

Создание мобильной игры «Четвертый лишний» на Unity

Голубева Евгения Павловна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Цель данной статьи – создание мобильной игры «Четвертый лишний» для android. Для создания игры была использована кроссплатформенная среда для создания интерактивных 2D- и 3D-приложений Unity, которая позволяет создавать 2D-игры, используя мощный физический движок и визуальные инструменты. В результате работы была создана мобильная игра «Четвертый лишний»

Ключевые слова: Unity, игра, лишний, предметы.

Creating a "Fourth-Person" Mobile Game on Unity

Golubeva Evgeniya Pavlovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The purpose of this article is to create a mobile game "the fourth extra" for Android. To create the game, we used the cross-platform environment for creating interactive 2D and 3D applications Unity, which allows you to create 2D games using a powerful physics engine and visual tools. As a result, we created a mobile game called "the fourth extra".

Keywords: Unity, game, extra, objects.

1 Введение

1.1 Актуальность

В современной цифровой среде мобильные игры прочно вошли в повседневную жизнь, представляя собой не только массовый вид развлечения, но и динамично развивающуюся многомиллиардную индустрию. Особой популярностью пользуются казуальные головоломки, которые сочетают интуитивно понятный интерфейс, минималистичный дизайн и геймплей, способствующий когнитивному развитию.

Разработка логической головоломки «Четвертый лишний», основанной на механике поиска и классификации объектов, соответствует актуальным трендам игрового рынка. Подобные игры востребованы широкой аудиторией, так как способствуют развитию аналитического и ассоциативного мышления, внимания и скорости реакции. Их ценность признана не только в сфере

развлечений, но и в образовательном контексте, где они могут использоваться в качестве тренажеров для разных возрастных групп.

Использование игрового движка Unity для реализации данного проекта является современным и практичным решением. Unity предоставляет мощный, гибкий и доступный инструментарий для быстрой разработки кроссплатформенных приложений. Это позволяет сосредоточиться на геймдизайне и содержании игры, обеспечивая при этом ее высокую производительность и простую публикацию во всех основных магазинах приложений.

Таким образом, создание мобильной игры «Четвертый лишний» на Unity представляет собой актуальную задачу, находящуюся на пересечении востребованного игрового жанра, образовательного потенциала и эффективного применения современных технологий разработки. Проект обладает четкой практической значимостью и соответствует запросам рынка на интеллектуальный и развивающий контент.

1.2 Обзор исследований

А. Е. Унарова представила разработку мобильной игры «Кустук» на Unity для дошкольников и младших школьников на якутском языке [1]. Посвятил статью обзору ключевых этапов, необходимых для разработки компьютерных игр с двухмерной графикой на платформе Unity в жанре приключения с элементами головоломок Я. А. Фаст [2]. Н.В.Максимов рассматривает метод разработки мобильной 2D игры «Bootur Jump» на платформе Unity [3]. Показал процесс разработки двухмерной платформенной игры на игровом движке Unity с использованием языка программирования C# Любич А.А. [4]. Д.В. Морозова в рамках работы разработала развивающую 2D игра-платформер для детей, написанная на Unity [5].

1.3 Цель исследования

Цель исследования – создание мобильной игры «Четвертый лишний» для ОС Android при помощи кроссплатформенной среды для создания интерактивных 2D и 3D приложений Unity.

2 Материалы и методы

Для разработки игры «Четвертый лишний» был выбран игровой движок Unity. Это решение обусловлено его доступностью для разработчиков разного уровня, понятным интерфейсом и поддержкой языка программирования C#, который хорошо подходит для реализации игровой логики. Работа над кодом велась в интегрированной среде разработки Visual Studio Community.

3 Результаты

Разработка мобильной игры «Четвертый лишний» начинается с создания проекта. Первым шагом в Unity Hub является выбор шаблона Universal Render Pipeline (2D), предназначенного для двумерной графики, указание имени проекта и его создание (см. рис. 1).

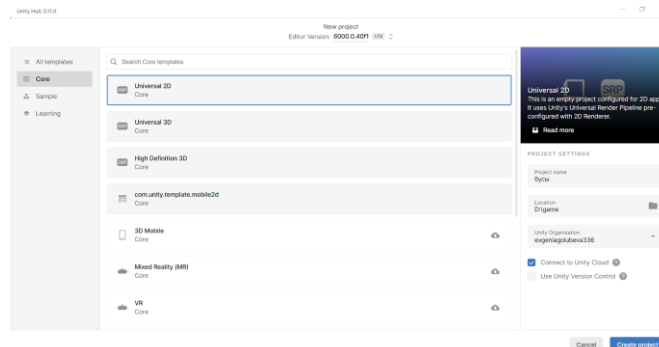


Рисунок 1 – Создание проекта

После инициализации проекта на базе шаблона Universal Render Pipeline (2D) требуется выполнить его базовую настройку для мобильных устройств. Первым шагом необходимо в меню File → Build Settings переключить целевую платформу на Android (см. рис. 2).

