МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТЮМЕНСКИЙ индустриальный университет»

Высшая школа цифровых технологий

Кафедра Математики и прикладных информационных технологий

КУРСОВАЯ РАБОТА

Дисциплина: «Интеллектуальные системы и их проектирование»

Тема: «Разработка игры на движке Godot с процедурной генерацией в реальном времени»

Выполнил:

Студент 4 курса группы РИСб-20-1

направления 45.03.04 «Интеллектуальные

системы в гуманитарной сфере»

Герасимов Е.В.

Руководитель:

Доцент, канд. пед. наук

Спирин И.С.

Тюмень, 2023 г.

Аннотация

Целью данной курсовой работы является создание игры на движке Godot.

Объектом работы является – игра на движке Godot.

Предметом курсовой работы является процесс создания игры на движке Godot.

Информационная база исследования: электронные ресурсы, статьи, документация движка Godot.

Результат: создана первая версия игры. Предоставлена вся актуальная информация.

Курсовая работа состоит из введения, трёх глав, заключения и списка использованных источников.

Техническое задание

* Изучить возможности для создания игр;
* Проанализировать принципы процедурной генерации;
* Создать дизайн-документ;
* Собрать необходимый список программ для работы;
* Разработать первую версию игры.

Содержание

ВВЕДЕНИЕ 5

1 Теоретическая часть 6

1.1 Основные понятия 6

1.2 Этапы создания игры 8

1.3 Анализ предметной области 10

2 Аналитическая часть 12

2.1 Выбор программ 12

2.2 Анализ видеоигр схожих по жанру 16

2.3 Описание игры 20

3 Практическая часть 22

3.1 Подготовка к разработке 22

3.2 Разработка геймплея 25

3.3 Разработка интерфейса 28

ЗАКЛЮЧЕНИЕ 31

Используемые источники 32

# ВВЕДЕНИЕ

В настоящее время игры являются одним из самых популярных видов развлечения, и разработка игр становится все более востребованной сферой. Разработка игры требует тщательного планирования, решения проблем и творческого мышления.

Актуальность и отличительная черта этого проекта – процедурная генерация уровня. Это ускоряет процесс создания, повышает разнообразие и реиграбельность игры.

В данной курсовой работе будет рассмотрена разработка игры на движке Godot с процедурной генерацией в реальном времени. Будут рассмотрены основные принципы процедурной генерации, а также способы ее реализации на практике с использованием движка Godot.

Цель – Изучение принципов процедурной генерации и создание игры на движке Godot.

Объект курсовой – игра на движке Godot.

Предмет курсовой – Процесс разработки игры с процедурной генерацией

Задачи:

* Изучить возможности для создания игр
* Проанализировать принципы процедурной генерации
* Создать дизайн-документ
* Собрать необходимый список программ для работы
* Разработать первую версию игры

# Теоретическая часть

## Основные понятия

Компьютерная игра – разновидность игр, в которых компьютер выполняет роль ведущего или партнера по игре, а возникающие по ходу игры игровые ситуации воспроизводятся на экране дисплея или телевизора.

Компьютерные игры бывают разных жанров, таких как:

* Стратегия;
* Экшен;
* Квест;
* Симулятор;
* Шутер;
* Защита замка;
* Roguelike;
* Платформер.

Платформер – жанр компьютерных игр, в которых основу игрового процесса составляют прыжки.

Rogue-like – жанр компьютерных игр, характеризующийся процедурной случайной генерацией уровней.

Endless runner – поджанр платформерных игр. Его отличительными признаками являются бесконечность уровня и препятствия, от которых игроку следует уклоняться.

Сайд-скроллер – компьютерная игра, в которой игрок наблюдает за игровым процессом посредством расположенной сбоку виртуальной камеры.

Геймплей – игровой процесс, взаимодействие с объектами игры и набор её механик.

Хитбокс – прямоугольный участок спрайта персонажа, попадание в который игра засчитывает как успешное столкновение персонажа с объектом.

Дизайн-документ — это детальное описание разрабатываемой компьютерной игры.

Графический интерфейс – система средств для взаимодействия пользователя с компьютером, основанная на представлении доступных пользователю функций в виде графических компонентов (окон, кнопок, меню, списков и т. п.).

Paint.NET — бесплатный растровый графический редактор для Windows, основанный на .NET Framework.

Aseprite — это редактор изображений с доступным исходным кодом, предназначенный в первую очередь для рисования и анимации в стиле пиксель-арт.

Движок (игровой движок, game engine) – программное ядро комплексной программной системы (игры), содержащее базовую функциональность игры, но, при этом, не включающее код, специфичный для конкретной игры.

Ассеты – ресурсы игры, отдельно от кода и платформы. Наиболее частые ассеты это звуки и изображения.

Godot Engine – мультиплатформенный движок и набор инструментов с открытым исходным кодом для создания 2D и 3D игр.

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.

Node (узел) – атомы проектов в Godot Engine, Они могут быть назначены дочерними узлами другого узла, что приводит к древовидной структуре. Заданный узел может содержать любое количество узлов в качестве дочерних.

