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**

**Тема работы**

КУРСОВАЯ РАБОТА

по дисциплине «Программная инженерия»

ЮУрГУ – 09.03.04.20241.308-0432.КР

|  |  |
| --- | --- |
| Нормоконтролер3, доцент  кафедры СП, к.ф - м.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.И. Радченко  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | Научный руководитель:  доцент кафедры СП, к.ф - м.н.  \_\_\_\_\_\_\_\_\_\_ Г.И. Радченко  Автор работы:  студент группы КЭ-303  \_\_\_\_\_\_\_\_\_\_ А.Ф. Бадамшина  Работа защищена  с оценкой: \_\_\_\_\_\_\_\_\_\_\_  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

Челябинск 2024

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

10.02.2024

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине «Программная инженерия»

студенту группы КЭ-303 Бадамшиной Ангелине Фидануровне,

обучающемуся по направлению 09.03.04 «Программная инженерия»

1. **Тема работы**Разработка компьютерной игры на базе игрового движка Unity.
2. **Срок сдачи студентом законченной работы:** 31.05.2024 г.
3. **Исходные данные к работе**
   1. Документация Unity. [Электронный ресурс] URL:

https://docs.unity3d.com/2022.1/Documentation/Manual/index.html (дата обращения: 16.05.2023 г.).

* 1. Мэннинг Д., Батфилд-Эддисон П. Unity для разработчика. Мобильные мультиплатформенные игры. // Питер, 2018. – 304 с.

1. **Перечень подлежащих разработке вопросов**
   1. Провести анализ предметной области и обзор аналогов.
   2. Сформулировать требования к приложению.
   3. Спроектировать приложение.
   4. Реализовать и протестировать приложение.
2. **Дата выдачи задания:** 9 февраля 2024 г.

Научный руководитель Г.И. Радченко

Доцент кафедры СП, к.ф - м.н.

Задание принял к исполнению А.Ф. Бадамшина

**ГЛОССАРИЙ**

1. *Префаб* - особый тип ассета, позволяющий хранить все компоненты и значения свойств игрового объекта, а также создавать его экземпляры. При этом экземпляры копируют свойства оригинального ассета [1].
2. *Unity* – один из ведущих игровых движков, обладающий широким спектром для интеграции, создания, поддержки и развития игровых продуктов, которые могут использоваться на любом этапе разработки продукта [2].
3. *2D-Платформер* – один из самых узнаваемых жанров компьютерных игр, основная задача игрока пройти уровни, преодолевая препятствия, передвигаясь по разным платформам, благодар чему жанр и получил данное название [3].
4. *Спрайт* – спрайты это графические 2D объекты (изображения, картинки) [4].

**ОГЛАВЛЕНИЕ**

[ГЛОССАРИЙ 3](#_Toc168882396)

[ВВЕДЕНИЕ 5](#_Toc168882397)

[1. АНАЛИЗ ПРЕДМТЕНОЙ ОБЛАСТИ 7](#_Toc168882398)

[1.1. Предметная область проекта 7](#_Toc168882399)

[1.2. Анализ аналогичных проектов и существующих решений для реализации проекта 7](#_Toc168882400)

[1.3. Вывод по первой главе 13](#_Toc168882401)

[2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ 14](#_Toc168882402)

[2.1. Концепция игры 14](#_Toc168882403)

[2.2. Функциональные требования к проектируемой системе 15](#_Toc168882404)

[2.3. Нефункциональные требования к проектируемой системе 16](#_Toc168882405)

[2.4. Диаграмма вариантов использования 16](#_Toc168882406)

[2.5. Вывод по второй главе 17](#_Toc168882407)

[3. АРХИТЕКТУРА СИСТЕМЫ 18](#_Toc168882408)

[3.1. Общее описание архитектуры системы 18](#_Toc168882409)

[3.2. Диаграмма компонентов 19](#_Toc168882410)

[3.3. Представление пользовательского интерфейса 21](#_Toc168882411)

[3.4. Вывод по третьей главе 26](#_Toc168882412)

[4. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ СИСТЕМЫ 27](#_Toc168882413)

[4.1. Реализация компонентов системы 27](#_Toc168882414)

[4.2. Демонстрация игрового процесса 39](#_Toc168882415)

[4.4. Вывод по четвертой системе 46](#_Toc168882416)

[ЗАКЛЮЧЕНИЕ 47](#_Toc168882417)

[ЛИТЕРАТУРА 48](#_Toc168882418)

[ПРИЛОЖЕНИЯ 50](#_Toc168882419)

[ПРИЛОЖЕНИЕ А. Спецификация вариантов использования 50](#_Toc168882420)

**ВВЕДЕНИЕ**

**Актуальность темы**

В современном обществе компьютерные игры пользуются популярностью не только у детей и подростков, но и у взрослых. Сегодня игры стали не просто способом развлечения, но и увлечением, а для некоторых они являются основным источником дохода. Они оказывают значительное воздействие на мировоззрение людей, становясь новым видом досуга и даже профессией.

Развитие рынка инди-игр предоставляет возможность разработчикам творить независимо от крупных издателей. Инди-игры (англ. Indie games, от англ. independent video games - «независимые компьютерные игры») — это проекты, созданные маленькими группами или отдельными разработчиками без поддержки издателей и распространяемые через цифровые платформы. Индустрия инди-игр активно развивается с начала 2000-х годов благодаря новым методам распространения и инструментам разработки.

**Цели и задачи**

Цель данной работы является разработка компьютерной 2D-игры в жанре «Платформер» с возможностью игры двумя игроками одновременно на платформе Unity. Для достижения этой цели необходимо решить следующие задачи:

1. Провести анализ предметной области и обзор аналогов;
2. Сформулировать требования к приложению;
3. Спроектировать приложение;
4. Реализовать и протестировать игровое приложение.

**Структура и содержание работы**

Работа включает введение, четыре главы, заключение и список литературы. Общий объем работы составляет \_ страницы, а список литературы включает \_ источник.

В первой главе «Анализ предметной области» описываются особенности игровых жанров, обосновывается выбор данной тематики и средств реализации. В этой главе также рассматриваются сильные стороны и недостатки аналогичных проектов.

Во второй главе «Анализ требований к программной системе» приводятся функциональные и нефункциональные требования к системе. Построена диаграмма вариантов использования и представлены основные спецификации. Рассматривается предметная область проекта и особенности сочетания жанров.

В третьей главе «Архитектура системы» описывается общая архитектура системы и ее компоненты. Представлены наброски игрового меню, меню настроек, меню выбора уровней и пользовательского интерфейса.

В четвертой главе «Реализация и тестирование системы» описывается реализация игрового приложения, особенности его компонентов, а также результаты функционального и юзабилити тестирования. По результатам тестирования сделаны выводы о достижении поставленных целей и внесены доработки и улучшения в итоговый проект.

В приложении А содержатся таблицы, описывающие основные варианты использования (ВИ) разрабатываемого игрового приложения.

В приложении Б представлены скриншоты финальной версии игры, демонстрирующие меню и игровой процесс.

В глоссарии даны определения некоторых терминов, характерных для работы в Unity.

1. **АНАЛИЗ ПРЕДМТЕНОЙ ОБЛАСТИ**
   1. **Предметная область проекта**

Платформер – это жанр игры, где основными элементами игрового процесса являются бег и прыжки по различным платформам, этажам, выступам, лестницам и другим объектам, а также преодоление ловушек, сбор предметов, сражение с врагами и решение различных задач и головоломок [3].

Игры-платформеры — это очень обширное понятие и как таковых, однозначных правил нет. Но в большинстве случаев они выполнены в двухмерном стиле, а уровни игры представлены в виде мира, который можно прокручивать как в вертикальном, так и в горизонтальном направлении. Основной целью в играх данного жанра является достижение главным героем конечной точки, начиная с начального местоположения, при этом необходимо собирать предметы, побеждать врагов, решать другие задачи и, само собой разумеется, избегать смерти персонажа.

Разрабатываемое игровое приложение будет сочетать в себе такие жанры, как «Головоломка», «Казуальная игра» и «2D-Платформер».

«Казуальная игра» – жанр, характеризующийся простыми правилами и легким для понимания игровым процессом. Игры этого жанра не требуют сложных стратегий, что делает их доступными для широкой аудитории. Одним из главных преимуществ казуальных игр является низкий порог вхождения, благодаря чему они подходят для пользователей практически любого возраста [5].

«Головоломка» – жанр компьютерной игры, целю которых является решение игроком логических задач, проявления интуиции или же просто удачи [6].

* 1. **Анализ аналогичных проектов и существующих решений для реализации проекта**

Ввиду простой реализации подобной концепции игрового мира на мировом рынке большое количество продуктов аналогичных разрабатываемому. Для анализа аналогичных проектов будут рассматриваться проекты для всех платформ. Ниже приведен список критериев, по которым были выбраны именно данные игровые проекты.

1. Игровой процесс проходит в 2D плоскости;
2. Цель игрока пройти игровые локации до конца;
3. Проекты стали наиболее популярными;
4. Для игроков есть препятствия или противники.

**Super Mario Bros.**

На рисунке 1 ниже представлен скриншот из игры «Super Mario Bros.» Данный проект сильно полюбился геймерам и получил широкую огласку [7]. 

Рисунок 1 – Скриншот из игры «Super Mario Bros.»

Основная задача игры – спасти принцессу. Игра представляет собой классическую формулу платформеров, где игроку необходимо добраться от точки А до точки Б, преодолевая препятствия в виде платформ, ловушек и врагов. Каждый уровень содержит различные виды препятствий, которые делают их уникальными. Главный герой может разбивать блоки, прыгая под ними и ударяясь головой. Некоторые блоки могут содержать улучшения для героя или пополнение здоровья.

Игра выполнена в пиксельном стиле, который обусловлен техническими ограничениями, но мир все равно интересен и красочен. Каждая локация имеет свой уникальный стиль, разноцветные платформы, различные фоны и разных врагов.

Саундтрек игры запоминается, но не надоедает, каждая локация имеет свою мелодию, а каждый враг - свой звук после гибели [8].

Сильные стороны:

1. Разнообразие в дизайне уровней;
2. Колоритные персонажи;
3. Вариативность прохождения уровней;
4. Запоминающийся саунтрек.

Слабые стороны:

1. Высокая сложность для новых игроков;
2. Однообразный геймплей.

**Cuphead**

На рисунке 2 представлен скриншот из игры «Cuphead». Этот проект завоевал любовь геймеров и получил широкую известность благодаря своему уникальному стилю и сложному игровому процессу [9].

Основная задача игры – победить множество боссов, чтобы вернуть контракты душ и спасти себя и своего брата Мугмена. Игра представляет собой комбинацию платформера и шутера, где игрокам необходимо преодолевать сложные уровни, сражаясь с разнообразными боссами. Каждый уровень содержит уникальные препятствия и врагов, что делает игру разнообразной и увлекательной. Главные герои могут стрелять в врагов, использовать специальные способности и уклоняться от атак. Некоторые уровни предлагают платформенные элементы, требующие точных прыжков и маневров.



Рисунок 2 – Скриншот из игры Cuphead

Игра выполнена в стиле анимации 1930-х годов, что придает ей уникальный визуальный шарм. Визуальное оформление тщательно продумано, каждый босс и уровень обладают уникальным дизайном и анимацией. Мир игры яркий и красочный, с разнообразными фонами и врагами, что создает насыщенную и атмосферную игровую среду.

Саундтрек игры состоит из оригинальной джазовой музыки, которая идеально дополняет визуальный стиль и темп игры. Каждое сражение с боссом сопровождается своей мелодией, что добавляет игре динамики и эмоционального накала [10].

Сильные стороны:

1. Уникальный визуальный стиль в духе анимации 1930-х годов;
2. Высокая сложность, которая обеспечивает интересный вызов;
3. Разнообразие и креативность боссов;
4. Запоминающийся саундтрек.

Слабые стороны:

1. Высокая сложность может отпугнуть некоторых игроков;
2. Повторяемость некоторых игровых элементов.

**Hollow Knight**

На рисунке 3 ниже представлен скриншот из игры «Hollow Knight». На первый взгляд, ничем не примечательная игра, которая ничем и не отличается от множества других платформеров [11].



Рисунок 3 – Скриншот из игры Hollow Knight

Основная задача игры – исследовать подземный мир Hallownest, сражаясь с различными врагами и боссами, чтобы раскрыть тайны этого мира и преодолеть его опасности. Игрокам необходимо путешествовать по обширным локациям, находить улучшения и новые способности, которые позволяют открывать ранее недоступные области. Каждый уровень полон уникальных врагов и препятствий, что делает игру разнообразной и увлекательной. Главный герой может использовать различные способности, такие как рывок, двойной прыжок и магические атаки, чтобы преодолевать трудности и сражаться с врагами.

Игра выполнена в мрачном, но красивом стиле, с детализированными фонами и анимацией. Визуальное оформление тщательно продумано, каждая область имеет свой уникальный дизайн и атмосферу. Мир игры разнообразен, с множеством разных локаций, от темных пещер до ярких лесов, что создает насыщенную и увлекательную игровую среду.

