МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет»

Кафедра инфокоммуникаций

Реферат

«Программная инженерия для игр: технологии, инструменты и методы разработки»

по дисциплине « Программная инженерия»

Выполнил студент группы ПИЖ-б-о-21-1
Ботвинкин Н.С. « »_____20__г.
Подпись студента_____
Работа защищена « »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Программная инженерия для игр является одной из самых динамично развивающихся областей в сфере разработки программного обеспечения. С ростом игровой индустрии, которая по доходам уже превзошла кино и музыку, разработка игр становится важнейшим направлением современной программной инженерии. Интерактивные развлечения становятся всё более сложными, требуя передовых подходов, инструментов и методов для создания качественных и инновационных продуктов. Игры включают в себя сложные графические и физические движки, искусственный интеллект, многопользовательские системы и интерактивные среды, что делает их разработку крайне технически сложной и многообразной задачей. В этом контексте понимание современных подходов и инструментов разработки игр является важным для программистов, инженеров и исследователей, стремящихся к созданию высококачественных и конкурентоспособных продуктов.

Цель данного реферата — исследовать и проанализировать современные подходы, инструменты и методы разработки в области программной инженерии для игр. Основные вопросы, которые будут рассмотрены, включают: какие подходы и методы используются для управления процессом разработки игр, какие инструменты применяются для реализации различных аспектов игры, и как современные технологии влияют на эффективность и качество разработки.

В реферате будут рассмотрены подходы к разработке игр, такие как инди и корпоративная. Кроме того, будут изучены положительные и отрицательные практики на основе реальных примеров игр.

1. ТЕОРЕТИЧЕСКАЯ БАЗА

1.1 Определения и основные понятия

В классике психологи определяют игру как непродуктивную деятельность, в которой сам процесс гораздо важнее, чем его результат.

Какие атрибуты присущи игре? Как мы можем отличить игру от любого другого процесса? Во-первых, игре присуще обучение. Это одна из ключевых характеристик. Также игра должна давать человеку обратную связь, чтобы он получал информацию о том, что же произошло в системе, с которой он играет. Без этой обратной связи игры быть не может.

Любая игра имеет правила — внешние инструкции, ограничивающие набор доступных действий. И последний атрибут любой игры — это некая роль, аватар, то есть то, что отыгрывает человек.

В Японии есть отдельный термин для людей, которые очень сильно вживаются в виртуальную роль [1]. Они провели исследование и попытались сопоставить те механизмы, которые работают в голове человека при шизофрении и при игре в компьютерные игры, и обнаружили некие аналогии. Игрок невольно начинает ассоциировать себя с некоей другой личностью, переносит на нее те атрибуты, которые он, может быть, хотел бы видеть у себя, и это тоже является частью игры. Иногда он видит себя в роли нескольких личностей, между которыми может мысленно переключаться.

Создание игры включает в себя много этапов, начиная с идеи и заканчивая сложными маркетинговыми решениями. Но обычно этот путь делится на три этапа: пре-продакшн, продакшн и пост-продакшн [1]. По сути, пре-продакшн определяет, о чем игра, зачем ее делать и что нужно для ее создания. У создателя игры может быть отличная идея для её типа, истории, которую он хочет воплотить в жизнь, или он может создать такую, которая использует определенный тип технологий. Этот этап может длиться от недели до года, в зависимости от типа проекта, имеющихся ресурсов и финансов, и обычно занимает до 20 % от общего времени производства. На данном этапе

команда довольно маленькая. Она может состоять из продюсера, программиста, концепт-художника.

Продакшн – самый длинный этап разработки. Создание игры занимает от 1 до 4 лет, и именно тогда игра действительно начинает обретать форму. История уточняется, ресурсы (персонажи, существа, объекты и окружение) создаются, правила игры устанавливаются, пишется код и многое другое.

Играбельность: первая играбельность дает гораздо лучшее представление о внешнем виде и игровом процессе (демо).

Вертикальный срез: это полностью воспроизводимый образец, который можно использовать для презентации игры студиям или инвесторам. Геймплей длится от нескольких минут до получаса и дает возможность увидеть игру из первых рук.

Пре-альфа: большая часть контента разрабатывается именно здесь. На этом этапе разработки игры необходимо будет принять несколько важных решений. Контент может быть вырезан, или для улучшения игрового процесса потребуется добавить новые элементы.

Альфа: игра «завершена», что означает, что все основные функции были добавлены, и в игру можно играть полностью от начала до конца. Некоторые элементы, такие как художественные объекты, все же, возможно, потребуется добавить, но элементы управления и функциональные возможности должны работать надлежащим образом.

Бета: на этом этапе весь контент и ресурсы интегрированы, и команде следует сосредоточиться на оптимизации, а не на добавлении новых функций или возможностей.

Золото: игра является финальной и готова к отправке в издательский центр и выпуску для широкой публики.

Пост-продакшн: после завершения производства и поставки игры процесс разработки игры продолжается, и некоторые члены команды переводятся на обслуживание (исправление ошибок, создание патчей) или создание бонусного или загружаемого контента (DLC). Другие могут перейти

к продолжению или следующему проекту. Может быть проведено вскрытие или анализ, чтобы обсудить, что сработало, а что не сработало, и определить, что можно было бы сделать лучше в следующий раз. Все проектные документы, активы и код дорабатываются, собираются и хранятся на случай, если они понадобятся в будущем.

1.2 Исторический обзор

Изначально игры создавались ради потехи программистов, чтобы проверить возможности старых ЭВМ и протестировать визуальный интерфейс. В 1952 году была создана первая логическая компьютерная игра «ОХО» – компьютерная реализация «крестиков-ноликов». Игра была создана А. С. Дугласом во время его обучения на докторскую степень в Кембриджском университете (Великобритания) [2]. Дуглас писал свою диссертацию на тему взаимодействия человека и компьютера, а игру использовал как наглядную иллюстрацию. Это событие ознаменовало собой начало игровой индустрии. В 1972 году американская компания по разработке игр «Atari» выпускает свою первую игру «Pong». Игру ждал оглушительный успех и мировая слава. Таким образом «Pong» стала первой коммерчески успешной игрой.

Первая игра с видеовставками (кат-сценами). Создан игровой автомат «Dragon’s Lair». Внутри автомата игра запускалась с оптического диска Laser Disc. Игра представляла собой интерактивный мультфильм.

На «Sega Mega Drive» выходит игра «Sonic the Hedgehog». Игра о синем ёжике, бегающем по экрану со сверхзвуковой скоростью. Игра стала самой популярной игрой на консоли, а затем ёжик Соник даже стал официальным символом компании «Sega».

На компьютерах выходит игра «Civilization». Эта игра стала родоначальником жанра «Глобальные стратегии». Также была создана первая графическая ролевая онлайн-игра – «Neverwinter Nights». Геймплей был всё тот же: выбор команд, ограниченных количеством строчек в меню, статичные картинки, текст, описывающий, что происходит вокруг.