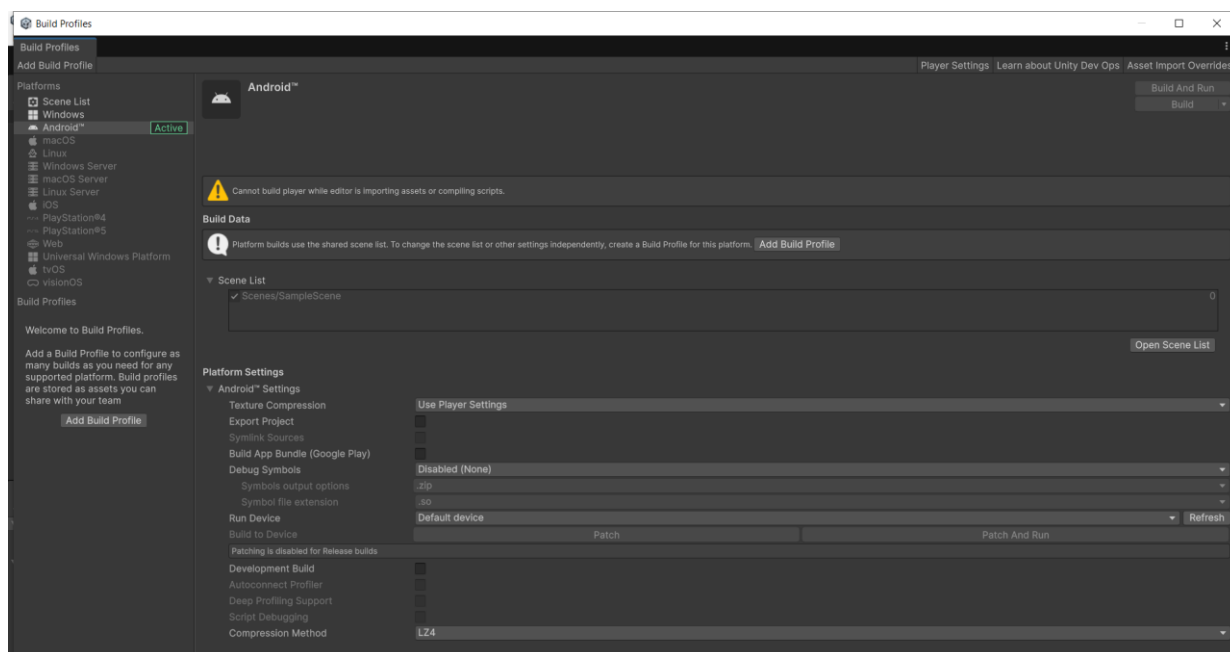


Рисунок 2 – Настройка проекта

Следующим шагом является создание основной игровой сцены, и начинается он с настройки камеры (см. рис. 3).

