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

РАБОТА ПРОВЕРЕНА

Рецензент,
И.о. начальника отдела распределенных вычислений и встроенных систем ЛСМ

_____ А.И. Рекачинский

“___”_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ

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

_____ Л.Б. Соколинский

“___”_____ 2019 г.

РАЗРАБОТКА МОБИЛЬНОЙ ИГРЫ В ЖАНРЕ "ГОЛОВОЛОМКА" НА ПЛАТФОРМЕ UNITY

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2019.115-083.ВКР

Научный руководитель,
доцент, кандидат техн. наук
_____ Долганина Н.Ю.

Автор работы,
студент группы КЭ-401
_____ Бурулев А.А.

Ученый секретарь
(нормоконтролер)
_____ О.Н. Иванова
“___”_____ 2019 г.

Челябинск-2019

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
09.02.2019

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-401
Бурулеву Антону Александровичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 25.04.2019 № 899)

Разработка мобильной игры в жанре "головоломка" на платформе Unity

2. **Срок сдачи студентом законченной работы:** 05.06.2019.

3. **Исходные данные к работе**

3.1. Nystrom R. Game programming patterns. – Genever Benning, 2014. – 354 p.
3.2. Pereira V. Learning Unity 2D Game Development by Example. – US: Packt Publishing Ltd, 2014. – 266 p.

4. **Перечень подлежащих разработке вопросов**

4.1. Обзор существующих аналогов игрового приложения
4.2. Обзор программных средств реализации игрового приложения
4.3. Проектирование игрового приложения
4.4. Реализация игрового приложения
4.5. Тестирование игрового приложения

5. **Дата выдачи задания:** 08.02.2019.

Научный руководитель

Доцент,

кандидат техн. наук Долганина Н.Ю.

Задание принял к исполнению Бурулев А.А.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

Человечество постоянно развивается во всех аспектах жизни. Одним из наиболее развивающихся направлений является индустрия развлечений. С каждым годом компьютерные игры укрепляют свои позиции на рынке развлечений и могут побороться по прибыли с кино и музыкой. Также, компьютерные игры двигают прогресс в области компьютерного оборудования для достижения более реалистичного игрового опыта. Помимо этого, многие алгоритмы машинного обучения тестируются в играх, так как они являются отличной имитацией реальных задач. Команда OpenAI оттачивала метод обучения с подкреплением на компьютерной игре Dota 2. 15 апреля 2019 года игровые боты, обученные при помощи этого метода, смогли победить чемпионов мира OG [2].

Доход рынка компьютерных игр в 2018 году составил 109.8 миллиардов долларов, из них доход мобильных игр составил 61.3 миллиарда долларов [3]. Мобильные игры отличаются простой игровой механикой и визуальной составляющей. Они отлично помогают скоротать время и требуют лишь наличие смартфона, доступность которого с каждым годом стремительно растет.

За счет своей простоты на рынке мобильных игр появляется много проектов, которые либо привносят новые жанры, либо изменяют уже существующие. Одним из таких жанров является «головоломка», основным отличием мобильной версии данного жанра от ее версий на старших платформах является очень простая идея, лежащая в основе «головоломки» и игровые уровни, на решение которых не требуется большое количество времени.

Цель и задачи

Целью данной работы является создание мобильной игры в жанре «головоломка» для ОС Android на платформе Unity.

Для решения поставленной цели необходимо выполнить следующие задачи:

1) обзор существующих аналогов игрового приложения;
2) обзор программных средств реализации игрового приложения;
3) проектирование игрового приложения;
4) реализация игрового приложения;
5) тестирование игрового приложения.

Структура и объем работы

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 41 страницу, объем списка литературы – 23 источников.

Краткое содержание работы

Введение состоит из трех частей: «Актуальность», «Цели и задачи работы» и «Структура и объем работы».

Первая глава содержит анализ аналогичных проектов и анализ существующих инструментов для реализации игрового приложения.

Во второй главе сформировано техническое предложение, определены функциональные и нефункциональные требования, сформирована диаграмма вариантов использования, а также представлено проектирование архитектуры.

В третьей главе представлена файловая структура приложения, приведены блок-схемы некоторых алгоритмов, описывается реализация сериализации игрового уровня, а также реализация игрового интерфейса.

Четвертая глава посвящена результатам тестирования разработанного игрового приложения.

В заключении приведены итоги разработки игрового приложения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Обзор аналогичных проектов

В ходе выполнения дипломной работы были изучены несколько игр в жанре «головоломка» в магазине приложений Google Play.

Connection

Connection – игра в жанре «головоломка» (рис. 1) [15]. Главной задачей игры является соединение точек линиями разных цветов. Через одну точку может проходить линия лишь одного цвета, помимо этого, есть различные точки-порталы, точки, вмещающие несколько цветов и т.д. В игре присутствует порядка 200 уровней, сложность которых постепенно увеличивается. Игра на данный момент имеет более 1 миллиона загрузок.

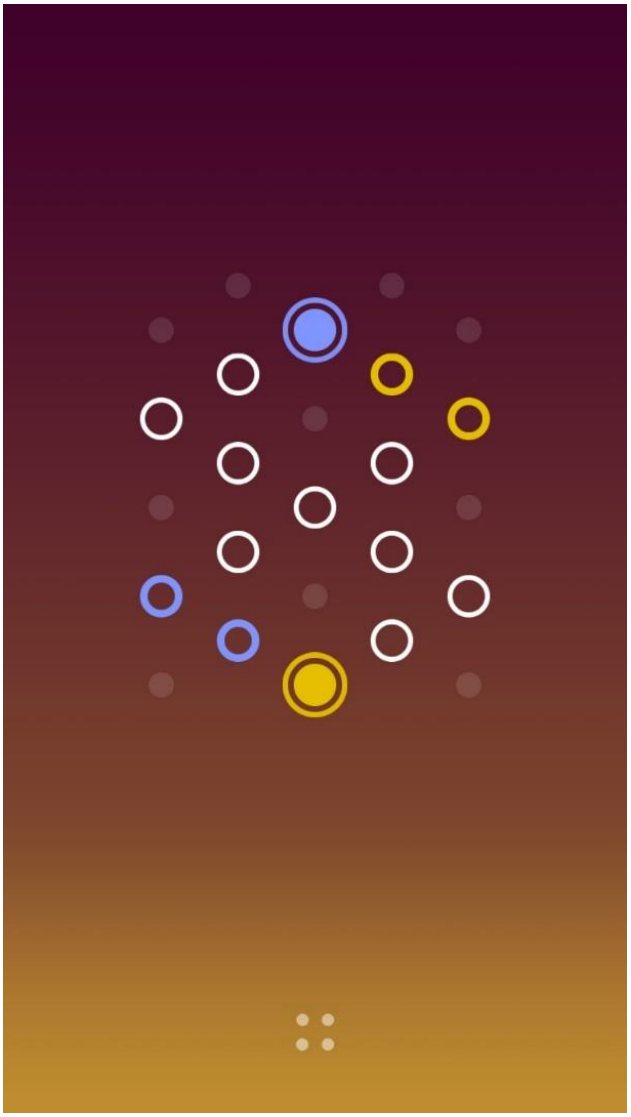

Рис. 1. Игра «Connection»

House Paint

House Paint – игра в жанре «головоломка» (рис. 2) [16]. Основной целью игры является закрашивание всех белых частей здания. Закрашивание возможно лишь в определенных направлениях – вверх, вниз, вправо, влево и происходит до ближайшего препятствия. На данный момент игру загрузили более 10 миллионов раз.

