МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТЮМЕНСКИЙ индустриальный университет»

Институт сервиса и отраслевого управления

Кафедра Бизнес-информатики и математики

КУРСОВОЙ ПРОЕКТ

Дисциплина: «Интеллектуальные системы и их проектирование»

Тема: «Разработка игры на базе движка godot с добавлением искусственного интеллекта»

Выполнил:

Студент 4 курса группы РИСб-20-1

направления 45.03.04 «Интеллектуальные

системы в гуманитарной сфере»

Никитин М.Д.

Руководитель:

Доцент, канд. пед. наук

Спирин И.С.

Тюмень, 2024 г.

Содержание

Введение 3

1 Теоретические аспекты разработки игр 4

1.1 Основные понятия 4

1.2 Этапы разработки видеоигры 7

1.3 Анализ предметной области 11

2 Аналитический аспект разработки игры 14

2.1 Выбор ПО, используемого в разработке 14

2.2 Анализ видеоигр схожих по жанру 17

2.3 Описание игры 21

3 Практическая часть разработки игры 24

3.1 Подготовка к разработке 24

3.2 Разработка геймплея 28

3.3 Разработка интерфейса 33

Заключение 36

Используемые источники 37

# Введение

С развитием информационных технологий и ростом популярности видеоигр, их разработка становится все более актуальной и востребованной областью в ИТ-индустрии. Видеоигры одновременно являются средством развлечения, популяризации новых технологий, а также обучения и образования. Одним из самых быстрорастущих игровых движков в индустрии, на данный момент, является Godot. Имея полностью бесплатную модель пользования, открытый код, свой собственный язык программирования, предназначенный специально для разработки игр и разработчиков, активно прислушивающихся к мнению пользователей, данный движок набирает популярность как среди разработчиков, так и среди инвесторов. В данной работе будет рассмотрена разработка видеоигры на движке Godot, включающая в себя элементы искусственного интеллекта.

Целью – изучение процесса разработки видеоигр на движке Godot и создание прототипа игры, отражающего полученные знания и навыки.

Объект – игра, созданная на движке Godot

Предмет – процесс разработки игры

Для достижения поставленной цели необходимо решить следующие задачи:

1. Проанализировать возможности и ограничения движка Godot.
2. Изучить процесс планирования и разработки игр.
3. Создать дизайн-документ.
4. Подобрать набор инструментов для создания ассетов.
5. Создать ассеты.
6. Разработать альфа-версию игры.

# Теоретические аспекты разработки игр

## Основные понятия

Компьютерная игра – компьютерная программа, служащая для организации игрового процесса (геймплея), связи с партнёрами по игре, или сама выступающая в качестве партнёра.

Геймплей – компонент игры, отвечающий за взаимодействие игры и игрока. Геймплей описывает, как игрок взаимодействует с игровым миром, как игровой мир реагирует на действия игрока и как определяется набор действий, который предлагает игроку игра.

Игровой движок – базовое программное обеспечение компьютерной игры.

Игровой ассет или игровой ресурс – цифровой объект, преимущественно состоящий из однотипных данных, неделимая сущность, которая представляет часть игрового контента и обладает некими свойствами. Понятие «игрового ассета» используется при разработке компьютерных игр по отношению к тем элементам контента, которые обрабатываются ресурсной системой как неделимые (атомарные, элементарные) сущности.

Игровой искусственный интеллект – набор программных методик, которые используются в компьютерных играх для создания иллюзии интеллекта в поведении персонажей, управляемых компьютером. Игровой ИИ, помимо методов традиционного искусственного интеллекта, включает также алгоритмы теории управления, робототехники, компьютерной графики и информатики в целом.

Игровая механика – набор правил и способов, реализующий определённым образом некоторую часть интерактивного взаимодействия игрока и игры. Все множество игровых механик игры формируют конкретную реализацию её игрового процесса или геймплея.

Godot – это универсальный 2D и 3D игровой движок, спроектированный для поддержки всех видов проектов.

GDScript – это высокоуровневый, объектно-ориентированный, императивный язык программирования с постепенной типизацией, созданный для Godot. Он использует синтаксис на основе отступов, аналогичный таким языкам, как Python. Его цель – оптимизация и тесная интеграция с Godot Engine, обеспечивая большую гибкость при создании и интеграции контента.

Aseprite – это редактор изображений с доступным исходным кодом, предназначенный в первую очередь для рисования и анимации в стиле пиксель-арт. Он работает на Windows, macOS и Linux и содержит различные инструменты для редактирования изображений и анимации, такие как слои, кадры, поддержка тайловых карт, интерфейс командной строки и другие.

Тайловая, плиточная или знакоместная графика (от англ. tile – плитка) – метод создания больших изображений (как правило, уровней в компьютерных играх) из маленьких фрагментов одинаковых размеров.

ChipTone – бесплатный инструмент для генерации звуковых эффектов, распространяющийся по лицензии CC0, что означает, любые коммерческие и некоммерческие проекты могут выполняться без разрешения автора приложения.

Основными элементами в Godot являются:

Узел (Или нода) – самый маленький строительный блок игры. В качестве свойств могут иметь имя, принадлежность к потоку обработки, свойства обработки (Обрабатывается ли узел только во время паузы, только когда нет паузы, всегда, никогда или наследует это свойство от родителя). К каждому узлу можно прикрепить не больше одного скрипта. Существует огромное количество узлов под самые разные задачи.

