МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-9 «Предметно-ориентированные информационные системы»

КУРСОВОЙ ПРОЕКТ (РАБОТА)

по дисциплине Технологии программирования

Тема курсового проекта (работы)«Разработка 3D игры Rolling на платформе Unity»

Студент группы БСБО-14-21 _____

(учебная группа) Бучнева М.В.

Руководитель курсового проекта (работы) _____

должность, звание, ученая степень Курбанисмаилов З.М

Работа представлена к защите«__»_____20__г. _____

подпись студента

Допущен к защите «__»_____20__г. _____

_____

подпись руководителя

Москва 2022 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В наше время игры набирают всё большую популярность. Видеоигры — стали частью современной культуры. Соревнования по киберспорту собирают огромное количество зрителей. Поэтому разработка игровых продуктов является большим развивающимся сегментом программного обеспечения.

Актуальность данной работы заключается в получении практических навыков создании программного обеспечения в области разработки игр.

Целью работы является разработка игры по поставленной задаче на платформе Unity.

Объектом исследования является игровая платформа Unity и её использование для разработки игр на PC, а именно 3-d игр, работающих на платформе Windows.

Предметом исследования является практическая разработка игр на платформе, выбранной объектом исследования.

Задачи курсовой работы:

- Изучение возможностей платформы Unity.
- Проектирование и разработка игры.
- Тестирование на соответствие поставленным требованиям.
- Тестирование собранного решения (.exe файла) независимыми пользователями на независимых машинах.
- Оптимизация и улучшение игры на основе обратной связи от пользователей.

1. ПОСТАНОВКА ЗАДАЧИ

Нужно реализовать игру, где шар собирает бананы, пока те не исчезли (после 3-х исчезновений игра заканчивается), и убегает от охотников (если охотник поймает шар, игра закончится). Управление движением шара должно выполняться с помощью стрелок. На экране идёт подсчёт очков, кроме того, должно присутствовать звуковое сопровождение.

2. ПОИСК И АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

2.1 Unity

Unity появился в 2005 году и стал ведущей платформой для разработки игр и используется для создания половины игр в мире. Программа поддерживает функции разработки как 2D, так и 3D игр.

Для использования необходимо знать C#. В Unity существует широкий спектр инструментов, благодаря которым, можно быстро редактировать и повторять во время рабочего процесса создания игры, включая режим игры, который помогает разработчику предварительно просмотреть свой дизайн в реальном времени.

В Unity существует большое количество возможностей экспорта, такие как: Windows, Linux, Mac, iOS, Android, HTML5, все разновидности систем виртуальной реальности наподобие Oculus Rift и Steam VR, поддержка игровых консолей Xbox One, PlayStation4, Nintendo Switch и Nintendo Wii U.

2.1.1 Плюсы Unity

- Удобный интерфейс.
- Использование компонентно-ориентированного подхода.
- Наличие обширной библиотеки ассетов и плагинов.
- Поддержка большого количества платформ и технологий.
- Бесплатность.

2.1.2 Минусы Unity

- Медлительность из-за тяжеловесности файлов.
- Дорогая Pro-версия.

2.2 Construct 2

Данный движок основан на HTML5 и используется для разработки 2D-игр. Он не требует программирования: программа управляется через графический интерфейс.

Construct 2 содержит такие функции, как физика и поиск пути, 8 направлений и другие удобные инструменты, такие как перетаскивание,

мигание, перенос, закрепление т.д. Игру можно тестировать в браузере и публиковать на широком спектре платформ, таких как: iOS, Android, Facebook, Chrome Web Store, Desktop Windows, Windows 8 Apps, Web (HTML5), Kongregate и т.д.

2.2.1 Плюсы Construct 2

- Мультиплатформенность.
- Не требует знаний языков программирования.
- Удобный интерфейс.
- Существует много руководств.

2.2.2 Минусы Construct 2

- Сильно урезанная бесплатная версия и очень дорогая платная.

2.3 Corona

Движок для разработки 2D игр. Использует универсальный язык Lua. Система Live Build обеспечивает тесты разрабатываемой программы без ручной установки. Модульная конфигурация поддерживает подключение внешних API и расширений. В магазине доступно более 200 плагинов.

2.3.1 Плюсы Corona

- Бесплатность.
- Легко осваиваема для новичка.
- Кроссплатформенность.
- Удобство отладки и тестирования проектов.
- Поддержка внешних расширений.

2.3.2 Минусы Corona

- Облачная сборка проектов;
- Слабый бесплатный саппорт (но есть комьюнити).

2.4 libGDX

Использует C++ и Java. Полностью бесплатен и поддерживает популярные современные ОС. Позволяет разрабатывать игры в 2D и 3D и работать с подключаемыми модулями и библиотеками. Проекты, созданные в libGDX, имеют модульную архитектуру, благодаря чему можно подключать специфичные отдельные модуля для каждой платформы (Windows, Android, Linux и т. д.).