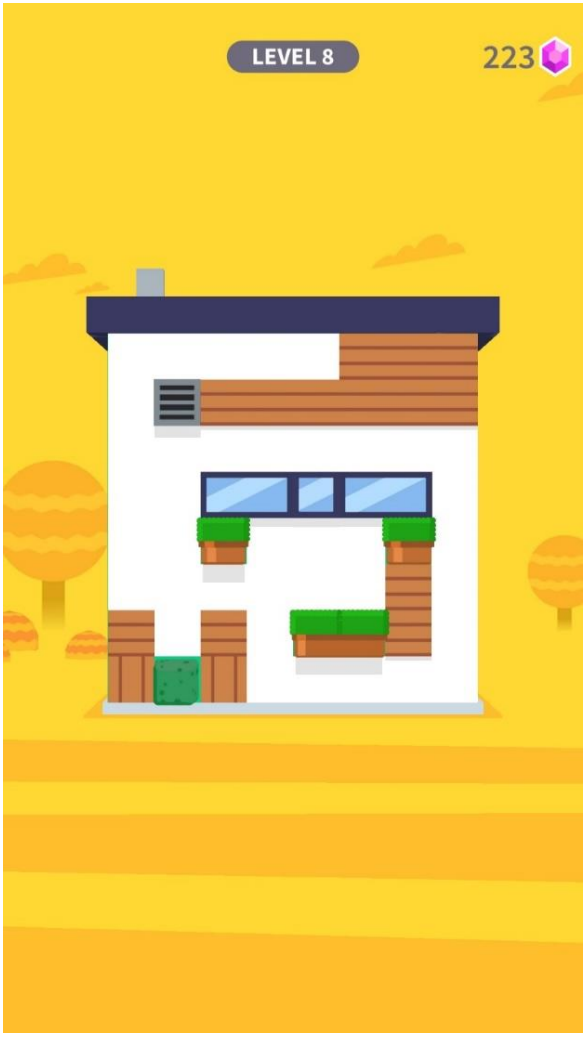

Рис. 2. Игра «House Paint»

Happy Glass

Happy Glass – игра в жанре «головоломка» (рис. 3) [17]. Основной задачей игры является доставка воды через уровень до сосуда. Игрок должен

рисовать препятствия и сооружения для ограничения водяного потока. Основной упор в данной игре делается на взаимодействие с физикой. На данный момент игра имеет более 100 миллионов загрузок.

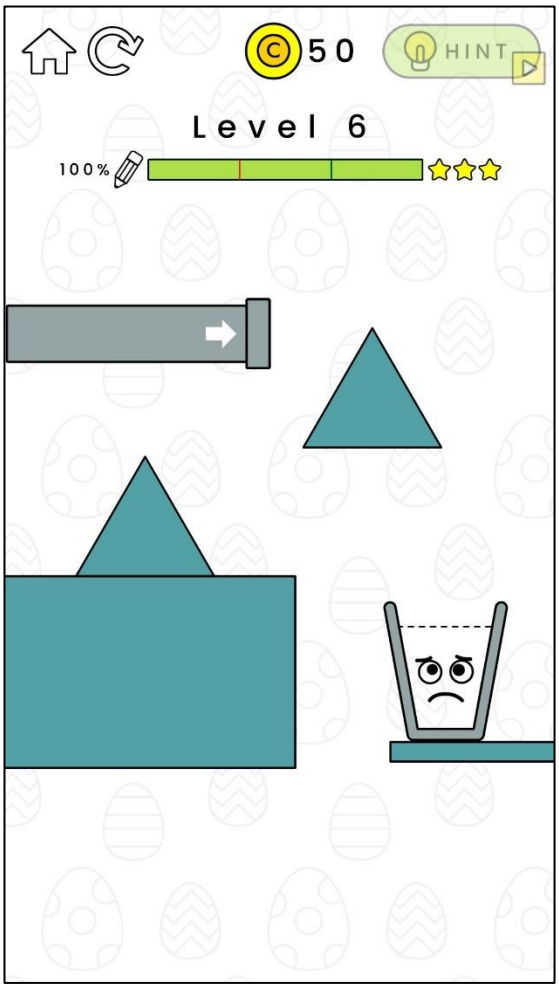

Рис. 3. Игра «Happy Glass»

Основываясь на результатах данного обзора, можно сделать вывод, что игры жанра «головоломка» являются достаточно интересными и пользуются успехом у людей, учитывая количество загрузок и положительные отзывы.

1.2. Обзор существующих решений для реализации проекта

В текущее время существует большое количество различных инструментов для разработки игр от простых библиотек до полноценных конструкторов. Для таких платформ используется понятие «игровой движок» – это

инструмент для разработки компьютерных игр, видеоигр и прочих интерактивных приложений с графикой для разных платформ, которые обрабатываются в реальном времени [22]. Также обеспечивают разработчиков основными технологиями и упрощают разработку. Рассмотрим самые популярные игровые движки, которые активно используются на рынке игровых приложений.

Unity

Unity [20] – межплатформенная среда разработки компьютерных игр. Unity позволяет создавать игровые приложения, работающие под более чем 20 различными операционными системами. Редактор Unity имеет простой Drag&Drop интерфейс, состоящий из множества окон, которые легко настраивать, благодаря чему можно легко производить отладку. Скрипты для данного игрового движка пишутся на языке C# [12, 23].

Плюсы:

1) выгодное решение с финансовой токи зрения;
2) легкое освоение;
3) мультиплатформенность;
4) большое сообщество разработчиков;
5) постоянное совершенствование игрового движка;
6) большой магазин ассетов.
7) документация, затрагивающая все аспекты игрового движка

Минусы:

1) сложность при работе с многокомпонентными системами;
2) затруднение подключения внешних библиотек;
3) малый набор инструментов.

Unreal Engine

Unreal Engine – игровой движок, разрабатываемый компанией Epic Games [21]. Популярный инструмент для создания игр AAA-класса для персональных компьютеров и консолей, однако позволяет разрабатывать игры и для мобильных устройств. Язык написания C++ [11].

Плюсы:

1) самое большое сообщество;
2) большое количество инструментов для разных целей;
3) мультиплатформенность;
4) визуальное программирование Blueprint.

Минусы:

1) сложность в освоении.

Godot Engine

Godot Engine – открытый кроссплатформенный 2D и 3D игровой движок под лицензией MIT, который разрабатывается сообществом Godot Engine Community [14]. Общая архитектура движка построена вокруг концепции дерева из наследуемых «сцен». Каждый элемент сцены (нода), в любой момент сам может стать полноценной сценой. Поэтому при разработке можно легко изменять полностью всю архитектуру проекта, расширять ее элементы в любую сторону и работать с комплексными сценами на уровне простых абстракций.

Плюсы

1) кроссплатформенность;
2) открытый исходный код;
3) поддержка визуального программирования.

Минусы:

1) плохая документация;
2) маленькое сообщество;
3) проект только в начале своего развития;
4) собственный язык скриптов GDScript.

Исходя из произведенного обзора, для разработки был выбран движок Unity [1, 4]. Он популярен, прост в освоении, имеет подробную документацию и отлично подходит для создания 2D игр. Язык C# [12, 23], на котором пишутся скрипты в Unity, обладает большим функционалом и крайне удобен для разработки подобных проектов.

Вывод

В ходе анализа предметной области был произведен обзор существующих аналогов, в ходе которого были выявлены основные принципы формирования жанра и правила построения пользовательского интерфейса.

После этого был произведен обзор существующих решений, выявлены их недостатки и достоинства. На основе этих данных были выбраны инструменты, которые будут использоваться в разработке.

