МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка компьютерной 2D-игры в жанре «Платформер» на платформе Godot Engine

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-348.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____ Я.А. Краева

Автор работы,
студент группы КЭ-404
_____ И.В. Стовбур

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-404
Стовбуру Илье Витальевичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка компьютерной 2D-игры в жанре «Платформер» на платформе Godot Engine.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**
3.1. Bethke E. Game development and production. – Wordware Publishing, 2003. – 432 p.
3.2. Документация к игровому движку Godot Engine. [Электронный ресурс] URL: https://docs.godotengine.org/ru/4.x/ (дата обращения: 20.05.2024 г.).
4. **Перечень подлежащих разработке вопросов**
4.1. Провести анализ предметной области и обзор аналогов.
4.2. Спроектировать приложение.
4.3. Реализовать и протестировать приложение.
5. **Дата выдачи задания:** 29.01.2024 г.

Научный руководитель,
ст. преподаватель кафедры СП Я.А. Краева

Задание принял к исполнению И.В. Стовбур

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Игровая индустрия является одной из главных в индустрии развлечений. Она сопоставима по масштабам с киноиндустрией и значительно опережает ее по скорости роста за последние пять лет [1].

2D-анимация и графика считается традиционной, так как известна с начала 19 века [2]. 2D-графику используют повсеместно и по сей день из-за ее простоты восприятия и работы с ней. Поэтому спрос на разработчиков, способных создавать качественные 2D-проекты, является актуальным. Также создание игр стало проще благодаря доступности различных инструментов, движков и сред разработки, которые позволяют создавать игры с минимальными техническими знаниями. И, так как разработка компьютерных игр требует знаний в различных областях, таких как программирование, дизайн, искусство и анимация, это делает ее отличной платформой для обучения и развития новых навыков.

2D-игры менее требовательные к производительности аппаратного обеспечения, чем 3D-игры, из-за особенностей отрисовки игровых объектов и не только, и поэтому их можно портировать на большинство устройств без страха того, что они будут работать нестабильно на данных устройствах [3]. В целом, разработка 2D-игр в жанре платформер актуальна, так как она удовлетворяет потребности игроков, а также предлагает новые возможности и перспективы для разработчиков.

Постановка задачи

Целью выпускной квалификационной работы является разработка компьютерной 2D-игры в жанре «Платформер» на платформе Godot Engine. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ предметной области и обзор аналогов;
2) спроектировать приложение;
3) реализовать и протестировать приложение.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения, списка литературы и приложения. Объем работы составляет 40 страниц, объем списка литературы – 16 источников.

В первой главе проводится обзор аналогов разрабатываемой компьютерной игры, и рассматриваются особенности для выбранного жанра игры.

Вторая глава посвящена описанию требований, составлению диаграммы вариантов использования и проектированию игры. Также в этой главе содержится описание основного концепта игры и выбор визуального стиля.

В третьей главе анализируются программные средства разработки, и приводится обоснование выбора игрового движка, а также описывается реализация различных игровых механик, и демонстрируется процесс создания персонажей, окружения, особых областей и интерфейса.

Четвертая глава содержит результаты тестирования полученного приложения, по которым сделан вывод о достижении поставленных требований.

В приложении содержатся скриншоты демонстрирующие виды врагов, окружение, игровые объекты, меню и пользовательский интерфейс разработанной игры.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Платформер – это один из основных жанров видеоигр, особенностью которого является передвижение игрока по уровням, представленным в виде платформ и предполагающим прыжки игрока между ними. Характерными особенностями платформеров являются наличие разнообразных врагов, препятствий и ловушек, а также необходимость сбора предметов и решения головоломок для прохождения уровней.

Этот жанр игр существует уже несколько десятков лет. Первые платформеры появились еще в 1980-х годах на игровых платформах, таких как аркадные автоматы и домашние компьютеры [4]. С тех пор этот жанр постоянно развивается и обновляется, появляются новые механики и возможности.

Разрабатываемая игра будет содержать в себе уровни с небольшими головоломками, скрытыми пространствами, ловушками и врагами, которые будут мешать игроку, тем самым испытывать его. Враги будут нескольких типов с различными характеристиками.

1.1. Анализ аналогичных проектов

Для анализа аналогичных проектов были выбраны следующие представители жанра: «Ori and the Blind Forest» [5], «Dead Cells» [6], «Hollow Knight» [7].

«Ori and the Blind Forest»

«Ori and the Blind Forest» – двухмерный приключенческий платформер, сюжет которого разворачивается в фэнтезийном мире, где главный герой, маленький дух по имени Ори, должен спасти свой дом – лес от разрушения. Для этого ему нужно пройти через множество уровней, сражаясь с врагами и решая головоломки. На фоне многих платформеров данная игра выделяется красивой графикой, плавной анимацией и атмосферным саундтреком.

На рисунке 1 изображен скриншот из игры «Ori and the Blind Forest». Игра получила множество положительных отзывов от критиков и игроков за свою уникальную атмосферу, сюжет и игровой процесс.

Рисунок 1 – Игра «Ori and the Blind Forest»

Моментами игра может показаться трудной для игроков, которые только недавно начали играть в платформеры. Некоторые уровни и боссы могут быть серьезным испытанием для них.

«Dead Cells»

«Dead Cells» – платформер с элементами «roguelike», с процедурно-генерируемым миром, который каждый раз создает новые уровни и врагов. Это делает каждое прохождение уникальным и непредсказуемым. Кроме того, в игре есть система прокачки персонажа, позволяющая улучшать его навыки и способности.

На рисунке 2 показан игровой процесс в «Dead Cells». Графика игры выполнена в пиксельном стиле, а музыкальное сопровождение создает атмосферу мрачного средневекового мира.

Рисунок 2 – Игра «Dead Cells»

Игра может быть очень сложной, особенно на высоких уровнях. Игровой процесс может показаться однообразным, ведь в игре приходится перепроходить раз за разом одни и те же уровни, хоть они и процедурно-генерируемые, они часто мало чем отличаются друг от друга.

«Hollow Knight»

«Hollow Knight» – платформер с элементами ролевой игры. Мир игры состоит из множества уровней, каждый из которых имеет свои особенности и врагов. Игрок должен сражаться с противниками, собирать предметы и решать головоломки, чтобы продвинуться дальше. Одним из главных достоинств игры является ее атмосфера и дизайн уровней. Каждый уровень имеет свой уникальный стиль и атмосферу, что создает неповторимое впечатление от игры. Графика, как и у прошлой рассмотренной игры, выполнена в пиксельном стиле.

На рисунке 3 представлен скриншот из игры «Hollow Knight». Игра довольно продолжительная и многим может наскучить, а ее сложность создает определенный порог вхождения в нее.

Рисунок 3 – Игра «Hollow Knight»

Вывод по первой главе

В данной главе был проведен обзор аналогов реализуемой игры. Были рассмотрены три игры, проанализированы их сильные и слабые стороны. Все три представленные игры получили в основном хвалебные отзывы, как от критиков, так и от игроков, поэтому стоит обратить внимание на их положительные стороны и постараться учесть моменты, которые игроки отметили как недостатки. Так, для того, чтобы завлечь игрока, иногда приходится заставлять его возвращаться туда, где он уже был, а элементы головоломки позволяют сделать перерыв внутри динамичного игрового процесса.

2. ПРОЕКТИРОВАНИЕ

2.1. Требования к приложению

Игра должна запускаться и стабильно работать, а также учитывать разрешение экрана устройства, на котором запущена.

Функциональные требования

Функциональные требования – это набор условий, определяющих ожидания пользователя от приложения. Они описывают, какие функции, и возможности должны быть реализованы в продукте, чтобы удовлетворить потребность конечного пользователя и выполнить поставленные задачи. На основе анализа предметной области были определены функциональные требования к разрабатываемому приложению.

Приложение должно обеспечивать для пользователя:

1) возможность управления персонажем;
2) возможность взаимодействия с игровыми объектами;
3) возможность изменения настроек;
4) возможность пройти игру до конца.

Нефункциональные требования

Нефункциональные требования – требования, определяющие свойства, которые приложение должно демонстрировать, или ограничения, которые оно должно соблюдать, не относящиеся к поведению приложения.

Разрабатываемое приложение должно удовлетворять следующим нефункциональным требованиям:

1) приложение должно работать на операционной системе Windows 10 и новее;
2) в приложении должна использоваться графика в пиксельном стиле;
3) приложение должно быть разработано на платформе Godot Engine.

2.2. Концепт игры и диаграмма вариантов использования

Основной концепт игры

Действие разрабатываемой игры разворачивается в средневековом мире. Игрок управляет персонажем, который должен преодолеть множество препятствий, сражаться с недоброжелателями и решать несложные головоломки. Графика представлена в пиксельном стиле, так как это популярный и относительно легкий в исполнении стиль [8]. Игроку предстоит находить средства, чтобы пройти дальше по игре, тем самым приближаясь к концу игры. На уровнях будут присутствовать различные секреты и предметы, которые помогут в дальнейшем прохождении игры.

В самом начале игрок появляется на пути к замку, и ему доступно полное передвижение и удары мечом. По ходу игры игрок должен будет найти предмет, с помощью которого он сможет пройти через мост к замку, в котором его ждет главный противник. Игра будет считаться пройденной, когда игрок победит этого главного противника.

Игра изначально разрабатывается для ПК на игровом движке Godot Engine. Для создания изображений игровых объектов, кадров их анимации и редактирования изображений игровых персонажей используется программа Aseprite [9].

Диаграмма вариантов использования

Диаграмма вариантов использования – это диаграмма, которая отражает отношения между актерами и вариантами использования системы. Она позволяет описать систему на концептуальном уровне и используется для спецификации внешних требований к системе [10].

Основные игровые возможности и минимальный функционал интерфейса представлены в разработанной диаграмме вариантов использования, которая изображена на рисунке 4.

