

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки

09.03.02 «Информационные системы и технологии»

КУРСОВОЙ ПРОЕКТ

Дисциплина: Разработка игровых приложений под мобильные платформы

Тема: Разработка игрового мобильного приложения

Выполнил(а): студент(ка) группы **201-726**

_____ Курносова Арсения _____
(Фамилия И.О.)

Дата, подпись _____ 28.05.2023 _____
(Дата) (Подпись)

Проверил: _____ Колодочкин А. А., ст. пр. _____
(Фамилия И.О., степень, звание)

Дата, подпись _____
(Дата) (Подпись)

Замечания: _____

Москва

2022

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

*Факультет Информационных технологий
Кафедра Информатики и информационных технологий*

направление подготовки 09.03.02 «Информационные системы и технологии»,
профиль «Программное обеспечение игровой компьютерной индустрии»

УТВЕРЖДАЮ

Зав. каф., к.т.н. Е.В. Булатников

«__» _____ 2023 г. _____
(Подпись)

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

Студент Курносова Арсения Валерьевна группа **201-726**

(Фамилия, Имя, Отчество)

Тема Разработка игрового мобильного приложения

1. Срок представления работы к защите «__» июня 2023 г.

2. Исходные данные для выполнения работы:

исходные материалы

3. Содержание курсового проекта: введение, аналитическая часть, практическая часть, заключение, библиографический список

4. Перечень графического материала рисунки, таблицы, листинги

5. Литература и прочие материалы, рекомендуемые студенту для изучения:
указаны в библиографическом списке


6. Дата выдачи задания: «30» марта 2023 г.

7. Руководитель:

_____ / Колодочкин А. А. /

(Подпись)

8. Задание к исполнению принял:

Курносова А.В. / ФИО / 

Оглавление

Введение	5
ГЛАВА 1. Проектирование	5
Цель.....	5
Техническое задание	5
Описание предметной области.....	6
Выбор инструментов.....	6
ГЛАВА 2. Разработка мобильной игры.....	8
Разработка дизайн-документа	8
Основные элементы разработки	8
Графическое решение	9
Создание игрового пространства	9
Разработка дерева узлов.....	11
Основная структура узлов	11
Структура игрового узла.....	12
Узел TileMap	13
Узел Items	13
Узел живых объектов	14
Узел Player.....	15
Узел NPS.....	17
Структура пользовательского узла	19
Узел инвентаря.....	19
Узел меню паузы	21
Сцена главного меню	21
Сцена меню рестарта.....	21
Разработка кода.....	22
Действия проекта.....	22
Структура кода главного меню	23
Код игровой сцены	24
Код персонажей	24
Скрипт глобальных переменных	24
Скрипт живых объектов.....	25
Скрипт игрока	26
Скрипт неигрового персонажа	27
Код объектов	30
Скрипт генерации объектов.....	30
Код боевой механики	35
Скрипт области нанесения урона.....	35

Скрипт шкалы жизни	36
Код пользовательского интерфейса.....	36
Код инвентаря и его элементов	36
Скрипт инвентаря	36
Скрипт управления элементами инвентаря	37
Скрипт управления панелью инвентаря.....	38
Скрипт управления пользовательскими панелями	39
Скрипт меню паузы.....	39
Скрипт управления сохранениями.....	42
Код сцены рестарта	42
Разработка сцен	43
Заключение.....	45
Список литературы.....	46

Введение

Цель проекта – разработка мобильного игрового приложения. Игровое приложение выполнено в двумерном пространстве с пиксельной графикой, жанр игры - «survival».

Главный герой игры – молодой парень по прозвищу Кеклик, попадает в фантастический мир, представляющий из себя связку из множества островов, соединённых узкими тропинками. На первый взгляд, фантастический мир мало отличается от реальности, но чем дальше идёт главный герой, тем страннее и необычнее становится окружающее его пространство. В окружающем пространстве находится множество различных предметов: яблоки, веточки, грибы... Но помимо ресурсов, этот мир наполнен и необычными живыми существами – зелёными собаками, которые недружелюбно настроены по отношению к Кеклику.

Убегать, драться, лечиться найденными ресурсами и искать выход в реальность – всё, что остаётся Кеклику.

Пользователь играет за главного героя – Кеклика. Основной задачей игрока является не дать значению уровня жизни Кеклика опуститься до нуля. Для этого игроку необходимо добывать ресурсы, расположенные на карте, наносить урон врагам при помощи механики боя и избегать получения урона от враждебных NPS. Игра продолжается до тех пор, пока жив Кеклик.

ГЛАВА 1. Проектирование

Цель

Разработка мобильного игрового приложения в жанре «survival» в двумерном пространстве с пиксельной графикой

Техническое задание

Разработать мобильное игровое приложение, содержащее элементы:

- Двумерное игровое пространство (2D-мир);
- Главный герой (игровой персонаж)
- Игровые элементы для взаимодействия (Items);
- Возможность собирать игровые элементы;
- Инвентарь игрока с отображением собранных элементов;
- Неигровые персонажи (NPS);
- Механика боя (нанесение урона NPS и получение урона игроком);
- Меню игры (меню сохранения и загрузки, выхода в главное меню);
- Главное меню и меню рестарта.

Описание предметной области

Игровое приложение направлено на проведение досуга игрока. Окончанием игрового раунда является уменьшение шкалы здоровья персонажа до нуля, при этом игроку предоставляется возможность перезапустить игру с начала или загрузить одно из сохранений (если сохранения имеются).

Игра позволяет игроку погрузиться в фантастический мир и соотнести себя с главным персонажем игры. При прохождении игры, пользователь может выбрать различные тактики: «драться» с враждебными неигровыми персонажами или же убегать от них, продолжая сбор ресурсов.

Выбор инструментов

Для разработки был выбран движок Godot, это удобное и бесплатное приложение, не требующее много места, с хорошим интерфейсом и функционалом.

Godot Engine — открытый кроссплатформенный игровой движок для разработки 2D/3D-видеоигр и приложений для ПК, мобильных устройств, веб-платформ. Адаптирован ко всем распространенным операционным системам, включая Linux, macOS, Windows, Android и iOS

Игровой движок прост в освоении. Низкий порог вхождения позволяет начать разработку объёмных, полноценных по смысловой и функциональной нагрузке приложений даже при создании первого проекта.

Разработка на движке происходит на языке программирования GDScript, который является встроенным для данной среды.

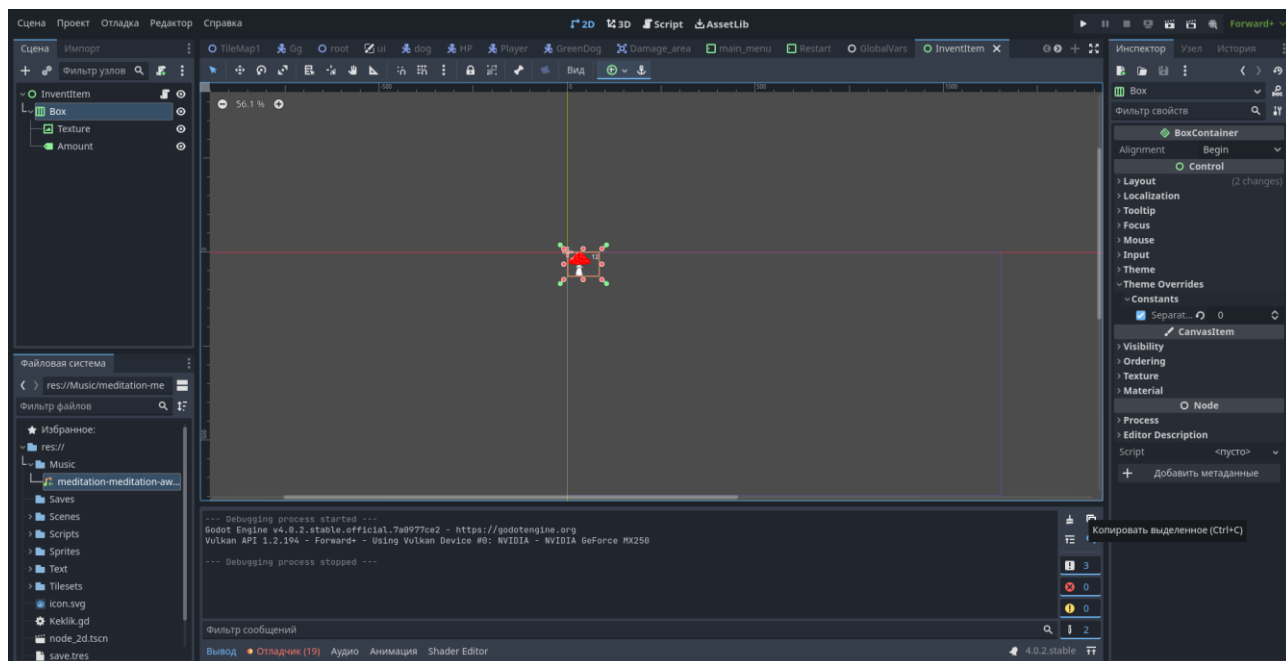


Рис.1 Сцена разработки приложения на игровом движке Godot

ГЛАВА 2. Разработка мобильной игры

Разработка дизайн-документа

Основные элементы разработки

При разработке приложения необходимо учитывать следующие параметры:

- Игра разрабатывается в жанре «выживание»;
- Двумерное игровое пространство;
- Пиксельная графика;
- Основная аудитория: 16+;
- Одиночный режим игры;
- Время прохождения не ограничено;
- Мобильная платформа разработки.

