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 «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой «ЭВМ»
_____Д. В. Топольский
«__»_____2022 г.

РАЗРАБОТКА АРКАДНОЙ 2D-ИГРЫ НА UNITY

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2022.686 ПЗ ВКР

Руководитель работы,
к.п.н., доцент каф. «ЭВМ»
_____Ю. Г. Плаксина
«___»_____2022 г.

Автор работы,
Студент группы КЭ-405
_____М. Х. Хамед
«___»_____2022 г.

Нормоконтролер,
к.п.н., доцент каф. «ЭВМ»
_____М. А. Алтухова
«___»_____2022 г.

Челябинск-2022

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 «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой «ЭВМ»
_____Д. В. Топольский
«___»_____2022 г.

ЗАДАНИЕ
на выпускную квалификационную работу бакалавра
студенту группы КЭ-405
Хамеду Моханнаду Хазему
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Разработка аркадной 2D-игры на Unity» утверждена приказом по университету от 12/12/2021 г. № 308/141

2. **Срок сдачи студентом законченной работы**: 1 июня 2022 г.

3. **Исходные данные к работе:**

К общим требованиям для персонального компьютера можно отнести минимальные системные требования:

- операционная система (ОС): Windows 7 или выше/Linux/Android 4.4 (KitKat);
- процессор: Intel Core i3 540 (1st gen.);
- оперативная память: 4 GB;
- видеокарта: видеокарта с DirectX 11.0 с 1 GB VRAM (NVidia GeForce 460);
- DirectX: версии 11;

− место на диске: 500 MB;

− звуковая карта: звуковая карта совместимая с DirectX.

Данные системные требования являются минимальными для продукции, которая создается на движке Unity.

Пользователь может:

– начать игру;

– управлять персонажам прыгая (кнопкой пробел);

– настроить игру, изменив громкость;

– выйти из игры.

Пользователь имеет возможность запустить файл игры и попасть в главное меню.

Пользователь может управлять персонажем во время игры, дважды прыгая с помощью клавиши пробела, в первый раз пользователь может удерживать клавишу пробела, чтобы персонаж взлетел так высоко, как необходимо, во второй раз может быть только одно нажатие, чтобы совершить двойной прыжок.

Пользователь может настроить игру, имея возможность регулировать громкость игры.

− Пользователь имеет возможность закрыть файлы игры и выйти из игры.

4. **Перечень подлежащих разработке вопросов:**

– проанализировать существующие аналоги и сравнить их

– провести анализ предметной области;

– спроектировать и разработать компьютерную игру;

– протестировать работоспособность разработанной компьютерной игры в нагрузочном тестировании.

5. **Дата выдачи задания:** 1 декабря 2021 г.

Руководитель работы _____/Ю. Г. Плаксина/

Студент _____/М. Х. Хамед /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	10.03.2022	
Разработка модели, проектирование		
Определение функциональных требований	13.03.2022	
Определение нефункциональных требований	15.03.2022	
Архитектура предлагаемого решения	19.03.2022	
Алгоритмы решения задачи	21.03.2022	
Реализация системы		
Реализация интерфейсов	04.04.2022	
Тестирование, отладка, эксперименты		
Методология тестирования	07.04.2022	
Проведение процедуры тестирования	25.04.2022	
Компоновка текста работы и сдача на нормоконтроль	16.05.2022	
Подготовка презентации и доклада	24.05.2022	

Руководитель работы_____/*Ю. Г. Плаксина* /

Студент _____/*М. Х. Хамед*/

АННОТАЦИЯ

М.Х. Хамед. Разработка аркадной 2D-игры на unity. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2022, 42 с., 13 ил., библиогр. список – 21 наим.

Цель данной работы – разработка двухмерной аркадной игры для компьютеров, с возможностью запуска на мобильных устройствах, содержащей бесконечную беговую среду с функциональной базовой игровой механикой.

В данной работе изучались методы разработки 2D-игр. Выполнен анализ требований, проектирование и тестирование разрабатываемой игры, где игрок – падающий снежок, и ему нужно продолжать прыгать по столбам, набирать очки и избегать падения. для разработки проекта был выбран и использован движок Unity.

Результат проделанной работы – работающая 2D игра.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 8

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 10

1.1 Обзор аналогов 10

1.2 Анализ основных технологических решений 14

1.3 Вывод 15

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ 16

2.1 Функциональные требования 16

2.2 Нефункциональные требования 17

3 ПРОЕКТИРОВАНИЕ 18

3.1 Архитектура предлагаемого решения 18

3.2 Алгоритмы решения задачи 27

4 РЕАЛИЗАЦИЯ 29

4.1 Реализация интерфейсов 29

5 ТЕСТИРОВАНИЕ 33

5.1 Методология тестирования 33

5.2 Проведение процедуры тестирования 34

ЗАКЛЮЧЕНИЕ 36

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 37

Приложение А «Исходный код» 39

ВВЕДЕНИЕ

Разработка игр – это искусство создания игр, описывающее дизайн, разработку и выпуск игры. Это может включать в себя создание концепции, дизайн, сборку, тестирование и выпуск. Пока игра создается, важно подумать об игровой механике, наградах, вовлеченности игроков и дизайне уровней [1].

В данной работе рассматривается создание компьютерной игры жанра «бесконечный раннер» на движке Unity. Игра написана с использованием языка программирования C#.

Бесконечный раннер (англ. Endless runner) – это поджанр платформера, в котором персонаж игрока вынужден бежать бесконечное количество времени, избегая препятствий. Цель игрока – набрать высокий балл, выживая как можно дольше. Метод, с помощью которого игровой уровень или окружающая среда постоянно появляются перед игроком, является примером процедурной генерации. Его популярность объясняется простым игровым процессом, который хорошо работает на устройствах с сенсорным экраном [2].

Платформенная игра – это жанр видеоигр, основной целью которого является перемещение персонажа игрока между точками в визуализируемой среде [3].

Спустя годы, в результате эволюции аппаратного обеспечения и потребности в быстрых циклах разработки игр, возникла концепция игрового движка. Игровой движок – это многоразовый программный слой, позволяющий отделить общие игровые концепции от игровых ресурсов (уровней, графики и т. д.) [4].