И начиная с нулевых в игровой индустрии произойдет революция. Новое поколение консолей, новые жанры игр, удешевление их и ПК, крайне быстрое улучшение качества игр ознаменовало собой начало новой эпохи. Начиная с этого периода игры будут крайне быстро развиваться. У них будет улучшаться графика, сюжет, геймплей, и они будут обретать всё большую и большую популярность в то время как в каждом доме появляется свой ПК.

2. ОСНОВНАЯ ЧАСТЬ

2.1 Анализ текущего состояния

Существует 2 типа разработки игр: инди разработка и корпоративная разработка.

Корпоративная разработка

Любая компания — это давно запущенный механизм, главная цель которого — долгосрочное выживание всей системы, а не красота её отдельных блестящих шестерёнок.

Каждая новая строчка кода должна не просто решать свою задачу — она должна гарантировать, что тот человек, который будет взаимодействовать с ней, ничего не сломает. А в идеале сможет еще и понять.

Для больших корпораций нормально, когда на проекте нет ни одного человека, который бы понимал систему целиком. Почти всегда оригинальный автор либо уже уволился, либо сменил проект, либо пошел на повышение.

Чаще всего, AAA игры работают на собственном движке студии.

Разработка видеоигр является сложным и многогранным процессом, который включает в себя множество этапов, от выбора жанра до тестирования и выпуска игры. Для контролирования такого производства международные компании используют широко принятую методологию Жизненного цикла разработки игр (Game Development Life Cycle или GDLC) [14]. Данный подход обеспечивает структурированную основу для планирования и выполнения процесса производства. Популярность GDLC объясняется ее способностью предоставлять дорожную карту (roadmap) для планирования и выполнения проекта, что помогает обеспечить принятие необходимых мер для вывода игры на рынок.

Под словом видеоигра подразумевается программное обеспечение, работающее на специальном оборудовании. В связи с этим в производстве могут быть применимы традиционные методы разработки программного

обеспечения. Одними из наиболее популярных являются: DevOps, Канбан, SCRUM.

DevOps — это подход к разработке ПО, который объединяет различные этапы жизненного цикла, такие как разработка, тестирование, развертывание и мониторинг. Этот метод направлен на ускорение и улучшение качества процесса разработки, на повышение эффективности команды разработчиков [17].

Канбан — это метод управления процессом разработки ПО, основанный на визуализации и оптимизации потока работы. Метод базируется на пяти принципах: концентрация на качестве, снижение количества незавершенных задач, частые релизы, баланс требований и пропускной способности, борьба с источниками вариативности для улучшения предсказуемости.

SCRUM — это гибкий метод управления проектами, основанный на итеративной и инкрементальной разработке. С помощью данного метода разработчики принимают решения продуктивно и качественно. Главными ценностями SCRUM являются: обязательство, смелость, фокус, открытость, уважение. Разработка по этому методу состоит из спринтов (sprint), в конце каждого из которых выпускается новая версия продукта

Обычно корпоративная разработка проводится в 7 этапов [1].

1. Концептирование

На этом первом шаге команда придумывает концепцию игры, и проводит начальную проработку игрового дизайна. Главная цель данного этапа – это геймдизайнерская документация, включающая в себя Vision (развернутый документ, описывающий игру, как конечный бизнес-продукт) и Concept Document (начальную проработку всех аспектов игры).

В продуктовой документации геймдизайнер формулирует и сохраняет свои идеи. Исполнителю документация позволяет правильно понимать свои задачи по реализации продукта. Тестировщик четко видит, что и как тестировать. Для Продюсера эта документация предоставляет материал для

формирования планов и контроля выполнения задач. Инвестор же (особенно на ранних этапах) получает понимание, на что именно он выделяет средства.

Принципиально важно, чтобы вся проектная и продуктовая документация поддерживалась в актуальном состоянии на всех этапах развития проекта. Для её эффективного использования и обновления правильно использовать специальные инструменты. Например, использование Confluence для ведения геймдизайнерской документации сильно упрощает процесс параллельного внесения изменений несколькими участниками разработки, а также позволяет всем членам команды оперативно получать любую актуальную информацию, касающуюся продукта и всех его изменений.

Среди ключевых принципов формирования продуктовой документации стоит отметить: структурированность, защищенность от разночтений, полное описание продукта, регулярную актуализацию.

2. Прототипирование

Важный этап проектирования любой игры – это создание прототипа. То, что хорошо выглядит «на бумаге», совершенно не обязательно будет интересно в реальности. Прототип реализуется для оценки основного игрового процесса, проверки различных гипотез, проведения тестов игровых механик, для проверки ключевых технических моментов.

Очень важно на этапе создания прототипа реализовывать только то, что нужно проверить и в сжатые сроки. Прототип должен быть простым в реализации, т.к. после достижения поставленных перед ним целей, он должен быть «выкинут». Серьёзная ошибка начинающих разработчиков – нести временную инфраструктуру и «костыли» реализации кода в основной проект.

3. Вертикальный срез

Цель Вертикального среза – получить минимально возможную полноценную версию игры, включающую в себя полностью реализованный основной игровой процесс. При этом высокое качество проработки обязательно нужно воплотить только для тех игровых элементов, которые существенно влияют на восприятие продукта. При этом все базовые фичи

игры присутствуют как минимум в черновом качестве. Реализован минимальный, но достаточный для воплощения полноценного игрового процесса набор контента (один уровень или одна локация).

4. Производство контента

На этом этапе производится достаточное количество контента для первого запуска на внешнюю аудиторию. Реализуются все фичи, запланированные к закрытому бета-тестированию. Это наиболее продолжительный этап, который может занимать, для крупных клиентских проектов год и более.

На этом этапе задействуется наибольшее количество специалистов, которые занимаются производством всего основного наполнения игры. Художники создают все графические ресурсы, геймдизайнеры настраивают баланс и заполняют конфиги, программисты реализуют и полируют все фичи.

5. Friends & Family / CBT (закрытое бета-тестирование)

На этапе CBT продукт впервые демонстрируется достаточно широкой публике, хотя и лояльной продукту или компании. Среди наиболее важных задач на этом этапе выступают: поиск и исправление гейм-дизайнерских ошибок, проблем игровой логики и устранение критических багов. На этом этапе в игре присутствуют уже все ключевые фичи, создано достаточно контента для полноценной игры продолжительное время, настроены сбор и анализ статистики. Тестирование идет по тест-плану, проводятся стресс-тесты уже с привлечением реальных игроков.

6. Soft Launch / OBT (открытый бета-тест)

На этом этапе продолжается тестирование игры, но уже на широкой аудитории. Идет оптимизация под большие нагрузки. Игра должна быть готова для приема большого трафика. В игре реализован биллинг и принимаются платежи.

На этом этапе полностью завершается разработка новых фичей. Происходит feature freeze, программисты перестают реализовывать что-то новое, а полностью переключаются на отладку и тюнинг имеющихся фичей.

Геймдизайнеры, продюсер и аналитики делают выводы из собранной на CBT статистики и проверяют эффективность монетизации.