Для реализации функционала игры, приложение должно содержать следующие экраны:

- Экран главного меню
 - Старг игры;
 - Опции;
 - Выход
- Основной экран игры
 - Игровое пространство;
 - Окно инвентаря;
 - Меню паузы
 - Возвращение в игру;
 - Сохранение;
 - Загрузка;
 - Выход в главное меню
- Окно рестарта
 - Выход в главное меню
 - Рестарт (переход на основной экран игры)

Графическое решение

Игра имеет простую пиксельную графику. Такое визуальное решение позволяет упростить определение положения персонажа в двумерном пространстве.

Графика игры не реалистичная, преимущественно используются яркие, тёплые цвета.



Рис. 2 Спрайты игры

Создание игрового пространства

Игровым пространством является связка островов. Острова окружены водой и имеют узкие соединения между собой.

Игровое пространство реализовано с применением пиксельной графики. Для создания окружения разработан набор тайлов в соответствующей стилистике.

При разработке тайлов необходимо учитывать следующие параметры:

- Изображения тайлов должны быть одного, заранее выбранного размера, для избежания проблем с конвертацией каждого тайла в единый набор тайлов. В качестве размера единичной клетки тайловой карты был выбран размер 128x128 пикселей;

- Острова окружены водой со всех сторон, вследствие чего необходимо разработать тайлы для каждой стороны острова, или разработать автоматические тайлы;
- Границы островов должны быть хорошо видны для игрока.

Для реализации окружения был использован узел TileMap. Этот узел позволяет настроить единичный размер клетки будущей карты и заполнить сетку карты необходимыми тайлами.

Движок имеет встроенный инструмент конвертации спрайтов в тайлы карты, а набор тайлов может быть разбит на отдельные тайлы по сетке автоматически.

Игрок не должен иметь возможности «упасть» с острова. Для этого, при разработке набора тайлов, на определённые тайлы (тайлы воды) была добавлена физическая среда. Таким образом, тайлы воды являются препятствием для персонажа.



Рис.3 Тайлы зелёных островов



Рис.4 Тайлы розовых островов

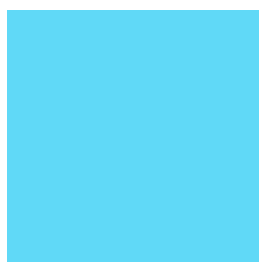


Рис.5 Тайл воды

Для создания наполнения мира (текстур на карте) был использован вышестоящий слой тайловой карты. Для создания вышестоящего слоя была использована настройка слоёв тайловой карты.

Для создания препятствий также была использована пиксельная графика. При реализации препятствий, на текстуры препятствий также была добавлена физическая среда.

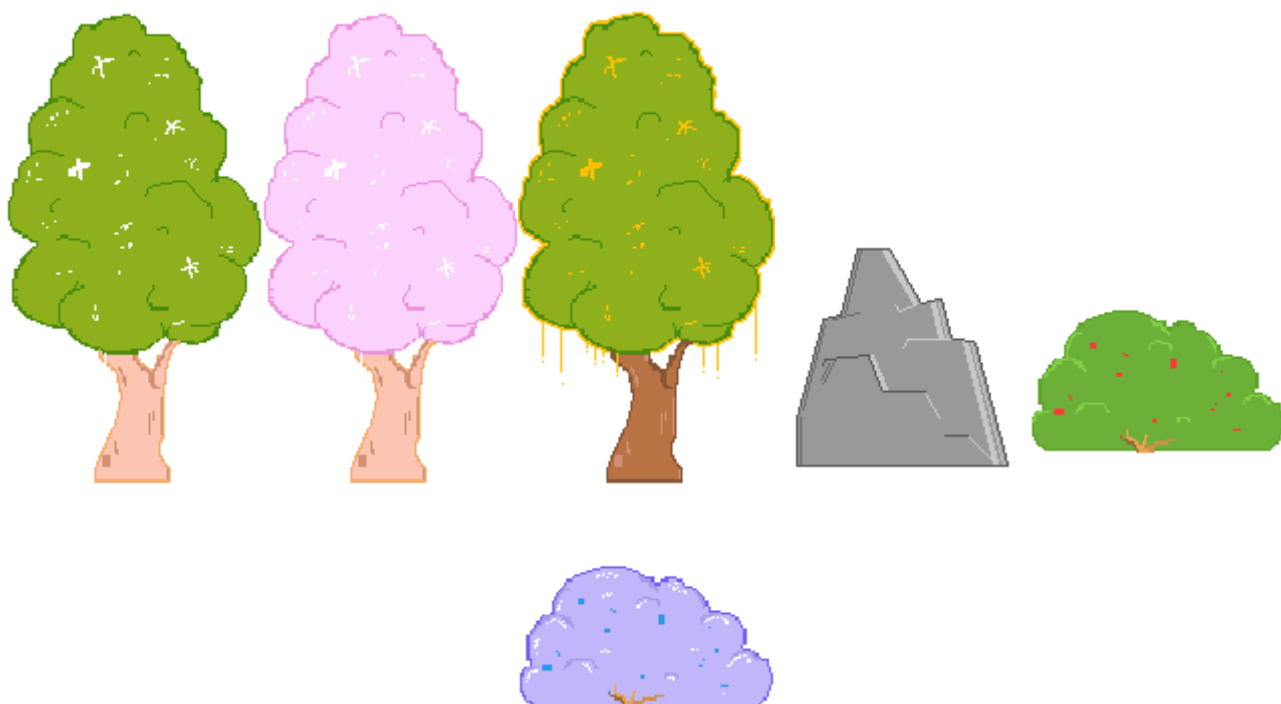


Рис.6 Тайлы препятствий

Разработка дерева узлов

Основная структура узлов

Общий массив узлов проекта может быть разбит на три основных узла:

1. Управляющий узел – узел, отвечающий за координацию остальных узлов;
2. Узел UI – узел, отвечающий за пользовательский интерфейс: главное меню, меню паузы и экран рестарта;
3. Игровой узел – узел, на котором располагается игровое окружение, игрок и неигровые персонажи.

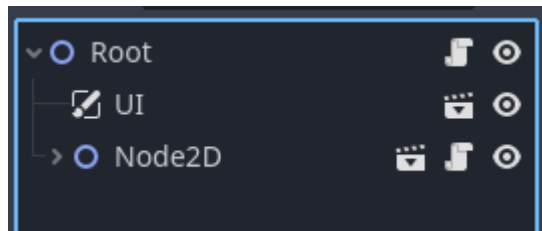


Рис.7 Реализация основной структуры узлов

Узел Root – управляющий узел, имеет тип Node2D. На управляющем узле находится скрипт, управляющий внутриузловыми процессами.

Узел UI – узел пользовательского интерфейса.

Узел Node2D – узел тайловой карты (игрового окружения).

Содержание каждого из основного узлов будет рассмотрено более подробно.

К игровому узлу, в управляющем узле, добавлен узел расположения NPS. Узел расположения NPS содержит сцену неигрового персонажа. Управление поведением неигрового персонажа содержится в дереве узлов сцены NPS.

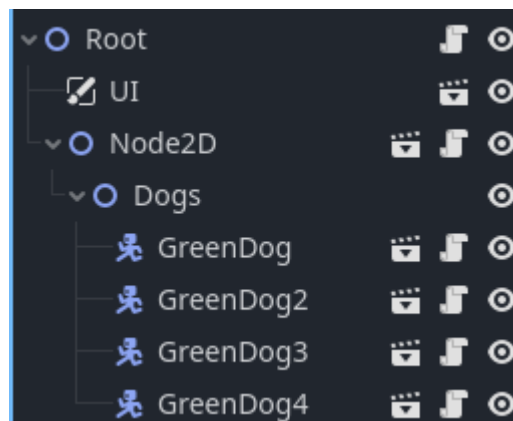


Рис.8 Развёрнутая структура управляющего узла

Структура игрового узла

В игровом узле содержится тайловая карта и сцена персонажа. Каждый узел игрового узла представлен отдельной сценой, содержащей более элементарные узлы.

