Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Направление: 09.04.04 Программная инженерия

Профиль: Разработка программно-информационных систем

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Автоматическая генерация игровых механик в соответствии с нарративом

Студент 2 курса

группы _____

«___»_____2019 г. Липатов В.В.

Научный руководитель

старший преподаватель

кафедры программной инженерии

«___»_____2019 г. Кугуракова В. В.

Директор Высшей школы ИТИС

«___»_____2019 г. Хасьянов А. Ф.

Казань – 2019 г.

Содержание

1 Введение **3**

2 Глоссарий **6**

3 Обзор литературы и существующих достижений **7**

3.1 Создание интерактивной драмы 7

3.2 Проблемы современного подхода к сюжету и механикам 10

3.3 Нарративно значимые механики в современных играх 12

3.4 Связь механик и нарратива 14

4 Концепция системы **16**

4.1 Определение ограничений системы 16

4.2 Объекты системы 18

4.3 Формализация 20

4.4 Визуализация формализованного сюжета 25

5 Реализация **27**

5.1 Структура классов 27

5.2 Алгоритмы 30

6 Методика работы с системой **32**

6.1 Подготовительная настройка системы 32

6.2 Развитие сюжета 37

6.3 Анализ динамики сюжета и геймплея 39

7 Пример использования системы **42**

7.1 Формализация сюжета 42

7.2 Настройка связей с механиками 45

7.3 Развитие по фазам 49

7.4 Результат 51

8 Анализ системы **59**

8.1 Эффективность системы 59

8.2 Ограничения системы 61

9 Заключение **62**

10 Список использованных источников **64**

11 Приложение **67**

11.1 Исходный код программы 67

11.2 Входные данные для эксперимента 73

1 Введение

Несмотря на необычайную скорость развития и роста популярности игр за последние пару десятилетий, теория игрового дизайна до сих пор находится в начальном состоянии. Не существует общепринятых сформированных теорий, как в более зрелых формах искусства: кинематограф, литература и других. Не существует даже общепринятого определения термина “игра”. Несмотря на это, игровая индустрия велика как никогда.

При отсутствии закрепленной и обоснованной теории, огромные игровые компании разрабатывают игры, основываясь на выработанные по опыту десятилетий принципы и подходы. Это позволяет создавать множество увлекательных игр, удовлетворяющих аудиторию, но не дает развития в понимании основ и потенциала игр. В то же время разработчики инди-игр стремятся к экспериментам и в своих поисках постоянно открывают миру новые свежие идеи. Открывают новые способы использовать ключевой аспект игр как формы искусства – интерактивность. Однако игры крайне сложны и дороги в своем создании, и зачастую свежие и рискованные идеи рушатся из-за недостатка ресурсов.

В научной же сфере не прекращаются исследования и попытки приблизиться к созданию эффективных систем и теорий, которые позволили бы создать интерактивную драму - историю в формате, в котором игрок является главным и активным участником, может предпринимать любые действия и видеть их последствия в развивающейся истории. Множество проблем было решено, множество систем предложено, но большая часть результатов не вышла за пределы исключительно научных публикаций и выступлений.

Причин существует множество: и хрупкость систем, и трудоемкость настройки генерации, и непостоянство принятых предположений и теорий. Это приводит к мысли о том, что пора попытаться найти источник проблем в самом

корне понимая игр как формы искусства; изменить точку зрения, с которой оцениваются и планируются игры. Например, игры заимствовали множество идей и теорий из кинематографа для создания собственных сюжетов и миров. Однако подходы неинтерактивной формы искусства в корне не учитывают важность и потенциал фактора интерактивности, который в играх обычно принимает форму геймплея. Это привело к тому, что большая часть игр в нынешние годы создаются как набор мало связанных частей – часть сюжетная и часть геймплейная. И системы генерации сюжета, предлагаемые научными статьями, не помогают в исправлении этой ситуации, так как не учитывают возможности и важность геймплея в игре.

Осознавая это, авторы научных статей всё чаще предпринимают попытки переосмыслить процесс создания и анализа игр, включая в основу понимание важности взаимосвязи геймплея и подачи истории в игре – нарратива. Следуя именно этому направлению, в данной статье будет произведена попытка формализовать связь между геймплеем и нарративом с помощью системы, определяющей набор сюжетных объектов, их состояний и правил, по которым игровые механики связываются с различными состояниями этих объектов.

Таким образом, актуальность работы в том, что в области разработки игр критически не хватает подходов к анализу геймплея с точки зрения его связи с нарративом, что, в частности, не позволяет создавать эффективные инструменты генерации самого геймплея и его связи с процедурно генерируемым контентом. Это ограничивает творческие возможности создателей игр в самом главном аспекте игр как искусства – интерактивности.

Объектом работы является исследование связи между сюжетом и игровыми механикам и исследование возможности передать часть смысла, эмоций и выводов истории через эти механики.

Предмет работы – система формализации сюжета и механик геймплея; алгоритм, конвертирующий сюжет в игровые механики.

Поэтому целью работы ставится описание системы формализации связи между сюжетом и геймплеем и реализация этой системы в виде библиотеки на языке C#.

Для этого должны быть выполнены следующие задачи:

- исследование нарративных возможностей игровых механик;
- исследование существующих систем генерации и формализации сюжета;
- формализация сюжета, адекватно отражающая динамику параметров сюжета, которые могут быть отражены с помощью игровых механик;
- разработка системы конвертации формализованного сюжета в игровые механики – в виде библиотеки классов;
- проверка эффективности системы на примере заданного сюжета.

2 Глоссарий

Геймдизайн – процесс создания формы и содержания игрового процесса (геймплея) разрабатываемой игры. Работа с геймдизайном может происходить как через соответствующий документ, так и существовать только в сознании разработчиков игры.

AAA-игры – неформальный термин, обозначающий класс высокобюджетных компьютерных игр. Термин «AAA-игра» используется в индустрии компьютерных игр для обозначения наиболее высокобюджетных компьютерных игр, рассчитанных на массовую аудиторию и требующих огромных затрат как на саму разработку игры, так и на её рекламу, в первую очередь, маркетинговую кампанию ко времени выхода.

Геймпад – тип игрового манипулятора. Представляет собой пульт, который удерживается двумя руками, для контроля его органов управления используются большие пальцы рук.

Нарратив – неформальный термин, обозначающий сюжет и атмосферу игру, переданные через инструменты, доступные игре – текст, звук, изображение, геймплей.

Инди-игра – компьютерная игра, созданная отдельным разработчиком или небольшим коллективом без финансовой поддержки издателя компьютерных игр. Масштаб явлений, связанных с инди-играми, ощутимо возрастает со второй половины 2000-х годов, в основном ввиду развития новых способов цифровой дистрибуции и средств разработки.

Интерактивная драма – история в формате, в котором читатель/зритель/игрок является главным и активным участником, может предпринимать любые действия и видеть их последствия в развивающейся истории.

3 Обзор литературы и существующих достижений

3.1 Создание интерактивной драмы

Еще с 90-х годов прошлого века публиковалось множество статей, нацеленных на создание теории интерактивной драмы [1], на развитие методов использования интерактивности в играх, чтобы позволить игроку быть активным участником истории и иметь возможность влиять на ее исход. Однако, спустя почти 30 лет всё еще практически не существует примеров коммерчески успешных игр, которые соответствуют всем критериям интерактивной драмы. В стандарты разработки пришли способы правдоподобно имитировать возможность и значимость выбора, но не создавать его или позволять ему значительно влиять на геймплей.

Это может показаться странным, ведь за десятки лет было создано множество систем и подходов, авторы которых заявляли об их работоспособности в реальных условиях разработки. Например, в [2] демонстрируется Suspenser – система генерации сюжета из фабулы, то есть из заранее заготовленного набора персонажей, локаций, действий и целей. Своей основной целью эта система ставит создание напряжение у читателя, полагая, что с уменьшением количества возможных положительных исходов (из конфликтной ситуации) также увеличивается и напряжение, испытываемое читателем. В [3] представляется IDtension – аналогичная система, которая при генерации сюжета исходит из целей и ценностей персонажей, и автоматически корректирует ход сюжета так, чтобы он ложился на стандартную схему завязки, развития, кульминации и развязки. В [4] разработали уже основанную на тексте систему, которая использует большой датасет написанных людьми историй.

Более того, существует пример полностью реализованной и выпущенной игры в жанре интерактивной драмы, создание которой описано в [5]. Игра

Facade была выпущена в 2005 году и была скептически принята аудиторией ввиду сомнительного развлекательного фактора, однако безусловно была отмечена как значительный прорыв в области интерактивной драмы.

В Façade рассказывается история семейной пары, находящейся на грани разрыва отношений. Игрок в роли друга семьи приглашается на обед и через общение с персонажами выясняет ситуацию и, возможно, помогает им решить им свои проблемы. Весь геймплей ограничивается перемещением по квартире, редким взаимодействием с предметами и, главное, общением с персонажами через текст. Игрок вводит слова и предложения, а игра пытается интерпретировать их так, чтобы персонажи могли корректно отреагировать на них.

Основная особенность игры в интерактивности и непредсказуемости сюжета. Каждое новое прохождение для каждого игрока должно отличаться от предыдущего, так как персонажи действуют в соответствии с действиями игрока, а также основываясь на случайном, но конкретном механизме последовательности сюжета. Так называемая система «битов», которая расставляет сюжетные арки в правильной последовательности (в соответствии с теориями развития сюжета из кинематографии), при этом корректно реагируя на действия игрока.

Хотя игра не произвела фурор и, к сожалению, даже последователей идеи, она была очень важна для индустрии и сферы игр в целом. Она показала возможность реализации интерактивной драмы, но помимо этого стала и олицетворением слабых сторон подхода, который пытается уместить в сфере игр теории из других сфер. Эти теории тяжело сочетаются с интерактивностью, скорее конфликтуют с ней, что выливается в замеченную игроками натянутость событий и рамок, в которую помещает игрока Façade.

Что же не позволяет вышеперечисленным достижения повлиять на современные игры? Во-первых, эти системы требуют нетривиальной настройки

и подготовки для полноценной работы, что не позволяет большинству разработчиков использовать ее на практике. Во-вторых, при всей своей сложности, результат, который способны создавать такие системы, нестабилен и непредсказуем, что недопустимо для коммерческих продуктов. И в-третьих, эти результаты сложно интерпретировать в контент и геймплей игры – и это косвенно указывает на корень проблемы.

3.2 Проблемы современного подхода к сюжету и механикам

Без надежных инструментов генерации сюжета разработчики вынуждены прибегать к имитации. Нарративная интерактивность в современных играх, как правило, воплощается небольшим набором строго определенных выборов, которые либо не имеют эффект на развитие истории вообще, либо изменяют содержание нескольких сцен или диалогов. То есть реализуется небольшое разветвление сюжета, в котором все ветки в итоге сливаются в общий поток. В качестве примеров сюжетно-ориентированных игр с таким подходом можно назвать серии игр The walking dead, Life is strange и Mass Effect.

Проблем в данной ситуации несколько. Во-первых, отсутствует повод для разработчиков развивать разработку игр в направлении полноценной интерактивности, так как проще и эффективнее, с точки зрения затрат, создавать лишь иллюзию выбора за счет ручного создания небольшого количества альтернативных частиц контента.

Во-вторых, в недостаточной степени используется основное выражение интерактивности игр – геймплей. Во всех приведенных играх игровые механики либо критически ограничены, либо существуют в отдельной от нарратива плоскости: то есть не влияют на него и не зависят от него. Это результат подхода, заимствованного из кинематографа, который, не имя интерактивности, не предоставляет платформы для вплетения игровых механик в процесс развития сюжета.

Эта проблема заимствования неподходящих методов из других форм искусства рассматривается также в [6]. В этой статье 2015-го года было приведено рассуждение на тему создания отдельной теории интерактивного виртуального нарратива. Автор вспоминает и разбирает существующие и нередко используемые в играх теории сценарного искусства из области кино, и отмечает неполноту этих теории при применении их к играм. Он отмечает, что

нарратив в играх невозможно рассматривать как отдельный элемент, написанный на бумаге и не связанный с механиками, кодом. Наоборот, игра является цельным элементом, и нарратив неразрывно связан с другими составляющими игры: интерактивностью, реализующими ее механиками; кодом, который реализует игру; и даже оборудованием, которое предоставляет возможность игроку участвовать в игре.

Эта статья обращает внимание на важность и механик, и сюжета, и указывает путь для развития теории: рассмотрения игры как сущность, в которой неотделимы нарратив и геймплей.

Чтобы убедиться в верности направления, заданном в этой статье, достаточно найти и проанализировать примеры игр, осознанно использующих геймплей и механики как часть нарративной системы игры.

3.3 Нарративно значимые механики в современных играх

Например, существуют игры, которые успешно рассказывают истории и дарят игрокам эмоции без использования текста и слов. В отличии от Facade, немалое количество таких игр являются финансово успешными (а значит тепло принятыми игроками) и высоко оцененными критиками. Это показывает, что путь, основывающийся на использовании механик и геймплея как основы нарратива в играх, является более верным подходом к развитию области.

Несколько таких игр проанализированы в [7]. Статья рассматривает 3 игры: Brothers:Tale of two sons, Bird story, Journey. В процессе анализа этих игр авторы попытались выделить основные приемы, которые помогли разработчикам передать историю без использования слов, и в результате на конкретных примерах указали, что игровые механики могут иметь значительный нарративный вес. Например, первая игра из списка – Brothers: tale of two sons, продемонстрировала эффективность приема связи конкретной игровой механики с персонажем через ассоциации и вырабатываемые игроком привычки: на протяжении всей игры правая часть геймпада отвечала за управление старшим из братьев, который, в частности, помогал неспособному плавать младшему брату переплыть через реку. И в заключительной части игры, когда старший брат уже погиб, а младший встретил препятствие в виде реки, которую необходимо переплыть, игра использовала ассоциации игрока с правой частью геймпада, чтобы создать аналогию тому, как младший брат вспоминает старшего брата и с помощью этого находит в себе силы переплыть реку. Этот прием способен привести к сильному эмоциональному отклику у игрока, так как он через погружение понимает чувства младшего брата.