С системой взаимодействует только один актер – игрок. Игрок может продолжить игру, если он ранее сохранял игру, или начать новую игру. Начав игру, он может использовать все игровые возможности, в част-

ности поставить паузу, открыв меню. Сохраниться можно в любой момент игры.

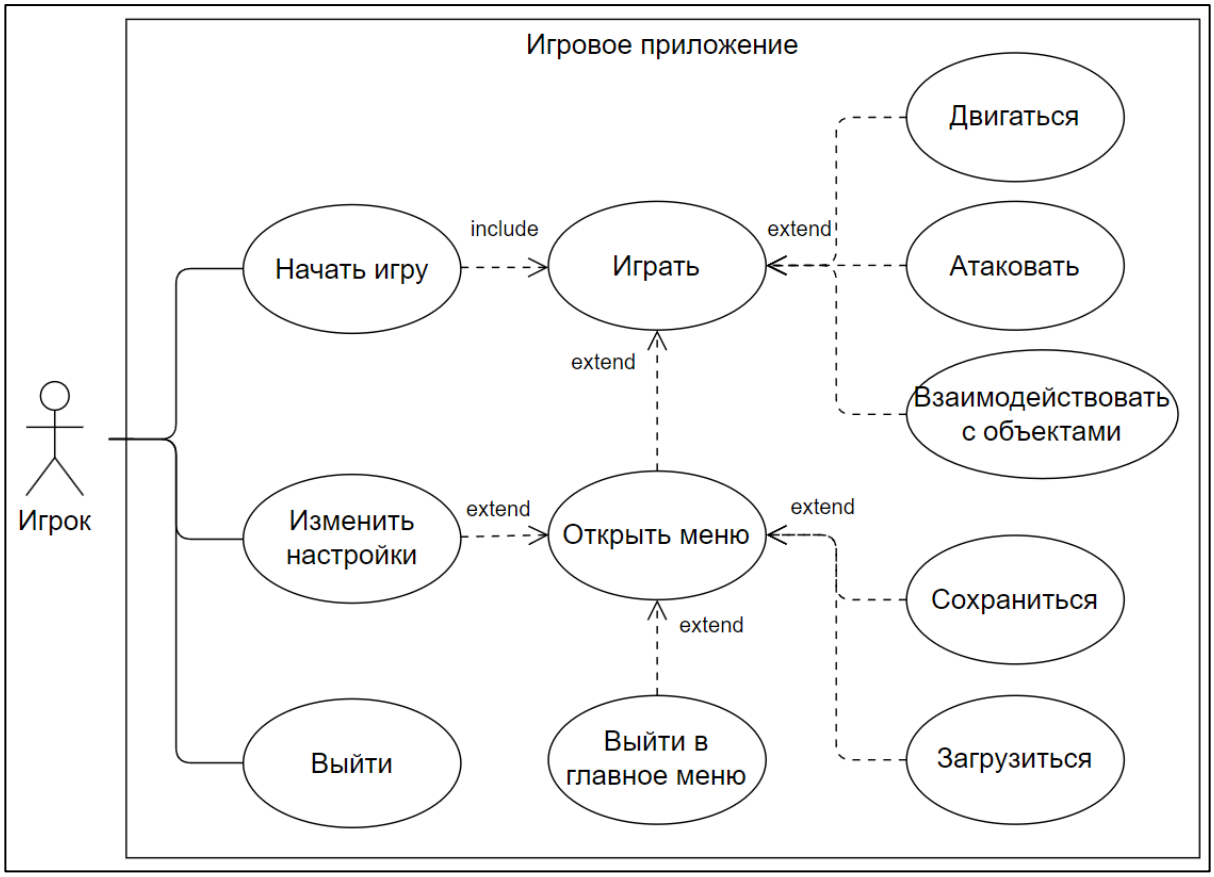

Рисунок 4 – Диаграмма вариантов использования

Вывод по второй главе

Во второй главе были определены основные требования к системе, и была составлена диаграмма вариантов использования в соответствии с требованиями. Представлено описание основного концепта игры и выбор визуального стиля.

3. РЕАЛИЗАЦИЯ

3.1. Программные средства реализации

Игровой движок – это программное обеспечение, состоящее из набора инструментов разработчика, включая программы для создания графики, физического взаимодействия, управления платформами, визуализации и звукового сопровождения, а также запуска скриптов и искусственного интеллекта [11]. Ключевые признаки игрового движка включают наличие архитектуры управления данными и возможность создания игр для разных платформ и устройств.

Игровой движок Unity – это кроссплатформенная среда разработки компьютерных игр, созданная американской компанией Unity Technologies. Он был выпущен в 2005 году и с тех пор постоянно развивается. Unity позволяет создавать приложения для более чем 18 различных платформ, включая персональные компьютеры, игровые консоли, мобильные устройства и интернет-приложения [12]. Движок использует язык C# для работы со скриптами.

Unreal Engine – это игровой движок, разработанный и поддерживаемый компанией Epic Games. Он был создан для создания шутеров от первого лица, но впоследствии использовался и в играх других жанров, таких как стелс-игры, файтинги и массовые многопользовательские ролевые онлайн-игры [13]. Движок написан на языке C++ и поддерживает множество операционных систем и платформ. Он использует модульную систему зависимых компонентов для упрощения портирования и поддерживает различные системы отрисовки, воспроизведения звука и сетевые технологии.

GameMaker – это игровой движок, разработанный Марком Овермарсом в 1999 году и развиваемый компанией YoYo Games с 2007 года. Он предоставляет инструменты для создания 2D-игр с использованием растровой графики, векторной графики и 2D-скелетной анимации [14]. GameMaker использует собственный язык сценариев Game Maker

Language, который является императивным языком с динамической типизацией, напоминающим JavaScript и языки семейства C.

В качестве игрового движка, на котором будет реализовываться приложение, был выбран Godot Engine. В таблице 1 представлено сравнение четырех популярных современных движков. В сравнении подмечены их достоинства и недостатки.

Таблица 1 – Сравнение популярных игровых движков

№	Название игрового движка	Достоинства	Недостатки
1	Unity	Наличие огромной библиотеки ассетов и плагинов.	Медлительность движка, политика компании.
2	Unreal Engine	Мультизадачность, универсальность, открытый исходный код.	Сложный интерфейс, требовательный к системе.
3	Godot Engine	Высокая гибкость, открытый исходный код, богатый функционал, интуитивный интерфейс, поддержка языка C#, легковесный.	Ограниченная документация.
4	GameMaker	Простота и удобство, поддержка множества интернет-площадок.	Game Maker Language используемый в качестве скриптового языка имеет ряд недостатков.

Godot Engine – это игровой движок с открытым исходным кодом, который используется для создания 2D- и 3D-игр. Он был создан для того, чтобы помочь разработчикам создавать игры быстрее и проще. Godot Engine имеет простой и интуитивный интерфейс, который позволяет даже новичкам в разработке игр создавать свои проекты. Кроме того, данный движок поддерживает свой язык программирования GDScript, а также C#. На рисунке 5 изображена рабочая область Godot Engine. Данный движок имеет весь необходимый инструментарий для создания 2D-игр. В нем также можно работать со звуком и с освещением.

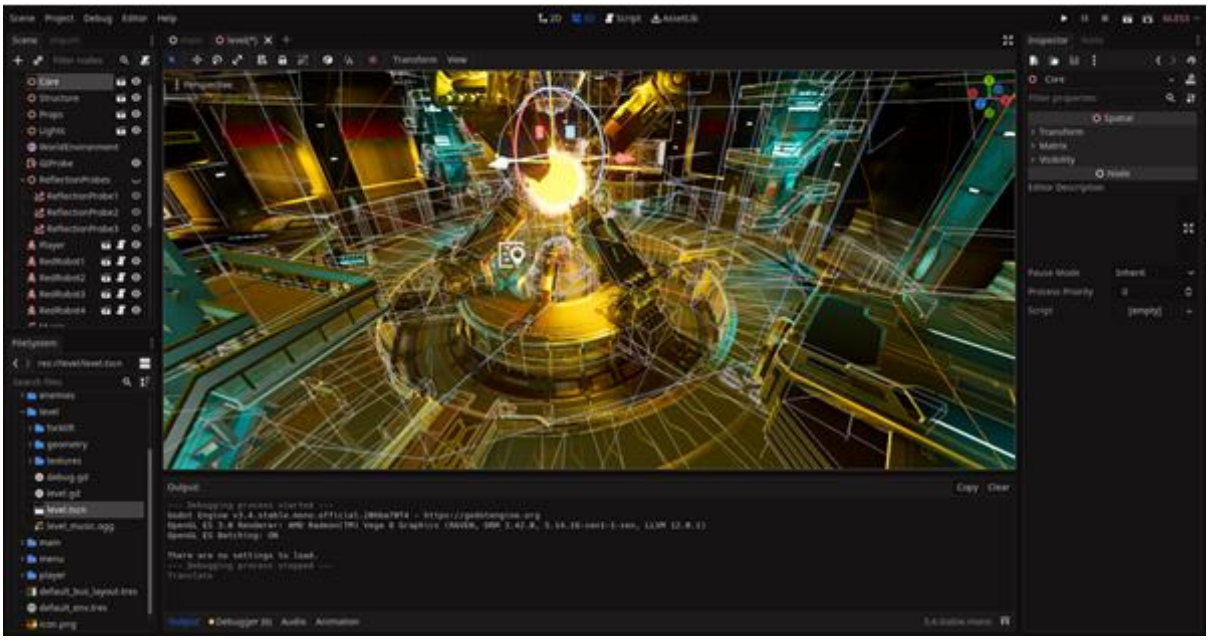

Рисунок 5 – Интерфейс движка Godot

3.2. Персонажи игры

Для создания поведения персонажей программируются скрипты. Скрипты – это последовательности действий, которые приводят к конкретному результату. Враги должны представлять угрозу для игрока, поэтому они запрограммированы на погоню за игроком в их области видимости и возможность перепрыгивать небольшие препятствия, а при достижении игрока они производят удар мечом. В листинге 1 представлен фрагмент скрипта одного из вражеских персонажей, а на рисунке 6 показан алгоритм его поведения.