2.4.1 Плюсы libGDX

- Бесплатность.
- Кроссплатформенность.
- Наличие виджетов и библиотек для разработки UI.
- Подключаемые внешние расширения и модули.
- Производительность.

2.4.2 Минусы libGDX

- Объемы ручного программирования;
- Слабые возможности работы с 3D.

2.5 Defold

Имеет визуальный редактор, предназначенный для создания 2D-игр, использует язык Lua. Отличается простыми инструментами для управления GUI и GO, подчиняемым редактором и широкими возможностями по работе с анимацией.

Возможна командная разработка: предусмотрена система доступа к проекту разрешенных пользователей. Создаваемые программы занимают мало места.

2.5.1 Плюсы Defold

- Удобный интерфейс.
- Кроссплатформенность.
- Производительность.
- Бесплатность.
- Простота использования.

2.5.2 Минусы Defold

- Своеобразный подход к организации объектов;
- Ограниченные возможности работы с 3D.

2.6 Unreal Engine 4

Был разработан движком Epic Games, который сегодня известен как Unreal Engine. Впервые он был показан в 1998 году в шутере от первого лица Unreal. Можно создавать элементы игры наглядно, перемещая объекты, без ручного ввода кода.

Считается один из самых мощных движков. Один из ключевых принципов – высокая скорость работы. Этому способствует возможность отладки в реальном времени, быстрого перезапуска приложения, удаленного предпросмотра, сотни ассетов и систем на базе алгоритмов ИИ, пост-эффектов и многого другого.

Для написания кодов используется C++. Поддерживается экспорт в: Windows, Linux, Mac, iOS, Android, HTML5, Xbox One, PlayStation4, Oculus VR и так далее. У Unreal Engine 4 есть собственный канал на YouTube, где раскрывают все аспекты работы с движком. Unreal Engine получил несколько наград и даже был удостоен награды Книги рекордов Гиннеса как «самый успешный движок для видеоигр в 2014 году.

2.6.1 Плюсы Unreal Engine 4

- Графический потенциал.
- Скорость.
- Поддержка стационарных и мобильных платформ.
- Большое сообщество.
- Легкость освоения и количество обучающих видеоматериалов.

- Саппорт.
- Имеет больше инструментов и функций, которых нет в других играх аналогичного тип.

2.6.2 Минусы Unreal Engine 4

- Отсутствие совместимости со старыми консолями.
- Настройка на высокоуровневые 3D-игры, делающая инструмент избыточным для простых проектов в 2D.
- Необходимость лицензионной копии и платы 5% налога, при прибыльности игры.
- Недостаточное количество экспертов.

3 ВЫБОР И ОБОСНОВАНИЕ РЕШЕНИЯ

Для реализации поставленной задачи была выбрана среда разработки Unity.

На данный момент среда разработки Unity является одной из самых распространенных систем для разработки игр: каждый месяц люди скачивают до 2 миллиардов копий игр Unity, которые занимают многочисленные доли рынка на различных платформах, более половины мобильных игр созданы на Unity.

Состоит Unity из трех частей:

Игровой движок, позволяющий создавать игры в различных средах.

Приложение – для размещения дизайна или пользовательского интерфейса вместе с опцией предварительного просмотра графики и функцией управления воспроизведением.

Редактор кода – называется IDE и предоставляет текстовый редактор для написания кода. Однако, как правило, используется отдельный текстовый редактор, чтобы избежать неудобств.

Программы в Unity разрабатываются на C#, одном из самых распространённых языков программирования. C# является усовершенствованным языком C++ с более удобным синтаксисом. Данный язык использует объектно-ориентированный подход, что делает его очень удобным. C# поддерживает универсальные методы и типы, которые повышают безопасность типов и производительность. Кроме того, он имеет большое количество библиотек и шаблонов, что позволяет значительно сократить время, затрачиваемое на программирование. Например, технология System.Windows.Forms включает в себя большое количество библиотек, таких как System, System.Collections.Generic, System.Data, System.Drawing, System.Drawing2D, System.Text и т. д., что открывает огромные возможности для разработчика.

Также нельзя не отметить широкие возможности настройки законов физики, действующих на игровые объекты. Например, можно подключить гравитацию и контролировать её интенсивность. Кроме того, благодаря Коллайдеру объекты сталкиваются, а не проходят сквозь друг друга. Также игровые объекты передвигаются по векторам, из-за чего движение задаётся

по координатам.

Большим плюсом является наличие большого количества обучающих курсов, методичек, блогов, где можно найти ответы на вопросы, возникающие в процессе разработки. Создатели программы даже запустили портал Unity Learn с курсами о разных аспектах работы с движком.