Можно найти и другие примеры. Например, [8] анализирует приемы придания веса механикам игру Papers please, которая через осмысленное осложнение геймплея создает метафору невыносимости жизни людей в стране с

тоталитарным режимом. В работе [9] исследуется возможность передвижения персонажа в играх быть инструментом нарратива, касаясь также игры Journey.

3.4 Связь механик и нарратива

Итак, существует не только желание людей создать интерактивную драму в форме игры, но и определенные достижения в области создания интерактивной драмы. Полноценная функциональность интерактивной драмы невозможна без процедурной генерации/упорядочивания событий в игре, то есть без генерации сюжета. Однако предложенные системы оказались неэффективными на практике, и привели к распространенности подходов по имитации нарративной интерактивности. В то же время всё больше исследователей обращает внимание на необходимости пересмотреть подход к созданию сюжета в играх, начав рассматривать геймплей как неотъемлемую от сюжета часть. А небольшие игры независимых разработчиков всё чаще на своих примерах показывают, что игровые механики могут иметь значительный нарративный вес и вызывать сильные эмоции у игроков. Значит, при генерации сюжета для интерактивных драм необходимо думать не только о самом сюжете, но и том, как он отразится на геймплее, и как геймплей сможет отразиться на сюжете

Эти рассуждения приводят к выводу, что необходимо в первую очередь разработать подходы к формализации связи между сюжетом и игровыми механиками. Вариации формализации сюжета представлены в многочисленных статьях на тему генерации историй, но для механик мало примеров реализованных подходов к формализации механик со всем их разнообразием. Статья [10] представляет фундамент системы генерации механик без привязки к конкретному жанру. Система использует заданные разработчиком правила в виде логических выражений, чтобы сгенерировать подходящие вариации механик. Статья [11] демонстрирует прототип игры с процедурной генерацией оружия боевых космических кораблей на основе ее эффективности в бою. В первой статьей формализация механик крайне обобщена, а во второй наоборот

– очень конкретна и ориентирована лишь на один тип механики. И главное, отсутствует связь механик с нарративом игры.

Поэтому в данной статье будет предложена экспериментальная система формализации сюжета и игровых механик с целью доказать, что интеграция систем генерации сюжета и механик способна привести к развитию способности игр рассказывать истории и передавать эмоции.

4 Концепция системы

Сюжет влияет на состояния персонажей и мира, в то время как геймплей является прямым инструментом выражения состояний персонажей и мира. Фундаментальное утверждение *Системы* состоит в том, что значительные изменения состояния объекта сюжета должно быть выражено в измененных механиках этого объекта в игре.

Под состоянием при этом можно понимать почти любую сущность: эмоции, физические возможности, отношения, успешность целей и так далее. А объектами могут выступать не только персонажи, но также и окружение, общество, артефакты: главное условие лишь в том, чтобы объект был частью сюжета и имел свое явное представление в геймплее.

Таким образом, цель *Системы* – отслеживать изменения состояний объектов сюжета и формализовать их в виде, который можно конвертировать в соответствующие изменения в геймплее. Чтобы конкретизировать систему и сузить ее охват, определяются ее основные ограничения.

4.1 Определение ограничений системы

В основании процесса разработки *Системы* лежит предположение, что core-механики игры, как правило определяемые жанром, не могут быть и не должны быть полностью продиктованы сюжетом – точно так же, как сюжет не может быть полностью продиктован механиками. Ключевые и фундаментальные решения как о геймплее, так и о сюжете игры должны приниматься дизайнером в самом начале разработки, и использование каких-либо алгоритмов или систем генерации на этом этапе лишь вредит креативной целостности игры.

Однако геймплей и сюжет игры определенно должны влиять друг на друга в достаточной степени для того, чтобы связь между ними была заметна.

Таким образом, в качестве главного ограничения было принято то, что *Система* не определяет core-механики с нуля. *Системы* влияет на геймплей через дополнительные механики, изменение характеристик и модификации core-механик. Это позволяет сохранять задуманную целостность игры, но при этом подстраивать ощущения игрока под нужный лад.

Со стороны нарратива игры, *Система* также в определенной степени ограничивает свой обзор. Во-первых, рассматриваются только игры, имеющие явного и единого протагониста: это типичный формат для игр, и он лучше всего подходит для целей данной работы, так как в наибольшей степени использует эффект ассоциации себя с протагонистом истории. Помимо этого, рассматриваются только события, прямо или косвенно влияющие либо на состояние протагониста, либо на состояние обозреваемых им объектов сюжета.

Наконец, *Система* нацелена на рассмотрение только цельных сюжетов, имеющих в своём основании главную цель протагониста. Предполагается, что история игры стремится описать процесс эволюции какого-либо аспекта жизни протагониста, и что все события сюжета так или иначе влияют на этот аспект. Данное ограничение лишь косвенно сужает применимость *Системы* , так как даже в случае множества главных целей сюжета – его в большинстве случаев можно разбить на последовательные этапы смены целей.

4.2 Объекты системы

Как уже упоминалось, со стороны геймплея *Система* использует дополнительные механики, изменение характеристик и модификации core-механик как свои инструменты. При этом эти механики и характеристики могут относиться как к игровому персонажу, так и к окружению и противникам. Эти объекты *Системы* являются крайне обобщенными и не привязанными к конкретному жанру: подобная нечеткость в определении *Системы* создана намеренно, чтобы минимально ограничивать творческую свободу дизайнера.

Сюжет, в свою очередь, как источник данных *Системы* определяется более детально. В первую очередь *Система* отслеживает изменения состояний различных аспектов истории. За эти аспекты принимаются три типа сущностей сюжета: **главная цель** протагониста, **сюжетные линии**, а также то, что в контексте данной работы будет называться “**гранями**”.

Грани – это отдельные стороны эмоционального или физического состояния протагониста. В простейшем случае гранями являются такие простые эмоции как ярость и грусть, в других случаях в качестве граней можно принять отношение протагониста к другим персонажам. Грани самым прямым и очевидным образом влияют на возможности и слабости персонажа, которые можно отобразить в игре через механики.

Под **сюжетными линиями** понимается любая логически связанная последовательность событий сюжета, в которых участвует протагонист. При этом каждая сюжетная линия должна иметь набор явных параметров, отделяющих одну сюжетную линию от другой. Например, определяющими параметрами одной сюжетной линии может быть локация и набор неигровых персонажей, другой – встреча с боссом, и так далее.

Одним из параметров сюжетной линии также должна являться грань. Эта связь указывает на то, что каждая сюжетная линия должна быть важна для протагониста, и что развитие сюжетной линии влияет на одно из его состояний.

Главная цель протагониста – явное или скрытое стремление игрового персонажа, стоящее в центре сюжета. Это означает, что любое событие сюжета в той или иной степени влияет на развитие главной цели, либо на ее провал. А значительное изменение в степени успешности главной цели максимально сильно влияет на протагониста, кардинально меняя его видение и/или возможности. Главную цель одновременно можно классифицировать и как сюжетную линию, и как грань, так как она является воплощением одновременно и внешнего мира, и внутреннего мира протагониста.

4.3 Формализация

Определив объекты *Системы*, появляется возможность описать способ их формализации. *Система* ставит своей целью отражение сюжета на игровые механики, а значит необходимо формализовать сюжет, состояния сюжета, изменения состояний и, наконец, сами механики.

Под формализацией понимается способ цельным и осмысленным образом описать целевой объект так, чтобы его можно было представить в виде отдельных объектов, их групп и простых связей. Подобное представление неизбежно упрощает объект описания и приводит к потере части информации, но при этом позволяет запрограммировать целевой объект и его использование.

Формализация сюжета. Сюжет представляется в виде набора событий, каждое из которых имеет уникальные “координаты” в сюжете. Этими координатами по сути являются время и тематики сюжета.

Тематика сюжета отражается в виде сюжетных линий, а время – в виде фаз. **Фаза** – это непрерывный отрезок времени, в течение которого сюжет статичен. Сюжет и состояния объектов меняются только в промежутках между фазами: в результате событий, которые произошли в пределах этих фаз (см. рис. 1).

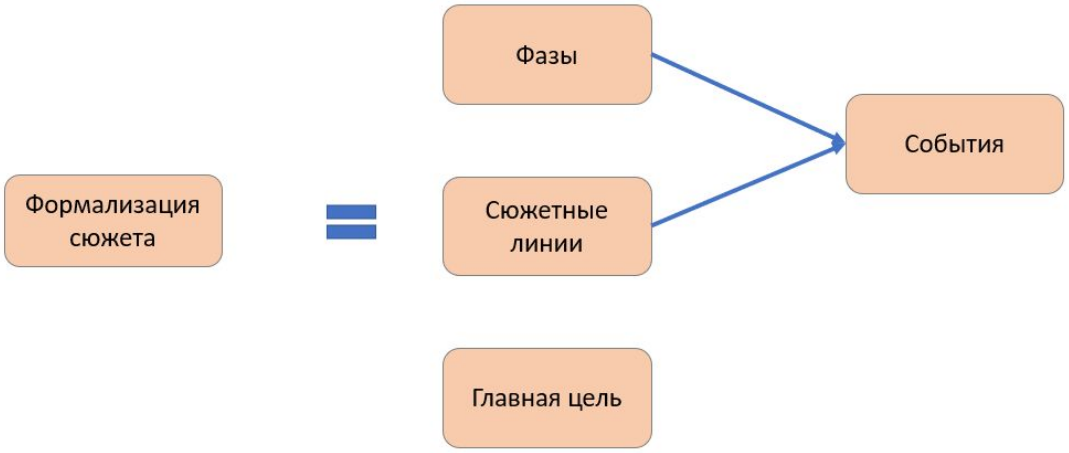

Рисунок 1 – Формализация сюжета

Комбинация конкретной фазы и конкретной сюжетной линии дает в результате “точку” сюжета. Именно в таких точках находятся все события сюжета. При этом не на каждой точке должно быть какое-либо событие, и на одной точке не может быть более одного события.

Формализация состояний. Под состоянием объектов сюжета принимается некая ограниченная мера, имеющая границы отрицательного и положительного значения. Для измеримой формализации состояние представляется в виде значения на отрезке [0, 100]. Соответственно, значение 0 – наиболее отрицательное значение, а 100 – наиболее положительное.

Однако само по себе численное значение имеет малый смысл – важно отслеживать значительные изменения в состояниях. Для этого вводятся три уровня состояния: положительное (значение больше или равно 75-ти), нейтральное (значение больше 25-ти и меньше 75-ти) и негативное (значение меньше или равно 25-ти). Таким образом, состояния могут меняться в числовых значениях, но для сюжета важным будут только переходы одного типа состояния в другое.

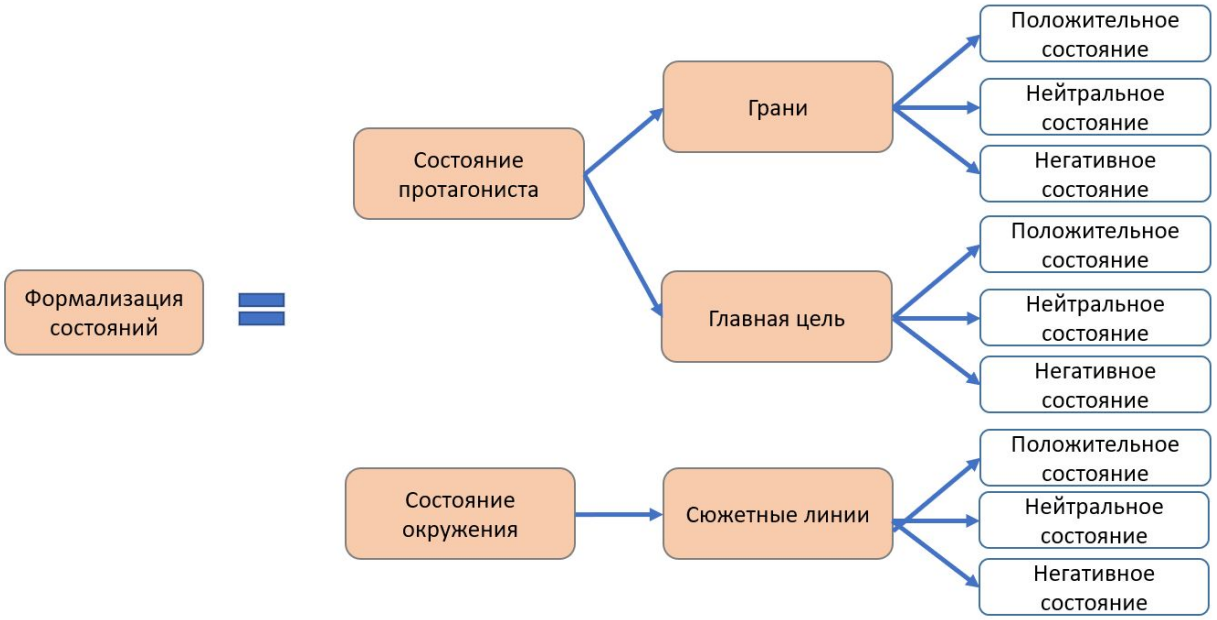

Рисунок 2 – Формализация состояний

Системе важно отслеживать состояния протагониста и окружения. Состояния протагониста выражается через грани и главную цель, а состояния окружения – через сюжетные линии. Таким образом, с точки зрения *Системы* грани, главная цель и сюжетные линии имеют состояния (см. рис. 2).

Формализация изменений. Наконец, каким образом в рамках *Системы* происходят изменения состояний, и на что влияют эти изменения.

Как было указано в главе “формализация сюжета”, каждое событие находится на конкретной точке сюжета, одна из координат которой – сюжетная линия. Таким образом, любое событие принадлежит какой-либо одной сюжетной линии.

Рисунок 3 – Формализация изменений

Смысл этой принадлежности в том, что событие напрямую изменяет состояние сюжетной линии своим **результатом**. С точки зрения *Системы*, каждое событие имеет измеряемый результат, который отображает степень успешности события с точки зрения протагониста. Этот результат может иметь

любое значение в отрезке [-100, 100], но предполагается, что для большинства событий значения находятся примерно на отрезке [-30, 30].

Результат события напрямую суммируется с состоянием сюжетной линии, которой принадлежит событие, что достаточно интуитивно. Однако объектами *Системы* с состояниями являются также грани и главная цель – и результаты события также влияют и на них (см. рис. 3).

