирования сгенерированного сценарного прототипа.
- Компонент редактирования разветвленной структуры.
- Компонент запуска сценарного прототипа.
- Компонент запуска балансных диаграмм.

В рамках этой архитектуры в данной работе представлен компонент генерации визуализации событий в пространстве (Рис. 19).

Рисунок 19 – Схема работы компонента

3.2 Выбор средств реализации

Из изученных инструментов Storyboarder был отмечен в качестве инструмента, который больше всего удовлетворяет требованиям, предъявляемым к компоненту визуализации событий в сцене.

Storyboarder обладает следующими преимуществами:

- визуализация соответствует принципам блочного дизайна;
- есть функция ввода запроса на генерацию кадра в виде текстового запроса;
- есть настройки постановки кадра, освещения;
- есть возможность экспорта в виде серии картинок.

Состав запроса для генерации кадра в Storyboarder включает следующие параметры:

Настройки кадра:

- Shot type: 9.
- Horizontal composition: 4.
- Vertical angle: 5.
- Horizontal angle: 4.
- Fov: 4.

Настройки освещения:

- Light direction: 9.

Настройка моделей:

- Content: 5.
- Head direction: 5. Настройка только для первой модели в сцене. Остальные модели в состоянии по умолчанию.
- Pose: 106. Настройка только для первой модели в сцене. Остальные модели в состоянии по умолчанию.
- Model: 3.

Настройка окружения:

- Background: 5.
- Room size: 6.

По одному и тому же запросу может быть сгенерировано несколько кадров. Из них можно выбрать наиболее удачный и включить в раскадровку.

Примеры кадров, сгенерированных для запроса: man, run, looking forward, medium long, single person, centered, right angle, eye level, long lens, light, frontrightlit, outside (рис. 20).

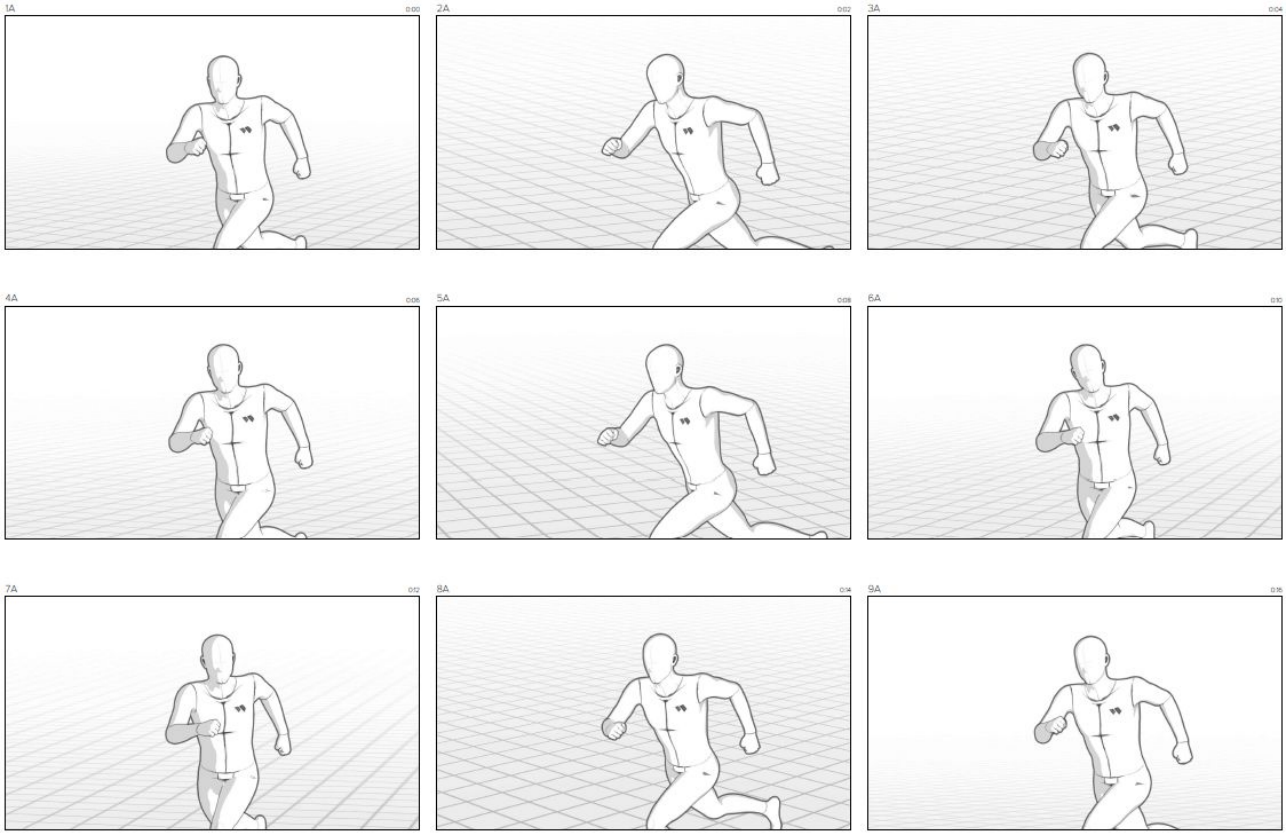


Рисунок 20 – Примеры кадров

Ограничения Storyboarder в следующем:

- нет примитивов в виде кубов или параллелепипедов;
- ввиду этого анимации моделей не привязаны к окружающим объектам, персонаж в позе «сидеть» сидит в воздухе;
- из-за отсутствия примитивов также нельзя сгенерировать окружение, например, лес. Сцена в Storyboarder генерируется либо пустая, либо с комнатой. Есть выбор размера комнаты;
- нет примитивов, которые можно использовать в качестве артефактов;
- генерируется не анимация, а раскадровка;
- генерация в Storyboarder происходит только на основе запроса на английском.

Ограничение о том, что генерируется не анимация, а раскадровка, компенсируется тем, что данная разработка является экспериментальной. Планируемый к разработке инструмент устранит эффект этого минуса наличием функционала генерации анимированной визуализации.

3.3 Обработка текста

В рамках эксперимента визуализируется повествование, представленное в виде текста. Каждое предложение должно включать в себя персонажа и его действие.

В качестве библиотеки машинного обучения используется Scikit-learn[19] для языка программирования Python.

Для обработки текстов используется пакет библиотек NLTK[20] для языка программирования Python.

Для обработки и анализа данных используется программная библиотека pandas[21]. Относительно NumPy[22] это библиотека высокого уровня.

Текст преобразуется в корпус предложений [36]. Корпус в простом смысле – это множество текстов. Тексты подбираются и обрабатываются по определенным правилам. В дальнейшем такая база используется для исследования языка: для проверки гипотез, для статистического анализа, для обучения или подтверждения лингвистических правил

Каждое предложение будет составлять один кадр. Обоснование приведено в Гл. 2.

Storyboarder принимает на вход текстовый запрос, сформулированный по определенным правилам.

Параметры генерации кадра: Model, Pose, Shot type, Content, Horizontal composition, Horizontal angle, Vertical angle, Fov, Head direction, Background,

---

[19]Scikit-learn – Режим доступа: https://scikit-learn.org/stable/ (дата обращения 5.05.20)
[20]NLTK – Режим доступа: https://www.nltk.org/ (дата обращения 5.05.20)
[21] pandas – Режим доступа: https://pandas.pydata.org/ (дата обращения 5.05.20)
[22] NumPy – Режим доступа: https://numpy.org/ (дата обращения 5.05.20)

Light direction, Room size. Не обязательно соблюдать последовательность, но, если какого-то параметра нет, Storyborder выбирает любое значение параметра из доступных.

Чтобы привести текст в такой вид, необходимо провести токенизацию и фильтрацию.

В результате токенизации предложения разбиваются на массив слов.

Слова, которые не влияют на конечную визуализацию, удаляются из текста в результате фильтрации. Такие слова называются стоп-словами. В русском языке это частицы – то, бы, же и др.. В английском – артикли – a, an, is, are и др.. Список стоп-слов приведен в приложении работы 2018 года [37]. Фильтрация происходит итеративно. Слова сравниваются посимвольно.

Кроме токенизации и фильтрации необходимо определить, какие персонажи участвуют в действии. В качестве моделей в Storyboarder представлены только модели мужчины и женщины. Если в тексте встретятся такие слова, они останутся без преобразований. Но если в тексте персонажи имеют имена, их необходимо преобразовать до понятных Storyboarder man и woman, для модели мужчины и женщины соответственно. Для этого используется классификатор.

В качестве классификатора используется мультиклассовый наивный байесовский классификатор. Использование данного классификатора обусловлено тем, что выбор имен не закономерен, некоторые имена не зависят от пола. Для обучения классификатора используются имена из публичной базы Службы социального обеспечения США, т.к. тексты написаны на английском языке [38].

Для использования классификатора каждое предложение корпуса преобразуется в вектор чисел с минимальной потерей информации, заложенной в текст.

В обучающей выборке все имена векторизируются в символьные биграммы. Кроме этого проведен анализ частоты использования имени в виде мужского и женского.

Затраты на время работы минимальны, т.к. обучение наивного байесовского классификатора – это вычисление независимых вероятностей, произведение которых лежит в знаменателе формулы по теореме Байеса [36].

При анализе векторов обученный байесовский классификатор возвращает ноль, если содержащееся в предложении имя женское, и единицу – если мужское.

Каждый раз, когда классификатор находит имя, алгоритм обработки заменяет его на понятные Storyboarder man и woman. Ограничение обусловлено тем, что парсер различает только мужскую или женскую сущность, а сам Storyboarder имеет только мужскую и женскую модель из тех, которые интересны в рамках разработки инструмента.

После всех этапов обработки текста, от каждого предложения остается два компонента, формирующих запрос для Storyboarder: тип модели и действие.

3.4 Визуализация

Настройки камеры, освещения и окружения вводятся вручную. Прототипирование данного функционала было проделано в работах [34], [35].