Игровой движок Unity был впервые анонсирован на Всемирной конференции разработчиков Apple в 2005 году, и с тех пор он произвел серьезные изменения в индустрии видеоигр. Он может экспортироваться на 15 платформ (с той же кодовой базой), таких как iOS, Android, Windows, PlayStation 5, Xbox Series X. Unity состоит из визуального редактора и IDE, что

позволяет быстро создавать прототипы. Простые сцены (без полноценной игровой механики) можно реализовать даже без строчки кода [4].

C# – это высокоуровневый, объектно-ориентированный язык программирования. В C# также есть перегрузка операторов, перечисления, типы значений и языковые конструкции для перебора коллекций [5]. Приложения C# делятся на две отдельные категории: приложения командной строки или консоли и приложения Windows. При использовании AppWizards будет понятно, что оба они легко создаются с точки зрения кода шаблона, необходимого для набросков проекта. Полноценное объектно-ориентированное приложение для Windows [6].

Цель работы: создание 2D игры.

Задачи работы:

1) провести обзор аналогичных решений и осуществить постановку задачи;

2) провести обзор современных средств реализации;

3) провести анализ требований и спроектировать игру;

4) реализовать 2D игру;

5) провести тестирование реализованной 2D игры.

Работа разделена на пять глав: Анализ предметной области, который содержит обзор аналогов и анализ основных технологических решений. Определение требований, которое содержит функциональные и нефункциональные требования. Проектирование, которое включает архитектуру предлагаемого решения и алгоритмы решения задачи. Реализация, в которой была показана реализация интерфейсов. Кроме того, тестирование, которое содержит методологию тестирования и проведение процедуры тестирования.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

Критерии, по которым сделан обзор, представлены ниже.

Язык программирования. C# – это простой, современный, объектно-ориентированный и типобезопасный язык программирования, а также самый популярный язык программирования при разработке игр. Кроме того, .NET поставляется с большим набором библиотек, поддерживающих строки, дженерики, структуры данных, наследование, делегаты, полиморфизм, доходность и т.д [7].

Жанр. Жанр бесконечных раннеров или аркад – хороший жанр игр, оптимизированный как для персонального компьютера, так и для мобильных платформ из-за небольших размеров игр этого жанра [8].

Платформа. Важно, чтобы игра могла поместиться на ПК и могла быть полностью оптимизирована для мобильных устройств [9].

Игровой движок, для того чтобы определить наиболее удобный игровой движок, который подходит независимым разработчикам [10].

Стоимость разработки игры. Важно разработать игру с минимальными финансовыми затратами, поскольку бюджет независимого разработчика обычно ограничен.

Стоимость покупки игры. Важно, чтобы стоимость игры была приемлема для пользователя.

Жанр Endless Runner (бесконечный раннер) – это постоянно движущийся персонаж, которому предстоит преодолевать препятствия. В этих играх могут быть уровни с началом и концом или они могут никогда не закончиться. Главный фактор – это персонаж, который никогда не перестает двигаться, рассчитывать время и ловкость. Основная цель большинства Endless Runner – пройти как можно дальше по уровню [11].

Далее следует описание нескольких известных игр в жанре бесконечный раннер:

Jetpack Joyride – это бесконечный раннер с боковой полосой прокрутки в 2011 году, созданный Halfbrick Studios на игровом движке EC2 на языке C++. Он был выпущен для устройств iOS в App Store 1 сентября 2011 г. и перенесен на другие системы. Игроку предлагаются три миссии, в которых он должен победить все из них, чтобы быть объявленным победителем [12].

Плюсы – Игра затягивает и захватывает, потому что игрок всегда оказывается в сложной ситуации, убегая от врагов и препятствий. В нее можно играть полчаса или 10 минут. Юмористические персонажи добавляют игре энтузиазма. И самое приятное в игре то, что она дает ощущение аркады благодаря классическому дизайну и игровому процессу (геймплей), который напоминает нам аркадные игры [13].

Минусы – Игра подвержена зависаниями, иногда, когда игрок находится в середине игры, игра зависает и закрывается [13].

Temple Run был выпущен Imangi Studios, LLC, создателями Harbour Master на игровом движке Unity на языке C#. Первоначально игра была выпущена для устройств iOS 4 августа 2011 года, а затем была перенесена на системы Android. В работающей платформере игрок выступает в роли охотника за сокровищами, спасающегося от разъяренных стражей древнего храма. Игрок должен прыгать, летать и быстро поворачиваться, собирая монеты и убегая от преследователей [14].

Игрок будет бегать по поросшим лозой скалам и сломанным деревянным мостам, перепрыгивать через пни древних деревьев и скользить над огном. Если игрок пропустит ход, он может упасть в реку, врезаться в мост или просто споткнуться о собственные ноги и быть пойманным другим персонажам игры.

Плюсы – игра очень увлекательная, отличная графика, быстрая навигация.

Минусы – перегревается устройство, постоянно прерывается финансовые затраты на покупку обновлений приложении [14].

Super Mario Run – платформенная видеоигра 2016 года на игровом движке Unity на языке C#, разработанная и изданная Nintendo для iOS и более поздних версий Android. Это первая мобильная игра Nintendo, которая является частью одной из давних и крупных франшиз компании [15].

В Super Mario Run игрок управляет Марио или другими персонажами, когда они автоматически бегают по экрану во время прыжков по времени, чтобы собирать монеты и уклоняться от врагов и опасностей. Игрок автоматически бежит слева направо и самостоятельно прыгает, чтобы преодолеть небольшие промежутки или препятствия. Игрок может управлять Марио, касаясь сенсорного экрана, чтобы он перепрыгивал через большие препятствия.

Плюсы – Легко установить и играть, хорошо оптимизированная игра, хорошо проработанный интерфейс, продуманная компьютерная графика.

Минусы – очень медленный режим создания королевства, зависимость от подключения к Интернету [15].

Результаты обзора существующих аналогов представлены в таблице 1 и 2.