Листинг 1 – Функция состояния погони

```
func chase():
	direction = (player.position - self.position).normalized()
	velocity.x = direction.x * speed
	animPlayer.play("run")
	if direction.x > 0 && animSprite.is_flipped_h():
		animSprite.flip_h = false
		$AttackDirection.scale.x = 1
	elif direction.x < 0 && !animSprite.is_flipped_h():
		animSprite.flip_h = true
		$AttackDirection.scale.x = -1

	if !groundDetector.has_overlapping_bodies() || is_on_wall():
		if !jumpBlock.has_overlapping_bodies():
			currentState = STATE.JUMP
```

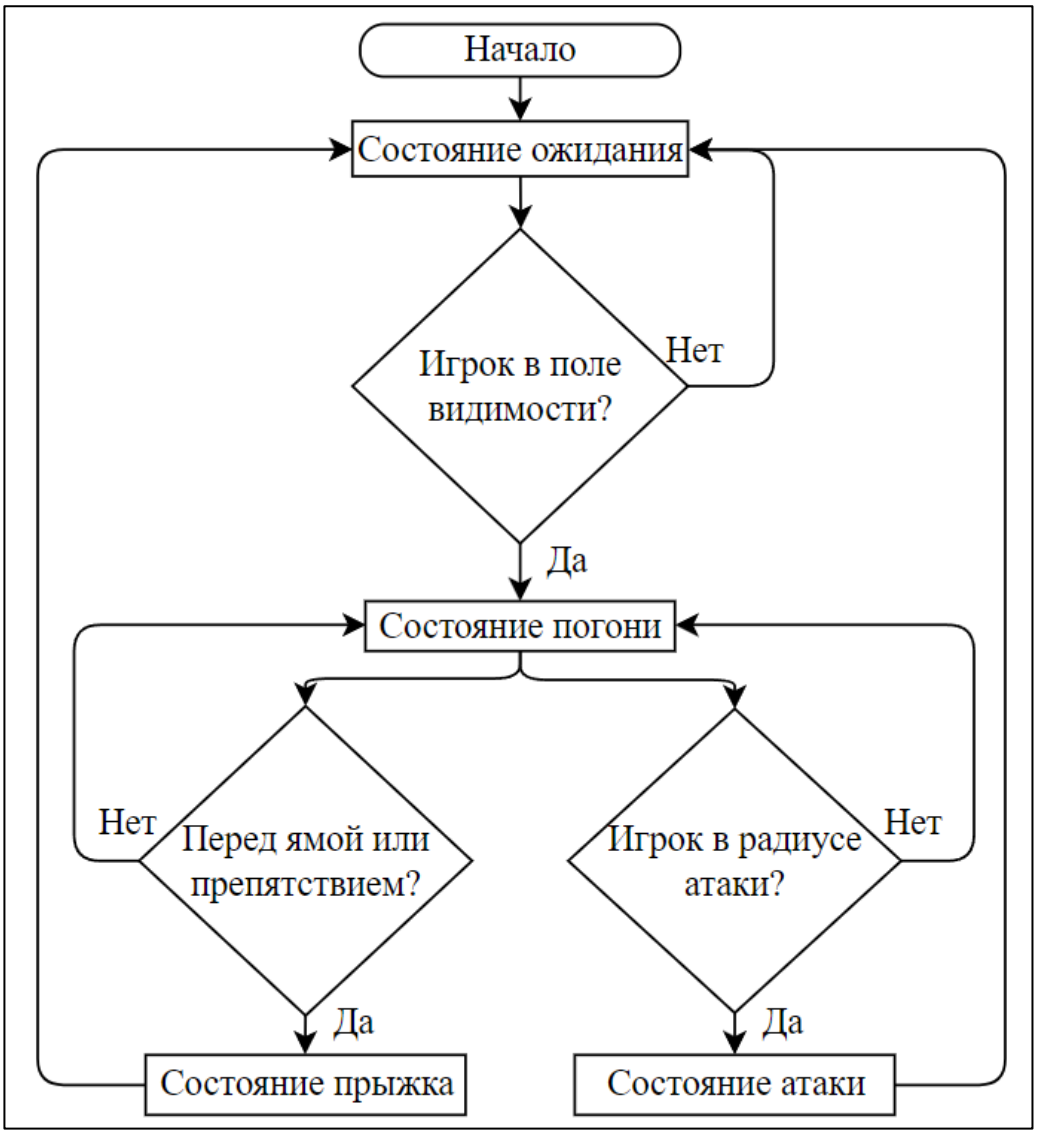

Рисунок 6 – Алгоритм поведения вражеских персонажей

Изображения персонажей были взяты в некоммерческих целях из бесплатных источников [15] и отредактированы в программе Aseprite. Также в этой программе были отредактированы их кадры анимаций. На рисунке 7 изображен интерфейс данной программы.

У каждого персонажа есть области столкновения, получения урона и нанесения урона, и еще различные области соприкосновения. Области столкновения реагируют друг на друга и позволяют объектам сталкиваться между собой. Области получения урона регистрируют вхождение в них областей нанесения урона. Области нанесения урона выключены по умолчанию и включаются только в момент атаки персонажа. Если область по-

лучения урона зарегистрирует вхождение области нанесения урона, то у персонажа отнимется количество здоровья, которое определенно в скрипте персонажа. Все эти области имеют слои и маски для более детальной настройки их взаимодействия.

Рисунок 7 – Редактирование анимации в программе Aseprite

Регистрация соприкосновений областей производится с помощью сигналов. Сигналы – это механизм делегирования, встроенный в Godot, который позволяет одному игровому объекту реагировать на изменение в другом без их привязки друг к другу.

На рисунке 8 изображена сцена главного персонажа с отображением всех областей соприкосновения и столкновения.

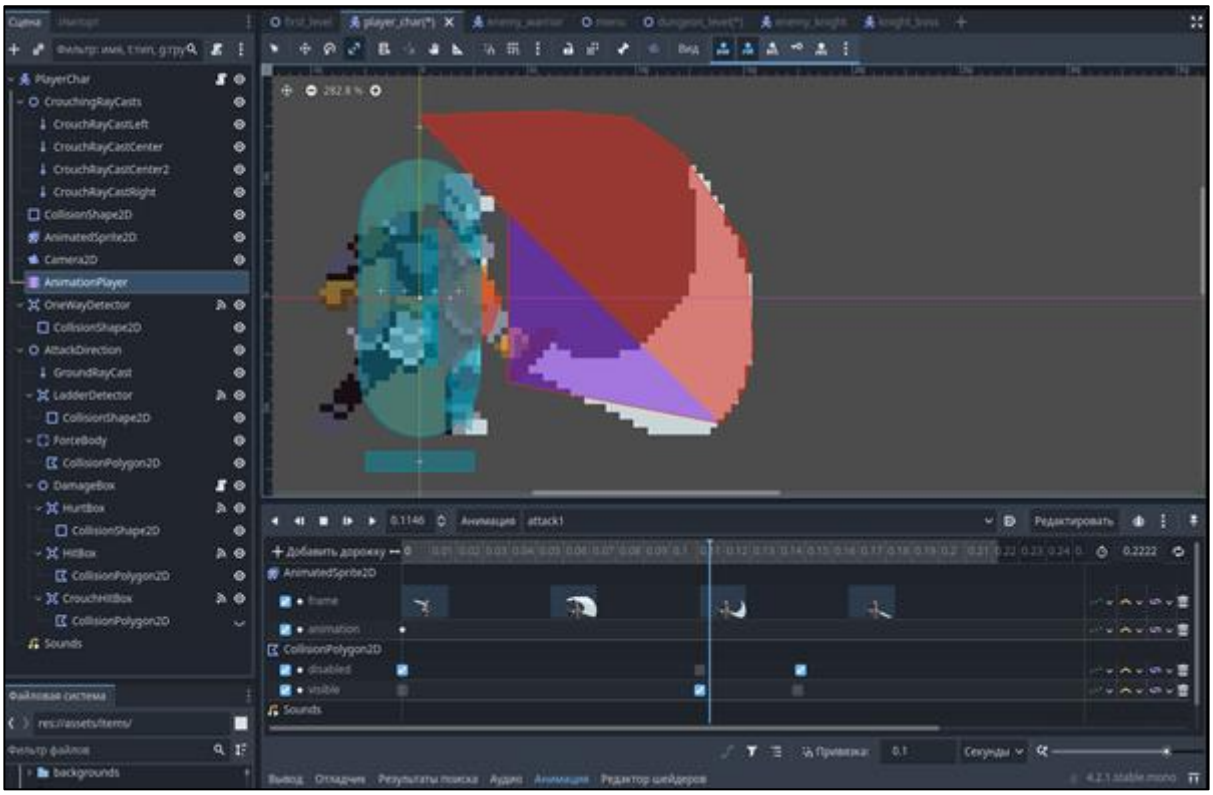

Рисунок 8 – Сцена «PlayerChar»

При программировании скриптов для главного персонажа, так же как и для вражеских, использовался конечный автомат. То есть у персонажей есть всегда одно состояние для текущего момента из всех возможных.