Генерация кадра в Storyboarder происходит по ряду параметров. В одной из версий параметры можно было вводить в виде текстовых запросов. В этой версии Storyboarder может интерпретировать несколько вариантов использования слов. Например, тип кадра medium можно ввести как 'ms', 'medium', 'medium shot', 'waist', 'mid', 'med'.

Слепок коммита, в котором есть функция генерации кадра по текстовой строке – abbf5f24 [40]. Данная версия стала доступна в июне 2019 года.

Чтобы отправить в Storyboarder запрос на генерацию кадра, был проведен реинжиниринг. По результатам реинжиниринга, чтобы включить Storyboarder в процесс генерации раскадровки, было принято решение изолировать необходимый функционал Storyboarder.

Изолированный функционал Storyboarder позволяет вызывать процедуру генерации кадра по необходимому запросу.

Алгоритм генерации кадра состоит в следующем:

- текст обрабатывается вне Storyboarder,
- сформированный запрос посылается в Storyboarder в качестве аргумента,
- Storyboarder принимает запрос и генерирует по нему кадр.

Для связи с Storyboarder внедрен микро веб-сервер express[23]. Он принимает на loopback интерфейса запрос с параметром в виде строки, сформированной для Storyboarder. После этого запрос отправляется в изолированный функционал Storyboarder. Запросы отправляются до тех пор, пока не закончатся предложения. В качестве успешного ответа о генерации кадра express отправляет принятую строку обратно.

[23]express – Режим доступа: https://expressjs.com/ru/ (дата обращения 5.05.20)

Глава 4. Обсуждение результатов

4.1 Результаты эксперимента

В качестве эксперимента был взят сценарий существующего проекта [41]. Сценарий был переписан в вид, в котором есть необходима информация для генерации кадра в Storyboarder (Прил. C).

После обработки этого варианта сценария сформировались запросы для Storyboarder с подготовленными настройками камеры (Прил. D). По полученным кадрам была собрана раскадровка (Прил. B).

Вместе с воспроизведением раскадровки процесс занял 42 секунды.

Для сравнения, текст, подготовленный для Storyboarder использовался для построения раскадровки вручную. Настройки камеры использовали такие же, как и для генерации раскадровки.

Время создания раскадровки вместе с воспроизведением заняло 236 секунд.

4.2 Эффективность системы

Генерация раскадровки оказалась быстрее ручной сборки в 6 раз.

Эксперимент с ручным созданием раскадровки был проведен в значительно облегченной форме. В качестве сценария использовался текст для входа в генератор. Настройки камеры также были взяты из документа для генерации. В действительности времени может уйти намного больше.

Гипотеза о том, что инструмент для генерации раскадровок является одним из эффективных инструментов для создания визуализации сценарного прототипа подтверждается.

4.3 Ограничения системы

Сложность разработки системы генерации сценарного прототипа – извлечение информации об интерактивности. Как было сказано ранее, иммерсивность интерактивной среды выше, чем иммерсивность исключительно визуальная.

Кроме того, визуализация, полученная на выходе из Storyboarder получилась достаточно ограниченной по следующим причинам: анимацию можно задать только для одного объекта в сцене; в анимации не участвуют другие объекты; регулирование персонажей по высоте и положению невозможно.

Перечисленные ограничения можно преодолеть в рамках разработки авторского инструмента.

4.4 Дальнейшее развитие

Развитие компонента визуализации будет реализовано в рамках дальнейшей разработки инструмента генерации сценарного прототипа.

Для решения задачи извлечения информации из текста необходимо выбрать самые актуальные и эффективные инструменты или разработать собственный подход.

Для визуализации на основании извлеченной информации необходимо реализовать компоненты генерации структуры сценария, а также визуализации анимированных сцен в полном объеме: окружение, события, постановка кадра.

Кроме реализации интеграции повествования через визуализацию и текст, актуальна генерация повествовательных и геймплейных механик.

На основе документации необходимо собирать статистику, которая поможет в дальнейшем в разработке: количество персонажей, локаций, реплик, внутриигровых событий, элементов визуализации и т.д..

Кроме этого планируется разработать метрики оценки эффективности инструмента и провести тестирование инструмента с участием разработчиков игрового повествования.

Заключение

В рамках работы были проанализированы текущие инструменты визуализации текста, а также профессиональный софт игровых сценаристов.

Были сформулированы требования к инструменту генерации сценарного прототипа видеоигр.

В рамках работы разработано экспериментальное решение, которое моделирует работу генератора: на этапе анализа текста извлекаются данные о персонажах и их действиях, а на этапе визуализации данные передаются в инструмент, который имеет в своем составе похожий функционал. Этот инструмент – Storyboarder.

На основании проведенного эксперимента было сделано заключение, что у инструмента есть перспективы значительно сократить время на предварительном этапе разработке игры.

В заключении представлены планы дальнейшей разработки инструмента генерации сценарного прототипа в виде авторского приложения.