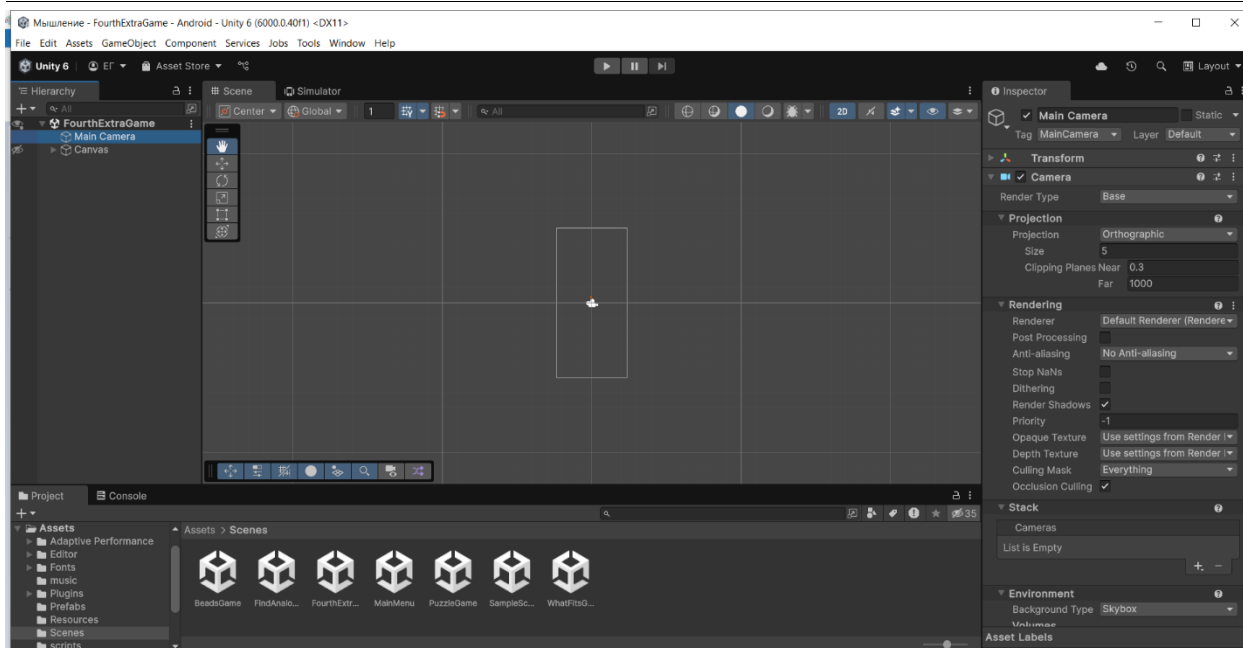


Рисунок 3 – Настройка камеры

В папку Sprites необходимо добавить заранее подготовленные изображения (см.рис.4).

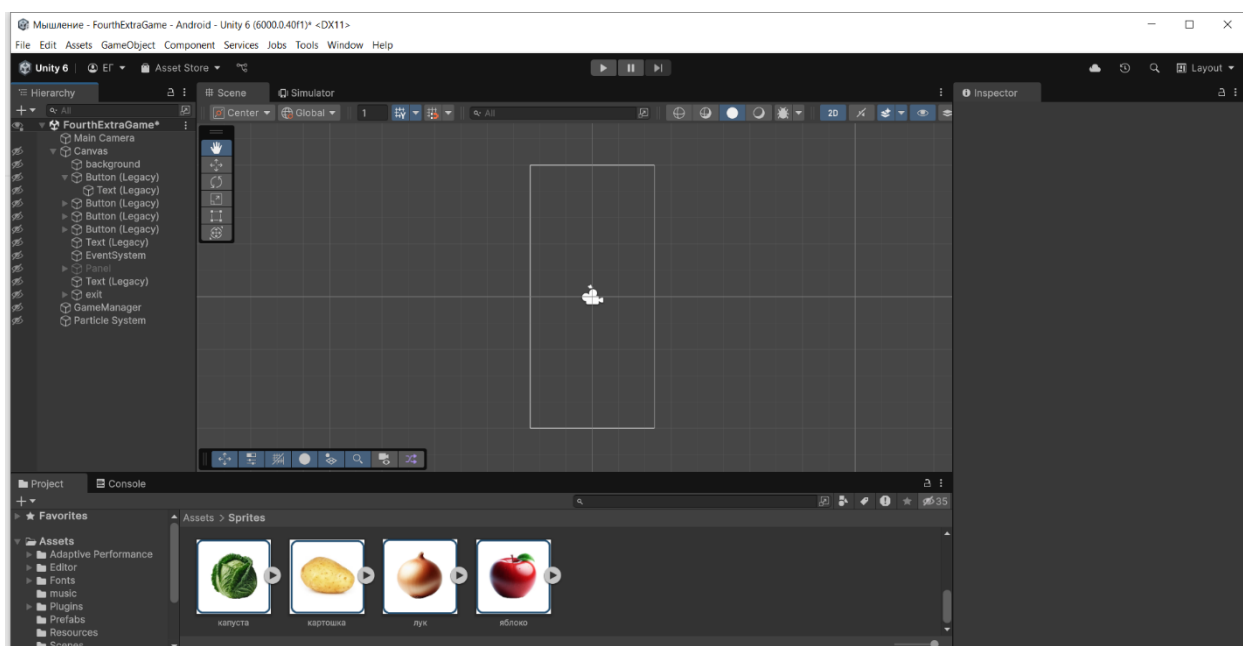


Рисунок 4 – Добавление изображений

Далее необходимо подготовить фон для игровой сцены. Для этого нужно перенести изображение background из папки Sprites на сцену и настроить его параметры так, чтобы оно полностью покрывало игровую область (см.рис.5).