Все материалы, полученные на основе обзора, позволяют определить требования к проектированию и реализации игрового приложения.

2. ПРОЕКТИРОВАНИЕ

2.1. Техническое предложение

Предлагаемое игровое приложение – игра в жанре «головомка», посвященная прохождению опасных уровней, в которых размещены различные ловушки, создающие увлекательные загадки для пользователя.

Цель игры

Основной целью игры является прохождение уровней, постепенно наращивающих свою сложность. Задача каждого уровня состоит в том, чтобы добраться до выхода и собрать максимально возможное количество монет, разбросанных на уровне.

Основная концепция

Игра является мобильным игровым приложением на платформе Android.

Камера в игре расположена в формате «вид сбоку».

Управление в игре реализовывается посредством передвижения пальца по экрану в различные стороны. Игровой персонаж передвигается в направлении движения пальца до ближайшей платформы. Пользовательский ввод не считывается до достижения игровым персонажем платформы для исключения смены направления движения. Направления движения строго заданы – вверх, вниз, влево, вправо.

Игровые предметы

В игре должны быть реализованы следующие объекты, взаимодействующие с игровым персонажем:

1) нестабильная платформа, которая исчезает после попадания на неё игроком (BreakPlatform);

2) ловушка, выпускающая снаряды (Cannon);

3) монетки, разбросанные по всему уровню (Collectable);

4) противник – существо, передвигающееся по заданному маршруту (Enemy);

5) платформа, выпускающая смертельные лазеры спустя несколько секунд после приземления игрового персонажа (LaserTrap);

6) платформа, изменяющая направление движения игрока (MovingChangingPlatform);

7) платформа, передвигающаяся по заранее созданным правилам (MovingPlatform);

8) портал, который моментально транспортирует игрока, к другому связанному порталу без потери направления движения игрока (Portal);

9) платформа, мгновенно убивающая игрока при попадании на неё (Spike);

10) обычная платформа, останавливающая передвижение игрока при попадании на неё (InnerWall);

11) неосязаемая платформа, которая становится обычной платформой на несколько секунд при прохождении через неё игровым персонажем (TimerWall);

12) выходной портал из уровня (LevelEnd).

Игровые уровни

Игровые уровни отличаются своей сложностью и количеством разбросанных ловушек, создающих различные головоломки. Уровни идут друг за другом, то есть для попадания на следующий уровень необходимо пройти предыдущий.

Интерфейс

Интерфейс состоит из главного меню, внутриигрового меню, игровой панели, панели пройденного уровня и панели смерти.

В главном меню находятся кнопки уровней. При нажатии кнопки загружается определенный уровень. Если уровень ещё не доступен для прохождения, он будет скрыт символом замка.

Внутриигровое меню открывается при нажатии кнопки пауза в правом верхнем углу. Внутриигровое меню может быть вызвано только во время

игры. При вызове внутриигрового меню приостанавливается игровой процесс до выхода из него. Во внутриигровом меню расположены кнопки «Продолжить игру» и «Выйти в главное меню».

Игровая панель находится сверху и показывает текущее количество собранных монет, что характеризует прогресс уровня, а также кнопки паузы и рестарта игры.

Панель пройденного уровня открывается при прохождении игроком уровня. Содержит информацию о результате прохождения уровня и кнопки для загрузки следующего уровня и возврата в главное меню.

Панель смерти открывается при смерти персонажа. В панели смерти расположены кнопки «Начать заново» и «Выйти в главное меню».

2.2. Определение требований

В ходе проектирования были определены требования к игровому приложению и возможности, которые должны предоставляться пользователю приложения.

Функциональные требования

1. В приложении должно присутствовать игровое меню для выбора уровня.
2. Приложение должно обеспечивать плавное и удобное взаимодействие с игровым персонажем.
3. Пользователь должен иметь возможность получить информацию об успешности прохождения уровня.
4. Пользователь должен иметь возможность перезапустить уровень.
5. В приложении должен присутствовать обучающий уровень, показывающий способы управления персонажем.

Нефункциональные требования

1. Приложение должно загружать уровни из JSON [18] файла.

2. Приложение должно при закрытии автоматически сохранять необходимые данные в реестр, а также автоматически загружать их при запуске приложения.

3. Приложение должно функционировать на операционной системе Android с версией не ниже 4.1.

2.3. Диаграмма вариантов использования

В ходе анализа требований была разработана UML-диаграмма [13] вариантов использования.

Рассмотрим варианты использования в игровой сцене, представленные на рисунке 4.

Рис. 4. Диаграмма вариантов использования в игровой сцене

Открыть меню паузы: пользователь может приостановить игровой процесс. После чего ему необходимо возобновить его или перейти в главное меню.

Перемещать игрового персонажа: пользователь может контролировать поведение персонажа путём перемещения вправо, влево, вверх, вниз.

Начать уровень заново: пользователь может начать уровень заново.

Получить информацию об успешности прохождения уровня: пользователь может получить информацию об успешности прохождения уровня.

2.4. Проектирование архитектуры

Платформа Unity использует аналог архитектурного паттерна Entity-Component System (ECS), каждый игровой объект в котором представляет собой сущность, состоящую из одного или нескольких компонентов [6]. Добавление новых компонентов к игровому объекту лишь расширяет его функциональность. Данный паттерн отлично подходит для разрабатываемого проекта. Помимо него были использованы порождающие шаблоны проектирования – Singleton, Object Pool [5].

2.4.1 Singleton

Singleton(одиночка) – порождающий шаблон проектирования, гарантирующий, что в приложении будет единственный экземпляр некоторого класса, и предоставляющий глобальную точку доступа к этому экземпляру. Данный шаблон проектирования отлично подходит для доступа к различным классам-менеджерам, контролирующим определенный аспект игрового приложения. Менеджеры как правило сохраняются от сцены к сцене без повторной реинициализации (наподобие глобального объекта).

2.4.2 Object Pool

Object pool (объектный пул) – порождающий шаблон проектирования, отвечающий за инициализацию игровых объектов. Во время запуска игры создаётся пул – набор инициализированных и готовых к использованию объектов. Когда системе требуется объект, он не создаётся, а берётся из пула. Когда объект больше не нужен, он не уничтожается, а возвращается в пул. Если в пуле нет ни одного свободного объекта происходит расширение пула.

Объектный пул применяется для повышения производительности, что особенно важно в мобильных проектах. Особенно заметно повышение производительности, когда объекты часто создаются-уничтожаются, но одновременно существует лишь небольшое их число. Помимо этого, данный шаблон отлично подходит под идею десериализации уровней, так как уже созданные объекты для одного игрового уровня можно переиспользовать на следующих.

Вывод

В ходе проектирования была разработана основная концепция игры, представленная в техническом предложении.

На основе данных анализа предметной области были определены функциональные и нефункциональные требования к разрабатываемому игровому приложению. На основе функциональных требований построена диаграмма вариантов использования.

В проектировании архитектуры были определены используемые шаблоны проектирования.

3. РЕАЛИЗАЦИЯ

3.1. Файловая структура приложения

Проект разработанного приложения состоит из списка каталогов, содержащих спрайты; уровни, заданные в формате JSON; готовые компоненты, которые являются игровыми объектами с заданными свойствами для повторного использования; игровые скрипты; файлы анимации; элементы Tile Pallete [9]; игровые сцены. Файловая структура приложения представлена на рисунке 5.