Во время создания «Виртуального полигона осмотра места происшествия» (см. дис. [42] Гл. 4.2 и [41]) автор участвовал в командной разработке в роли сценариста. В процессе создания сценария были применены описанные в работе подходы к разработке и прототипированию повествования игр. По результатам было оформлено Свидетельство о государственной регистрации программы для ЭВМ [43].

К публикации принято две статьи. Одна их них [44] будет представлена на Всероссийской научной конференции “Научный сервис в сети Интернет”, а вторая [45] в рамках конференции Kazan Digital Week – обе будут опубликованы позже с рейтингованием в Scopus/Wos.

Демонстрацию функционала разработанного инструмента можно посмотреть на Youtube [46]. Код доступен в репозитории Высшей школы ИТИС КФУ [47].

Список использованных источников

1. Koenitz H., Roth C., Dubbelman T. Creating and Sharing Interactive Narrative Design Knowledge – A Multipronged Approach [Текст] // ICIDS 2018: Interactive Storytelling. – 2018. – С. 165-170.
2. Davies D., Bathurst D., Bathurst R. The Telling Image: The Changing Balance between Pictures and Words in a Technological Age [Текст] // Technology and Culture. – The Johns Hopkins University Press and the Society for the History of Technology Stable. – 1992. – V. 33. – No 4. – P. 845–846.
3. Raja D., Bowman D., Lucas J., North C. Exploring the benefits of immersion in abstract information visualization [Текст] // Proc. of the 8th Int'l Immersive Projection Technology Workshop. – 2004.
4. Cairns P., Cox A., Nordin I. Immersion in Digital Games: Review of Gaming Experience Research [Текст] // Handbook of Digital Games. – Wiley-IEEE Press. – 2014. – P. 337–361.
5. Schell J. The Art of Game Design: A Deck of Lenses, Second Edition. [Текст] // CRC Press. – 2014. – 594 p.
6. Koenitz H., Dubbelman T., Knoller N., Roth C., Haahr M., Sezen D., Sezen T. Card-Based Methods in Interactive Narrative Prototyping [Текст] // ICIDS 2018, Proceedings International Conference on Interactive Digital Storytelling. – Springer Nature Switzerland AG. – 2018. – P. 552–555.
7. Antonisse J. Paper Tales: A Guide to Narrative Prototyping [Video] // Game Developer Conference. – 2014. – Режим доступа: https://www.gdcvault.com/play/1020509/Paper-Tales-A-Guide-to (дата обращение 5.05.20)
8. Koivisto E., Suomela R. Using prototypes in early pervasive game development [Текст] // Sandbox '07: Proc. of the 2007 ACM SIGGRAPH

symposium on Video games. – Association for Computing Machinery. – 2007. – P. 149–156.

9. Al-Husseini Kh., Obaid A. Usage of prototyping in software testing [Текст] // Multi-Knowledge Electronic Comprehensive Journal For Education And Science Publications. – 2018. – V. 14. – P. 1–15.
10. Hassani K., Lee W.-S. Visualizing Natural Language Descriptions: A Survey [Текст] // ACM Computing Surveys – Association for Computing Machinery – 2016. – V. 49. – No 1.
11. Twine [Электронный ресурс]. – Режим доступа: https://twinery.org/ (дата обращения 12.05.2020).
12. RenPy [Электронный ресурс]. – Режим доступа: https://www.renpy.org/ (дата обращения 12.05.2020).
13. Lee N., Madej K. Disney Stories: Getting to Digital [Текст] // Springer. – 2012. – C. 114.
14. Articy:draft [Электронный ресурс]. – Режим доступа: https://www.articy.com/en/ (дата обращения 12.05.2020).
15. Cradle - play Twine stories in Unity [Электронный ресурс]. – Режим доступа: https://forum.unity.com/threads/released-cradle-play-twine-stories-in-unity.333720/ (дата обращения 12.05.2020).
16. RivetAI [Электронный ресурс]. – Режим доступа: https://www.rivetai.com/ (дата обращения 12.05.2020).
17. Gupta T., Schwenk D., Farhadi A., Hoiem D., Kembhavi A. Imagine This! Scripts to Compositions to Videos [Текст] // Cornell University. – 2018.
18. Joshi D., Wang J. Z., Li J. The story picturing engine – a system for automatic text illustration [Текст] // ACM Transactions on Multimedia Computing, Communications, and Applications. – 2006. – C. 68–89.