Саундтрек игры состоит из оригинальной оркестровой музыки, которая идеально дополняет визуальный стиль и атмосферу игры. Каждая область и сражение с боссом сопровождаются своей мелодией, что добавляет игре эмоционального накала и глубины [12].

Сильные стороны:

1. Огромный атмосферный мир с уникальными локациями;
2. Разнообразие типов врагов;
3. Большое количество секретов, предметов и скрытых проходов;
4. Интересная идея и захватывающий сюжет.
5. Превосходный дизайн.

Слабые стороны:

1. Сложность улучшения игровой мини-карты;
2. Запутанность локаций.

Для создания игрового приложения существует множество платных и бесплатных инструментов по разработке. Было принято решение использовать готовые инструменты для разработки игр. Для получения больших возможностей подберем оптимальную среду разработки.

**Unity** [2]

Unity является современным кроссплатформенным движком для создания игр и приложений, разработанный Unity Technologies. С помощью данного движка можно разрабатывать не только приложения для компьютеров, но и для мобильных устройств, игровых приставок и других девайсов.

В среду разработки Unity интегрирован игровой движок, иными словами, можно протестировать игру, не выходя из редактора. Также, Unity поддерживает импорт огромного количества различных форматов, что позволяет разработчику игры конструировать сами модели в более удобном приложении, а Unity использовать по прямому назначению – разработки продукта. Наконец, написание сценариев (скриптов) осуществляется на наиболее популярных языках программирования – C# и JavaScript.

**Unreal Engine** [13]

Unreal Engine 4 — это набор инструментов для разработки игр, имеющий широкие возможности.

Разработка в Unreal Engine 4 очень проста для начинающих. С помощью системы визуального создания скриптов Blueprints Visual Scripting можно создавать готовые игры, не написав ни строчки кода! Blueprints — это система визуального скриптинга Unreal Engine 4. Она является быстрым способом создания прототипов игр. Вместо построчного написания кода все можно делать визуально: перетаскивать ноды (узлы), задавать их свойства в интерфейсе и соединять их «провода».

* 1. **Вывод по первой главе**

По данным, полученным в ходе анализа существующих аналогов, можно сделать следующий вывод. Главная цель в большинстве платформеров заключается в достижении персонажем определенной точки, по пути преодолевая различные препятствия. Чтобы во время всего игрового процесса сохранялся интерес, препятствия должны быть увлекательными и не чрезмерно сложными. Музыкальное сопровождение в игре должно быть ненавязчивым, а звуковые эффекты должны гармонировать со всем процессом игры. Управление персонажем и настройки камеры должны делать игру динамичной, чтобы она не ощущалась недоработанной.

Исходя из обзора программного средства разработки, можно прейти к выводу, что на данный момент Unity остается лучшим бесплатным движком для разработки игровых приложений. Также было принято решение создать игру с возможностью играть сразу двум игрокам. Таким образом будет реализована механика кооперативной игры на одном компьютере.

1. **АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ**
   1. **Концепция игры**

**Введение**

"Dino Game" — платформер с элементами казуальной игры и головоломки, созданный для воссоздания ностальгии по детским играм на двоих.

**Сюжет и сеттинг**

Игроки берут под управление двух дружелюбных динозавро, которым предстоит преодолевать различные препятствия и находить безопасные места на высоких холмах, чтобы избежать наводнения и спастись. Сюжет игры разворачивается в доисторическом мире, где каждый уровень представляет собой живописный пейзаж, соответствующий одному из четырех сезонов года: лесные поля, засушливый регион и зимний регион. Такое разнообразие пейзажей позволяет игрокам насладиться различными климатическими условиями и атмосферой, усиливаемой слегка напряженной фоновой музыкой.

**Геймплей**

В процессе игры персонажи перемещаются по уровням, преодолевая препятствия, собирая бонусы и взаимодействуя с окружающей средой. Управление персонажами интуитивно понятно и динамично, что позволяет игрокам полностью погружаться в игровой процесс.

Основные элементы геймплея включают:

1. Прыжки по платформам: Игроки могут прыгать по различным платформам, избегая шипов и других опасностей.
2. Перемещение объектов: Динозавры могут перемещать коробки, чтобы создавать новые пути или активировать механизмы.
3. Открытие дверей и активация платформ: Игроки могут использовать кнопки для открытия дверей или активации подвижных платформ.
4. Выбор персонажей: Каждый игрок может выбрать внешний вид динозавра, что добавляет своего шарма.

**Кооперативный аспект**

Хотя персонажи не обладают особыми способностями, их координация и совместные действия являются ключом к успешному прохождению уровней. Например, один динозавр может подтолкнуть другого для преодоления высокого препятствия, или они могут совместно перемещать тяжелые объекты.

**Испытания и препятствия**

Уровни полны различных препятствий, включая:

1. Шипы: Наносят урон персонажам при соприкосновении;
2. Наводнение: Вода поднимается по таймеру, добавляя дополнительную напряженность и требуя от игроков быстроты и точности в действиях.
   1. **Функциональные требования к проектируемой системе**

Функциональные требования – это требования, которые определяют действия, которые должна выполнять система, без учета ограничений, связанных с ее реализацией, то есть определяют поведение системы в процессе обработки информации.

Функциональные требования к проектируемому игровому приложению:

1. Пользователи должны иметь возможность перемещается по игровому миру;
2. Пользователи должны иметь возможность взаимодействовать с внутриигровыми объектами;
3. Игровая система должна предоставлять доступ к запуску игры;
4. Игровая система должна предоставлять возможность управления двум игрокам одновременно;
5. Пользователи должны иметь возможность изменять громкость аудио;
6. Пользователи должны иметь возможность вернуться в главное меню в любой момент времени;
7. Реализован переход на следующие уровни;
8. Должна быть реализована система жизней персонажей.
   1. **Нефункциональные требования к проектируемой системе**

Нефункциональные требования – требования, определяющие качественные характеристики проектируемого игрового приложения.

Нефункциональные требования к проектируемому игровому приложению:

1. Игровая система должна производить загрузку уровня не более чем за 3 секунды;
2. Игровая система должна реагировать на действия игроков с задержкой не более 0,1 секунды;
3. Игровая система должна быть создана с использованием платформы Unity.
   1. **Диаграмма вариантов использования**

Для проектирования системы был использован язык графического описания UML. В соответствии с требованиями была построена диаграмма вариантов использования (представлена на рисунке 4 ниже), отражающая модель взаимодействия актера Игрок. Так как у первого пользователя и второго одинаковое взаимодействие с разрабатываемым игровым приложением.

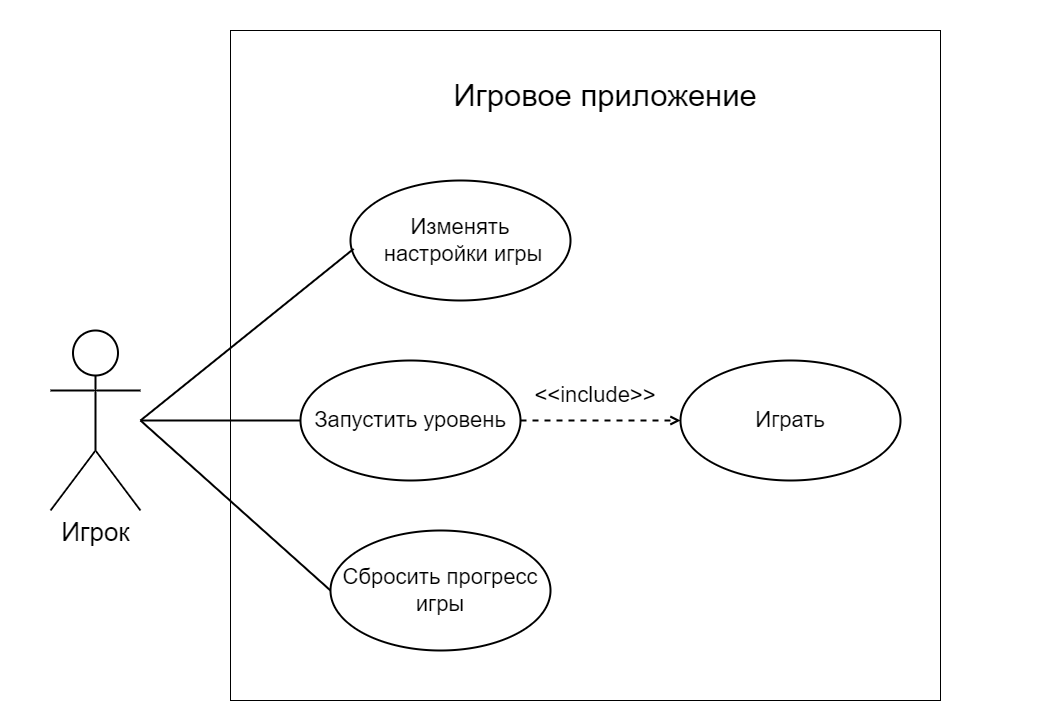


Рисунок 4 – Диаграмма вариантов использования

В ходе игры «Игрок» может изменять настройки игры, запустить уровень и играть, сбросить прогресс игры. Варианты использования связанные с прецендентом «Играть» такие, как передвигаться, восполнять запасы здоровья, взаимодействовать с предметами для решения головоломок, собирать кристаллы, не были указаны, чтобы исключить декомпозицию варианта.

Спецификации основных вариантов использования представлены в таблицах 1 – 4 приложения А.

* 1. **Вывод по второй главе**

Во второй главе были определены функциональные и нефункциональные требования к системе. На основе этих требований была построена диаграмма вариантов использования, определены основные актеры, взаимодействующие с системой, а также приведены краткое описание и спецификация вариантов использования.

1. **АРХИТЕКТУРА СИСТЕМЫ**
   1. **Общее описание архитектуры системы**

Архитектурно система состоит из сцен главного меню, меню выбора уровней и игровых уровней.

При запуске игрового приложения первой сценой загружается «Главное меню». В главном меню присутствует пользовательский интерфейс, состоящий из кнопок «Играть», «Информация», «Выйти», «Настройки». При нажатии «Играть», загружается меню выбора уровней, в «Настройках» есть возможность выбрать внешний вид персонажа, настроить громкость звуков и музыки. При нажатии кнопки «Выйти» игровое приложение завершает работу.

В меню выбора уровней также присутствует пользовательский интерфейс, состоящий из кнопок «Выход в основное меню», «1 уровень», «2 уровень», «3 уровень», «Сбросить прогресс прохождения». При нажатии на кнопку «Выход в основное меню», загружается основное меню, при нажатии на кнопку одного из уровней, загружается сцена выбранного уровня, а при нажатии кнопки «Сбросить прогресс прохождения» сбрасывается прогресс прохождения уровней.

Любая сцена игрового уровня позволяет игроку перейти в основное игровое меню нажатием кнопки «Esc» на клавиатуре или на кнопку «Настройки» на пользовательском интерфейсе. В предложенном меню есть три кнопки «Продолжить», «Возобновить уровень», «Меню». Первая кнопка позволяет выйти из меню паузы и продолжить игру, вторая перезапустить текущий уровень, третья переносит игроков в сцену главного меню, игровой уровень закрывается.

При удачно прохождение уровня в сцене игрового уровня всплывает панель, на которой присутствуют три кнопки «Возобновить уровень», «Меню», «Следующий уровень». Первая кнопка перезапустить текущий уровень, вторая переносит игроков в сцену главного меню, игровой уровень закрывается, третья загружает сцену следующего уровня.

При неудачном прохождение уровня в сцене игрового уровня всплывает панель, на которой присевают две кнопки «Возобновить уровень», «Меню». Первая кнопка перезапустить текущий уровень, вторая переносит игроков в сцену главного меню.

Файловая система игры [14]:

1. Animations – содержит в себе анимации игровых объектов и их параметры;
2. Scenes – содержит в себе сцены главного меню и уровней;
3. Material – содержит материалы и их свойства для поверхностей и объектов игры, аудиофайлы, используемые игровым приложением, изображения (спрайты) игровых объектов;
4. Prefabs – содержит в себе готовые префабы персонажей, ловушек, предметов;
5. Scripts – содержит в себе скрипты, описывающие поведение и свойства игровых объектов;
6. Scenes – содержит в себе заготовки сцен уровней;
7. Samples – содержит в себе базовые образцы функции Unity.
   1. **Диаграмма компонентов**

На рисунке 5 представлена диаграмма компонентов игрового приложения. Данные компоненты содержат в себе классы, описывающие поведение, свойства, параметры и зависимости игровых объектов.