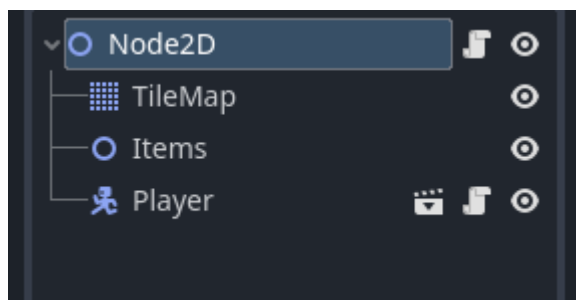


Рис.9 Основная структура игрового узла

Узел TileMap

Тайловая карта в игровом узле представлена элементов типа TileMap. Этот узел является встроенным в игровой движок. На данном узле соержатся оба слоя игрового пространства, а при его раскрытии отображается набор тайлов для тайловой карты и настройки тайловой карты.

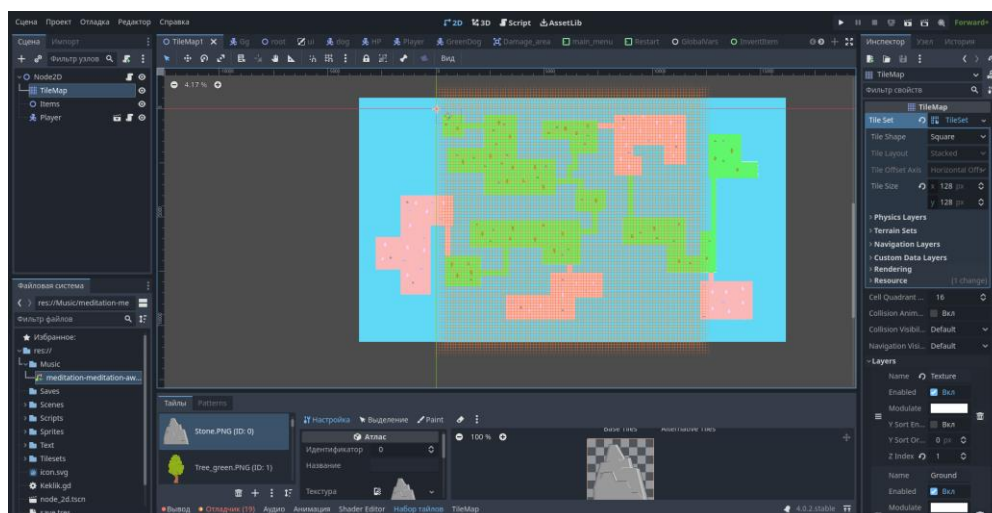


Рис.10 Узел тайловой карты

Узел Items

В узле items (узел атрибутов) содержится сцена атрибутов, которые генерируются рандомно на тайловой карте в районе островов. Узел представлен собранной сценой типа Node2D. Внутри сцена содержится два других узла:

- Item – управляющий узел атрибутов, на котором располагается скрипт, управляющий генерацией атрибутов на карте;
- Sprite – узел, содержащий шаблон изображений атрибутов. По данному шаблону отображаются все атрибуты на тайловой карте.

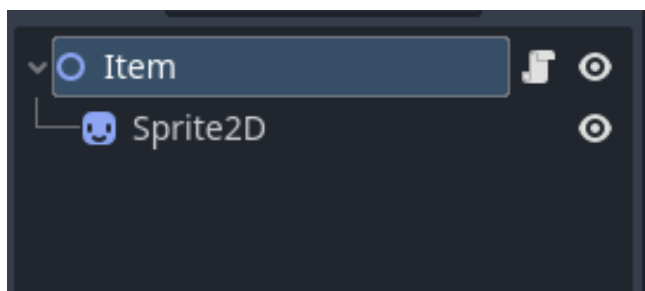


Рис.11 Узел атрибутов



Рис.12 Сцена узла атрибутов

Узел живых объектов

Узел живых объектов является родительским для всех живых объектов игрового пространства. Структура родительского узла наследуется дочерними сценами (сценой игрока и сценой неигрового персонажа).

Узел живых объектов содержит узлы, общие для дерева узлов игрового и неигрового персонажей.

Для создания дочерних сцен использовалось создание наследуемых сцен, от сцены живых существ.

Все живые существа содержат:

- Спрайт – изображение персонажа;
- Узел коллизии – узел, позволяющий сталкиваться живым существам с препятствиями;
- Узел Hit Box – позволяющий живым существам отслеживать область нанесения урона;
- Узел шкалы жизни – У каждого существа имеется своя шкала жизни.

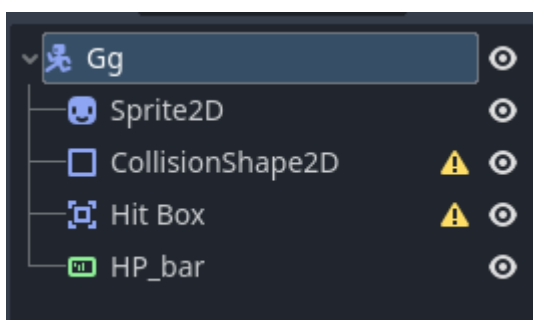


Рис.13 Структура узла живых объектов

Узел Player

Узел игрока является дочерним корневого узла «Gg», который управляет поведением всех живых существ в игровом пространстве.

В узле игрока содержится сцена игрока. Корневым узлом игрока является CharacterBody2D, на котором находится скрипт управления игроком.



Рис.14 Узел игрока

В корневом узле игрока находятся следующие узлы:

- Узел нанесения урона – узел, представленный сценой и отслеживающий положение мыши игрока. В момент нажатия на левую кнопку мыши, в допустимой области нанесения урона игроком по враждебным неигровым персонажам, появляется область нанесения урона, отслеживающая наличие NPS в своей области. При наличии враждебного неигрового персонажа в данной области, по NPS наносится урон;
- Узел анимации персонажа – содержит анимацию персонажа;
- Узел спрайта – содержит изображение персонажа;
- Узел коллизии – узел, содержащий область столкновения персонажа с препятствиями;
- Узел Hit Box – узел, содержащий область, по которой персонаж может получать урон;

- Узел камеры – узел, на котором располагается камера, привязанная к игровому персонажу. Камера позволяет следить за персонажем, при перемещении персонажа по игровому пространству;
- Узел уровня здоровья – узел, представленный сценой, содержащей изображение шкалы жизни персонажа и скрипт управления шкалой жизни;
- Узел музыки – узел, позволяющий проигрывать музыку в игре, пока в игре есть персонаж.

Далее представлены сцены и системы узлов внутри сцен для каждого из вышеперечисленных узлов:

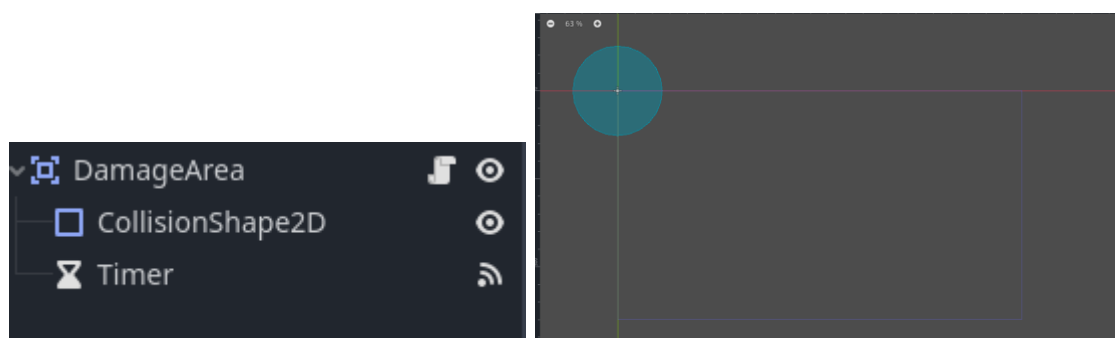


Рис.15 Узел нанесения урона

Сцена анимации содержит встроенные элементы анимации. При создании анимации, элементам анимации задаются названия, по которым к ним происходит обращение в коде. Для проигрывания анимации с нужной скоростью, настраивается параметр FPS, а для заикливания проигрывания анимации, устанавливается пункт «заикливать».