Scene (сцена) – древо узлов. Сцены в Godot могут быть сохранены на диске и затем загружены в другие сцены. Это позволяет высокую гибкость в архитектуре и модели данных проектов Godot.

## Этапы создания игры

Как и все приложения, в процессе разработки видеоигры проходят следующие 7 этапов:

1. Концептуализация. Этап, когда игра осмысливается и обретает первоначальную форму, которая записывается в дизайн-документ. Задача дизайн-документа в том, чтобы структурировать и описать продукт, чтобы опираться на него в дальнейшем при разработке.
2. Прототипирование. Черновой вариант игры, созданный для проверки теоретических задумок предыдущего этапа. Прототип должен быть прост и включать в себя только основы игры и механики, требующие тестирования, чтобы создать его за короткое время.
3. Вертикальный срез – максимально проработанная основная часть игры. На этом этапе требуется сделать минимальное количество функций для полноценной игры, но сделать их максимально правильно.
4. Создание контента. Поверх стабильной и отполированной основы пишутся все функции, которые должны быть готовы к закрытому бета-тесту. Это самый долгий и трудоёмкий этап, так как требует работы всего штата сотрудников: художников, геймдизайнеров, программистов, композиторов и т.д.
5. Закрытое бета-тестирование. Бета-тестирование - это возможность для реальных пользователей использовать продукт в производственной среде, чтобы обнаружить возможные ошибки или проблемы перед общим выпуском. Бета-тестирование является последней стадией тестирования перед выпуском продукта на широкую аудиторию. Цель состоит в обнаружении и исправлении максимального количества ошибок или проблем в контролируемом окружении.
6. Открытое тестирование. В открытом тесте любой пользователь может использовать продукт, который сопровождается сообщением о том, что он находится в стадии бета-тестирования с возможностью оставить обратную связь.
7. Выпуск. К этому этапу работа продукта должна быть налажена, соблюдаются маркетинговые и финансовые планы. Команда разработчиков занимается исправлением технических ошибок, обнаруженных в процессе эксплуатации, оптимизацией продукта и дополнительным контентом для игры.

Первым документом при разработке игры становится дизайн-документ. Он хранит в себе описание бизнес-аспектов игры, целевой аудитории, игрового процесса, графики, дизайна уровней, истории (сюжета), персонажей, пользовательского интерфейса, инструментов разработки и т. д. Каждая секция должна быть написана достаточно подробно, чтобы описать задачи каждого сотрудника и примерные способы их решения. Требуется определить такие аспекты проекта, как:

1. Концепт, история и структура игры
2. Игроки, их цели и возможности. В этой секции описываются игроки, их цели и возможности. Описывается, кто является целевой аудиторией игры, какие у них могут быть интересы и какие возможности им предоставляет игра. Также описываются цели игроков и как они могут достигать их в игре.
3. Графика, визуальный стиль и внешний вид объектов: Элементы, из которых создан игровой мир, текстуры и спрайты, которые используются для объектов, эффекты, создающие атмосферу и т.д.
4. Программное обеспечение, используемое для создания игры. Описываются языки программирования, фреймворки, графические, звуковые редакторы и другие инструменты, которые будут использоваться для разработки игры.
5. Геймплей и управление.
6. Меню и интерфейс
7. Команда разработки и сроки

## Анализ предметной области

Ещё перед созданием дизайн-документа нужно определить примерный вектор разработки продукта путём анализа предметной области.

Выбор игровой платформы. Выпуск игры на консолях сопровождается бюрократическими, техническими и экономическими трудностями, включающими в себя получение девкита для разработки, регистрацию юридического лица, сертификацию проекта и т.д. Также, разработка игр для ПК проще и быстрее.

Мобильные игры зарекомендовали себя, как поле для казуальных игр с однокнопочным управлением, созданных чтобы убить время, что сильно ограничивает возможности при разработке, а также не соответствует целевой аудитории проекта.

Целевая аудитория проекта – самый большой сегмент игроков, а именно мужчины в возрасте 16-24 лет, играющие в игры от 2 до 4 часов в день. Это важно учитывать, чтобы определить продолжительность, формат и жанр игры.

Для игрового процесса была выбрана смесь из жанров rogue-like, платформер и endless runner. Rogue-like имеет высокую реиграбельность за счёт своей случайной генерации, делающей каждый забег отличающимся от предыдущих. Endless runner позволяет игроку самому выбирать продолжительность игровой сессии, так как её длина зависит от умений игрока, а не количества контента. Это расширит целевую аудиторию, включив в неё и тех, кто может посвящать игре целый вечер и тех, кто играет в игры 15 минут на перерыве. Платформерная составляющая позволит быстро погрузиться в игру, так как этот жанр является самым узнаваемым, а также позволит регулировать сложность под уровень умений игрока.

Дизайн. Для визуального стиля игры был выбран пиксель-арт, так как его быстро создать, легко стилизовать, он узнаваем и не нагружает начинающего игрока. Такому внешнему виду хорошо будет соответствовать электронная музыка, создающая атмосферу недалёкого будущего, а значит и киберпанка, как популярного современного жанра литературы. Весь внешний вид выполнен в ярких неоновых тонах на тёмном фоне.