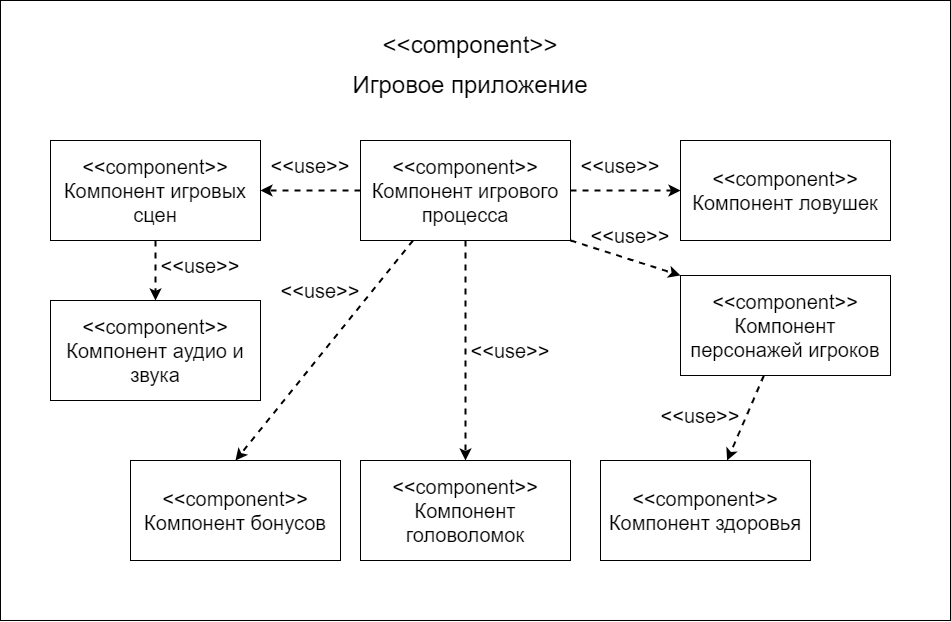


Рисунок 5 – Диаграмма компонентов

Описание компонентов, показанных на рисунке 5:

1. Компонент игровых сцен – содержит набор классов, реализующих главное меню, меню паузы, переключение между данными меню. А также отвечает за переход между сценами;
2. Компонент аудио и звука – позволяет в меню настроек изменять громкость звуков и музыки;
3. Компонент игрового процесса – содержит набор классов, реализующих игровой процесс системы. Управление персонажем, вызов отображения анимации, контроль здоровья и игровой логики;
4. Компонент ловушек – содержит набор классов, определяющих свойства, логику и поведение различных ловушек (шипов и поднятие воды);
5. Компонент персонажей игроков – содержит набор классов и параметров, хранящих и позволяющих настраивать персонажей, управляемых игроками. Например, высоту прыжка, скорость передвижения;
6. Компонент здоровья – содержит класс, описывающий и задающий параметры здоровья, а также вычитания здоровья при получении урона;
7. Компонент бонусов – содержит набор классов, описывающий поведение, свойства и параметры бонусов (сердца и кристаллы) для игрового персонажа;
8. Компонент головоломок – содержит набор классов, описывающий поведение, свойства и параметры головоломок (передвижение предметов, нажатие кнопок, открывание дверей).
   1. **Представление пользовательского интерфейса**

В данном разделе будут представлены спроектированные макеты пользовательского интерфейса игрового приложения. Данные макеты будут являться лишь примерным представлением итогового продукта и содержат в себе основные необходимые функции. В ходе реализации или тестирования данные элементы могут быть изменены или переработаны, также может быть изменено их положение на экране.

В главном меню игроку будут необходимы следующие элементы:

1. Заголовок меню – информирует игроков о том, что он находится в главном меню игрового приложения;
2. Играть – переход в меню выбора уровней;
3. Настройки – открывает меню настроек;
4. Информация – открывает панель, где описывается краткая информация о предметах, встречающихся в игре;
5. Выход – завершить работу игрового приложения;

На рисунке 6 ниже представлен макет главного меню игрового приложения.

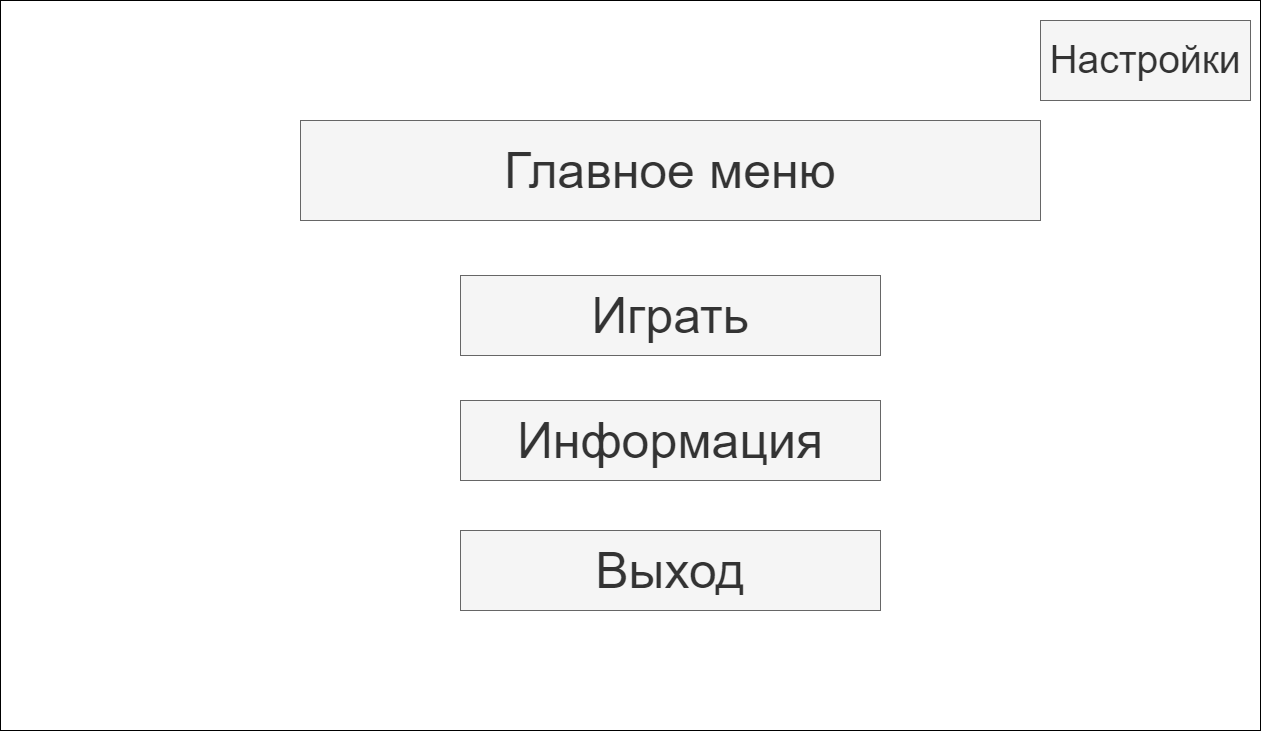


Рисунок 6 – Макет главного меню

В меню выбора уровней игроку будут необходимы следующие элементы:

1. Выход в главное меню – переход в главное меню;
2. 1 уровень – запуск первого игрового уровня;
3. 2 уровень – запуск второго игрового уровня;

3 уровень – запуск третьего игрового уровня;

1. Счетчик кристаллов – количество полученных кристаллов после прохождения определённого уровня;
2. Сбросить прогресс прохождения – сбрасывает прогресс прохождения уровней.

На рисунке 7 ниже представлен макет меню выбора уровней игрового приложения.



Рисунок 7 – Макет меню выбора уровней

В меню паузы будут необходимы следующие элементы:

1. Заголовок паузы – информирует игроков о том, что игра наохотится на паузе;
2. Продолжить – пауза прекращается, игроки возвращается к игре;
3. Вернуться в главное меню – возвращает игроков в главное меню;
4. Возобновить уровень – перезагружает сцену уровня;

На рисунке 8 ниже представлен макет меню паузы.



Рисунок 8 – Макет меню паузы

В меню панели удачного завершения уровня будут необходимы следующие элементы:

1. Заголовок победы– информирует игроков о том, что они завершили уровень;
2. Следующий уровень – загружает сцену следующего уровня;
3. Вернуться в главное меню – возвращает игрока в главное меню;
4. Возобновить уровень – перезагружает сцену уровня;

На рисунке 9 ниже представлен макет меню панели удачного завершения уровня.

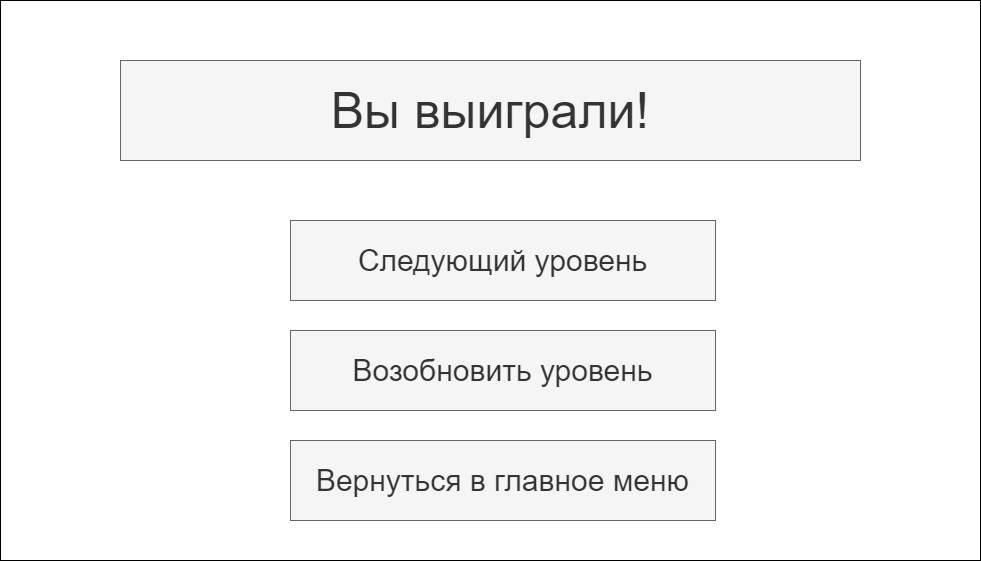


Рисунок 9 – Макет панели удачного завершения уровня

В меню панели не удачного завершения уровня будут необходимы следующие элементы:

1. Заголовок поражения– информирует игроков о том, что они не прошли уровень;
2. Вернуться в главное меню – возвращает игрока в главное меню;
3. Возобновить уровень – перезагружает сцену уровня;

На рисунке 10 ниже представлен макет меню выбора уровней игрового приложения.

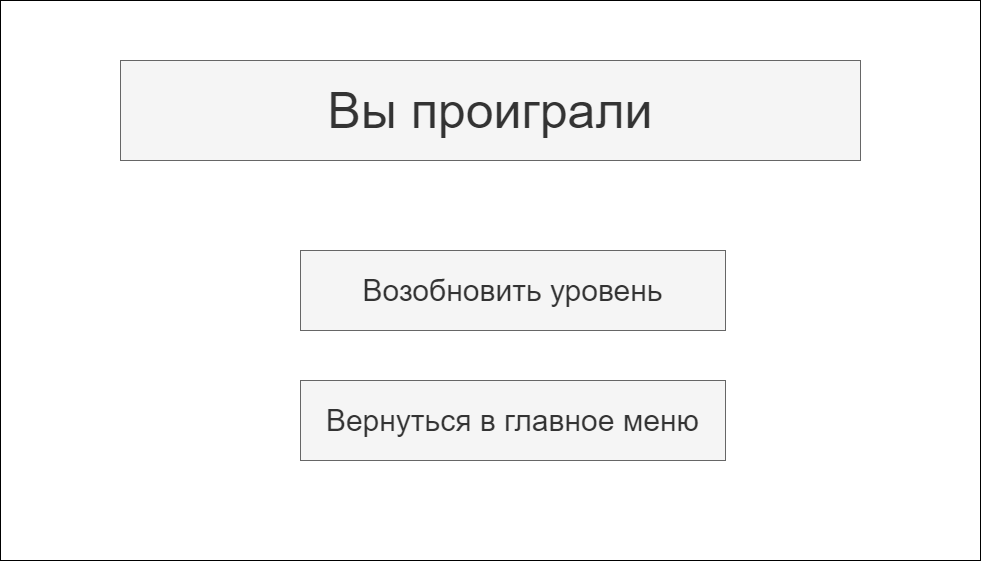


Рисунок 10 – Макет панели не удачного завершения уровня

Во время игрового процесса, когда игроки проходит игровые уровни, также есть элементы, которые необходимо отражать для игроков. В нашем случае необходимо отразить количество здоровья игровых персонажей, количество собранных кристаллов, таймер подъема воды, кнопка «Настройки». Таким образом игроки будет знать, сколько здоровья осталось у их персонажей, сколько кристаллов они собрали на уровне и сколько времени у них осталось до следующего подъема воды.

Макет интерфейса игрового процесса отображен на рисунке 11 ниже.