Скрипт – файл, содержащий в себе определенный код. В Godot скрипты обрабатываются только если они прикреплены к узле или загружены отдельно путем подключения автозагрузки (Принцип проектирования «Одиночка» иди «Singleton»).

Одиночка – порождающий шаблон проектирования, гарантирующий, что в однопоточном приложении будет единственный экземпляр некоторого класса, и предоставляющий глобальную точку доступа к этому экземпляру.

Сцена – многократно используемый шаблон. В сцена может быть чем угодно, начиная от персонажа или оружия и заканчивая интерфейсом, домом или отдельным уровнем. В Godot сцены очень гибки и одновременно выполняют роль шаблонов и сцен из других игровых движков. Также следует упомянуть что одной из сцен присваивается титул главной, и именно она загружается первой при запуске игры.

Дерево сцены – все сцены игры собираются в дереве сцен, буквально дереве из сцен. И, так как сцены – деревья узлов, дерево сцены также является деревом узлов.

Сигнал – это шаблон «Наблюдателя» в исполнении движка Godot.

Шаблон «Наблюдателя» – это поведенческий шаблон проектирования. Также известен как «подчинённые». Реализует у класса механизм, который позволяет объекту этого класса получать оповещения об изменении состояния других объектов и тем самым наблюдать за ними.

Шина – в программирование, класс "Издатель" из паттерна "Издатель-подписчик". При возникновении события в «издателе», вызываются методы «подписчика(ов)» на это событие.

Ресурс – обобщенное название всех контейнеров данных. Ресурсом может быть изображение, скрипт, анимация, звуковой файл, сцена и т.д. Также Godot позволяет создавать свои шаблоны ресурсов, содержащие в себе другие ресурсы.

Хитбокс – в компьютерных играх тело, которое проверяется на столкновение.

## Этапы разработки видеоигры

Как и любое другое приложение, видеоигра проходит несколько различных этапов в процессе разработки:

1. Концептирование. На этом первом шаге команда придумывает концепцию игры, и проводит начальную проработку игрового дизайна. Главная цель данного этапа – это геймдизайнерская документация, в нашем случае, дизайн-документ. 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. 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. Закрытое бета тестирование. На этапе закрытого бета тестирования 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. Открытое бета тестирование. На этом этапе продолжается тестирование игры, но уже на широкой аудитории. Происходит заморозка разработки фич, программисты перестают реализовывать что-то новое, а полностью переключаются на модификацию и улучшение имеющегося функционала.
7. Релиз. На этом этапе должно быть полностью отлажено оперирование продукта (техническая поддержка, работа с сообществом), 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 ММО проектов). Также реализует различные внутриигровые функции, поддерживающие новые монетизационные схемы. Также идет разработка и интеграция в продукт нового контента, поддерживающего интерес игроков.

Соответственно, самым первым официально документируемым этапом разработки является создание дизайн-документа. Цель дизайн-документа заключается в том, чтобы однозначно описать коммерческие аспекты игры, целевую аудиторию, игровой процесс, графику, дизайн уровней, историю (сюжет), персонажей, пользовательский интерфейс, средства и т. д. Для осуществления поставленной цели, каждое требование к разработке определенной части игры должно быть достаточно подробно описано для соответствующих разработчиков (дизайнеров, программистов и т. д.). Документ намеренно разделен таким образом, чтобы разработчики игры могли поддерживать отдельные его части.

Дизайн-документ – это план работы от начала и до конца проекта. А это значит, что все основные задачи должны в нем находиться, и не только задачи, а и приблизительные методы решения. В частности:

1. Схема игры. Что должен делать игрок, какова конечная цель, что мешает ее достижению.
2. Интерфейс. Подробно описанная функциональная часть (что можно делать, каким образом – меню, мышь, горячие клавиши, кнопки и т.д.).
3. Игровая механика. Как устроен игровой мир, какие характеристики есть у его объектов, формулы движения, боя и всего остального, ролевая система, физика – по вкусу.
4. Программные механизмы и алгоритмы. Какими характеристиками будут обладать графический движок, ИИ, сетевой код, интерфейс, редактор карт, звук.
5. Графика. Сколько и каких вам понадобится моделей, анимаций, двумерной графики, роликов, обоев (да, и их тоже стоит запланировать заранее). Здесь крайне желательны (может, лучше сказать – необходимы) хоть какие-то наброски, концепт арты, по которым можно почувствовать визуальный стиль игры.
6. Звуки и музыка. Темы, вид и способ отображения звуков, набор звуковых эффектов.
7. Сюжет. Общая сюжетная канва, план кампаний, основные задания и т. п. – в зависимости от жанра. Каждая из предполагаемых карт должна быть запланирована здесь.
8. Игровой мир. Основные персонажи / монстры / виды войск с параметрами и примерным расположением / способом добычи и производства.
9. Сотрудники, зарплаты, сроки и план работы.

## Анализ предметной области

Перед созданием видеоигры необходимо ответить на ряд важных вопросов. Данные ответы предоставят достаточно полную информацию для написания дизайн-документа, что, соответственно, значительно ускорит и упростит процесс разработки.

Определение целевой аудитории. Как самый большой сегмент геймеров, целевой аудиторией были выбраны мужчины в возрасте от 16 до 24 лет, владеющие небольшим количеством свободного времени.