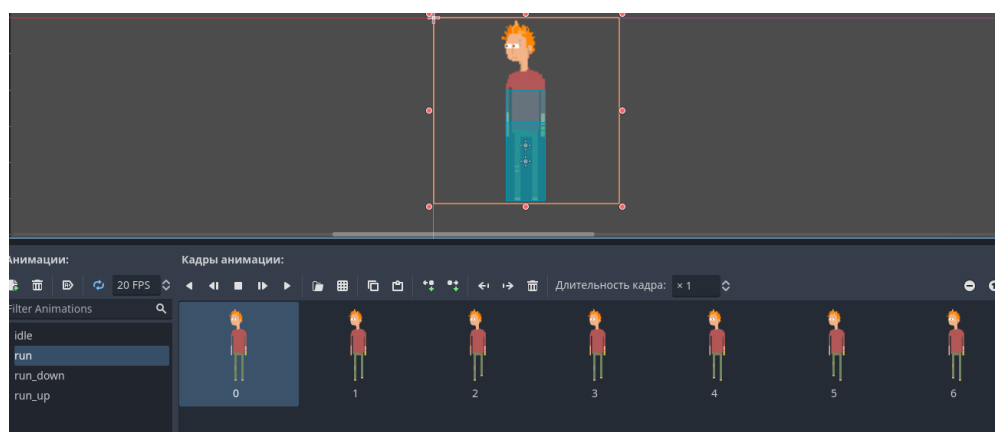


Рис.16 Узел анимации персонажа

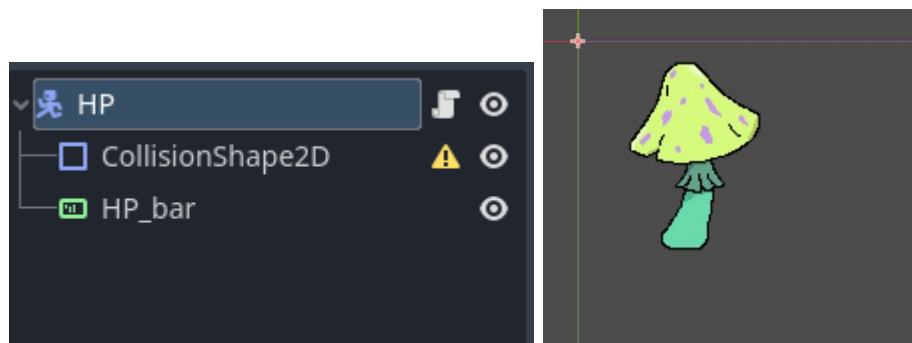


Рис.17 Узел уровня здоровья

Узел NPS

Узел неигрового персонажа является вторым дочерним узлом «Gg», который управляет поведением всех живых организмов в игровом пространстве. В управляющем узле, сцены неигровых персонажей собраны в отдельный (корневой для данной ветки сцен) узел «Dogs». Данная реализация позволяет управлять общим массивом неигровых персонажей.

Сцена NPS представлена следующим деревом узлов:

- Корневой узел – содержит скрип управления поведением неигрового персонажа;
- Узел анимации – содержит анимацию неигрового персонажа;
- Спрайт – содержит изображение NPS;
- Узел коллизии – содержит область столкновения с препятствиями;
- Hit Box – содержит область получения урона неигровым персонажем;
- Узлы таймеров – узлы, содержащие таймеры, для управления поведением NPS;
- Узел шкалы жизни – узел, содержащий сцену с изображением шкалы жизни и скриптом её управления;
- Узел области нанесения урона – область, в которой неигровой персонаж может наносить урон. Проверяет наличие игрового персонажа в области триггера и области нанесения урона. При наличии игрового персонажа в области, наносится урон.

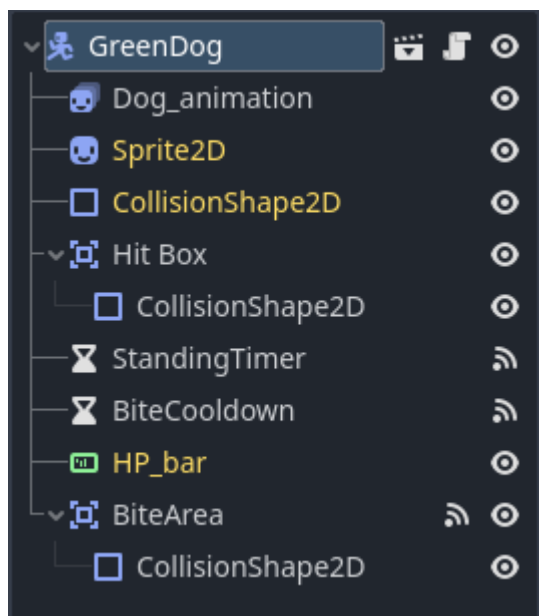


Рис.18 Дерево узлов NPS

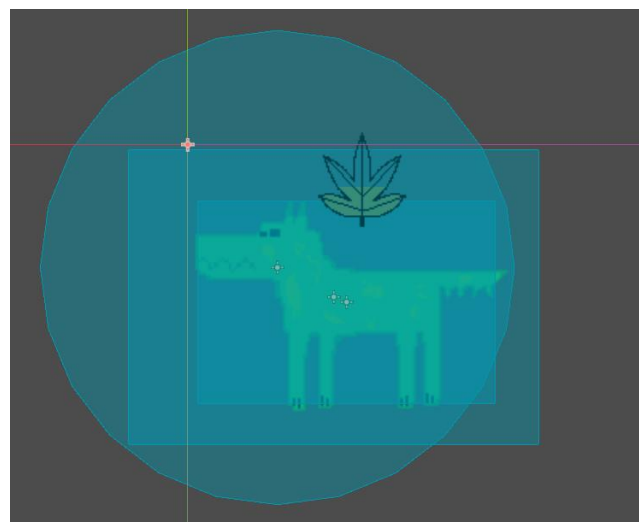


Рис.19 Сцена NPS

Наследуемые узлы представлены узлами, полученными от родительской сцены. Наследуемые узлы не имеют возможности быть изменены в сцене-наследнике. Для создания уникальных элементов в наследуемых узлах, в родительской сцене элементы создаются с базовыми (общими для всех живых элементов) настройками. Таким образом, спрайты внутри узлов сцен-наследников, анимации и размеры коллайдеров могут быть изменены.

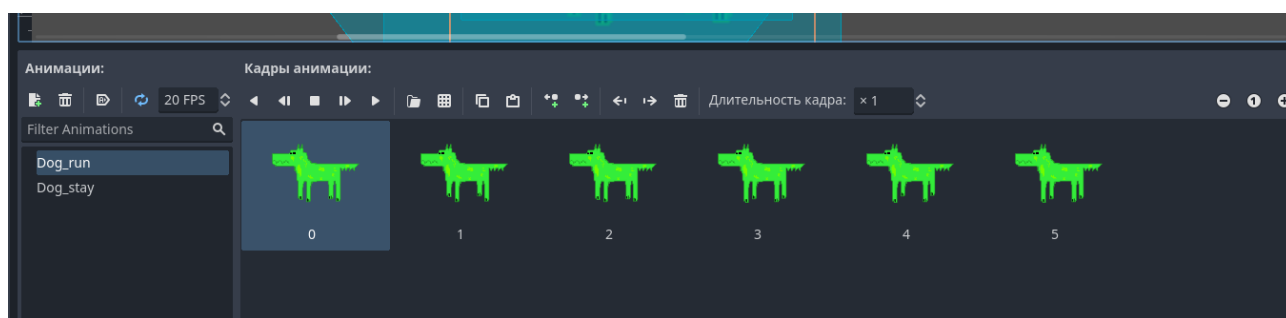


Рис.20 Узел анимации NPS

Для создания шкалы жизни, были добавлены два спрайта: спрайт рамки шкалы жизни и спрайт внутреннего наполнения шкалы жизни. Движок имеет встроенную функцию заполнения рамки шкалы жизни. При изменении параметра заполненности, на сцене можно посмотреть на изменение заполненности рамки шкалы жизни.

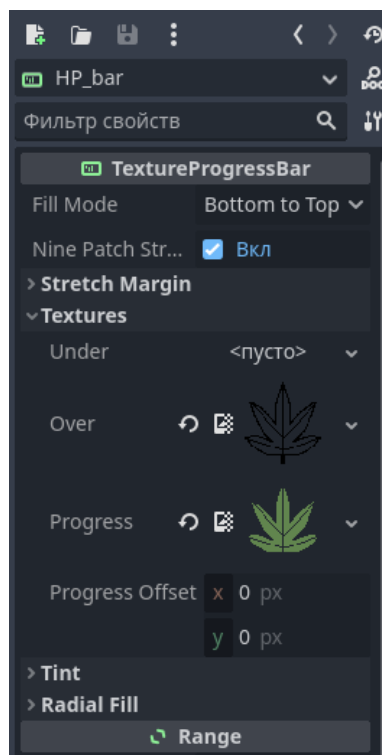


Рис.21 Узел шкалы жизни

Структура пользовательского узла

В пользовательском узле содержится корневой узел, служащий для отображения сцены. Внутри корневого узла содержится контрольный узел, содержащий инвентарь и меню паузы игры.