Через сюжетную линию, каждое событие также связано с определенной гранью, на которую влияет сюжетная линия. То есть результат события суммируется и состоянию грани.

Наконец, из самого определения главной цели сюжета можно вывести, что любое событие также напрямую влияет на состояние главной цели.

Механики. Исходя из концепции *Системы*, изменения состояний объектов сюжета должны отражаться на объектах геймплея – механиках и характеристиках. То есть механики должны быть привязаны к значениям состояний. При этом на этапе формализации состояния было указано, что есть состояния трех типов: положительное, нейтральное и негативное. Поэтому в рамках *Системы* объекты геймплея привязываются к состояниям следующим образом:

- К положительному и негативному состояниям каждого объекта сюжета (грань, цель, сюжетная линия) привязывается по одной любой **активируемой механике**. Соответственно, одна из механик активируется только тогда, когда состояние является положительным, а вторая - только тогда, когда состояние является негативным.
- К числовому значению состояния привязывается одна **зависимая характеристика**. Характеристика меняет своё значение пропорционально изменению состояния объекта. При этом для

привязанной характеристики не имеет значения, является ли состояние в данный момент положительным, нейтральным или негативным.

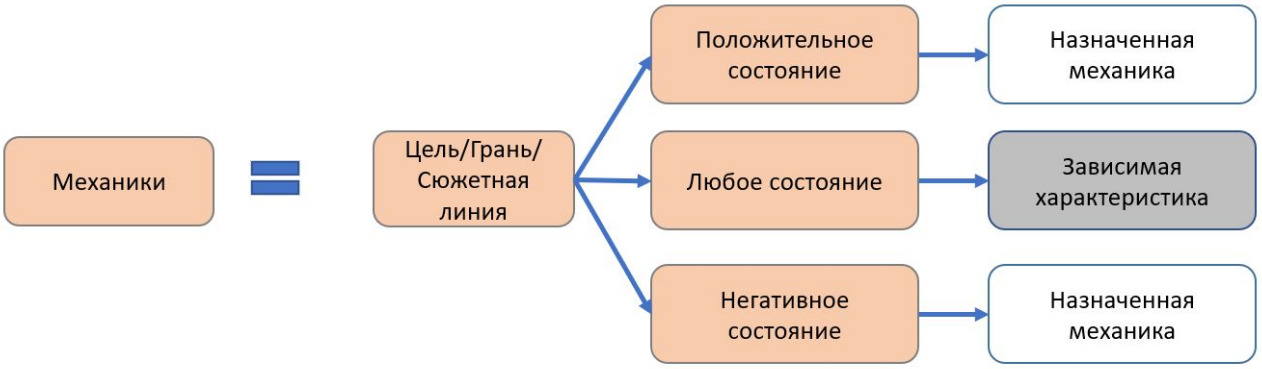

Рисунок 4 – Формализация механик

Таким образом, к главной цели, каждой грани и каждой сюжетной линии назначается по две механики и одной зависимой характеристики (см. рис. 4). Механики граней и главной цели, являясь отражением состояния протагониста, должны быть механиками, которые влияют на управление или эффективность действий игрового персонажа.

В то же время механики сюжетных линий должны отражать изменяемое состояние внешнего мира, а значит механики должны быть привязаны к поведению неигровых персонажей, к окружению или к другим внешним факторам, влияющих на процесс игры, но не находящимся в прямом контроле игрока.

Также важной особенностью механик, привязанных к сюжетным линиям, является то, что эти механики могут быть активны только в тех фазах, в которых одно из событий принадлежит этой сюжетной линии. Иначе говоря, механики сюжетной линии не должны быть активными, если эта сюжетная линия не имеет отношения к происходящему в игре в данный момент (в данную фазу).

4.4 Визуализация формализованного сюжета

Определив способ формализации всех объектов сюжета, можно не только использовать это как параметры для алгоритмов *Системы*, но также и визуализировать сюжет.

Например, предположим, что сюжет имеет четыре объекта сюжета: две сюжетные линии, одну грань и главную цель. Мы можем обозначить изменение состояния каждого объекта в виде графика, где ось X – время (фазы), а Y – значение состояния. Таким образом, визуализация простейшего сюжета из четырех объектов представляется в виде четырех графиков (см. рис. 5).

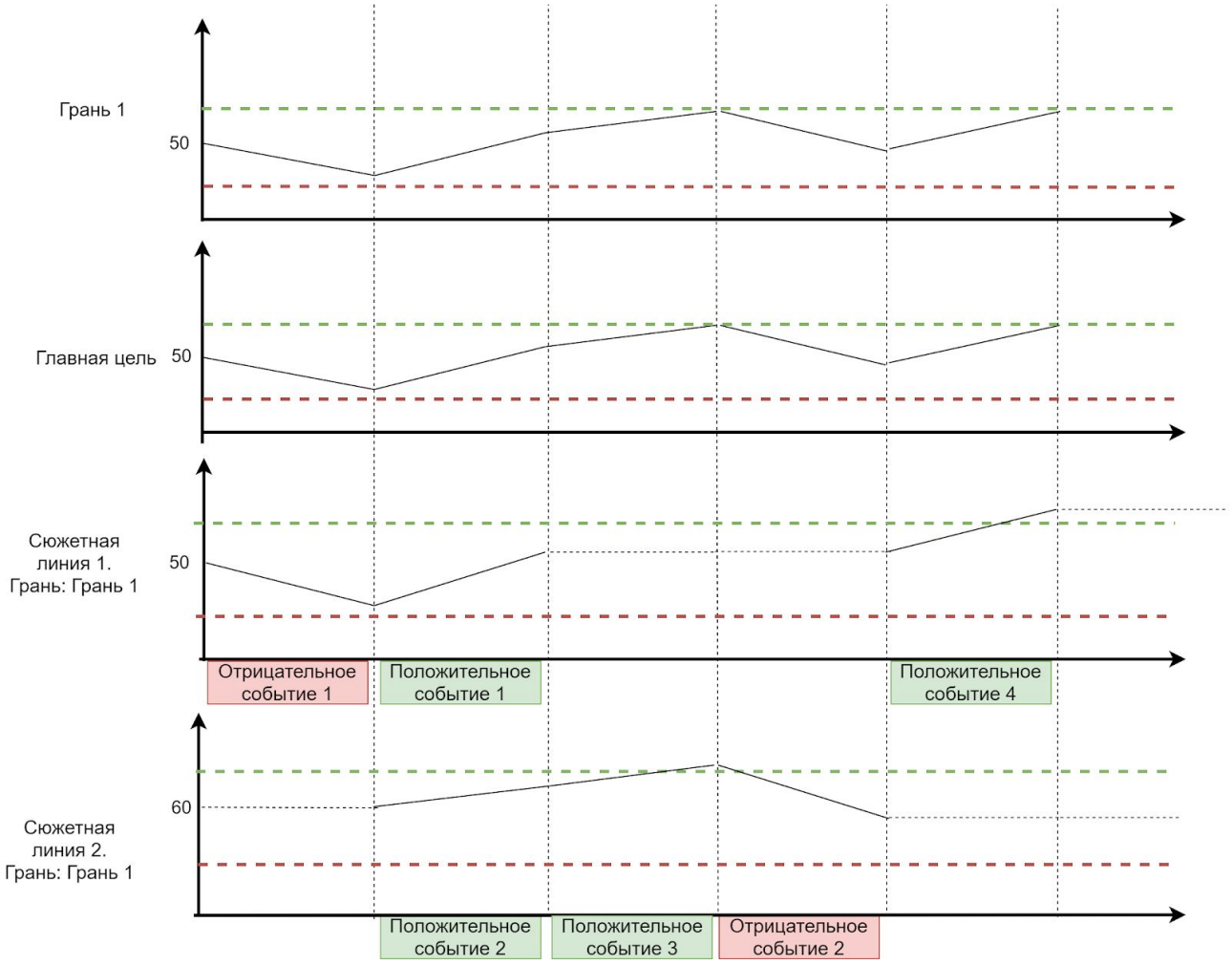

Рисунок 5 – Изменение состояния каждого объекта

Пороговые значения положительных и негативных состояний обозначены пороговым линиями сверху и снизу. События визуально отображены в виде блоков, также отличающимися друг от друга по цвету в зависимости от результата. Вертикальными пунктирными линиями обозначены фазы, а горизонтальными пунктирными линиями в графиках сюжетной линии обозначено неактивное состояние сюжетной линии во время конкретной фазы.

По визуализации можно явно видеть динамику сюжета, и проанализировать перегруженность той или иной сюжетной линии, или грани, или главной цели. Превышение порогов состояний также позволяет увидеть, в какие фазы активируются активные механики состояний, а само изменяющееся значение состояния показывает изменение зависимой характеристики.

В данном примере присутствует всего одна грань, и это приводит к идентичности графиков состояний грани и главной цели. Однако во всех практических случаях предполагается, что возможно выделить не менее двух граней, каждая из которых задействована хотя бы в одной сюжетной линии. Это приведет к различиям значений граней между собой, а также отличию от значений главной цели.

5 Реализация

5.1 Структура классов

Для проверки работоспособности *Системы* был реализован ее прототип на языке программирования C# без использования сторонних библиотек. Прототип представлен в виде библиотеки классов, состоящей из основного класс-менеджера, набора классов-контейнеров информации и интерфейсов для механик (см. рис. 6).

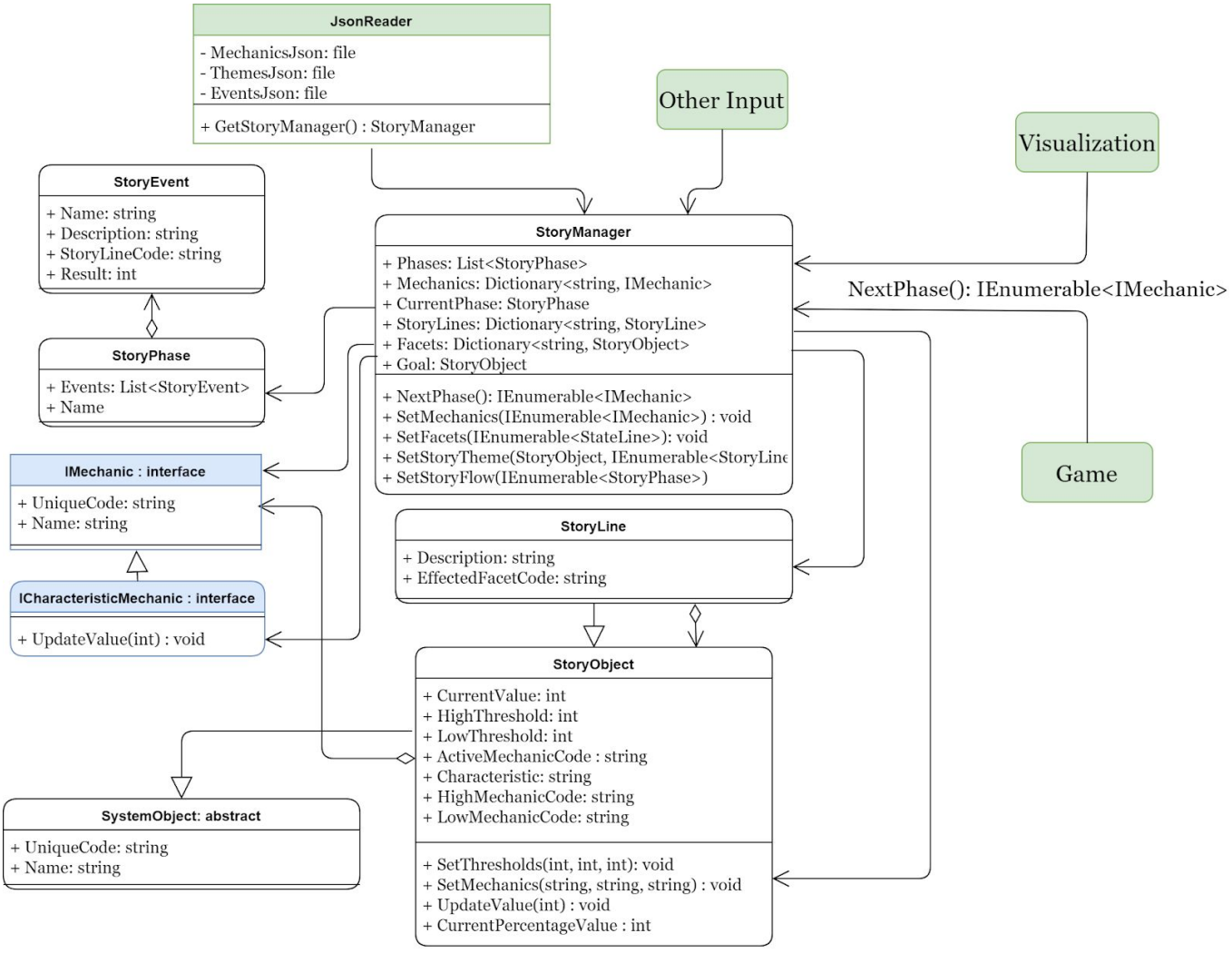

Рисунок 6 – Библиотеки классов прототипа системы

- StoryManager – класс-менеджер, использующийся также и для диалога со внешними системами. Дает доступ ко всем данным и предоставляет функции для настройки *Системы* и ее использования.
- NextPhase() – функция “движения” по сюжету, активирующая работу по изменению состояний объектов *Системы* и изменяющая состояния игровых механик.
- StoryObject – сюжетный объект *Системы* , обладающий своим состоянием и назначенными игровыми механиками. Используется для реализации *Граней* и *Главной цели* протагониста.
- UpdateValue(int) – функция, активирующая изменение состояния объекта, включение/отключение активной механики и изменение зависимой характеристики.
- StoryLine – наследник StoryObject, отличающийся лишь дополнительным набором свойств. Самое важное из которых – зависимый сюжетный объект, то есть *Грань* в контексте *Системы*. Соответственно, данный класс используется для реализации *Сюжетных линий*.
- StoryEvent – класс для реализации сюжетных событий в *Системе*. Ссылается на *Сюжетную линию*, которой принадлежит, и имеет свойство, в числовом виде обозначающее изменение состояния зависимых сюжетных объектов.
- StoryPhase – контейнер событий, происходящих в один измеримых промежуток времени. Используется для реализации *Фаз*.
- IMechanic и ICharacteristicMechanic – интерфейсы для механик, использующихся в *Системе*. Они диктуют лишь необходимость иметь уникальный идентификатор, а для механики изменяемой

характеристики – еще и функции для изменения значения характеристики.

