КАЗАХСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ ИМЕНИ АЛЬ-ФАРАБИ

Информация о документе:

Автор: Гаврилин Илья Владимирович
Имя документа: Дипломный проект

Данные пользователя:

Факультет	Информационных технологий	**Ступень обучения**	бакалавриат
Шифр специальности:	6B06102	**Форма обучения**	дневная
Специальность	Информационные системы	**Уровень обучения**	Первое высшее сокращенное образование
Курс обучения	3	**Отделение**	русское

Источник	Ссылка на источник	Коллекция/модуль поиска	Доля в отчёте	Доля в тексте
[1] 2023 ИЭИТУС 09.03.02...		crosslang_apring	2,09 %	2,09 %
[2] Разработка браузерно...		crosslang_apring	1,87 %	1,87 %
[3] sbornik_nauch_statey...	https://imsit.ru/wp-content/uploads/2024...	crosslang_internet_ru_2022	0,8 %	0,8 %
[4] Анализ рентгенограмм...		crosslang_apring	0 %	0,61 %
[5] 13 бесплатных програ...	https://GameCreating.ru/besplatnye-progr...	bundle_internet	0 %	0,35 %
[6] 13 бесплатных програ...	https://gamecreating.ru/besplatnye-progr...	bundle_internet	0 %	0,35 %
[7] Creating an interact...	https://dspace.spbu.ru/handle/11701/3554	crosslang_internet_ru_2022	0,34 %	0,34 %
[8] Право цифровой среды...	http://ivo.garant.ru/#/document/76880033	crosslang_garant_analytics_2022	0 %	0,31 %
[9] VKR_TurpanovAA_HTML5...		crosslang_apring	0 %	0,29 %
[10] vkr_baceko_r.v_isit....	https://lpi.sfu-kras.ru/files/vkr_baceko...	paraphrase_internet_ru_2022	0 %	0,24 %
[11] Источник 11		stockphrase	0 %	0,22 %
[12] Учебная практика. 2 ...		apring	0 %	0,21 %
[13] 20+ лучших инструмен...	https://freelance.today/poleznoe/20-luch...	bundle_internet	0 %	0,21 %
[14] 20+ лучших инструмен...	https://freelance.today/poleznoe/20-luch...	bundle_internet	0 %	0,21 %
[15] 20+ лучших инструмен...	https://freelance.today/poleznoe/20-luch...	bundle_internet	0 %	0,21 %
[16] Дешево и сердито: ге...	https://habr.com/ru/companies/selectel/a...	bundle_internet	0 %	0,2 %
[17] 13 бесплатных програ...	https://gamecreating.ru/besplatnye-progr...	bundle_internet	0 %	0,2 %
[18] Что такое питон джан...	https://alfacasting.ru/faq/cto-takoe-pit...	bundle_internet	0 %	0,19 %
[19] Фэнтези. Большая рос...	https://bigenc.ru/c/fentezi-06cb07	bundle_internet	0 %	0,13 %
[20] Современные тенденци...	https://elibrary.ru/item.asp?id=47971530	Elibrary	0 %	0,12 %
[21] Графический дизайнер...	https://blog.vill-institute.com/kak-stat...	bundle_internet	0 %	0,12 %
[22] ВКР_БИВТ-19-4_Бузин		apring	0 %	0,11 %
[23] Источник 23		citations	0 %	0,09 %
[24] Исчерпывающее руково...	https://xvpn.io/ru/blog/wireguard	bundle_internet	0 %	0,09 %

Итоговая оценка оригинальности: **94,89%**
Общий процент заимствования: **5,11%**
Из них процент легального цитирования: **0%**
Проверить: Перейти на страницу системы "Антиплагиат"

Введение В свете быстрого развития технологий и влияния цифровой эры видеоигры не только стали одной из наиболее прибыльных отраслей развлекательной индустрии, но и открыли перед разработчиками уникальные возможности воплотить свои творческие идеи. Современная видеоигровая индустрия выделяется не только высокими технологическими стандартами, но и непрерывным стремлением к творчеству. Миллионы талантливых художников, программистов и дизайнеров по всему миру работают над созданием уникальных виртуальных миров. С точки зрения финансов, видеоигры привлекают инвестиции и генерируют значительную прибыль, делая индустрию соперником традиционных развлечений, таких как кино. Искусственный интеллект становится неотъемлемой частью современных видеоигр, революционизируя взаимодействие между игроками и виртуальным миром. Алгоритмы машинного обучения применяются для создания интеллектуальных персонажей, динамичного изменения сценария и поддержания интересного геймплея. Целью дипломной работы является создание игрового приложения с использованием искусственного интеллекта; попытка создания новой тенденции в игровой индустрии прямого взаимодействия пользователя с AI моделью, а также анализ готовности рынка и технологий к данным видам проектов. Ключевой особенностью проекта является использование искусственного интеллекта для создания персонажей, эмулирующих поведение живых существ. Модель AI, обладая знаниями о вселенной игры, предоставляет игроку возможность взаимодействия через внутриигровой чат, создавая неповторимый опыт. Технические характеристики проекта: рассмотрим тщательно продуманный концепт проекта, с учетом технических деталей, таких как трассировка PixelArt и использование темной фантастической эстетики. Это не только подчеркивает визуальную привлекательность игры, но и обеспечивает соответствие жанровому контексту. Сюжет и мир игры: проект предлагает уникальный гибрид RPG, экшена, платформера и хоррора, что позволяет широко охватить аудиторию игроков. Сюжет, пропитанный темной фантастикой, создает увлекательное и многогранное повествование, поддерживаемое частично открытым миром и блочными уровнями. Для

реализации данного проекта необходимо решить следующие задачи: – рассмотреть динамику мира видеоигр как с творческой, так и финансовой стороны, – изучить основные аспекты разработки игр с использованием искусственного интеллекта (AI)[1]

. 3 1 ВЛИЯНИЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА (AI) НА ИГРОВУЮ ИНДУСТРИЮ 1.1 Анализ рынка видеоигр Согласно последним статистическим данным, количество активных игроков в мире достигло нескольких миллиардов человек. Этот огромный пул потенциальных пользователей создаёт благоприятную среду для разработчиков, стимулируя конкуренцию и инновации. Рассмотрим актуальные тенденции в мире жанров. Экшен, RPG и платформеры остаются в числе лидеров, но также возникают новые, менее исследованные направления, такие как симуляторы выживания и кроссплатформенные многопользовательские игры, предоставляя разработчикам широкие возможности для экспериментов.

Рисунок 1.1. Объем мирового рынка игр за 2023год 1.2 Вклад искусственного интеллекта в игровую индустрию Развитие технологий AI Искусственный интеллект вне[1]

с огромный вклад в индустрию, повышая степень реализма ведения боя, динамичность сценариев и реакцию персонажей на действия игрока. Это не только улучшает качество игрового опыта, но и позволяет разработчикам более эффективно использовать ресурсы. Ускорение процесса разработки Интеграция искусственного интеллекта в разработку игр существенно ускоряет процессы создания. Алгоритмы машинного обучения позволяют 4 автоматизировать создание персонажей, настройку баланса игры и оптимизацию ресурсов, снижая трудозатраты и время разработки. Влияние на финансовые показатели Внедрение AI в игровую индустрию сказывается на финансовых показателях, обеспечивая увеличение прибыли благодаря повышению интереса со стороны игроков, улучшению качества продукта и более эффективному масштабированию. Похожие проекты и Инструменты AI в игровой разработке Похожие проекты Проведём анализ существующих игр, использующих искусственный интеллект в качестве ключевой игровой механики. 1. "Detroit: Become Human"[1] (Квантическая мечта, 2018): Этот проект предлагает игрокам управлять персонажами-андроидами, обладающими высокой степенью искусственного интеллекта. Решения игрока влияют на ход сюжета и развитие персонажей. 2. "Alien: Isolation"[2] (Creative Assembly, 2014): В этом хорроре искусственный интеллект используется для создания инопланетных хищников, адаптирующих свою стратегию под действия игрока, что поддерживает напряженность и страх. 3. "Middle-earth: Shadow of Mordor"[3] (Monolith Productions, 2014): Игра использует технологию Nemesis,

основанную на искусственном интеллекте, чтобы создать динамичных врагов, которые запоминают действия игрока и адаптируются к его стратегии. Инструменты на основе AI в разработке игр 1. Unity ML-Agents: программа с открытым исходным кодом разработанная Unity Technologies, при помощи которой можно интегрировать машинное обучение и искусственный интеллект в игровые проекты[3]

. 2. TensorFlow for Games: была создана Google, предоставляет мощный инструмент для создания моделей на основе AI в играх, позволяя разработчикам создавать персонажей с умным поведением. 3. GANs (Generative Adversarial Networks) в графическом дизайне: Служит для создания текстур, анимации и уровней, основываясь на принципе противостояния между двумя нейронными сетями. Состояние проекта относительно похожих В настоящее время существует огромное количество игр, в которых AI модели служат ключевым элементом игровой механики. Однако, проект с уникальной концепцией, подобной этому, выделяется среди большинства игр на рынке. Примеры успешных проектов с использованием AI 1. "AlphaGo"[4] (DeepMind, 2016): Этот проект, созданный на основе искусственного интеллекта, демонстрирует возможности машинного 5 обучения в стратегических играх, подняв стандарты в области искусственного интеллекта. 2. "OpenAI Five"[5] (OpenAI, 2018): Команда искусственного интеллекта OpenAI создала ботов, способных соревноваться с профессиональными игроками в Dota 2[6], что стало прорывом в области многопользовательских онлайн-игр. 3. "IBM Watson" в создании "Cognitive Cooking with Chef Watson"[7]: Хотя это не игра, использование искусственного интеллекта в кулинарии, создание уникальных рецептов, иллюстрирует разнообразие возможностей применения AI.