Интерфейс определяется жанрами игры. Так как игра по своей сути бесконечная, требуются показатели умений пользователя, которыми стали очки и время, проведённое на уровне. Меню использует нетипичную шестиугольную форму и компьютерный шрифт, чтобы соответствовать выбранной атмосфере.

Сюжет. Действие игры происходит в антиутопичном будущем, в виртуальном мире, который разрушается под действием вируса. Одна из клеток вируса преследует главного героя, который убегает от неё в попытках выжить.

# Аналитическая часть

## Выбор программ

Основой разработки игры выступает её движок. Логично начать выбор с него. Для начинающих или инди-разработчиков лучшими являются GameMaker, Unity и Godot Engine. Рассмотрим их подробнее.

GameMaker – один из самых популярных игровых движков, позволяющий разрабатывать приложения под множество платформ. Изначально выпущенный в 1999-м году, за свой долгий срок успел зарекомендовать себя и собрать активное сообщество разработчиков. GameMaker предлагает визуальный инструмент для написания кода, который позволяет создавать игры без необходимости писать код, а также собственный язык программирования GameMaker Language (GML).

Достоинства GameMaker:

* Кроссплатформенность, поддерживаемые платформы: Windows, macOS, Linux, Opera GX, Android, iOS, Windows Phone, Tizen, Xbox, PlayStation;
* Поддержка библиотек и расширений, в том числе на разных языках;
* Гибкая ценовая категория, Free версия GameMaker абсолютно бесплатна;
* Интеграция с несколькими системами управления версиями;
* Интеграция со Steam, Google Play, App Store.

Недостатки GameMaker:

* Собственный язык программирования Game Maker Language (GML), который часто критикуется профессионалами;
* Бесплатная версия позволяет создавать игры только для Opera GX;
* Несмотря на возможность работы с 3D, в Game Maker она крайне неудобна.

GameMaker является одним из самых популярных движков для 2D игр и на нём были созданы такие популярные игры как Undertale, Hyper Light Drifter, Hotline Miami, Katana Zero.

Unity – кроссплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. В 2005-м появился как движок строго для 3D игр, пока не начал поддержку 2D в 2013-м. Один из самых поддерживаемых движков для инди-игр с тысячами гайдов.

Достоинства Unity:

* Самое большое сообщество;
* 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:

* Неудобная структура и интерфейс;
* Высокий вес файлов на выходе;
* Ошибки, не исправленные на протяжении многих версий;
* Трудность в освоении;
* Ориентирован на 3D графику.

В сентябре 2023 года Unity объявила о планах взимать плату с разработчиков, зарабатывающих свыше определенного порога, каждый раз при установке их игры на любой игровой платформе. Эта политика нанесла урон репутации Unity и вместе с перечисленными недостатками не позволяет выбрать для работы его. Тем не менее, ранее на этом движке были созданы такие известные проекты, как Cuphead, Hollow Knight, Among Us и Enter the Gungeon.

Godot – кроссплатформенный игровой движок с открытым исходным кодом под лицензией MIT, который разрабатывается сообществом Godot Engine Community. Godot разработан для поддержки как 2D, так и 3D игр, но в 2D работает немного лучше благодаря своей системе измерения на основе пикселей. Он также предлагает свой собственный язык программирования, известный как GDScript. Почти все данные, которые будет использоваться для создания своей игры, включая объекты и анимации, существуют в виде "узлов", которые можно разместить в "сценах". 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;
* 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 и C#, 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.

Из-за новизны движка, на нём было создано мало проектов. Среди них есть: Sonic Colors Ultimate, Deponia, Lumencraft и Beat Invaders.

Поскольку Unity более ориентирован на 3D, а проект было решено создавать в 2D, этот вариант сразу не подходит. GameMaker предоставляет свой функционал на платной основе, что мешает его использовать начинающему разработчику. Следовательно, выбор для этого проекта пал на Godot. Его новизна может выступать как плюсом, так и минусом, поскольку обозначает

Godot был создан в первую очередь для работы с встроенным языком программирования GDScript, и хотя есть возможность подключить другие, он предоставляет наибольшие возможности без затруднений.

Для работы со спрайтами использовались программы Paint.NET и Aseprite по двум причинам: удобство и стоимость. Обе программы бесплатные, Aseprite ориентирован на работу с пиксель-артом, а Paint.NET предоставляет больше функций.