Сама библиотека классов не реализует функционал получения данных напрямую от пользователя и не заточена под геймплей конкретной игры: для этого должны использовать внешние компоненты, примеры которых помечены на схеме классов зеленым цветом. В рамках данной работы был реализован тестовый модуль JsonReader, который в качестве входных данных использует данные в формате json. Эти данные обрабатываются модулем, конвертируются классы-контейнеры библиотеки классов системы, и подаются на вход классу-менеджеру StoryManager.

5.2 Алгоритмы

Процесс получения актуальных механик для каждой из фаз представлен на рисунке 7. В нем из всех активных сюжетных объектов (грани, главная цель, и сюжетные линии актуальных событий) выделяются модификация характеристики, а также негативная или положительная механика, если значение состояния объекта находится вне границ нейтрального значения.

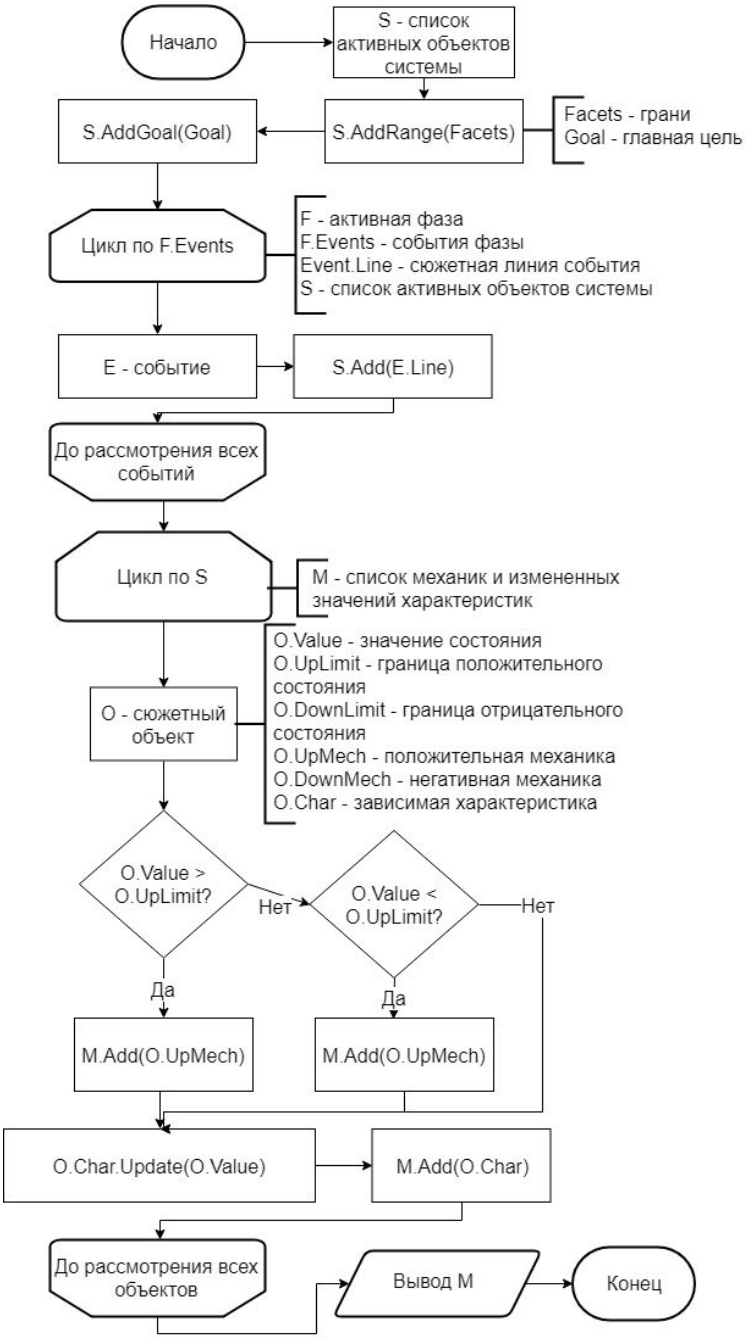

Рисунок 7 – Алгоритм получения активных механик

Алгоритм обновления состояний сюжетных объектов при изменении фазы показан на рисунке 8. В нем результаты всех событий новой фазы изменяют значения состояний всех связанных сюжетных объектов.

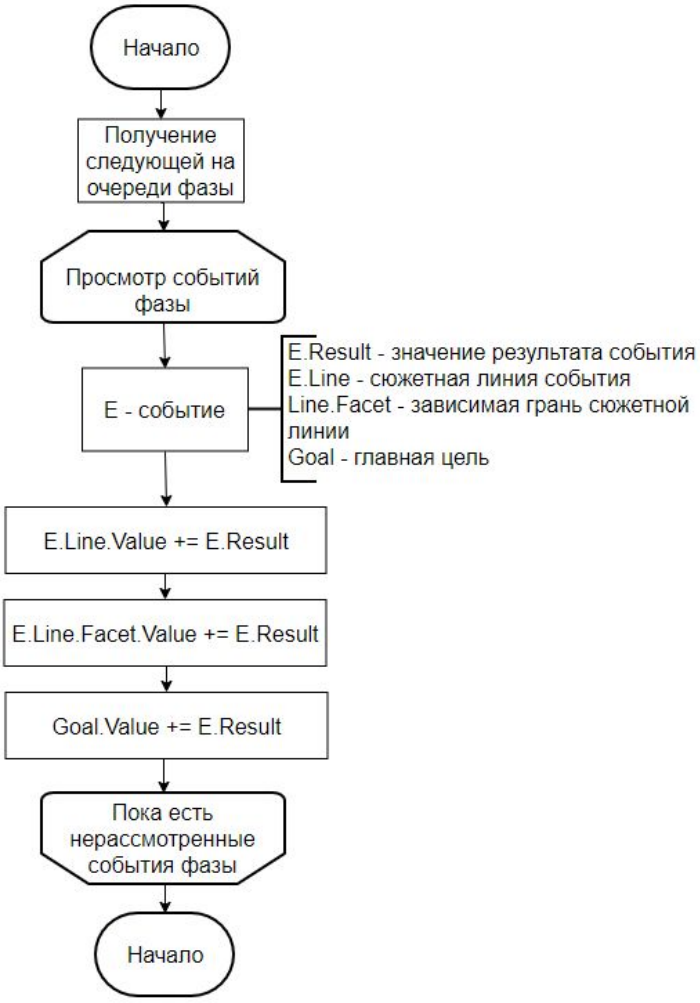

Рисунок 8 – Алгоритм обновления состояния сюжетных объектов

6 Методика работы с системой

Конечная цель *Системы* состоит в том, чтобы она могла быть легко интегрируемым инструментом генерации механик из заданного сюжета. То есть, чтобы *Система* была автономной в процесс работы. Однако для достижения этой цели необходимо правильно настроить *Систему* перед ее использованием. То есть для получения правильного результата создателям игры необходимо вручную настроить параметры.

6.1 Подготовительная настройка системы

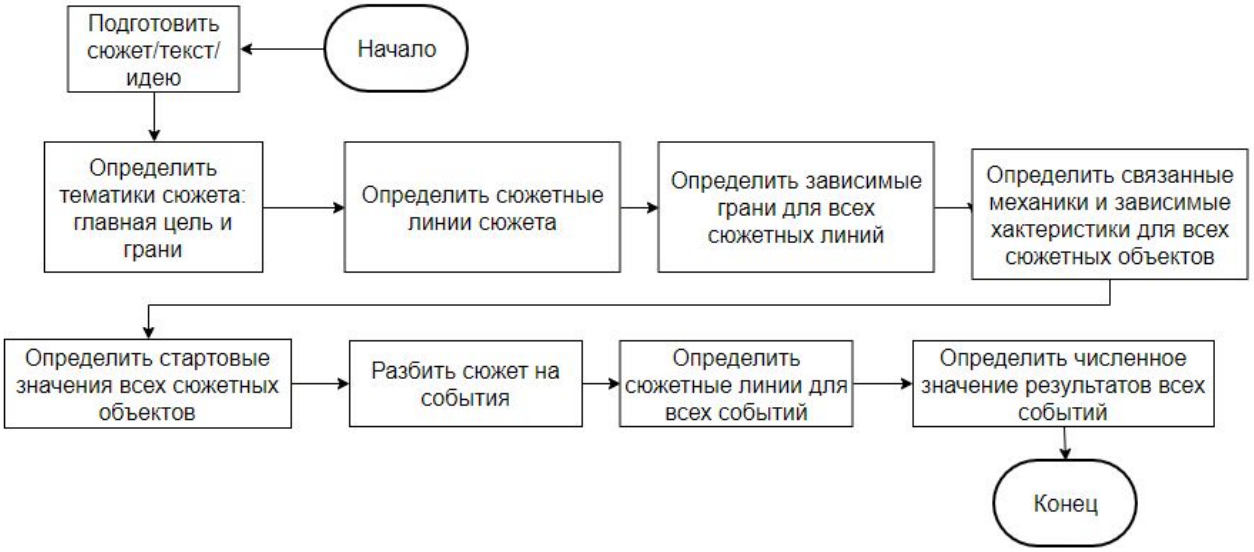

Рисунок 9 – Алгоритм подготовительной настройки *Системы*

Со стороны сюжета, *Система* требует заранее провести следующие шаги:

- определить список граней протагониста;
- определить список сюжетных линий и зависимых от них граней;
- определить главную цель протагониста в сюжете;
- определить начальный уровень состояний для всех вышеописанных объектов сюжета.

То есть требуется определить то, о чем будет сюжет, и какие ощущения он планирует дать игрок.

Далее, имея формализованный и введенный в систему сюжет, можно настраивать связанные механики согласно условиям, описанным в главе “Формализация”. То есть для главной цели, для каждой грани и каждой сюжетной линии задать по 2 активируемой механики и по одной зависимой характеристики.

В целом, нацеленность *Системы* на максимально возможную обобщенность и гибкость означает, что однозначных требований к тому, как подбирать механики – нет. И всё же сам принцип проектирования *Системы* основан на том, чтобы ее можно было использовать для получения выгоды от комбинации и динамики подобранных механик. Поэтому можно утверждать, что механики должны быть связаны с объектом сюжета, к которому они привязаны, через смысл и ощущения для игрока, и помимо прочего иметь синергию между собой.

Для наглядного объяснения можно привести пример. Пусть рассматриваемая игра – некая Action-RPG с основным игровым циклом в виде битв с противниками. Пусть объект сюжета – главная цель протагониста. Цель - “стать доблестным героем”. Как для любого объекта сюжета, система требует назначить цели три механики: 2 активируемых и одну постоянно зависимую характеристику. Какие можно подобрать механики для данной цели?

Во-первых, эти механики должны быть осмысленно близки к цели. Например, если протагонист приписывает самоотверженность к качествам героев, то в качестве положительной активируемой механики можно подобрать механику, которая при низком уровне здоровья активирует какой-либо положительный эффект, помогающий в бою.

Во-вторых, эти механики должны быть связаны с ощущениями, наиболее близко соответствующими различным состояниям *Цели*. В отличие от связи по

смыслу (в которой необходимо проанализировать взгляд протагониста на сюжетный объект), для связи по ощущениям необходимо рассматривать взгляд игрока на сюжетный объект. Данный процесс субъективен, зависит от целевой аудитории игры и от предполагаемого бэкграунда (заранее известной информации) игрока на момент рассматриваемого участка игры.

Например, если в рамках рассматриваемого примера предположить, что дизайнер системы уверен в существовании у игрока ассоциации: “герой” – “уверенность в своей неуязвимости”, то в качестве зависимой характеристики цели “стать героем” можно подобрать что-либо, что влияет на чувство безопасности игрока.

Наконец, в идеале связанные механики должны иметь синергию между собой. В рамках *Системы* под синергией механик можно рассматривать взаимодействие положительной активируемой механики с высоким состоянием зависимой характеристики, и наоборот – негативной активируемой механики с низким состоянием зависимой характеристики. То есть при положительном и негативном состояниях, эффекты механик должны дополнять друг друга и приводит к более заметным изменениям.

Рисунок 10 – Синергия механик

Ориентируясь на описанные параметры, можно подобрать следующий набор связанных механик (см. рис. 10):

- положительная активируемая механика: пассивный эффект *“Чем меньше здоровья у персонажа протагониста, тем больше урона он наносит”*;
- зависимая характеристика: *скорость регенерации здоровья*;
- негативная активируемая механика: пассивный эффект *“При нанесении урона, персонаж протагониста восстанавливает часть здоровье*”.

Анализируя их, можно заметить следующие соответствия рекомендациям:

- Положительная механика соответствует смыслу “самопожертвования” героя и побуждает игрока с большей готовностью получать урон в бою.
- Зависимая характеристика напрямую влияет на выживаемость персонажа, что в итоге влияет на ощущение безопасности игрока: низкое значение характеристики привносит опасение и страх, а высокое – уверенность и бесстрашие.
- Негативная механика соответствует жестокости и беспощадности, и побуждает игрока ориентироваться на нанесении вреда противникам.
- Положительная механика и высокое значение зависимой характеристики имеют синергию, которая совмещает в себе повод жертвовать своим здоровьем ради увеличение эффективности, и уверенность в том, что все потерянное здоровье быстро восстановится. Это побуждает игрока действовать самоотверженно, в соответствии с моделью поведения “героя”.
- Негативная механика и низкое значение характеристики имеют синергию, которая совмещает в себе страх за свою безопасность и желание навредить окружающим ради собственной выгоды. Это побуждает игрока

действовать осторожно и эгоистично, что полностью противоположно поведению “героя”.

Таким образом, на приведенном примере можно увидеть положительный эффект от подбора механик, которые связаны с сюжетным объектом по смыслу, ощущению и имеют между собой синергию.

6.2 Развитие сюжета

После назначения всех сюжетных объектов и связанных игровых механик, подготовка *Системы* является оконченной. Дальнейшие этапы – этапы использования *Системы*, которые дают некий результат при каждом взаимодействии с ней.