1.3 Godot и язык программирования C# Godot Engine Godot Engine - это бесплатный, открытый исходный код игровой движок, разработанный и поддерживаемый сообществом разработчиков[2]

. Он предоставляет обширный инструментарий для создания 2D, 3D и даже мобильных игр. Одной из ключевых особенностей Godot является его легковесность, гибкость и поддержка разнообразных платформ. Основные характеристики Godot: 1. Многоплатформенность:

Godot поддерживает развертывание игр на различных платформах, включая Windows, macOS, Linux, Android, iOS и веб- браузеры. 2. Языковая поддержка: Godot предоставляет свой собственный язык программирования GDScript[2]

, который оптимизирован для использования в игровой разработке. Кроме того, Godot поддерживает языки программирования C# и VisualScript.

3. Редактор сцен: Интуитивный и мощный редактор сцен в Godot упрощает процесс создания и управления элементами игры. 4. Многие встроенные инструменты: Godot включает в себя различные инструменты, такие как редактор анимаций, редактор партиклов[2]

, инструменты для работы с физикой и многое другое.

5. Сообщество: Godot активное сообщество разработчиков, которые делают документацию, обновляют код и делают туториалы, давая поддержку для новичков и опытных разработчиков[3]

. Язык программирования C# Godot поддерживает пару языков программирования: GDScript (язык, разработанный для Godot), VisualScript (визуальное программирование) и C#. C# в Godot: 1. Удобство использования: C# - это мощный и простой в освоении язык программирования, что делает его привлекательным для разработчиков различного уровня опыта. 6 2. Экосистема .NET: Использование C# в Godot открывает доступ к обширной экосистеме .NET, это поможет при интеграции сторонних библиотек и решений. 3. Кроссплатформенность: Также, как и Godot, C# обеспечивает кроссплатформенную совместимость, что позволяет разрабатывать игры для различных устройств. Проекты, созданные на Godot 1. "Hyper Light Drifter"[8]: Это изометрическая экшн-игра, которая получила высокие оценки за свой арт-дизайн и геймплей. Разработана серийным независимым разработчиком Alex Preston. 2. "One Bit Adventure"[9]: Эта ретро-пиксельная приключенческая игра, разработанная Just1337 Studio, предоставляет игрокам захватывающий опыт в стиле старых RPG. 3. "Hearts of Strangers"[10]: Это 2D-платформер, созданный студией Maybe Later Games. Игра впечатляет оригинальным искусством и увлекательным геймплеем. 4. "MoonQuest"[11]: Интересная песочница с процедурно генерируемыми мирами, в которой игроки исследуют и создают свой путь. Разработана benjamin_soule. 5. "Godot Engine Demo Projects"[12]: В самом Godot есть различные демо-проекты, которые помогают новичкам освоить инструментарий и начать работу над своими собственными идеями. Godot Engine, с поддержкой языка программирования C#, предоставляет разработчикам мощный и гибкий инструментарий для создания

разнообразных игр. Проекты, созданные на Godot, демонстрируют его способность к обработке различных жанров и стилей, делая его хорошим выбором для широкого спектра разработчиков. Godot Engine vs. Unity Преимущества Godot: Бесплатный и открытый код: Godot является полностью бесплатным и открытым для. В отличие от Unity, который требует платить за лицензии для определенных условий. Интегрированный язык GDScript: GDScript разработанный для Godot, предоставляет простой, но мощный способ программирования. Это уменьшает издержки на изучение и разработку. Кроссплатформенность: Godot обеспечивает поддержку различных платформ, включая Windows, macOS, Linux, Android, iOS и веб-браузеры. Легковесность и гибкость: Godot имеет небольшой объем, что делает его отличным выбором для разработки легких и быстрых игр. Гибкость движка позволяет легко изменять и дополнять функционал. Недостатки Godot сравненивая с Unity: Меньшее сообществ: В сравнении с Unity, Godot имеет меньшее разработчиков и меньший объем расширений. 7 Меньшая документация: Документация Godot, менее подробна и структурирована, чем у Unity. Это может повлиять на процесс изучения. Godot Engine vs. Unreal Engine Преимущества Godot: Легковесность и простота использования: в Godot более просой интерфейс и процесс разработки, что делает его подходящим для небольших и средних проектов. Бесплатность: Godot предоставляет открытый исходный код, что обеспечивает большую свободу для разработчиков. Недостатки Godot по сравнению с Unreal Engine: Менее продвинутая графика: Unreal Engine славится своей графикой и поддержкой фотореалистичных 3D-проектов, чего можно не достичь с использованием Godot. Меньше возможностей для разработки 3D-проектов: Unreal Engine предоставляет более широкий набор инструментов и возможностей для создания высококачественных 3D-игр. 1.4 Жанры игр Dark Fantasy (Тёмное Фэнтези): Определение: Тёмное фэнтези - это поджанр фэнтези, который характеризуется более мрачной, готической и жесткой атмосферой по сравнению с традиционным высоким фэнтези. Этот жанр часто переносит читателей или зрителей в миры, наполненные магией, монстрами и темными тайнами. Основные черты: Суровый мир: Миры тёмного фэнтези часто известны своей суровостью и неприветливостью. Здесь доминируют темные леса, тлетворные болота и мрачные замки. Магия и Тёмные Силы: Магия в тёмном фэнтези часто представлена как опасная и таинственная сила, порой вызывающая непредсказуемые последствия. Тёмные силы, демоны и некромантия также являются распространенными элементами. Сложные персонажи: Персонажи в тёмном фэнтези часто имеют моральные дилеммы, внутренние борьбы и темные прошлые. Герои не всегда являются безупречными, а злодеи могут обладать человеческими чертами. Темные Темы: Жанр часто касается темных и трудных тем, таких как смерть, предательство, безнадёжность и человеческие страхи. Готический Стиль: Архитектура, одежда и обстановка в тёмном фэнтези могут быть сильно вдохновлены готическим искусством и эстетикой. Примеры Тёмного Фэнтези в литературе и кино: 8 1. "Песнь Льда и Пламени" (A Song of Ice and Fire) - Джордж Р. Р. Мартин: Эпическая серия книг, на которой основан сериал "Игра Престолов", является ярким примером тёмного фэнтези. 2. "Властелин Колец" (The Lord of the Rings) - Дж. Р. Р. Толкин: Хотя это высокое фэнтези, оно также содержит элементы тёмного фэнтези, особенно в мире Мордора и взаимодействии с Кольцом Всевластия. 3. "Берсерк" (Berserk) - Кентаро Миура: Манга и аниме "Берсерк" известны своей тёмной и насильственной атмосферой, а также глубоким изучением человеческой природы. 4. "Ведьмак" (The Witcher) - Анджей Сапковский: Серия книг, адаптированная в успешные видеоигры и сериал, представляет собой мрачное и сложное тёмное фэнтези. 5.