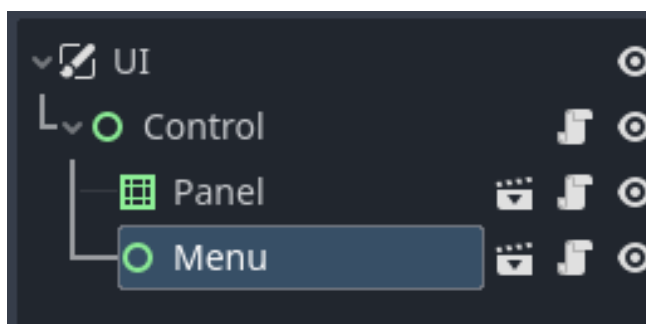


Рис.22 Пользовательский узел

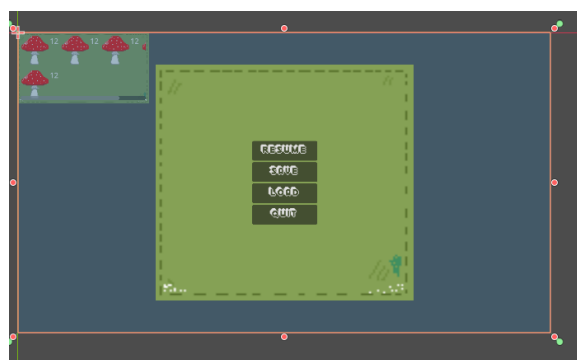


Рис.23 Сцена пользовательского узла

Узлы инвентаря и меню паузы представлены собранными сценами, содержащими дерево узлов.

Узел инвентаря

Узел инвентаря представляется деревом узлов. В дереве узлов содержится:

- Панель – корневой узел, содержащий остальные узлы инвентаря и содержащий скрипт управления элементами инвентаря;
- Узел scroll – узел, содержащий шаблоны элементов инвентаря и позволяющий прокручивать список элементов, содержащихся в инвентаре;
- Узел Grid – узел, создающий сетку расположения дочерних элементов, и позволяющий располагать элементы инвентаря в структурированном порядке;
- Элементы инвентаря – узел, представленный собранной сценой и содержащий шаблон элемента инвентаря.

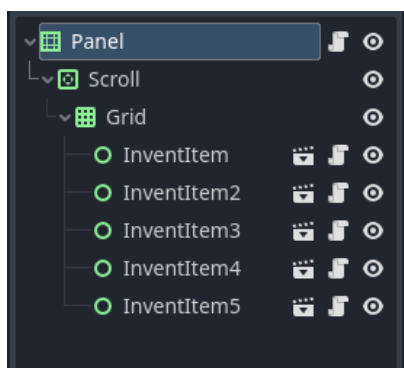


Рис.24 Дерево узлов инвентаря

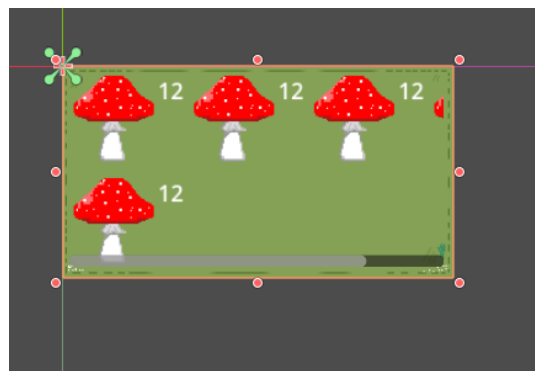


Рис.25 Сцена панели инвентаря

Шаблон элемента инвентаря содержит контейнер горизонтального расположения объектов, внутри которого содержатся текстура элемента инвентаря – изображение объекта, и Label – элемент, содержащий количество элементов данного типа в инвентаре.

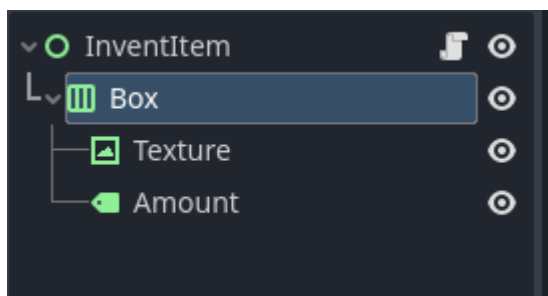


Рис.26 Шаблон элемента инвентаря



Рис.27 Сцена шаблона

Узел меню паузы

Узел меню паузы представлен деревом с структурой дочерних элементов. В меню паузы содержится структура расположения меню паузы. На корневом узле содержится скрипт управления нажатиями на кнопки меню.

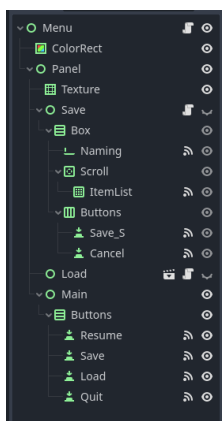


Рис.28 Структура узлов меню паузы



Рис.29 Сцена меню паузы

Сцена главного меню

Сцена главного меню представлено в виде отдельной сцены, переходом на которую управляет пользовательский узел в основном дереве узлов. В сцене главного меню содержится дерево структуры главного меню и скрипт управления нажатиями на кнопки в меню, содержащийся на корневом узле.

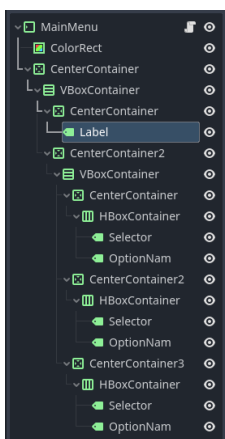


Рис.30 Дерево узлов главного меню



Рис.31 Сцена главного меню

Сцена меню рестарта

Сцена меню рестарта содержится отдельной сценой, а за управление перехода на неё отвечает скрипт, находящийся на корневом узле пользовательского узла.

В сцене меню рестарта содержится дерево узлов структуры меню рестарта.

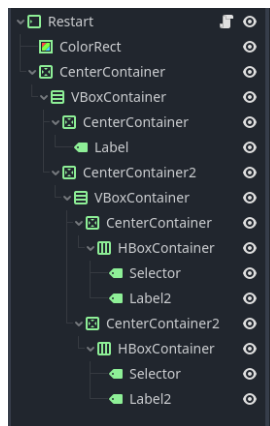


Рис.32 Дерево узлов меню рестарта



Рис.33 Сцена меню рестарта

Разработка кода

Действия проекта

Для создания функциональных кнопок в проекте, в проект были добавлены наименования действия и созданы связки клавиш с наименованиями действий.

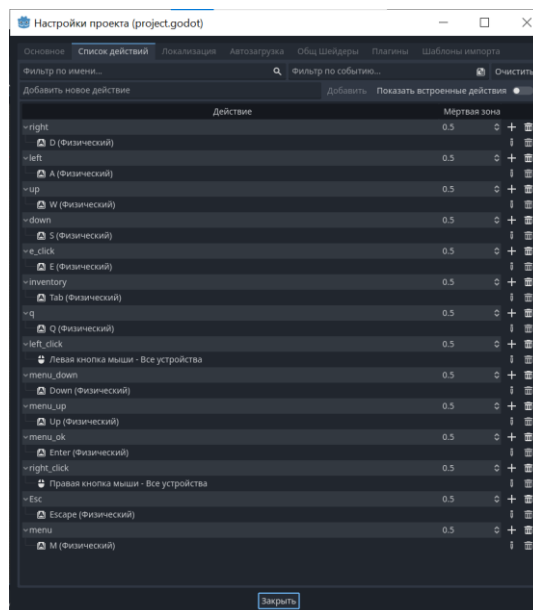


Рис.34 Настройки списка действий проекта

При разработке кода, для создания связи между клавишей и действием объекта игры, будет происходить обращение к действиям по их наименованиям в списке действий проекта.

Структура кода главного меню

Структура кода и система запуска функций будет описана в хронологическом порядке запуска игры.

Скрипт, содержащийся на корневом узле главного меню, отвечает за управление нажатием на клавиши управления меню и переход на сцены, соответствующие выбранным через пользовательский интерфейс.