Этап развития сюжета – этап, на котором *Система* генерирует свои состояния и, соответственно, активные игровые механики для каждой из фаз.

Однако для получения таких результатов этап развития сюжета требует дополнительных действий:

- формализовать все события, определить их принадлежность фазам и сюжетным линиям;
- определить значение результата для каждого события в сюжете;
- подать формализованные события *Системе* для генерации состояния механик для каждой из фаз.

Примечание: *В идеале, эти работы должны проводиться вручную сценаристом, однако вполне разумно также рассмотреть способы автоматически формализовать текст сюжета в необходимый формат с помощью инструментов машинного обучения. К сожалению, это выходит за рамки данной работы, поэтому далее будет считаться, что тем или иным образом события сюжета формализованы для использования системой и подаются ей на вход постепенно или в виде завершенного сюжета.*

.

Наконец, при подаче *Системе* каждой новой фазы, можно получать информацию о соответствующем фазе состоянии игровых механик. *Система* просчитывает эти данные, исходя из предыдущего состояния сюжетных

объектов, изменений активных Сюжетных линий и изменений, которые привнесли события фазы на все объекты *Системы*.

Таким образом, этап развития сюжета можно использовать для автоматической генерации игровых механик для каждой новой фазы сюжета.

6.3 Анализ динамики сюжета и геймплея

Если у дизайнера *Системы* не стоит цели генерировать механики “в реальном времени”, и уже готов полноценный участок сюжета с набором фаз и событий, то с помощью визуализации сюжета есть возможность проанализировать введенные данные, обнаружить слабые и сильные стороны сюжета, неподходящий темп геймплея на разных фазах. Это позволяет исправить данные так, чтобы они удовлетворяли желаемому балансу.

В частности, *Система* позволяет обнаружить следующие слабые стороны игры:

Перегрузка сюжетного объекта. Если какое-либо событие сюжета приводит к попытке сюжетного объекта выйти за пределы минимального или максимального значения состояния – это означает либо слишком большое количество однотипных событий в сюжете, либо неправильную оценку числовых значений событий (см. рис. 11).

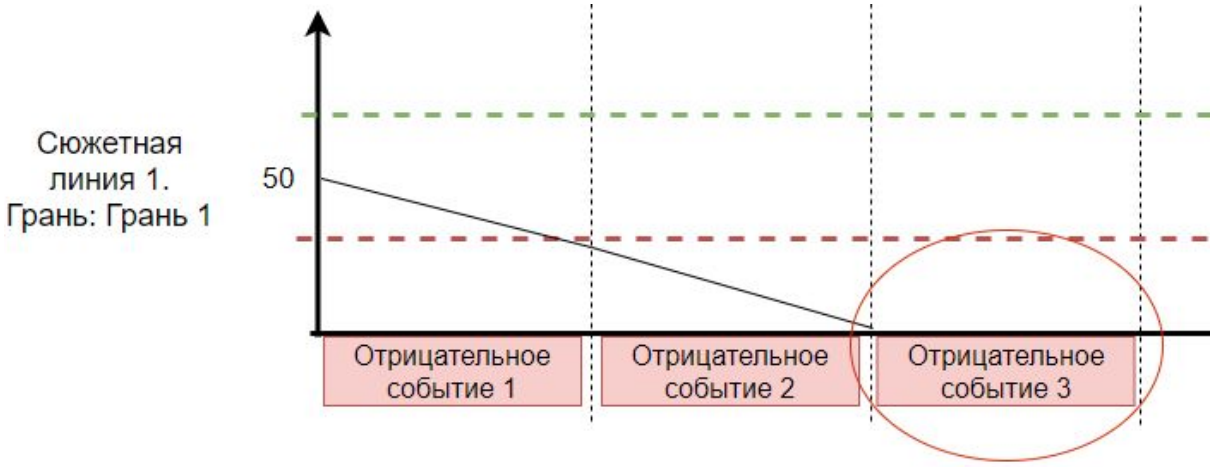

Рисунок 11 – Пример неправильной оценки числовых значений событий

Застой сюжетного объекта. Если сюжетный объект на протяжении нескольких близких фаз не меняет своего состояния, несмотря на наличие связанных событий, то значимость событий для игрока теряется, и сюжетный

объект теряет влияние на игру в глазах игрока. Подобная проблема может говорить о некорректной динамике событий (см. рис. 12).

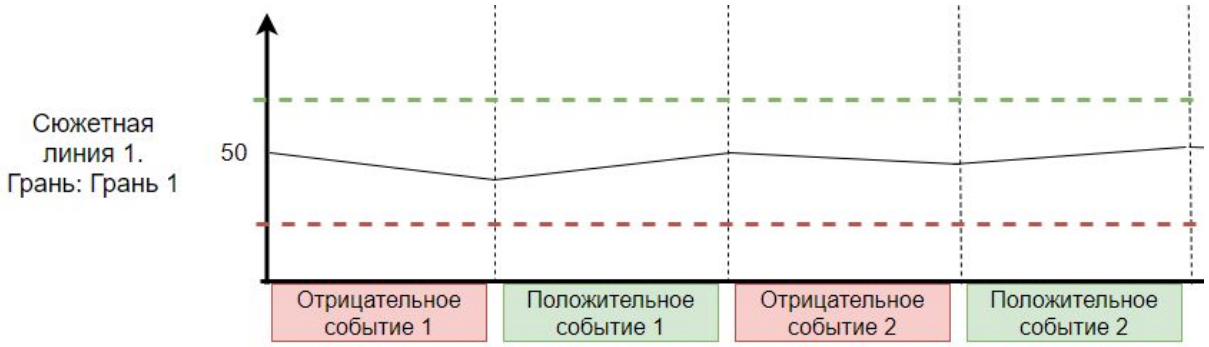

Рисунок 12 – Пример некорректной динамики событий

Слишком большая изменчивость сюжетного объекта. Противоположность предыдущей проблемы – слишком большая изменчивость приведет к путанице для игрока, и быстро приведет к тому, что игрок перестанет придавать большую значимость изменениям объекта и механикам, связанным с ним.

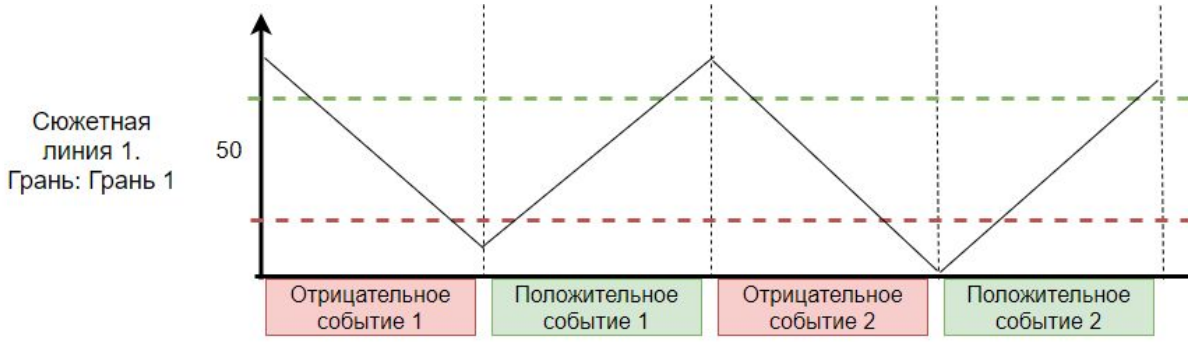

Рисунок 13 – Пример слишком большой изменчивости сюжета

Перегрузка механиками и эмоциями. Если одновременно множество сюжетных объектов находятся в положительном или отрицательном состоянии, то это может говорить о перегрузке фазы: слишком большое количество разнообразных механик тоже может запутать игрока и обесценить связь объектов с механиками. Стоит, правда, отметить, что в некоторых ситуациях

перегрузка может быть намеренной и соответствовать кульминационному этапу сюжета.

Недостаток фаз расслабления. Если в большинстве фаз у игрока находится слишком много сюжетных объектов в положительном или отрицательном состоянии, или если фазы слишком часто меняют состояния объектов, то это может привести к переутомлению игрока. В таких случаях может быть необходимо пересмотреть последовательность и набор событий и, возможно, разбавить динамичные моменты более спокойными и растянутыми.

Недостаток фаз напряжения. В противоположность предыдущей проблеме, *Система* может указать на малую динамичность сюжета, если большая часть фаз сюжета имеют недостаточно активных механик или не меняют состояния сюжетных объектов.

Список проблем, которые может помочь обнаружить *Система*, может быть продолжен и в целом может сильно отличаться в каждом конкретном случае. Но в любом случае стоит отметить, что анализ результатов *Системы* позволяет думать о балансе игры не как двух слабо связанных между собой сущностях: сюжете и геймплее, – а как о единой системе, где состояние сюжета и геймплея неразрывно связаны между собой.

7 Пример использования системы

Для проверки эффективности *Системы* был продуман с нуля сюжет, который после был формализован по заданным в *Системе* правилам. После настройки тем и сюжетных линий, были настроены связанные механики, а после – установлены значения изменений для каждого из событий сюжета.

Ниже будет продемонстрирована визуализация процесса и результата работы с *Системой* в контексте этого сюжета.

7.1 Формализация сюжета

В качестве примера была выбрана небольшая история, повествующая о рыцаре-изгнаннике, который решил стать героем назло всему миру. Рыцарь был безобразен внешне, из-за чего никто не видел его истинных стремлений к доблести и чести. Поэтому, чтобы доказать всем свое право зваться героем, он решил сразить дракона, похитившего принцессу.

Дальнейшую можно представить в виде последовательности формализованных событий, также отображенных на рисунке 14:

1. Первая битва с драконом, которая окончилась поражением рыцаря.
2. Вторая битва с драконом, в которой рыцарь одержал победу благодаря помощи, которую оказывала принцесса в ходе битвы.
3. Путь рыцаря к освобождению принцессы из замка, наполненного смертельными ловушками
4. Спасение принцессы из заключения в замке. Однако принцесса, мечтавшая о прекрасном принце, лишь показывает своё разочарование и с трудом скрываемое презрение к рыцарю из-за его безобразности.
5. Последняя битва с драконом, который напал на рыцаря на выходе из замка. Принцесса, воспользовавшись ситуацией, убежала, а

переполненный яростью ко всему миру рыцарь убил дракона в результате кульминационной битвы.

6. Возвращение рыцаря в родной город и неожиданное для рыцаря народное признание за то, что он сразил угрожающего всему городу дракона.

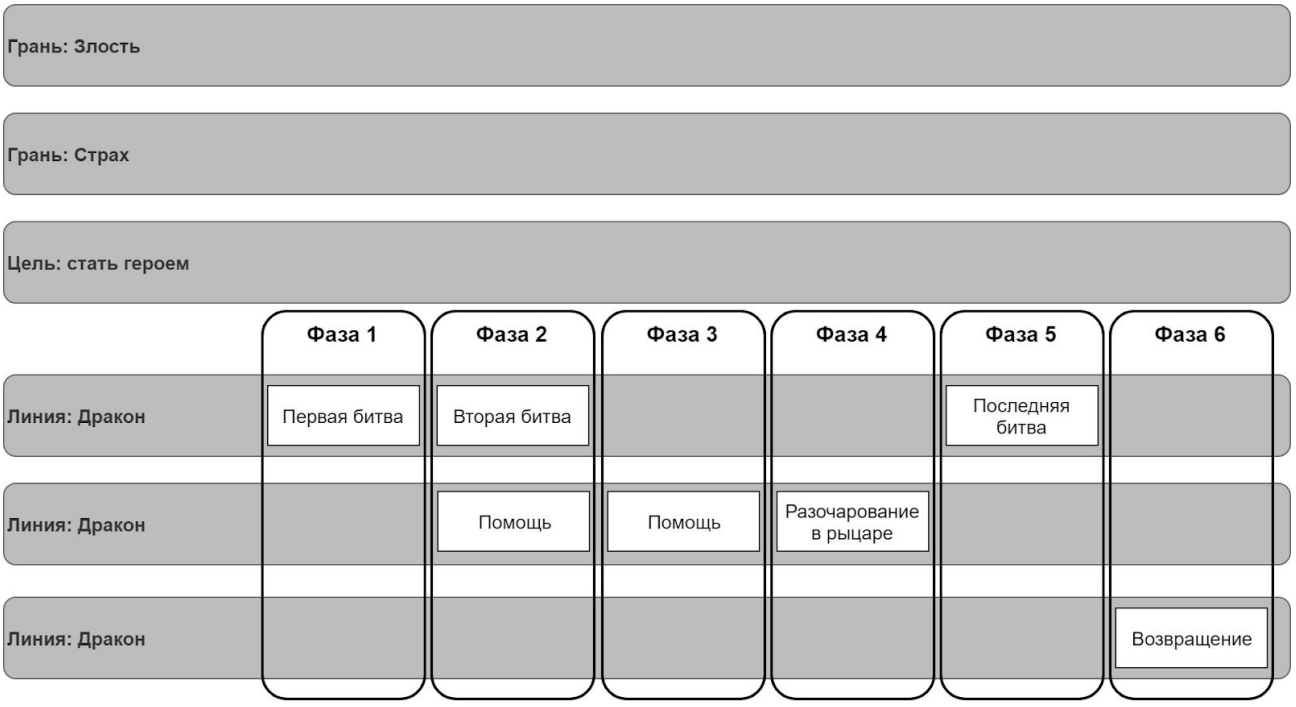

Рисунок 14 – Фазы сюжета

Согласно методике работы с *Системой*, первым шагом в формализации сюжета необходимо определить тематику и сюжетные линии.

Главной целью протагониста, что напрямую выводится из описания сюжета, можно выделить цель “Стать героем”.

В качестве **Граней** сюжета были выделены “Злость” и “Страх”. Злость отражает отношение рыцаря к внешнему миру, а страх его неуверенность в себе, которую он пытается побороть.

Сюжетные линии были разделены по признаку персонажей, участвующих в них персонажей:

- Сюжетная линия “Дракон”. Зависимая грань – страх, так как дракона можно воспринимать как выражение страха протагониста.

- Сюжетная линия “Принцесса”. Зависимая грань – злость, так как принцесса является важнейшей для рыцаря частичкой внешнего мира, который его ненавидит и который рыцарь хочет переубедить.
- Сюжетная линия “Город”. Зависимая грань – злость, так как город и есть тот самый внешний мир, на который обозлен рыцарь.