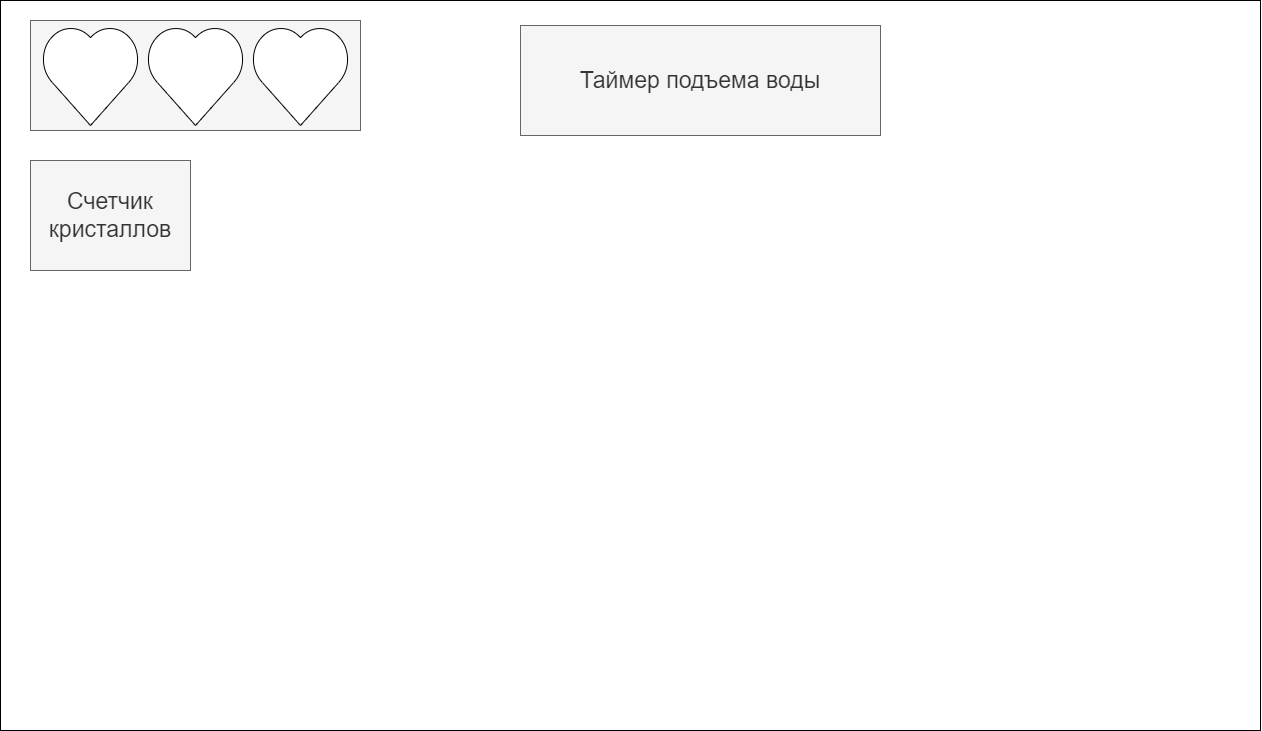


Рисунок 11 – Макет отражения необходимых элементов

* 1. **Вывод по третьей главе**

В третьей была спроектирована архитектура приложения, а также приведено ее краткое описание. Для описания структурного аспекта системы была построена диаграмма компонентов. Также были представлены макеты основного меню, меню выбора уровней, меню паузы, панели успешного и не успешного завершения уровней, пользовательского интерфейса во время игры и их краткое описание.

1. **РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ СИСТЕМЫ**

Для создания 2D-платформера была выбрана платформа Unity, поскольку она обладает всеми необходимыми инструментами для разработки игр данного жанра. Unity поддерживает широкий спектр платформ и предоставляет обширную официальную документацию, а также учебные материалы и уроки по разработке игр [2].

Для поиска графических элементов, таких как спрайты персонажей, предметов, фонов, будет использован магазин – «Itch.io». В нем можно найти бесплатные ресурсы, которые могут быть свободно использованы и распространены [15].

Основные инструменты и функции Unity, используемые в разработке 2D-платформера, включают:

1. Animation – используется для анимации спрайтов и выполнения программного кода при воспроизведении анимации.
2. Box Collider 2D – определяет область столкновения объектов в 2D-пространстве.
3. Animator – управляет анимациями в зависимости от действий пользователя и позволяет взаимодействовать с анимациями через программный код.
4. Rigidbody 2D – добавляет физические свойства объектам.
5. Collider – реализует столкновения объектов.
6. Audio Source – инструмент для создания, настройки и обработки аудио в игре.
   1. **Реализация компонентов системы**

**Реализация компонента игровых сцен**

Были реализованы два вида меню: основное меню с кнопками и возможностью переключения между сценами, а также меню выбора уровней с тремя игровыми сценами.

Основное меню реализовано с использованием скрипта Menu. Этот скрипт предоставляет базовые функции, такие как начало игры и выход из приложения. Метод PlayGame() используется для начала игры и загружает следующую сцену, увеличивая текущий индекс сцены на один. Метод ExitGame() завершает приложение, вызывая Application.Quit(). Остальные кнопки настраиваются через Button в Unity редакторе, что позволяет легко добавлять и управлять различными функциями в меню.

Меню выбора уровней реализовано с помощью скрипта MainMenu. Этот скрипт отвечает за отображение и разблокировку уровней, а также за отображение собранных кристаллов на уровнях. В методе Start() происходит инициализация, включая проверку прогресса игрока и разблокировку уровней на основе сохраненных данных. Также отображаются собранные кристаллы для каждого уровня. Метод LoadTo(int level) загружает сцену по её индексу, позволяя игроку перейти к выбранному уровню. Метод Reset() сбрасывает прогресс игрока, блокируя кнопки уровней и удаляя сохраненные данные о пройденных уровнях и собранных кристаллах.

**Реализация компонента аудио и звука**

Для реализации воспроизведения аудио и звуков в игре был использован встроенный в Unity компонент AudioSource. В скрипте MusicController используются два компонента AudioSource: один отвечает за воспроизведение звуковых эффектов, а другой — за проигрывание музыкального сопровождения. Эти компоненты работают совместно с UI-слайдерами для регулировки громкости. Компонент AudioSource, отвечающий за воспроизведение фоновой музыки, сохраняется между сценами с помощью метода DontDestroyOnLoad, что позволяет избежать постоянного пересоздания объектов. Внутри скрипта прописан программный код, который обеспечивает сохранение и загрузку уровней громкости через PlayerPrefs, позволяя игроку регулировать и сохранять предпочтения звуковых настроек между сеансами игры.

**Реализация компонента игрового процесса**

Для реализации игрового процесса в проекте были разработаны несколько ключевых компонентов, включая управление камерой и отображение здоровья на экране. Про отображение времени наводнения подробнее написано ниже, в реализации компонента ловушек.

Класс CameraController отвечает за управление камерой. В методе Awake() инициализируется компонент камеры и сохраняется начальная позиция. Метод Update() вызывается каждый кадр и обеспечивает плавное следование камеры за игроками с помощью линейной интерполяции (Lerp). Камера также регулирует масштаб в зависимости от расстояния между игроками, обеспечивая оптимальное отображение игрового пространства. Метод Reset() сбрасывает масштаб и позицию камеры к начальным значениям. На рисунке 12 представлены на стройки камеры из редактора Unity.

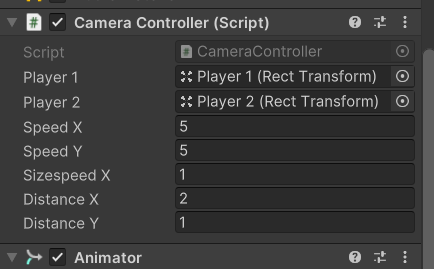


Рисунок 12 – Настройки камеры

Класс DisplayHealthCharacters управляет отображением здоровья персонажей на экране с помощью слайдера. В методе Start() устанавливается максимальное значение здоровья, а метод Awake() инициализирует компонент слайдера. Метод Update() обновляет значение здоровья на слайдере каждый кадр. Методы SetMaxHealth() и SetHealth() устанавливают максимальное значение здоровья и текущее значение соответственно.

Также были реализованы классы для отображения вспомогательных панелей, таких как пауза, победы и поражения.

Класс LevelController управляет игровыми уровнями. Он отслеживает завершение уровней игроками и запускает логику победы, если оба персонажа завершили уровень. Также он сохраняет самый высокий достигнутый уровень и количество собранных кристаллов.

Класс LossSettings отвечает за управление проигрышем в игре. Он активирует панель поражения при поражении игрока и позволяет перезапустить уровень или перейти в меню игры.

Класс SceneSettings обеспечивает управление настройками сцены, такими как пауза и продолжение игры. Он позволяет игроку приостанавливать игру, перезапускать уровень или переходить в меню игры.

Класс VictorySettings управляет логикой победы в игре. Он активирует панель победы при успешном завершении уровня и позволяет перезапустить уровень, перейти к следующему уровню или выйти в главное меню.

**Реализация компонента ловушек**

Для реализации ловушек в игре были использованы два типа: шипы и вода. Шипы и вода наносят урон, описанный в классе Damage, а управление поведением воды осуществляется классом WaterRiseController.

Шипы размещаются на карте и наносят игроку урон в 1 единицу при столкновении. Вода представляет собой наводнение, которое постепенно поднимается по таймеру и наносит урон в 6 единиц.

Класс WaterRiseController управляет подъемом воды. Он использует переменные для начальной скорости подъема (initialRiseSpeed), ускорения подъема (riseAcceleration) и времени между подъемами (riseInterval). В методе Start() задаются начальная скорость и время следующего подъема воды. В методе Update() отсчитывается время до следующего подъема, и вода поднимается, когда время истекает. Метод RiseWater() поднимает воду по вертикали, а метод UpdateCountdown() обновляет текст на экране, показывая оставшееся время до следующего подъема. Класс также имеет методы для остановки и возобновления таймера, а также для сброса уровня воды.

Класс Damage отвечает за нанесение урона персонажам при столкновении с ловушками. При столкновении с ловушкой, если у объекта есть компонент HealthHero, происходит проверка на тэги персонажей, и урон применяется обоим персонажам. После этого сбрасывается уровень воды с помощью вызова метода ResetWaterLevel() из класса WaterRiseController, и персонажи перемещаются к точкам возрождения.

**Реализация компонента бонусов**

В игре реализована система бонусов в виде кристаллов, которые игроки могут собирать для увеличения своих очков. Эти кристаллы аналогичны системе звезд в играх: сбор всех трёх кристаллов на уровне является признаком полного прохождения уровня и получения максимального результата. Система состоит из двух основных классов: CrystalPicker и CrystalCounter.

Класс CrystalPicker отвечает за обнаружение столкновений с игроками и обработку подбора кристаллов. Когда объект с этим скриптом сталкивается с игроком, скрипт проверяет, имеет ли объект тэг Player1 или Player2. Если да, то вызывается метод IncreaseCrystalCount() из класса CrystalCounter, который увеличивает счетчик кристаллов. После этого воспроизводится звуковой эффект и кристалл уничтожается.

Класс CrystalCounter хранит количество собранных кристаллов и отображает его на экране с помощью текстового поля UI. Метод IncreaseCrystalCount() увеличивает значение переменной crystal и обновляет текстовое поле. Таким образом, каждый раз, когда игрок подбирает кристалл, значение счетчика увеличивается, и на экране отображается актуальное количество собранных кристаллов.

**Реализация компонента головоломок**

Были разработаны два вида головоломок.

Первая, не сильно трудная, позволяет игрокам передвигать объект, который настраивается в редакторе Unity с помощью компонента Rigidbody 2D. Этот компонент позволяет управлять физическими свойствами объекта, такими как масса, гравитация, и другими параметрами, влияющими на его движение и взаимодействие с другими объектами. На рисунке 13 ниже представлен скриншот настроек этого компонента.

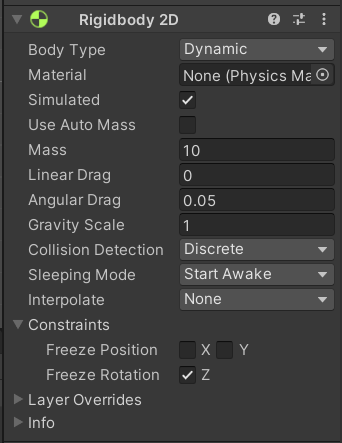


Рисунок 13 – Настройки Rigidbody 2D для передвигаемого объекта

Вторая включает в себя механизмы управления объектами с помощью кнопок. Эта система состоит из двух основных компонентов: кнопочных плит (ButtonPlate) и дверей (Door).

Класс ButtonPlate отвечает за обработку нажатий кнопок. Когда персонаж или объект (например, ящик) наступает на кнопку, срабатывает триггер, меняющий состояние кнопки и визуально отображающий ее нажатие. На рисунке 14 представлен листинг этого класса.