При этом, к началу этапа должна полностью функционировать инфраструктура проекта: сайт, группы соц. сетях, каналы привлечения (User Acquisition), поддержка пользователей.

7. Release

Ключевая цель – это получение прибыли. Базовый применяемый для оценки прибыльности критерий: количество денег, принесенных в среднем одним игроком за все время (LTV aka lifetime value), должно превосходить расходы на привлечение этого игрока (CPI aka cost per install).

На этом этапе должно быть полностью отлажено оперирование продукта (техническая поддержка, работа с комьюнити), соблюдаются маркетинговые и финансовые планы, ведутся работы по улучшению финансовых показателей, активно отрабатываются каналы по привлечению трафика.

Команда разработки на этом этапе занимается исправлением технических багов, выявляемых в процессе эксплуатации и оптимизацией продукта. Геймдизайнеры занимаются тонкой настройкой геймплея под реальную ситуацию в игровом мире (особенно актуально для MMO проектов). Также реализует различные внутриигровые фичи, поддерживающие новые монетизационные схемы. И конечно идет разработка и интеграция в продукт нового контента, поддерживающего интерес игроков.

Инди разработка

Инди-игра (От Independent video game) – это видеоигра, созданная отдельными лицами или небольшими командами разработчиков без финансовой и технической поддержки крупного издателя игр [10].

Игры, которые делались годами, на которые потрачены миллионы долларов, могут быть неуспешными. Да, они интересны для нас, для конкретных геймдизайнеров, которые их придумали, но не для широкой аудитории. Первая ошибка геймдизайнера – считать, что если игра, которую

он придумал, интересна ему и его друзьям, то она будет интересна широкой платящей аудитории

К 2010 году разработка игр перестала быть монополией крупных компаний. Без бюджета и опыта небольшая команда энтузиастов могла создать игру и выпустить её для широкой аудитории.

Из-за своей независимости и свободы развития инди-игры часто фокусируются на инновациях, экспериментальном игровом процессе и принятии рисков, которые обычно не допускаются в играх AAA.

Чаще всего, инди игры разрабатываются с помощью уже существующего движка.

Разработка инди игры проходит у всех по-разному. Игра может стать финансово успешной, даже если разрабатывалась без четкого видения проекта.

Не только тип разработки влияет на продажи.

Грамотная сегментация пользователей — один из краеугольных камней, на которых зиждется успех игры

Несколько десятилетий назад профессор Университета Эссекса Ричард Алан Бартл придумал модель сегментации игроков по психологическим типам. Сегодня её используют разработчики игр во всем мире

30 лет назад Бартл написал одну из первых многопользовательских игр — MUD (Multi-User Dungeon), по имени которой теперь называют целый жанр. Фактически, это прародитель всех современных MMO.

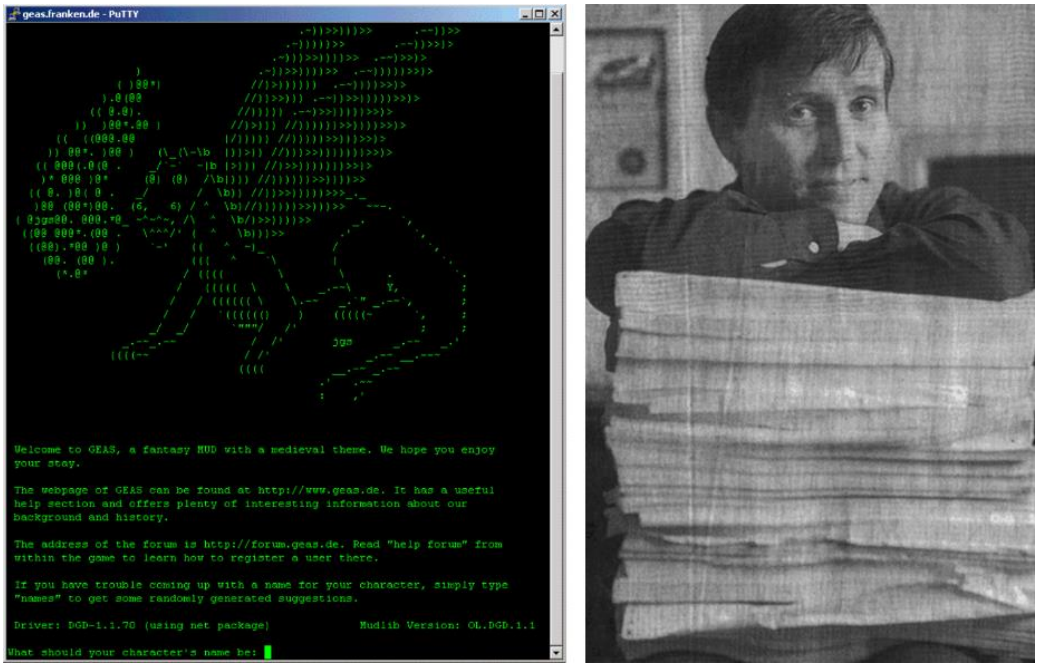

Рисунок 1 – Multi-User Dungeon

На русский язык это чаще всего переводят как «многопользовательские миры» (МПМ). Игру на любой платформе можно назвать многопользовательским миром, если она удовлетворяет трём главным критериям:

- Она должна быть бесконечной, такую игру нельзя «пройти».
- В ней участвует очень много активных игроков: тысячи, десятки тысяч.
- Игроки имеют возможность самостоятельно создавать игровой контент.

К классическим МПМ можно отнести Ultima Online или Minecraft.

В свое время Бартл задался вопросом «А что же интересно людям? Зачем они приходят играть в многопользовательские игры?». Пытаясь понять мотивацию игроков, он выделил две шкалы: «действие — взаимодействие» и «игроки — мир». Место их пересечения он назвал плоскостью интересов [18].

Рисунок 2 – Психотипы Бартла

На основании этих наблюдений Бартл выделил следующие психотипы, или группы игроков:

Накопители (Achievers). Также встречается название Карьеристы. Для них важно накопление мощи, денег, крутых артефактов — любых игровых благ и ресурсов.

Карьеристы любят получать всевозможные игровые блага. Они добывают много ресурсов, производят массу вещей. Но наибольшее удовлетворение получают от различных наград, медалей, нашивок за всевозможные достижения. Достижение поставленных целей, решение все более сложных и сложных задач, которые приводят к долгожданной награде — вот что приносит им удовольствие. То есть Карьеристов стимулирует внутриигровой рост, прогресс.

Карьеристы готовы платить за то, чтобы ускорять своё развитие, иметь время и дополнительные возможности. Но ради того, чтобы быть просто лучше других, они не раскошелятся. Для них важно доказать свое превосходство собственными усилиями. Больше всего Накопителю нравятся различные вариации на тему «потрать время — получи награду».

Карьеристы хорошо монетизируются и обладают неплохим ретеншн — показателем удержания пользователей в игре.

Киллеры (Killers). Для них главная мотивация — превосходство над другими игроками, доминирование, властвование. Они жаждут только победы.