Далее предстоял выбор платформы. Ввиду материальных ограничений, разработка консольных игр не является доступным вариантом, соответственно остается мобильный и ПК гейминг. Исходя из статистики, хоть мобильный гейминг и растет, спрос на ПК гейминг все еще сохраняет свои позиции. Однако, разработка на ПК является более простым и быстрым вариантом ввиду гораздо большего разброса производительности, разрешения экрана и объема памяти на мобильных устройствах. Также, в России, ПК гейминг все еще является приоритетным. Исходя из этих факторов, а также учитывая личные предпочтения и доступность людей для будущего закрытого бета тестирования, платформой разработки был выбран ПК.

Далее необходимо было определить продолжительность игрового процесса. Это является настолько же важным параметром, как и платформа, ввиду того что разные целевые аудитории предпочитают разную продолжительность игры. В целях ускорения тестирования было решено остановиться на длительности игры в 5 минут. Такая продолжительность игры позволяет сократить время, необходимое на тестирование баланса игры, а также позволяет войти в нишу игр, доступных в короткий период времени, например обеденный перерыв или 15-ти минутная разгрузка в процессе выполнения домашнего задания.

Определение игрового процесса. Игровой процесс должен быть динамичным и полноценным, соответственно стратегические игры и игры жанра Idle не подходят. Пазлы также не являются хорошим вариантом, ввиду необходимости расслаблять человека, а не напрягать. Мой выбор пал на жанр шутеров (стрелялок). Данный жанр являлся популярным на протяжении всей истории видеоигр и все еще не начал терять популярности. Для поддержания вовлеченности игроков дополнительным жанром был выбран rogue-lite. Суть этого жанра заключается в случайных улучшениях, получаемых во время каждой отдельной игровой сессии, также называемой «забегом», улучшением характеристик между «забегами» и процедурной генерации врагов и локаций. Основной игровой цикл был выбран следующий: Игрой заходит в игру, выбирает персонажа и оружие, после чего на протяжении 5 минут старается пережить нападение постоянно увеличивающихся волн врагов, параллельно получая опыт и улучшения. В конце он получает валюту для улучшений между «забегами», тратит эту валюту и повторяет цикл пока не закончится время или игра не наскучит ему.

Определение графики. Выбор графики был совершен исключительно исходя из личных умений ввиду того, что в наше время больше ценится стиль, а не фотореализм в графике. Создание графики также должно было занимать минимальное количество времени при сохранении приемлемых результатов. Соответственно, идеальным вариантом стала графика в стиле пиксель-арт и разрешением 32 на 32 пикселя. При необходимости допускается возможность увеличения размера холста или размера самих пикселей.

Определения дизайна уровня. Дизайн уровней должен быть простым и не отвлекать игрока от геймплея. Создание атмосферы планируется за счет использования визуальных шейдеров и эффектов в целях экономии времени на создание ассетов уровней. Оптимальным вариантом в таком случае являются подземные локации, где большая часть уровня будет скрыта темнотой.

Определение истории (Сюжета). Исходя из выбора подземной локации и механики получения улучшений, главный герой должен обладать причиной быть под землей и навыками для модификации своего оружия. Таким образом была выбрана роль инженера-шахтера, решено посадить главного героя внутрь робота, а врагов сделать инопланетными подземными существа ввиду отсутствия подземных хищников на нашей планете. Также, исходя из того, что игрок остается один против множества врагов, было решено, что по сюжету, он упал в пещеру уровнем ниже и ждет подмоги. Приходом подмоги, по совместительству, можно объяснить ограничение по времени.

Определение персонажей. Персонажи в игре должны соответствовать работе инженера-шахтера. Соответственно, они должны быть изобретательными, старательными, возможно грубые или недовольные условиями труда.

Определение пользовательского интерфейса. Пользовательский интерфейс должен четко отражать нужную информацию, но при этом оставлять как можно больше пространства на экране не занятым. Оптимальным размещением стали 3 шкалы в левом верхнем углу и одна шкала в центре снизу.

# Аналитический аспект разработки игры

## Выбор ПО, используемого в разработке

Самым главным средством разработки является движок, соответственно, с него выбор и начался. Самыми крупным на данный момент являются Unity, Unreal Engine и Godot.

Unity – кроссплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Плюсы Unity:

* самое большое сообщество из трех вариантов;
* встроенная реклама и аналитика;
* возможность разработки на множество платформ;
* бесплатная версия для игр с доходом меньше ста тысяч долларов;
* самый большой выбор готовых ассетов;
* самый большой выбор обучающих материалов;
* язык программирования C#;
* большие возможности самого движка.

Минусы Unity:

* не удобная структура и интерфейс;
* высокий вес файлов на выходе;
* ошибки, не исправленные на протяжении многих версий;
* несмотря на количество обучающего материала и ассетов, очень сложный для новичка движок;
* 3D графика приемлема, но уступает Unreal Engine.

Unreal Engine – игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. Первой игрой на этом движке был шутер от первого лица Unreal, выпущенный в 1998 году. Хотя движок первоначально был предназначен для разработки шутеров от первого лица, его последующие версии успешно применялись в играх самых различных жанров, в том числе стелс-играх и файтингах. Плюсы Unreal Engine:

* лучшая графика из трех вариантов;
* удобные инструменты моделирования уровней;
* большое сообщество и количество обучающего материала;
* встроенный визуальный скриптинг;
* бесплатный;
* готовых ассетов меньше количественно, но лучше по качеству.

Минусы Unreal Engine:

* большой вес и программы и финальной игры;
* очень высокий порог входа;
* интерфейс хоть и удобный, но очень объемный;
* плохая производительность.