"Dark Souls" - FromSoftware: Эта видеоигра прекрасно иллюстрирует элементы тёмного фэнтези в игровой индустрии, смешивая сложный геймплей с атмосферным и мистическим миром[1]

. Тёмное фэнтези продолжает привлекать поклонников своей атмосферой, сложными сюжетами и глубоким изучением человеческой природы в условиях тьмы и магии. Ролевая Игра (RPG): Ролевая игра (RPG) - жанр, в котором

игрок управляет персонажем или группой персонажей, развивая их через выполнение заданий, сражения с врагами и принятие важных решений, влияющих на ход сюжета. Механики: Система Уровней и Опыта: Персонажи могут повышать свой уровень, получая опыт за выполнение задач и уничтожение врагов. Каждый новый уровень приносит улучшения в характеристиках и доступ к новым навыкам. Инвентаризация и Экипировка: Игроки могут собирать, покупать и использовать различные предметы, такие как оружие, броня и зелья[1]

, для улучшения своих персонажей. Ветвление Сюжета: Решения, принятые игроком, могут влиять на ход сюжета и развитие персонажей, создавая индивидуальные повествовательные линии. Боевая Система: Боевая механика включает в себя тактические решения, виртуальные бои в реальном времени или пошаговые сражения. Некоторые RPG предлагают комбинированные подходы. Экшен: Экшен - жанр, в котором акцент сделан на физической активности, быстрых реакциях и борьбе. Включает в себя сражения, стрельбу, прыжки и другие действия, требующие ловкости и умения. Механики: Боевая Динамика: Основная механика — это активное участие игрока в сражениях, где важны не только стратегия, но и реакции. 9 Платформинг (Прыжки и Перемещения): Экшен-игры могут включать элементы платформинга, где игрок преодолевает препятствия, прыгает с платформы на платформу и выполняет другие акробатические действия. Стрельба и Ближний Бой: В зависимости от поджанра, игрок может использовать огнестрельное оружие, мечи, кулаки или другие средства для атаки противников. Скриптованные Сцены и Кинематография: Многие экшен-игры предлагают спектакульные сцены и кинематографические элементы, чтобы подчеркнуть динамичность игрового процесса. Платформер: Платформер - жанр, фокусирующийся на преодолении платформенных уровней, выполнении прыжков и избегании препятствий. Механики: Прыжки и Перемещения: Основной элемент - способность персонажа прыгать, двигаться по платформам и преодолевать пространственные препятствия. Рефлексы и Тайминг: Точные прыжки и правильный тайминг являются ключевыми механиками, особенно в сложных уровнях. Решение Головоломок: Некоторые платформеры включают головоломки, требующие логического мышления и использования окружающей среды для преодоления. Скорость и Темп Игры: Многие платформеры стремятся создать динамичный темп игры, где игроки должны быстро реагировать на изменяющуюся обстановку. Хоррор: Хоррор - жанр, цель которого вызвать у игрока чувство страха, тревоги или напряжения. Это может быть достигнуто с использованием ужасающей атмосферы, устрашающих существ или пугающих событий. Механики: Ограниченные Ресурсы: Часто хорроные игры ограничивают количество доступных ресурсов, что создает ощущение беспомощности перед угрозой. Атмосферная Графика и Звук: Хоррор обычно использует темную и мрачную графику, а также звуковые эффекты для создания мрачной атмосферы. Игра Скрытности: Механика скрытности может быть важной, позволяя игрокам избегать врагов и оставаться в тени. Неожиданные События и Повороты Сюжета: Хоррорные игры обычно стремятся шокировать игроков неожиданными событиями и переворотами сюжета. Исследование и Разгадывание Головоломок: Введение элементов исследования и разгадывания головоломок может поддерживать напряженную атмосферу. 10 Каждый из этих жанров предоставляет уникальные механики и опыт игры, а разнообразие их комбинаций часто приводит к созданию увлекательных и разнообразных проектов в мире видеоигр. 11 2 КОНЦЕПТ ИГРЫ 2.1 Сюжет и мир игры Сюжет: Осколок души, призванный для возражения божества вернувшего его к жизни. Персонаж не имеет, чёткого представления о своём прошлом(жизни до смерти). В ходе игры необходимо найти способ воскрешения божества, параллельно данным событиям необходимо собрать все осколки души в ключевых местах прошлого, что будет восстанавливать часть воспоминаний. В ходе игры действия персонажа влияют, на его внешний вид и способности, останется ли персонаж человеком или станет подобным богам. Мир: В 2050 когда население земли представляло, чуть более 10 миллионов человек, а технологии шагнули, так далеко, что человечество, уже могло изучать ближайшие солнечные системы, в ходе войны было уничтожено 60% населения, выжившее же поделились на отдельные общины. Большинство из них занимались борьбой за оставшиеся ресурсы, многие в ходе этой борьбы были заражены мутацией, что деформировало их тела или привело к смерти. Через 300 лет в ходе сюжета игры, уже сформировались новые города, уровень жизни в которых напрямую зависел от главенствующей стороны. В ходе эволюцию появились новые разновидности людей, мир заполонили опасные мутанты, из-за которых перемещение между городами стали затруднительны. Так же появились довольно редкие мутанты(аномалии), способные обладающие особыми способностями. Этих существ из-за их силы, религиозные фанатики прозвали богами, потом название перешло в общий речевой оборот. Они обладали различным интеллектом от схожих с зверьми, до превышающих человеческое понимание. Многие из них привели к массовому геноциду среди человеческого вида, а также природы и фауны. Технические характеристики: Устройство: ПК Размер экрана: Полноразмерный Дизайн: PixelArt, Deep dark fantasy Реализация дизайна: Трассировка Стиль: Казуальный реализм Форма: Квадратные угловые формы Цветовая палитра: тёмные фиолетовые / тёмные зелёные / тёмные красные оттенки (в зависимости от локации) Атмосфера: Загадочная, таинственная Целевая аудитория: 14+ Жанр: RPG, экшен, платформер, хоррор Камера: Вид от третьего лица с фиксированным углом съёмки Геймплей: Быстрый, имеются элементы платформера и головоломок Тип уровней: Блочный Тип мира: Частично открытый 12 Музыкальная стилистика: Электронная, спокойная Среднее время прохождения: 20ч Среда распространения: Steam 2.2 Игровой интерфейс Главное меню: Продолжить(с места последнего сохранения) Начать новую игру Настройки Титры Выход Внутриигровой интерфейс: HP bar (Здоровье персонажа) SP bar (Выносливость персонажа) MP bar (Количество энергии, для проведения способностей) Чат Мини-карта Дополнительный интерфейс: Панель распределения характеристик Карта Инвентарь Логи Меню паузы: Продолжить(Выход из меню паузы) Настройки Выход в главное меню OST Example: 1. [LobotomyCorporation OST] - First Warning[13] 2. [LobotomyCorporation OST] - Second Warning[14] 3. [LobotomyCorporation OST] - Third Warning[15] 4. 下剋ノ上 feat.初音ミク / Supplanting his lord feat.Hatsune Miku[16] 5. Dino Crisis Ost 6 - The place is deserted throgh[17] 6. Killing

Time[18] 7. Further Down[19] 8. Vinland Saga OST - Relaxing Medley[20] 9. Hollow Knight OST - City of Tears[21] 10. The Grimm Troupe (Hollow Knight: Gods & Nightmares) [22] 11. Dreamers (Hollow Knight: Gods & Nightmares) [23] 12. Bonebottom (Silksong OST Sample) [24] 13. Chainsaw Man OST - 03 - Imagine Devils[25] 14. Chainsaw Man OST - 04 - The Devil Hunter[26] 13 15. Chainsaw Man OST - 02 - The Door[27] 16.『チェンソーマン』第3話ノンクレジットエンディング / CHAINSAW MAN #3 Ending｜マキシマム ザ ホルモン 「刃渡り2億センチ」[28] 2.3 Концепты персонажей (Сгенерировано Bing Image Creator) Пометка: Изображения выше только содержат набор стандартного цветового диапазона, примерно формы объектов и не отражает конкретного графического стиля. Главный герой: Рисунок 2.1 -

Концепт арты главного героя Поведение: Обычно действия главного героя, зависят от игрока и он сам не умеет широко мыслить, по сравнению с другими игровыми персонажами Группировка: Главный герой Редкость: Уникальный Локация: Отсутствует Описание: Управляемый персонаж Военные(Специальны[2]