19. Zhu X., Goldberg A. B., Eldawy M., Dyer C. R., Strock B. A text-topicture synthesis system for augmenting communication. [Текст] // AAAI Press. – 2007.
20. Wonder Unit – Режим доступа: https://wonderunit.com/storyboarder/ (дата обращения: 12.05.2020).
21. Liu Z.-Q., Leung K.-M. Script visualization (ScriptViz): A smart system that makes writing fun [Текст] // Soft Computing. – Springer Nature Switzerland AG. – 2006. – С. 34–40.
22. Akser M., Bridges B., Campo G., Cheddad A., Curran K., Fitzpatrick L., Hamilton L., Harding J., Leath T., Lunney T., Lyons F., Ma M., Macrae J., Maguire T., McCaughey A., McClory E., McCollum V., Mc Kevitt P., Melvin A., Moore P., Mulholland E., Muñoz K., O'Hanlon G., Roman L. SceneMaker: Creative technology for digital storytelling [Текст] // Springer Nature Switzerland AG. – 2016. – С. 29–38.
23. S. Sekine. Corpus-based parsing and sublanguage studies [Текст] // Department of Computer Science. – 1998.
24. Ma M. Automatic Conversion of Natural Language to 3D Animation. [Текст] // University of Ulster. – 2006.
25. Valitutti R. WordNet-Affect: An affective extension of wordnet. [Текст] // 4th International Conference on Language Resources and Evaluation. – 2004.
26. Padia K., Bandara K., Healey C. A system for generating storyline visualizations using hierarchical task network planning [Текст] // Computers & Graphics. – Elsevier. – 2019. – С. 64–75.
27. Orange 3 [Электронный ресурс]. – Режим доступа: https://orange.biolab.si/ (дата обращения: 12.05.2020).
28. Standard Patterns in Choice-Based Games [Электронный ресурс]. – Режим доступа:

https://heterogenoustasks.wordpress.com/2015/01/26/standard-patterns-in-choice-based-games/ (дата обращения: 12.05.2020).

29. Small-Scale Structures in CYOA [Электронный ресурс]. – Режим доступа: https://emshort.blog/2016/11/05/small-scale-structures-in-cyoa/ (дата обращения: 12.05.2020).
30. Adams E., Joris D. The Designer's Notebook: Machinations, A New Way to Design Game Mechanics [Электронный ресурс]. – Режим доступа: https://www.gamasutra.com/view/feature/176033/the_designers_notebook (дата обращения: 12.05.2020).
31. Ильяхов М., Сарычева Л. Пиши, сокращай. [Текст] // Альпина Паблишер. – 2019. – С. 440.
32. Сахибгареева Г.Ф., Кугуракова В.В. Концепт инструмента автоматического создания сценарного прототипа компьютерной игры [Текст] // Электронные библиотеки. – 2018. – Т. 21. – № 3-4. – С. 235–249.
33. Доброквашина А.С., Газизова Э.А. Автоматизация проектирования игрового прототипа на основании обработки формализованного игрового дизайн-документа [Текст] // Ученые записки ИСГЗ. – Казань: Институт социальных и гуманитарных знаний. – 2019. – Т. 17. – № 1. – С. 583–589.
34. Астафьев А.А. Разработка инструмента для сборки сцен по тегам / Выпускная квалификационная работа на степень бакалавра [Текст] // Казанский федеральный университет. – 2020. – 42 с.
35. Нгуен А.З. Генерация окружения на основе текстового описания / Выпускная квалификационная работа на степень бакалавра [Текст] // Казанский федеральный университет. – 2020. – 31 с.
36. Николаев И.С., Митренина О.В., Ландо Т.М. Прикладная и компьютерная лингвистика [Текст] // M.URSS. – 2016. – 320 с.

37. Nothman J., Qin H., Yurchak R. Stop Word Lists in Free Open-source Software Packages. [Текст] // Proc. Workshop for NLP Open Source Software. – Association for Computational Linguistics. – 2018. – P. 712.
38. Data.Gov [Электронный ресурс]. – Режим доступа: https://catalog.data.gov/dataset/baby-names-from-social-security-card-applications-national-level-data (дата обращения: 12.05.2020).
39. McCreery, C. First-year Statistics for Psychology Students Through Worked Examples. 1. Probability and Bayes' Theorem [Текст] // Oxford Forum. – 2018. – С. 29.
40. Слепок коммита abbf5f24 открытого репозитория Wonderunit по Storyboarder [Электронный ресурс]. – Режим доступа: https://github.com/wonderunit/storyboarder/search?q=abbf5f24&type=Commits (дата обращения: 12.05.2020).
41. Антонов И.О. и др. Программирование запахов для виртуального осмотра места происшествия [Текст] / И.О. Антонов, К.В. Зезегова, В.В. Кугуракова, Е.Н. Лазарев, М.Р. Хафизов // Электронные библиотеки. – 2018. – Т. 21. – № 3-4. – С. 301-313.
42. Кугуракова В.В. Математическое и программное обеспечение многопользовательских тренажеров с погружением в иммерсивные виртуальные среды [Текст] / Диссертация на соискание ученой степени кандидата технических наук // Казанский федеральный университет. – 2019. – 187 с.
43. Программа для обучения следственным действиям по осмотру места происшествия в виртуальной реальности: Свидетельство о государственной регистрации программы для ЭВМ №2020613666 Российская Федерация / В.В. Кугуракова, М.Р. Хафизов, В.Д. Абрамов, Е.Н. Лазарев, Р.А. Шараева, Р.Р. Газизов, Г.Ф. Сахибгареева, И.О. Антонов; заявитель и правообладатель Фед. гос. автоном. образоват.

учреждение высш. образ. Казанский фед. ун-т. - №2020612601; заявл. 10.03.2020; зарегистрировано в реестре программ для ЭВМ 19.03.2020. - [1] с.