Так как суть платформеров – это преодоление препятствий, то прыжки в них немаловажная часть. Поэтому чтобы прыжок ощущался лучше, высота прыжка зависит от времени нажатия на кнопку. Состояние прыжка главного персонажа представлено в листинге 2. Помимо прыжков и бега, персонаж может присаживаться и проходить под низкими проходами. В состоянии приседа можно атаковать, но нельзя метать топор. Если находясь в приседе попытаться встать там, где это невозможно, то персонаж не встанет, так как в сцену главного персонажа добавлены лучи прикосновения, которые посылают сигналы при соприкосновении с препятствием.

Листинг 2 – Функция состояния прыжка

```
func jump(direction, delta):
	changeHorizontalPos(direction)
	if is_on_floor() && !is_jumping:
		jumpPower = -300
		jumpTimer = 0.0
		is_jumping = true
		animPlayer.play("jump")
		var randStream = AudioStreamRandomizer.new()
		randStream.add_stream(0, soundJump0)
		randStream.add_stream(1, soundJump1)
		$Sounds.stream = randStream
		$Sounds.play()
	elif is_on_floor() || groundRayCast.is_colliding():
		currentState = STATE.IDLE

	if Input.is_action_pressed("jump") && is_jumping && jumpTimer <
jumpTimerMax:
		jumpPower += JUMP_VELOCITY_STEP
		applyJumpForce(jumpPower)

	if is_climbing && Input.is_action_just_pressed("jump"):
		currentState = STATE.CLIMB

	if !is_on_floor() && !groundRayCast.is_colliding():
		jumpTimer += delta

	if !is_on_floor() && !groundRayCast.is_colliding() && jumpTimer >=
jumpTimerMax:
		velocity.y += gravity * delta * 2

	if Input.is_action_just_pressed("attack") && !is_cooldown:
		numOfAttacksInAir -= 1
		currentState = STATE.ATTACK1

	if velocity.y > 0:
		posBeforeFall = position.y
		currentState = STATE.FALL
```

В игре присутствуют платформы с односторонним движением, сквозь которые можно прыгать и вставать на них, а слезть с них можно присев. Для реализации таких платформ в сцену главного персонажа была добавлена область определения таких платформ и скрипт, который включал столкновение с такими платформами после того, как эта область с ними соприкасалась.

Игрок может подниматься по лестницам и веревкам и спрыгивать с них. Они помогают строить уровни вертикально и добавляют в игру разнообразие передвижения. Реализация данной возможности показана в листинге 3.

Листинг 3 – Функция состояния карабканья

```
func climb():
	animPlayer.play("idle")
	if !is_on_floor():
		animPlayer.play("hang")

	if is_climbing:
		velocity.y = 0
		velocity.x = 0
		if Input.is_action_pressed("jump"):
			velocity.y = -SPEED
		elif Input.is_action_pressed("crouch"):
			velocity.y = SPEED
		elif Input.is_action_pressed("left") ||
Input.is_action_pressed("right"):
			is_climbing = false
			$AttackDirection/LadderDetector/CollisionShape2D.disabled
= true
			await get_tree().create_timer(0.2).timeout
			$AttackDirection/LadderDetector/CollisionShape2D.disabled
= false
	elif !is_on_floor():
		posBeforeFall = position.y
		currentState = STATE.FALL

	if is_climbing && is_on_floor() && Input.is_anything_pressed() &&
!Input.is_action_pressed("jump"):
		currentState = STATE.IDLE
```

3.3. Окружение

Для того, чтобы фон уровня не выглядел статичным используется эффект параллакса. Эффект параллакса достигается за счет движения нескольких слоев фона с разной скоростью. Параллакс создает ощущение глубины и динамики, что в свою очередь улучшает общее восприятие игры. На рисунке 9 представлен процесс создания фона с данным эффектом для первого уровня в игре. Движок предусматривает работу с эффектом параллакса и имеет типы узлов для настройки различных параметров этого эффекта, но в силу специфики данных узлов, они вызывали некоторые проблемы в процессе игры, поэтому они были заменены на собственный аналог.

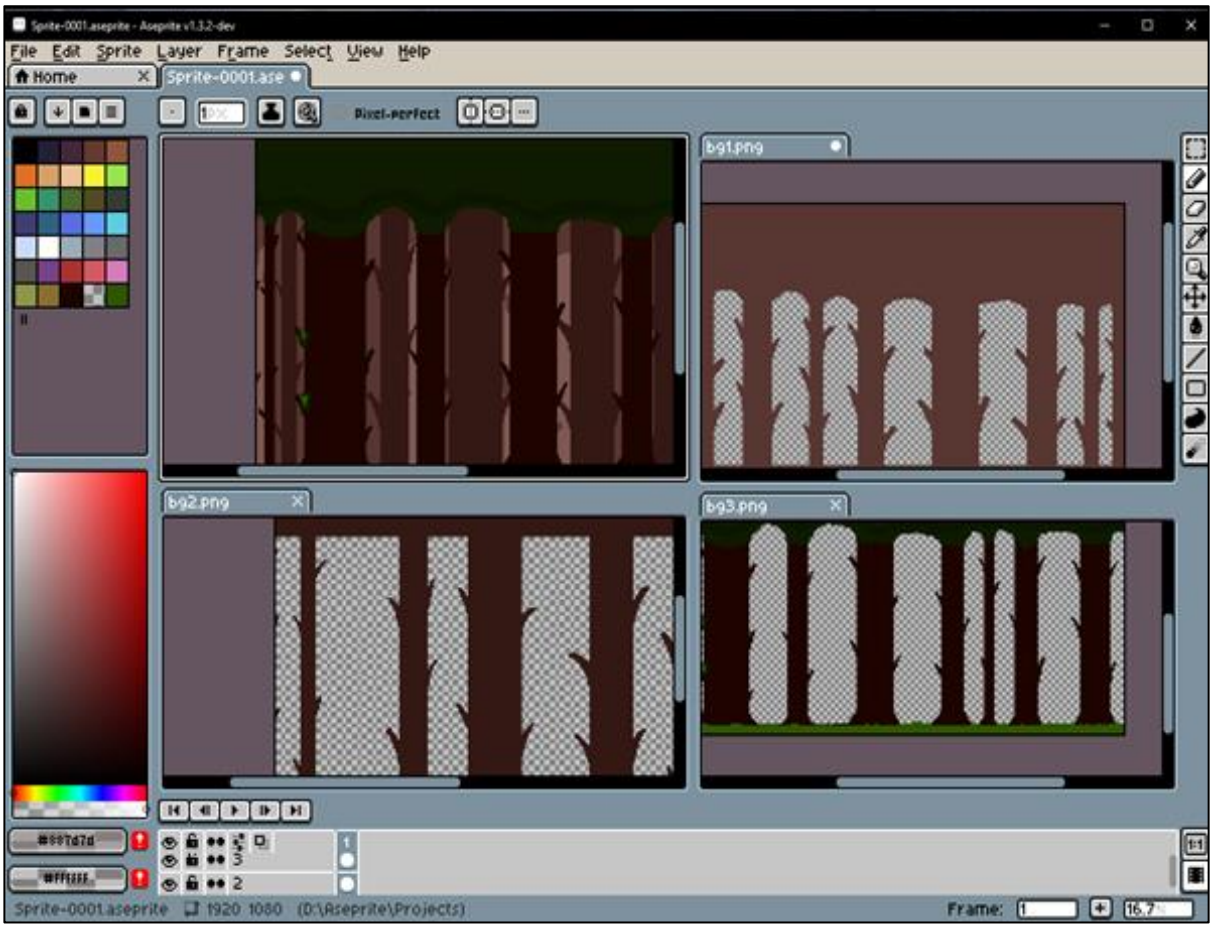

Рисунок 9 – Создание фона для первого уровня

Ландшафт уровней строится из специального узла под названием «TileMap», в нем можно задать слои видимости и физические слои, которые отвечают за взаимодействие с областями столкновения. Слои видимости нужны для того, чтобы задавать видимость за объектами, на одном уровне, либо же перед ними. На рисунке 10 показана вкладка с настройками этого узла. С помощью этого узла можно создавать специальные карты местности, которые нужно перед этим разметить на имеющемся наборе плит. Разметка набора плит рисуется 2D-полигонами. В этом же узле настраивается отбрасывание теней от источников света с помощью специального окклюзионного слоя.

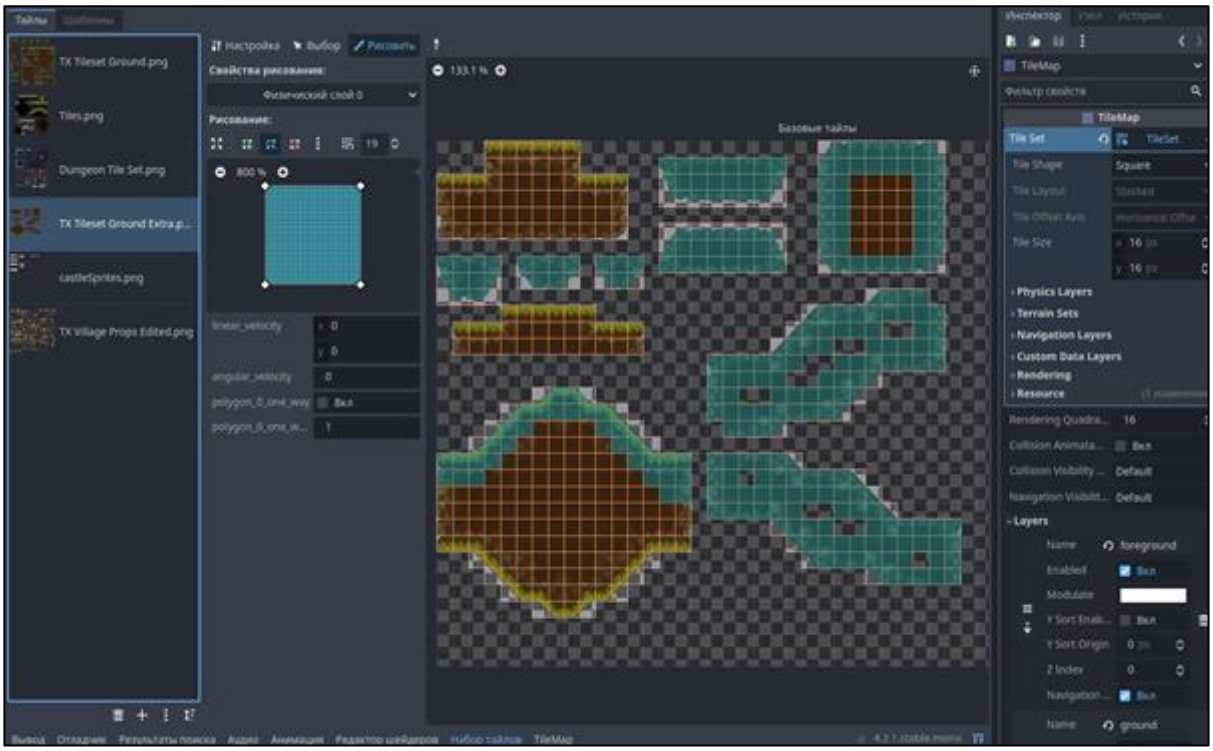

Рисунок 10 – Настройки узла «TileMap»