й отряд): 14 Рисунок 2.2. Концепт арт персонажей военной фракции Поведение: По одиночке не составляют опасности, сильные в большом количестве. Группировка: Военные Редкость: Обычная Локация: Любая Описание: Основное предназначение регулирование порядка, часто можно встретить в оживлённых локациях, вне города можно встретить не большими скоплениями по 5-10 человек. Используют импланты из-за чего могут пережить несколько прямых попаданий. Охотник: 15 Рисунок 2.3. Концепт арт специального солдата военных Поведение: Быстрое перемещение и атаки Группировка: Военные Редкость: Обычная Локация: Любая Описание: Особая боевая единица, обычно перемещаются по одиночке. Служат для выполнения заданий вне городов или для быстрого устранения цели. Танк: 16 Рисунок 2.4 - Концепт арт специального солдата военных Поведение: медленное перемещение, акцентирует внимание на защите. В бою старается вывести врага из укрытия Группировка: Военные Редкость: Обычная Локация: Любая Описание: Особая боевая единица, обычно перемещаются с группой обычных солдат. Служат для подавления врага. Бандиты: Рисунок 2.5 - Концепт арт группировки бандиты Поведение: По одиночке не составляют опасности, сильные в большом количестве. Группировка: Бандиты Редкость: Обычная Локация: Лес, деревня 17 Описание: Зарабатывают при помощи грабежей. Встречаются не агрессивные особи. Могут встречаться, как по одиночке, так и большими скоплениями. Сюжетный персонаж 1: Рисунок 2.6 - Концепт арт торговец чёрного рынка Поведение: Не агрессивный. Продаёт предметы Группировка: Торговцы Редкость: Уникальный Локация: Деревня Описание: Предлагает уникальные задания, после которых пополняет ассортимент. Позволяет запастись припасами и расходниками. Участвует в сюжете. Хаос: Рисунок 2.7. Концепт арт бога хауса и беспорядка 18 Поведение: Ведёт диалог с игроком, наставляет на действия приводящие к хаотичным последствиям Группировка: Боги Редкость: Уникальный Локация: Отсутствует Описание: Постоянно находиться в голове персонажа, помогает в случае трудностей. Большинство времени находиться в обличие светящегося шара, может освещать помещение. Предписание: Является AI моделью, реагирующей на действия игрока Способности: Может влиять на шанс выпадения лута, силу соперников и использование фильтров, управление главным персонажем. Монстр 1: Рисунок 2.8 - Концепт арт мутант 1 Поведение: Агрессивное. Быстрые, слабые удары. Группировка: Мутанты Редкость: Обычный Локация: Проклятый лес Описание: Во времена противостояния, двух ведущих стран, был членом элитных войск, имеющих на время своей жизни продвинутые импланты. Спустя много лет был воскрешён, для участия в войне богов. Появляются редко по одиночке, но представляют большую опасность для игрока. Монстр 2: 19 Рисунок 2.9. Концепт арт мутант 2 Поведение: Агрессивное. Быстрые, сильные удары, охотиться ночью. Группировка: Мутанты Редкость: Редкий Локация: Глухой забой Описание: Служат Марене. Во времена войны богов, служили для быстрого устранения особо важных целей. Монстр 3: Рисунок 2.10 - Концепт арт мутант 3 20 Поведение: Агрессивное. Дальняя быстрая, слабая атака. Ближняя сильная, медленная. Группировка: Мутанты Редкость: Редкий Локация: Болота белого хлада Описание: Является боссом локации болта белого хлада. Является хранителем древней церкви. До потери человечности, был служителем данной церкви. Монстр 4: Рисунок 2.11 - Концепт арт мутант 4 Поведение: Агрессивное. Медленные сильные атаки. Группировка: Мутанты Редкость: Редкий Локация: Болота белого хлада Описание: Дерево, поглотившее человека, утопленного в болотах. Постоянно голоден. С пожиранием тел, становиться сильнее. Агрессивно настроен к схожим особям. Аномалия 1: 21 Рисунок 2.12 - Концепт арт повелительница мира мёртвых (Хельхейма) Поведение: Агрессивна по отношению к игроку, только в моменты сюжета. Группировка: Боги Редкость: Уникальный Локация: Лес Описание: Должна найти и убить бога хауса. Является одним из опаснейших врагов. Может менять реальность. 2.4 Концепты локаций (Сгенерировано Bing Image Creator) Пример леса: Рисунок 2.13 - Концепт арт локация лес 1 22 Рисунок 2.14 - Концепт арт локация лес 2 Описание локации: Служит связующим звеном, между точками интереса Пример города: Рисунок 2.15 - Концепт арт локация деревня греха 23 Рисунок 2.16 - Концепт арт локация руины старого мира Описание локации: Служит местом встречи многих персонажей, является большой частью внутриигровой активности 24 3 РАЗРАБОТКА ИГРОВОГО ПРИЛОЖЕНИЯ Разработка началась с создания Логлайна. Логлайн – это короткое и емкое описание сюжета или концепции проекта, которое предназначено для привлечения внимания аудитории и краткого представления основной идеи произведения. Он играет важную роль в маркетинге и продвижении игры, поскольку помогает потенциальным игрокам быстро понять суть проекта и заинтересоваться им. Логлайн: "Вернуться из мертвых, чтобы покончить с богами: бывший репортёр становится ключом к изменению безумного мира, который кишит порождениями богов, после того как он был возрождён могущественным существом, обещавшим ему новую жизнь в обмен на помощь в своём возрождении." В данном логлайне подчеркивается важность главного героя и его уникальной судьбы: от обычного репортёра до могущественного существа, которому предстоит сражаться с богами и изменить мир. Сюжетный поворот, связанный с возрождением и новой жизнью, добавляет интриги и непредсказуемости. Таким образом, логлайн представляет центральную тему игры - борьбу за новую жизнь, силу и справедливость, захватывая внимание аудитории и предвкушая захватывающие приключения в игре.

3.1 Инструменты, используемые в разработке игры Программа Krita для создания анимации и рисования текстур В мире развивающейся игровой индустрии, где визуальное восприятие игры играет ключевую роль[2]

, графические элементы становятся неотъемлемой частью создания увлекательного игрового опыта. В этой главе я рассмотрел один инструмент, используемых для создания анимации и текстуры в играх программу Krita. Krita - это многофункциональный графический редактор с открытым исходным кодом, который обеспечивает широкие возможности для рисования и создания анимации. С его помощью разработчики могут создавать качественные текстуры, анимированные персонажи, фоны и спецэффекты, придавая игровому миру живость и оригинальность. Рассмотрел основные возможности Krita и его применение в игровой индустрии. Изучил методы создания анимированных текстур, используемых для поверхностей и объектов в игре, а также рассмотрел процесс создания анимированных персонажей с помощью программы. Также проанализировал примеры использования Krita в игровых проектах различных жанров, чтобы продемонстрировать разнообразие и гибкость этого инструмента. Рассмотрел, как создание красочных миров в 25 жанре фэнтези и научной фантастики, так и создание стилизованных графических элементов в жанре платформеров и рогаликов. Создание 3D моделей и объектов с помощью Blender В мире игровой разработки трехмерные модели и объекты играют ключевую роль в создании визуального облика игрового мира. В этой

главе мы я рассмотрел, один из инструментов инструментов для создания 3D контента это программа Blender. Blender - это бесплатный и открытый 3D редактор с широкими возможностями моделирования, анимации, текстурирования и рендеринга[7]

. С его помощью разработчики могут создавать разнообразные 3D модели, персонажей, архитектурные объекты, а также анимированные сцены и спецэффекты. В этой главе я изучил основные возможности Blender и его применение в процессе создания 3D контента для видеоигр. Рассмотрел методы моделирования различных объектов и персонажей, а также процесс текстурирования и создания материалов для них. Далее я проанализировал примеры использования Blender в игровых проектах различных жанров. Я рассмотрел как создание детализированных персонажей и окружений в жанре RPG и action, так и создание стилизованных и абстрактных объектов в жанре аркад и платформеров. Использование Rive для создания партиклов и интерактивной графики В мире игровой разработки создание динамичных и интерактивных эффектов играет важную роль в создании атмосферы игры и вовлечении игроков. В этой главе я рассмотрел инструмент Rive, который предоставляет возможности для создания анимированных эффектов и интерактивной графики. Rive - это мощный инструмент для создания векторной анимации и интерактивных эффектов, который позволяет разработчикам создавать сложные и динамичные анимации с минимальными усилиями. С его помощью можно создавать партикловые эффекты, анимированные переходы, пользовательские интерфейсы и многое другое. Так же использовались: 1. Figma: Онлайн-платформа для дизайна интерфейсов, которая обеспечивает совместную работу между дизайнерами и разработчиками. Figma позволяет создавать прототипы, макеты и интерактивные элементы. 2. Adobe After Effects: Программа для создания анимаций и спецэффектов в видео. After Effects предлагает широкий спектр инструментов для создания движущейся графики, композитинга и работы с визуальными эффектами. 3. Adobe XD: Инструмент для дизайна пользовательских интерфейсов и прототипирования. XD позволяет создавать макеты, проводить тестирование пользовательских интерфейсов и делиться проектами с командой. 26 4. Adobe Illustrator: Программа для