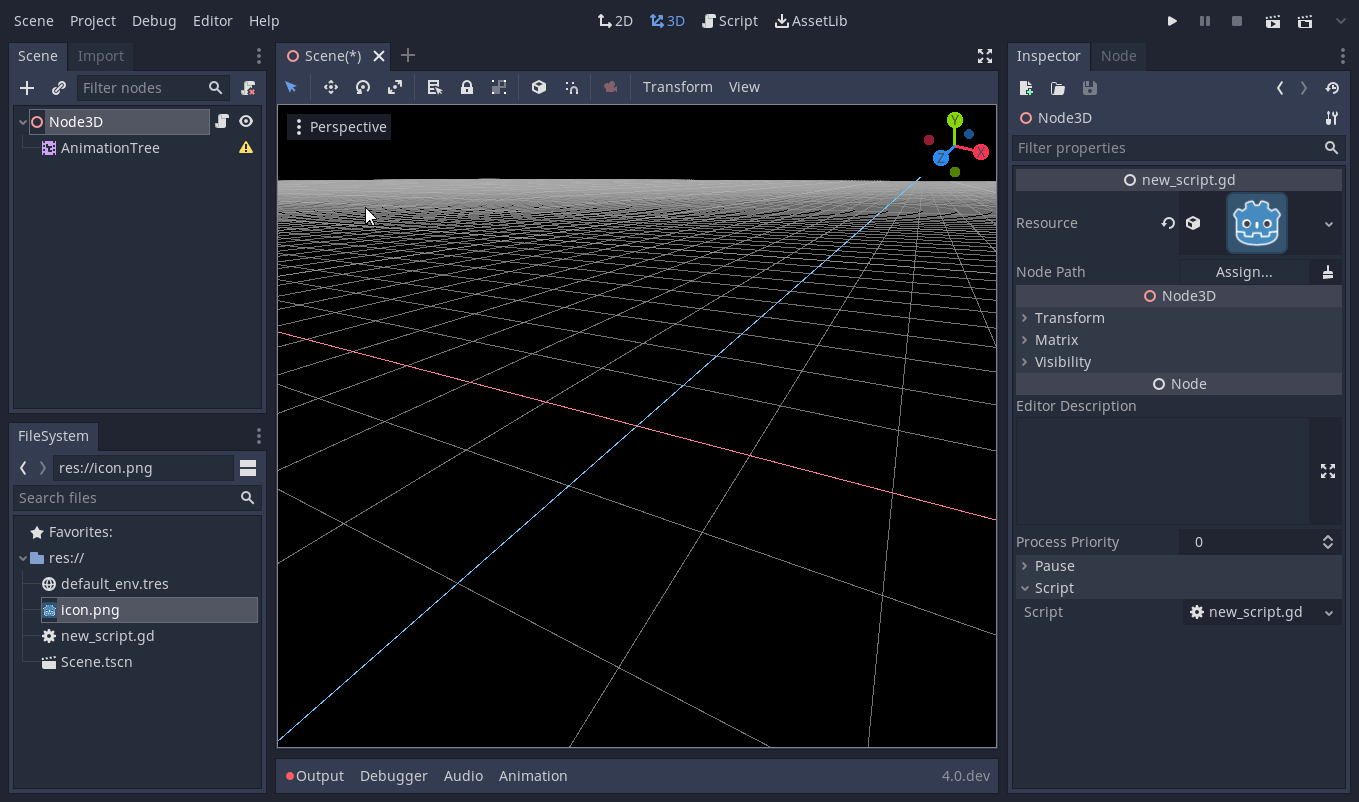


Рисунок 2.1 – окно Godot Engine

## Анализ видеоигр схожих по жанру

Так как бесконечные раннеры в большинстве своём выходят на мобильные устройства, они не представляют конкуренции проекту на ПК, а потому конкурентов нужно искать среди игр жанра rogue-like.

Dead Cells – компьютерная инди-игра в смешанном жанре roguelike и метроидвании, разработанная и выпущенная французской студией Motion Twin для платформ Windows, MacOS и Linux, игровых консолей Nintendo Switch, PlayStation 4 и Xbox One в 2018 году. В ходе Dead Cells игрок управляет существом, пытающемся выбраться из лабиринта. Уровни игры, выполненные в духе двухмерного платформера, генерируются процедурным образом.

Достоинства Dead Cells:

* Динамичная боевая система
* Стильный дизайн
* Разнообразие врагов и видов оружия
* Качественная проработка локаций

Недостатки:

* Отсутствие инноваций
* Отталкивающая сложность
* 5 обязательных прохождений
* Однообразные локации



Рисунок 2.2 – Dead Cells

Spelunky 2 — инди-игра в жанре 2D action-adventure с элементами roguelike. Игрок управляет искателем приключений, который исследует пещеры и собирает сокровища. Целью игры является сбор золота, при этом следует не допускать смерти главного героя.

Игра известна за:

* Приятный визуальный стиль
* Разнообразие локаций
* Реиграбельность
* Уникальные ситуации
* Отличную генерацию уровней

Недостатков у игры меньше:

* Чрезмерная сложность
* Ограничение по времени мешает исследованию
* Непроходимые участки из-за генерации



Рисунок 2.3 – Spelunky 2

The Binding of Isaac – Игра-эксперимент, черпающая вдохновение из The Legend of Zelda (1989), откуда был взят вид сверху, а также перемещение из комнаты в комнату. Задача предельно простая: зачищать подземелье, убивать противников, и не дать убить себя.

К положительным особенностям игры относятся:

* отзывчивое управление
* обилие предметов – 611
* реиграбельность
* множество персонажей, врагов и боссов

Однако у игры есть и недостатки:

* чрезмерная сложность
* игра ориентирована на изучение и заучивание
* гротескный визуальный стиль
* скудное описание предметов
* затянутые забеги



Рисунок 2.4 – The Binding of Isaac

Люди любят этот жанр за вариативность, реиграбельность и непредсказуемость, за перманентную смерть – начинать необходимо каждый раз с самого начала, за чувство исследования и совершенствования – даже полный рестарт не сможет отнять знаний игрока. Также жанр rogue-like смешивается с другими жанрами и темами, что создает новые интересные игры.