Godot – открытый кроссплатформенный 2D и 3D игровой движок под лицензией MIT, который разрабатывается сообществом Godot Engine Community. До публичного релиза в виде открытого ПО движок использовался внутри некоторых компаний Латинской Америки. Среда разработки запускается на Android, HTML5, Linux, macOS, Windows, BSD и Haiku и может экспортировать игровые проекты на ПК, консоли, мобильные и веб-платформы. Плюсы Godot:

* лучшая документация из трех вариантов;
* бесплатный;
* размер сообщества уступает только двум другим движкам;
* открытый исходный код, быстрое исправление багов;
* работает на всех платформах и позволяет экспортировать почти на все платформы;
* производительный, игры имеют малый вес;
* удобная организация структуры файлов;
* поддержка GDScript, неполная поддержка C#, C++;
* возможность поиска документации внутри приложения.

Минусы Godot:

* относительно молодой движок, существует необходимость устанавливать дополнительные модули;
* плохая производительность с большим количеством объектов;
* небольшое количество обучающих материалов, частая смена версий и последующее устаревание старых материалов;
* малое количество успешных игр и общая популярность движка.

Unreal Engine данному проекту не подходит ввиду объема и сроков, данный движок больше подходит для крупных проектов, создаваемых командой профессионалов. Unity же хоть и является одним из самых популярных движков, последние несколько лет страдает от плохого руководства и постепенно теряет популярность ввиду этого. Также, мой личный опыт работы с Unity оставил впечатление гораздо хуже, чем Godot. Исходя из всего этого, выбор движка пал на Godot.

Выбор языка программирования соответственно, был сделан при выборе движка, ведь Godot полностью построен вокруг использования языка GDScript и использование других языков хоть и увеличит производительность, но значительно увеличит время разработки.

Выбор программы для рисования ассетов имеет значение только в трех критериях: удобство, стоимость и лицензия на использование. Ввиду предыдущего опыта использования Aseprite, их лицензии на свободное использование созданных ассетов в разработке игр, а также низкой цены (435 рублей на момент написания курсовой), был выбран именно он. Для создания звуков было решено использовать единственный удобный полностью бесплатный редактор ChipTone, распространяемый на itch.io по лицензии, позволяющей выполнять любые манипуляции, не запрашивая разрешения автора.

## Анализ видеоигр схожих по жанру

Главным вдохновением, и, одновременно, конкурентами являются три игры: Nova drift, 20 minutes until dawn и vampire survivors. Все три игры являются представителями жанра rogue-lite с видом сверху и два из них также являются стрелялками.

Vampire Survivors – компьютерная игра в жанрах shoot’em up и roguelike, разработанная и изданная Лукой Галанте. Игра была первоначально выпущена по модели раннего доступа в 2021 году; полные версии для macOS и Windows вышли в октябре 2022 года. Основными элементами геймплея являются:

* ограничение по времени 30 минут (в некоторых случаях меньше), после чего игроку засчитывается победа;
* улучшение характеристик между «забегами»;
* случайный выбор улучшений во время «забега»;
* большое количество врагов, постепенно заполняющих весь экран;
* все оружие стреляет автоматически, от игрока требуется только ходить и выбирать улучшения.

Рисунок . – Vampire survivors

20 minutes until dawn – это игра стрелялка в жанре rogue-lite от Flanne. Игрок управляет персонажем, который сражается с непрерывными волнами монстров, с целью пережить натиск до рассвета. Основными элементами геймплея являются:

* ограничение по времени в 20 минут, после чего игроку засчитывается победа;
* улучшение характеристик между «забегами»;
* случайный выбор улучшений во время «забега»;
* все улучшения разделены на ветки по 4, где первое открывает второе и третье, а второе или третье открывает четвертое и самое сильное;
* суперулучшения, появляющиеся если игрок выбрал определенную комбинацию обычных;
* выбор персонажа и оружия перед началом игры;
* большое количество врагов, постепенно заполняющих весь экран;
* игрок может как сам выбирать направление стрельбы, так и позволить игре выбирать направление автоматические.

Рисунок . – 20 minutes until dawn

Nova drift – это космический шутер в жанре rogue-lite, который объединяет в себе впечатления от классических аркадных игр и некоторые возможности современных ARPG. Основными элементами геймплея являются:

* отсутствие ограничения по времени, вместо этого игра может продолжаться бесконечно;
* между «забегами» открываются новые улучшения, но отсутствует улучшение характеристик;
* случайный выбор улучшений во время «забега»;
* все улучшения разделены на ветки по 4, где первое открывает второе и третье, а второе или третье открывает четвертое и самое сильное;
* суперулучшения, появляющиеся если игрок выбрал определенную комбинацию обычных;
* выбор персонажа и оружия в начале игры, а также возможность заменить его вместо получения нового улучшения;
* враги появляются волнами, новая волна приходит или когда предыдущая уничтожена или если у игрока уходит слишком много времени;
* игрок может как сам выбирать направление стрельбы, но движение и стрельбы происходит в одну сторону, также, игрок останавливается не сразу и может скользить по полю боя.

Рисунок . – Nova drift

Исходя из этого анализа, игроки хотят:

* большое количество врагов;
* различные улучшения между «забегами»;
* большое количество улучшений, взаимодействующих друг с другом во время самого «забега»;
* контроль над направлением стрельбы;
* вариативность при выборе персонажа и оружия;
* демоверсию перед покупкой.