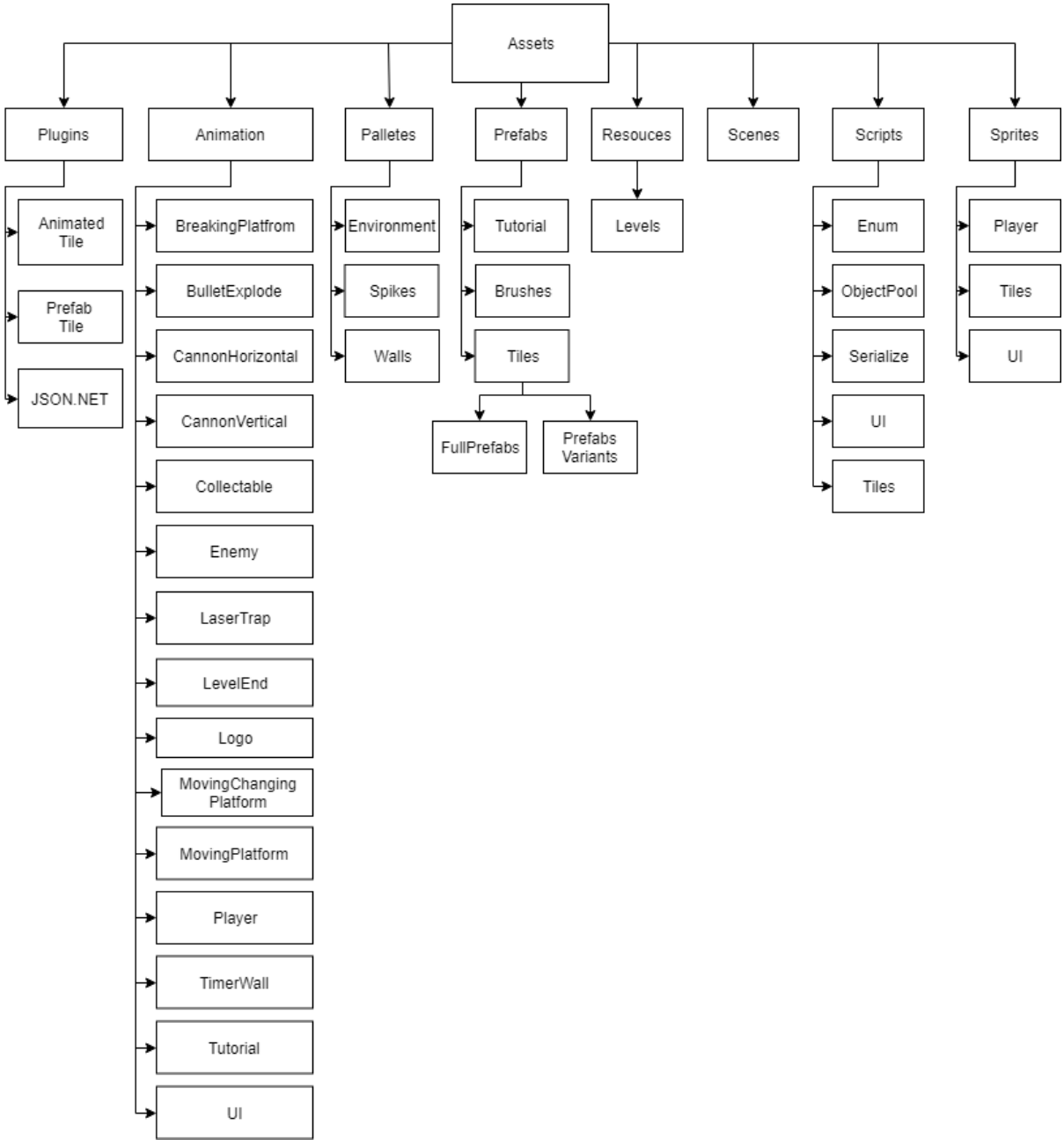

Рис. 5. Файловая структура приложения

3.2. Диаграмма компонентов

Диаграмма компонентов отображена на рисунке 6.

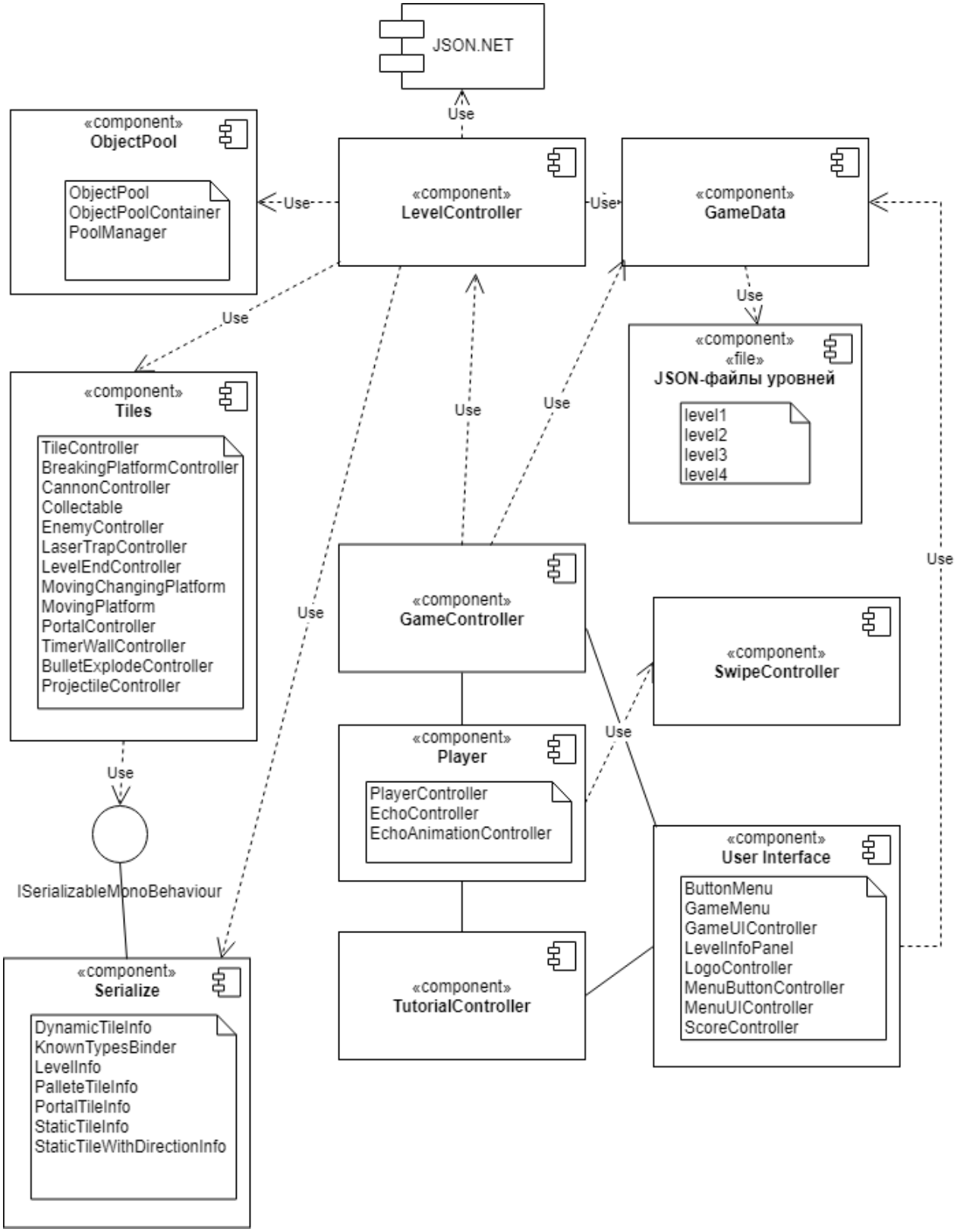

Рис. 6. Диаграмма компонентов игрового приложения

1) Компонент GameController осуществляет контролирование всех игровых аспектов приложения. Реализует шаблон проектирования «одиночка».