Их меньшинство. В среднем по миру около 10%. Но ряд национальных рынков немного отличается по этому показателю в большую сторону: Россия, Турция и Южная Корея. Киллеры ценят силу, навык, влияние, доминирование. Обратите внимание, что влияние ценят и Карьеристы. А вот навык, как будет показано ниже, ценят и Исследователи. Определённые пересечения в целях есть между всеми классами, ведь они имеют пересекающиеся грани на плоскости интересов. Но мотивация у них разная.

Киллеры любят соревнования, турниры и рейтинги. Это позволяет им доказывать, что они круче других, что они кого-то превосходят. Поэтому им нравится всё, что связано с непосредственным взаимодействием с другими игроками. В ММО-играх Киллеры обожают PvP, в более мирных «фермах» они без ума от рейтингов соседей и сравнения с друзьями. Например, всякие режимы соревнования со знакомыми им людьми в играх делаются в первую очередь для Киллеров. «У моего друга на 2 монеты больше? Нет, я буду играть еще, так нельзя». Здесь мы видим пересечение этой группы с Социальщиками, которые тоже любят рейтинги друзей и взаимодействие с ними. Вот только цели у них немного разные.

Киллеры самая монетизируемая аудитория, они готовы платить за то, чтобы убивать, побеждать и доминировать. Однако ретеншн этой группы достаточно средний.

Исследователи (Explorers). Им интересно изучать игровой мир и раскрывать его тайны. Они не гонятся за активными действиями и сражениями.

Лучший повод для Исследователя прийти в вашу игру — большой выбор игровых механик, наличие вызова для ума и многообразие контента.

Исследователи любит игровые обзоры. Их нужно убедить, что ваша игра действительно хороша и стоит потраченного времени. Как и Киллеры, они ценят навык. В то же время, этот класс ориентирован на знание, на поглощение игрового контента, на изучение игровой механики и возможностей игры. Чем больше игровой мир, тем лучше. Для Исследователей важно развивать уникальные наборы талантов, чтобы иметь возможность изучать все допустимые сочетания, скрытые аспекты, нюансы и т.д. Они также любят квесты, диалоги, сюжет, который подогревает их интерес.

Исследователи обладают максимальным значением ретеншн [17], и если уж вам удалось завлечь их в игру, они будут ей достаточно преданы, пока проект подбрасывает им вызовы. Тем не менее, финансовые показатели этой группы не самые лучшие. Исследователей трудно спровоцировать на спонтанный платёж. Часто они лучше других относятся к подписке, ценят некий элемент честности. Если эти люди пришли в проект, то будут за него держаться, это преданная аудитория, хоть и слабо монетизируемая. Они часто пишут на форумах, что игра недоработана и недостаточно разнообразна, но при этом отстаивают её перед нападками представителей других психотипов.

Социальщики (Socializes). Их еще называют Тусовщики. Для них важно общение с другими игроками, социальное взаимодействие и взаимопонимание.

Они любят общение, отношения, популярность. Они любят выделиться. И здесь нужно сравнить их со всеми прочими психотипами: доля гордыни присуща всем людям. Киллеры гордятся доминированием над другими; Накопители — тем, что у них больше всех ресурсов и благ; Исследователи — тем, что они умнее других, а Социальщики — тем, что они популярнее. Поэтому, когда мы даем игрокам возможность выделяться, конкурировать между собой, соревноваться, это всегда идет на пользу проекту.

Социальщикам обязательно нужны конкурсы, чаты, сходки, ивенты, форумы. Всё, что касается массового взаимодействия. Это благодатный класс, но не без проблем. Во-первых, они, как и Исследователи, довольно слабо

монетизируются. В основном они любят покупать appearance — то есть внешний вид, красивые костюмы и т.д. При этом Социальщики имеют минимальный ретеншн среди всех классов, то есть их тяжелее всего удержать. Они пришли, поиграли, ушли в другую игру. Их удержанию способствуют акции и игровые события. Часто они приводят в игру много друзей, среди которых может быть очень платящий Киллер.

Ричард Алан Бартл, когда придумал всю эту систему, сделал несколько далеко идущих выводов относительно балансировки аудитории. Чтобы максимизировать выручку с игры и увеличить retention проекта, нам нужно поддерживать грамотное соотношение психотипов. И здесь нам помогут несколько разработанных Бартлом и его последователями сценариев.

Первый сценарий заключается в стабильном балансе, когда соотношение между всеми психотипами практически никогда не меняется и доля каждого из них достаточно велика. Это удобно с точки зрения гейм-дизайна. Но у этого сценария есть недостаток: абсолютный баланс крайне плохо совместим с моделью free2play и постоянной необходимостью нагонять в игру свежий трафик. Чуть лучше этот сценарий работает с подпиской. Он предлагает высокий retention, который легко рушится монетизацией. Пример: World of Warcraft.

Второй сценарий: баланс Киллеров и Карьеристов. Идея заключается в том, что большинство игроков — карьеристы, которые приносят выручку на заточке, предметах и бустах. Они могут обеспечивать больше трети всей выручки. Доля Киллеров в этом сценарии должна быть значительно выше, чем в других проектах. Это как раз и является залогом успеха. В этом случае Киллеры приносят чуть ли не 2/3 выручки. В то же время Социальщиков и Исследователей здесь не так много. Пример: Lost Ark.

Третий сценарий: доминирование Социальщиков. Достаточно популярный вариант. Очень простой с точки зрения гейм-дизайна, но не очень выгодный финансово. Критическая масса Социальщиков позволяет постоянно поддерживать большой размер аудитории. Самый простой вариант разрушить

эту модель — выключить трафик. Количество Социальщиков падает, критическая масса теряется, игра умирает. Ее достаточно легко убить обычным отказом от маркетинга. Но зато такая модель устойчива к потрясениям, ошибкам в акциях, ивентах и марафонах, потому что Социальщики воспринимают это как часть игры. Пример: genshin impact

Четвертый сценарий: доминирование Исследователей. Исследователей среди игроков вообще не так много – до 30%. В данном же сценарии их доля достигает 50%. Их тяжело заманить в новые игры, но играют они достаточно долго. Они могут играть в ваш проект, даже когда вы отказались от маркетинга, причем будут по инерции платить. Это очень хороший самоподдерживающийся проект, для которого публикация новостей раз в месяц — приемлемая активность для поддержания жизни. Пример: stardew valley

Лабораторный способ определения карты психотипов игрока — это тест Бартла. Таких тестов в сети немного и все они недостаточно объективны, так как предлагают игроку ответить на вопросы о многопользовательских играх. Имея игровой опыт, можно предугадать ответы. Не имея его, нельзя пройти тест адекватно. Большинство русскоязычных тестов – перевод оригинального теста Бартла с английского.

Вне лабораторий мы пользуемся эвристикой — алгоритмами, которые присваивают баллы по каждой из 4 шкал на основании действий игрока. Написал три сообщения в чате — получай медаль «Социальщик»; убил 24 игрока — получай +10 баллов к Киллеру; создал три «склянки великой скорби» — получи +1 к Исследователю и +3 к Карьеристу. Благодаря эвристике мы можем примерно оценить психотип каждого игрока и построить общую картину распределения.