Также благодаря тому, что Unity является бесплатным (если прибыль меньше 100 000 долларов), он является идеальным движком для начинающих разработчиков. Кроме того, Unity подходит для мобильной разработки и инди. В нем можно быстро создавать прототипы, что особенно удобно для создания игр для смартфонов с очень простыми механиками.

Движок поддерживает множество популярных форматов: модели, звуки, текстуры, материалы и другие. Звуки можно использовать при движении, при столкновении и для общего фона. Разнообразные текстуры и материалы позволяют сделать игровой интерфейс более привлекательным для пользователей, при этом сокращая время разработчика на дизайн игры. 2D и 3D модели позволяют создавать уникальные декорации и персонажей, что помогает сделать игру более запоминающейся.

4 РЕШЕНИЕ

В игре существует несколько игровых объектов: сцена, стены, шар, бананы (бонусы), охотники (кубы). Сцена ограничена стенами, чтобы шар не выходил за её пределы. Камера установлена так, что вид на сцену представлен сверху, что позволяет иметь большую зону обзора.

Ход работы:

1. Игра начинается;
2. Шар катится, управляемый пользователем;
3. Появляются бонусы и охотники;
4. Шар подбирает бонусы, если же не успевает за время, отведенное на существование бонуса, то бонус исчезает (после 3-х исчезновений игра заканчивается и начинается заново);
5. Охотники движутся за шаром в попытке его поймать, при поимке игра заканчивается и начинается заново.

Рис. 1 Схема поведения шара при столкновении с бонусами

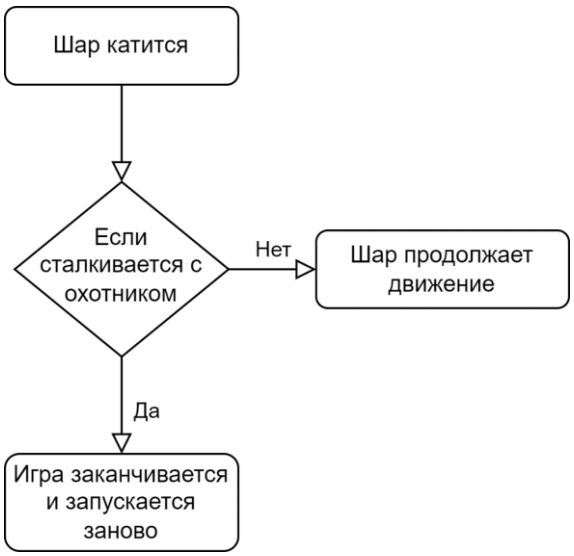

Рис.2 Схема поведения шара при столкновении с охотниками

На рисунке ниже представлены классы, созданные во время решения задачи и связи между ними.

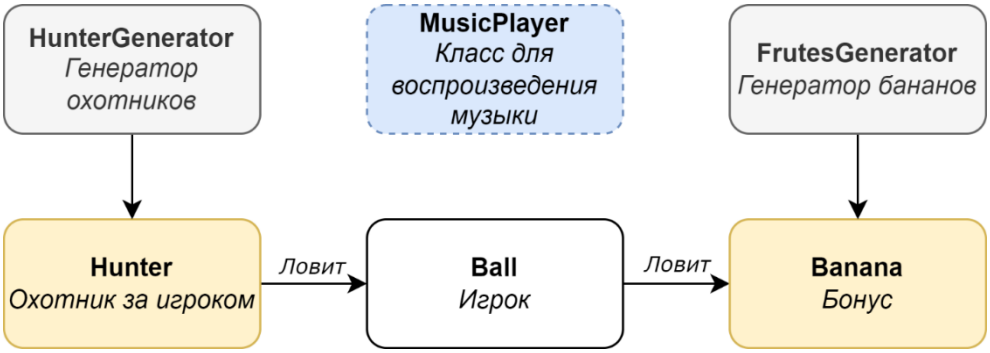

Рис. 3 Связь классов

4.2 Класс Ball

Класс Ball – это основной класс, описывающий поведение игрока на поле. Кроме этого, он содержит общие статические методы, которые могут понадобиться разным объектам (например перезапуск игры, выход из игры по кнопке Escape и т. д.)

Движение шара по игровому полю осуществляет игрок с помощью стрелок на клавиатуре. Шар движется, подчиняясь законам физики (не проходит сквозь стены и землю, движется с ускорением благодаря

приложенной силе).

Такое поведение достигается с помощью компонента Rigidbody, который позволяет объекту подчиняться гравитации и отскакивать от стен. Движение шара осуществляется с помощью метода GetAxis, который возвращает направление движения, заданного пользователем с клавиатуры.