В игре присутствует множество объектов, с которыми игрок может взаимодействовать. Передвижные ящики, на которые можно забираться, рычаги, камень, открывающий проход в пещеру, зелья лечения, зелье, которое увеличивает максимальное здоровье персонажа, а также метательный топор, которым, помимо нанесения урона врагам, можно активировать рычаги в зоне недосягаемости.

Несколько видов ловушек приносят некоторое разнообразие. Статичные шипы наносят урон при соприкосновении с ними, после них ловушка с нажимной плитой выглядит непредсказуемо. Передвижная пила добавляет динамики. Ловушка-маятник может не только нанести урон, но и столкнуть с платформы. На рисунке 11 продемонстрированы ловушки и объекты.

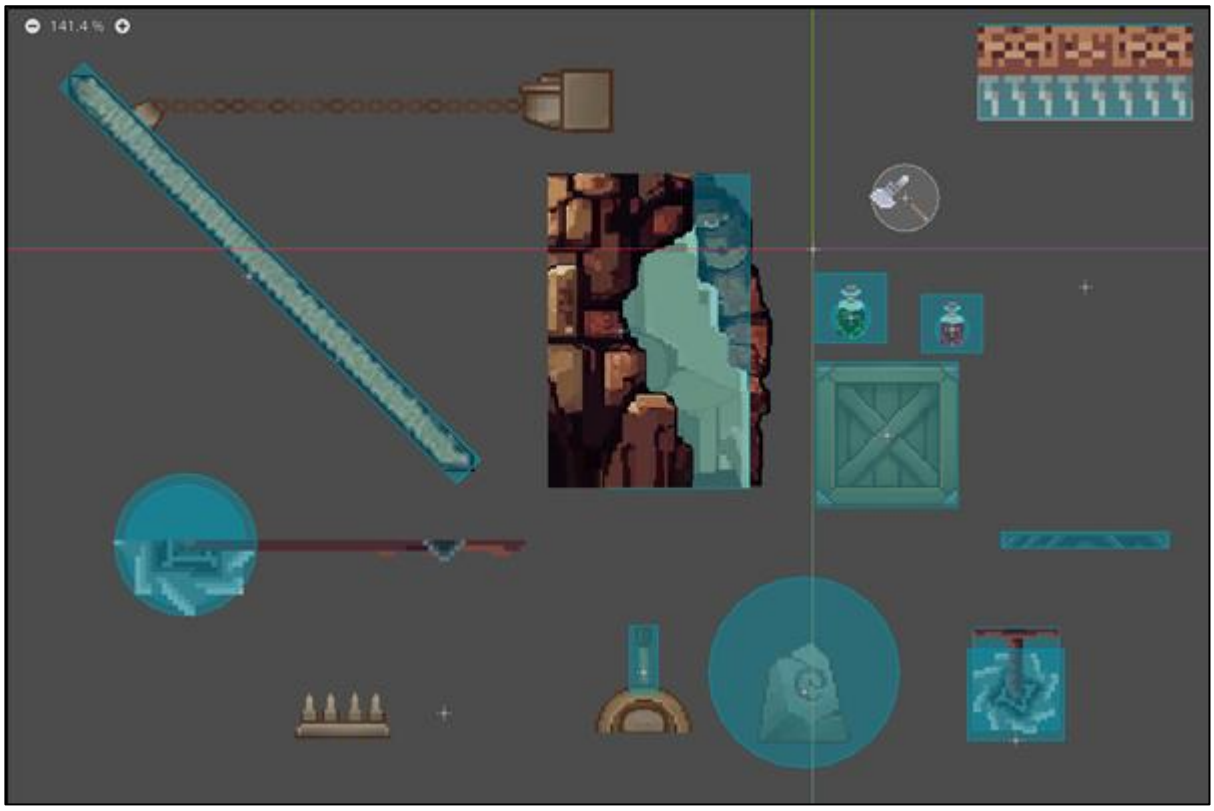

Рисунок 11 – Ловушки и различные объекты, используемые в игре

3.4. Особые области

На уровнях есть места, при входе в которые происходит определенное событие. Например, такое событие как переход на новый уровень, или скрытая область, которая раскрывается при достижении определенной позиции персонажа. Пример представлен на рисунках 12 и 13. Такие области являются локальными для сцены уровня. Также такие области позволяют застать игрока врасплох, скрывая от него опасность вблизи. Они включают в себя анимации и объекты, которые скрывают, а связаны они сигналами со сценой уровня.

Такая же область используется для одной из ловушек. Ловушка с нажимной плитой срабатывает только тогда, когда игрок наступил на плиту. Эта плита сначала проигрывает собственную анимацию и по завершению запускает анимацию ловушки, у которой смещается область нанесения урона по игроку.

Рисунок 12 – Окружение до вхождения в область

Рисунок 13 – Окружение после вхождения в область

3.5. Сохранение игры

Функция сохранения игры позволяет игроку сохранить свой прогресс и вернуться к игре позже, продолжая с того места, где он остановился. Функция сохранения игры важна для игроков, так как она позволяет им не

бояться потерять свой прогресс из-за неожиданного выключения компьютера или других проблем. Реализация данной функции показана в листинге 4.

Листинг 4 – Функция сохранения игры

```
func save_game():
	var saveFile = FileAccess.open("user://savegame.save",
FileAccess.WRITE)
	var saveNodes = get_tree().get_nodes_in_group("Persist")
	var levelNode = get_tree().get_first_node_in_group("Persist")
	var scene = PackedScene.new()
	scene.pack(levelNode)
	match levelNode.get_name():
		"FirstLevel":
			Global.savedLevels[0] = scene
		"DungeonLevel":
			Global.savedLevels[1] = scene
		"CastleLevel":
			Global.savedLevels[2] = scene

	var playerNode = levelNode.find_child("Player").get_child(0)

	# firstly saving level and player nodes
	saveFile.store_line(JSON.stringify(saveNodes[saveNodes.find(levelNode
)].call("saveData")))
	saveFile.store_line(JSON.stringify(saveNodes[saveNodes.find(playerNod
e)].call("saveData")))
	for node in saveNodes:
		if node.scene_file_path.is_empty():
			print("сохраняемый узел '%s' не является экземпляром
сцены, пропущен" % node.name)
			continue

		if !node.has_method("saveData"):
			print("'%s' не имеет функции saveData(), пропущен" %
node.name)
			continue

		if node == playerNode || node == levelNode:
			continue

		var nodeData = node.call("saveData")
		var jsonString = JSON.stringify(nodeData)
		saveFile.store_line(jsonString)
```

Для работы данной функции необходимо прописать функции сохранения свойств узлов, которые нужно сохранить, и уже они вызываются в основной функции сохранения. Пример функции сохранения определенного узла приведен в листинге 5. Все эти узлы должны находиться в одной группе. Все сохраняемые данные сериализуются в файл в формате JSON, то есть в набор пар «ключ: значение», для удобства работы над ним. При

загрузке сохранения происходит десериализация файла встроенным в движке методом.

Листинг 5 – Функция сохранения свойств узла главного персонажа

```
func saveData():
	Global.playerDirection = -1 if animSprite.is_flipped_h() else 1
	var saveDict = {
		"filename" : get_scene_file_path(),
		"parent" : get_parent().get_path(),
		"pos_x" : position.x,
		"pos_y" : position.y,
		"currentHealth" : currentHealth,
		"currentState" : currentState,
		"is_cooldown" : is_cooldown,
		"velocity.x" : velocity.x,
		"velocity.y" : velocity.y,
		"numOfAttacksInAir" : numOfAttacksInAir
	}
	return saveDict
```

При каждом переходе на другой уровень игра сохраняет состояния обоих уровней. Уровень, с которого игрок переходит, сохраняется для того, чтобы при переходе на него обратно он оставался прежним. Уровень, на который переходит игрок, сохраняется на случай, если персонаж умрет на этом уровне, чтобы начать уровень сначала с тем количеством очков здоровья, с которым он зашел на данный уровень. Данная функция показана в листинге 6.

Листинг 6 – Функция контрольных точек при переходе на уровень

```
func _process(_delta):
	if is_nearEntrance && Input.is_anything_pressed():
		var scene = PackedScene.new()
		scene.pack(self)
		Global.savedLevels[1] = scene
		var nextScene = Global.savedLevels[0]
		queue_free()
		Global.playerCurrentPosition = playerSpawnPosition
		Global.playerDirection = playerDirection
		Global.playerTransitionHP = Global.playerCurrentHealth
		SceneTransition.change_scene_to_packed(nextScene)
```

3.6. Интерфейс

При входе в игру игрока встречает главное меню, из которого можно начать игру, зайти в меню настроек, где можно поменять громкость звука и перевести игру в полноэкранный режим, либо выйти из игры. Меню настроек приведено на рисунке 14.