7.2 Настройка связей с механиками

Далее необходимо настроить связанные механики для каждого сюжетного объекта. В главе “подготовительная настройка системы” был разобран пример подбора механик для главной цели “Стать героем” – то есть, на самом деле, была разобрана главная цель сюжета, представленного в этой главе. Поэтому мы уже имеет набор механик, привязанных для Главной цели (см. рис. 15):

- Положительная механика: Чем меньше у протагониста здоровья, тем больше урон он наносит.
- Негативная механика: Восстановление здоровья протагониста при нанесении урона.
- Зависимая характеристика: скорость регенерации здоровья.

Рисунок 15 – Набор механик для конкретной цели

По аналогии с разобранным примером, были назначены связанные механики для всех оставшихся объектов: граней и сюжетных линий. Конкретные рассуждения для каждого отдельного объекта не так важны для эксперимента, поэтому ниже будет приведен список назначенных механик без подробного разбора причин выбора (см. рис. 16).

	Грань: Злость	Грань: Страх
Положительное состояние	Оставшиеся в конце хода очки действия переносятся на следующий ход	Способность: «Бесстрашный рывок». Перемещение, атака по площади.
Характеристика	Наносимый урон	Шанс промаха
Негативное состояние	Наносимый урон дает очки действия на следующий ход	Получение урона уменьшает очки действия на следующий ход

Рисунок 16 – Список всех назначенных механик

- Грань: Злость
  - Положительная механика: Оставшиеся неиспользованными в конце хода очки действия (ОД) переносятся на следующий ход.
  - Негативная механика: Наносимый урон дает больше ОД на следующий ход.
  - Зависимая характеристика: величина наносимого урона (обратная зависимость).
- Грань: Страх

- Положительная механика: активируемая способность, благодаря которой протагонист делает рывок вперед, нанося большой урон противникам на пути.
- Негативная механика: получение урона уменьшает количество ОД на следующем ходу.
- Зависимая характеристика: шанс промахнуться атаками.

Перед разбором механик сюжетных линий следует напомнить, что, в отличие от механик граней и главной цели, механики сюжетных линий влияют не напрямую на протагониста, а на окружение. Таким образом, игровые механики сюжетных линий – это способности и характеристики персонажей или окружения (см. рис. 17). Также эти механики активны только в фазах, в которых происходит какое-либо событие этой сюжетной линии.

	Линия: дракон Грань: страх	Линия: принцесса Грань: злость	Линия: родной город Грань: злость
Положительное	Способность: Полет (отступление)	Увеличивает количество очков действия протагониста	Шанс появления нового союзника в начале хода
Характеристика	Наносимый урон	Шанс восстановить немного здоровья протагонисту	Шанс найти сокровища
Негативное	Способность: Конус огня (поджигает часть территории)	Шанс появления дополнительного противника во время боя	При уничтожении противника, из толпы кидают препятствие в протагониста

Рисунок 17 – Механики различных сюжетных линий

- Сюжетная линия: Дракон (механики дракона)
  - Положительная механика: способность, позволяющая дракону улететь на безопасное расстояние от протагониста.

    - Негативная механика: способность, наносящая большой урон по территории и вынуждающая протагониста прятаться от огня.
    - Зависимая характеристика: величина наносимого драконом урона.
- Сюжетная линия: Принцесса (механики принцессы)
    - Положительная механика: активируемая способность, дающее протагонисту увеличение очков действие на несколько ходов.
    - Негативная механика: шанс появления нового слабого противника в начале хода.
    - Зависимая характеристика: шанс восстановить небольшое количество здоровья протагониста в начале хода.
- Сюжетная линия: Родной город
    - Положительная механика: шанс появления нового слабого союзника в начале хода.
    - Негативная механика: при победе над каждым противником, из толпы в протагониста кидается какой-либо предмет, наносящий урон при попадании.
    - Зависимая характеристика: шанс найти сокровище при исследовании города.

7.3 Развитие по фазам

Следующим шагом в использовании *Системы* является формализация развития сюжета – а именно, события. Сама последовательность событий была сформирована еще при задании сюжета, поэтому остается лишь задать конкретные значения результатам событий. К сожалению, не существует возможности с помощью вычислений получить точную оценку влияния события на протагониста, поэтому значения каждого из событий задается вручную дизайнером, исходя из его ощущений.

Как указано в методике работы с системой, даже в случае некорректного ввода данных при настройке, *Система* может показать на конкретные проблемы в развитии сюжета и помочь по-другому оценить результаты событий.

Выбранные результаты событий для выбранного сюжета см. на рис. 18.

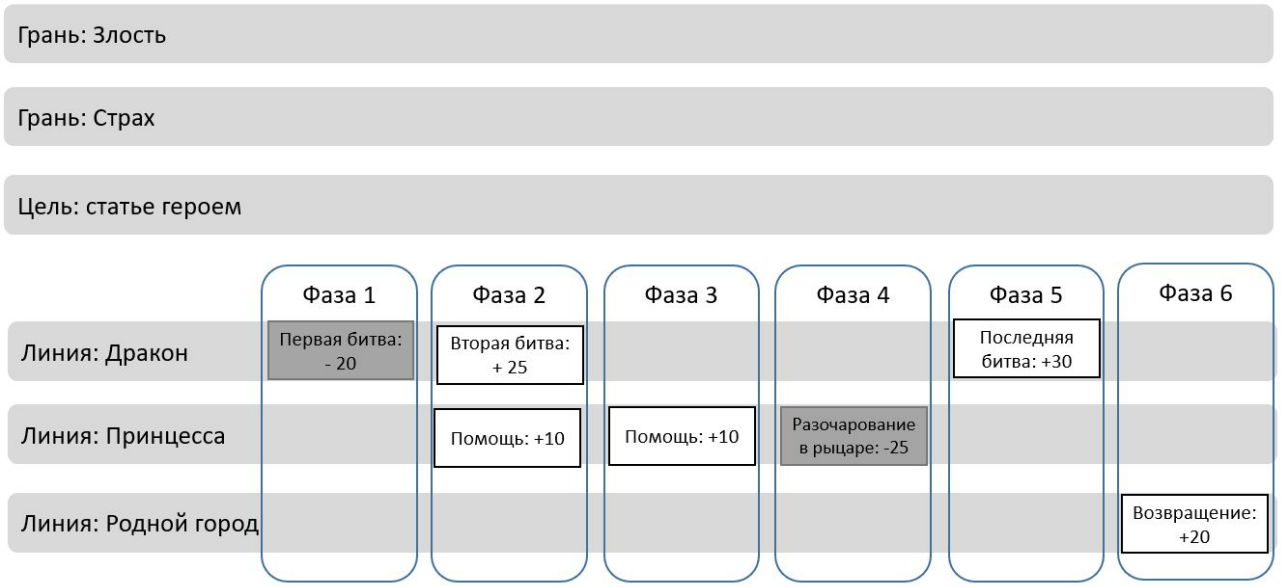

Рисунок 18 – Выбранные результаты событий для выбранного сюжета

Имея заданные значения результатов событий, можно также визуализировать динамику развития сюжета в виде графиков (см. рис. 19).

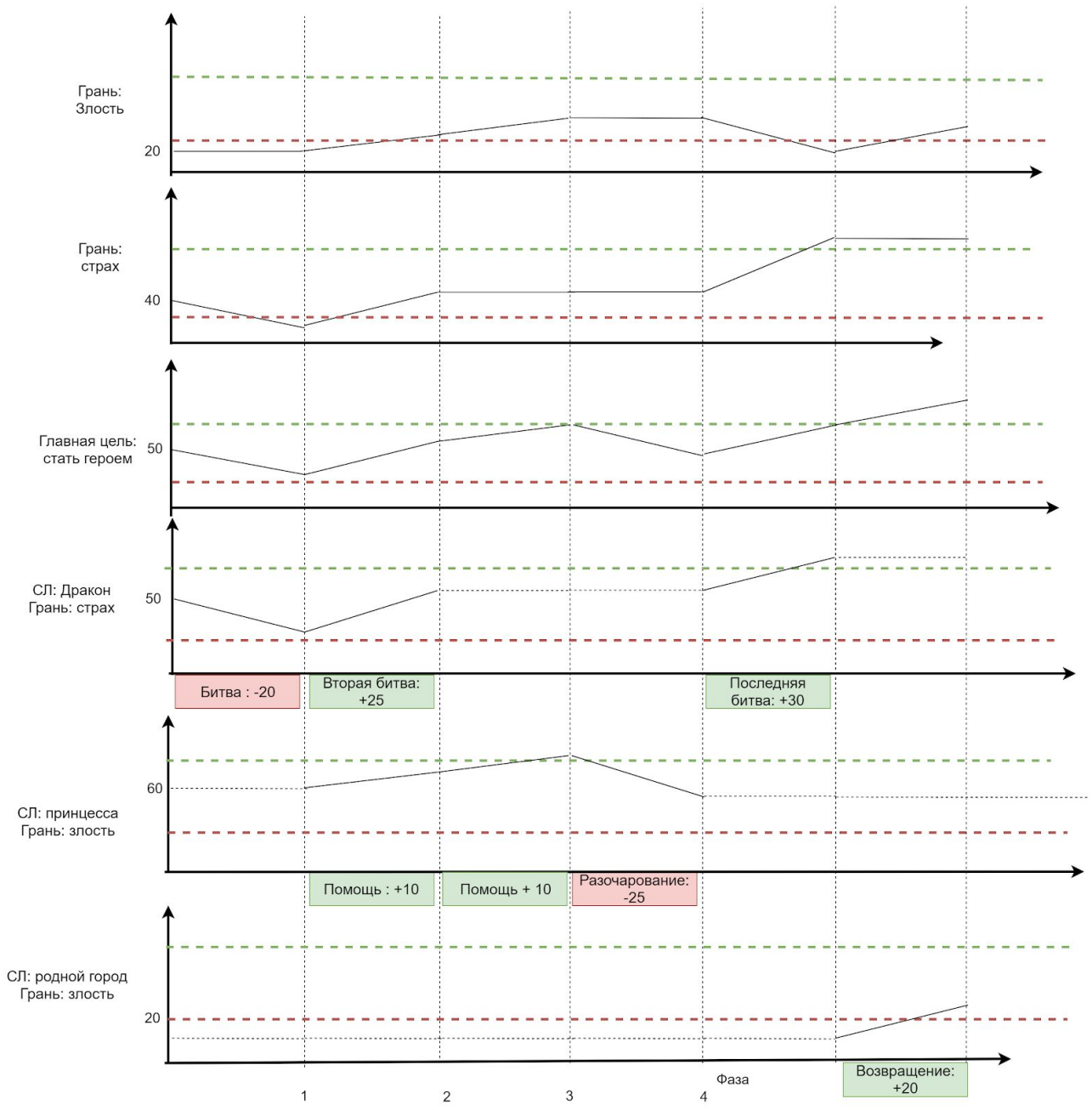

Рисунок 19 – Динамика развития сюжета

7.4 Результат

Наконец, после полной настройки *Системы* появляется возможность ее использования для получения набора подходящих механик для каждой из фаз.

Фаза 1.

	Значение	Характеристика	Критическая механика	Результат
Грань: злость	20	Высокий бонус к урону	Наносимый урон дает очки действия на следующий ход	
Грань: страх	40	Шанс промахнуться чуть завышен	--	- 20
Цель: герой	50	Стандартная регенерация	--	- 20
Линия: дракон	50	Стандартный урон дракона	--	- 20
Линия: принцесса	Не активна	--	--	
Линия: родной город	Не активна	--	--	

Рисунок 20 – Пример 1-ой фазы генерации игровых механик

Активные механики:

- Наносимый протагонисту урона дает дополнительные очки действия (ОД) на следующем ходу.

Заметные отклонения характеристик протагониста:

- Высокий наносимый персонажем урон.

Таким образом, первая фаза имеет небольшой набор механик и заметных отклонений характеристик (см. рис. 20). Помимо этого, комбинация высокого урона и механики получения дополнительных ОД при нанесении урона поощряет простой, агрессивный стиль боя.

С точки зрения ощущений игрока, фаза олицетворяет агрессию.

С точки зрения геймплея, фаза дает игроку простое, не перегруженное механиками введение.

Фаза 2.

	Значение	Характеристика	Активная механика	Результат
Грань: злость	20	Высокий бонус к урону	Наносимый урон дает очки действия на следующий ход	+10
Грань: страх	20	Значительный шанс промаха	Получение урона уменьшает очки действия на следующий ход	+25
Цель: герой	30	Заниженная регенерация	--	+35
Линия: дракон	50	Стандартный урон дракона	--	+25
Линия: принцесса	60	Шанс восстановить немного здоровья	--	+10
Линия: родной город	Не активна	--	--	

Рисунок 21 – Пример 2-ой фазы генерации игровых механик

Активные механики:

- Наносимый протагонисту урон дает дополнительные очки действия (ОД) на следующем ходу.
- Получение урона уменьшает очки действия на следующий ход.

Заметные отклонения характеристик:

- Высокий наносимый персонажем урон.
- Пониженная скорость регенерации здоровья.
- Высокий шанс промаха.

Вторая фаза дополняет игру одной активной механикой и двумя заметным отклонением характеристик (см. рис. 21). Помимо этого, активируется сюжетная линия принцессы, и вместе с ней – ее механики, помогающие герою в бою.

С точки зрения ощущений игрока, фаза олицетворяет неуверенность в себе и зависимость от сторонней помощи.

С точки зрения геймплея, фаза намеренно усложняет геймплей.

Фаза 3.

	Значение	Характеристика	Активная механика	Результат
Грань: злость	30	Значительный бонус к урону	--	+10
Грань: страх	45	Средний шанс промаха	--	
Цель: герой	65	Повышенная регенерация	--	+10
Линия: дракон	Не активна	--	--	
Линия: принцесса	70	Значительный шанс восстановить здоровье	--	+10
Линия: родной город	Не активна	--	--	

Рисунок 22 – Пример 3-ой фазы генерации игровых механик

Активные механики: (нет).

Заметные отклонения характеристик:

- Высокий наносимый персонажем урон.
- Повышенная скорость регенерации здоровья.
- Высокий шанс повышения здоровья в начале хода.