Таблица 1 – Результаты обзора существующих аналогичных продуктов с техническими критериями

Название игры	**Кроссплатформенность**	**Игровой движок**	**Язык программирования**
Jetpack Joyride	+	EC2	C++
Temple Run	+	Unity	C#
Super Mario Run	+	Unity	C#

Таблица 2 – Результаты обзора существующих аналогичных продуктов с финансовыми критериями

Название игры	**Внутриигровые покупки**	**Стоимость разработки игры**	**Стоимость покупки игры**
Jetpack Joyride	+	не объявлено	бесплатно
Temple Run	+	($20 – $50)k	бесплатно
Super Mario Run	+	не объявлено	$10

В наши дни многие игры основаны на микротранзакции (внутриигровые покупки), и представленные игры не являются исключением. Некоторые из них достаточно просты, но удовлетворяют всем требованиям игроков. Ряд других игр имеют достаточно сложную графическую реализацию. Простые и сложные, с точки зрения графической реализации, игры имеют одни и те же проблемы [16]. Разработчики могут создавать простые творческие игры, используя простые игровые движки, которые удобные как для опытных программистов, так и для начинающих. Использование достаточно простого Unity, который упрощает создание кроссплатформенных игр, а не такого сложного как Unreal Engine, также отвечает требованиям игроков без необходимости вставлять микротранзакции. Стабильный игровой процесс (геймплей) и минимальное количество системных сбоев достигается применением Unity. Сбои игры, зависания или ошибки в игровом процессе (глюки), плохо повлияют на игровой процесс независимо от того какая среда разработки используется [17].

1.2 Анализ основных технологических решений

В настоящее время существует множество платформ для разработки игр. Они значительно облегчают процесс разработки приложений, экономя время и трудовые ресурсы.

Unity – инструмент для создания 2D- и 3D-игр, а также интерактивного контента. Платформа поддерживает два языка сценариев: C# и JavaScript (модифицированный для Unity).

Плюсы:

– выгодная лицензионная политика;

– простота использования;

– совместимость с любой платформой;

– популярен среди разработчиков (имеется в виду, что ошибки быстро находят и исправляют).

Минусы:

– ограниченный набор инструментов;

– процесс разработки игры занимает много времени [18].

Unreal Engine позволяет создавать игры для большинства операционных систем и платформ, а также для различных портативных устройств, таких как устройства Apple (iPad, iPhone), управляемые iOS и другие. Платформа поддерживает различные системы получения изображений (Direct3D, OpenGL, Pixomatic), воспроизведения звука (EAX, OpenAL, DirectSound3D), преобразования текста в речь, распознавания речи, а также имеет различные модули для работы в сети и поддержки различных устройств ввода.

Плюсы:

– одно из лучших сообществ разработчиков;

– отличная техническая поддержка;

– совместимость с любой платформой;

– новые инструменты выпускаются с каждым обновлением;

– широкий выбор инструментов для различных целей.

Минусы:

– некоторые разработчики жалуются, что к некоторым инструментам трудно – привыкнуть.

– подписка на 19 долларов в месяц и 5%, если игра зарабатывает более 5000 долларов [18].

1.3 Вывод

В данной работе рассматривается создание компьютерной игры на движке Unity. После обзора существующих аналогов и анализа основных технологических решений Unity оказался лучшим выбором в качестве движка для независимого разработчика, за что имеет наличие огромной библиотеки ассетов и плагинов, с помощью которых можно значительно ускорить процесс разработки игра.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

К общим требованиям для персонального компьютера можно отнести минимальные системные требования:

– операционная система (ОС): Windows 7 или выше/Linux/Android 4.4 (KitKat);

– процессор: Intel Core i3 540 (1st gen.);

– объем оперативной памяти: 4 GB;

– видеокарта: видеокарта с DirectX 11.0 с 1 GB VRAM (NVidia GeForce 460);

– DirectX: версии 11;

– место на диске: 500 MB;

– звуковая карта: звуковая карта совместимая с DirectX.

Данные системные требования являются минимальными для продукции, которая создается на движке Unity.

2.1 Функциональные требования

В рамках реализованного приложения присутствует один пользователь, взаимодействующий с игровым сюжетом и компонентами меню игры.

Пользователь может:

– начать игру;

– управлять персонажам прыгая (кнопкой пробел);

– настроить игру, изменив громкость;

– выйти из игры.

Пользователь имеет возможность запустить файл игры и попасть в главное меню.

Пользователь может управлять персонажем во время игры, дважды прыгая с помощью клавиши пробела, в первый раз пользователь может удерживать клавишу пробела, чтобы персонаж взлетел так высоко, как

необходимо, во второй раз может быть только одно нажатие, чтобы совершить двойной прыжок.

Пользователь может настроить игру, имея возможность регулировать громкость игры.

Пользователь имеет возможность закрыть файлы игры и выйти из игры.

2.2 Нефункциональные требования

К нефункциональным требованиям относятся:

– разработанное приложение должно соответствовать определенным минимальным системным требованиям;

– разработанное приложение должно быть написано на языке C# на платформе Unity;

– разработанное приложение должно сохранять максимальные баллы, набранные во время игры.

3 ПРОЕКТИРОВАНИЕ

3.1 Архитектура предлагаемого решения

Проект разрабатываемого игрового приложения представляет собой список каталогов, содержащих:

– скрипты;

– сцены;

– объекты;

– изображения персонажей, столбцов, и фонов.

Файлы игры содержат скрипты для главного меню, позволяющие пользователю либо запустить игру, либо выйти из игры, либо перейти к настройкам.

В настройках есть скрипт для регулировки громкости и возврата в главное меню.

Скрипты игрока и движения предназначены для того, чтобы сцена игры двигалась горизонтально, и чтобы игрок мог совершать прыжки, увеличивать свой размер со временем и определять, когда игрок проигрывает, то есть, когда игрок либо попадает сторону столба или упасть.

На рисунке 1 представлен алгоритм движения игрока.

Рисунок 1 – Алгоритм движения игрока

На рисунке 2 представлен алгоритм перемещения объектов по оси x.

Рисунок 2 – Алгоритм перемещения объектов по оси x

На рисунке 3 представлен алгоритм перепасовки столбов.