Однако вместе с этим жанром возникает множество проблем, которые необходимо решить. Случайная природа rogue-like очень часто создаёт непроходимые или очень сложные варианты. Требуется сбалансировать все случайности так, чтобы игра не ощущалась нечестной. А также донести до игрока, что проигрыш – лишь очередной шаг к победе.

Сложность и баланс таких игр – самая слабая часть. Игру нужно сделать так, чтобы игрок не умирал слишком часто, ведь тогда он просто выключит игру. Но и слишком простой игру делать нельзя, ведь развитие геймплея строится именно на проигрышах.

## Описание игры

Жанры: бесконечная, платформер, аркада, rogue-like.

Виртуальный мир был заражён вирусом, который постепенно его уничтожает. Главному герою требуется убежать как можно дальше от клетки вируса, которая его преследует по разрушающейся реальности.

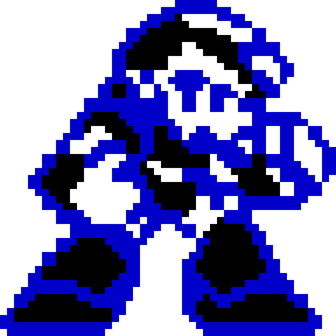


Рисунок 2.5 – главный герой

Персонаж имеет стандартный набор способностей:

* бег на стрелки;
* прыжок на пробел;
* быстрое падение на стрелку вниз;
* Z для рывка;
* X для блинка – телепортации вперёд на короткое расстояние и сквозь стены;
* В воздухе можно выполнить до приземления два дополнительных прыжка и два рывка/блинка;
* По стенам персонаж скользит и способен от них отталкиваться.

После старта генерируется уровень, по которому требуется двигаться игроку. Угроза начинает своё движение после небольшого таймера и преследует игрока по маршруту.

Во время игры каждый тик у игрока отнимаются очки. Касание вируса или достижение 0 очков грозит смертью. Уровень создаётся постепенно, с каждым дополнительным куском уровня игрок получает 100 очков.

Мир использует два элемента типа TileMap(карта из клеток) – большую для кусков уровня, определяющих направление движения и маленькую, которая уже строит сам уровень, на основе чанков.

Уровень состоит из заранее заготовленных секций 40х40 клеток, соответствующих всем вариантам направлений. Все чанки записываются в упорядоченный список. Карта имеет ограничение на количество чанков и генерируется только если главный герой находится на определённом расстоянии от конца. Это сделано, чтобы не загружать память компьютера лишними данными.



Рисунок 2.6 – вирус

Вирус использует список чанков, чтобы построить свой маршрут точно по пути игрока. Его скорость регулируется игрой, чтобы игра не была слишком простой или нечестной.

Аркадная составляющая игры подразумевает счётчик очков и времени в левом верхнем углу. Также существует временный дополнительный счётчик, показывающий сколько очков игрок получил в быстрой последовательности.

В таблице рекордов есть по 10 записей в каждой категории:

• Время жизни

• Пройденное расстояние

• Кол-во очков

• Кол-во очков, собранных в быстрой последовательности

Кроме этих данных в таблице указана частота использования рывков и блинков

# Практическая часть

## Подготовка к разработке

Все возможности главного героя необходимо записать и разделить на различные состояния: на земле, в воздухе, в рывке, в телепортации.

Для этого создаётся узел с новым классом CharacterStateMachine, содержащий несколько дочерних узлов, которые имеют класс State. Этот узел будет контролировать переключение между состояниями персонажа.

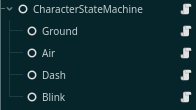


Рисунок 3.1 – все возможные состояния персонажа

Затем для каждого состояния пишется скрипт, описывающий их принцип работы.

Блинк – приём персонажа, позволяющий ему после небольшой подзарядки телепортироваться на заданное расстояние в указанную сторону. Ранние версии этого приёма позволяли игроку часто застревать в стене, в связи с чем был добавлен ещё один хитбокс.

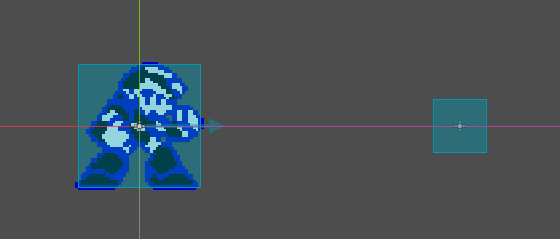


Рисунок 3.2 – персонаж и его хитбоксы в редакторе

Этот хитбокс передвигается в сторону, указанную игроком и, если не пересекается с другими хитбоксами, позволяет выполнить перемещение.

Также были созданы два файла, которые загружаются при запуске игры и исполняются всегда: шина сигналов и шина звуков. Шина сигналов необходимо для передачи данных между сценами, когда нет возможности подключить сигнал напрямую.