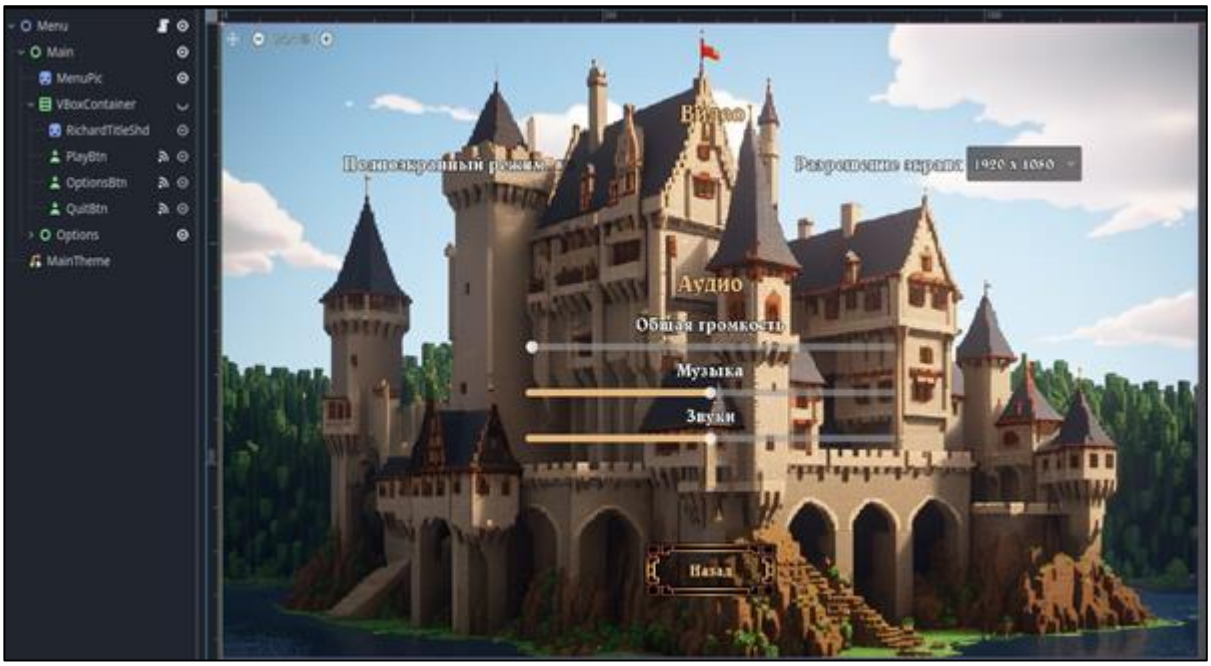

Рисунок 14 – Сцена с главным меню и настройками

В самой игре присутствует полоска здоровья персонажа, которая реагирует на изменение здоровья персонажа с помощью сигналов. Индикатор отображает как получение урона от противника, ловушки или падения с большой высоты, так и восстановление здоровья от зелья или увеличение максимального показателя. На рисунке 15 показана сцена с меню паузы и индикатором здоровья персонажа.

Также элементами интерфейса можно считать всплывающие подсказки, появляющиеся при заходе в соответствующую область определенных объектов и появление сообщений при совершении определенных действий.

Здоровье врагов скрыто, чтоб игрок не знал, какой урон он наносит, и сколько ударов противник еще может выдержать – это прибавляет игре сложность.

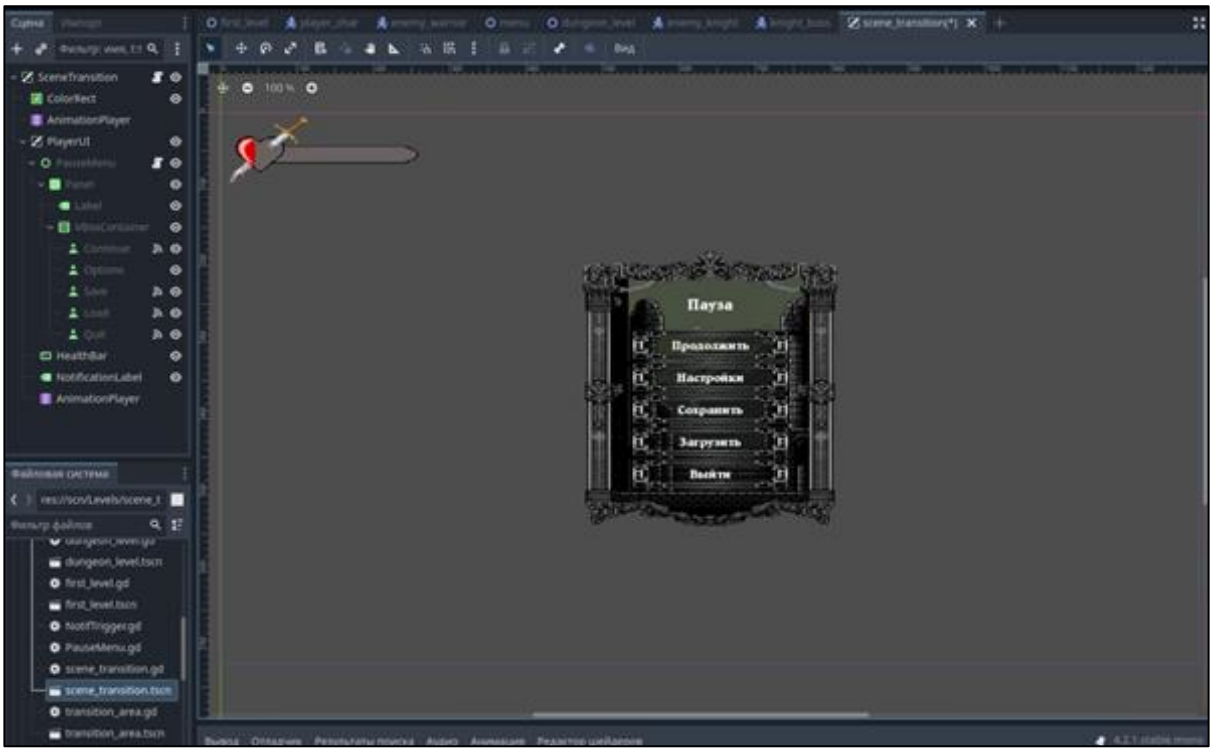

Рисунок 15 – Сцена с пользовательским интерфейсом

Переходы между уровнями сопровождены небольшой анимацией плавного закрашивания экрана в черный цвет и обратно. Это позволяет скрыть передвижение камеры при ее подгрузке на уровне и придает игре более презентабельный вид.

Вывод по третьей главе

Было приведено сравнение различных средств реализации и, в соответствии с заданием, была реализована 2D-игра в жанре платформер с возможностью сохранения игрового прогресса. Персонаж, управляемый игроком, может бегать, прыгать, красться, метать топор, наносить удары и взаимодействовать с объектами. Вражеский персонаж преследует игрока, перепрыгивает ямы, забирается на возвышенности и при достижении игрока атакует его, при этом отнимая у него здоровье, которое показывается на соответствующем индикаторе.

4. ТЕСТИРОВАНИЕ

Тестирование необходимо для обеспечения наилучшего опыта игроков, устранения проблем с игрой и ошибок дизайна [16].

Для проверки соответствия требованиям, представленным к приложению, было проведено функциональное тестирование. Набор функциональных тестов представлен в таблице 2.

Таблица 2 – Функциональное тестирование игры

№	Название	Действия	Ожидаемый результат
1	Движение по оси абсцисс.	Нажатие клавиш «A», «D», стрелки влево, стрелки вправо.	Персонаж движется и смотрит в соответствующем направлении.
2	Движение по оси ординат.	Нажатие клавиш «W», «SPACE», стрелки вверх, либо падение с игровой платформы.	Персонаж прыгает и падает.
3	Уйти от преследования.	Войти в область обнаружения врага и выйти из него.	Враг перестал преследовать игрока.
4	Получение урона.	Подойти к врагу и получить урон.	Персонаж проигрывает анимацию попадания по нему, индикатор здоровья изменяется.
5	Смерть персонажа.	Получить от врага четыре удара.	Персонаж проигрывает анимацию смерти, и игра переходит в главное меню.
6	Пауза.	В игре нажать клавишу «ESC».	Игра останавливается и показывает меню.
7	Сохранение и загрузка.	В игре нажать клавишу «ESC» и нажать кнопку «Сохранить», после чего снова зайти в меню и нажать кнопку «Загрузить».	Создается файл с данными и эти данные загружаются.
8	Получение урона от падения.	Спрыгнуть с большой высоты.	Персонаж проигрывает анимацию попадания по нему, индикатор здоровья изменяется.
9	Подбирание топора.	Подойти к топору близко и нажать клавишу действия «E».	Топор пропадает с карты.
10	Бросок топора.	Если топор был подобран, нажать правую кнопку мыши.	Топор вылетает с позиции игрока и дальше приземляется.
11	Движение ящика.	Подвинуть ящик, направляясь в его сторону.	Ящик приводится в движение.
12	Подобрать зелье лечения при недостатке здоровья.	Подойти к зелью близко.	Зелье пропадает и восстанавливает определенное количество здоровья, и это отображается на индикаторе.