И игрокам не нравятся:

* слишком долгие «забеги»;
* персонажи, отличающиеся только характеристикам;
* излишнее облегчение игры за счет улучшений между «забегами»;
* излишняя сложность.

В своей игре я хочу совместить все самое лучшее, при этом избежав недостатков и одновременно привести новшество в жанр, используя искусственный интеллект, отвечающий за динамическую сложность.

## Описание игры

Основные жанры игры – топ-даун скролл шутер, rogue-lite.

Игрок управляет роботом, оснащенным шахтерским оборудованием и застрявшим в пещере в ожидании подмоги.

Конечная цель игрока – выжить на протяжении 5 минут, после чего проигрывается анимация прибытия подмоги, все враги погибают, и игрок побеждает.

Основные механики – различный выбор модификаций оружия и меха, возможность как автоматической, так и ручной стрельбы из оружия, возможность прокачки между «забегами», ИИ-директор, управляющий всеми событиями, не попадающими под прямой контроль игрока, в том числе:

* частота появления волн врагов;
* количество и виды врагов, появляющихся в волнах;
* появление временных усилений.

Игрок управляет инженером, сидящим внутри меха и застрявшим в бесконечной пещере. Перед началом игры на выбор дается несколько разных видов стартового оружия и мехов.

Мех оборудован различными системам:

* бронепластины, защищающие его от повреждений;
* топливная система;
* две ячейки под вооружение, одно ближнего и одно дальнего боя;
* инструменты для улучшений.

Бронепластины. Являются аналогом здоровья в остальных играх, в стандартной вариации персонаж стартует с тремя пластинами, каждая защищающая от одного удара или снаряда врага. После того как все пластины будут уничтожены, последующее нанесение урона полностью уничтожит мех игрока, и игра будет закончена. Получить больше пластин можно, используя улучшения или случайные бонусы, падающие с потолка.

Топливная система. Меха игрока требует топливо для функционирования всех систем. Топливо можно получить из трех источников. Биомасса врагов, улучшения и случайные бонусы, падающие с потолка. Минимальное количество топлива – 0, максимальное – 400 единиц. В игре отображается процентами от 0 до 200, где 100% это 200 единиц и все проценты выше 100 будут с увеличенной скоростью сжигать топливо, избавляясь от излишков, но дополнительно усиливая персонажа. В тоже время, если топливо упадет до нуля, персонаж начнет сжигать патроны для поддержания работы всех систем, во время этого все системы работают на таком же уровне, как и при 100% топлива, но игрок теряет 5% максимального боезапаса в секунды. Если же и патроны кончаются, игра заканчивается проигрышем. Уровень топлива отвечает сразу за несколько игровых механик:

* уровень света, излучаемой лампой;
* скорость передвижения игрока;
* количество появляющихся врагов;
* скорострельность оружия.

Уровень света варьируется от 0.5м при 5% и ниже, до 15м при 100% и 30м (полный экран) при 200%.

Скорость передвижения варьируется от 30% при 5% топлива и ниже, до 100% при 100% топлива и 150% при 200% топлива.

Количество появляющихся врагов будет подробно расписано в разделе про ИИ директора.

Скорострельность оружия варьируется от 80% при 5% топлива и ниже, до 100% при 100% топлива и 120% при 200% топлива.

Вооружение. Перед началом игры, игроку предоставляется выбор одного оружия ближнего боя и дальнего боя. (Дополнительные виды оружия кроме стартовых планируются как часть дипломной работы). Оружие дальнего бля будет иметь ограниченный максимальный боезапас и магазин. Получить патроны можно за счет улучшений, случайно появляющихся в пещере останков других мехов и случайных бонусов, падающих с потолка. Оружие ближнего боя не имеет магазина, тратит топливо и является оружием с высоким уровнем риска и награды при использовании.

Инструменты для улучшений. При убийстве врагов, с них будет выпадать металлолом (Аналог опыта в других играх). Весь металлолом в небольшом радиусе вокруг игрока будет собран и складирован. При сборе определенного количества металлолома (10+x^1.5 где x – количество уже полученных улучшений) игроку будут предложены на выбор 2 улучшения меха и/или оружия. Или же 3, в случае, когда уровень топлива выше 100%.

Каждое улучшение является часть мини ветки из четырех различных улучшений. Каждый раз при появлении возможности выбрать улучшение, игра использует случайные несколько из массива. Изначально в массиве находятся только улучшения первого уровня, помеченного на шаблоне цифрой один. При выборе улучшения первого уровня, в массив добавляются улучшения второго и третьего уровня. Если же игрок со временем получил оба улучшения второго и третьего уровня, в массив добавляется финальное улучшение четвертого уровня. Каждая из веток фокусируется на улучшение отдельной характеристики персонажа. Всего планируется 20 улучшений, разбитых на 5 веток.

# Практическая часть разработки игры

## Подготовка к разработке

Перед началом разработки необходимо создать ресурсы, которые будут хранить всю нужную информацию об игроке, улучшениях, оружии и врагах.

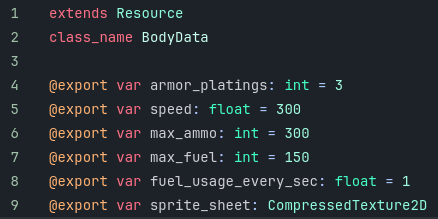
Шаблоны ресурсов в Godot оформляются путем создания скрипта, базирующегося на классе «Resource», после чего им присваивается имя класса и, используя команду @export var задаются все необходимые переменные и их стандартное значение.