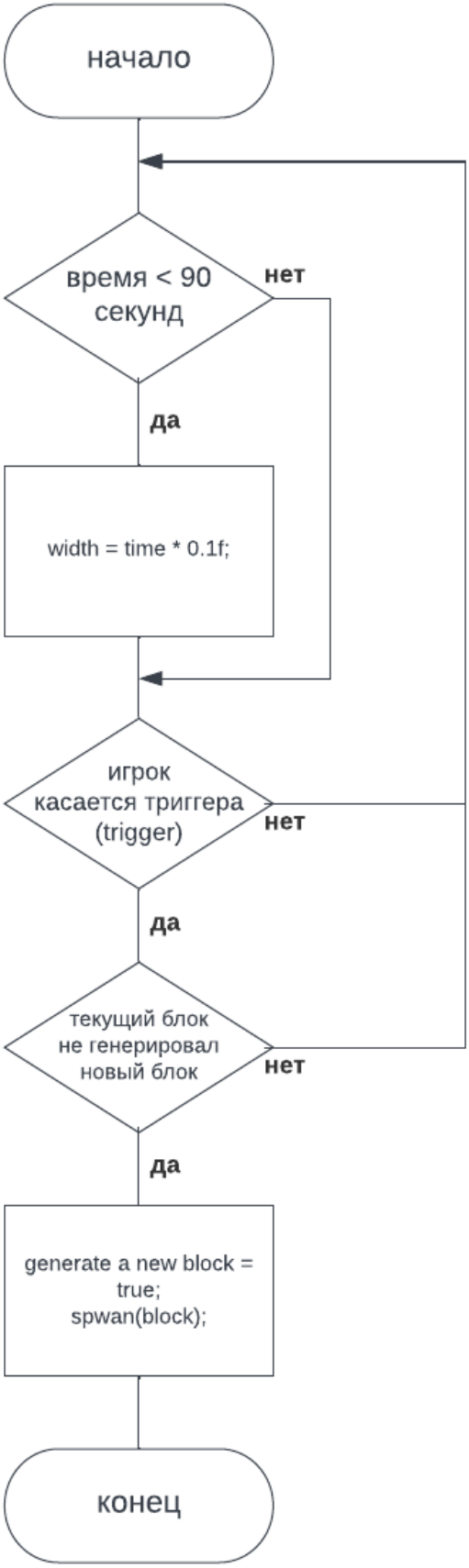

Рисунок 3 – Алгоритм перепасовки столбов

Скрипт времени для подсчета очков в зависимости от того, сколько времени прошло.

На рисунке 4 представлен алгоритм подсчета очков.

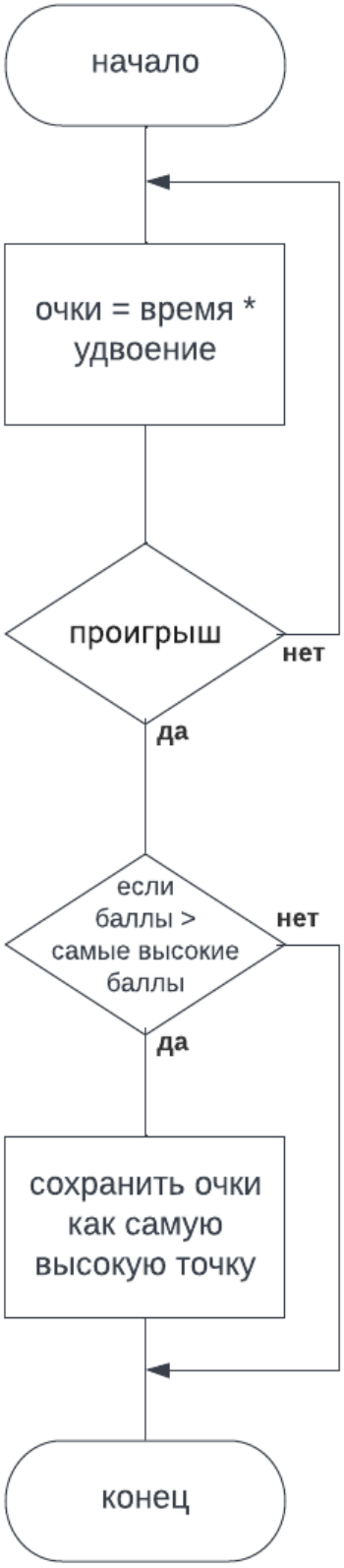

Рисунок 4 – Алгоритм подсчета очков

Так же есть скрипты для иконок огня и звезды, спаунятся они скриптом спавнера.

Скрипт пламени возвращает начальный размер игрока, он активируется, когда игрок касается значка пламени.

На рисунке 5 представлен алгоритм получения начального размера игрока (значок пламени).

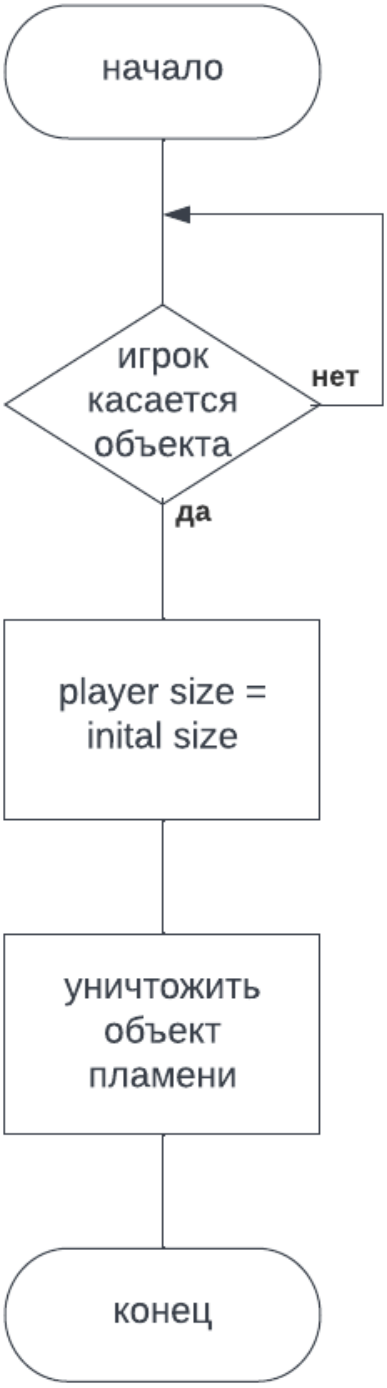

Рисунок 5 – Алгоритм получения начального размера игрока

Скрипт звезды удваивает текущие набранные очки, удваивая пройденное время, он активируется, когда игрок касается значка звезды.