создания векторной графики, которая широко используется для разработки логотипов, иллюстраций, иконок и других графических элементов. Система конечных автоматов (Finite State Machine, FSM) - это модель поведения, которая состоит из конечного набора состояний, переходов между этими состояниями и действий, связанных с каждым состоянием и переходом. FSM широко используется в игровой разработке для управления поведением персонажей, объектов и игровых систем. Применение FSM система в играх 1. Управление поведением персонажей: FSM позволяет программистам определить различные состояния, в которых может находиться персонаж (например, бег, прыжок, атака, ожидание), и определить переходы между этими состояниями в зависимости от условий (например, нажатие клавиш, обнаружение врага). Это позволяет создать плавное и естественное поведение персонажа в игре. 2. Управление ИИ: FSM может использоваться для реализации поведения искусственного интеллекта (ИИ) вражеских персонажей. Например, FSM может определять различные тактики атаки и обороны в зависимости от текущей ситуации на поле боя. 3. Управление игровыми системами: FSM может использоваться для управления различными игровыми системами, такими как системы заданий, системы управления ресурсами или системы управления игровым интерфейсом. Например, FSM может определять состояния игрового меню (главное меню, настройки, пауза) и переходы между ними. 4. Управление анимациями и эффектами: FSM может использоваться для управления анимациями персонажей, объектов и спецэффектов в игре. Например, FSM может определять различные состояния анимации персонажа (идет, бежит, атакует) и переходы между ними в зависимости от действий игрока или внешних событий. Настройка камеры в играх играет ключевую роль в обеспечении удобства игрового процесса, создании атмосферы и визуального восприятия игрового мира. Вот несколько основных аспектов настройки камеры: Тип камеры: Перспективная камера: Обеспечивает эффект глубины и трехмерности. Это наиболее распространенный тип камеры в 3D-играх. Ортографическая камера: Предоставляет плоское, без перспективы изображение. Часто используется в 2D-играх. Положение и ориентация камеры: Позиция: Задает местоположение камеры в игровом мире. Направление: Определяет, куда смотрит камера. Дистанция от игрового объекта: В 3D-играх это может быть расстояние от камеры до игрового персонажа или точки интереса. 27 В 2D-играх это может быть фиксированное расстояние или масштабирование сцены. Угол обзора и зум: Определяет угол обзора камеры и масштаб сцены. В 3D-играх это может быть угол обзора вокруг игрового объекта. В 2D-играх это может быть масштабирование сцены для показа более широкого или узкого участка. Следование за объектом: Камера может автоматически следовать за игровым персонажем или другими объектами для удобства игрока. Это может быть полностью автоматизированным или контролируемым игроком. Эффекты и анимации камеры: Поддержка эффектов, таких как размытие при движении или зуме. Анимации при изменении положения камеры для плавного перехода между различными точками обзора. Настройка пользовательского интерфейса: Отображение информации на экране, такой как миникарта или HUD (головной дисплей). Регулировка интерфейса для удобства игрока в зависимости от положения камеры. "2D" и "3D" элементы: 2D относится к двумерному пространству, которое имеет только ширину и высоту, но не глубину. Графика 2D представляет собой плоские изображения, состоящие из двух осей: X (горизонтальная) и Y (вертикальная). Примеры 2D графики включают в себя спрайты, текстуры, интерфейсы пользовательского интерфейса и фоны в играх. Рисунок 3.1. Создание двумерных элементов 3D относится к трехмерному пространству, которое имеет ширину, высоту и глубину. Графика 3D представляет собой трехмерные модели, которые могут быть рассмотрены с различных точек зрения и имеют объем и глубину. 28 Примеры 3D графики включают в себя трехмерные объекты, персонажей, окружения и анимации в играх. Основные различия между 2D и 3D в игровой разработке: Рисунок 3.2 Создание трёхмерных элементов Глубина и объем: В 3D графике объекты имеют объем и могут быть видны с разных сторон, в то время как в 2D графике объекты представлены только на плоскости. Реалистичность: 3D графика обычно более реалистична и имеет больше потенциала для создания глубоких и интересных игровых миров, в то время как 2D графика часто используется для создания более стилизованных и аркадных игр. Сложность разработки: Разработка игр с использованием 3D графики часто более сложна и требует больше времени и ресурсов, чем игры с 2D графикой, из-за необходимости моделирования объектов в трехмерном пространстве и управления освещением и анимацией. Используемые инструменты: для создания и редактирования 2D графики обычно используются графические редакторы, такие как Photoshop или GIMP, в то время как для 3D графики требуются специализированные программы для моделирования и анимации, такие как Blender или Maya. 3.2 Создание персонажа Определение концепции и стиля персонажа На данном этапе было важно определить внешний вид и основные характеристики главного героя. Была проведена работа по созданию концепт- артов, которые помогли визуализировать идеи и задать общий стиль персонажа. Основные аспекты включали: 29 Определение пропорций и форм тела. Разработка деталей одежды и аксессуаров. Выбор цветовой палитры. Создание основных положений персонажа Для разработки спрайтов основных положений персонажа были выбраны следующие позиции: Стояние. Ходьба. Бег. Прыжок. Анимация атаки. Каждая из этих позиций была проработана отдельно с целью создания плавной и естественной анимации. Использование программного обеспечения Для создания спрайтов были использованы следующие программы: Adobe Photoshop[29] для рисования и редактирования графических элементов. Figma[30] для создания векторной графики. Создание базовых кадров анимации Каждое положение персонажа разбил на несколько ключевых кадров, которые формируют основную анимацию. Например, для анимации ходьбы было создано 8 кадров, отображающих различные фазы движения ног и рук. Детализация и доработка спрайтов После создания базовых кадров начался процесс детализации. Он включал: Добавил мелкие детали, такие как складки одежды, тени и световые акценты. Корректировка пропорций и поз персонажа для более реалистичной анимации. Рисунок 3.3. Создание главного персонажа Создание UV полигонов и привязка костей Для анимации персонажа по ключевым точкам необходимо было создать UV полигоны для каждой отдельной части тела персонажа. Этот процесс включал: Определение и разрезание текстур: Текстуры персонажа были разделены на отдельные части, такие как руки, ноги, туловище и голова. 30 Для каждой части тела была создана UV-развертка, чтобы обеспечить правильное отображение текстур. Рисунок 3.4. Создание костей Привязка костей: В Godot Engine для каждой части тела персонажа создал кости, которые привязывались к соответствующим полигонам. Каждая кость имела заданные веса, чтобы контролировать, какие части полигонов она будет двигать. Это обеспечивало естественные и плавные движения. Рисунок 3.5. UV развёртка правой ноги 31 3.3 Анимация с использованием ключевых точек После создания UV полигонов и привязки костей начал процесс анимации. Для этого использовал ключевые точки, такие как rotate и position. Основные шаги включали: Определение ключевых кадров: Для каждой анимации были определены ключевые кадры. Например, для анимации ходьбы это начальное положение ноги, среднее положение и конечное положение. Эти ключевые кадры служили опорными точками для создания плавных переходов. Анимирование костей: Кости анимировал путем изменения их позиции и вращения в ключевых точках. Например, для ходьбы кость ноги изменяла свое положение и угол наклона в каждом ключевом кадре, создавая иллюзию шага. Настройка плавных переходов: Настроил плавные переходы между ключевыми кадрами, чтобы избежать резких движений и сделать анимацию более реалистичной. Этот процесс включал интерполяцию позиций и углов наклона костей между ключевыми кадрами. Рисунок 3.6. Создание анимации AnimationPlayer 32 Тестирование и корректировка анимаций На заключительном этапе спрайты импортировал в тестовое окружение, где проводил тестирование анимаций. Выявил и исправил ошибки, такие как: Неплавные переходы: Обнаружил и скорректировал резкие или неестественные переходы между кадрами. Несоответствие скоростей: Скорости движения различных частей тела скорректировал для достижения более синхронизированных движений. Корректировка весов костей: Весы костей скорректировал для более естественных и реалистичных движений персонажа. Использование AnimationPlayer и AnimationTree Для управления и воспроизведения анимаций в Godot Engine использовал следующие инструменты: AnimationPlayer: Создание анимаций: В редакторе AnimationPlayer создал и настроил анимации для различных действий персонажа, таких как ходьба, бег, прыжок и атака. Ключевые кадры: В анимациях использовал ключевые кадры для параметров rotate и position костей, что позволяло точно контролировать движение каждой части тела персонажа. Управление анимациями: AnimationPlayer обеспечивал управление воспроизведением анимаций, включая запуск, остановку, паузу и циклическое воспроизведение. AnimationTree: Система состояний: В AnimationTree была настроена система состояний, которая позволяла переключаться между различными анимациями в зависимости от действий и состояний персонажа. Узлы анимации: AnimationTree использовал узлы анимации для определения логики переходов между различными анимациями. Например, переход от состояния стояния к состоянию ходьбы. Смешивание анимаций: С помощью AnimationTree было настроил смешивание анимаций, что обеспечивало плавные переходы между разными состояниями персонажа. Использование SkeletonModificationStack2D с IK Target Для более реалистичной и естественной анимации конечностей персонажа использовал SkeletonModificationStack2D с IK Target: SkeletonModificationStack2D: Модификации скелета: Этот инструмент позволял применять различные модификации к скелету персонажа, такие как инверсная кинематика (IK) и коррекция поз. 33 Стек модификаций: SkeletonModificationStack2D использовал стек модификаций, который позволял применять и комбинировать несколько модификаций для достижения желаемого эффекта. IK Target: Инверсная кинематика: IK Target использовал для создания реалистичных движений конечностей персонажа. Этот метод позволял задавать конечную точку (таргет), к которой должна стремиться конечность, а система IK автоматически рассчитывала углы вращения костей для достижения этой точки. Настройка IK: В редакторе Godot Engine задал таргеты для ног и рук персонажа, что позволило создать реалистичные анимации, такие как шаги при ходьбе и движения рук при атаках. Рисунок 3.7. Древо сцены 34 3.4 Создание уровня Создание уровня началось с проектирования его структуры и общей компоновки. Были определены ключевые зоны и элементы, такие как комнаты, коридоры и объекты взаимодействия. Структура уровня: Основные зоны уровня были созданы с помощью узлов сцены, таких как Node2D, которые представляют собой различные комнаты и области. Размещение объектов: Объекты, такие как двери (например, OldLabPrisonDoor), лифты