Листинг 1. Скрипт главного меню

```
extends MarginContainer

const first_scene = preload("res://Scenes/root.tscn")

@onready var selector_one =
$CenterContainer/VBoxContainer/CenterContainer2/VBoxContainer/CenterContainer/HBoxContai
ner/Selector
@onready var selector_two =
$CenterContainer/VBoxContainer/CenterContainer2/VBoxContainer/CenterContainer2/HBoxCont
ainer/Selector
@onready var selector_three =
$CenterContainer/VBoxContainer/CenterContainer2/VBoxContainer/CenterContainer3/HBoxCont
ainer/Selector
var current_selection = 0

func _ready():
    set_current_selection(0)

func _process(delta):
    if Input.is_action_just_pressed("menu_down") and current_selection < 2:
        current_selection += 1
        set_current_selection(current_selection)
    elif Input.is_action_just_pressed("menu_up") and current_selection > 0:
        current_selection -= 1
        set_current_selection(current_selection)
    elif Input.is_action_just_pressed("menu_ok"):
        handle_selection(current_selection)

func handle_selection(_current_selection):
    if _current_selection == 0:
        get_parent().add_child(first_scene.instantiate())
        queue_free()

func set_current_selection(_current_selection):
    selector_one.text = ""
    selector_two.text = ""
    selector_three.text = ""
    if _current_selection == 0:
```

```

selector_one.text = ">"
    elif _current_selection == 1:
        selector_two.text = ">"
    elif _current_selection == 2:
        selector_three.text = ">"

```

При выборе пункта в главном меню, происходит переход на соответствующую сцену

Код игровой сцены

Игровая сцена включает в себя код игрового и неигрового персонажей, а также код генерации интерактивных предметов на карте. Помимо этого, при запуске игровой сцены, запускается код инвентаря и код меню паузы. Результаты работы этих скриптов заметны при открытии соответствующих панелей игроком.

Код персонажей

При переходе из главного меню к игре (при нажатии на «Start»), начинает работать структура скриптов.

Скрипт глобальных переменных

В скрипте глобальных переменных содержатся глобальные переменные, которые должны быть видимы для всех скриптов.

Листинг 2. Скрипт глобальных переменных

```

extends Node

@onready var entity_group = "ENTITY"
@onready var dog_group = "DOGS"

@onready var save_dir = "user://Saves"
@onready var save_temp = "%s.tres"

```


Скрипт живых объектов

Скрипт живых объектов является родительским скриптом для скриптов всех живых персонажей (игрока и неигровых персонажей игры). В скрипте содержатся общие элементы для всех живых объектов:

- Функция управления передачей данных в шкалу жизни;
- Функция смерти персонажа.

Листинг 3. Скрипт живых объектов

```
extends CharacterBody2D

@onready var ui = get_viewport().get_node("Root/UI/Control")

var speed = 8
@onready var hp = 50
@export var max_hp = 100
const restart = preload("res://Scenes/Buttons/Restart.tscn")

func _ready():
    set_start_hp(hp, max_hp)

func set_start_hp(hp, max_hp):
    $HP_bar.value = hp
    $HP_bar.max_value = max_hp

func update_hp():
    $HP_bar.value = hp

func toggle_hp_bar():
    $HP_bar.visible = !$HP_bar.visible

func die():
    get_parent().remove_child(self)
    queue_free()

func reduce_hp(val):
    self.hp -= int(val)
    update_hp()
    if self.hp <= 0:
        die()
    return false
    return true

func increase_hp(val):
    self.hp = min(self.hp + val, self.max_hp)
    update_hp()
```

Скрипт игрока

Скрипт игрока, помимо функций родительского скрипта, содержит уникальные функции:

- Функция перехода к сцене рестарта;
- Функция управления персонажем;
- Функция управления музыкой;
- Функция сбора интерактивных элементов окружения;
- Функция открытия инвентаря;
- Функция перехода к меню паузы.

Листинг 3. Скрипт игрока

```
extends "res://Scripts/Gg.gd"

var inventory = {}

func _ready():
    self.hp = 100
    set_start_hp(self.hp, self.max_hp)
    add_to_group(GlobalVars.entity_group)

func pick(item):
    var it = item.get_item()
    if it in inventory.keys():
        inventory[it] += item.get_amount()
    else:
        inventory[it] = item.get_amount()
    ui.update_inventory(inventory)

func _physics_process(delta):
    var velocity = Vector2.ZERO
    if Input.is_action_pressed("right"):
        velocity.x += speed
        $Gg_Animation.flip_h = true
        $Gg_Animation.play("run")
    elif Input.is_action_pressed("left"):
        velocity.x -= speed
        $Gg_Animation.flip_h = false
        $Gg_Animation.play("run")
    elif Input.is_action_pressed("up"):
        velocity.y -= speed
        $Gg_Animation.flip_h = false
        $Gg_Animation.play("run_up")
    elif Input.is_action_pressed("down"):
        velocity.y += speed
```

```

        $Gg_Animation.flip_h = false
        $Gg_Animation.play("run_down")
    else:
        $Gg_Animation.play("idle")
    move_and_collide(velocity)

    position.x = clamp(position.x, -3000, 16000)
    position.y = clamp(position.y, 0, 10500)

    $DamagePos.position = get_global_mouse_position() - position
    $DamagePos.position.x = clamp($DamagePos.position.x, - 24, 40)
    $DamagePos.position.y = clamp($DamagePos.position.y, - 24, 40)

func animate(velocity):
    var anim = "idle"

    if velocity.y > 0 or velocity.x > 0:
        anim = "run"

func _unhandled_input(event):
    if event.is_action_pressed("inventory"):
        ui.toggle_inventory(inventory)
    if Input.is_action_pressed("left_click"):
        var a = load("res://Scenes/Damage_area.tscn").instantiate()
        a.set_damage(20)
        get_parent().add_child(a)
        a.position = position + $DamagePos.position

func die():
    get_tree().change_scene_to_packed(restart)

```

Скрипт неигрового персонажа

Скрипт NPS, помимо функций родительского скрипта, содержит уникальные функции, такие как:

- Функция управления поведением неигрового персонажа;
- Функция поиска объекта атаки (игрока на карте в области видимости);
- Функция триггера на игрока.

```

extends "res://Scripts/Gg.gd"

var stands = true
var destination = Vector2()
var velocity_dog = Vector2()

```

```

var prev_pos = Vector2()
var target = null
var default_speed = 2

var target_intercepted = false
var can_bite = true
var bite_strenght = 10

func _physics_process(delta):
    if velocity_dog:
        prev_pos = position
        move_and_collide(velocity_dog)
        position.x = clamp(position.x, -3000, 15000)
        position.y = clamp(position.y, 0, 10000)
        wander()
        search_for_target()

    if target_intercepted and can_bite:
        bite(target)

func search_for_target():
    var pl = get_parent().get_parent().get_player()
    if pl:
        if position.distance_to(pl.position) < 420:
            cancel_movement()
            speed = default_speed * 2 if speed == default_speed else speed
            target = pl
        else:
            if target:
                cancel_movement()
                target = null
            if target:
                set_destination(target.position)

func _ready():
    speed = default_speed
    self.hp = 100
    self.max_hp = 100
    set_start_hp(self.hp, self.max_hp)
    add_to_group(GlobalVars.entity_group)
    add_to_group(GlobalVars.dog_group)

func set_destination(dest):
    destination = dest
    velocity_dog = (destination - position).normalized() * speed

    if velocity_dog.x > 0:

```

```

        $Dog_animation.flip_h = true
    else:
        $Dog_animation.flip_h = false
    $Dog_animation.play("Dog_run")

    stands = false

func cancel_movement():
    velocity_dog = Vector2()
    destination = Vector2()
    speed = default_speed
    $Dog_animation.play("Dog_stay")
    $StandingTimer.start(2)

func wander():
    var pos = position
    if stands:
        randomize()
        var x = int(randf_range(pos.x - 200, pos.x + 200))
        var y = int(randf_range(pos.x - 200, pos.x + 200))

        x = clamp(x, -3000, 15000)
        y = clamp(y, 0, 10000)

        set_destination(Vector2(x, y))

    elif velocity_dog != Vector2():
        if pos.distance_to(destination) <= speed:
            cancel_movement()
        elif pos.distance_to(prev_pos) <= 0.6:
            cancel_movement()

func bite(targ):
    var is_alive = targ.reduce_hp(bite_strenght)
    can_bite = false
    $BiteCooldown.start(1)
    if not is_alive:
        cancel_movement()

func _on_standing_timer_timeout():
    stands = true
    pass # Replace with function body.

func _on_bite_cooldown_timeout():
    can_bite = true
    pass # Replace with function body.

func _on_bite_area_area_entered(area):

```

```

if area.get_parent() == target:
    target_intercepted = true
pass # Replace with function body.

func _on_bite_area_area_exited(area):
    if area.get_parent() == target:
        target_intercepted = false
    pass # Replace with function body.

```

Код объектов

При запуске игровой сцены происходит генерация интерактивных объектов на карте.