Рисунок . – Шаблон ресурса BodyData

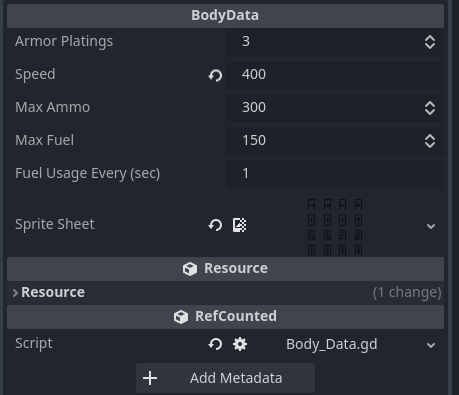
Далее используя пкм в файловой системе, можно создавать инстансы данного ресурса и присваивать им значения. На примере стандартного корпуса приведен пример интерфейса редактирования ресурса.

Рисунок . – Интерфейс редактирования ресурса

По такому же принципу был создан ресурс, хранящий в себе данные об оружии с данными характеристиками:

* наносимый урон;
* вместимость обоймы;
* стоимость одного выстрела;
* скорострельность;
* время перезарядки;
* разброс;
* количество снарядов;
* количество врагов, которые пуля способна пробить;
* количество отскоков между врагами;
* текстура;
* смещение дула (необходимо для создания пули на конце оружия, а не в середине);
* скорость пули;
* максимальная дальность пули;

Также этот ресурс содержал в себе свойство «снаряд», которое имеет типа «упакованная сцена». За счет этого можно отдельно создать снаряд (Будь то пуля, удар дрелью и т.д.), настроить его поведение, внешний вид и хитбокс, после чего просто использовать как свойство ресурса и игра будет автоматически создавать нужный снаряд при выстреле.

Еще одним необходимым ресурсом был «UpgradeInfo», который хранит в себе всю информацию об улучшениях, доступных игроку. В нем помимо уже известных свойств и типов используется @export\_multiline var, позволяющий писать многострочный текст и словарь, хранящий в себе все возможные характеристики. Полный список характеристик:

1. Изображение.
2. Название улучшения.
3. Название дерева.
4. Уровень улучшения
5. Описание.
6. Характеристики.
   1. Урон.
   2. Скорость атаки.
   3. Пробой.
   4. Максимальный боезапас.
   5. Размер обоймы.
   6. Отскок.
   7. Снаряды.
   8. Максимальный запас топлива.
   9. Скорость.
   10. Броня.
   11. 8 булевых переменных, маркирующих включено ли улучшение, меняющее поведение персонажа.

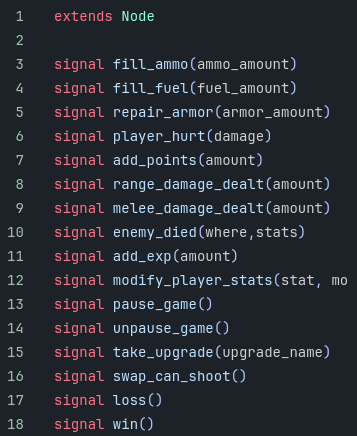
В качестве двух глобальных «Одиночек» была создана шина сигналов и шина звуков. Шина сигналов необходимо для передачи данных между сценами, когда нет возможности подключить сигнал напрямую. Тут же можно увидеть синтаксис оформления сигналов

Рисунок . – Шина сигналов

Доступ к шине сигналов происходит через команды «SignalBus.имя\_сигнала).emit(Значения)», а подключение происходит по команде «SignalBus.имя\_сигнала).connect(Callable(имя функции.bind()))»

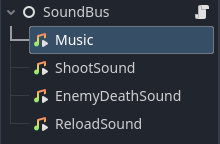
Шина звуков же представляет из себя сцену, где создано 4 узла «AudioStreamPlayer», отвечающие за проигрывание аудио файлов.

Рисунок . – Шина звуков

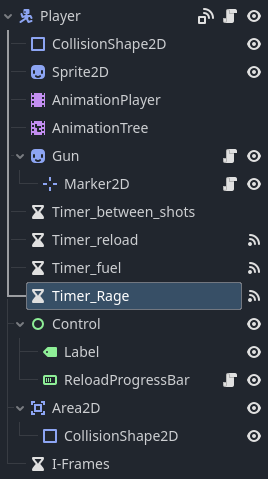
В скрипте, прилепленном к шине, создано 3 метода, отвечающих за включение звуков стрельбы, перезарядки и смерти врагов. Музыка включается автоматически при запуске игры и не требует метода.

Доступ к одиночкам выполняется по их имени, например, команда «SoundBus.play\_shoot\_sound()». запустит звук выстрела.

## Разработка геймплея

Самой первой и самой важной сценой является игрок. Его скрипт отвечает за движение, выбор оружия, стрельбу, перезарядку и функции различных улучшений. Одним из необычных элементов движка Godot является необходимость создавать любому физическому объекту или зоне дочерний объект, назначающий ему форму. Эта необходимость вызвана существованием двух разных узлов формы, один является просто формой (квадрат, круг, овал и т.д.), а второй настраиваемый полигоном. Главными элементами героя являются:

1. Главный узел класса «CharacterBody2d».
2. Форма хитбокса.
3. Узлы, составляющие визуал:
   1. Текстура.
   2. Проигрыватель анимаций.
   3. Дерево анимаций.
   4. Текстура оружия.
4. Маркер дула.
5. 5 таймеров:
   1. Время между выстрелами.
   2. Время перезарядки.
   3. Время траты топлива.
   4. Время ярости (Одно из улучшений).
   5. Время неуязвимости при получении урона.
6. Узел «Control», отвечающий за пользовательский интерфейс:
   1. Текст.
   2. Шкала прогресса, показывающая одновременно остаток обоймы в процентах и прогресс перезарядки.
7. Зона, используемая для отбрасывания врагов при получении урона:
   1. Форма этой зоны.