2) Компонент LevelController осуществляет контролирование игровой сцены, правильную загрузку уровней и т.д. Реализует шаблон проектирования «одиночка».

3) Компонент Player представляет реализацию игрового персонажа. Включает в себя:

А) PlayerController – основная логика персонажа;

Б) EchoController – инициализация копий персонажа во время его движения;

В) EchoAnimationController – удаление копий персонажа после окончания анимации.

4. Компонент UIController отвечает за формирование интерфейса игрового приложения. Включает:

А) ButtonMenu – анимация кнопок интерфейса;

Б) GameMenu – игровые меню;

В) GameUIController – служит для управления пользовательским интерфейсом на игровой сцене;

Г) LevelInfoPanel – панель, появляющаяся после прохождения игрового уровня;

Д) LogoController – служит для управления анимацией логотипа;

Е) MenuButtonController – кнопки в главном меню, показывающие текущий прогресс уровня;

Ж) MenuUIController – служит для управления пользовательским интерфейсом в главном меню;

З) ScoreController – панель, показывающая текущий прогресс игрока на уровне.

5. Компонент TutorialController отвечает за обучающий уровень, в котором действуют особые правила перемещения и пользовательского интерфейса.

6. Компонент SwipeController отвечает за обработку пользовательского ввода.

7. Компонент Serialize отвечает за правила сериализации элементов. Включает в себя:

А) DynamicTileInfo – информация о динамическом игровом объекте;

Б) KnownTypesBinder – информация о классах, необходимая плагину JSON.NET для правильной сериализации классов;

В) LevelInfo – вся информация об игровых объектах;

Г) PalleteTileInfo – информация о игровом объекте, расположенном на Tilemap;

Д) PortalTileInfo – информация о портале;

Е) StaticTileInfo – информация о статическом игровом объекте;

Ж) StaticTileInfoWithSomeDirectionInfo – информация о статическом игровом объекте, имеющим направление;

З) ISerializableMonoBehaviour – интерфейс, обеспечивающий что игровой объект будет сериализован.

8. Компонент Tiles включает в себя реализацию различных игровых объектов, с которыми может взаимодействовать игрок. Большинство из них были описаны в техническом предложении, остальные:

А) TileController – базовый игровой объект, который может быть сериализован;

Б) BulletExplodeController – взрыв после попадания пули, выпущенной ловушкой;

В) ProjectileController – пуля, выпущенная из ловушки.

9. Компонент ObjectPool реализует шаблон проектирования объектный пулинг. Включает в себя:

А) ObjectPoolContainer – контейнер для объектов;

Б) ObjectPool – пул объектов;

В) PoolManager – управляет всеми созданными пулами объектов. Реализует шаблон проектирования одиночка.

10. Компонент GameData отвечает за все данные, хранящиеся в игре. Считывает и записывает данные в PlayerPrefs. Загружает json представления всех уровней.

11. JSON-файлы уровней отвечают за хранение десериализованных уровней игрового приложения;

12. JSON.NET – плагин для десериализации объектов. [19]

3.3. Игровой персонаж

Игрок представляет собой игровой объект, который предоставляется для управления. Он состоит из 2D изображения - спрайта, коллайдера (Collider2D), компонента Rigidbody2D, наличие которого подразумевает, что этот объект подчиняется законам физики. Помимо этого, данный игровой объект содержит в себе поведенческий скрипт PlayerController.

Скрипт PlayerController в первую очередь необходим для обрабатывания коллизий игрока с различными платформами и осуществления перемещения. Для получения направления перемещения было создано событие OnSwipe, которое вызывает скрипт SwipeController при обработке пользовательского ввода путём встроенных методов Unity. Дальнейшие действия алгоритма показаны на блок-схеме на рисунке 7.

Для показа анимаций игрового персонажа используется компонент Animator, который позволяет программно переключаться между анимациями при выполнении различных условий. Управление условиями анимаций также осуществляется в скрипте PlayerController. Всего существуют следующие типы анимации персонажа:

1) Idle – анимация по умолчанию, выполняется, когда с игроком ничего не происходит;

2) Fly – анимация полета;

3) Land – анимация приземления;

4) Death – анимация смерти;

5) LevelPass – анимация прохождения уровня.

Для добавления ощущения быстрого передвижения персонажа был добавлен скрипт EchoController, который во время движения игрока создаёт копии персонажа с периодичностью 0.005 секунд.