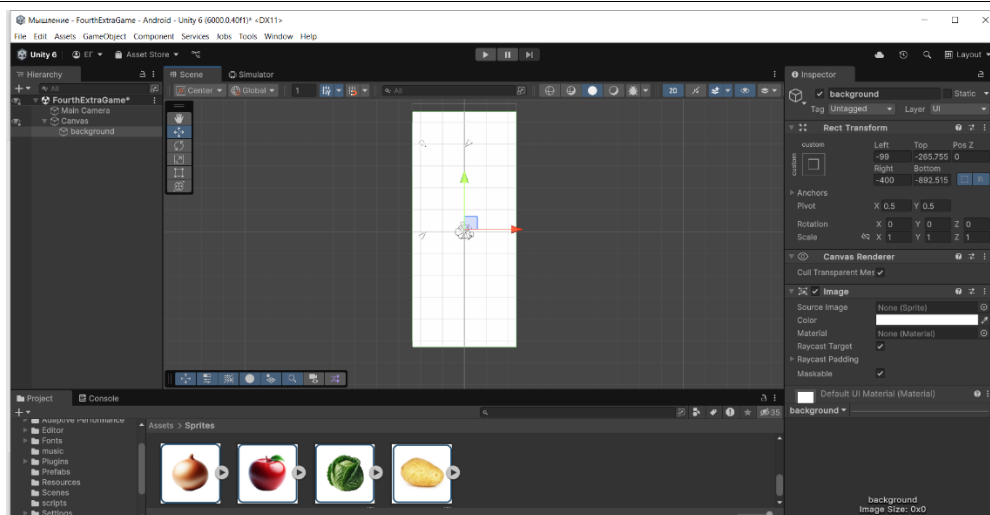


Рисунок 5 – Настройка фона игры

Следующим шагом добавляем компонент Button для того, чтобы реализовать взаимодействие с предметами, позволяя игроку кликнуть на выбранный элемент и получить обратную связь. Для этого добавляем 4 компонента Button в Hierarchy и настраиваем параметры. Каждая карточка (Card_1 – Card_4) настраивается как полноценная интерактивная кнопка. Для этого в её компонент Button в поле Source Image назначается текущий спрайт предмета, чтобы визуальная реакция применялась именно к нему. Тип перехода (Transition) устанавливается в режим Color Tint, что позволяет задавать различные цвета для состояний кнопки: обычного (Normal Color), при наведении (Highlighted Color), в момент нажатия (Pressed Color), а также для выбранного или заблокированного состояния. Длительность перехода между цветами (Fade Duration) задаётся 0.1 секунда, чтобы анимация была плавной и отзывчивой. Навигация (Navigation) оставляется автоматической для корректной работы с альтернативными устройствами ввода. Таким образом, каждая карточка становится интерактивным элементом, который визуально реагирует на действия игрока (см.рис.6).