Для создания компьютерных игр используется какой-то фреймворк (движок) [5]. Использование фреймворка существенно упрощает (следовательно, ускоряет) процесс разработки игры. Одним из таких средств является Unity. Unity-межплатформенная среда разработки компьютерных игр

Основная задача Unity - демократизировать разработку игр. Теперь разработка собственной игры доступна каждому желающему, в том числе студенту и для старта вам не понадобится космическая сумма денег

На протяжении многих лет основное внимание разработчиков Unity было сосредоточено на демократизации разработки игр, чтобы любой желающий мог написать игру и сделать ее доступной самой широкой аудитории. Однако никакой программный пакет не может идеально подходить для всех ситуаций, поэтому вы необходимо знать, когда с успехом можно использовать Unity, а когда лучше поискать другой программный пакет.

Движок Unity особенно хорошо подходит в следующих ситуациях.

- При создании игры для нескольких устройств. Кроссплатформенная поддержка Unity является, пожалуй, лучшей в индустрии, и если необходимо создать игру, действующую на нескольких платформах (или даже на нескольких мобильных платформах), Unity может оказаться лучшим выбором для этого.
- Когда важна скорость разработки. Вы можете потратить месяцы на разработку игрового движка, обладающего всеми необходимыми функциями. Или можете использовать сторонний движок, такой как Unity. Справедливости ради нужно сказать, что существуют другие движки, такие как Unreal или Cocos2D;
- Когда требуется полный набор функций, но нет желания создавать собственный комплект инструментов. Unity обладает идеальными возможностями для создания игр и поддерживает очень простые способы формирования их контента.

При этом в некоторых ситуациях Unity оказывается не так полезен. В том числе:

В играх, не требующих частой перерисовки сцены.

- Unity хуже подходит для реализации игр, не требующих интенсивных операций с графикой, так как движок Unity перерисовывает

каждый кадр. Это необходимо для поддержки анимации в масштабе реального времени, но требует больших затрат энергии.

- Когда требуется точное управление действиями движка.
- Отказавшись от приобретения лицензии на исходный код Unity теряется возможность контролировать поведение движка на низком уровне. Это не значит, что вы вообще теряете контроль над работой движка Unity (в большинстве случаев этого и не требуется), но кое-что окажется недоступно.

Рисунок 3 – Пример игры на Unity

Unreal Engine — это мощный движок для создания игр и других интерактивных приложений, разработанный и поддерживаемый компанией Epic Games. Он используется для разработки игр на различных платформах, включая ПК, консоли, мобильные устройства и VR. Первая версия движка была выпущена в 1998 году, и с тех пор он прошел множество обновлений и улучшений [5].

Рисунок 4 – Пример игры на Unreal Engine

Unreal Engine известен своими возможностями в области графики. Он поддерживает передовые визуальные эффекты, включая трассировку лучей (ray tracing), что позволяет создавать фотореалистичное изображение.

Система blueprints визуального скриптинга, позволяет создавать сложную логику без написания кода. Это делает движок доступным для дизайнеров, не обладающих навыками программирования.

Unreal Engine предоставляет доступ к исходному коду на C++, что позволяет разработчикам глубоко модифицировать движок под свои нужды.

Существует обширный рынок контента, где можно найти множество готовых моделей, анимаций, текстур и других ресурсов для разработки игр.

Встроенные инструменты для работы с анимацией, физикой, искусственным интеллектом и сетевыми возможностями.

Поддерживает множество платформ, от ПК до мобильных устройств и VR.

Unreal Engine Известен своими передовыми графическими возможностями и качеством визуализации. Часто используется для AAA-игр.

Unity: Тоже обладает мощными графическими возможностями, но считается более подходящим для разработки игр средней и малой сложности,

а также 2D-игр. Основной язык Unreal Engine — C++. Также есть визуальный скриптинг (Blueprints). В Unity же, основной язык — C#. Unity предоставляет удобный API для работы с различными аспектами разработки.

Система скриптинга Unreal Engine: Blueprints позволяют создавать логику игры без написания кода, что удобно для дизайнеров и прототипирования. Unity: Использует визуальный скриптинг Bolt, но он менее интегрирован, чем Blueprints в Unreal.

Оба движка поддерживают разработку под множество платформ, но Unity имеет преимущество в мобильных и 2D-играх.

Unreal Engine может быть более сложным для начинающих, особенно если учитывать C++ и мощные графические возможности. Unity считается более дружелюбным для новичков и имеет большую базу учебных материалов.

Unity имеет большое сообщество разработчиков и множество доступных ресурсов, учебников и курсов. В то же время Unreal Engine обладает активным сообществом и обширной библиотекой ресурсов, но количество учебных материалов может быть меньше, чем у Unity.

Выбор между Unreal Engine и Unity зависит от конкретных потребностей проекта. Unreal Engine лучше подходит для проектов, требующих высококачественной графики и сложной логики, тогда как Unity может быть предпочтительнее для мобильных приложений, 2D-игр и проектов с меньшим бюджетом и временем разработки.

2.2 Преимущества и недостатки

Преимущества корпоративной разработки:

- Издатель обычно берет на себя организацию маркетинговых кампаний, что значительно упрощает задачу для разработчиков.
- Издатель помогает с финансированием команды разработчиков, что позволяет сосредоточиться на процессе создания игры, не беспокоясь о денежных вопросах.

- Доступ к передовым технологиям и оборудованию обеспечивает высокое качество конечного продукта.
- Признанный бренд издателя может повысить доверие к игре и увеличить ее популярность.
- Поддержка крупной корпорации обеспечивает финансовую стабильность проекта, снижая риски банкротства.

Недостатки корпоративной разработки:

- Издатель может активно вмешиваться в процесс разработки, что иногда ограничивает креативность команды.
- Разработчикам может не хватать целостного видения проекта из-за раздробленности решений.
- Издатель получает значительную долю прибыли, что снижает доходы самой команды разработчиков.
- Сложная иерархия и необходимость одобрения на нескольких уровнях могут замедлять процесс разработки.
- Жесткие сроки и высокие ожидания часто приводят к стрессу и переработкам.
- Стоимость игры может быть выше из-за необходимости окупить большие затраты на разработку и маркетинг.
- Основной упор делается на коммерческий успех, что может негативно сказаться на оригинальности и качестве игры.

Преимущества инди-разработки:

- Инди-разработчики могут позволить себе экспериментировать и внедрять нестандартные решения, которых нет у крупных компаний.
- Проекты часто отличаются уникальностью и оригинальностью, привлекая аудиторию, ищущую новые впечатления.
- Близкий контакт с игроками позволяет оперативно получать отзывы и улучшать игру.
- Большинство инди-игр создаются из любви к искусству, а не ради прибыли.

- Игры обычно продаются по более доступной цене, что привлекает больше покупателей.
- Инди-разработчики часто избегают тем, вызывающих общественные разногласия, что снижает риск негативной реакции.