Затем вычисленное направление движения используется для движения, которые выполняется с помощью метода AddForce, который заключается в применении силы к объекту. Сила применяется внутри метода FixedUpdate (т. е. Update, который выполняется с фиксированными промежутками времени в независимости от частоты кадров в игре. Используется именно он, т. к. механизм, отвечающий за физику в Unity, также работает с той же фиксированной частотой).

Кроме того, объект обладает массой и для того, чтобы упростить игровой процесс для пользователей, она была уменьшена со значения по умолчанию 1 до 0,6 кг.

Когда объект Ball сталкивается с другим объектом, то проверяется, является ли данный объект бонусом (для этого все объекты класса Banana помечены тегом «Fruit»). Если действительно игрок столкнулся с бонусом, то объект бонуса уничтожается (сопровождается звуком, подробнее в описании класса MusicPlayer), а переменная с текущим счетом увеличивается на 1.

Чтобы мотивировать игрока играть далее, игра сохраняет самый высокий счет для текущего пользователя (иначе пользователь только проигрывает, и к него нет мотивации играть дальше). Для этого используется механизм PlayerPrefs, который позволяет хранить значения между сессиями игры.

4.3 Класс FruitsGenerator

Класс FruitGenerator создан для генерации бонусов (бананов) и управления ими. С помощью StartCorutine, генератор в бесконечном цикле с фиксированным таймаутом создает копии объекта Banana. Процедура, вызванная с помощью StartCorutine с помощью метода yield return возвращает объект WaitForSeconds, который прекращает работу процедуры на заданный таймаут, а после его истечения – продолжает выполнение с предыдущего места.

Созданные объекты появляются в случайном месте в рамках игрового

поля и им присваивается случайный таймаут до момента исчезновения (в диапазоне от 10 до 20 секунд).

Также, у класса есть публичный метод, чтобы вести подсчет количества объектов Banana, которые не были собраны игроком и исчезли по таймауту. Если исчезло 3 или более бонусов, то вызывается метод для завершения игры.

Объект класса также поддерживает надпись, которая информирует пользователя о текущем количестве исчезнувших бонусов, и меняет её цвет, когда счетчик исчезнувших бананов увеличивается.

4.4 Класс Banana

Класс Banana описывает поведение бонусов. В методе Update указывается вращение бонусов вокруг своей оси, также подсчитывается время, пройденное со времени создания бонуса. Если таймаут задан, и время, пройденное со времени создания бонуса, превышает таймаут, тогда бонус уничтожается, и вызывается метод класса FruitsGenerator для увеличения счетчика бананов и завершения игры, при необходимости. Также, вызывается статический метод класса MusicPlayer для воспроизведения музыки при исчезновении бонуса.

4.5 Класс HuntersGenerator

Появление охотников описано в классе HuntersGenerator. Аналогично с классом FruitsGenerator используется метод StartCorutine, охотники появляются по всей сцене в случайном месте, при этом на достаточном расстоянии от игрока. Охотник движется точно за шаром. Через заданный период он исчезает.

4.6 Класс Hunter

Мы задаём условие, чтобы по сцене могли двигаться только клоны охотника. При соприкосновении с тегом шара игра перезапускается.

4.7 Класс MusicPlayer

Это отдельный класс для звуковых эффектов. Чтобы объект звука не создавался при перезапуске игры заново, мы используем DontDestroyOnLoad и уничтожаем появляющуюся копию объекта при перезапуске игры. Мы создаём методы для каждого файла со звуком и запускаем их в других

классах.

5 ТЕСТИРОВАНИЕ

Для демонстрации работоспособности игры было проведено тестирование.

5.2 Тест №1. Движение шара

Входные данные: игрок нажимает кнопки стрелок.

Ожидаемый результат: шар движется в соответствующем направлении.

Полученный результат: совпадает с ожидаемым. (Рис.4, Рис.5).

Тест пройден успешно.