Скрипт генерации объектов

Интерактивные объекты на карте генерируются случайно. Их генерация ограничена по группам: Группа объектов для генерации на зелёных островах и группа объектов для генерации на розовых островах.

```

extends Node2D

@onready var item = preload("res://Scenes/item.tscn")

func get_player():
    if has_node("Player"):
        return $Player
    else:
        return false

func _ready():
    var items = ["mushroom1", "apple1", "stik1"]
    var items2 = ["mushroom2", "apple2", "stik2"]
    for i in range(8):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(300,1250)), int(randf_range(300,
1250)))
    for i in range(14):
        randomize()

```

```

        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(2300,3650)), int(randf_range(300,
3900)))
    for i in range(14):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(510,2200)), int(randf_range(1800,
2900)))
    for i in range(8):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(3750,5450)), int(randf_range(2250,
3850)))
    for i in range(6):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(4800,7400)), int(randf_range(510,
1300)))
    for i in range(4):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(6150,7450)), int(randf_range(1400,
2000)))
    for i in range(4):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(6450,7650)), int(randf_range(3750,
4650)))
    for i in range(16):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()

```

```

        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(1600, 6600)), int(randf_range(5150,
6840)))
    for i in range(6):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(4950, 6660)), int(randf_range(6350,
7100)))
    for i in range(4):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(450, 2050)), int(randf_range(6800,
8150)))
    for i in range(10):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(8620, 11500)), int(randf_range(5050,
5250)))
    for i in range(6):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(10700, 11500)), int(randf_range(5050,
7500)))
    for i in range(6):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(11500, 11900)), int(randf_range(6800,
7500)))
    for i in range(6):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])

```



```

        new_item.position = Vector2(int(randf_range(12200, 12900)), int(randf_range(5700,
6800)))
    for i in range(6):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(12600, 14050)), int(randf_range(1200,
3150)))
    for i in range(6):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items[a])
        new_item.position = Vector2(int(randf_range(14100, 14800)), int(randf_range(2450,
3150)))

    for i in range(6):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items2[a])
        new_item.position = Vector2(int(randf_range(-2750,-1680)), int(randf_range(5900,
7350)))
    for i in range(20):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items2[a])
        new_item.position = Vector2(int(randf_range(-1600,-270)), int(randf_range(4000,
8500)))
    for i in range(6):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items2[a])
        new_item.position = Vector2(int(randf_range(-200,750)), int(randf_range(4000,
5300)))
    for i in range(20):
        randomize()
        var a = int(randf_range(0, 3))
        var new_item = item.instantiate()
        $Items.add_child(new_item)
        new_item.set_item(items2[a])

```

```

        new_item.position = Vector2(int(randf_range(3250,6350)), int(randf_range(8600,
9700)))
        for i in range(14):
            randomize()
            var a = int(randf_range(0, 3))
            var new_item = item.instantiate()
            $Items.add_child(new_item)
            new_item.set_item(items2[a])
            new_item.position = Vector2(int(randf_range(5300, 7600)), int(randf_range(7500,
8600)))
        for i in range(10):
            randomize()
            var a = int(randf_range(0, 3))
            var new_item = item.instantiate()
            $Items.add_child(new_item)
            new_item.set_item(items2[a])
            new_item.position = Vector2(int(randf_range(12200, 14550)), int(randf_range(8000,
9650)))
        for i in range(16):
            randomize()
            var a = int(randf_range(0, 3))
            var new_item = item.instantiate()
            $Items.add_child(new_item)
            new_item.set_item(items2[a])
            new_item.position = Vector2(int(randf_range(8250, 9050)), int(randf_range(300,
3000)))
        for i in range(16):
            randomize()
            var a = int(randf_range(0, 3))
            var new_item = item.instantiate()
            $Items.add_child(new_item)
            new_item.set_item(items2[a])
            new_item.position = Vector2(int(randf_range(9150, 10500)), int(randf_range(300,
1600)))
        for i in range(10):
            randomize()
            var a = int(randf_range(0, 3))
            var new_item = item.instantiate()
            $Items.add_child(new_item)
            new_item.set_item(items2[a])
            new_item.position = Vector2(int(randf_range(10550, 11500)), int(randf_range(300,
2500)))
        for i in range(16):
            randomize()
            var a = int(randf_range(0, 3))
            var new_item = item.instantiate()
            $Items.add_child(new_item)
            new_item.set_item(items2[a])
            new_item.position = Vector2(int(randf_range(9100, 9450)), int(randf_range(2200,
3000)))

```

```

func _unhandled_input(event):
    if event.is_action_pressed("q"):
        for i in get_tree().get_nodes_in_group(GlobalVars.entity_group):
            i.toggle_hp_bar()

```

Код боевой механики

Код боевой механики задействуется всеми живыми персонажами в игровом пространстве.

Скрипт области нанесения урона

При нажатии на левую кнопку мыши, игрок может наносить урон по NPS, если неигровой персонаж находится в области нанесения урона. Для генерации области нанесения урона используется скрипт области нанесения урона.

Листинг 6. Скрипт области нанесения урона

```

extends Area2D

var damaged = false
var damage = 0

func _ready():
    $Timer.start(0.1)
    pass

func set_damage(dmg):
    damage = dmg

func _physics_process(delta):
    if not damaged and get_overlapping_bodies() != []:
        for i in get_overlapping_bodies():
            if i in get_tree().get_nodes_in_group(GlobalVars.dog_group):
                i.reduce_hp(damage)
        damaged = true

func _on_timer_timeout():
    get_parent().remove_child(self)
    queue_free()
    pass # Replace with function body.

```

Скрипт шкалы жизни

Для управления шкалой жизни используется соответствующий скрипт. Скрипт содержит функции передачи и обновления данных в HP Bar.

Листинг 7. Скрип управления шкалой жизни

```
extends TextureProgressBar

@onready var ui = get_viewport().get_node("Root/UI/Control")
@onready var player = $".."

var hp = 50
@export var max_hp = 100

func _ready():
    set_start_hp(player.hp_gg, max_hp)

func set_start_hp(hp, max_hp):
    value = hp
    max_value = max_hp

func update_hp():
    value = player.hp_gg
```

Код пользовательского интерфейса

Для корректной работы инвентаря используется система скриптов, каждый из которых отвечает за свою область функций.

Код инвентаря и его элементов

В коде инвентаря и его элементов содержится структура скриптов по управлению отображения и обновления информации в инвентаре.

Скрипт инвентаря

Для управления инвентарём используется скрипт инвентаря. Скрипт инвентаря содержит функции обновления элементов инвентаря. Код содержит ссылку на скрипт управления элементами инвентаря.

Все элементы инвентаря представлены готовой сценой, которая служит шаблоном отображения добавляемых в инвентарь объектов. Скрип содержит ссылку на сцену.

```

extends Control

var item_name = ""
var item_amount = 0
var properties = {}

func set_item(item_name, amount, props):
    self.item_name = item_name
    self.item_amount = item_amount
    self.properties = props.duplicate()

    $Box/Texture.texture = load("res://Sprites/%s.PNG" %item_name)
    $Box/Amount.text = str(amount)

func get_item_name():
    return item_name

func add_amount(am):
    item_amount += am
    $Box/Amount.text = str(item_amount)

func get_amount():
    return item_amount

func get_props():
    return properties

func can_stack():
    return properties["can_stack"]

func save():
    return [item_name, item_amount, properties]

```

Скрипт управления элементами инвентаря

Для управления инвентарём, необходимо управлять элементами инвентаря. Элементы инвентаря должны получать данные о поднятом игроком предмете, после чего должно происходить обновление информации в инвентаре. Скрипт управления элементами инвентаря содержит соответствующие функции.

```

extends Node2D

@onready var pre_inv = preload("res://Scenes/InventItem.tscn")

var properties = {}
var item = ""
var amount = 1

```

```

var stack_limit = 8

func set_item(props):
    $Sprite2D.texture = load("res://Sprites/%s.PNG" % props[0])
    item = props[0]
    stack_limit = props[1]
    self.properties = props[2]

# Called when the node enters the scene tree for the first time.
func _ready():
    pass # Replace with function body.

func get_item():
    return item

func get_amount():
    return amount

func set_amount(am):
    amount = am

func _input(event):
    if event.is_action_pressed("e_click"):
        var pl = get_parent().get_parent().get_player()
        if abs(pl.position.x - position.x) < 128 and abs(pl.position.y - position.y) < 128:
            var new_item = pre_inv.instantiate()
            new_item.set_item(item, amount, properties)
            var is_picked = pl.pick(new_item)
            if is_picked:
                get_parent().remove_child(self)
                queue_free()
            else:
                new_item.queue_free()

```

Скрипт управления панелью инвентаря

Помимо управления самим инвентарём, необходим скрипт, связывающий действия игрока с данными, содержащимися в инвентаре. Для этого служит скрипт управления панелью инвентаря.