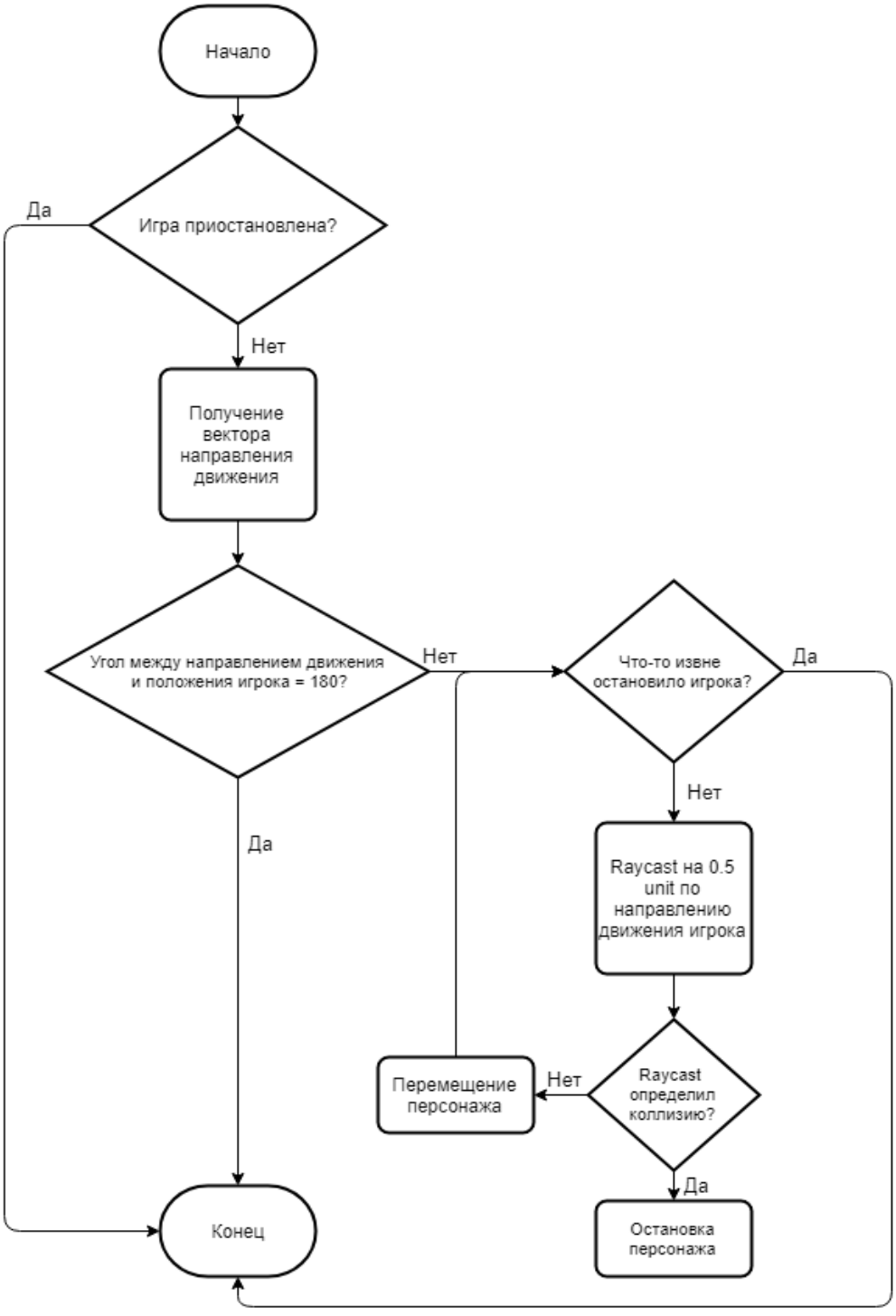

Рис. 7. Блок-схема алгоритма передвижения игрока

3.4. Сериализация и десериализация игрового уровня

Игровой уровень представлен в виде объектов, подверженных различному поведению. Иерархия таких объектов на игровой сцене представлена на рисунке 8.

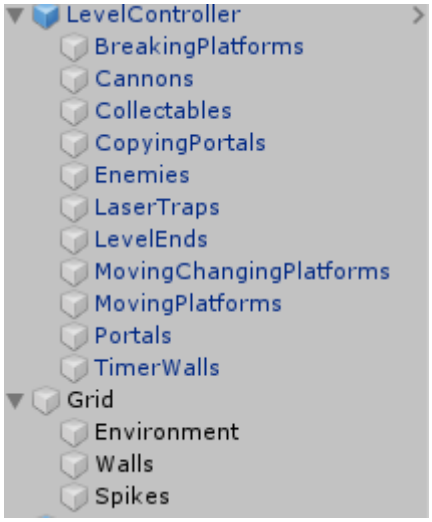

Рис. 8. Иерархия игрового уровня

Для быстрого доступа по категориям создан пустой игровой объект LevelController, потомками которого являются пустые игровые объекты, отражающие определенную категорию.

Помимо этого, на сцене существует компонент Grid, который хранит несколько компонентов Tilemap [8], содержащих самые используемые статичные игровые объекты, добавляемые на сцену инструментом TilePallete:

1) Walls – все обычные платформы уровня, содержит TillemapCollider для обработки коллизий;

2) Spikes – все платформы, моментально убивающие игрока, содержит TilemapCollider для обработки коллизий;

3) Environment – визуальное окружение, не влияющее на игровой процесс.

Для удобного хранения уровней и простого расширения их количества был сделан скрипт LevelController, привязанный к объекту LevelController, который управляет сериализацией и десериализацией необходимых

игровых объектов на сцене, что позволяет без смены сцены загружать новые игровые уровни.

Все сериализуемые игровые объекты наследуются от абстрактного класса TileController, который имеет два абстрактных метода Serialize и Deserialize. Данные методы используются для сериализации/десериализации игровых объектов (рис. 9)

```
public abstract class TileController: MonoBehaviour,
                        ISerializableMonoBehaviour{
    public abstract StaticTileInfo Serialize();
    public abstract bool Deserialize(StaticTileInfo tileInfo);
}
```

Рис. 9. TileController

Были выделены основные категории свойств, которые имеют объекты.

1. StaticTile (BreakPlatform, Collectable, LevelEnd, TimerWall) (рис. 10).

```
public class StaticTileInfo{
        public TileType TileType { get; set; }
        public Vector3 Position { get; set; }
        public Vector3 Rotation { get; set; }
}
```

Рис. 10. StaticTile

2. StaticTileWithDirection (Cannon, MovingChangingPlatform) (рис. 11).

```
public class StaticTileWithDirectionInfo: StaticTileInfo {
        public Vector3 Direction { get; set; }
}
```

Рис. 11. StaticTileWithDirection

3. PortalTile (Portal) (рис. 12).

```
public class PortalTileInfo: StaticTileInfo{
        public Vector3 OtherPortalPosition { get; set; }
}
```

Рис. 12. PortalTileInfo

4. DynamicTile (MovingPlatform, Enemy) (рис. 13).

```
public class DynamicTileInfo:StaticTileInfo{
        public Vector3 FromDirection { get; set; }
        public Vector3 ToDirection { get; set; }
}
```

Рис. 13. DynamicTile

5. PalleteTile (Spike, InnerWall, Environment) (рис. 14).

```
public class PaletteTileInfo{
        public int X;
        public int Y;
        public int IndexOfArray;
}
```

Рис. 14. PalleteTile

Вся информация об уровне заключается в классе LevelInfo, который подвергается сериализации при помощи плагина JSON.NET (рис. 15).

```
public class LevelInfo {
        public Dictionary<TileType,List<StaticTileInfo>> TileGameObjects
{ get; set; }

        public List<PaletteTileInfo> PaletteTileInfos { get; set; }
        public List<PaletteTileInfo> TileWalls { get; set; }
        public List<PaletteTileInfo> TileSpikes { get; set; }
        public Vector3 PlayerPos { get; set; }
}
```

Рис. 15. LevelInfo

Пример сериализованного уровня представлен на рисунке 16.


```
{
  "TileGameObjects": {
    "TimerWall": [
      {
        "TileType": 12,
        "Position": {
          "x": 1.5,
          "y": -5.5,
          "z": 0.0
        },
        "Rotation": {
          "x": 0.0,
          "y": 0.0,
          "z": 0.0
        }
      }
    ]
   },
   "PaletteTileInfos": [
     {
"X": -5,
      "Y": -9,
      "IndexOfArray": 20
     },
     {
      "X": -5,
      "Y": -8,
      "IndexOfArray": 18
     }
    ],
    "TileWalls": [
     {
      "X": -4,
      "Y": -6,
      "IndexOfArray": 63
     },
     {
      "X": -4,
      "Y": -5,
      "IndexOfArray": 63
     },
    ],
    "PlayerPos": {
    "x": -2.5,
    "y": -5.5,
    "z": 0.0
    }
}
```


Рис. 16. Пример сериализованного уровня

3.5. Интерфейс приложения

Пользовательский интерфейс реализован при помощи системы Unity UI, которая позволяет разрабатывать пользовательские интерфейсы прямо в

редакторе. Все объекты UI являются игровыми объектами, но располагаются в плоскости прямоугольника камеры. Основным объектом является Canvas, который является контейнером всех UI элементов и отвечает за их отрисовку. Canvas имеет скрипт Canvas Scaler [7], позволяющий растягивать элементы интерфейса в зависимости от размера экрана. Объекты пользовательского интерфейса имеют компонент RectTransform, которая отвечает за положение и масштабирование объекта в прямоугольной плоскости камеры.

Основные виды элементов Unity UI [10], которые были использованы в реализации пользовательского интерфейса:

1) UI.Text – используется для отображения текста и различной информации;
2) UI.Button – кнопка, при нажатии на которую выполняются определенные действия;
3) UI.Panel – используется для всплывающих окон. На объектах Panel расположены другие элементы пользовательского интерфейса;
4) UI.Slider – используется для показа прогресса загрузки уровня;
5) UI.Image – используется для показа прогресса выполнения уровня.

Пользовательский интерфейс состоит из следующих объектов:

1) логотип;
2) меню выбора уровня;
3) экран загрузки;
4) меню паузы;
5) меню смерти;
6) кнопки вызова меню и перезагрузки уровня;
7) прогресс уровня;
8) панель начала уровня;
9) панель прохождения уровня;
10) подсказки управления.