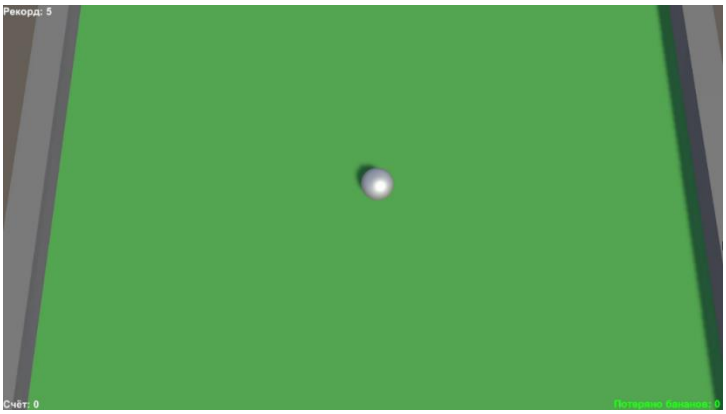

Рис.4 Начальное положение шара

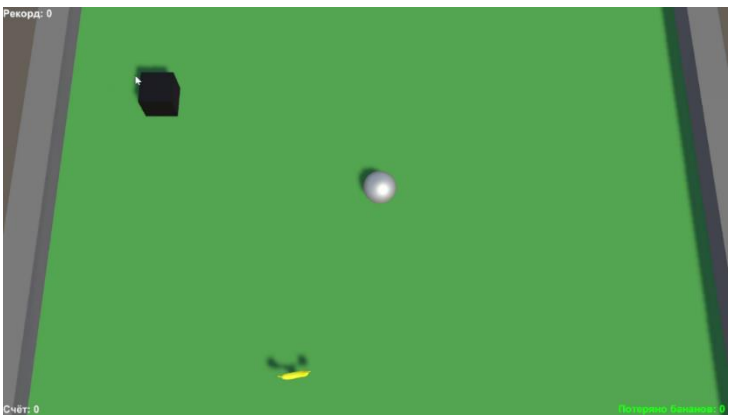

Рис.5 Шар сдвинулся

5.3 Тест №2. Появление и исчезновение бонусов

Входные данные: шар движется по сцене

Ожидаемый результат: появляются бонусы и исчезают через некоторое время. При их исчезновении цвет текста меняется. (Рис.6, Рис.7,Рис.8).

Тест пройден успешно.

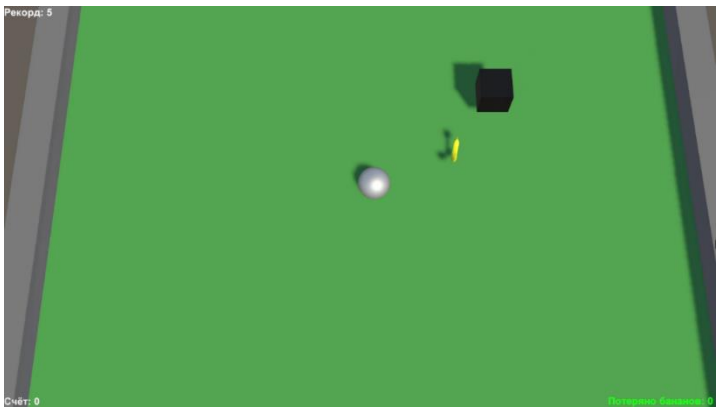

Рис.6 Появление бонуса

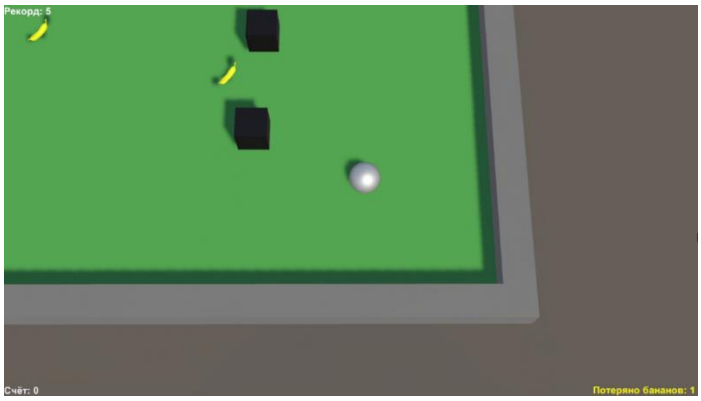

Рис.7 Исчезновение 1 бонуса

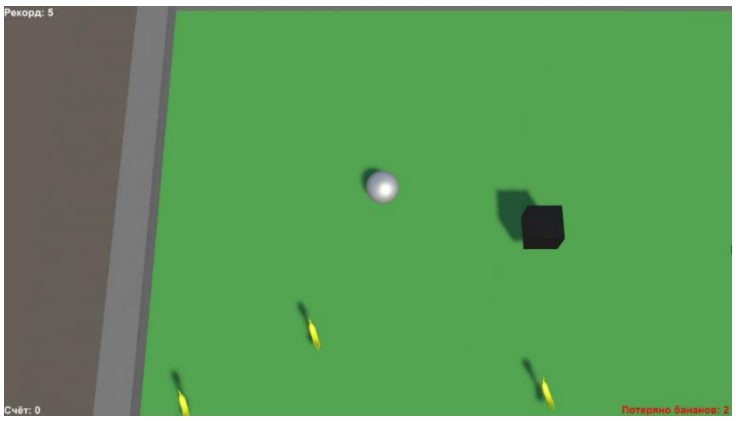

Рис.8 Исчезновение 2 бонусов

5.4 Тест №3. Шар собирает бонусы, за что начисляются баллы

Входные данные: шар движется по сцене

Ожидаемый результат: при столкновении с шаром бонусы исчезают, сразу же начисляются баллы. Исчезновение бонуса сопровождается звуком. В углу мы видим рекорд. (Рис.9, Рис.10).