(ElevatorTexture), и различные интерактивные элементы (Apparatus), были добавлены на сцену и размещены в нужных местах. Рисунок 3.8. Макет первой локации Наложение текстур Текстуры были наложены на элементы уровня для придания им визуального стиля и уникальности. Текстуры для объектов: Для каждого объекта сцены назначил соответствующие текстуры, чтобы визуально идентифицировать их. Это включало текстуры для дверей, стен и пола. Использование атласов: Для оптимизации работы с текстурами использовались атласы, объединяющие несколько текстур в один файл, что ускоряло загрузку и отображение объектов. 35 Рисунок 3.9. Текстуры объектов Создание коллизий Для обеспечения физического взаимодействия объектов в уровне настроил коллизии. Коллизии для объектов: Для каждого объекта, с которым игрок мог взаимодействовать, был добавлен компонент CollisionShape2D. Этот компонент определяет область, где будет происходить столкновение. Например, для дверей OldLabPrisonDoor и стен WallMap были созданы соответствующие коллизии, чтобы предотвратить прохождение сквозь них. Настройка форм коллизий: Форма коллизий была тщательно настроена, чтобы максимально точно соответствовать геометрии объектов. Использовались такие формы, как прямоугольники, круги и многоугольники. TileMap для создания стен и пола Для создания стен и пола уровня использовался компонент TileMap, который позволяет легко и эффективно размещать повторяющиеся элементы. Создание тайлов: Создал тайлы (плитки) для различных типов стен и пола. Эти тайлы объединил в наборы тайлов (TileSet). Каждому тайлу назначил соответствующие текстуры, а также настроил коллизии, чтобы предотвратить прохождение через стены и другие препятствия. 36 Рисунок 3.10. Текстуры окружения Размещение тайлов: С помощью TileMap стены и пол разместил в нужных местах уровня. Этот процесс включал рисование уровня с использованием тайлов, что значительно ускоряло процесс создания и редактирования уровня. Разработка физики окружения Для придания уровню реалистичного поведения и взаимодействия объектов разработал физику окружения. Статические тела (StaticBody2D): Для всех неподвижных объектов, таких как стены и пол, использовал узлы StaticBody2D. Эти узлы обеспечивают статические коллизии, что позволяет игроку и другим объектам взаимодействовать с ними. Например, для пола FloorMap и стен WallMap создал соответствующие StaticBody2D с компонентами CollisionShape2D. Рисунок 3.11. Локация с учётом коллизий Освещение: Для улучшения визуального восприятия уровня добавлил источники света, такие как DirectionalLight2D и PointLight2D. Эти источники создавали реалистичное освещение и тени, что улучшало атмосферу уровня. LightOccluder2D использовались для создания теней от объектов, таких как стены и двери, что добавляло глубины и реализма сцене. 37 Рисунок 3.12. Локация с учётом освещения 3.5 Создание движения персонажа Движение персонажа обеспечивается использованием компонента CharacterBody2D и написанием соответствующих скриптов для управления его движением. Основные этапы включают настройку физических параметров и разработку логики управления. Основные параметры Параметры движения: Скорость (SPEED): Определяет, насколько быстро персонаж перемещается по горизонтали. За максимальную высоту прыжка отвечает переменная MAX_JUMP_HEIGHT и за ускорение прыжка переменная JUMP_ACCELERATION они определяют параметры прыжка персонажа. За Параметры рывка отвечают переменные DASH_SPEED, DASH_DURATION и DASH_COOLDOWN вместе они определяют скорость и продолжительность рывка, а также время восстановления между рывками персонажа. Гравитация: Используется настройка гравитации из конфигурации проекта (physics/2d/default_gravity), чтобы персонаж правильно падал под действием силы тяжести. Основная логика движения: В функции _process(delta) обрабатываются действия пользователя и обновляется состояние персонажа. В функции _physics_process(delta) выполняется обработка физики персонажа, включая движение и гравитацию. Создание Finite State Machine (FSM) Для управления состояниями персонажа используется конечный автомат состояний (FSM). Это позволяет легко переключаться между различными состояниями, такими как "Стоит", "Движение" и "Прыжок". FSM_Meta: Основной класс FSM, который содержит перечисление состояний и входные данные для FSM. 38 Рисунок 3.13. Код машины состояний 1 Базовый класс FSM_State: Базовый класс для всех состояний FSM. Содержит методы для обработки входных данных (input), обновления состояния (update), входа в состояние (enter) и выхода из состояния (exit). Рисунок 3.14. Код машины состояний 2 Конкретные состояния FSM: Класс FSM_State_Idle для состояния "Стоит": Проверяет входные данные и переключается на состояния "Прыжок" или "Движение" в зависимости от действий пользователя 39 Рисунок 3.15. Код машины состояний 3 Класс FSM_State_Jump для состояния "Прыжок": Обрабатывает логику прыжка и переключается на другие состояния в зависимости от того, находится ли персонаж на земле или выполняет движение. Рисунок 3.16. Код машины состояний 4 40 Класс FSM_State_Move для состояния "Движение": Обрабатывает логику движения и переключается на другие состояния в зависимости от действий пользователя. Рисунок 3.17. Код машины состояний 4 Подключение скриптов Скрипты подключаются к узлу игрока для управления его поведением и состояниями. Основной скрипт игрока: Управляет движением и состояниями игрока, используя FSM. Рисунок 3.18. Скрипт управления персонажем 1 41 Рисунок 3.19. Скрипт управления персонажем 2 Рисунок 3.20. Скрипт управления персонажем 3 Обновление 42 3.6 Создание звуков в программе FL Studio Что бы создать звуковых эффекты и музыкальное сопровождение в проекте я использовал программу FL Studio. Этап создания звуков включал разработку и экспорт звуковых файлов, которые впоследствии использовал в игровом движке.

Запуск и настройка проекта: Открыл FL Studio и создал новый проект. Настроил параметры проекта, такие как темп (BPM) и ключ, в зависимости от нужд проекта[2]

. Создание треков и звуковых эффектов, для игры: Выбор инструментов: Я использовал встроенные инструменты и плагинами FL Studio для создания разных звуков. Например, использовал Sytrus для синтезирования звуков ходьбы или FPC для создания ударных звуков. Создание мелодий и ритмов: С помощью инструмента Piano Roll создавал мелодию. Расположил ноты в нужной последовательности, чтобы создать музыкальные треки. Редактирование и обработка: Использовал микшер FL Studio для добавления эффектов (реверберация, задержка, эквализация и т.д.) к трекам и звуковым эффектам. Запись звуков: Если требовалось записать звуки с микрофона или внешних источников, настроил аудио входы и записал нужные звуки. Экспорт звуковых файлов: Настройка экспорта: Перешел в меню File -> Export и выбрал нужный формат файла (например, WAV или MP3). Установил параметры экспорта, такие как битрейт и частота дискретизации. Экспорт: Нажал на кнопку экспортирования и сохранил звуковые файлы в нужную директорию. Рисунок 3.21. FL Studio 43 Добавление звуков в проект на движке Godot Импорт звуковых файлов: Переместил экспортированные звуковые файлы из FL Studio в папку проекта Godot. В окне "FileSystem" Godot, звуковые файлы автоматически импортировались и были готовы для использования. Создание аудио узлов: AudioStreamPlayer: Создал узел AudioStreamPlayer для воспроизведения звуковых эффектов. Выбрал нужный узел в дереве сцены и нажал "Add Child Node". Нашел и выбрал AudioStreamPlayer. AudioStreamPlayer2D или AudioStreamPlayer3D: В зависимости от того, хотел ли я, чтобы звук воспроизводился в 2D или 3D пространстве, выбрал соответствующий узел (AudioStreamPlayer2D или AudioStreamPlayer3D). Настройка звуковых узлов: Загрузка аудио файлов: В инспекторе узла AudioStreamPlayer выбрал свойство Stream и загрузил соответствующий звуковой файл. Настройка параметров: Установил параметры воспроизведения звука, такие как громкость, зацикливание (loop) и автозапуск (autoplay), в инспекторе. Подключение звуков к игровым событиям: Скрипты для воспроизведения звуков: Написал скрипты, которые будут запускать звуковые эффекты при определенных игровых событиях. Например, при движении персонажа, прыжке или атаке.