На рисунке 6 представлен алгоритм объекта удвоения баллов (значок звезды).

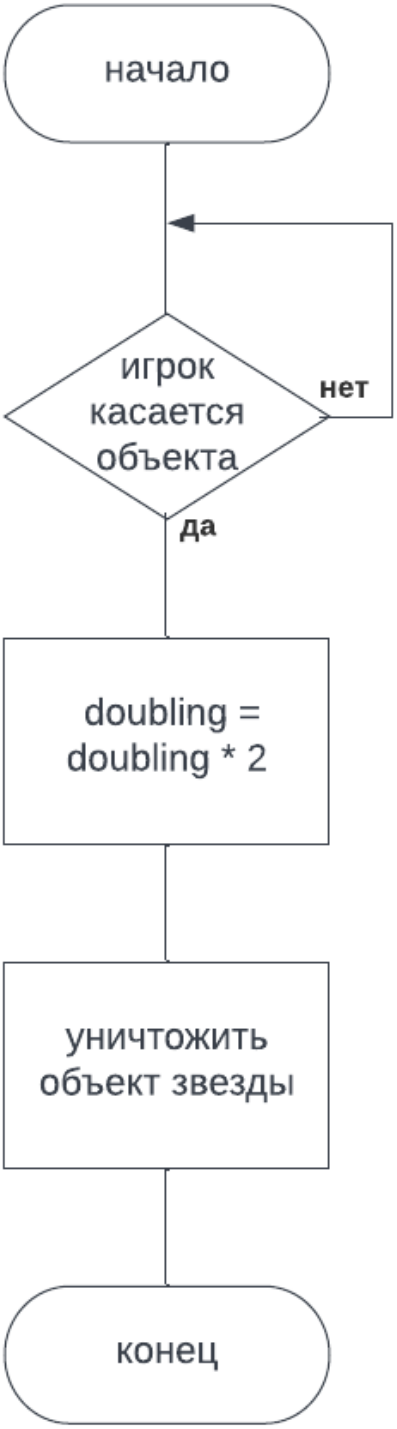

Рисунок 6 – Алгоритм объекта удвоения баллов

Скрипт экрана камеры предназначен для отслеживания положения игрока, не теряя его за пределы сцены.

И, наконец, скрипт перехода в главное меню, который показывает набранные очки и возвращает игрока в главное меню после проигрыша.

Игра содержит четыре основные сцены.

Первая сцена – это главное меню, где она показывает пользователю главное меню, в котором он может видеть три параметра: запуск, настройки и выход.

Вторая сцена – это настройки, где игроку показывается интерфейс настроек, который содержит возможность регулировать громкость игрового звука.

Третья сцена – это игровой процесс, где пользователю показывается среда игры, в которой он может управлять персонажем.

Последняя сцена – это сцена проигрыша игры, где пользователю показывается интерфейс, который информирует его о проигрыше, а также показывает ему текущий набранный счет и набранные высокие баллы.

В игре пять объектов (префабов).

Объект фона, который для игрока может показаться просто интерфейсом, но на самом деле фон движется горизонтально вместе со столбами, чтобы игрок думал, что персонаж на самом деле движется.

Объект столбов, которые необходимы игроку, чтобы удерживать персонажа (снежок) на них во время движения, а по бокам столбов есть коллайдер, который приводит к окончанию игры (проигрышу), если игрок ударяет по нему.

Объект игрока, то есть объект (снежок), которым пользователь может управлять во время игры, который со временем увеличивается в размерах, и он должен оставаться либо в воздухе, либо на поверхности столбов, чтобы не проиграть игру, которая может произойти, ударившись о столбы, как упоминалось ранее, или упав с карты.

Объект огня, имеющий форму значка огня, который, если игрок коснется его, восстанавливает свой первоначальный размер, как указано в объяснении скриптов.

Объект звезды, имеющий форму значка звезды, который, если игрок коснется его, удвоит текущие набранные очки (очки), как указано в объяснении скриптов.

На рисунке 7 представлена файловая структура приложения.