Тест пройден успешно.

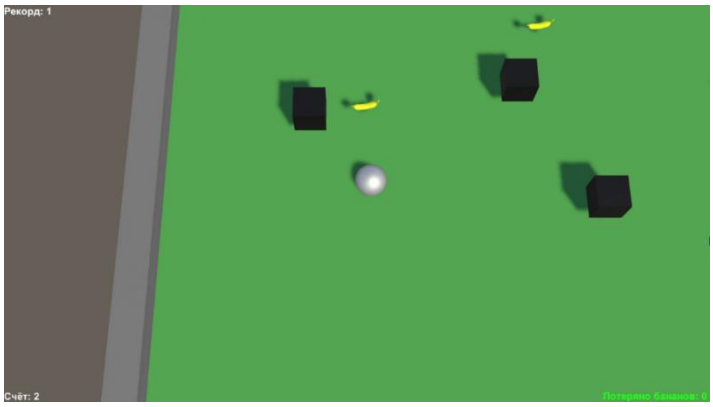

Рис.9 Шар катится в сторону бонуса

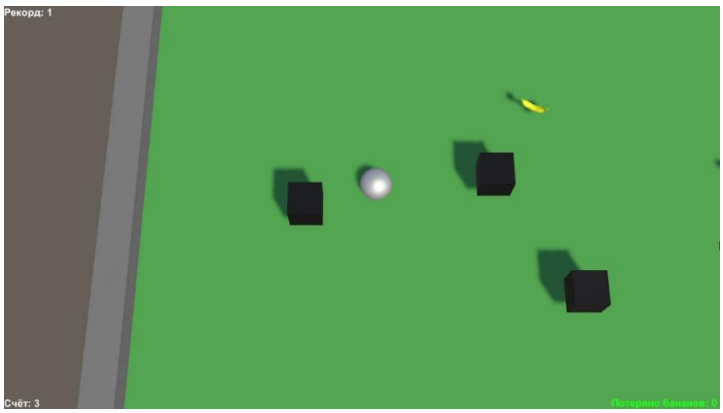

Рис.10 Шар собирает бонус и ему начисляется балл

5.5 Тест №4. Охотник ловит шар, перезапуск игры

Входные данные: шар движется по сцене

Ожидаемый результат: при столкновении с охотником происходит перезапуск игры. (Рис.11, Рис.12).

Тест пройден успешно

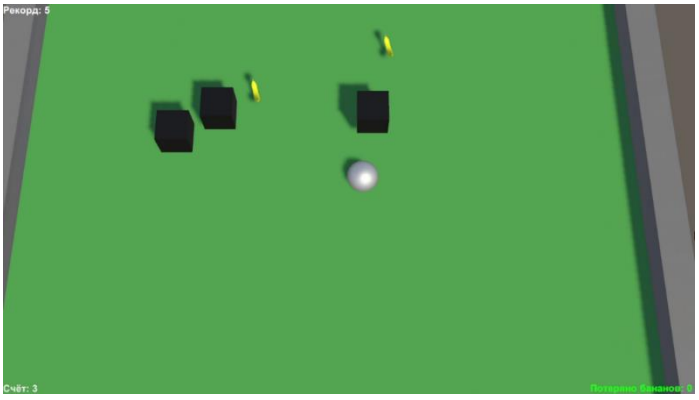

Рис.11 К шару приближается охотник

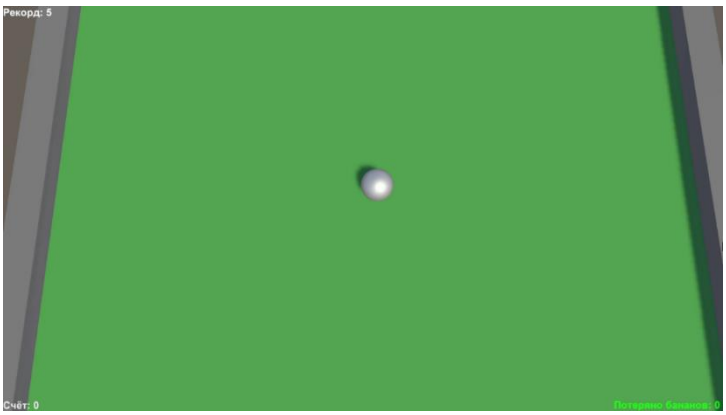

Рис.12 Игра перезапустилась

6 ЗАКЛЮЧЕНИЕ

В результате была разработана игра с помощью Unity, удовлетворяющая поставленным требованиям.