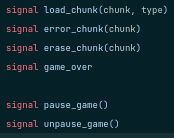


Рисунок 3.3 – список основных сигналов

Доступ к шине сигналов происходит через команды «Events.emit\_signal(«имя\_сигнала»,Значения)», а подключение происходит по команде «Events.имя\_сигнала.connect(self.название\_функции)»

Для персонажа нужно создать древо анимаций, которое будет учитывать все возможные варианты, переключения между анимациями. Так, например после рывка персонаж может оказаться в воздухе, а может оказаться на земле и анимация должна это отражать. Также, анимация прыжка состоит из двух частей.

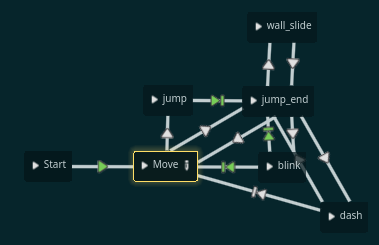


Рисунок 3.4 – древо анимаций персонажа

## Разработка геймплея

Самая важная часть игры – процедурно генерирующийся мир, который состоит из двух узлов TileMap.

Главная карта скрыта от игрока и используется игрой для определения маршрута. В ней хранится массив из клеток, каждая из которых соответствует определённой форме чанков

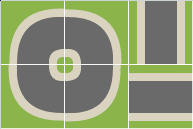


Рисунок 3.4 –отображение чанков, обозначающее начало и конец маршрута

Генерация строго линейная: начало новой клетки всегда совпадает с концом предыдущей. Также во избежание наложения маршрута самого на себя, перед генерацией идёт проверка соседних тайлов на наличие чанка там. Однако из-за случайной природы программы, иногда генерация замыкается в спирали, из-за чего всё же приходится переписывать часть маршрута.

С каждым сгенерированным чанком этот узел передаёт в другую карту сигнал, который содержит координаты и тип чанка. Вторая карта при запуске сперва сохраняет наборы клеток, чтобы потом их использовать на местах, указанных из карты чанков.



Рисунок 3.6 – наборы клеток, соответствующие чанкам

Вирус в игре выступает главной угрозой и мотивацией двигаться вперёд. Он начинает своё движение через 5 секунд после игрока, чтобы у того был шанс убежать как можно дальше. При приближении к игроку спрайт окрашивается полностью в красный цвет, чтобы показать игроку опасность.

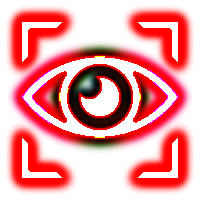


Рисунок 3.7 – вирус при приближении к игроку

Его движение определяется по трём пунктам: чанк, в котором вирус находится, следующий чанк в списке, позиция внутри чанка. В большинстве случаев вирус движется к центру чанка, выбирает следующий чанк, движется к нему.

Если чанк игрока совпадает с чанком, на который нацелен вирус, он будет двигаться не к центру, а к позиции игрока.

Скорость вируса также зависит от нескольких условий: если он отстаёт от игрока больше чем на 7 чанков по маршруту, он мгновенно передвигается пока не сократит расстояние достаточно. Если игрок находится выше вируса, вирус движется медленнее. Это сделано для того, чтобы у игрока был шанс спастись, так как обычная скорость вируса выше, чем скорость, с которой игрок может подниматься.

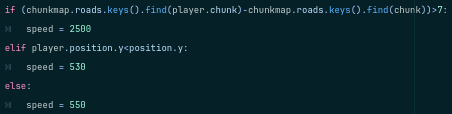


Рисунок 3.8 – формула скорости вируса

Персонаж использует встроенный класс CharacterBody2D, созданный специально для движущихся игровых объектов, на которые влияет гравитация. Кроме этого, у него есть дочерний узел Sprite, придающий ему внешний вид, узел CollisionShape, который определяет хитбокс персонажа, Area2D для проверки на возможность телепортации и RayCast2D для определения, находится ли игрок у стены, чтобы от неё можно было оттолкнуться. Также для анимации используются узлы AnimationTree и AnimationPlayer