Недостатки инди-разработки:

- Высокая конкуренция и ограниченные ресурсы снижают вероятность коммерческого успеха.
- Малочисленные команды ограничены в ресурсах и возможностях, что может сказываться на масштабах и качестве проекта.

2.3 Сравнение различных подходов

Корпоративную разработку и инди разработку можно назвать противоположностями. Корпорация рассматривает игру не как потенциальный шедевр, а как коммерческий продукт.

Идея здесь тоже важна, но она базируется на понимании спроса, конкурентов, перспектив окупаемости — и так далее. Разработчик, таким образом, отталкивается не от собственных желаний, а от рыночных реалий.

У инди всегда есть шанс стать автором новой Minecraft или Loop Hero и разбогатеть. Просто шанс этот невелик. Бизнес же полагается не на удачу, а на логику и экономический анализ. Причем все расчеты делаются до начала разработки, а потом корректируются по ходу — в этом ключевое отличие от инди. По сути, разница в том, что бизнес изучает среду и принимает решение на основе той информации, которую он получил. А инди-подход по отношению к продукту можно сформулировать следующим образом: «Я делаю что хочу, но считаю, что результат должен понравиться всем».

Беда в том, что инди-разработчик за блеском своей идеи зачастую не видит деталей. Все решения он принимает на ходу, ситуативно, и каждое самостоятельно принятое решение считает более важным и верным, чем любое, подсказанное со стороны. Бизнес работает иначе: здесь все решения

принимаются системно и своевременно, исходя из рациональных экономических соображений.

Инди-подход предполагает два пути финансирования. Либо мы заражаем идеей знакомых и тратим свои средства, чтобы довести проект до финала — либо идем с идеей к тем, у кого есть деньги.

Цель инди — воплотить свою идею. Цель бизнеса — построить устойчивый продукт, который сможет окупить себя и заработать. При этом это не противоречащие друг другу понятия. Когда вы встаете на бизнес-рельсы, никто не мешает вам делать проект мечты. Просто это меняет подход к его реализации… и сильно увеличивает шансы на успех. Достаточно лишь трансформировать проект, чтобы он стал более востребованным на рынке.

Издатель вторгается в работу творцов не просто так. Нередко именно вмешательство извне позволяет проекту вообще выйти хотя бы в каком-нибудь виде, потому что внутренний перфекционизм может заставлять шлифовать свою игру и добавлять новые фичи бесконечно. Далеко не все визионеры способны эффективно работать самостоятельно, без контроля сверху.

Корпоративное стремление к контролю не гарантирует игре коммерческий успех. Талантливый дизайнер может создать гениальный продукт в стенах опытного издателя — и этот продукт окажется финансовым провалом.

3. ПРАКТИЧЕСКАЯ ЧАСТЬ

3.1 Примеры и кейсы

Рассмотрим эти 2 типа разработки игр на примере следующих игр:

- BattleBit Remastered
- Battlefield 2042
- Yandere Simulator
- Смута

BattleBit Remastered – это низкополигональный шутер, рассчитанный на огромное количество игроков. Здесь есть 3 режима, в которых одновременно могут находиться 64, 128 или 254 игрока. Благодаря «простой» графике, не нужен мощный игровой компьютер, а сервера нагружаются незначительно, позволяя не сидеть в очереди часами в ожидании эпичного сражения. Игру создают 3 разработчика, которые занимаются проектом уже 6 лет. Актуальная версия носит подзаголовок Remastered, потому что за три года разработки энтузиасты переделали игру почти с нуля.

Рисунок 5 – BattleBit Remastered

Battlefield 2042 — шутер от первого лица, ориентированный на сетевые баталии, в связи с чем не имеет сюжетной кампании. Вместо этого игрокам предоставляется возможность поиграть на сетевых картах в одиночном режиме с друзьями и/или ботами на манер Battlefield 2. История игрового мира будет рассказываться с помощью различных историй разнообразных специалистов, игровых событий, фильмов и тому подобного.

Рисунок 6 – Battlefield 2042

Yandere Simulator — это инди-игра в жанре стелс-экшена. В ней вы принимаете на себя роль Аяно Аиши, девушки-яндере, влюбившейся в парня, которого все называют Сенпаем. Вам предстоит следить за ним и устранять любую девушку, что попытается признаться ему в любви. Причем действовать можно любыми способами: вы можете жестоко расправиться с ней в подворотне или же подставить и добиться её исключения из школы.

Рисунок 7 – Yandere Simulator

Смута - компьютерная игра в жанре action/RPG от третьего лица, разработанная российской игровой студией Cyberia Nova. Выход игры состоялся 4 апреля 2024 года. Действие игры разворачивается в 1612 году, во времена Смуты. Сюжет основан на романе Михаила Загоскина «Юрий Милославский, или Русские в 1612 году»

Рисунок 8 – Смута

3.2 Анализ успешных и неуспешных примеров

Одним из наиболее уместных сравнений корпоративной и инди разработки может быть на примере 2-ух практически одинаковых играх: BattleBit Remastered и Battlefield 2042.

BattleBit Remastered является результатом инди разработки студии SgtOkiDoki. Игра вышла в 2023 году и на данный момент имеет оценку 8.4 от пользователей на сайте Metacritic. All-time peak за все время составлял 86636 человек [11].

Рисунок 9 – Статистика онлайна игры BattleBit Remastered

Battlefield 2042 является результатом корпоративной разработки компании EA. Игра вышла в 2021 году и на данный момент имеет оценку 2.2 от пользователей на сайте Metacritic. All-time peak за все время составлял 107006 человек [11].

Рисунок 10 – Статистика онлайна игры Battlefield 2042

Изначально может показаться что BattleBit Remastered получилась в разы лучше, чем Battlefield 2042, однако это не так. Копия Battlefield стоит в 4 раза дороже чем копия BattleBit. Онлайн игры BattleBit оказался меньше, чем у его соперника Battlefield, хотя если судить по оценкам, в игру никто не должен играть. Как же так получилось?

Первая и самая важная причина такого успеха Battlefield 2042 является бренд издателя. EA выпустила множество финансово успешных игр, таких как Mirror’s Edge, The Sims, Dead Space и т.д.

Второй причиной является сама серия Battlefield. Эта игра является уже 17 частью серии battlefield. Самые первые части серии стали классикой, которые вспоминают с теплотой.

Рассмотрим причины, которые привели к такому рейтингу игры:

Техническое состояние игры. Огромное количество багов, гличей и недостатков сетевого кода сильно подпортили первое впечатление от игры большинству пользователей

Игровые режимы. В battlefield весьма скудный набор игровых локаций. DICE решили ограничиться двумя слегка измененными, но по своей сути старыми режимами, и преподнести их в качестве нового игрового опыта

Монетизация. В игре невероятно огромное количество платного косметического контента.

В основном все негативные причины связаны с «сыростью» проекта. Хорошо видно, как у разработчиков не хватило времени на реализацию всех игровых функций. Сроки – главная проблема корпоративной разработки игр.