Были изучены возможности Unity, моделирования с его помощью физических объектов, написание и использование классов и кода на языке C#, использование методов библиотек Unity для запуска корутин, воспроизведения музыки, инициализации и уничтожения объектов, движения и позиционирования объектов и проч.

Была проведена сборка и тестирование готового решения, также в тестировании участвовали независимые пользователи, для непредвзятой оценки качества полученной игры.

Кроме того, были выполнены работы по улучшению пользовательского опыта: добавлены веселые звуки при окончании игры, сохранение рекордного счета и цветовая изменения цвета текста об исчезнувших бонусах.

Таким образом, поставленные в курсовой работе цели достигнуты, а задачи выполнены.

7 СПИСОК ЛИТЕРАТУРЫ

https://geeker.ru/games/programmy-dlya-sozdaniya-igr/

https://geekingup.org/ru/%D1%82%D0%BE%D0%BF-10-%D0%BB%D1%83%D1%87%D1%88%D0%B8%D1%85-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC-%D0%B4%D0%BB%D1%8F-%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D1%8F-%D0%B8%D0%B3%D1%80

https://cubiq.ru/luchshie-igrovye-dvizhki/#UNITYhttps://3dgame-creator.ru/catalog/igrovye-dvizhki/obzor-luchshix/

https://cubiq.ru/dvizhok-unity/

https://www.newgenapps.com/ru/blogs/unreal-engine-review-pros-cons-and-suitability/

https://shwanoff.ru/plus-minus-c-sharp/

http://web.spt42.ru/index.php/chto-takoe-unity-3d

https://citrusbits.com/a-unity-review-pros-and-cons/

8 ПРИЛОЖЕНИЕ

Файл Scripts/Ball.cs

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


public class Ball : MonoBehaviour
{
    [SerializeField]
    private Rigidbody _rigidbody;

    public static int score;
    public Text countText, highScoreText;

    [SerializeField]
    [Range(1f, 50f)]
    private float _speed = 1f;

    private void Start()
    {
        score = 0;
        SetCountText();

        highScoreText.text = "Рекорд: " + GetSetHighScore().ToString();
    }

    void Update()
    {
        if (Input.GetKey("escape"))
        {
            PlayerPrefs.SetInt("highScore", 0);
            Application.Quit();
        }
    }

    public void FixedUpdate()
    {
        var horizontal = Input.GetAxis("Horizontal") * _speed;
        var vertical = Input.GetAxis("Vertical") * _speed;

        _rigidbody.AddForce(new Vector3(horizontal, 0f, vertical));
    }

    private void OnTriggerEnter(Collider other)
    {
        if (!other.CompareTag("Fruit"))
            return;
        {
            Destroy(other.gameObject);

            MusicPlayer.PlayPickUp();

            score++;
            SetCountText();
        }
    }
    void SetCountText()
```

```
    {
        countText.text = "Счёт: " + score.ToString();
    }

    private static int GetSetHighScore(int currentScore = 0)
    {
        if (!PlayerPrefs.HasKey("highScore"))
        {
            PlayerPrefs.SetInt("highScore", currentScore);
        } else
        {
            int highScore = PlayerPrefs.GetInt("highScore");
            if (currentScore > highScore)
            {
                PlayerPrefs.SetInt("highScore", currentScore);
                PlayerPrefs.Save();
            }
            currentScore = highScore;
        }
        return currentScore;
    }

    public static void RestartGame(string text="")
    {
        if (text != "")
        {
            Debug.Log(text);
        }
        MusicPlayer.PlayRestart();

        GetSetHighScore(score);

        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
}
```

Файл Scripts/FruitsGenerator.cs

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

public class FruitsGenerator : MonoBehaviour
{
    [SerializeField]
    private Banana _bananaPrefab;

    [SerializeField]
    [Range(1f, 5f)]
    private float _period = 3f;

    public int disappeared = 0;
    public Text disappearedText;

    private void Start()
    {
        StartCoroutine(GenerateEnumerator());
        disappeared = 0;
        SetDissapearedText();
    }
```

```
    private IEnumerator GenerateEnumerator()
    {
        while (true) // т.е. выполняется непрерывно на протяжении игры
        {
            // возвращаем значения, а затем, после заданного таймаута - продолжаем
выполнение с этого же места
            yield return new WaitForSeconds(_period);

            var banana = Instantiate(_bananaPrefab);
            banana.transform.position = new Vector3(Random.Range(-9f, 9f), 0.5f,
Random.Range(-9f, 9f));
            banana.transform.parent = transform;
            banana.timeout = Random.Range(10f, 20f);
            banana.generator = this;
        }
    }
    void SetDissapearedText()
    {
        disappearedText.text = "Потеряно бананов: " + disappeared.ToString();
        if (disappeared == 0)
        {
            disappearedText.color = Color.green;
        } else if (disappeared == 1)
        {
            disappearedText.color = Color.yellow;
        }
        else if (disappeared == 2)
        {
            disappearedText.color = Color.red;
        }
    }

    public void UpdateDissapeared()
    {
        disappeared++;
        SetDissapearedText();
        if (disappeared >= 3)
        {
            Ball.RestartGame("Исчезло 3 банана");
        }
    }

    }
```