Логотип (рис. 17) показывает приветственный экран и используется для загрузки ресурсов. При нажатии на любое место экрана появляется меню выбора уровня.

Рис. 17. Логотип

Меню выбора уровня включает в себя 4 кнопки с информацией о лучшем прохождении уровня. В зависимости от прогресса игрока меню будет динамически изменяться, открывая новые уровни для прохождения (рис. 18). Недоступные уровни будут заблокированы. При нажатии на кнопку доступного уровня начнёт загружаться сцена игры и запустится процесс десериализации игрового уровня

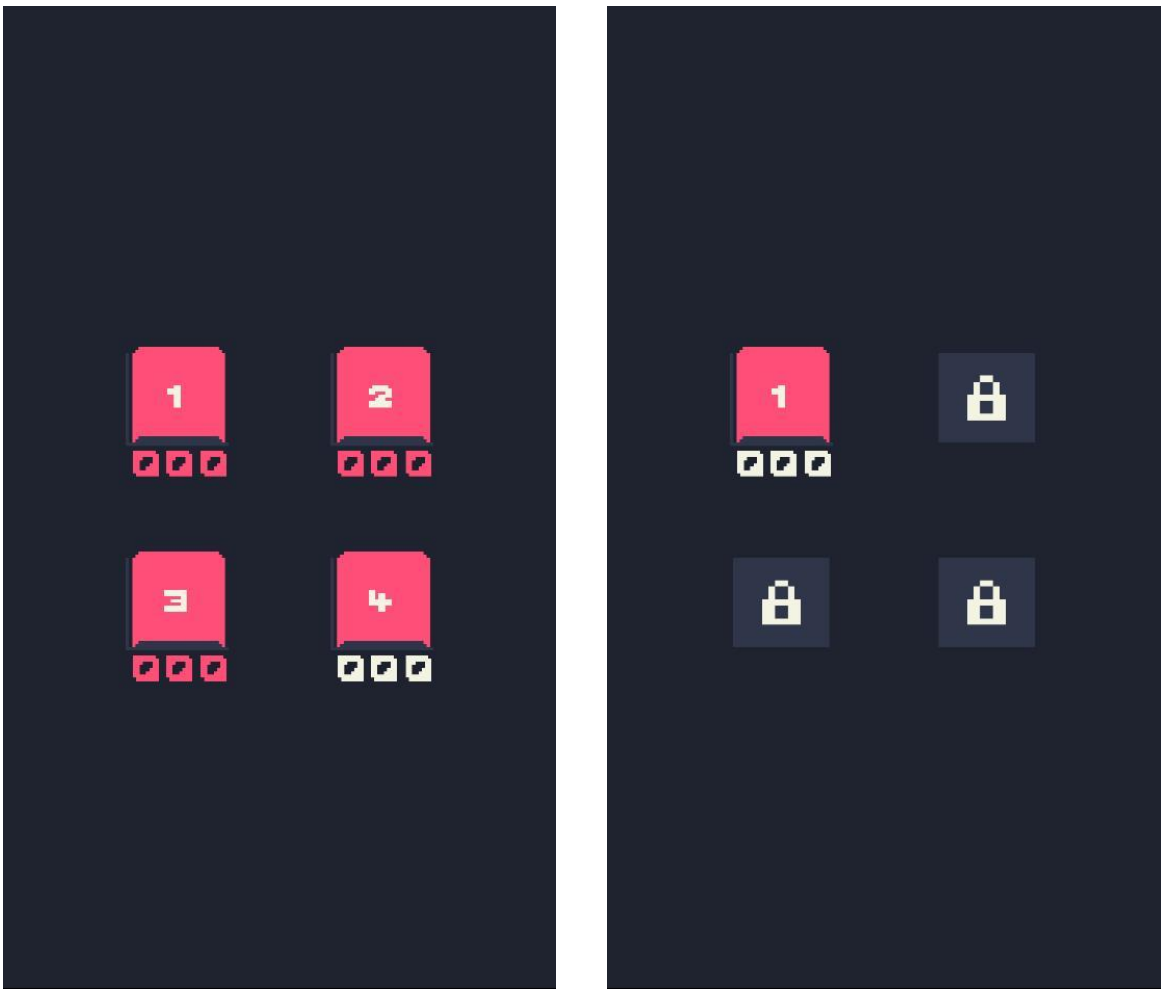

Рис. 18. Меню выбора уровня. (Слева – все уровни доступны, справа – доступен только первый уровень)

Экран загрузки представлен UI.Slider, показывающий прогресс загрузки игровой сцены (рис. 19). После полноценной загрузки происходит переходит на другую сцену.

Рис. 19. Экран загрузки

Кнопки вызова меню и перезагрузки уровня (рис. 20) являются элементами UI.Button.

Рис. 20. Кнопки вызова меню и перезагрузки уровня

При нажатии на кнопку перезагрузки уровня происходит повторная десериализация уровня и уровень начинается сначала. При нажатии на кнопку вызова меню открывается меню паузы (рис 21). Игра приостанавливается до закрытия меню паузы. При нажатии кнопки «Продолжить» закрывается меню паузы. При нажатии кнопки «Меню» происходит загрузка сцена с выбором уровня.

Рис. 21. Меню паузы

При смерти игрока появляется панель смерти (рис 22), имеющая кнопки перезагрузки уровня и возвращения в главное меню. Пока активна панель смерти игра приостановлена.

Рис. 22. Панель смерти

Прогресс уровня (рис. 23) представляет собой три UI.Image, показывающие сколько монет собрал игрок. Собранные монеты отображаются розовым, несобранные – белым. Обновление спрайтов осуществляется через ScoreController.

Рис. 23. Прогресс уровня

Панель начала уровня (рис. 24) представляет UI.Panel с UI.Text, показывающим номер уровня. Игра приостановлена до исчезновения панели.

Рис. 24. Панель начала уровня

При прохождении уровня игроком появляется панель прохождения уровня (Рис. 25). На ней показан прогресс прохождения уровня и кнопки «Меню» и «Продолжить». При нажатии кнопки «Продолжить» происходит запись текущего результата в реестр, если он является лучшим, и десериализация нового игрового уровня.

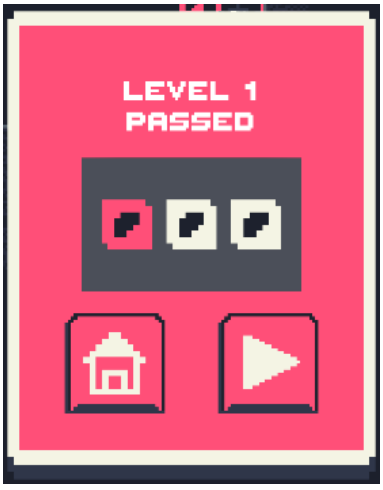

Рис. 25. Панель прохождения уровня

Подсказки управления (рис. 26) представляют анимированные UI.Image, показывающие как правильно управлять игроком. Каждая подсказка показывает направление движения пальца для изменения направления движения игрока. Все остальные действия пользователя, отличающиеся от данного направления, будут блокироваться до выполнения нужного действия.

Рис. 26. Подсказки управления

Вывод

В ходе реализации была построена и описана диаграмма компонентов, отражающая основные компоненты игрового приложения. Также была продемонстрирована реализация игрового персонажа и процесса сериализации и десериализации игрового уровня. После этого была описана реализация интерфейса игрового приложения.