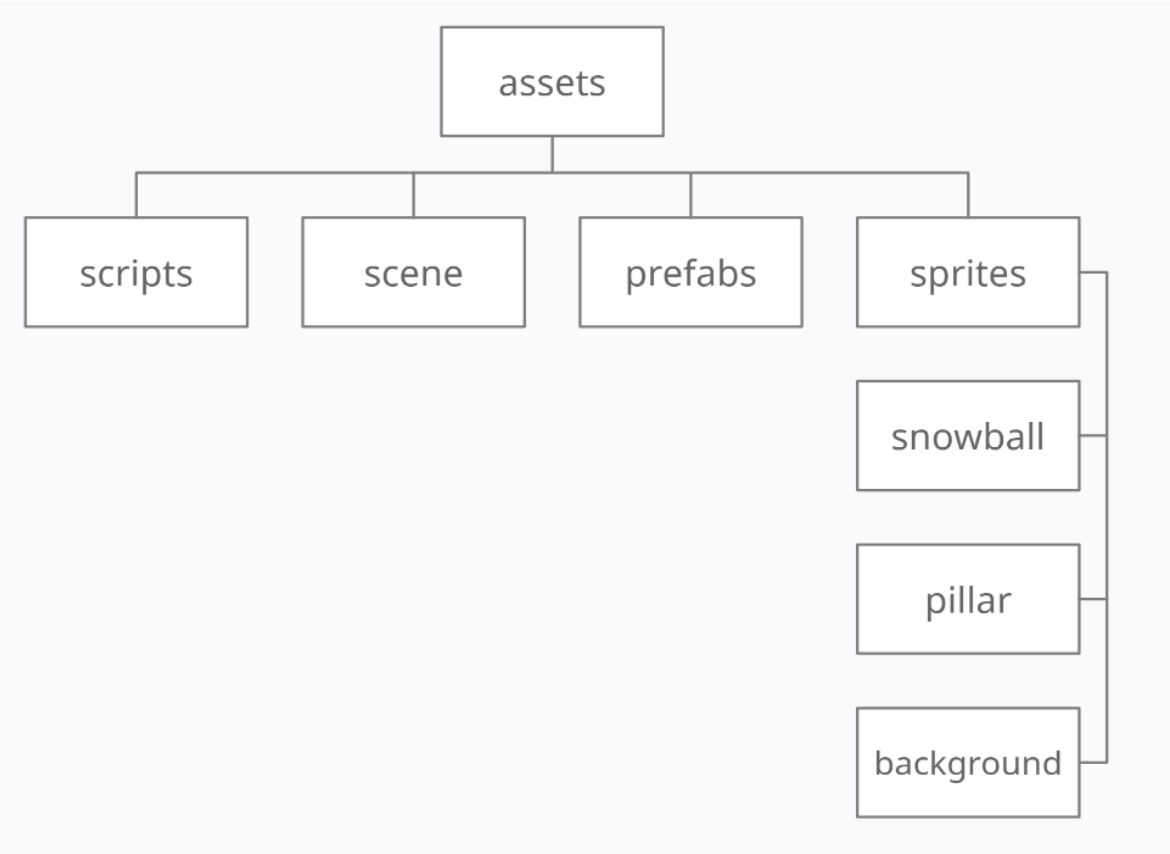

Рисунок 7 – Файловая структура приложения

В директории scripts содержаться скрипты, которые используются в игровом приложении.

В директории scene содержатся игровые сцены, в которые пользователь может пройти.

В директории prefabs содержатся готовые шаблоны игровых объектов, с помощью которых можно составить игровой уровень после его проектирования.

В директории sprites содержатся подкаталоги:

– Snowball содержит спрайт главного персонажа;

– Pillar содержит столбы, на которые игрок должен прыгать;

– Background содержит фоновый вид игровой платформы;

На рисунке 8 представлена структура главного окна игры.

Рисунок 8 – Структура главного окна игры

3.2 Алгоритмы решения задачи

Во время запуска игрового мира инициализируются данные, хранящиеся в корневых папках игры, включающие в себя все модели и текстуры, необходимые для создания локации.

Игра начинается с того, что главный герой находится на столбе. Есть три столба, которые появляются (некоторые из них за кулисами) и уничтожаются каждые десять секунд с помощью функций Invoke и Destroy, что позволяет снизить нагрузку на процессор. Игроку нужно продолжать прыгать по столбам

и не падать между ними. Если игрок прикоснется к значку звезды, он удвоит текущий счет, что происходит с использованием сценария звездного объекта, который удваивает время, отсчитываемое в игре, и именно так работают результаты (по умолчанию они увеличиваются на 10 каждую секунду).

Если игрок касается значка пламени, размер снежка уменьшается до исходного размера с помощью скрипта в объекте пламени, что позволяет игроку двигаться более свободно.

Если игрок падает с карты, он проигрывает, и игра заканчивается, то игрок возвращается в главное меню. Если нет (находясь в воздухе или на поверхности столба), игрок продолжает играть.

4 РЕАЛИЗАЦИЯ

4.1 Реализация интерфейсов

Название игры: «SnowFall».

Главный герой – снежок, который должен прыгать по столбам и набирать очки, не падая.

Цель игры – набрать наибольшее количество очков.

Смысл этой игры: управлять снежком на склоне и продолжать прыгать по ледяным столбам, увеличиваясь в объеме со временем из-за льда, с которым он взаимодействует. Если снежок падает или поднимается слишком высоко, игра заканчивается.

При создании игры были разработаны следующие интерфейсы для взаимодействия программы с игроком:

– главное меню;

– интерфейс персонажа.

Неотъемлемой частью каждой игры является главное меню. В ней пользователь может настроить игру под себя. Главное меню содержит следующие пункты:

– start (начать игру);

– settings (настройки), где можно регулировать громкости;

– exit (выход).

На рисунке 9 представлено главное меню.

Рисунок 9 – Главное меню

На рисунке 10 представлено настройки.

Рисунок 10 – Настройки

На рисунке 11 представлен скрин начало игры.

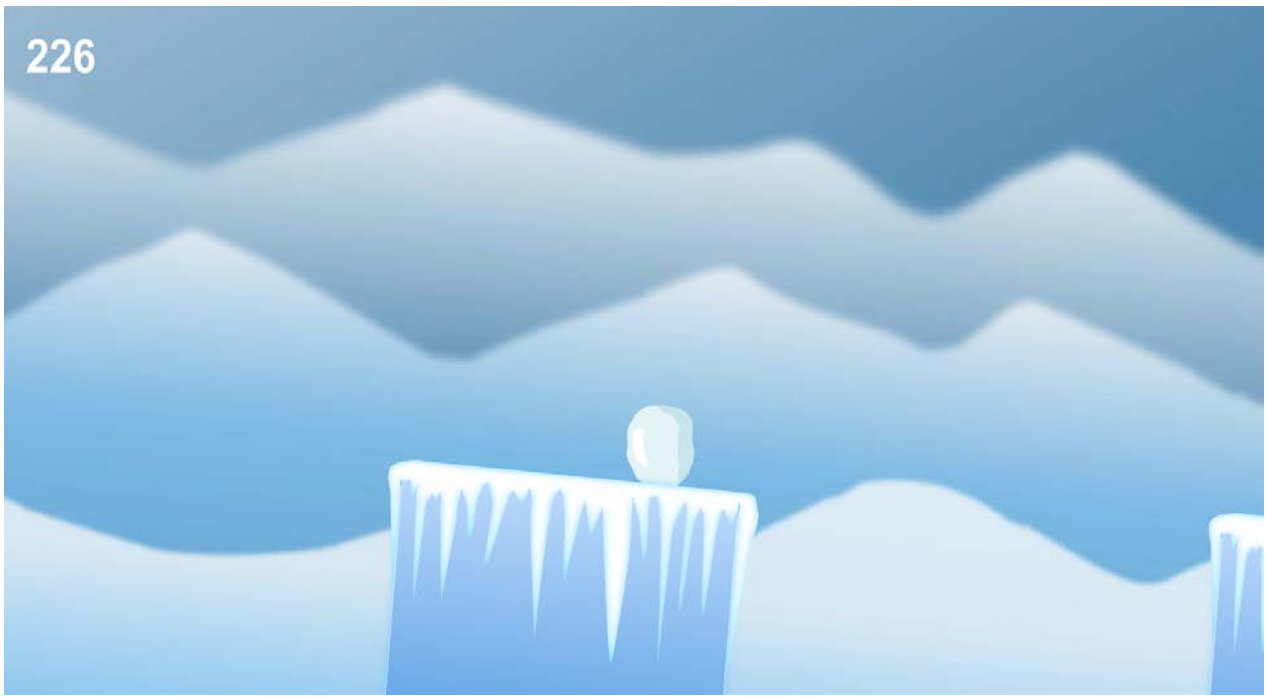

Рисунок 11 – Начало игры

На рисунке 12 представлен игровой процесс.