Рисунок 14 – Листинг класса ButtonPlate

using UnityEngine;

public class ButtonPlate : MonoBehaviour

{

private int pressed = 0;

public bool Pressed => pressed > 0;

[SerializeField] private Sprite releasedImage;

[SerializeField] private Sprite pressedImage;

private SpriteRenderer img;

private void Awake()

{

img = GetComponent<SpriteRenderer>();

}

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.GetComponent<MovementHero>() == null && !collision.gameObject.name.Contains("Ящик"))

return;

img.sprite = pressedImage;

}

private void OnTriggerExit2D(Collider2D collision)

{

if (collision.GetComponent<MovementHero>() == null && !collision.gameObject.name.Contains("Ящик"))

return;

img.sprite = releasedImage;

pressed--;

}

}

Класс Door управляет движением двери в зависимости от состояния кнопок. Если хотя бы одна кнопка нажата, дверь начинает открываться, если все кнопки отпущены – дверь закрывается. Так как в данном классе необходимо указывать стартовые, конечные координаты передвижения и скорость движения, можно использовать не только для создания дверей, но и для перемещающихся платформ. На рисунке 15 представлен скриншот настроек этого класса в редакторе Unity.

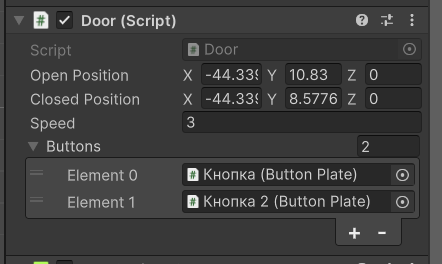


Рисунок 15 – Настройки класса Door

**Реализация компонента персонажей игроков**

Компонент персонажей игроков реализован с использованием скрипта MovementHero, который управляет перемещением и анимацией персонажей, а также проверяет их состояние на земле. Этот скрипт обеспечивает функции движения и прыжков для двух персонажей, каждый из которых управляется отдельным игроком.

В скрипте используется компонент Rigidbody2D для управления физикой персонажа и компонент Animator для управления анимациями. Переменная FacingRight отслеживает направление персонажа, чтобы корректно отразить его спрайт при изменении направления движения. Параметры перемещения, такие как скорость (speed) и сила прыжка (jumpForce), задаются в редакторе Unity. Все параметры представлены на рисунке 16 ниже.

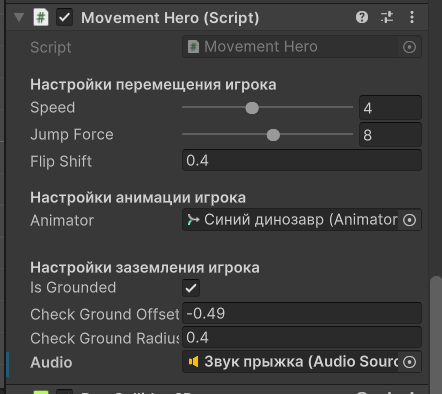


Рисунок 16 – Настройки параметров персонажа

Метод Flip() отвечает за поворот персонажа в другую сторону. Он изменяет значение переменной FacingRight, инвертирует масштаб персонажа по оси X и сдвигает персонажа в сторону поворота для корректной работы асимметричных коллайдеров. На рисунке 17 представлен листинг метода Flip().

Рисунок 17 – Листинг метода Flip()

private void Flip()

{

FacingRight = !FacingRight; // Меняем направление персонажа

Vector3 theScale = transform.localScale;

theScale.x \*= -1;

if (FacingRight)

transform.position += flipShift \* Vector3.right;

else

transform.position += flipShift \* Vector3.left;

transform.localScale = theScale;

}

В методе Start() происходит инициализация компонента Rigidbody2D. Метод Update() обрабатывает ввод от пользователя, отвечающий за перемещение и прыжки персонажа. Внутри этого метода определяется, какой из игроков управляет персонажем, и в зависимости от этого обрабатывается соответствующий ввод. Для первого игрока используются клавиши "W" для прыжка и горизонтальная ось ввода "Horizontal" для перемещения, для второго игрока — клавиши "UpArrow" для прыжка и горизонтальная ось ввода "Horizontal2". Если персонаж находится на земле и нажата соответствующая клавиша, к компоненту Rigidbody2D применяется сила прыжка, а также воспроизводится звук прыжка. Затем, в зависимости от ввода, обновляется скорость персонажа, и устанавливается параметр анимации для перемещения. Команда Input.GetAxisRaw("Horizontal") используется для получения значения оси перемещения, что позволяет обрабатывать аналоговый ввод от контроллера или клавиатуры. На рисунке 18 представлен листинг метода Update().

Рисунок 18 – Листинг метода Update()

void Update()

{

float moveInput = 0f;

if (gameObject.CompareTag("Player1"))

{

moveInput = Input.GetAxisRaw("Horizontal");

if (Input.GetKeyDown(KeyCode.W) && isGrounded)

{

rb.velocity = new Vector2(rb.velocity.x, jumpForce);

Audio.Play();

}

}

else if (gameObject.CompareTag("Player2"))

{

moveInput = Input.GetAxisRaw("Horizontal2");

if (Input.GetKeyDown(KeyCode.UpArrow) && isGrounded)

{

rb.velocity = new Vector2(rb.velocity.x, jumpForce);

Audio.Play();

}

}

rb.velocity = new Vector2(moveInput \* speed, rb.velocity.y);

animator.SetFloat("HorizontalMove", Mathf.Abs(moveInput));

if (moveInput > 0 && !FacingRight)

{

Flip();

}

else if (moveInput < 0 && FacingRight)

{

Flip();

}

}

Метод FixedUpdate() вызывается на каждом кадре физики и используется для проверки заземления персонажа через метод CheckGround(). Метод CheckGround() определяет, находится ли персонаж на земле, проверяя наличие коллайдеров в указанной области под персонажем. Если количество коллайдеров больше одного, считается, что персонаж находится на земле. На рисунке 19 представлен листинг метода CheckGround().

Рисунок 19 – Листинг метода CheckGround()

private void CheckGround()

{

Collider2D[] colliders = Physics2D.OverlapCircleAll(new Vector2(transform.position.x, transform.position.y + checkGroundOffsetY), checkGroundRadius);

if (Physics2D.OverlapPointAll(new Vector2(transform.position.x, transform.position.y + checkGroundOffsetY), LayerMask.GetMask("Dinosaur")).Length == 1

&& Physics2D.OverlapPointAll(new Vector2(transform.position.x, transform.position.y + checkGroundOffsetY - 0.2f), LayerMask.GetMask("Dinosaur")).Length > 0)

{

rb.includeLayers = LayerMask.GetMask("Dinosaur");

}

else

{

rb.includeLayers = LayerMask.GetMask("Nothing");

}

if (colliders.Length > 1)

{

isGrounded = true;

}

else

{

isGrounded = false;-

}

}

Для визуализации области проверки заземления в редакторе Unity используется метод OnDrawGizmos(). Этот метод рисует линию и сферу, обозначающие область проверки, что упрощает настройку параметров заземления в редакторе. На рисунке 20 представлен листинг метода OnDrawGizmos().

Рисунок 20 – Листинг метода OnDrawGizmos ()

void OnDrawGizmos()

{

Gizmos.color = Color.red;

Gizmos.DrawLine(transform.position, transform.position + Vector3.up \* checkGroundOffsetY);

Gizmos.DrawWireSphere(transform.position + Vector3.up \* checkGroundOffsetY, checkGroundRadius);

}

**Реализация компонента здоровья**

Данный компонент содержит в себе классы HealthHero, HeroTreatment, Damage.

Класс HealthHero управляет здоровьем главных персонажей в игре. При инициализации устанавливается максимальное значение здоровья. Когда хотя бы один персонаж получает урон, общее текущее здоровье уменьшается. Если здоровье достигает или падает ниже нуля, вызывается метод Die() для обработки смерти. При увеличении здоровья (при лечении), оно не может превысить максимальное значение. При получении урона также воспроизводится анимация и сбрасывается положение камеры.

Класс Damage отслеживает нанесение урона персонажем при столкновении с определенными объектами. После столкновения у персонажей уменьшается здоровье на указанное количество единиц. После получения урона герои перемещаются к своим точкам возрождения, а уровень воды сбрасывается.

Класс HeroTreatment отвечает за лечение персонажей. Когда персонаж взаимодействует с объектом лечения, общее здоровье увеличивается на указанное количество единиц. Процесс лечения сопровождается звуковым эффектом. После использования объект лечения он уничтожается. На рисунке 21 ниже представлен листинг метода, обеспечивающего увеличение здоровья.

Рисунок 21 – Метод увеличения здоровья

private void OnTriggerEnter2D(Collider2D other)

{

if (other.CompareTag(tagPlayer1) || other.CompareTag(tagPlayer2))

{

HealthHero health = other.GetComponent<HealthHero>();

if (health != null)

{

Audio.Play();

health.SetHealth(collisionHeal);

}

Destroy(gameObject);

}

}

* 1. **Демонстрация игрового процесса**

Для демонстрации игрового процесса была запущена игра и проведен ряд действий, совершенных в ходе игры.

При запуске игры пользователей встречает вступительная сцена, кратко рассказывающая предысторию персонажей. Скриншот вступительной сцены представлен на рисунке 22.

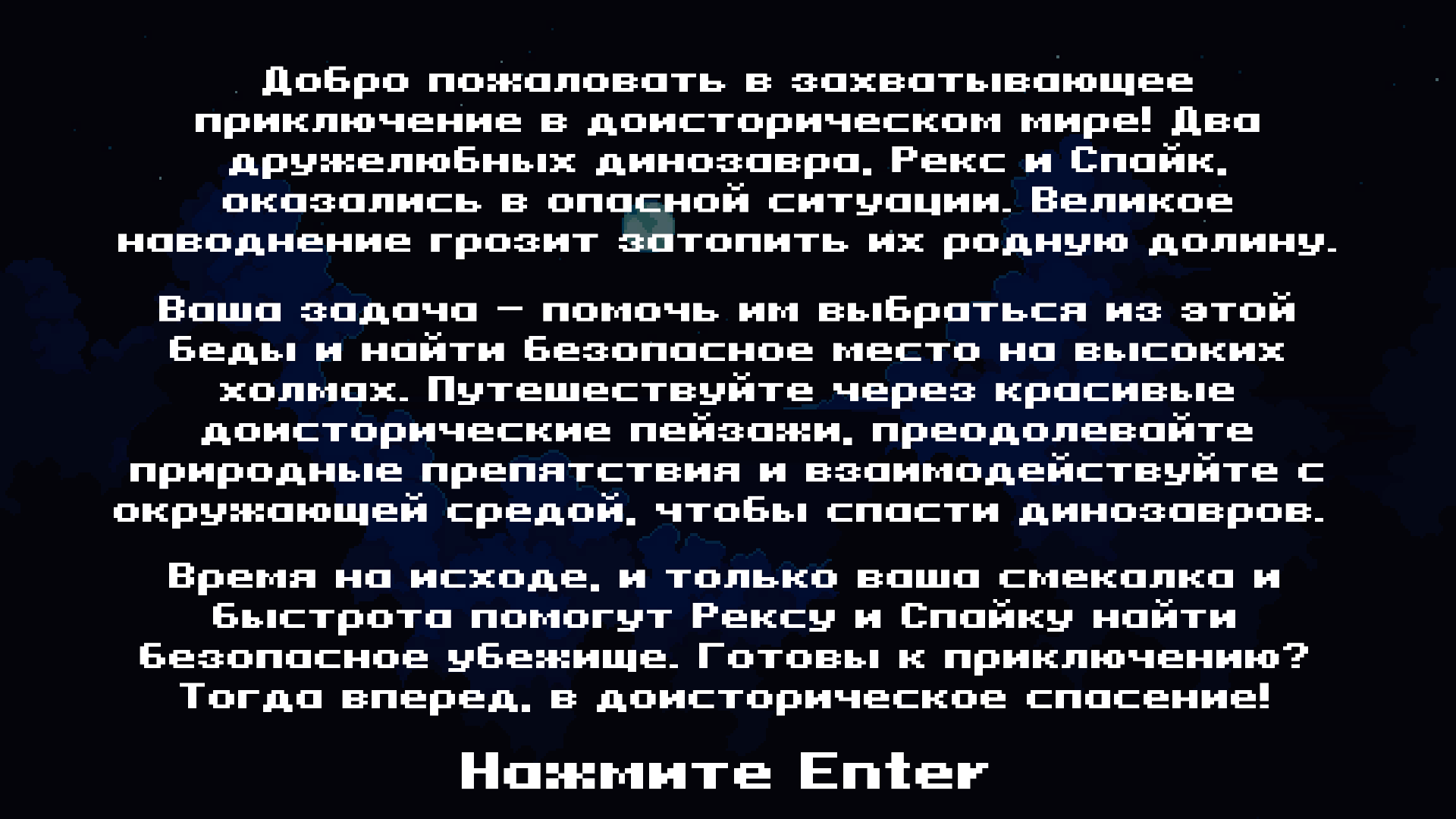


Рисунок 22 – Вступительная сцена

После ознакомления с предысторией пользователей встречает главное меню. В этом меню можно посмотреть краткую информацию, настроить звук, выбрать модели персонажей, а также начать игру или выйти из приложения. Скриншот главного меню представлен на рисунке 23.