4. ТЕСТИРОВАНИЕ

4.1. Функциональное тестирование

Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определенных условиях решать задачи, нужные пользователям. Рассматривает указанное заранее поведение и основывается на анализе спецификаций функциональности компонента или системы в целом.

По результатам функционального тестирования была составлена таблица с итогами. (табл.)

Табл. Функциональное тестирование

№	Название теста	Действия тестировщика	Ожидаемый результат	Прохождение теста
1	Неправильные действия при обучении	Совершать действия, противоположные необходимым на текущей шаге обучения	Персонаж не переместится в нужном направлении	Да
2	Правильные действия при обучении	Передвигать палец по экрану по направлению, показанному в обучении	Игрок перемещается в нужном направлении	Да
3	Отображение обучения	Зайти в игру с нового устройства	При первом заходе в игру отображается обучение	Да
4	Прохождение обучения	Пройти обучение	Появление главного меню	Да

Продолжение табл.

№	Название теста	Действия тестировщика	Ожидаемый результат	Прохождение теста
6	Сохранение пройденного обучения	Зайти в игру повторно с устройства, где пройдено обучение	При повторном запуске игры показывается главное меню	Да
7	Запуск уровня	Запустить первый уровень	Загрузка сцены с игрой и десериализация уровня	Да
8	Остановка игры	Нажать на кнопку «Пауза»	Появление меню паузы. Остановка игры.	Да
8	Продолжение игры	Нажать на кнопку «Продолжить» в меню паузы	Исчезновение меню паузы. Возобновление игры	Да
9	Возврат в главное меню	Нажать на кнопку «Домой» в меню паузы	Исчезновение меню паузы. Загрузка сцены меню.	Да
10	Перемещение персонажа	Переместить персонажа при помощи перемещения пальца по экрану	Персонаж движется в нужном направлении до ближайшей платформы.	Да

Продолжение табл.

№	Название теста	Действия тестировщика	Ожидаемый результат	Прохождение теста
11	Перезапуск уровня	Нажать на кнопку «Начать заново»	Сброс всех анимаций. Перемещение игрока в начальную точку уровня.	Да
12	Смерть персонажа	Попасть в ловушку, убивающую персонажа	Проигрывается анимация смерти персонажа. Появление меню смерти	Да
13	Прохождение уровня	Пройти уровень.	Проигрывается анимация прохождения уровня. Появляется панель прохождения уровня	Да
14	Переход на следующий уровень	Нажатие кнопки «Продолжить» на панели прохождения уровня	Очистка сцены. Десериализация следующего уровня. Сохранение результата предыдущего уровня	Да

Окончание табл.

№	Название теста	Действия тестировщика	Ожидаемый результат	Прохождение теста
15	Сбор монет	Собрать монету на уровне	Изменение количества монет на панели прогресса уровня	Да
16	Сохранение пройденных уровней	Зайти в игру после нескольких пройденных уровней	В главном меню отобразится текущий прогресс прохождения игры.	Да

4.2. Юзабилити тестирование

Юзабилити-тестирование – исследование, выполняемое с целью определения, удобен ли продукт в использовании. Выполняется с помощью привлечения пользователей приложения в качестве тестировщиков, получения от них информации об удобстве приложения и последующем подведении итогов от их выводов.

Участникам тестирования необходимо было выполнить следующие задачи:

1) пройти обучение;
2) запустить уровень;
3) приостановить игру;
4) начать уровень заново;
5) определить прогресс на уровне;
6) вернуться в главное меню;
7) запустить следующий уровень.

Все задачи были выполнены тестировщиками успешно без затруднений. Пользователи отметили хорошие анимации, лёгкость управления и сложность уровней. Юзабилити тестирование пройдено успешно.

Вывод

Для реализованного игрового приложения было проведено функциональное и юзабилити-тестирование. Все тесты были выполнены успешно, существенных проблем обнаружено не было.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была создано мобильная игра в жанре «головоломка» для ОС Android на платформе Unity.

Для решения поставленной цели выполнены следующие задачи:

1) проведен обзор существующих аналогов игрового приложения;
2) выполнен обзор программных средств реализации игрового приложения;
3) спроектировано игровое приложение;
4) реализовано игровое приложение;
5) протестировано игровое приложение.

СПИСОК ЛИТЕРАТУРЫ

1. Calabrese D. Unity 2D Game Development. – USA: Packt Publishing Ltd, 2014. – 126 p.

2. How to train your OpenAI Five. [Электронный ресурс] URL: https://www.openai.com/blog/how-to-train-your-openai-five/ (дата обращения: 02.04.2019).

3. Market Brief — 2018 Digital Games & Interactive Entertainment Industry Year In Review. [Электронный ресурс] URL: https://www.superdataresearch.com/market-data/market-brief-year-in-review/ (дата обращения: 02.04.2019).

4. Menard M., Wagstaff B. Game development with Unity. – Nelson Education, 2015. – 433 p.

5. Nystrom R. Game programming patterns. – Genever Benning, 2014. – 354 p.

6. Pereira V. Learning Unity 2D Game Development by Example. – US: Packt Publishing Ltd, 2014. – 266 p.

7. Unity – Руководство: Создание интерфейса (UI) под разные разрешения экрана. [Электронный ресурс] URL: https://docs.unity3d.com/ru/current/Manual/HOWTO-UIMultiResolution.html (дата обращения: 02.04.2019).

8. Unity – Manual: Tilemap. [Электронный ресурс] URL: https://docs.unity3d.com/Manual/class-Tilemap.html (дата обращения: 02.04.2019).

9. Unity – Manual: Tile Palette. [Электронный ресурс] URL: https://docs.unity3d.com/Manual/Tilemap-Palette.html (дата обращения: 02.04.2019).

10. User Interface (UI). [Электронный ресурс] URL: https://unity3d.com/ru/learn/tutorials/s/user-interface-ui (дата обращения: 02.04.2019).

11. Zheng L., Dong Y., Yang F. C++ Programming. – Walter de Gruyter GmbH & Co KG, 2019. – 486 p.

12. Албахари Д., Албахари Б. C# 6.0. Справочник. Полное описание языка. – М.: Вильямс, 2017. – 1040 с.

13. Буч Г., Якобсон И., Рамбо Д. Язык UML. Руководство пользователя. – Litres, 2017. – 496 с.

14. Официальный сайт Godot Engine. [Электронный ресурс] URL: https://godotengine.org/ (дата обращения: 02.04.2019).

15. Официальный сайт Google Play. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.infinitygames.connection (дата обращения: 02.04.2019).

16. Официальный сайт Google Play. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.housepaint.game (дата обращения: 02.04.2019).

17. Официальный сайт Google Play. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.game5mobile.lineandwater (дата обращения: 02.04.2019).

18. Официальный сайт JSON. [Электронный ресурс] URL: https://www.json.org/json-ru.html (дата обращения: 02.04.2019).

19. Официальный сайт JSON.NET. [Электронный ресурс] URL: https://www.newtonsoft.com/json (дата обращения: 02.04.2019).

20. Официальный сайт Unity. [Электронный ресурс] URL: https://unity3d.com (дата обращения: 02.04.2019).

21. Официальный сайт Unreal Engine. [Электронный ресурс] URL: https://www.unrealengine.com (дата обращения: 02.04.2019).

22. Пасько Д.Н. Современные игровые движки. // Инновационная наука, 2016 – № 2-3 (14). – С. 127–130.

23. Троэлсэн Э., Джепикс Ф. Язык программирования C# 6.0 и платформа .NET 4.6. – М.: Вильямс, 2016. – 1440 с.