Окончание таблицы 2

№	Название	Действия	Ожидаемый результат
13	Переход между уровнями.	Пройти в область перехода.	Экран затемняется, уровень меняется, и экран возвращается в прежнее состояние.
14	Убийство врага.	Нанести врагу смертельный удар.	Враг проигрывает анимацию смерти и пропадает с уровня.
15	Изменение громкости музыки и звуков.	Зайти в настройки и изменить положение ползунков.	Музыка становится тише или громче в зависимости от положения ползунка.
16	Забраться по лестнице.	Зажать кнопку «W» при нахождении рядом с лестницей.	Персонаж поднимается вверх по лестнице.
17	Присесть и пройти через низкий проход.	Нажать кнопку «S» и двигаться в направлении прохода.	Персонаж приседает и проходит через проход.
18	Ударить в положении сидя.	В положении сидя нажать левую кнопку мыши.	Персонаж проигрывает анимацию удара сидя.
19	Забраться на «oneway» платформу.	Запрыгнуть на платформу стоя под ней.	Персонаж запрыгивает на платформу и не падает вниз.
20	Активировать рычаг, кинув в него топор.	Имея при себе топор, кинуть его в неактивированный рычаг.	Топор, задевая рычаг, активирует его и приземляется на землю.
21	Увеличить максимальное здоровье персонажа.	Подобрать зеленое зелье на клавишу «E».	Появляется сообщение о том, что максимальное здоровье увеличено и вносятся изменения в индикатор здоровья.
22	Выход в главное меню.	В игре нажать клавишу «ESC» и нажать кнопку «Выйти».	Возвращение в главное меню.

Вывод по четвертой главе

Для подтверждения работоспособности и выявления ошибок в разработанной игре было проведено функциональное тестирование. Все тесты были успешно пройдены, тестирование ошибок не выявило.

ЗАКЛЮЧЕНИЕ

В заключение данной работы можно отметить, что цель работы – разработка 2D-игры в жанре платформер на игровом движке Godot Engine – была успешно достигнута. В процессе работы были решены следующие задачи.

1. Проведен анализ предметной области и обзор аналогов.
2. Спроектировано приложение.
3. Реализовано приложение.
4. Проведено тестирование приложения.

В ходе работы были освоены методы разработки 2D-игр на игровом движке Godot Engine, а также был освоен встроенный в движок язык программирования GDScript. В процессе работы над игрой было необходимо использовать не только навыки программирования, но и навыки работы с 2D-графикой и игровым дизайном.

В дальнейших планах добавить еще несколько уровней перед концом игры, доработать функцию сохранения игры, добавить фоновую музыку, разнообразить звуковое сопровождение, улучшить графику и добавить другие виды врагов и ловушек. Также в планах получить обратную связь от игроков и учесть их отзывы и пожелания, возможно добавив их идеи в игру.

ЛИТЕРАТУРА

1. Уточкин В., Сахнов К. Игровая Индустрия: Геймдев (Gamedev). [Электронный ресурс] URL: https://hsbi.hse.ru/articles/igrovaya-industriya-geymdev/ (дата обращения: 24.03.2024 г.).

2. Что такое 2D-анимация и где ее используют. [Электронный ресурс] URL: https://www.etxt.ru/subscribes/chto-takoe-2d-animatsiya-i-gde-ee-ispolzuyut/ (дата обращения: 10.02.2024 г.).

3. Unity How-to. [Электронный ресурс] URL: https://unity.com/ru/how-to/difference-between-2D-and-3D-games (дата обращения: 10.02.2024 г.).

4. McDonald P. D. Run and Jump: The Meaning of the 2D Platformer. // The MIT Press, 2024. – 171 p. DOI: 10.7551/mitpress/14478.001.0001.

5. Сайт игры «Ori and the Blind Forest». [Электронный ресурс] URL: https://www.orithegame.com/blind-forest/ (дата обращения: 10.02.2024 г.).

6. Сайт игры «Dead Cells». [Электронный ресурс] URL: https://dead-cells.com/ (дата обращения: 10.02.2024 г.).

7. Сайт игры «Hollow Knight». [Электронный ресурс] URL: https://www.hollowknight.com/ (дата обращения: 10.02.2024 г.).

8. Winkler M. What is Pixel Art? [Электронный ресурс] URL: https://design.tutsplus.com/articles/what-is-pixel-art--cms-21759 (дата обращения: 10.02.2024 г.).

9. Aseprite – Animated Sprite Editor & Pixel Art Tool. [Электронный ресурс] URL: https://www.aseprite.org/ (дата обращения: 10.02.2024 г.).

10. Фаулер М. UML. Основы. // Символ Плюс, 2004. – 192 с.

11. Грегори Д. Игровой движок. Программирование и внутреннее устройство. // Питер, 2022. – 1136 с.

12. Сайт игрового движка Unity. [Электронный ресурс] URL: https://unity.com/ru/our-company (дата обращения: 15.05.2024 г.).

13. Сайт игрового движка Unreal Engine. [Электронный ресурс] URL: https://www.unrealengine.com/en-US (дата обращения: 15.05.2024 г.).

14. Сайт игрового движка GameMaker. [Электронный ресурс] URL: https://gamemaker.io/ru (дата обращения: 15.05.2024 г.).

15. Сайт с игровыми ассетами Itch.io. [Электронный ресурс] URL: https://itch.io/game-assets (дата обращения: 15.05.2024 г.).

16. Schultz C. P., Bryant R., Langdell T. Game Testing: All in One. // Mercury Learning & Information, 2011. – 300 p.

ПРИЛОЖЕНИЕ. Скриншоты разработанной игры

Скриншоты разработанной 2D-игры приведены на рисунках 1–14.