Рисунок . – Сцена героя

Используя экспортируемые переменные, игроку добавляется ресурс BodyData и два ресурса WeaponData, которые используются для определения всех характеристик и даже внешнего вида персонажа.

Следующим важным элементом геймплея были враги, каждый враг имеет одинаковый набор характеристик, содержащихся в ресурсе. Здоровье, скорость, урон, названия и количество выпадающего с него опыта. Также они имеют схожие узлы, хитбокс, текстура и т.д. Однако, они отличаются поведением.

Самый маленький враг имеет стайное поведение и, если рядом с ним никого нет, но он видит союзника, они идут друг к другу и только находясь рядом идут в сторону игрока, в противном случае они по одному идут к игроку.

Большие враги идут напрямую к игроку.

А враги-стрелки подходят на определенное расстояние, встают на месте и начинают обстреливать игрока, подходя ближе только если игрок отойдет от них.

И у всех врагов есть общий алгоритм избегания столкновений друг с другом. Если противник видит еще одного или больше противника слишком близко к себе, они начинают идти ровно противоположно друг другу.

С каждого врага при смерти выпадает опыт, который игрок может собрать и при сборе достаточного количества опыта выполнить улучшение.

Все эти враги призываются за счет узла «ИИ директор», действующего по следующему алгоритму:

ИИ директор получает и тратит очки, адаптируясь под успешность игрока. Некоторые подсчеты происходят независимо от времени:

1. Оружие при выстреле добавляет очки по формуле урон+количество выстрелов в минуту/20
2. При потере бронепластины игроком ИИ теряет 10 очков
3. При потере последней бронепластины ИИ получает 10 очков (Не компенсируется обычной потерей, призвано добить игрока)

Остальные же производятся каждые 15 секунд в данном порядке (Модификатор очков стартует с 0%)

1. +15\*(количество полных минут с начала игры+1)
2. Ии подсчитывает количество нанесенного урона ближнего и дальнего боя, сравнивая их процентное соотношение и, если разница больше 10%, выставляет приоритет врагам, противоположным преобладающему типу. (Если враг использует оружие дальнего боя, на него начинает идти больше крупных врагов ближнего боя и наоборот, если игрок использует оружие ближнего боя, начинает идти больше врагов-стрелков)
3. Ии подсчитывает соотношение убитых врагов к появившимся и, если игрок убил >80%, игра добавляет к модификатору очков 15%
4. Если топливо >80%, игра добавляет к модификатору очков 10% и уменьшает приоритет сброса топлива на следующую минуту
5. Если топливо <20%, игра вычитает из модификатора очков 5% и увеличивает приоритет сброса топлива на следующую минуту
6. Если патроны >80%, игра добавляет к модификатору очков 10% и уменьшает приоритет сброса патронов на следующую минуту
7. Если патроны <20%, игра вычитает из модификатора очков 5% и увеличивает приоритет сброса патронов на следующую минуту
8. ИИ подсчитывает финальные очки по формуле (Все полученные за 15 секунд очки) \*(100%+модификатор)

После всего этого, ИИ тратит очки следуя следующей логике:

1. Приоритеты всех видов сброса помощи (Бронепластины, патроны, топливо и ничего(всегда4)) суммируются, и эта сумма присваивается к максимуму. К примеру, у игрока меньше 20% патронов и 50% топлива. Соответственно, приоритеты будут выглядеть так: Бронепластины – 1, патроны – 2, топливо – 1, ничего – 4. Всем этим событиям присваивается соответствующая количеству вероятность. 12.5%, 25%, 12.5% и 50% соответственно. Далее используя это распределение случайно выбирается одно событие из четырех.
2. Выбирается распределение врагов. 50% врагов всегда будут маленькими и быстрыми, остальные 50% будут разделены между сильными и медленными бойцами ближнего боя или бойцами дальнего боя соответственно приоритету. Если приоритета нет, 25 на 25%, если у одного из типов есть приоритет, то 35 на 15%.
3. Очки делятся между типами врагов соответственно процентному соотношению и за следующие 15 секунд призывается количество врагов равное их доле разделить на их стоимость.

Система улучшений является очень важной, ведь позволяет игроку становится сильнее и противостоять толпам врагов. Во время старта игры, все ресурсы улучшений, хранящиеся в отдельной папке, по очереди перебираются программой и на основе каждого из них создается узел, к нему прикрепляется заранее созданный скрипт, содержащий в себе одну функцию, вызываемую при выборе данного улучшения. Все улучшения, чей уровень равен единице, отправляются в активный массив улучшений, а остальные отправляются на «Склад» улучшений. Функция, прикрепленная к каждому улучшению, вызывается при выборе данного улучшения.

Рисунок . – Функция, вызываемая при выборе улучшения

При выборе улучшения, в случае если это улучшение было первого уровня, все улучшения второго и третьего уровня данного дерева перемещаются в активный массив, в случае же если это улучшение было второго или третьего уровня, то улучшение четвертого улучшения перемещается в активные.