Листинг 10. Скрипт управления панелью инвентаря

```

extends NinePatchRect

@onready var item = preload("res://Scenes/InventItem.tscn")
@onready var container = $Scroll/Grid

func _ready():
    clear()

```

```

    visible = false

func clear():
    for i in container.get_children():
        container.remove_child(i)
        i.queue_free()

func show_inventory(inventory):
    clear()
    for i in inventory.keys():
        var new_item = item.instantiate()
        container.add_child(new_item)
        new_item.set_item(i, inventory[i])

```

Скрипт управления пользовательскими панелями

Для управления всеми пользовательскими панелями используется скрипт управления пользовательскими панелями.

Листинг 11. Скрипт управления пользовательскими панелями

```

extends Control

@onready var pack = $Panel

func toggle_inventory(inventory):
    if pack.visible:
        pack.clear()
        pack.visible = false
    else:
        pack.visible = true
        pack.show_inventory(inventory)

func update_inventory(inventory):
    if pack.visible:
        pack.show_inventory(inventory)

func _unhandled_input(event):
    if event.is_action_pressed("menu"):
        $Menu.open()

```

Скрипт меню паузы

В коде инвентаря и его элементов Скрипт управления меню паузы содержит функции распознавания нажатий на кнопки меню и функции действий, происходящих при нажатии на кнопки.

```

extends Control

const menu = preload("res://Scenes/Buttons/main_menu.tscn")
var save_name: set = _set_name
@onready var save_file = preload("res://Scripts/Resources/SaveData.gd")

func _ready():
    update_saves()
    hide()
    pass

func open():
    update_saves()
    $Panel/Save/Box/Naming.text = ""
    if visible:
        hide()
        get_tree().paused = false
    else:
        show()
        get_tree().paused = true
        $Panel/Load.hide()
        $Panel/Save.hide()
        $Panel/Main.show()

func _on_resume_pressed():
    open()
    pass # Replace with function body.

func _on_quit_pressed():
    get_tree().change_scene_to_packed(menu)
    pass # Replace with function body.

func _on_save_pressed():
    $Panel/Main.hide()
    $Panel/Save.open()
    pass # Replace with function body.

func _on_load_pressed():
    $Panel/Main.hide()
    $Panel/Load.open()
    pass # Replace with function body.

func _on_cancel_pressed():
    $Panel/Save.hide()
    $Panel/Main.show()
    pass # Replace with function body.

func _on_naming_text_changed(new_text):
    _set_name(new_text)

```



```

pass # Replace with function body.

func _set_name(n):
    save_name = n

func _on_item_list_item_selected(index):
    change_line(index)
    pass # Replace with function body.

func change_line(ind):
    var sname = $Panel/Save/Box/Scroll/ItemList.get_item_text(ind)
    $Panel/Save/Box/Naming.text = sname
    _set_name(sname)

func update_saves():
    $Panel/Save/Box/Scroll/ItemList.clear()
    var dir = DirAccess.open("res://")
    if not dir.dir_exists(GlobalVars.save_dir):
        dir.make_dir_recursive(GlobalVars.save_dir)
    dir.change_dir(GlobalVars.save_dir)

    dir.list_dir_begin()
    var file = dir.get_next()
    while file != "":
        $Panel/Save/Box/Scroll/ItemList.add_item(file.slit('.')[0])
        file = dir.get_next()
    dir.list_dir_end()

func save():
    if save_name != "":
        #var file = FileAccess.open("res://Saves", FileAccess.WRITE)
        var file = save_file.new()
        file._set_name(save_name)
        var save_path = FileAccess.open(GlobalVars.save_temp % save_name,
FileAccess.WRITE)
        # $Panel/Save/Box/Scroll/ItemList.add_item(save_path)
        # ResourceSaver.save(save_path, file)

        $Panel/Main.show()
        $Panel/Save.hide()

func _on_save_s_pressed():
    save()
    pass # Replace with function body.

```

Скрипт управления сохранениями

Для управления сохранениями используется глобальный скрипт управления сохранениями. Он содержит глобальные функции, одинаковые для всех функций сохранений (сохранение инвентаря, сохранение состояний игровых и неигровых персонажей, сохранение состояний интерактивных элементов на карте).

Листинг 13. Скрипт управления сохранениями

```
extends Resource

var save_name: String = "": set = _set_name
var data = { }: set = _set_data, get = _get_data

func _set_data(new_data):
    data = new_data.duplicate(true)

func _get_data():
    return data

func _set_name(n):
    save_name = n
```

Код сцены рестарта

При проигрыше, пользователь попадает на сцену рестарта. Скрипт сцены рестарта содержит функции управления меню рестарта (распознавание нажатий на кнопки и переход на соответствующие нажатию сцены).

Листинг 14. Скрипт рестарта

```
extends MarginContainer

const first_scene = preload("res://Scenes/root.tscn")
const menu = preload("res://Scenes/Buttons/main_menu.tscn")

@onready var selector_one =
$CenterContainer/VBoxContainer/CenterContainer2/VBoxContainer/CenterContainer/HBoxContainer/Selector
@onready var selector_two =
$CenterContainer/VBoxContainer/CenterContainer2/VBoxContainer/CenterContainer2/HBoxContainer/Selector
var current_selection = 0

func _ready():
    set_current_selection(0)
```

```

func _process(delta):
    if Input.is_action_just_pressed("menu_down") and current_selection < 1:
        current_selection += 1
        set_current_selection(current_selection)
    elif Input.is_action_just_pressed("menu_up") and current_selection > 0:
        current_selection -= 1
        set_current_selection(current_selection)
    elif Input.is_action_just_pressed("menu_ok"):
        handle_selection(current_selection)

func handle_selection(_current_selection):
    if _current_selection == 0:
        get_parent().add_child(first_scene.instantiate())
        queue_free()
    if _current_selection == 1:
        get_parent().add_child(menu.instantiate())
        queue_free()

func set_current_selection(_current_selection):
    selector_one.text = ""
    selector_two.text = ""
    if _current_selection == 0:
        selector_one.text = ">"
    elif _current_selection == 1:
        selector_two.text = ">"

```

Разработка сцен

При разработке игровой сцены, все сцены, с соответствующими скриптами, были собраны в одну, основную сцену.

Основная сцена выглядит следующим образом:



Рис. 35 Основная сцена проекта

Основной, корневой сценой проекта является сцена главного меню. При запуске проекта, игрок попадает в главное меню, откуда может перейти к сцене игры. Перемещать курсор в главном меню пользователь может на стрелки клавиатуры. При переходе в сцену игры, пользователь может:

- Перемещать персонажа - WASD;
- Скрыть или открыть шкалу здоровья - Q;
- Открыть или закрыть инвентарь – Tab;
- Наносить урон NPS – ЛКМ;
- Открыть меню паузы – M.

В меню паузы игрок может сохраниться, выйти в главное меню, продолжить играть или загрузить прошлое сохранение.

Заключение

В результате выполнения проекта была разработана мобильная игра в жанре «survival», с пиксельной графикой, в двумерном пространстве.

В игре задействовано множество различных механик:

- Механика боя;
- Механики перемещения игровых и неигровых персонажей по карте;
- Механики взаимодействия пользователя с пользовательским интерфейсом;
- Механики сбора интерактивных элементов на карте;
- Механика инвентаря.

Разнообразие механик позволяет дополнять и совершенствовать игру по мере развития проекта.

В процессе выполнения проекта были выполнены следующие задачи:

- Разработка тайловой карты;
- Разработка и добавление анимаций для игрового и неигровых персонажей;
- Разработан инвентарь;
- Разработана рандомная генерация интерактивных элементов на карте;
- Разработана шкала жизни и её корректная работа;
- Разработана механика боя;
- Разработана механика игровой паузы;
- Разработано главное меню и меню рестарта;
- Добавлена музыка.

Список литературы

1. Уроки разработки на платформе Godot [Электронный ресурс]. -
https://www.youtube.com/watch?v=OG_fXf1uJL0&list=LL&index=5
2. Документация Godot [Электронный ресурс]. -
<https://docs.godotengine.org/en/stable/>
3. Информация об изменениях в Godot 4 [Электронный ресурс].-
<https://www.reddit.com/>
4. Изменения Godot 4 [Электронный ресурс]. -
<https://www.youtube.com/watch?v=5R1xgo-wz1Y>
5. Разработка меню на Godot [Электронный ресурс]. -
<https://www.youtube.com/watch?v=Jjv2MWbQVhs&t=96s>
6. Сохранение на Godot 4 [Электронный ресурс]. -
<https://www.youtube.com/watch?v=Ulg-VXoAtTY&t=127s>