Рисунок 23 – Главное меню

При нажатии на кнопку «Info» на экране отображается информация об игровом процессе. В этом разделе рассказывается о бонусах и опасностях. Скриншот панели «Info» представлен на рисунке 24.

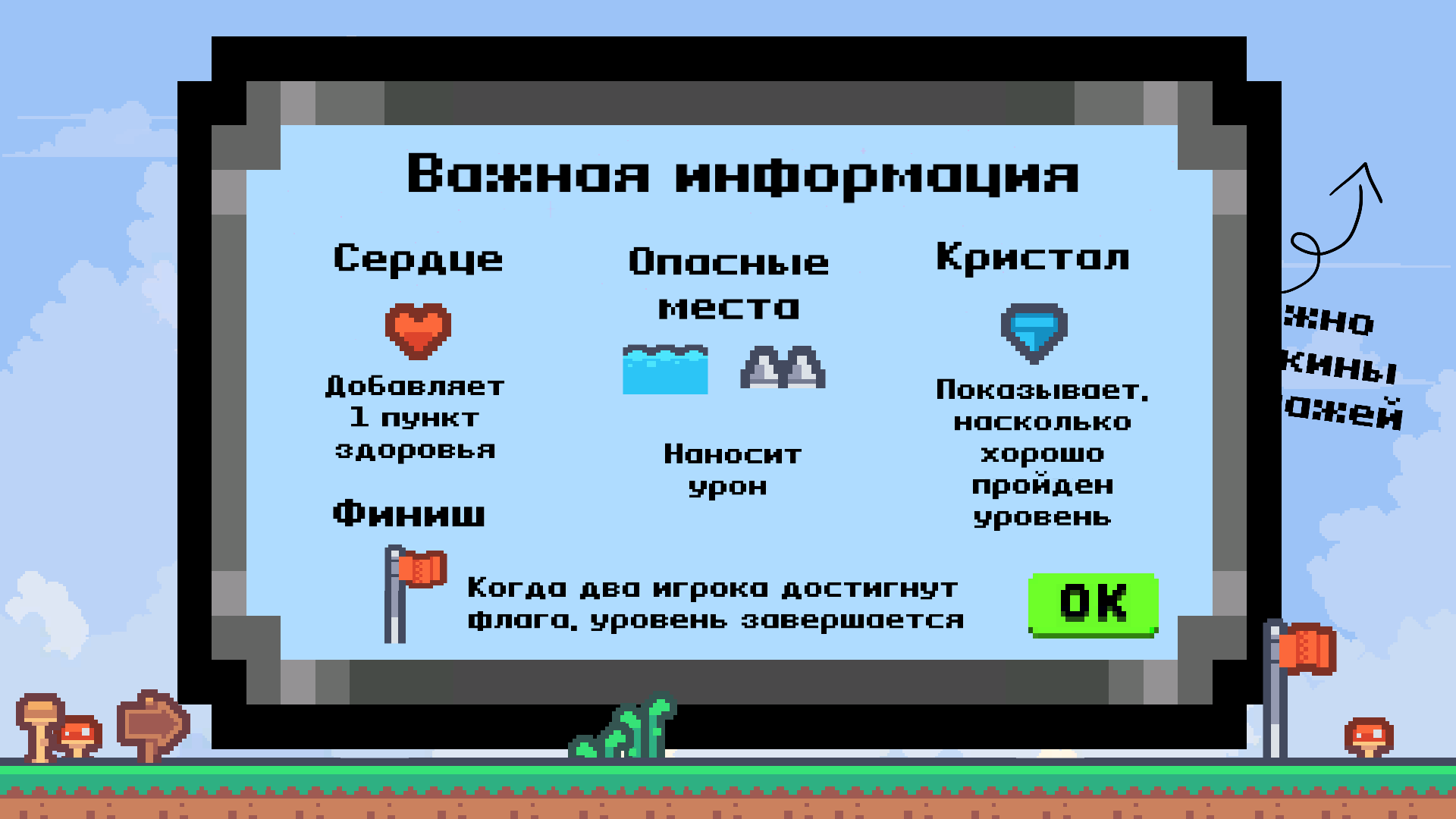


Рисунок 24 – Панель «Info»

При нажатии на кнопку настроек, расположенную в правом верхнем углу экрана, открывается панель настроек. В этом меню пользователи смогут выбрать модель персонажей, ознакомиться с управлением для первого и второго игроков, а также настроить громкость музыки и игровых звуков. Скриншот панели настроек представлен на рисунке 25.

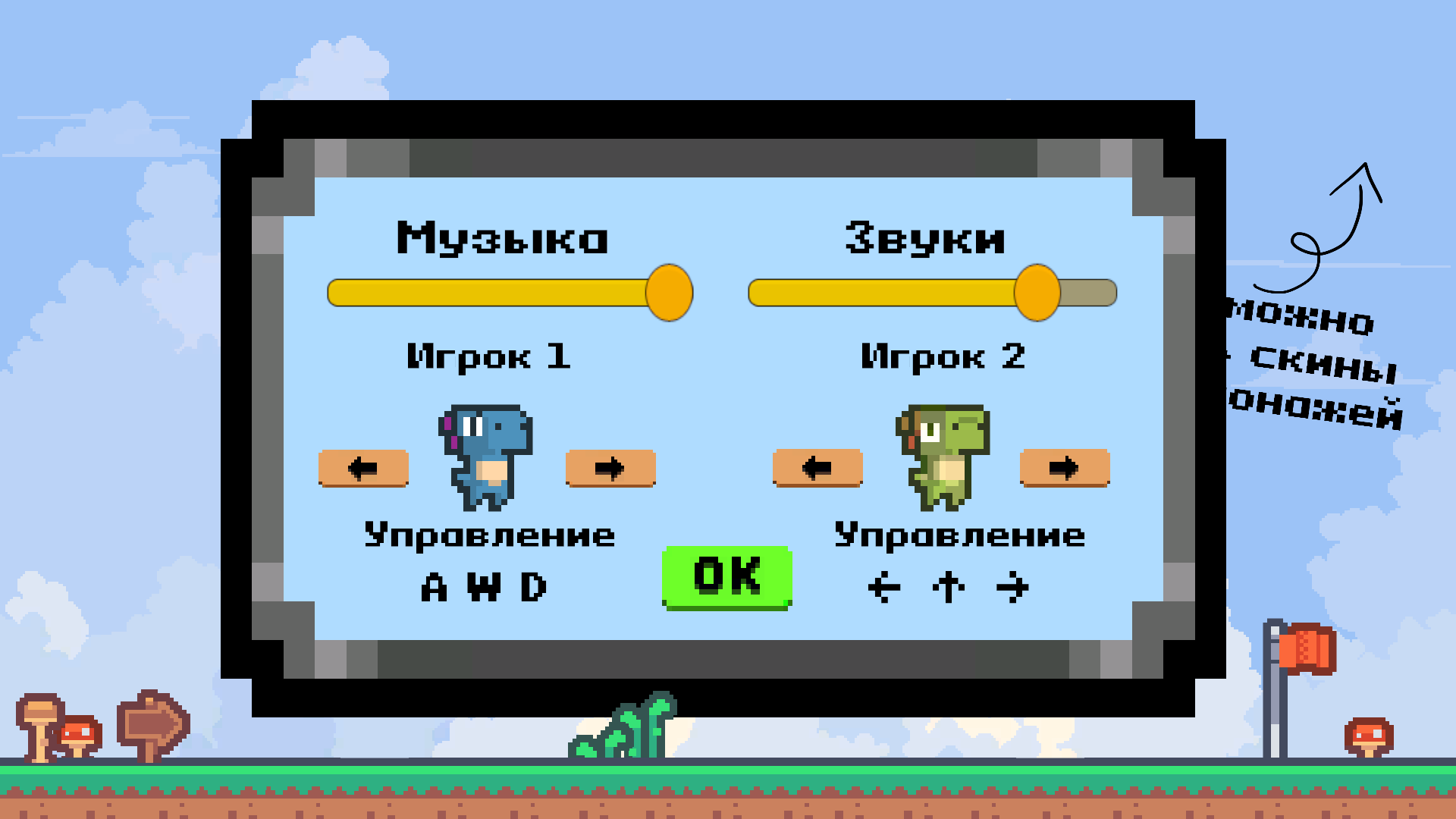


Рисунок 25 – Панель настроек

При нажатии на кнопку «Play» главное меню сменяется на меню выбора уровней. На данный момент разблокирован только первый уровень, так как остальные будут открываться по мере прохождения предыдущих. При нажатии на кнопку «Домик» загружается основное меню, а при нажатии на кнопку «Сбросить прогресс» происходит сброс прогресса прохождения уровней. Скриншот меню выбора уровней представлен на рисунке 26.

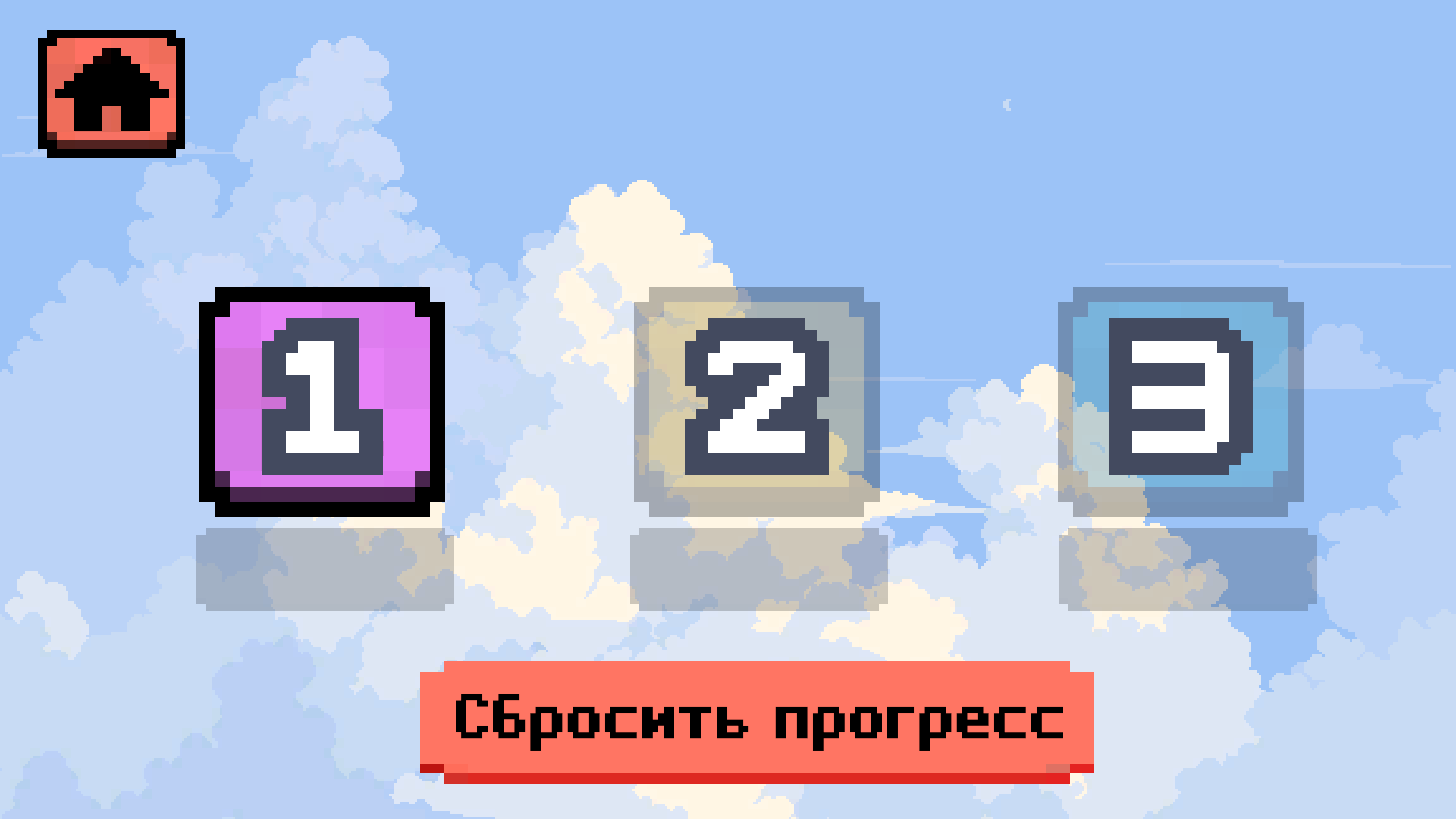


Рисунок 26 – Меню выбора уровней

При нажатии на кнопку «1 уровень» загружается сцена первого уровня. На экране будут отображаться количество здоровья, собранные кристаллы, а также таймер до следующей волны. На первом уровне пользователей также будет предоставлена небольшая подсказка по управлению персонажами. Скриншот первого уровня представлен на рисунке 27.