Третья фаза резко упрощает игру и в плане количества механик, и в плане защитных характеристик персонажа (см. рис. 22). Сюжетная линия дракона поставлена на паузу, и ее механики не активны в данной фазе.

С точки зрения ощущений игрока, фаза олицетворяет воодушевление и чувство безопасности.

С точки зрения геймплея, фаза упрощает геймплей для возможности игроку передохнуть после скачка сложности предыдущей фазы.

Фаза 4.

	Значение	Характеристика	Критическая механика	Результат
Грань: злость	40	Стандартный урон	--	-25
Грань: страх	45	Средний шанс промаха	--	
Цель: герой	75	Высокая регенерация	Чем меньше здоровья, тем больше урон	-25
Линия: дракон	Не активна	--	--	
Линия: принцесса	80	Шанс восстановить немного здоровья	Увеличивает количество очков действия	-25
Линия: родной город	Не активна	--	--	

Рисунок 23 – Пример 4-ой фазы генерации игровых механик

Активные механики:

- Урон персонажа увеличивается при понижении его здоровья.
- Увеличенное количество ОД протагониста.

Заметные отклонения характеристик:

- Высокая регенерации здоровья.
- Высокий шанс повышения здоровья в начале хода.

Четвертая фаза полностью использует синергию механик главной цели (см. рис. 23), и благодаря высоким защитным характеристикам призывает игрока пускаться в гущу битвы без опасений. При этом урон персонажа уменьшается, что в итоге замедляет темп игры.

С точки зрения ощущений игрока, фаза олицетворяет героизм и уверенность в себе.

С точки зрения геймплея, фаза замедляет темп игры, но поощряет использование синергии механик.

Фаза 5.

	Значение	Характеристика	Критическая механика	Результат
Грань: злость	15	Высокий бонус к урону	Наносимый урон дает очки действия на следующий ход	
Грань: страх	45	Средний шанс промаха	--	+30
Цель: герой	50	Средняя регенерация	--	+30
Линия: дракон	50	Высокий урон дракона	--	+30
Линия: принцесса	Не активна	--	--	
Линия: родной город	Не активна	--	--	

Рисунок 24 – Пример 5-ой фазы генерации игровых механик

Активные механики:

- Наносимый протагонистом увеличивает количество очков действия в следующем ходу.

Заметные отклонения характеристик:

- Высокий наносимый урон протагониста.
- Высокий наносимый урон дракона.

Пятая фаза отнимает синергию механик главной цели, а также отнимает защитные способности принцессы (см. рис. 24). Взамен игрок снова получает набор из инструментов агрессии.

С точки зрения ощущений игрока, фаза олицетворяет ярость.

С точки зрения геймплея, фаза требует от игрока приспособиться к более рискованному геймплею, и возвращает для этого инструменты для динамичного геймплея.

Фаза 6.

	Значение	Характеристика	Критическая механика	Результат
Грань: злость	15	Высокий бонус к урону	Наносимый урон дает очки действия на следующий ход	+20
Грань: страх	75	Средний шанс промаха	Бесстрашный рывок	
Цель: герой	80	Заниженная регенерация	Чем меньше здоровья, тем больше урон	+20
Линия: дракон	Не активна	--	--	
Линия: принцесса	Не активна	--	--	+20
Линия: родной город	20	Низкий шанс найти сокровища	Препятствия после победы над противником	

Рисунок 25 – Пример 6-ой фазы генерации игровых механик

Активные механики:

- Наносимый протагонистом увеличивает количество очков действия в следующем ходу.
- Рывок.
- Наносимый урон увеличивается по мере уменьшения здоровья.
- При победе над противником, окружение создает препятствия в виде наносящих урон предметов.

Заметные отклонения характеристик:

- Высокая регенерации здоровья.
- Низкий шанс промаха.
- Высокий наносимый урон.

- Низкий шанс найти сокровища.

Шестая фаза дает игроку и защитные, и агрессивные механики, позволяя ему подбирать свой стиль по ходу битвы (см. рис. 25). В то же время окружение создает новые типы препятствий для игрока.

С точки зрения ощущений игрока, фаза олицетворяет заработанную уверенность в себе протагониста.

С точки зрения геймплея, фаза дает максимальное количество инструментов для полной свободы выбора стиля игры.

Анализируя **результаты** использования *Системы* для генерации механик под заданный сюжет, можно заметить, что полученные механики действительно соответствуют ожидаемому для каждого события настроению:

- Фаза 1 – ярость и неопытность;
- Фаза 2 – страх;
- Фаза 3 – воодушевление;
- Фаза 4 – уверенность в себе;
- Фаза 5 – вернувшаяся ярость;
- Фаза 6 – опытность протагониста.

Более того, полученные механики образуют жизнеспособную кривую сложности и загруженности геймплея:

- Фаза 1 – введение в игру;
- Фаза 2 – подъем сложности и увеличение количества механик;
- Фаза 3 – период для отдыха и простого геймплея;
- Фаза 4 – увеличение возможностей игрока;
- Фаза 5 – возвращение к базовому динамичному геймплею с новыми, дополнительными препятствиями;
- Фаза 6 – максимальные возможности персонажа.

Таким образом, можно сделать вывод, что *Система* может автоматически создавать рабочую динамику развития геймплея из заданного формализованного сюжета.

8 Анализ системы

8.1 Эффективность системы

Система как инструмент дизайнера. *Система* ставит жесткие условия на формализацию параметров сюжета:

- границы значений сюжетных объектов;
- заранее определенные темы и сюжетные линии;
- строго определенное и ограниченное количество механик, связанных с сюжетными объектами.

При работе с *Системой* некоторым дизайнерам эти ограничения могут показаться отрицательными, однако каждое из этих ограничений имеет свой положительный эффект на процесс разработки сюжета и генерируемых механик:

- границы значений позволяют оценивать уровень состояний, обнаруживать перегрузку или недостаточную развитость объектов;
- четко определенные темы и сюжетные линии помогают изначально понять направление, тематику и смысл сюжета, позволяя сосредоточиться на главных его аспектах;
- количество механик определяется ровно такое, чтобы они давали возможность дизайнеру перенести изменения состояний сюжета на геймплей. При этом не позволяется назначать большее количество механик, так как любое дальнейшее усложнение уменьшает вероятность того, что игрок сможет заметить явную связь между изменениями состояний объектов и изменениями геймплея.

Также, как продемонстрировано на примерах в главе “Методика работы с системой”, визуализация – основной инструмент *Системы*, позволяющий геймдизайнеру здраво оценить сюжет на всём его протяжении. Помимо этого,

визуализация позволяет мгновенно увидеть эффект внесенных в сюжет изменений.

Таким образом можно утверждать, что *Система* является мощным инструментом для правильного расставления приоритетов и акцентов игры еще на этапе ее планирования.

Система как формализация связи сюжета и геймплея. *Система* предоставляет способ формализовать связь сюжета и геймплея, что в контексте данной работы используется для возможности анализа сбалансированности сюжета и геймплея без необходимости рассматривать сюжет и геймплей как две слабо связанные между собой сущности. Помимо этого, сам факт работоспособности итоговой системы доказывает, что связь существует и ее осознанное использование способно приносить положительные результаты.

То есть даже если представленная в данной работе *Система* не будет использоваться при разработке реальных проектов, продемонстрированный подход к рассмотрению сюжета и геймплея может быть использован для создания в будущем более совершенных систем. Или систем, заточенных на использование в контексте конкретных жанров и наборов механик.

Система как инструмент генерации. Фрагментированность входных данных о развитии сюжета и отсутствие зависимости от изменений “в будущем” позволяют использовать *Систему* для генерации механик автоматически, без ручного вмешательства в процессе. Это делает *Систему* пригодной для использования в алгоритмах и программах процедурной генерации контента игр.

8.2 Ограничения системы

Намеренные ограничения. При определении концепции *Системы*, был принят список ограничений: обход генерации core-механик, требование к типу жанра (история одного протагониста) и единая цель сюжета. Они отрицательно влияют на область применимости Системы, но для целей данной работы было достаточно показать эффективность подхода хотя бы на ограниченном количестве жанров и сюжетов.

Необходимость широкой ручной настройки. Несмотря на то, что одна из целей *Системы* – частично автоматизировать процесс дизайна и/или генерации механик, для своего полного функционирования *Система* требует большое количество ручной настройки. В некоторых случаях сложность этих настроек может перекрывать пользу, извлекаемую из результатов использования *Системы*.

Вероятность создания неиспользуемого контента. Четкое задание количества механик на каждый сюжетный объект потенциально может привести к тому, что в процессе настройки *Системы* придется реализовывать механики, которые так и не будут использоваться в итоговом варианте игры. Это проблема может перерасти в недопустимые масштабы при использовании *Системы* при разработке линейных игр AAA-уровня. В то же время в ряде других случаев это ограничение незначительно или просто неактуально: например, в небольших проектах или играх с процедурной генерацией сюжета.

9 Заключение

В результате обзора литературы в данной работе была выделена актуальная проблема сферы процедурной генерации контента для игр: несмотря на потенциал, демонстрируемый отдельными проектами, сейчас критически не хватает подходов к анализу геймплея с точки зрения его связи с нарративом. В качестве одного из шагов к решению данной проблемы, в данной работе была предложена система формализации связи между геймплеем и сюжетом.

Эта система разбивает обобщенное понятие “сюжет” в контексте игр на набор сюжетных объектов, которые в сумме образуют достаточно полное представление о тематике и содержании истории. Далее, с помощью определения различных состояний этих объектов, к ним по заданным правилам привязываются механики, которые активируются и меняют собственные параметры в зависимости от состояний сюжетных объектов в каждый момент времени.

Предложенная система была реализована в виде библиотеки классов на языке C# (ссылка на репозиторий в открытом доступе: http://gititis.kpfu.ru/VlVLipatov/StoryMechanicsGeneration.git), и протестирована на примере короткого сюжета. В результате эксперимента была показано, что система генерирует наборы механик, которые поддерживают требуемый для сюжета настрой, создают периодические подъемы и спады сложности и загруженности геймплея. Помимо этого, было выявлено, что *Система* может использоваться как инструмент визуализации сюжета для дизайнера, а также как внутренний модуль системы генерации, способный корректно работать даже без информации о будущем развитии сюжета.

Таким образом, можно сделать вывод, что *Система* успешно выполняет свою роль как способ формализации сюжета и геймплея. На ее основании можно проводить анализ уже разработанных игр, разрабатывать системы

процедурной генерации контента, продумывать сюжет и улучшать ощущения игроков. Она не завязана на конкретный жанр или тип игры, и потому может быть использована в широком спектре проектов.

Однако, на момент завершения данной работы, *Система* всё еще требует более объемного тестирования и развития. Поэтому в качестве целей для будущих работ по **автогенерации сценарного прототипа** (см., например [23]) ставится реализация гибкого **инструмента визуализации** входных и выходных данных системы (ср., например [24]), создание **демонстрационного прототипа игры** с использованием разработанной системы (ср., например [25]), а также расширение возможностей с помощью добавления **Глав**, позволяющих рассматривать сюжеты с множеством протагонистов и целей. В случае успеха также планируется начать исследование связи сюжета и геймплея с другой стороны: возможность **генерации нарратива из геймплея**, то есть зависимость событий от использования игроком различных механик.

10 Список использованных источников

1. Adams E. W. Resolutions to some problems in interactive storytelling //University of Teesside. – 2013. – 189 p.
2. Cheong Y. G., Young R. M. Suspenser: A story generation system for suspense //IEEE Transactions on Computational Intelligence and AI in Games. – 2015. – V. 7. – №. 1. – P. 39-52.
3. Szilas N. IDtension: a narrative engine for Interactive Drama //Proceedings of the technologies for interactive digital storytelling and entertainment (TIDSE) conference. – 2003. – V. 3. – №. 2. – P. 1-11.
4. Swanson R., Gordon A. S. Say anything: Using textual case-based reasoning to enable open-domain interactive storytelling //ACM Transactions on Interactive Intelligent Systems (TiiS). – 2012. – V. 2. – №. 3. – P. 16.
5. Mateas M., Stern A. Façade: An experiment in building a fully-realized interactive drama //Game developers conference. – 2003. – V. 2. – P. 4-8.
6. Koenitz H. Towards a Specific Theory of Interactive Digital Narrative //Interactive Digital Narrative. – Routledge. – 2015. – P. 107-121.
7. Sim Y. T., Mitchell A. Wordless Games: Gameplay as Narrative Technique //International Conference on Interactive Digital Storytelling. – Springer, Cham. – 2017. – P. 137-149.
8. Dubbelman T. Repetition, Reward and Mastery: The Value of Game Design Patterns for the Analysis of Narrative Game Mechanics //International Conference on Interactive Digital Storytelling. – Springer, Cham. – 2017. – P.286-289.
9. Shibolet Y. Game Movement as Enactive Focalization //Press Start. – 2018. – V. 4. – №. 2. – P. 51-71.
10. Zook A., Riedl M. O. Automatic Game Design via Mechanic Generation // AAAI. – 2014. – P. 530-537.

11. Hastings E. J., Guha R. K., Stanley K. O. Evolving content in the galactic arms race video game //Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on. – IEEE. – 2009. – P. 241-248.
12. Campbell J. The hero with a thousand faces. – New World Library. – 2008. – V.17. – 432 p.
13. Riedl M. O., Young R. M. Narrative planning: Balancing plot and character //Journal of Artificial Intelligence Research. – 2010. – V. 39. – P. 217-268.
14. Shibolet Y. Game Movement as Enactive Focalization //Press Start. – 2018. – V. 4. – №. 2. – P. 51-71.
15. Caroux L. et al. Player–video game interaction: A systematic review of current concepts //Computers in Human Behavior. – 2015. – V. 48. – P.366-381.
16. Szilas N. A new approach to interactive drama: From intelligent characters to an intelligent virtual narrator // Proceedings of the AAAI Spring Symposium on AI and Interactive Entertainment. – 2001. – P. 72-76.
17. Szilas N., Marty O., Réty J. H. Authoring highly generative interactive drama //International Conference on Virtual Storytelling. – Springer, Berlin, Heidelberg. – 2003. – P. 37-46.
18. Szilas N., Richle U., Dumas J. E. Structural writing, a design principle for interactive drama //International Conference on Interactive Digital Storytelling. – Springer, Berlin, Heidelberg. – 2012. – P. 72-83.
19. Szilas N. The mutiny: An interactive drama on idtension //Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts. – ACM. – 2008. – P. 539-540.
20. Boyan A., GrizzArd M. A., Bowman N. A massively moral game? Mass Effect as a case study to understand the influence of players' moral intuitions on adherence to hero or antihero play styles //Journal of Gaming & Virtual Worlds. – 2015. – V. 7. – №. 1. – P. 41-57.