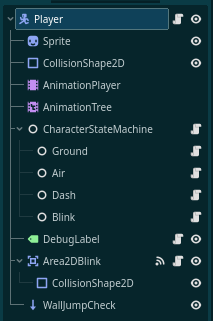


Рисунок 3.9 – полное древо узлов персонажа

## Разработка интерфейса

При запуске игры, игрока встречает меню из 3 основных кнопок

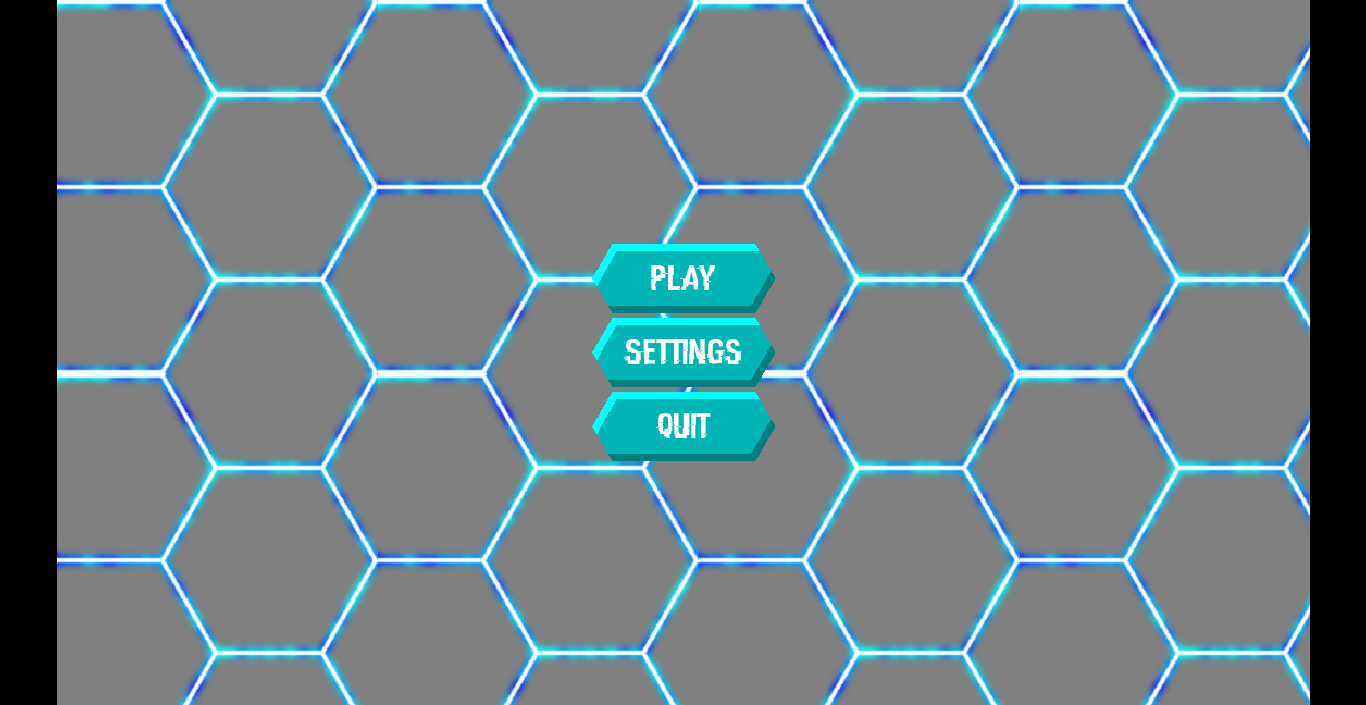


Рисунок 3.10 – главное меню игры

Кнопка Play активирует сцену с игровым процессом.

Кнопка Settings активирует сцену с настройками.

Кнопка Quit закрывает игру.

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.