Оптимизация и тестирование Оптимизация звуков: Сжатие и конверсия: Если необходимо, использовал инструменты сжатия и конвертации для уменьшения размера звуковых файлов без потери качества[1]

. Баланс громкости: Проверил баланс громкости всех звуковых эффектов и музыкальных треков, чтобы они не заглушали друг друга и были слышны в нужные моменты. Тестирование звуков в игре: Запустил игру и протестировал все звуковые эффекты и музыкальные треки в различных ситуациях. Убедился, что звуки воспроизводятся корректно, синхронизированы с игровыми событиями и не вызывают задержек или лагов. 3.7 Компаньон Хаоса Компаньон Хаоса представляет собой чат-бота, способного генерировать ответы на основе введенных пользователем сообщений с использованием нейронной сети Seq2Seq и классифицировать их характер с помощью другой нейронной сети. 44 Для реализации чат-бота я использовал глубокое обучение и библиотеку PyTorch. Модель Seq2Seq состоит из энкодера и декодера с использованием LSTM (Long Short-Term Memory) для обработки последовательностей слов. Энкодер преобразует входные сообщения пользователя в векторы, а декодер генерирует ответы на основе этих векторов. Для классификации характера ответа я использовал еще одну нейронную сеть, которая принимает на вход векторы слов и предсказывает категорию ответа. Для обучения и оценки моделей я использовал набор данных, содержащий сообщения пользователей и соответствующие им ответы компаньона Хаоса. Набор данных был предварительно обработан, включая токенизацию и преобразование текста в числовые последовательности. Модели Seq2Seq и классификатора были обучены на этих данных с использованием метода обратного распространения ошибки и оптимизатора Adam. После обучения модели были экспортированы в формат ONNX для использования в приложении на платформе Godot. Это позволяет интегрировать модели непосредственно в игровое окружение и использовать их для генерации ответов и классификации типов ответов в реальном времени. Для дальнейшего использования чат-бота в приложении на платформе Godot был создан скрипт, который загружает модели и использует их для генерации ответов на ввод пользователя. Компаньон Хаоса представляет собой сложную систему, объединяющую обработку естественного языка и методы машинного обучения для создания диалоговой модели. Основной целью данной работы

является разработка модели, способной генерировать ответы на сообщения пользователя, а также классифицировать типы ответов. В данном разделе подробно рассмотрены этапы создания и обучения модели. Подготовка данных Для обучения модели была использована база данных, содержащая диалоги между пользователем и ботом. Пример данных включал следующие столбцы: User_Message: сообщения пользователя. Bot_Response: ответы бота. Response_Type: типы ответов (например, вопрос, утверждение и т.д.). Данные были загружены с использованием библиотеки pandas и предварительно обработаны. Для удобства обучения к ответам бота добавлялись специальные токены и , указывающие на начало и конец последовательности. Рисунок 3.22. Скрипт загрузки и обработки данных 45 Токенизация и создание словаря Для работы с текстом необходимо было преобразовать текстовые данные в числовые представления. Для этого были использованы токенизация и создание словаря. Кодирование и разбиение на обучающую и тестовую выборки Далее, текстовые данные были преобразованы в числовые последовательности на основе созданного словаря. Рисунок 3.23. Скрипт разбиение на обучающую и тестовую выборки 1 Данные были разделил на обучение и тест выборки с помощью метода под названием train_test_split. Рисунок 3.24. Скрипт разбиение на обучающую и тестовую выборки 2 Рисунок 3.25 – Скрипт разбиение на обучающую и тестовую выборки 3 Модель была обучена с использованием функции потерь CrossEntropyLoss и оптимизатора Adam. Функция потерь CrossEntropyLoss (кросс-энтропийная потеря) является одной из наиболее часто используемых функций потерь в задачах классификации, в том числе для обучения моделей Seq2Seq. Она измеряет разницу между предсказанными вероятностными распределениями и истинными распределениями классов. 46 Основные характеристики: Классификационная задача: Кросс-энтропия используется для задач классификации, где цель состоит в том, чтобы предсказать правильный класс из множества возможных классов. Входные параметры: input: Содержит логиты (не нормализованные вероятности) размерности (N, C) или (N, C, H, W), где N - размер батча, C - количество классов, H и W - высота и ширина (для задач сегментации). target: Истинные метки классов размерности (N) или (N, H, W), содержащие индексы целевых классов. Как это работает: Логиты и вероятности: Модель выдает логиты, которые преобразуются в вероятности с помощью функции softmax. Softmax превращает логиты в распределение вероятностей по классам. Расчет кросс-энтропии: Кросс-энтропия измеряет "непохожесть" предсказанных вероятностей на истинные метки классов. Она определяется как: Где pi - истинная вероятность класса i (в one-hot виде), qi - предсказанная вероятность класса i. Оптимизатор Adam (Adaptive Moment Estimation) является одним из наиболее популярных оптимизаторов, используемых для обучения нейронных сетей благодаря его эффективности и стабильности. Основные характеристики: Адаптивное обучение: Adam адаптирует скорость обучения для каждого параметра, используя оценки первых (среднее) и вторых (дисперсия) моментов градиентов. Входные параметры: params: Параметры модели, которые необходимо обновить. lr: Коэффициент обучения (learning rate). betas: Коэффициенты для вычисления средних и дисперсий градиентов (по умолчанию (0.9, 0.999)). eps: Малая величина для численной стабильности (по умолчанию 1e-8). weight_decay: Параметр для регуляризации (по умолчанию 0). 47 Рисунок 3.26. Скрипт тренировочной модели Создание и обучение классификатора ответов Помимо генерации ответов, система должна была классифицировать типы ответов. Для этого был создан отдельный классификатор. Процесс обучения моделей Seq2Seq и классификатора был объединен в одну функцию для совместной оптимизации. Рисунок 3.27. Скрипт обучения классификатора и Seq2Seq моделей 48 Экспорт моделей После того, как я завершил обучения модели были экспортированы в формат ONNX для дальнейшего использования в различных средах. Рисунок 3.28. Скрипт для экспорта моделей Таким образом, был разработан и обучен компаньон Хаоса, способный генерировать ответы на сообщения пользователей и классифицировать типы ответов. Модель продемонстрировала хорошую способность к генерации текстов и классификации, что открывает возможности для дальнейшего улучшения и использования в различных диалоговых системах. Импорт моделей и библиотеки Microsoft.ML.OnnxRuntime и Microsoft.ML.OnnxRuntime.DirectML Microsoft.ML.OnnxRuntime Microsoft.ML.OnnxRuntime - это библиотека для выполнения моделей машинного обучения, сохраненных в формате ONNX (Open Neural Network Exchange). Она даёт использовать заранее обученные модели для выполнения предсказаний на различных платформах и устройствах. Основные характеристики: Поддержка ONNX: ONNX является открытым форматом для представления моделей машинного обучения, который поддерживается многими фреймворками, включая PyTorch, TensorFlow и другими. Платформенная независимость: ONNX Runtime можно использовать на различных платформах, таких как Windows, Linux и macOS. Производительность: Оптимизирован для высокопроизводительного выполнения моделей, поддерживает аппаратное ускорение с помощью различных бекендов. 49 Основные классы и методы: InferenceSession: Основной класс для выполнения инференса (предсказаний) с использованием модели ONNX. NamedOnnxValue: Класс для работы с входными и выходными данными модели. Microsoft.ML.OnnxRuntime.DirectML Microsoft.ML.OnnxRuntime.DirectML - это расширение ONNX Runtime, которое обеспечивает аппаратное ускорение с использованием DirectML (Direct Machine Learning). DirectML - это высокопроизводительный API для выполнения операций машинного обучения на графических процессорах (GPU), разработанный для Windows. Основные характеристики: Аппаратное ускорение: Использует возможности GPU для ускорения выполнения моделей машинного обучения. Интеграция с DirectX 12: DirectML использует инфраструктуру DirectX 12 для выполнения операций на GPU. Производительность: Значительно ускоряет выполнение моделей по сравнению с использованием только CPU. Рисунок 3.29. Скрипт обучения классификатора и Seq2Seq моделей 50 Рисунок 3.30. Скрипт обучения классификатора и Seq2Seq моделей Рисунок 3.31. Скрипт обучения классификатора и Seq2Seq моделей 51 Рисунок 3.32. Скрипт обучения классификатора и Seq2Seq моделей 52 Для реализации перемещения персонажа-компаньона в Godot, можно использовать различные подходы, включая использование кинематического тела (KinematicBody2D) player: Этот объект связывает компаньона с игроком, что позволяет компаньону реагировать на действия игрока. movement_area: Прямоугольная область (Rect2), в пределах которой компаньон может случайным образом перемещаться. target_position: Векторная позиция, к которой компаньон будет стремиться. move_speed: Константа, определяющая скорость движения компаньона. Метод _ready: set_process(true): Включает обработку кадров для данного узла, что позволяет обновлять его состояние каждый кадр. set_target_position(): Устанавливает начальную целевую позицию для компаньона. player.u_turn.connect(test_emit): Подключает метод test_emit к сигналу u_turn игрока, позволяя реагировать на его повороты. Метод test_emit: Изменяет положение movement_area в зависимости от направления поворота игрока (left или другое). set_target_position(): Устанавливает новую целевую позицию после изменения области движения. Метод set_target_position: Генерирует случайную целевую позицию в пределах movement_area с использованием функции randi_range. Метод _process: Вызывается каждый кадр и обновляет направление движения компаньона. direction_to_target: Нормализованный вектор направления к целевой позиции. Если расстояние до цели менее определенного значения (1.45), то устанавливается новая целевая позиция. velocity: Скорость компаньона, вычисляемая на основе направления и заданной скорости, умноженной на delta для корректного перемещения. move_and_slide(velocity): Перемещает компаньона с учетом возможных столкновений. 53 Рисунок 3.33. Скрипт обучения классификатора и Seq2Seq моделей Создание чат-бота "Компаньон Хаоса" представляет собой актуальное исследование искусственного интеллекта и обработки естественного языка. Наш подход позволяет создать мощный инструмент коммуникации, способный взаимодействовать с пользователями на естественном языке и адаптироваться к различным ситуациям и контекстам. 3.8 Создание новой сцены: Открыл Godot и создал новый проект. Создал новую сцену (Scene -> New Scene). В корне сцены добавил узел типа CanvasLayer, который будет содержать все элементы UI. Добавление элементов интерфейса: Внутри CanvasLayer добавил два узла типа Label для отображения текста. Первый Label для отображения здоровья, второй для отображения уровня. 54 Шаг 2: Настройка элементов интерфейса Настройка Label для здоровья: Переименовал узел в HealthLabel. В инспекторе установил текст (Text) на "Health: 100". Настроил размер и позицию, чтобы текст был виден в верхнем левом углу экрана. Настройка Label для уровня: Рисунок 3.34. Скрипт обучения классификатора и Seq2Seq моделей Переименовал узел в LevelLabel. В инспекторе установил текст (Text) на "Level: 1". Настроил размер и позицию, чтобы текст был виден в верхнем правом углу экрана. Добавление ProgressBar для отображения здоровья: 55 Внутри CanvasLayer добавил узел типа ProgressBar. Переименовал узел в HealthBar. Настроил размер и позицию, чтобы полоска здоровья находилась под текстом "Health". Установил начальное значение (Value) на 100 и максимальное значение (Max) на 100. 3.7 Создание новой сцены для врага: В Godot создал новую сцену (Scene -> New Scene). Добавил узел типа CharacterBody2D в качестве корневого узла сцены. Переименовал корневой узел в Enemy. Добавление спрайта и анимации: Рисунок 3.35. Спрайт врага Внутри узла Enemy добавил узел типа Sprite для отображения внешнего вида врага. Загрузил текстуру для спрайта из имеющихся ресурсов. Настроил позицию и размер спрайта по необходимости. Добавил узел типа AnimationPlayer для управления анимациями врага. Создал анимации для движения и атак врага, используя ключевые кадры для спрайта. Создание коллайдера: Внутри узла Enemy добавил узел типа CollisionShape2D. Выбрал форму коллайдера (например, RectangleShape2D) и настроил её размеры, чтобы она совпадала с размером спрайта. 56 Создание зоны обнаружения Добавление зоны обнаружения: Внутри узла Enemy добавил узел типа Area2D. Переименовал узел в DetectionArea. Внутри DetectionArea добавил узел типа CollisionShape2D. Выбрал форму коллайдера (например, CircleShape2D) и настроил её радиус, чтобы он соответствовал радиусу обнаружения, указанному в скрипте. Настройка сигналов зоны обнаружения: Подключил сигнал body_entered зоны DetectionArea к методу _on_DetectionArea_body_entered в скрипте врага. Добавление врага в игровую сцену Импорт врага в основную сцену: Открыл основную игровую сцену. Импортировал сцену врага как инстанс (Instance Child Scene) в нужное место в игровой сцене. Разместил врагов в нужных местах на карте, настроив их начальные позиции. Тестирование и настройка Тестирование поведения врага: Запустил игру и проверил, как враг обнаруживает игрока, следует за ним и атакует. Проверил патрулирование врага, когда игрок находится вне зоны обнаружения. Настройка параметров врага: Настроил параметры скорости, урона и радиуса обнаружения для врагов, чтобы они соответствовали