Рисунок 12 – Игровой процесс

На рисунке 13 представлено завершение игры после проигрыша.

Рисунок 13 – Завершение игры после проигрыша

5 ТЕСТИРОВАНИЕ

5.1 Методология тестирования

Для тестирования приложения использовался метод функционального тестирования.

Функциональное тестирование – это тестирование программного обеспечения с целью проверки выполнимости функциональных требований, то есть способности программного обеспечения при определенных условиях решать задачи, которые нужны пользователям [19].

Функциональное тестирование имеет так много преимуществ, чтобы предоставить пользователям эффективное и надежное прикладное программное обеспечение, ниже приведены различные преимущества функционального тестирования.

Функциональное тестирование сосредоточено на требованиях руководств для конечных пользователей по тестированию различных моделей приложений. Он принимает реалистичные факты и цифры для составления отчетов об испытаниях и обеспечивает правильную работу всех функций приложения. Функциональное тестирование также производит бездефектный продукт/программное обеспечение и гарантирует, что все требования должны быть выполнены [20].

Функциональное тестирование также имеет некоторые недостатки, которые указаны ниже.

Это процесс, при котором в процессе тестирования не обнаруживаются различные логические ошибки в программном обеспечении.

Он не учитывает, как разработчик реализует фактический исходный код, поскольку фокусируется только на результатах исходного кода [21].

5.2 Проведение процедуры тестирования

Технические характеристики персонального компьютера, использованного в тестировании:

– операционная система (ОС): Windows 10;

– процессор: Intel Core/ i3-9100f;

– оперативная память: 16 GB;

– видеокарта: Nvidia GTX 1060 6gb.

В таблице 3 отражено тестирование игры «SnowFall».

Таблица 3 – Тест-план

№	Назначение	Действия	Ожидаемый результат	Полученный результат	Итог
1	Проверка правильности реакции персонажа на поступающие команды движения	Используя клавишу прыжка, пробежать часть уровня	Персонаж будет прыгать в правильном направлении	Персонаж прыгнул правильно	Пройден
2	Проверка на корректную реализацию события проигрыша	Разрешить персонажу упасть со столбов	Появляется экран проигрыша, а затем переход в главное меню.	Появилось экран проигрыша	Пройден
3	Проверка на корректную работу главного меню	Запустить игру, перейти к настройкам, выйти из игры	Есть возможность запустить игру, перейти к настройкам и выйти из игры	В зависимости от выбора запускается игру, настройки, и закрывается игра	Пройден
4	Проверка корректности при взаимодействии с объектами.	Позволить персонажу коснуться значка звезды или значка пламени.	При прикосновении к пламени размер персонажа уменьшается до исходного размера. При прикосновении к звезде очки удваиваются.	При прикосновении к пламени размер персонажа уменьшался до исходного размера. При прикосновении к звезде очки удваивались.	Пройден

Данное тестирование показывает, что на данном этапе реализации проекта игра соответствует функциональным требованиям, определенным на этапе постановки задачи.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы была разработана игра «SnowFall» на платформе Unity.

В ходе данной работы была проделана большая работа, начиная с изучения литературы по программированию и заканчивая тестированием и отладкой готового программного продукта.

Были решены следующие задачи:

– проведен обзор подобных решений и осуществлена постановка задачи;

– проведен обзор средств реализации;

– проведен анализ требований и разработана компьютерная игра;

– протестирована реализованная версия игры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Dave, C. Unity 2D Game Development / C. Dave. – Birmingham : Packt Publishing, 2014. – 126 p. – ISBN: 978-1-84969-256-4.

2. The Leaderboard: The Loneliness of the Endless Runner. – Текст : электронный // Pastemagazine : [сайт]. – URL: https://www.pastemagazine.com/games/the-leaderboard-the-loneliness-of-the-endless-runn (Дата обращения: 28.03.2022).

3. Gillian, S. A framework for analysis of 2D platformer levels / S. Gillian. – New York : ACM SIGGRAPH, 2008. – P. 75–80. – DOI:10.1145/1401843.1401858.

4. Andrade, A. Game engines: a survey / A. Andrade. – Porto : Antonio Andrade, 2015. – 6 p. – DOI: 10.4108/eai.5-11-2015.150615.

5. David, R. H. A research C# compiler / R. H. David, T.P. Todd. – Redmond : Wiley InterScience, 2004. – 1224 p. – DOI: 10.1002/spe.610.

6. Research paper on C#. – Текст : электронный // Courses edu : [сайт]. – URL: https://courses.cs.vt.edu/~cs5314/Lang-Paper-Presentation/Papers/HoldPapers/C-Sharp.pdf (Дата обращения: 28.03.2022).

7. Thomas, H. A. Genre and game studies: Toward a critical approach to video game genres / H. A. Thomas. – Parkville : Sage Publications, 2006. – 236 p. – DOI: 10.1177/1046878105282278.

8. Joe, H. Multiplatform Game Development in C# / H. Joe. – 3-d ed. – New York : Manning Publications, 2022. – 385 p. – ISBN: 978-1-61729-933-9.

9. Alan, T. Pro Unity Game Development with C# / T. Alan. – New York : Paul Manning, 2014. – 335 p. – ISBN: 978-1-4302-6745-4.

10. Endless Runner. – Текст : электронный // VideoGameGeek : [сайт]. – URL: https://videogamegeek.com/videogamegenre/2723/endless-runner (Дата обращения: 09.03.2022).