Рисунок 1 – Главное меню игры

Рисунок 2 – Первый противник

Рисунок 3 – Карабканье по лестнице

Рисунок 4 – Толкание ящика в приседе

Рисунок 5 – Второй противник

Рисунок 6 – Подъемный мост перед замком

Рисунок 7 – Препятствие с шипами над ямой

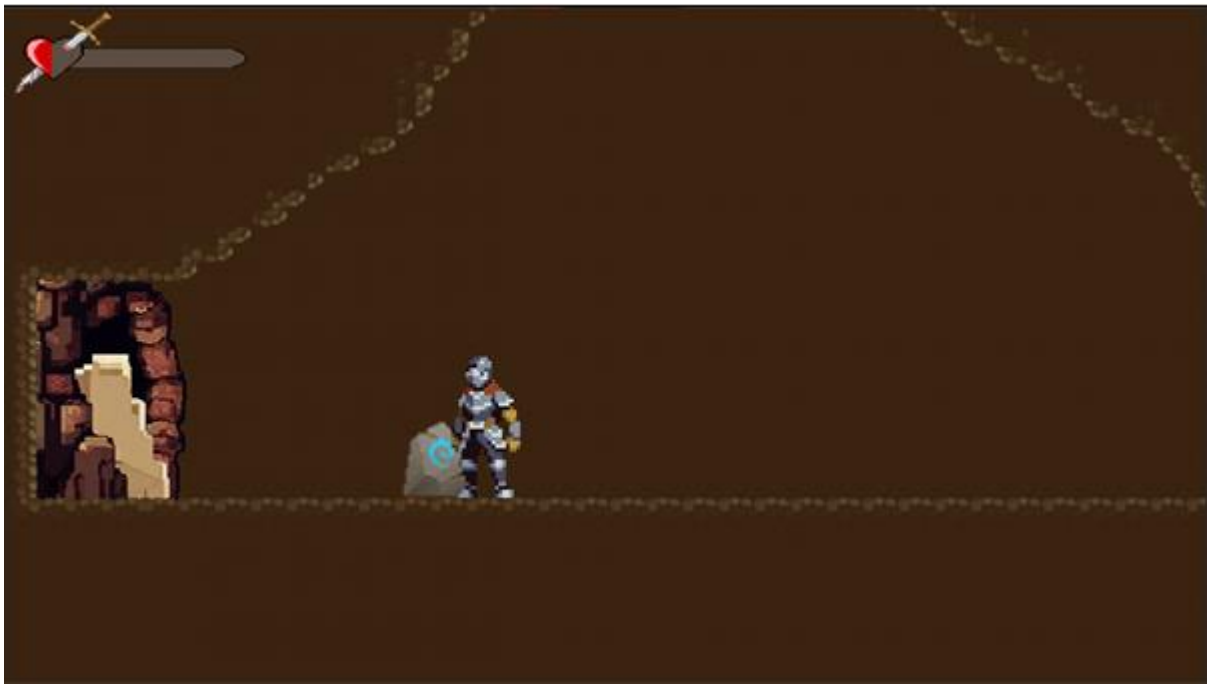

Рисунок 8 – Открытие прохода на другой уровень

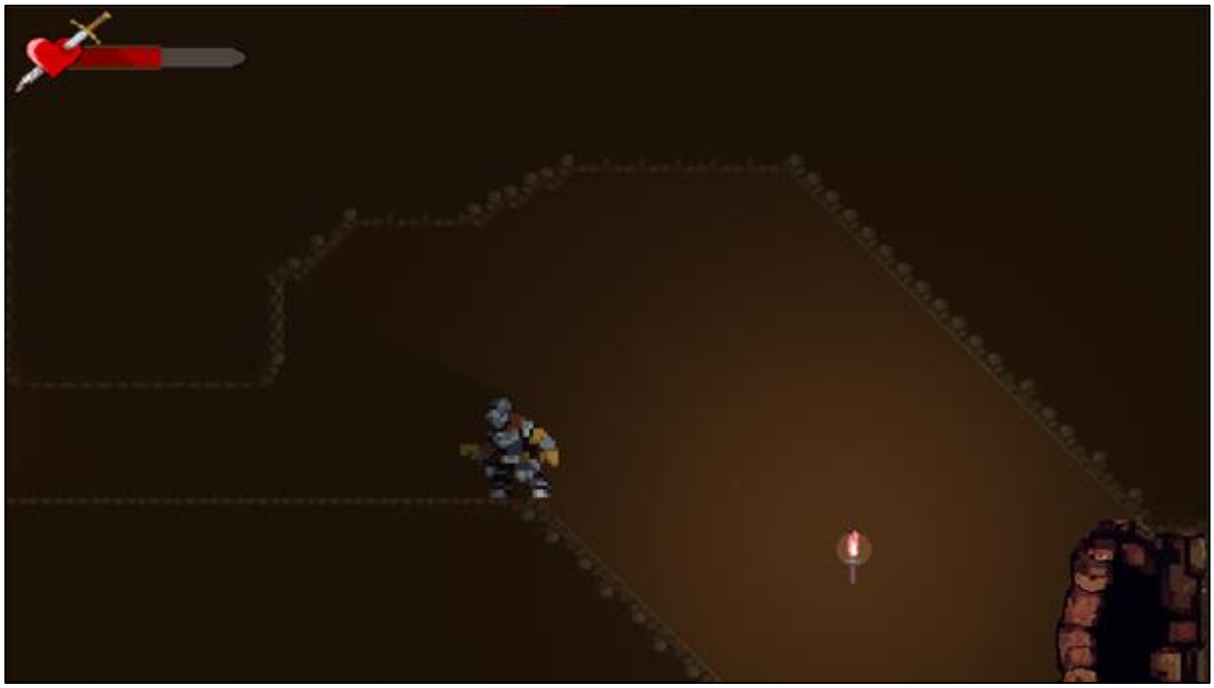

Рисунок 9 – Начало уровня в пещере

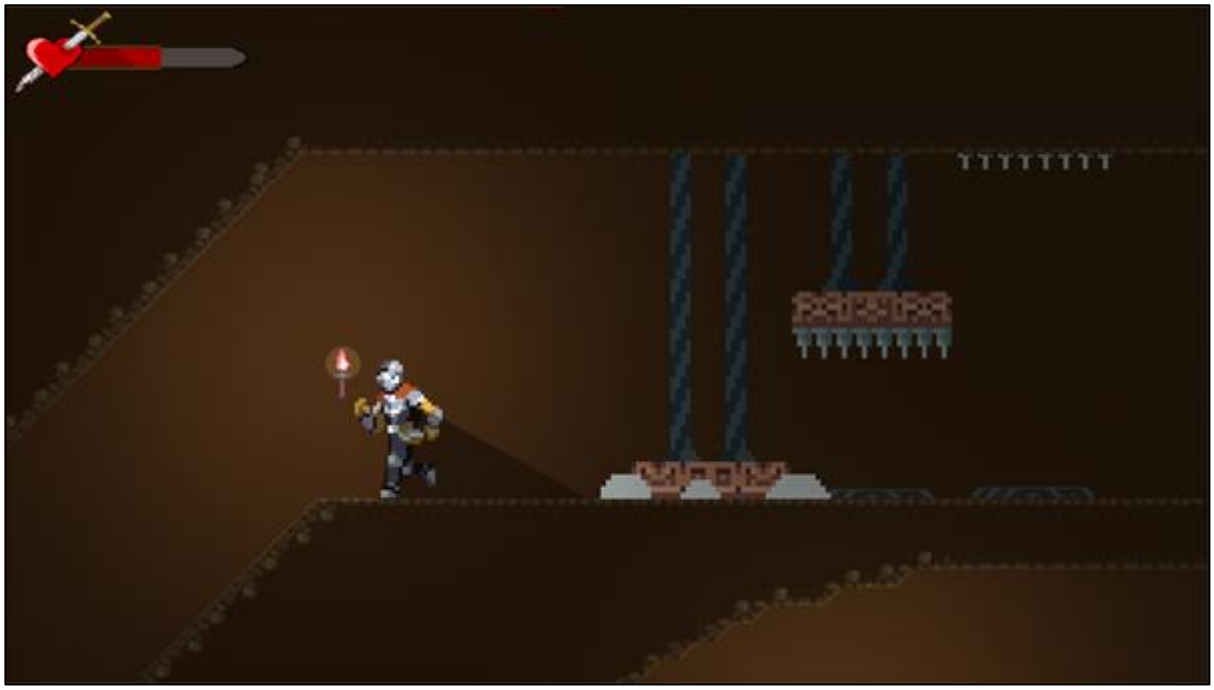

Рисунок 10 – Ловушки с нажимными плитами

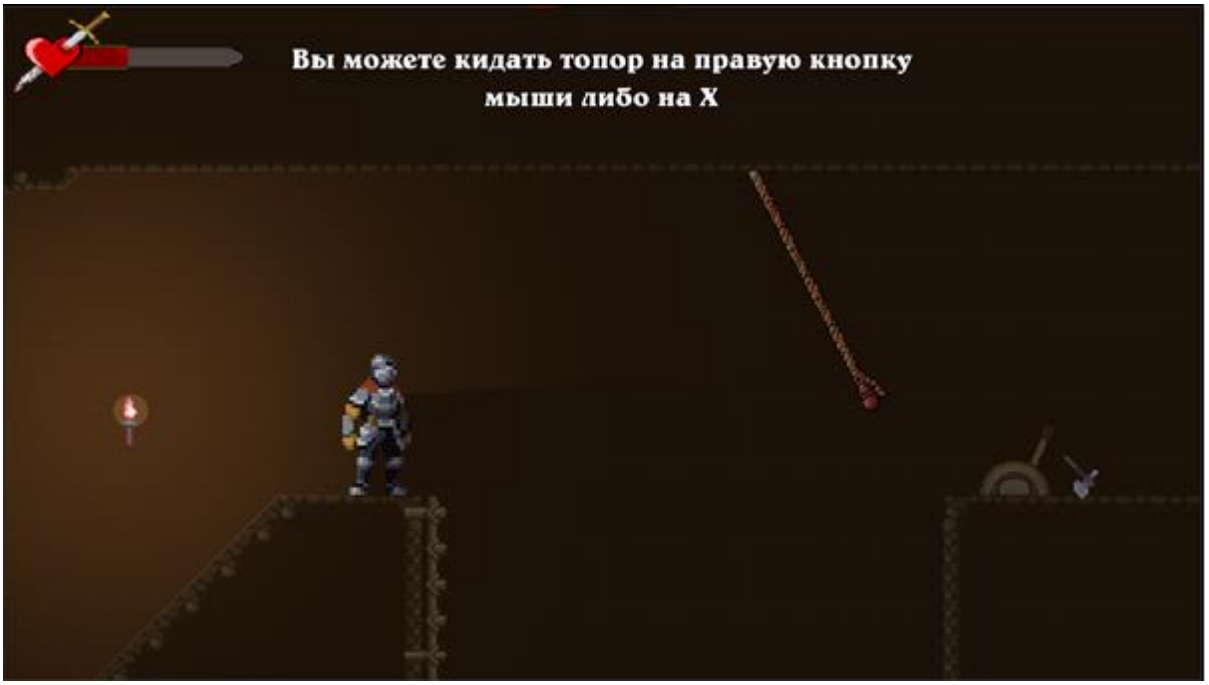

Рисунок 11 – Активация рычага с помощью топора

Рисунок 12 – Комната с зельем увеличения максимального здоровья

Рисунок 13 – Начало уровня в замке

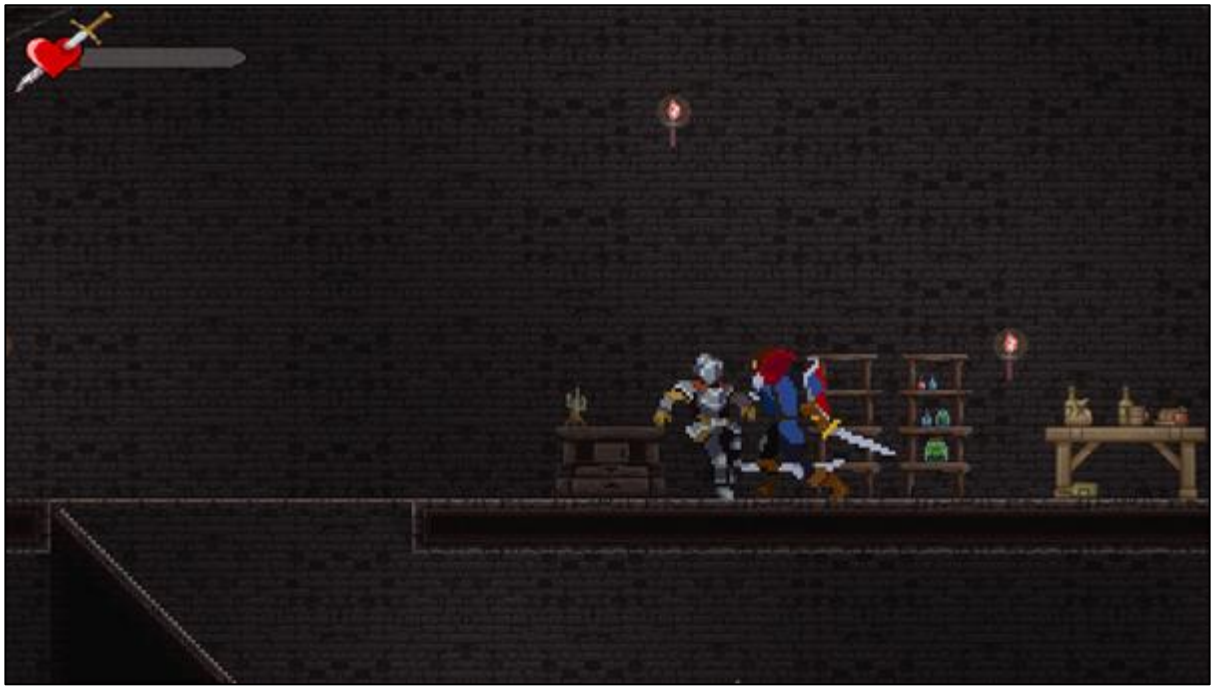

Рисунок 14 – Главный противник в игре