21. Barnett M., Sharp C. The moral choice of inFAMOUS: law and morality in video games //Griffith Law Review. – 2015. – V. 24. – №. 3. – P. 482-499.
22. Schell J. The Art of Game Design: A book of lenses. – AK Peters/CRC Press. – 2014. – 600 p.
23. Сахибгареева Г.Ф., Кугуракова В.В. Концепт инструмента автоматического создания сценарного прототипа компьютерной игры / Г. Ф. Сахибгареева, В. В. Кугуракова // Электронные библиотеки. – 2018. – Т. 21. – №3-4. – С. 235-249.
24. Доброквашина А.С., Газизова Э.А. автоматизация проектирования игрового прототипа на основании обработки формализованного игрового дизайн-документа. // Ученые записки ИСиГЗ, Казань. – 2019. – С.583–589.
25. Газизова Э.А., Доброквашина А.С. Автоматическая генерация игрового прототипа // Ученые записки ИСиГЗ, Казань. – 2019. – С. 578–582.

11 Приложение

11.1 Исходный код программы

StoryManager.cs

```
using System.Collections.Generic;
using Assets.Scripts.MechanicsGeneration.Lines;
namespace Assets.Scripts.MechanicsGeneration
{
	public class StoryManager
	{
		public Dictionary<string, IMechanic> Mechanics { get; } = new
Dictionary<string, IMechanic>();

		public Dictionary<string, StoryObject> Facets { get; } = new
Dictionary<string, StoryObject>();

		public StoryObject Goal { get; private set; }

		public List<StoryPhase> Phases { get; } = new List<StoryPhase>();

		public Dictionary<string, StoryLine> StoryLines { get; } = new
Dictionary<string, StoryLine>();

		private int _currentPhaseCode;

		public StoryPhase CurrentPhase => _currentPhaseCode < Phases.Count ?
Phases[_currentPhaseCode] : null;

		public void SetMechanics(IEnumerable<IMechanic> mechanics)
		{
			foreach (var mechanic in mechanics)
			{
				Mechanics[mechanic.UniqueCode] = mechanic;
			}

		}

		public void SetFacets(IEnumerable<StoryObject> characterStateLines)
		{
			foreach (var characterStateLine in characterStateLines)
			{
				Facets[characterStateLine.UniqueCode] =
characterStateLine;
			}
		}

		public void SetStoryTheme(StoryObject goalLine, IEnumerable<StoryLine>
storyLines)
```

```
            {
                  Goal = goalLine;
                  foreach (var storyLine in storyLines)
                  {
                        StoryLines[storyLine.UniqueCode] = storyLine;
                  }
            }

            public void SetStoryFlow(IEnumerable<StoryPhase> storyPhases)
            {
                  Phases.AddRange(storyPhases);
            }

            /// <returns>null if story ended</returns>
            public IEnumerable<IMechanic> NextPhase()
            {
                  if (Phases.Count <= _currentPhaseCode)
                  {
                        return null;
                  }

                  var currentPhase = Phases[_currentPhaseCode];
                  _currentPhaseCode += 1;

                  var activeStoryObjects = GetActiveStoryObjects(currentPhase);

                  var activeMechanics = new List<IMechanic>();
                  foreach (var storyObject in activeStoryObjects)
                  {
                        CheckMechanic(activeMechanics,
storyObject.ActiveMechanic, storyObject);
                        CheckMechanic(activeMechanics,
storyObject.Characteristic, storyObject);
                  }

                  UpdateValues(currentPhase);

                  return activeMechanics;
            }

            private void CheckMechanic(List<IMechanic> mechanicsList, string code,
StoryObject storyObject)
            {
                  if (code != null && Mechanics.TryGetValue(code, out var
mechanic))
                  {
                        mechanicsList.Add(mechanic);
                        if (mechanic is ICharacteristicMechanic
characteristicMechanic)
                        {

characteristicMechanic.UpdateValue(storyObject.CurrentPercentageValue);
                        }
```

```
                }
            }

            private void UpdateValues(StoryPhase phase)
            {
                foreach (var storyEvent in phase.Events)
                {
                    StoryLine storyLine;
                    if (StoryLines.TryGetValue(storyEvent.StoryLineCode, out
storyLine))
                    {
                        storyLine.UpdateValue(storyEvent.Result);
                    }

                    if (storyLine != null &&
Facets.TryGetValue(storyLine.EffectedFacetCode, out var facetLine))
                    {
                        facetLine.UpdateValue(storyEvent.Result);
                    }

                    Goal.UpdateValue(storyEvent.Result);
                }
            }

            private List<StoryObject> GetActiveStoryObjects(StoryPhase storyPhase)
            {
                var activeObjects = new List<StoryObject>();

                foreach (var facet in Facets)
                {
                    activeObjects.Add(facet.Value);
                }

                activeObjects.Add(Goal);

                foreach (var storyEvent in storyPhase.Events)
                {
                    if (StoryLines.TryGetValue(storyEvent.StoryLineCode, out
var storyLine))
                    {
                        activeObjects.Add(storyLine);
                    }
                }

                return activeObjects;
            }
        }
}
```

StoryPhase.cs

```
using System.Collections.Generic;
```

```
namespace Assets.Scripts.MechanicsGeneration
{
	public class StoryPhase
	{
		public List<StoryEvent> Events { get; } = new List<StoryEvent>();

		public string Name { get; set; }
	}
}
```

StoryEvent.cs

```
namespace Assets.Scripts.MechanicsGeneration
{
	public class StoryEvent
	{
		public StoryEvent(string name,  string storyLineCode, int result)
		{
			Name = name;
			StoryLineCode = storyLineCode;
			Result = result;
		}

		public string Name { get; }
		public string Description { get; set; }
		public string StoryLineCode { get; }
		public int Result { get; }
	}
}
```

IMechanic.cs

```
namespace Assets.Scripts.MechanicsGeneration
{
	public interface IMechanic
	{
		string UniqueCode { get; }

		string Name { get; }
	}


}
```

ICharacteristicMechanic.cs

```
namespace Assets.Scripts.MechanicsGeneration
{
	public interface ICharacteristicMechanic : IMechanic
	{
		void UpdateValue(int percent);
	}
}
```

SystemObject.cs

```
namespace Assets.Scripts.MechanicsGeneration.Lines
{
	public abstract class SystemObject
	{
		public string UniqueCode { get; }

		public string Name { get; }

		protected SystemObject(string code, string name)
		{
			UniqueCode = code;
			Name = name;
		}
	}
}
```

StoryObject.cs

```
using System;
namespace Assets.Scripts.MechanicsGeneration.Lines
{
	[Serializable]
	public class StoryObject : SystemObject
	{

		public int CurrentValue { get; private set; }

		public int HighThreshold { get; private set; }

		public int LowThreshold { get; private set; }


		public const int MaxValue = 100;

		public const int MinValue = 0;

		public string ActiveMechanic
		{
			get
			{
				if (CurrentValue >= HighThreshold)
				{
					return HighMechanic;
				}

				if (CurrentValue <= LowThreshold)
				{
					return LowMechanic;
				}

				return null;
			}
		}
```

```
            public string Characteristic { get; private set; }

            private string HighMechanic { get; set; }

            private string LowMechanic { get; set; }

            public StoryObject(string code, string name) : base(code, name)
            {
            }

            public void SetThresholds(int currentValue, int highThreshold, int
lowThreshold)
            {
                  CurrentValue = currentValue;
                  HighThreshold = highThreshold;
                  LowThreshold = lowThreshold;
            }

            public void SetMechanics(string highMechanic, string lowMechanic,
string middleMechanic)
            {
                  HighMechanic = highMechanic;
                  LowMechanic = lowMechanic;
                  Characteristic = middleMechanic;
            }

            public void UpdateValue(int valueChange)
            {
                  CurrentValue = Math.Min(Math.Max(CurrentValue + valueChange,
MinValue), MaxValue);
            }

            public int CurrentPercentageValue
            {
                  get
                  {
                        if (CurrentValue >= HighThreshold)
                        {
                              return 100 * (CurrentValue - HighThreshold) /
(MaxValue - HighThreshold);
                        }
                        else if (CurrentValue <= LowThreshold)
                        {
                              return 100 * (CurrentValue - MinValue) /
(LowThreshold - MinValue);
                        }
                        else
                        {
                              return 100 * (CurrentValue - LowThreshold) /
(HighThreshold - LowThreshold);
                        }
                  }
            }
```

```
        }
}
```

StoryLine.cs

```
namespace Assets.Scripts.MechanicsGeneration.Lines
{
        public class StoryLine : StoryObject
        {
                public string Description { get; set; }

                public string EffectedFacetCode { get; private set; }

                public void SetFacet(string facetLineCode)
                {
                        EffectedFacetCode = facetLineCode;
                }

                public StoryLine(string code, string name) : base(code, name)
                {
                }
        }
}
```

11.2 Входные данные для эксперимента

Mechanics.txt


```
{
 "StateMechanics" : [
   {
    "Name" : "Berserk",
    "Id" : "player_low_hp_high_dmg"
   },
        {
    "Name" : "Regeneration",
    "Id" : "player_regeneration_char"
   },
        {
    "Name" : "Vampire",
    "Id" : "player_kill_heal"
   },

        {
    "Name" : "Patient",
    "Id" : "player_next_turn_action"
   },
        {
    "Name" : "Damage buff",
    "Id" : "player_damage_char"
   },
        {
    "Name" : "Agressive",
    "Id" : "player_attack_action"
   },

        {
    "Name" : "Charge",
```

```
      "Id" : "player_charge"
    },
            {
      "Name" : "Miss chance",
      "Id" : "player_miss_char"
    },
            {
      "Name" : "Fear of pain",
      "Id" : "player_attacked_action"
    }
  ],
  "StoryLineMechanics" : [
            {
      "Name" : "Flight",
      "Id" : "dragon_flight"
    },
            {
      "Name" : "Damage buff",
      "Id" : "dragon_dmg_char"
    },
            {
      "Name" : "Cone of flame",
      "Id" : "dragon_cone"
    },
            {
      "Name" : "Helping hand",
      "Id" : "princess_action"
    },
            {
      "Name" : "Healing",
      "Id" : "princess_heal_char"
    },
            {
      "Name" : "Angry admirers",
      "Id" : "princess_foe"
    },
            {
      "Name" : "Fans",
      "Id" : "town_ally"
    },
            {
      "Name" : "Treasures",
      "Id" : "town_treasure_char"
    },
            {
      "Name" : "Thugs",
      "Id" : "town_foe"
    },
  ]
}
```


StoryEvents.txt


```
{
  "Phases" : [
    {
      "Events" : [
        {
          "Name" : "First fight",
          "Result" : -20,
          "StoryLine" : "dragon"
```

```
      }
    ]
  },
          {
    "Events" : [
      {
        "Name" : "Second fight",
        "Result" : 25,
        "StoryLine" : "dragon",
      },
      {
        "Name" : "Help",
        "Result" : 10,
        "StoryLine" : "princess",
      }
    ]
  },
          {
    "Events" : [
      {
        "Name" : "Help",
        "Result" : 10,
        "StoryLine" : "princess"
      }
    ]
  },
          {
    "Events" : [
      {
        "Name" : "Dissapointment",
        "Result" : -25,
        "StoryLine" : "princess",
      }
    ]
  },
  {
    "Events" : [
      {
        "Name" : "Last fight",
        "Result" : 30,
        "StoryLine" : "dragon",
      }
    ]
  },
          {
    "Events" : [
      {
        "Name" : "Returning",
        "Result" : 20,
        "StoryLine" : "hometown",
      }
    ]
  }
 ]
}
```


StoryThemes.txt


```
{
 "CharacterStates": [
   {
```

```
    "Id": "fear",
    "Name" : "Fear",
            "StartValue" : 40,
    "HighThreshold" : 75,
    "LowThreshold" : 25,
    "HighMechanic" : "player_charge",
    "MiddleMechanic" : "player_miss_char",
    "LowMechanic" : "player_attacked_action"
  },
  {
    "Id": "anger",
    "Name" : "anger",
            "StartValue" : 20,
    "HighThreshold" : 75,
    "LowThreshold" : 25,
    "HighMechanic" : "player_next_turn_action",
            "MiddleMechanic" : "player_damage_char",
    "LowMechanic" : "player_attack_action"
  }
],
"Goal": {
    "Id" : "hero",
    "Name" : "Hero",
            "StartValue" : 50,
    "HighThreshold" : 75,
    "LowThreshold" : 25,
    "HighMechanic" : "player_low_hp_high_dmg",
    "MiddleMechanic" : "player_regeneration_char",
    "LowMechanic" : "player_kill_heal"
},
"StoryLines" : [
  {
            "Id" : "dragon",
            "StateLine" : "fear",
    "Name" : "Dragon",
            "StartValue" : 50,
    "HighThreshold" : 75,
    "LowThreshold" : 25,
            "HighMechanic" : "dragon_flight",
    "MiddleMechanic" : "dragon_dmg_char",
    "LowMechanic" : "dragon_cone"
  },
  {
           "Id" : "princess",
    "Name" : "Princess",
            "StateLine" : "anger",
    "StartValue" : 60,
    "HighThreshold" : 75,
    "LowThreshold" : 25,
    "HighMechanic" : "princess_action",
    "MiddleMechanic" : "princess_heal_char",
    "LowMechanic" : "princess_foe"
  },
   {
           "Id" : "hometown",
    "Name" : "Hometown",
            "StateLine" : "anger",
    "StartValue" : 20,
    "HighThreshold" : 75,
    "LowThreshold" : 25,
```

"HighMechanic" : "town_ally",
"MiddleMechanic" : "town_treasure_char",
"LowMechanic" : "town_foe"
}
]

}