11. Jetpack Joyride. – Текст : электронный // Wikipedia : [сайт]. – URL: https://en.wikipedia.org/wiki/Jetpack_Joyride (Дата обращения: 09.03.2022).

12. Jetpack Joyride Android game Review. – Текст : электронный // Techulator : [сайт]. – URL: https://www.techulator.com/resources/7649-Jetpack-Joyride-Android-game-Review.aspx (Дата обращения: 10.03.2022).

13. Temple Run Review. – Текст : электронный // Tapscape : [сайт]. – URL: https://www.tapscape.com/temple-run-review (Дата обращения: 10.03.2022).

14. Super Mario Run. – Текст : электронный // Wikipedia : [сайт]. – URL: https://en.wikipedia.org/wiki/Super_Mario_Run (Дата обращения: 10.03.2022).

15. Jere, M. Unity 3D and PlayMaker Essentials / M. Jere. – Florida : CRC Press, 2016. – 506 p. – ISBN: 978-1-31741-431-5.

16. Паласиос, Х. Unity 5.x. Программирование искусственного интеллекта в играх / Х. Паласиос; пер. с англ. Р. Н. Рагимова. – Москва : ДМК Пресс, 2017. – 272 с. – ISBN: 978-5-97060-436-6.

17. Ryan, W. Procedural Content Generation for Unity Game Development / W. Ryan. – Birmingham : Packt Publishing, 2016. – 236 p. – ISBN: 978-1-78528-747-3.

18. Какой игровой движок выбрать: Unity, Udk или Cryengine? . – Текст : электронный // 3dpapa : [сайт]. – URL: https://3dpapa.ru/what-game-engine-to-choose (Дата обращения: 03.04.2022).

19. Майерс, Г. Искусство тестирования программ / Г. Майерс, Т. Баджет. – М : Вильямс, 2012. – 272 с. – ISBN: 978-5-907203-66-2.

20. Functional Testing Types and its Benefits. – Текст : электронный // Xenonstack : [сайт]. – URL: https://www.xenonstack.com/insights/what-is-functional-testing (Дата обращения: 20.05.2022).

21. Functional testing and its advantages and disadvantages. – Текст : электронный // Negosentro : [сайт]. – URL: https://negosentro.com/functional-testing-and-its-advantages-and-disadvantages (Дата обращения: 20.05.2022).

Приложение А

«Исходный код»

Листинг 1 – Кнопки меню

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using TMPro;

public class Main_Menu : MonoBehaviour
{
    public TextMeshProUGUI highScore;
    private void Start()
    {
        string point = PlayerPrefs.GetFloat("highScore", 0).ToString("0");
        highScore.text = "Highscore: " + point;
    }
    public void PlayGame()
    {
        SceneManager.LoadScene("gamePlay");
    }
    public void SettingsButton()
    {
        SceneManager.LoadScene("Settings");
    }
    public void BackButton()
    {
        SceneManager.LoadScene("menu");
    }
    public void QuitGame()
    {
        Application.Quit();
    }
}
```

Листинг 2 – Движение игрока

```
    private void sizeUp()
        {
            transform.localScale = new Vector2(transform.localScale.x +
(Time.deltaTime * 0.01f), transform.localScale.y + (Time.deltaTime * 0.01f)); ;
        }

        private void movement()
        {
            //Since the movement function is working the speed will increase, same
as size.
            sizeUp();

            //Keep pressing while the player is on ground.
            if (Input.GetKey(KeyCode.Space) && ground)
            {
                rb.AddForce(new Vector3(0f, fly, 0f));
            }
            //if the player stop pressing space, while it's flying, it means the
falling started, so ground = false.
            else if (Input.GetKeyUp(KeyCode.Space) && ground)
            {
                ground = false;
```

```
            }
            //the player press the space one press, while he's falling or (not on
ground) and he didn't doubled jumped yet (doubleJump = false)
            else if (Input.GetKeyDown(KeyCode.Space) && doubleJump && !ground)
            {
                rb.AddForce(new Vector3(0f, jumpAmount, 0f));
                doubleJump = false;
            }

            if (Input.GetKeyDown(KeyCode.Q))
            {
                Dead();
            }
        }
```

Листинг 3 – Движение объектов по оси x

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class movement : MonoBehaviour
{
    [Header("Speed")]
    [SerializeField] private float speed;
    [SerializeField] private float time;

    // Update is called once per frame
    void Update()
    {
        transform.Translate(new Vector3(speed, 0f, 0f));
        //destory it after time.
        Invoke("destroy", time);
    }

    //destory the movement object
    private void destroy()
    {
        Destroy(gameObject);
    }
}
```

Листинг 4 – Проигрыш

```
private void Dead()
    {
        save();
        SceneManager.LoadScene("lose");
    }
```

Листинг 5 – Вернуться в главное меню

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class switchToMenu : MonoBehaviour
{
    // Update is called once per frame
```

```
    void Update()
    {
        Invoke("toMenu", 3f);
    }

    void toMenu()
    {
        SceneManager.LoadScene("menu");
    }
}
```

Листинг 6 – Уменьшение размера до исходной формы

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class fire : MonoBehaviour
{
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if(collision.gameObject.CompareTag("Player"))
        {
            //change the size to intial size.
            collision.gameObject.transform.localScale = new Vector2(0.35f,
0.35f);
            Debug.Log("touch fire");
            Destroy(gameObject);
        }
    }
}
```

Листинг 7 – Удвоение очков

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class star : MonoBehaviour
{
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {

            timing.doubling *= 2;
            Debug.Log(timing.doubling);
            Destroy(gameObject);
        }
    }
}
```

Листинг 8 – Счетчик

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class timing : MonoBehaviour
```

```
{

    //TEXT TO SHOW
    private Text text;

    //POINT STARTS
    public static float point = 0;
    //DOUBLING RATIO IN BEGINING IS 10

    public static float doubling = 10f;
    // Start is called before the first frame update
    void Start()
    {
        text = gameObject.GetComponent<Text>();
    }

    // Update is called once per frame
    void Update()
    {
        point =
GameObject.FindGameObjectWithTag("Player").GetComponent<player>().time * doubling;
        text.text = (point).ToString("0");

    }
}
```