С другой стороны, имеем инди проект BattleBit Remastered, у которого также не мало плюсов и минусов. Он построен на движке Unity. Игра представляет себя как альтернатива battlefield 2042. Их нельзя не сравнивать между собой.

Рассмотрим причины такой высокой оценки от игроков.

Во-первых, в игре менее надоедливые предложения о покупке контента, в сравнении с battlefield. Кнопка начала игры не спрятана между баннерами предложений о покупке боевого пропуска. Игроки чувствуют, когда разработчики относятся к ним как к кошелькам.

Во-вторых, стоит отметить низкий порог вхождения. Игра меньше требует системных ресурсов для стабильной работы.

В-третьих, BattleBit remastered хочет казаться той самой классикой, которая при игре вызывает теплые воспоминания [9].

Следующим примером анализа возьмем Yandere Simulator

Yandere Simulator — один из тех проектов, о котором интереснее говорить, чем в него играть

Идея Yandere Simulator принадлежит Алексу Махану. Оригинальный концепт был придуман еще в 2014-ом году и выложен на различных форумах. Получив одобрительные комментарии, Алекс начал разработку первых версий игры, которые он впоследствии и выложил в интернет. В 2015-ом игру заметили крупные блогеры. Тогда YandereDev и решил создать аккаунт на Patreon, чтобы любой желающий мог поддержать его начинания. Это был однозначный успех. В свои лучшие времена, Patreon приносил Алексу свыше пяти тысяч долларов ежемесячно [8].

Крупные студии не могут пойти на такую тему для игры. Для них это огромный репутационный риск. Yandere Simulator могла бы стать идеальной историей про инди-разработчика, добившегося невероятного успеха, но это было бы слишком скучно для такой игры как Yandere Simulator.

Череда случайностей вывела Алекса на олимп инди разработки. Одна только идея игры возмущала общественность, создавая инфоповоды.

Никто не ждал что адекватный билд выйдет быстро, ведь его разрабатывает один человек. Однако нетерпеливость фанатов взяла верх. Алекс охотно принимал новые идеи для своего проекта, а самые интересные, по его мнению, он тут же старался воплотить. По этой причине обновления выходили почти каждый день, добавляя вторичные механики.

Из-за такого подхода к разработке, банальная система сохранений, которая свойственна подобным играм, появится только в 2018 году. Аналогично системе сохранений, в игре долгое время не было главной цели, хотя для подобной игры, она должна быть с самого начала. Только спустя 6 лет в игре появится девушка, которая является главной целью игрока.

За 7 лет в игре нет финала, нет адекватных целей, нет структуры, но зато есть куча мелких второстепенных механик, например, характеры персонажей или система репутации.

Сам Алекс сравнивает свою игру с парком аттракционов, где каждый может найти себе увлечение по душе

Вот что бывает, когда идей, денег и перфекционизма слишком много, а контроля практически нет.

Таких примеров куда больше, чем кажется. Абсолютно аналогичная ситуация сейчас у Star Citizen, только на данный момент репутация Star Citizen в разы больше, чем у Yandere Simulator.

С одной стороны, Yandere Simulator является крайне успешным проектом, ведь он начал приносить огромные деньги, еще до официального выхода. Если абстрагироваться от личности автора, то так оно и есть.

Разработка подобной игры в огромной корпорации невозможна. Никто не станет разрабатывать игру с таким скандальным сюжетом. Yandere Simulator не предназначена для широкой аудитории. Как известно, корпорации стараются угодить абсолютно всем. Очевидно, что это связано с продажами.

Рассмотрим отечественный продукт корпоративной разработки студии Cyberia Nova, а именно Смуту. Бюджет игры составлял 1 млрд руб.

После просмотра дизайн документа создавалось смутное впечатление, что сами разработчики не знают, как делать большие игры, мало в них играют и не понимают, чего они хотят от игры. Серьезный исторический эпос или игру сервис.

Рисунок 11 – Слайд дизайн документа

С получением денег от ИРИ (Институт развития интернета) все кардинально изменилось. Поубавились амбиции, а игра получила более приземленное, реалистичное направление. И это хорошо, ведь лучше изначально определиться как будет выглядеть игра

Однако это не помогло, ведь такой сложной игрой, про невероятно сложный исторический период, занимались люди, которые не делали раньше крупных игр.

Браться за амбициозный проект за такие короткие сроки с новой командой – это брать на себя риски того, что все может накрыться медным тазом просто из-за того, что команда не сработается. Вот эта первая проблема всего данного предприятия – новая команда [13].

Даже если мы набираем самых профессиональных и опытных сотрудников, эффективность взаимодействия их между собой в первый год будет довольно низкая, а денег уйдет на содержание такой команды довольно много.

В этом проявляется еще один недостаток корпоративной разработки – люди приходят и уходят.

Эту проблему подметил Алексей Копцев, глава студии Cyberia Nova. Спустя год разработки, он заявил: «Во-первых, начали активно общаться, помогать друг другу, делиться какими-то ресурсами, рекомендациями, книгами, информацией. Если раньше мы как-то общались на конференциях, но все равно это было какое-то коммерческое противостояние, то теперь мы, как я придумал такую максиму: мы третье ополчение».

Разработчики смуты – компания, которая вела разработку игры по инди принципам. Спустя год разработки, Алексей Копцев, глава Cyberia Nova, заявил, что «разработчики победили панические настроения».

Еще одна причина негодования аудитории – геймплей, а точнее его отсутствие. Игрокам не показывали реального игрового процесса почти до самого выхода игры. Вместо того чтобы дать игрокам то, что успокоит их волнение и вызовет больше доверия к продукту, люди из Cyberia Nova банили людей за вопросы о геймплее.

При разработке смуты больше подошел мы продуктовый подход. Его сутью является наличие готового продукта, который можно отдать пользователю в том или ином виде, на каждом этапе разработки.

4. БУДУЩИЕ НАПРАВЛЕНИЯ И ПЕРСПЕКТИВЫ

В будущем разработка игр будет сильно изменена под воздействием множества передовых технологий и новых подходов. Искусственный интеллект (ИИ) станет ключевым элементом, используемым для процедурной генерации контента, создания игровых миров, а также улучшения NPC, которые станут более реалистичными и реактивными. Уже сейчас появляются инструменты, с помощью которых в играх делают динамичные диалоги с NPC.

Генеративные технологии и нейронные сети будут автоматизировать создание текстур, моделей и анимаций, что значительно ускорит процесс разработки. Кроме того, ИИ сможет динамически развивать сюжетные линии, создавая уникальные игровые сценарии на основе действий игроков.

Развитие графических процессоров и технологий рендеринга приведет к созданию игр с почти фотореалистичной графикой, а улучшенные физические модели и симуляции динамики объектов сделают игровой процесс более реалистичным.

Все эти тенденции тем или иным способом связаны с ускорением разработки элементов игры, используя искусственный интеллект. ИИ будет способствовать созданию более сложных, увлекательных и разнообразных игровых проектов, которые будут радовать игроков по всему миру, предлагая им захватывающий игровой опыт.