Файл Scripts/Banana.cs

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

public class Banana : MonoBehaviour
{
    [SerializeField]
    [Range(0f, 360f)]
    private float _rorateSpeed = 90f;

    private float _time_passed = 0.0f;
    public float timeout = 0;

    public FruitsGenerator generator;
```

```
    private void Update()
    {
        transform.Rotate(0f, _rorateSpeed * Time.deltaTime, 0f);
        _time_passed += Time.deltaTime;

        if ((_time_passed > timeout) && (timeout != 0) && (gameObject != null))
        {
            Destroy(gameObject);
            generator.UpdateDissapeared();
            MusicPlayer.PlayBananaDissapear();
        }
    }

    }
```

Файл Scripts/HuntersGenerator.cs

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

public class HuntersGenerator : MonoBehaviour
{
    [SerializeField]
    private Hunter _hunterPrefab;
    [SerializeField]
    private Transform _target;

    [SerializeField]
    [Range(1f, 5f)]
    private float _period = 3f;

    private void Start()
    {
        StartCoroutine(GenerateEnumerator());
    }

    private float GeneratePosition(float position)
    {
        int multiplier = Random.Range(-1, 1) < 0 ? -1 : 1;
        float result = position + multiplier * Random.Range(3f, 9f);
        if (result < -9f)
        {
            result = -9f;
        }
        else if (result > 9f)
        {
            result = 9f;
        }
        if (Mathf.Abs(result - position) < 1)
        {
            // Если из-за границ поля охотник оказался слишком близко к цели, то убираем
его на другую сторону поля.
            result *= -1;
        }
        return result;
    }

    private IEnumerator GenerateEnumerator()
    {
        while (true)
```

```
        {
            yield return new WaitForSeconds(_period);

            var hunter = Instantiate(_hunterPrefab);
            float x, y, z;
            y = 0.2f;
            x = GeneratePosition(_target.position.x);
            z = GeneratePosition(_target.position.z);

            hunter.transform.position = new Vector3(x, y, z);
            hunter.transform.parent = transform;
            hunter.clone = true;
            Destroy(hunter.gameObject, _period * 3);
        }
    }

}
```

Файл Scripts/Hunter.cs

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

public class Hunter : MonoBehaviour
{
    [SerializeField]
    private Transform _target;

    [SerializeField]
    [Range(1f, 10f)]
    private float _speed = 0.5f;

    public bool clone = false;

    private void Update()
    {
        // Только клонам можно двигаться
        if (clone)
        {
            var pos = Vector3.MoveTowards(transform.position, _target.position, _speed *
Time.deltaTime);
            pos.y = 1f;

            transform.position = pos;
        }
    }
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("ball"))
        {
            Ball.RestartGame("Охотник съел");
        }
    }

}
```

Файл Scripts/CameraControler.cs

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

public class CameraControler : MonoBehaviour
{
    public GameObject player;
    private Vector3 offset;
    void Start()
    {
        offset = transform.position - player.transform.position;
    }

    void LateUpdate()
    {
        transform.position = player.transform.position + offset;
    }
}
```

Файл Scripts/MusicPlayer.cs

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

public class MusicPlayer : MonoBehaviour
{
    static public AudioSource music;
    static public AudioClip PickUp, RestartSound, BananaDissapearSound;
    static MusicPlayer instance = null;

    public void Awake()
    {
        if (instance != null)
        {
            // Мы уже создали один экземпляр объекта и поэтому все остальные - уничтожаем
            Destroy(gameObject);
        }
        else
        {
            // Ещё нет ни одного экземпляра.
            instance = this;
            GameObject.DontDestroyOnLoad(gameObject); // Не уничтожаться при перезапуске
сцены

            music = gameObject.AddComponent<AudioSource>();
            music.playOnAwake = false;

            PickUp = Resources.Load<AudioClip>("music/CollectCoin");
            RestartSound = Resources.Load<AudioClip>("music/RestartSound");
            BananaDissapearSound = Resources.Load<AudioClip>("music/Sound");
        }
    }
    static public void PlayPickUp()
    {
        music.PlayOneShot(PickUp);
    }

    static public void PlayRestart()
    {
```

```
        music.PlayOneShot(RestartSound);
    }

    static public void PlayBananaDissapear()
    {
        music.PlayOneShot(BananaDissapearSound);
    }
}
```