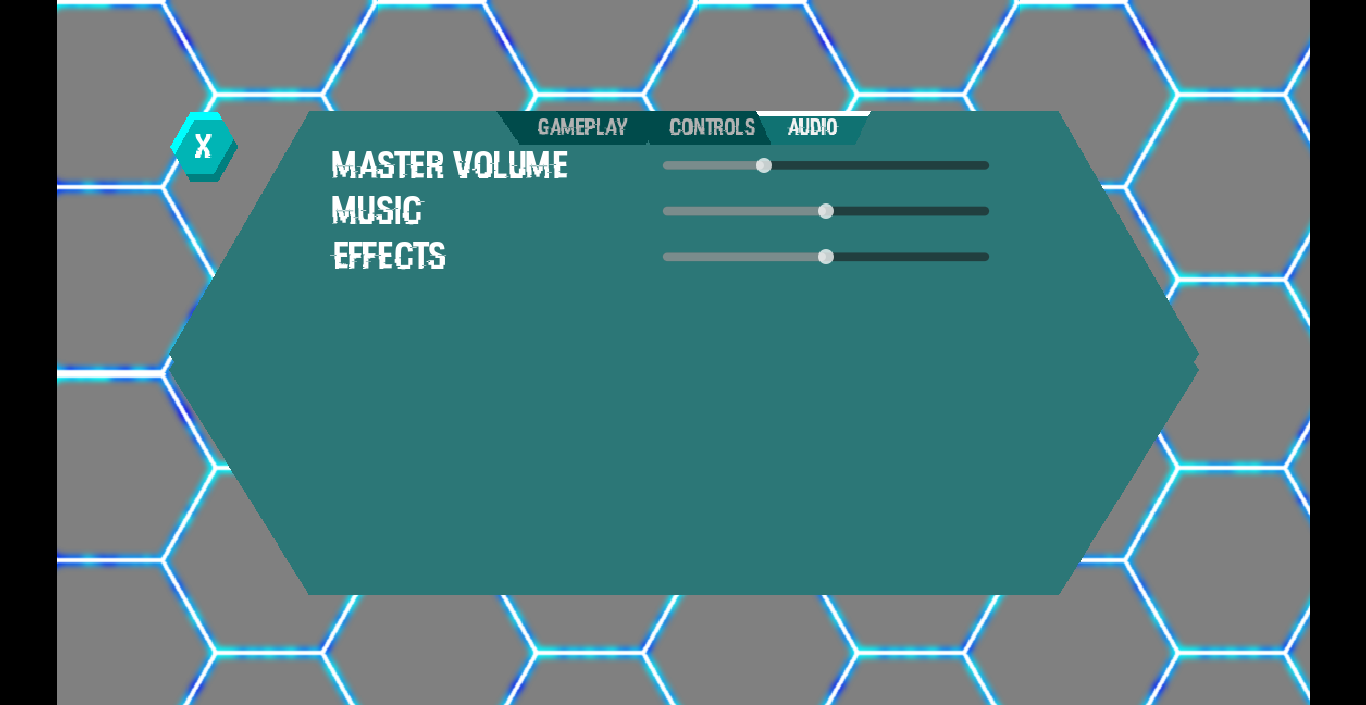


Рисунок 3.11 – Меню настроек

Во время игры нажатие на кнопку Esc вызывает меню паузы и приостанавливает игру.

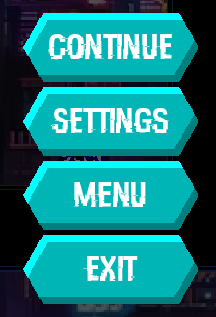


Рисунок 3.12 – Меню настроек

Кнопки Settings и Exit соответствуют кнопкам из главного меню.

Кнопка Menu заканчивает игру и переключает игрока в главное меню.

Кнопка Continue продолжает игру.

Во время игрового процесса в левом верхнем углу находятся два счётчика: счётчик времени, проведённого в этом забеге и счётчик очков, заработанных за забег

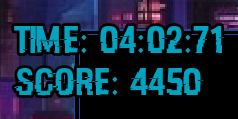


Рисунок 3.13 – интерфейс во время игры

Также с каждым новым сгенерированным чанком, появляется временный счётчик очков, показывающий, сколько очков игрок получил за новые чанки. Этот текст со временем становится прозрачнее, и если вызывается несколько раз за короткий промежуток времени, то показывает столько очков, сколько раз подряд он был вызван. Как только он снова станет полностью прозрачным, он сбросится обратно до «+100»

Внешний вид интерфейса настроен с помощью Темы интерфейса – файла ресурса, содержащего свойства такие свойства, как текстура, отступы, цвет и размер текста, шрифт. Этот файл используется чтобы сохранять один внешний вид для всех элементов одного типа. Так, для всех элементов имеющих фон используются шестиугольные текстуры

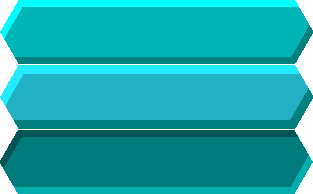


Рисунок 3.14 – текстуры кнопок: обычная, выделенная, нажатая

# ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы был получен обширный набор знаний. Разработка игр затрагивает различные сферы, включая программирование и решение проблем в целом. рынок сайтов-визиток и разработан желаемый стиль будущего сайта-визитка личного пользования, а также выполнены все поставленные ранее задачи:

* Изучены возможности для создания игр
* Проанализированы принципы процедурной генерации
* Написан дизайн-документ
* Собран необходимый список программ для работы
* Разработана первая версия игры

# Используемые источники

1. «5 INVALUABLE Godot 4 Code Patterns (feat. @Gdquest)» [Электронный ресурс]. Режим доступа: – https://youtu.be/UuEqUN98uZo (Дата обращения: 20.10.2023).
2. «Building a UI in godot» [Электронный ресурс]. Режим доступа: – https://youtu.be/i8ySmMGx--0?list=PL5N89Bry725MaCTKQRWhVoV58JCVikQqe (Дата обращения: 20.10.2023).
3. «Procedural Generation in Godot» [Электронный ресурс]. Режим доступа: – https://youtu.be/YShYWaGF3Nc?list=PLsk-HSGFjnaH82Bn6xbQNehatj3sIvtMQ (Дата обращения: 20.10.2023).
4. «State Machine Setup for 2D Platformer Character ~ Godot 4 GameDev Tutorial» [Электронный ресурс]. Режим доступа: – https://youtu.be/fuGiJdMrCAk (Дата обращения: 20.10.2023).
5. «Официальная документация Godot» [Электронный ресурс]. Режим доступа: – https://docs.godotengine.org/ru/4.x/ (Дата обращения: 20.10.2023).
6. «Официальная страница reddit Godot» [Электронный ресурс]. Режим доступа: – https://www.reddit.com/r/godot/ (Дата обращения: 20.10.2023).
7. «Официальный сайт GameMaker» [Электронный ресурс]. Режим доступа: – https://gamemaker.io/ru/blog/whats-the-best-game-making-software-for-beginners (Дата обращения: 20.10.2023).
8. «Официальный форум Godot» [Электронный ресурс]. Режим доступа: – https://forum.godotengine.org/ (Дата обращения: 20.10.2023).
9. Марек Хефнер. Дизайн-документ/Марек Хефнер//Лучшие компьютерные игры/ – 2004. – №11.
10. Хокинг, Джозеф. Unity — в действии. Мультиплатформенная разработка на C# — СПб : Питер, 2016. — 336 с.