Возможно, для улучшения качества игр, необходимо совместить инди и корпоративный подход к разработке. Необходимо разработать такой механизм создания игр, который возьмет в себя лучшее из двух практик. С одной стороны смелость и оригинальность, а с другой стороны финансы и правила. Необходимо чтобы, разрабатывая игру, люди думали о качестве продукта, а не о сроках его сдачи и количестве проданных копий. Однако не стоит забывать про финансовый успех игры, ведь иначе они будут никому не нужны.

ЗАКЛЮЧЕНИЕ

В ходе исследования программной инженерии для игр были выявлены и проанализированы ключевые подходы, инструменты и методы, используемые в современной игровой индустрии. Основные выводы включают, что разработка игр является многоэтапным и комплексным процессом, требующим интеграции различных технических и творческих аспектов. Успешная разработка игр требует глубокого понимания программирования и дизайна пользовательского интерфейса и опыта. Использование передовых технологий, таких как искусственный интеллект и процедурная генерация, может значительно улучшить игровой процесс и взаимодействие с пользователями. Тестирование и оптимизация являются критическими этапами разработки, обеспечивающими стабильность и производительность игры.

Применение гибких методологий, таких как Agile или Scrum, может значительно улучшить управление проектами, позволяя быстро адаптироваться к изменениям и обеспечивать высокое качество конечного продукта. Инвестиции в обучение и развитие помогут командам оставаться конкурентоспособными и эффективными, а регулярное тестирование на всех этапах разработки и постоянная оптимизация кода и ресурсов помогут предотвратить проблемы на поздних стадиях и улучшить производительность игры.

Изучение применения искусственного интеллекта и машинного обучения в разработке игр, особенно в создании умных NPC и процедурной генерации контента, может открыть новые возможности для индустрии. Разработка новых методов и инструментов для создания более погруженных и интерактивных игр в виртуальной и дополненной реальности требует дальнейших исследований и экспериментов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сахнов К. Семь этапов создания игры: от концепта до релиза / Сахнов К. [Электронный ресурс] // Хабр : [сайт]. — URL: https://habr.com/ru/companies/miip/articles/308286/ (дата обращения: 22.06.2024).

2. John P. Flynt Software Engineering for Game Developers [Текст] / John P. Flynt — 1-е изд.. — : , 2005 — 890 с.

3. Luiz F. C. Game development software engineering process life cycle: a systematic review / Luiz F. C. [Электронный ресурс] // SpringerOpen : [сайт]. — URL: https://jserd.springeropen.com/articles/10.1186/s40411-016-0032-7 (дата обращения: 22.06.2024).

4. Understanding the Role of a Video Game Software Engineer / [Электронный ресурс] // INSTITUTE OF DATA : [сайт]. — URL: https://www.institutedata.com/blog/role-of-video-game-software-engineer/ (дата обращения: 22.06.2024).

5. Что такое программирование игр и как стать игровым программистом? / [Электронный ресурс] // Хабр : [сайт]. — URL: https://habr.com/ru/articles/819723/ (дата обращения: 22.06.2024).

6. Липаев, В. В. ПРОГРАММНАЯ ИНЖЕНЕРИЯ СЛОЖНЫХ ЗАКАЗНЫХ ПРОГРАММНЫХ ПРОДУКТОВ [Текст] / В. В. Липаев — 1-е изд.. — Москва: Макс Пресс, 2014 — 311 с.

7. Зеленко Л. С. Программная инженерия [Текст] / Зеленко Л. С. — 1-е изд.. — Самара: Макс Пресс, 2012 — 263 с.

8. YandereDev. Умри героем или живи до тех пор, пока не станешь негодяем / [Электронный ресурс] // DTF : [сайт]. — URL: https://dtf.ru/gameindustry/191317-yanderedev-umri-geroem-ili-zhivi-do-teh-por-poka-ne-stanesh-negodyaem (дата обращения: 22.06.2024).

9. Онлайн-шутер BattleBit Remastered — «убийца Battlefield» или временный хайп? / [Электронный ресурс] // VGTimes : [сайт]. — URL:

https://vgtimes.ru/articles/98619-onlayn-shuter-battlebit-remastered-ubiyca-battlefield-ili-vremennyy-hayp.html (дата обращения: 22.06.2024).

10. Инди-игры: как маленькие студии завоевывают мир вопреки? / [Электронный ресурс] // IXBT : [сайт]. — URL: https://www.ixbt.com/live/games/indi-igry-kak-malenkie-studii-zavoevyvayut-mir-vopreki.html (дата обращения: 22.06.2024).

11. An ongoing analysis of Steam's concurrent players. / [Электронный ресурс] // steamcharts : [сайт]. — URL: https://steamcharts.com/ (дата обращения: 22.06.2024).

12. Соколов, Т. А. Базовая концепция жизненного цикла разработки компьютерных игр (GDLC) / Т. А. Соколов. — Текст: непосредственный // Молодой ученый. — 2023. — № 6 (453). — с. 11–14. — URL: https://moluch.ru/archive/453/99965/ (дата обращения: 22.06.2024).

13. В чем на самом деле проблемы игры «Смута»? / [Электронный ресурс] // Хабр : [сайт]. — URL: https://habr.com/ru/articles/810863/ (дата обращения: 22.06.2024).

14. The top 10 game development issues of 2022. — Текст: электронный // Gamasutra: [сайт]. — URL: https://www.gamasutra.com/view/news/287546/The_top_10_game_development_issues_of_2022.php (дата обращения: 22.06.2024).

15. Тимофеев, М. Д. Проблемы организации работы и коммуникации в сфере разработки видеоигр //Аналитические технологии в социальной сфере: теория и практика. — 2020. — с. 76.

16. Brathwaite B. Challenges for Game Developers / Brathwaite B. [Электронный ресурс] // Хабр : [сайт]. — URL: https://research.tedneward.com/reading/gamedev/challenges-for-game-developers/index.html (дата обращения: 22.06.2024).

17. 5 Common Game Development Challenges & Solutions. — Текст: электронный // GameAnalytics: [сайт]. — URL:

https://gameanalytics.com/blog/common-game-development-challenges-solutions.html (дата обращения: 22.06.2024).

18. Чистовская, О. В. Влияние классификации видеоигры на ее нарратив / О. В. Чистовская // Молодой ученый. — 2021. — № 17(359). — с. 11–14. — EDN DYRWHS.

19. Данилюк, М. Д., Шпаковский Ю. Ф. Разработка видеоигр: проблемы современных исследований //Труды БГТУ. Серия 4: Принт-и медиатехнологии. — 2017. — №. 4 (195). — с. 118–122.

20. Ветеранова, Д. С. Основные этапы разработки видеоигр //Информационно-компьютерные технологии в экономике, образовании и социальной сфере. — 2019. — №. 2. — с. 51–56.

21. Сахнов К Идея без плана — ничто? / Сахнов К [Электронный ресурс] // Хабр : [сайт]. — URL: https://habr.com/ru/articles/734978/ (дата обращения: 22.06.2024).