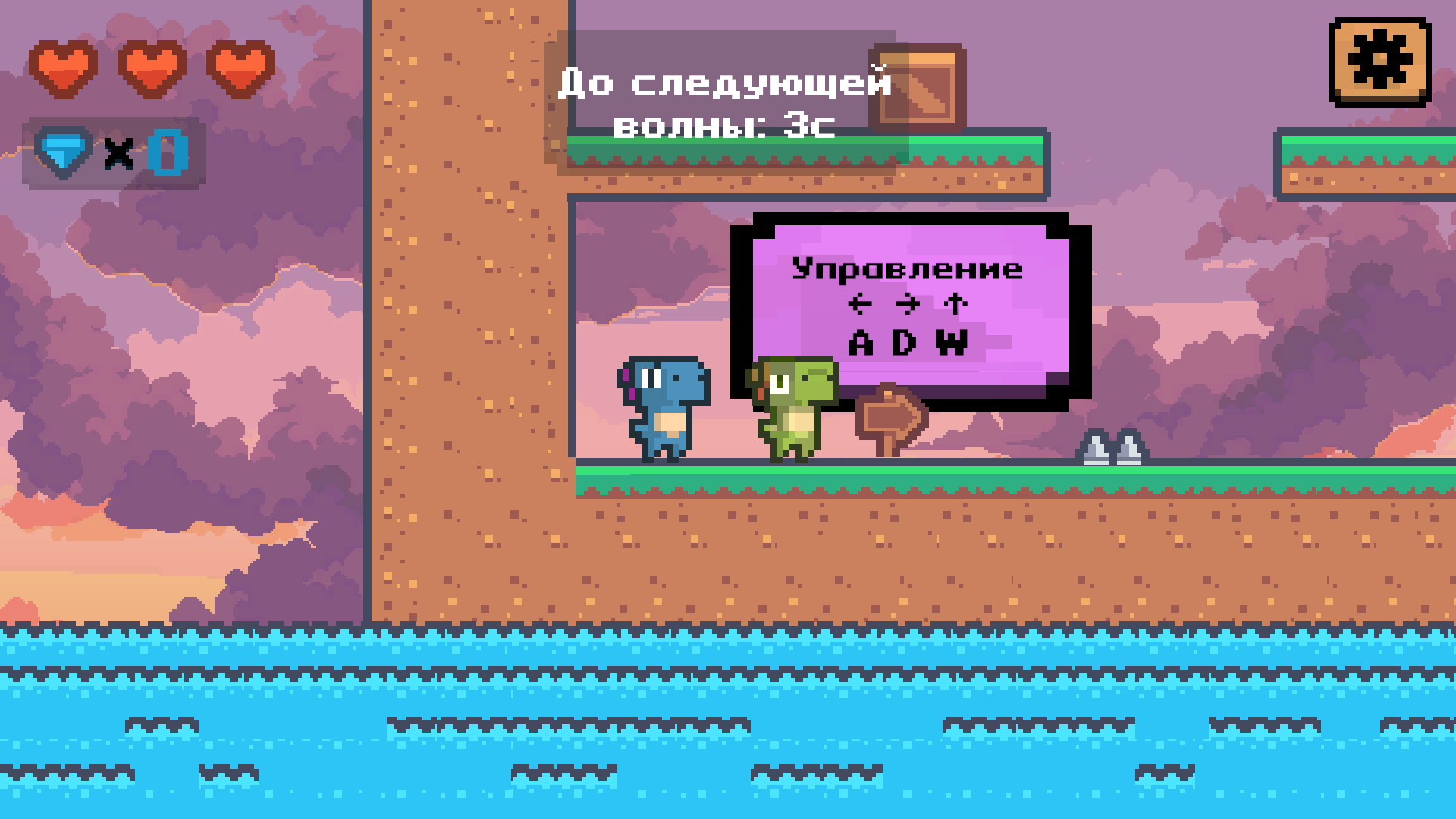


Рисунок 27 – Первый уровень

Также на всех уровнях пользователи будут иметь доступ к меню паузы. В этом меню можно будет регулировать звук, а также с помощью кнопок выйти в главное меню, продолжить игру или возобновить уровень. Скриншот меню паузы представлен на рисунке 28.



Рисунок 28 – Меню паузы

При неудачном прохождение уровня пользователей встречает на экране панель не удачного завершения уровня. С помощью кнопок на этой панели можно выйти в главное меню или возобновить уровень. Скриншот панели не удачного завершения уровня представлен на рисунке 29.

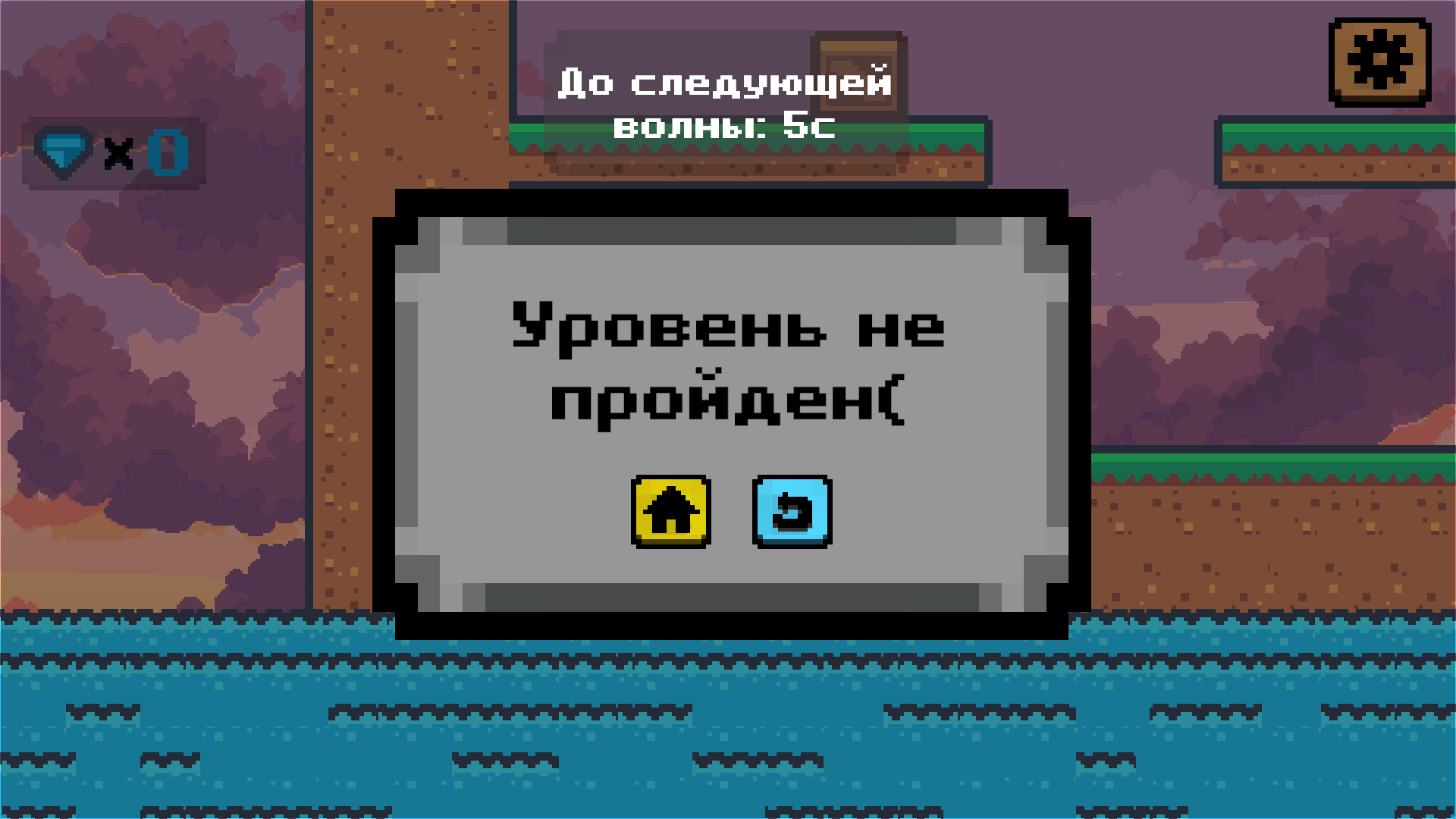


Рисунок 29 – Панель неудачного завершения уровня

При удачном прохождение уровня пользователей встречает на экране панель удачного завершения уровня. С помощью кнопок на этой панели можно перейти на следующий уровень, выйти в главное меню или возобновить уровень. Скриншот панели не удачного завершения уровня представлен на рисунке 30.

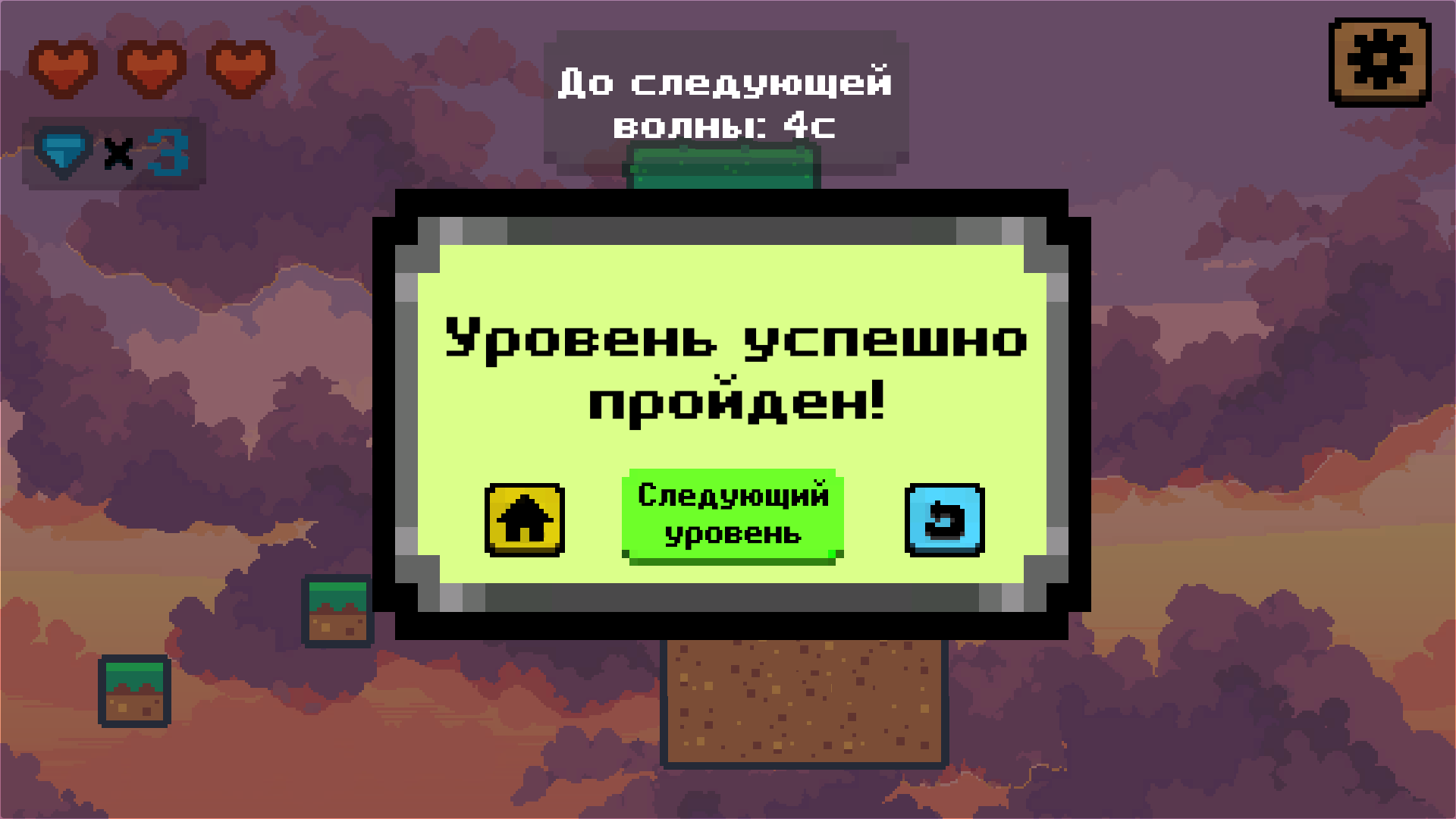


Рисунок 30 – Панель удачного завершения уровня

Также при прохождении уровней пользователи могут собирать кристаллы, которые будут отображать насколько хорошо пройден уровень. Скриншот идеального прохождения уровня представлен на рисунке 31.