Также эта функция перебирает каждое свойство, хранящееся в словаре внутри ресурса, и посылает сигнал игроку для изменения нужных характеристик.

## Разработка интерфейса

При входе в игру игрок видит главное меню, состоящее из 3 кнопок:

1. Начать игру. При нажатии запускается главная игровая сцена.
2. Настройки. При нажатии показывается меню настроек.
3. Выйти из игры. При нажатии приложение закрывается.

Рисунок . – Главное меню

Меню настроек является отдельно сценой ввиду того, что оно используется и в главном меню, и в меню паузы. Меню настроек на момент написания содержит функционал изменения громкости в целом, а также музыки и звуков отдельно.

Рисунок . – Меню настроек

В самой игре имеется 4 важных интерфейса:

1. Основной интерфейс. Показывается всегда, при показе других интерфейсов увеличивается прозрачность. Содержит в себе отображение здоровья, патронов и топлива в левом верхнем углу, отображение времени по центре сверху и полоску опыта по центру снизу.
2. Интерфейс паузы. Вызывается на клавишу «Esc», также при вызове ставит всю игру на паузу. Содержит в себе 4 кнопки:

Рисунок . – Меню паузы

* 1. Продолжить. При нажатии скрывает меню паузы и продолжает игру. Также можно вызвать по клавише «Esc».
  2. Настройки. При нажатии показывает меню настроек.
  3. Выйти в главное меню. При нажатии возвращает в главное меню, при этом прогресс не сохраняется.
  4. Выйти из игры. При нажатии закрывает приложение.

1. Интерфейс улучшений. Вызывается на клавишу «Проблем» при условии сбора игроком достаточного количества опыта. Показывает игроку 3 улучшения на выбор, при наведении на любое из них показывает соответствующую ему ветку, при наведении на улучшения в ветке, показывает их описание. При клике на любое улучшение, данное улучшение выбирается, меню закрывается, и игрок получает усиление.

Рисунок . – Меню улучшений

1. Меню победы/проигрыша. Показывается при достижении 5 минут или при получении урона без брони. Надпись меняется на «You won» в первом случае и на «You lost» во втором. Содержит также 3 кнопки:
   1. Попробовать заново. При нажатии запускает заново главную игровую сцену.
   2. Выйти в главное меню. При нажатии открывает главное меню.
   3. Выйти из игры. При нажатии закрывает приложение.

# Заключение

В ходе выполнения курсовой работы был получен огромный набор знаний. Разработка игр затрагивает различные сферы как программирования, так и решения проблем в целом. Начиная от редактирования строк и векторной математики до создания комплексных систем загрузки данных, вывода этих данных, построение интерфейса и обработки действий игрока. Также были изучены возможности движка Godot, освоена его модульная система и принципы эффективной работы в нем. За счет полученных знаний уже сейчас есть возможность значительного улучшения кода и принципа работы игры.

Были решены следующие задачи:

1. Проанализированы возможности и ограничения движка Godot
2. Изучен процесс планирования и разработки игр
3. Создан дизайн-документ
4. Подобран набор инструментов для создания ассетов
5. Созданы ассеты
6. Разработана альфа-версия игры

# Используемые источники

1. «Building a UI in godot» [Электронный ресурс]. Режим доступа: – https://youtu.be/i8ySmMGx--0?list=PL5N89Bry725MaCTKQRWhVoV58JCVikQqe (Дата обращения: 20.10.2023).
2. «Custom resources – a godot workflow game changer» [Электронный ресурс]. Режим доступа: – https://youtu.be/vzRZjM9MTGw?list=PL5N89Bry725MaCTKQRWhVoV58JCVikQqe (Дата обращения: 20.10.2023).
3. «Making infinite background for your game (Godot tutorial)» [Электронный ресурс]. Режим доступа: – https://youtu.be/Qd\_vJ1fjYHQ?list=PL5N89Bry725MaCTKQRWhVoV58JCVikQqe (Дата обращения: 20.10.2023).
4. «Resource gathering RPG in godot 4 tutorial series» [Электронный ресурс]. Режим доступа: – https://youtube.com/playlist?list=PLyH-qXFkNSxlANk9EwZmbtECBfbHeW68-&si=Ux7Yh7ut8kDP58Y2 (Дата обращения: 20.10.2023).
5. «Save countless lines of code with animations in godot» [Электронный ресурс]. Режим доступа: – https://youtu.be/5KBNGKYV-vU?list=PL5N89Bry725MaCTKQRWhVoV58JCVikQqe (Дата обращения: 20.10.2023).
6. «Starter state machines in godot 4» [Электронный ресурс]. Режим доступа: – https://youtu.be/oqFbZoA2lnU (Дата обращения: 20.10.2023).
7. «Using composition to make more scalable games in godot» [Электронный ресурс]. Режим доступа: – https://youtu.be/rCu8vQrdDDI?list=PL5N89Bry725MaCTKQRWhVoV58JCVikQqe (Дата обращения: 20.10.2023).
8. «Официальная документация Godot» [Электронный ресурс]. Режим доступа: – https://docs.godotengine.org/ru/4.x/ (Дата обращения: 20.10.2023).
9. «Официальная страница reddit Godot» [Электронный ресурс]. Режим доступа: – https://www.reddit.com/r/godot/ (Дата обращения: 20.10.2023).
10. «Официальный форум Godot» [Электронный ресурс]. Режим доступа: – https://forum.godotengine.org/ (Дата обращения: 20.10.2023).