44. Сахибгареева Г.Ф., Кугуракова В.В., Бедрин О.А. Разработка решения для визуализации сценарного прототипа инструментами генерации раскадровок [Текст] // Всероссийская научная конференция “Научный сервис в сети Интернет”. – 2020. – 23 с. – статья принята к публикации.
45. Сахибгареева Г.Ф. Применимость разветвленных структур для генерации сценарных прототипов видеоигр [Текст] // Форум DigitalKazanWeek. – 2020. – 10 с. – статья принята к публикации.
46. Сахибгареева Г.Ф. Демонстрация работы инструмента [Электронный ресурс]. – Режим доступа: https://youtu.be/AUK0Nj19DNE (дата обращения: 22.06.2020).
47. Сахибгареева Г.Ф. Исходный код проекта [Электронный ресурс] // Репозиторий Высшей школы ИТИС КФУ. – 2020. – Режим доступа: http://gititis.kpfu.ru/GuFSahibgareeva/Demo_project (дата обращения: 22.06.2020).

Приложение

Приложение А. Фрагменты исходного кода

helper.py

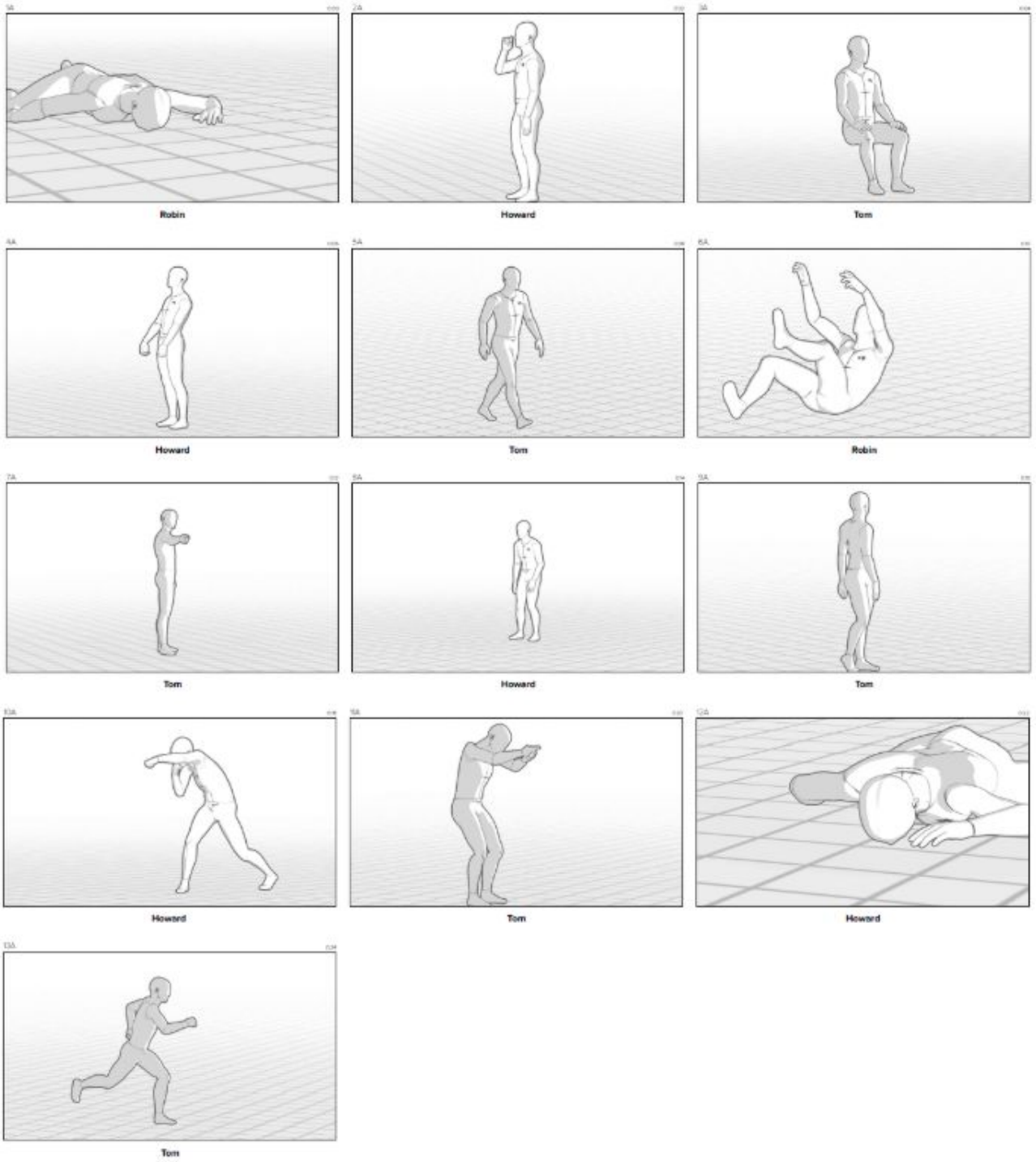
```
from sklearn.feature_extraction.text import
CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from sklearn import metrics

import pandas as pd
import numpy as np
import requests
import nltk

nltk.download('stopwords')
nltk.download('punkt')
port = 3000

with open('helper_initial_text.txt', 'r') as
initial_text:
    input_text = initial_text.read()

scene_configs = []
with open('scene_config.txt', 'r') as scene_config:
    for line in scene_config:
        scene_configs.append(line)

print('Input text loaded:\n', input_text)
print('----------------------------------------')

# Clear from stop-words
# stop_words = pd.read_csv('stop_words.csv',
index_col=0)['stop_words'].tolist()
stop_words = set(stopwords.words('english'))
print('Stop-words loaded:\n', stop_words)
print('----------------------------------------')
```

```
corpus = sent_tokenize(input_text)

for idx, sentence in zip(range(len(corpus)), corpus):
    sentence_tokens = word_tokenize(sentence)
    tokenized_sentence = [w for w in sentence_tokens if
not w.lower() in stop_words]
    tokenized_sentence = [w for w in tokenized_sentence
if w != '.']
    corpus[idx] = tokenized_sentence

print('Corpus created and cleared from stop-words and
punctuation:\n', corpus)
print('----------------------------------------')

# Set classifier and replace names by man or woman
names = pd.read_csv('NationalNames.csv', dtype =
{'Count': np.int32})
names = names.fillna(0)
print(names.head())
print('----------------------------------------')

namechart = names.groupby(['Name', 'Gender'], as_index =
False)['Count'].sum()
print(namechart.head(5))
print('----------------------------------------')

namechartdiff = namechart.reset_index().pivot('Name',
'Gender', 'Count')
namechartdiff = namechartdiff.fillna(0)
namechartdiff["Mpercent"] = ((namechartdiff["M"] -
namechartdiff["F"]) / (namechartdiff["M"] +
namechartdiff["F"]))
namechartdiff['gender'] =
np.where(namechartdiff['Mpercent'] > 0.001, 'male',