уровню сложности игры. Для улучшения пользовательского опыта и расширения возможностей взаимодействия с приложением, я добавил поддержку геймпада. Теперь игроки могут управлять персонажем с помощью геймпада, что делает игровой процесс более удобным и интуитивным. Описание функционала Геймпад используется для управления движением персонажа по игровому полю. При нажатии на кнопки геймпада персонаж перемещается в указанном направлении в пределах заданной области движения. Для обеспечения плавности и естественности движения я использовал алгоритм, который плавно перемещает персонажа к целевой позиции. Интеграция в игровое окружение Поддержка геймпада интегрирована в игровое окружение с помощью скриптов на платформе Godot. Я определил кнопки геймпада и привязали их к соответствующим действиям в игре, таким как перемещение персонажа влево, вправо, вверх и вниз. Управление движением персонажа При нажатии на кнопки геймпада персонаж начинает движение в указанном направлении. Для обеспечения плавности и естественности 57 движения я использовал алгоритм, который плавно перемещает персонажа к целевой позиции. Это создает ощущение управления персонажем с помощью геймпада более приятным и интуитивным. Расширение возможностей Добавление поддержки геймпада позволяет расширить аудиторию вашего приложения, поскольку многие игроки предпочитают использовать геймпады для управления игровыми персонажами. Это также делает ваше приложение более доступным для людей с ограниченными возможностями, которые могут испытывать трудности при использовании клавиатуры и мыши. 58 Заключение В ходе исследования динамики видеоигровой индустрии и проекта "Имитация человеческого поведения в мире тьмы и фэнтези" было продемонстрировано, как синтез творчества и технологий, в частности искусственного интеллекта, формирует будущее индустрии. Проект представляет собой не только значительный технический прогресс, но и воплощение амбициозных идей, направленных на создание захватывающего игрового опыта. Работа охватила различные аспекты разработки игры на движке Godot, начиная с создания AI-компаньона и заканчивая интеграцией сложных алгоритмов машинного обучения через ONNX модели. Созданный пользовательский интерфейс и реализованные враги с разнообразным поведением продемонстрировали возможности Godot как мощного инструмента для разработки игр и внедрения современных технологий AI. Проделанная работа стала хорошей основой для дальнейшего расширения и улучшения игрового проекта. В процессе выполнения проекта были реализованы несколько ключевых аспектов разработки игры на движке Godot, охватывая различные механики и компоненты, необходимые для создания полноценного игрового опыта. Основные этапы включают: Создание AI-компаньона: разработан AI-компаньон, следующий за игроком и перемещающийся в заданной области. Использован класс, наследующий CharacterBody2D, и реализован скрипт на GDScript для управления поведением компаньона. Оптимизация и функции потерь: Обсуждены функции потерь и оптимизаторы, используемые в обучении нейронных сетей, такие как CrossEntropyLoss для задач классификации и Adam для адаптивного обучения. Импорт модели и библиотеки ONNX: Рассмотрен процесс импорта модели ONNX в Godot с использованием библиотек Microsoft.ML.OnnxRuntime и Microsoft.ML.OnnxRuntime.DirectML, что позволяет интегрировать сложные алгоритмы AI в игровой процесс.

Создание пользовательского интерфейса: Описан процесс создания UI в Godot, включая добавление и настройку элементов интерфейса, таких как кнопки, текстовые метки и панели[1]

, для взаимодействия с игроком. Создание врага: Реализован враг, способный патрулировать, обнаруживать игрока, следовать за ним и атаковать. Создана отдельная сцена для врага с добавлением спрайта, анимаций и коллайдеров. Скрипт врага включал логику патрулирования, слежения и атаки, а также обработку столкновений. Таким образом, данный проект не только подтвердил потенциал использования Godot для разработки игр с интеграцией AI, но и продемонстрировал возможности создания интерактивного и увлекательного 59 игрового опыта. Тщательное изучение и реализация ключевых аспектов разработки игр с использованием искусственного интеллекта позволили создать прочную основу для дальнейших исследований и усовершенствований в данной области. Данный проект подтверждает готовность современных технологий и рынка к внедрению подобных инновационных проектов, что открывает новые горизонты для развития видеоигровой индустрии. Рисунок 4.1.1 – Законченный проект