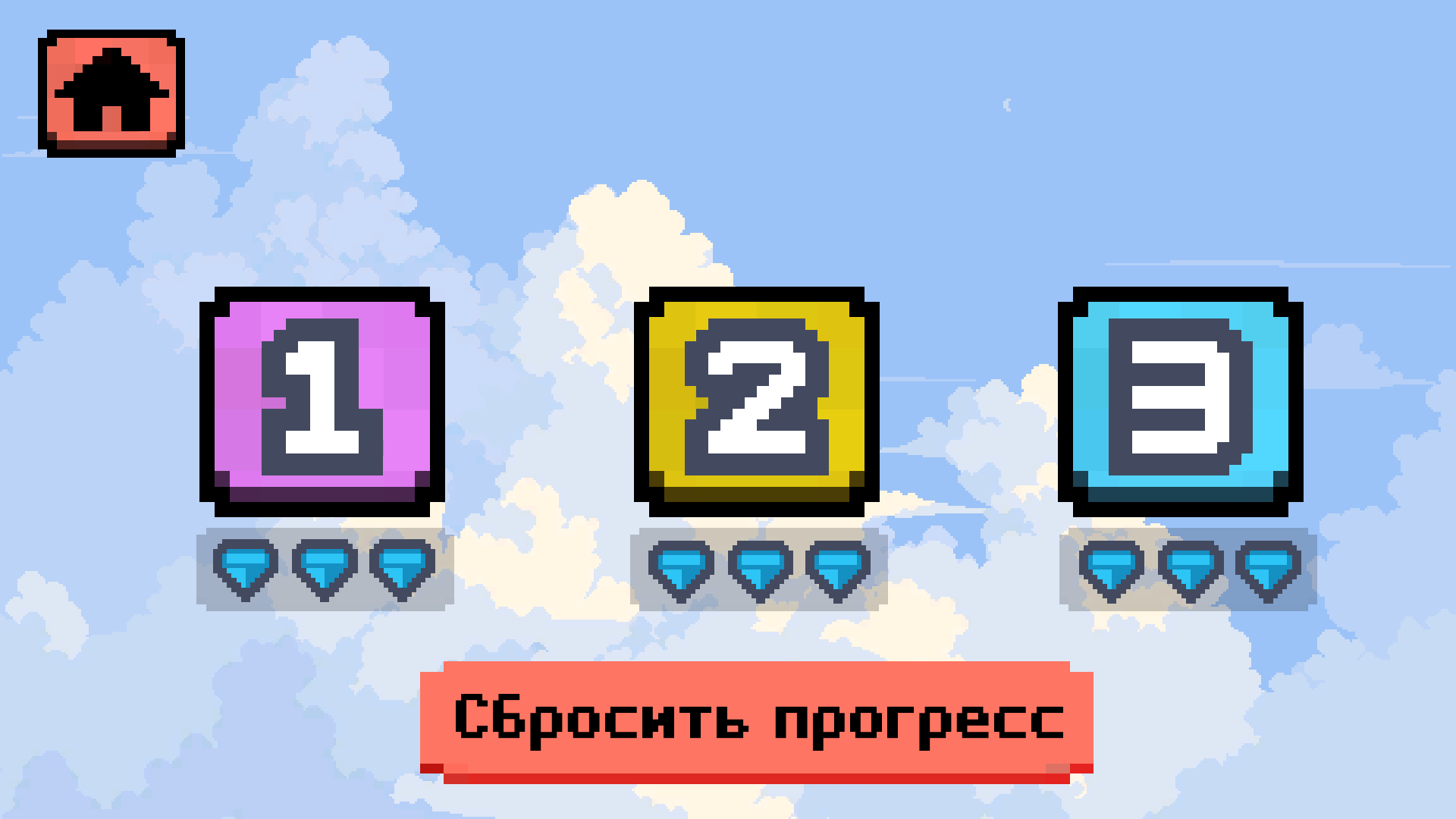


Рисунок 31 – Идеальное прохождение уровней

* 1. **Тестирование системы**

В ходе данного тестирования осуществлялась проверка соответствия игрового приложения заявленным функциональным требованиям. В таблице 1 ниже приведены результаты тестирования. Эти функциональные тесты оценивали работу основных и базовых механик игры, а также ее логики. Успешное выполнение этих функций необходимо для бесперебойной работы игры.

Таблица 1 – Таблица тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название теста** | **Действие** | **Ожидаемый результат** | **Тест пройден?** |
| 1 | Определение первого запуска. | Запустить игровое приложение. | Выводится экран предыстории. | Да |
| 2 | Работоспособность главного меню. | В главном меню нажать на кнопку «Info» | На экране отображается панель информации. | Да |
| 3 | Проверка управления персонажами. | Во время прохождения уровня нажать на клавиши перемещения (A, W, D и стрелочки). | Персонажи перемещаются в игровом мире. | Да |

Окончание таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название теста** | **Действие** | **Ожидаемый результат** | **Тест пройден?** |
| 4 | Проверка нажатия персонажем кнопки. | Во время прохождения уровня одним персонажем встать на кнопку. | Спрайт кнопки меняется, механизм, зависящий от кнопки, начинает передвижение. | Да |
| 5 | Проверка возможности регулировать громкость музыки. | В настройках передвигать ползунок регулировки музыки. | При передвижении ползунка громкость фоновой музыки меняется. | Да |
| 6 | Проверка вызова меню паузы. | При прохождении уровня нажать кнопку «Esc» или на иконку настроек. | На экране отображается панель настроек. | Да |
| 7 | Возрождение персонажей. | Потерять одну жизнь. | Оба персонажа перемещаются в начало уровня. | Да |
| 8 | Получение урона от «шипов». | Во время прохождения уровня, персонажем напрыгнуть на «шипы». | Здоровье уменьшается, персонажи перемещаются на начальные точки. | Да |

Юзабилити тестирование проводилось при участии друзей и одногруппников. Всего в тестировании приняли участие тринадцать человек. В ходе этого тестирования выявлены технические недочеты игры, обнаружены баги, а также получены предложения по улучшению игрового процесса. Эти предложения касались как визуального стиля, так и добавления новых игровых механик. Участники тестирования запускали игровое приложение на различных системах, таких как «Windows 10» и «Windows 11», а также на различных разрешениях и соотношениях сторон монитора. При этом не было обнаружено никаких проблем при использовании приложения. Ниже приведен список изменений, внесенных в игру после анализа результатов тестирования:

1. Внесены изменения в дизайн. Весь изменяемый текст был переведен на русский язык, а также добавлены вспомогательные окна с управлением;
2. Из-за асимметричных коллайдеров персонажей, при повороте рядом с объектами «опасности» персонаж погибал. Для исправления этого бага был внесен метод Flip();
3. Отсутствие возможности регулировки звука игровых объектов вызывало дискомфорт во время игры. Для устранения этой проблемы был добавлен отдельный пользовательский интерфейс (UI) с ползунками для регулировки громкости игровых звуков.
   1. **Вывод по четвертой системе**

В четвертой главе были подробно рассмотрены методы реализации разрабатываемой игры, а также описана реализация ключевых классов и объектов. Разработанная система прошла тестирование на десяти функциональных тестах. Кроме того, было проведено юзабилити тестирование, по результатам которого в игровое приложение были внесены доработки. Особое внимание уделялось интерфейсу игры, который был улучшен для обеспечения удобства и интуитивной понятности для пользователей.

# **ЗАКЛЮЧЕНИЕ**

В рамках данной работы была разработана компьютерная игра в жанре 2D-Платформер с элементами казуальной игры и головоломки на игровом движке Unity.

Основные задачи, которые удалось решить:

1. Проведен обзор аналогов в рамках предметной области. Выявлены сильные и слабые стороны проектов;
2. Проведен анализ функциональных и нефункциональных требований. На основе этого анализа выбран игровой движок Unity и обоснован выбор данной платформы;
3. Спроектирована архитектура игрового приложения, подробно описаны его компоненты, а также разработаны макеты меню и интерфейса;
4. Разработано игровое приложение, которое успешно прошло этапы функционального и юзабилити тестирования.

В ходе работы был приобретен значительный опыт использования игрового движка Unity. Изучена документация для решения необходимых задач. Полученные наработки и опыт создания данного игрового приложения могут стать основой для будущих проектов или улучшения текущего. Приложение разработано с возможностью дальнейших нововведений и поддержки благодаря архитектуре, использованным компонентам Unity и написанным классам.

# **ЛИТЕРАТУРА**

1. Unity User Manual. [Электронный ресурс] URL: https://docs.unity3d.com/ru/530/Manual/Prefabs.html (дата обращения: 01.04.2024 г.).
2. Unity. [Электронный ресурс] // URL: https://unity.com (дата обращения: 05.03.2024)
3. What is a Platform Game. [Электронный ресурс] URL: https://www.lifewire.com/what-is-a-platform-game-812371 (дата обращения: 05.03.2024 г.).
4. Unity User Manual. [Электронный ресурс] URL: https://docs.unity3d.com/ru/530/Manual/Sprites.html (дата обращения: 01.04.2024 г.).
5. Vokigames – Казуальные игры. [Электронный ресурс] URL: https://vokigames.com/kazualnye-igry-chto-eto-kakimi-oni-byvayut-i-kakrazvivaetsya-zhanr/ (дата обращения: 05.03.2024 г.).
6. Iskateligr – Игры головоломки. [Электронный ресурс] URL: https://iskateligr.ru/igry-golovolomki-na-pk/ (дата обращения: 05.03.2024 г.).
7. Официальная страница игры Super Mario Bros. [Электронный ресурс] URL: https://www.nintendo.ru/-/NES/Super-Mario-Bros-803853.html (дата обращения: 05.03.2024 г.).
8. Metacritic – «Super Mario Bros.». [Электронный ресурс] URL: https://www.metacritic.com/game/super-smash-bros (дата обращения: 01.06.2024 г.)
9. Cuphead. [Электронный ресурс] // URL: http://www.cupheadgame.com (дата обращения: 05.03.2024)
10. Metacritic – «Cuphead». [Электронный ресурс] URL: https://www.metacritic.com/game/cuphead (дата обращения: 01.06.2024 г.)
11. Hollow Knight. [Электронный ресурс] // URL: https://www.hollowknight.com (дата обращения: 05.03.2023)
12. Metacritic – «Hollow Knight». [Электронный ресурс] URL: https://www.metacritic.com/game/pc/hollow-knight (дата обращения: 01.06.2024 г.)
13. Unreal Engine. [Электронный ресурс] // URL: https://www.unrealengine.com/en-US (дата обращения: 05.03.2024)
14. Unity 2D. [Электронный ресурс] URL: https://docs.unity3d.com/ru/530/Manual/Unity2D.html (дата обращения: 01.04.2024 г.).
15. Itch.io. [Электронный ресурс] URL: https://itch.io/game-assets/free/tag-sprites/tag-unity (дата обращения: 01.04.2024 г.).

# **ПРИЛОЖЕНИЯ**

## **ПРИЛОЖЕНИЕ А. Спецификация вариантов использования**

Спецификация основных вариантов использования (ВИ) системы приведена в таблицах 1-4.

Таблица 1 – Спецификация ВИ «Запустить игру»

|  |
| --- |
| Прецедент: запустить уровень |
| ID: 1 |
| Краткое описание: запуск уровня позволяет начать проходить игровой уровень |
| Главные актеры: игрок |
| Второстепенные актеры: отсутствуют |
| Предусловия:  1. Успешный запуск игрового приложения. 2. Игрок должен находится в меню выбора уровней |
| Основной поток:  1. Вариант использования начинается, когда игрок нажимает кнопку выбранного уровня. 2. Приложение загружает игровой уровень и начинается игра. |
| Постусловия: отсутствуют |
| Альтернативные потоки: отсутствуют |

Таблица 2 – Спецификация ВИ «Играть»

|  |
| --- |
| Прецедент: играть |
| ID: 2 |
| Краткое описание: управление персонажами и прохождение игровых уровней |
| Главные актеры: игрок |
| Второстепенные актеры: отсутствуют |
| Предусловия:  1. Успешный запуск игрового уровня |
| Основной поток:  1. Вариант использования начинается, когда завершается вариант использования 1. 2. Игрок управляет игровым персонажем 1. |
| Постусловия: отсутствуют |
| Альтернативные потоки: отсутствуют |

Окончание приложения А

Таблица 3 – Спецификация ВИ «Изменять настройки игры»

|  |
| --- |
| Прецедент: изменять настройки игры |
| ID: 3 |
| Краткое описание: игрок может изменять настройки, а именно менять громкость музыки и звуков, выбирать модельку персонажей |
| Главные актеры: игрок |
| Второстепенные актеры: отсутствуют |
| Предусловия: отсутствует |
| Основной поток:  1. Вариант использования начинается, когда игрок заходит в меню настроек в главном меню игры 2. Игрок может менять громкость музыки, звука и модельку пересажай на свое усмотрение. |
| Постусловия: отсутствуют |
| Альтернативные потоки: отсутствуют |

Таблица 4 – Спецификация ВИ «Сбросить прогресс игры»

|  |
| --- |
| Прецедент: сбросить прогресс игры |
| ID: 4 |
| Краткое описание: игрок может обнулить прогресс прохождения уровней. |
| Главные актеры: игрок |
| Второстепенные актеры: отсутствуют |
| Предусловия: отсутствуют |
| Основной поток:  1. Вариант использования начинается, когда игрок переходит в меню выбора уровней. 2. При нажатии на кнопку «Сбросить прогресс» игрок обнулят весь прогресс прохождения. |
| Постусловия: отсутствуют |
| Альтернативные потоки: отсутствуют |