'female')
print(namechartdiff.head())
print('----------------------------------------')

char_vectorizer = CountVectorizer(analyzer='char',
ngram_range=(2, 2))
X = char_vectorizer.fit_transform(namechartdiff.index)
X = X.tocsc()
```

```
y = (namechartdiff.gender ==
'male').values.astype(np.int)

itrain, itest =
train_test_split(range(namechartdiff.shape[0]),
train_size=0.7)
mask = np.ones(namechartdiff.shape[0], dtype='int')
mask[itrain] = 1
mask[itest] = 0
mask = (mask == 1)
Xtrainthis = X[mask]
Ytrainthis = y[mask]
Xtestthis = X[~mask]
Ytestthis = y[~mask]

name_clf = MultinomialNB()
name_clf.fit(Xtrainthis, Ytrainthis)

training_accuracy = name_clf.score(Xtrainthis,
Ytrainthis)
test_accuracy = name_clf.score(Xtestthis, Ytestthis)
print('Name Classifier training accuracy:',
training_accuracy)
print('Name Classifier test accuracy:', test_accuracy)
print('----------------------------------------')

def translate_if_name(x):
    if not x[0].isupper():
        return x
    new = char_vectorizer.transform([x])
    y_pred = name_clf.predict(new)
    if (y_pred == 1):
        return 'man'
    else:
        return 'woman'

print('Pre-translation corpus:\n', corpus)

for sentence_idx, sentence in zip(range(len(corpus)),
corpus):
    for idx, word in zip(range(len(sentence)),
sentence):
        sentence[idx] = translate_if_name(word)
```

```
    corpus[sentence_idx] = ' '.join(sentence)

print('Translated without scene settings corpus:\n',
corpus)
print('----------------------------------------')

for idx, scene_config in zip(range(len(scene_configs)),
scene_configs):
    corpus[idx] += ', ' + scene_config.rstrip()

print('Translated corpus:\n', corpus)
print('----------------------------------------')

print('Sending to storyboarder...')

def send_sentences(sentences):
    for sentence in sentences:
        result = sentence.replace(' ', '%20')
        print(result)
        request = 'http://localhost:%s/text/%s' % (port,
result)
        print(request)
        r = requests.get(request)
        print(r.status_code)
        print(r.text)

send_sentences(corpus)
```

Приложение В. Пример сгенерированной раскадровки



Приложение С. Входные данные для генерации раскадровки

Оригинальный текст (фрагмент сценария «Виртуального полигона осмотра места происшествия» (см. дис. [42] Гл. 4.2 и [41])):

Зек садится на скамейку и пока Трус раскапывает могилу, рассказывает об ужасах, которые творились с ним в тюрьме, курит и оставляет вокруг скамейки множество окурков.

Трус раскапывает могилу, делает подкоп со стороны верхней части гроба. Трус поднимает гроб на попа. При этом он изрядно пачкается землей.

Пока Трус придерживает гроб, Зек сталкивает труп в яму. Трус опускает гроб.

Зек наводит на Труса пистолет. Пока Зек говорит о том, какой Трус подлец, Трус в удобный момент ударяет Труса лопатой по ноге. Зек падает.

Трус выбирается из могилы, пытается встать. В этот момент Зек оборачивается и выстреливает Трусу в ногу. Трус падает, ползет от могилы. Зек встает и стреляет Трусу в спину.

Зек ковыляет до окровавленной рубашки, принадлежащей Трусу. Подходит с ней к скамейке, садиться. Рвет рубашку и перевязывает ногу. Ковыляет к забору, чтобы сбежать.

Труп Труса лежит в нескольких шагах от могилы, на боку.

Сформированный на основе оригинального текста файл входных данных (helper_initial_text.txt)

Robin is dead.

Howard is smokes.

Tom is sitting on a chair.

Howard lifted.

Tom walks.

Robin is falling.

Tom gun point side.

Howard in despair.

Tom turn around.

Howard punches.

Tom gun tactical.

Howard is dead.

Tom is running.

Приложение D. Файл с настройками камеры

scene_config.txt

```
looking forward, long shot, single person, centered, left angle, eye level, long lens, light, frontrightlit, outside
looking forward, long shot, single person, centered, right angle, eye level, long lens, light, frontrightlit, outside
looking forward, long shot, single person, centered, left angle, eye level, long lens, light, frontrightlit, outside
looking forward, long shot, single person, centered, right angle, eye level, long lens, light, frontrightlit, outside
looking forward, long shot, single person, centered, left angle, eye level, long lens, light, frontrightlit, outside
looking forward, long shot, single person, centered, right angle, eye level, long lens, light, frontrightlit, outside
looking forward, long shot, single person, centered, left angle, eye level, long lens, light, frontrightlit, outside
looking forward, long shot, single person, centered, right angle, eye level, long lens, light, frontrightlit, outside
looking forward, long shot, single person, centered, left angle, eye level, long lens, light, frontrightlit, outside
looking forward, long shot, single person, centered, right angle, eye level, long lens, light, frontrightlit, outside
looking forward, long shot, single person, centered, left angle, eye level, long lens, light, frontrightlit, outside
looking forward, long shot, single person, centered, right angle, eye level, long lens, light, frontrightlit, outside
looking forward, long shot, single person, centered, left angle, eye level, long lens, light, frontrightlit, outside
```

Приложение Е. Список упомянутых видеоигр и сериалов

1. Hellblade: Senua's Sacrifice / Видеоигра // Ninja Theory. – 2017. – Режим доступа: https://store.steampowered.com/app/414340/Hellblade_Senuas_Sacrifice/ (дата обращения 12.04.2020).
2. Черное зеркало: Брандашмыг / Серия сериала // Netflix. – 2018. – Режим доступа: https://www.netflix.com/ru/title/80988062 (дата обращения 13.04.2020).
3. Тетрис / Видеоигра // Infogrames Entertainment, SA. – 1984. – Режим доступа: https://tetris.com/ (дата обращения 13.04.2020).
4. Halo / Серия видеоигр // Xbox Game Studios. – 2001. – Режим доступа: https://www.xbox.com/ru-RU/games/halo (дата обращения 13.04.2020).
5. Heavy Rain / Видеоигра // Quantic Dream. – 2010. – Режим доступа: https://www.quanticdream.com/ru/heavy-rain (дата обращения 13.04.2020).
6. Beyond: Two Souls / Видеоигра // Quantic Dream. – 2013. – Режим доступа: https://www.quanticdream.com/ru/beyond-two-souls (дата обращения 13.04.2020).
7. Detroit: Become Human / Видеоигра // Quantic Dream. – 2018. – Режим доступа: https://www.quanticdream.com/ru/detroit-become-human (дата обращения 13.04.2020).
8. Dragon Age: Origins / Видеоигра // Electronic Arts. – 2009. – Режим доступа: https://store.steampowered.com/app/17450/Dragon_Age_Origins/ (дата обращения 13.04.2020).
9. The Witcher 3 / Видеоигра // CD Projekt RED. – 2015. – Режим доступа: https://thewitcher.com/ru/witcher3/ (дата обращения 13.04.2020).

10. The Elder Scrolls V: Skyrim / Видеоигра // Bethesda Softworks. – 2011. – Режим доступа: https://elderscrolls.bethesda.net/ru/skyrim/ (дата обращения 13.04.2020).
11. Divinity: Original Sin / Видеоигра // Larian Studios. – 2014. – Режим доступа: http://www.divinityoriginalsin.com/ (дата обращения 13.04.2020).
12. The Sims / Серия видеоигр // Electronic Arts. – 1999. – Режим доступа: https://web.archive.org/web/20010206194247/http://thesims.ea.com/us/ дата обращения 13.04.2020).
13. The Stanley Parable / Видеоигра // Steam. – 2011. – Режим доступа: https://store.steampowered.com/app/221910/The_Stanley_Parable/?l=russian (дата обращения 13.04.2020).
14. Mass Effect / Серия видеоигр // Electronic Arts. – 2007. – Режим доступа: https://masseffectarchives.com/ru_RU/ (дата обращения 13.04.2020).