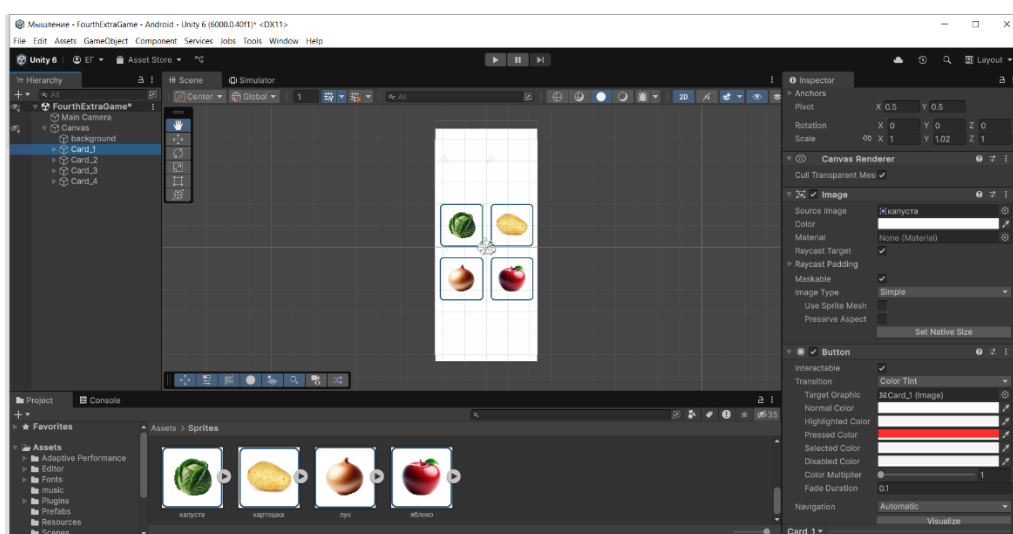


Рисунок 6 – Добавление компонентов Button сцену

Для реализации отображения инструкций и системных сообщений, таких как «Правильно!», в интерфейс игры необходимо добавить и настроить соответствующие элементы пользовательского интерфейса. Это выполняется путём создания в иерархии сцены через контекстное меню панели (UI -> Panel), которая будет служить фоновой основой, и текстового блока (UI -> Text) для непосредственного вывода информации. После добавления ключевым этапом является их детальная настройка в окне Inspector: в первую очередь для панели устанавливаются якоря (Anchors) в центральное положение (0.5, 0.5), что гарантирует корректное адаптивное позиционирование и масштабирование на устройствах с разным разрешением экрана; затем задаются конкретные размеры области Width 1049 и Height 481 пикселей. Завершающим шагом является визуальное оформление: в компоненте Image панели выбирается подходящий спрайт в поле Source Image, а с помощью параметра Color настраиваются его оттенок и прозрачность для гармоничного соответствия общему стилю игры, после чего тип отображения устанавливается в значение Simple (см.рис.7).

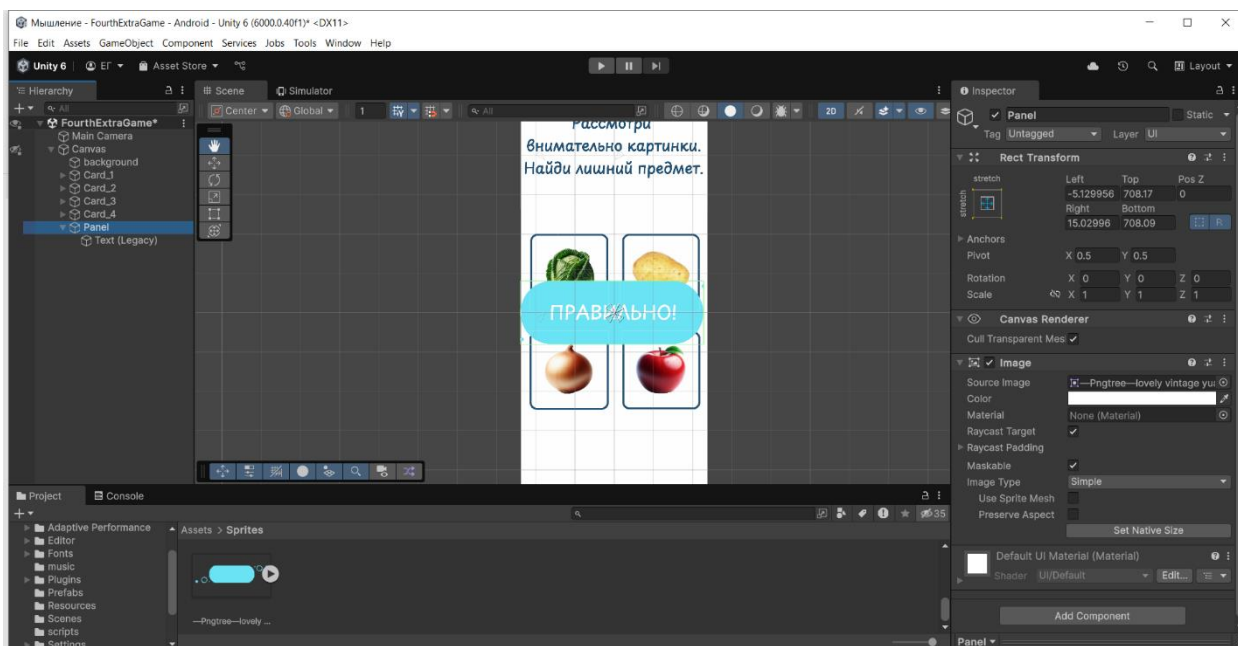


Рисунок 7 – Настройка панели

Также необходимо настроить текстовый блок с уведомлением "Правильно!". В компоненте Text (Legacy) выбираем шрифт PlaypenSans-VariableFont_wght с жирным начертанием Bold и устанавливаем размер шрифта 100 пунктов для обеспечения хорошей читаемости. В разделе Rect Transform задаем точные размеры текстового поля: Width 726,9, Height 243,82. Особое внимание уделяется настройке якорей (Anchors) с центром в точке (0.5, 0.5), что гарантирует правильное отображение текста на экранах с разным разрешением. (см.рис. 8).

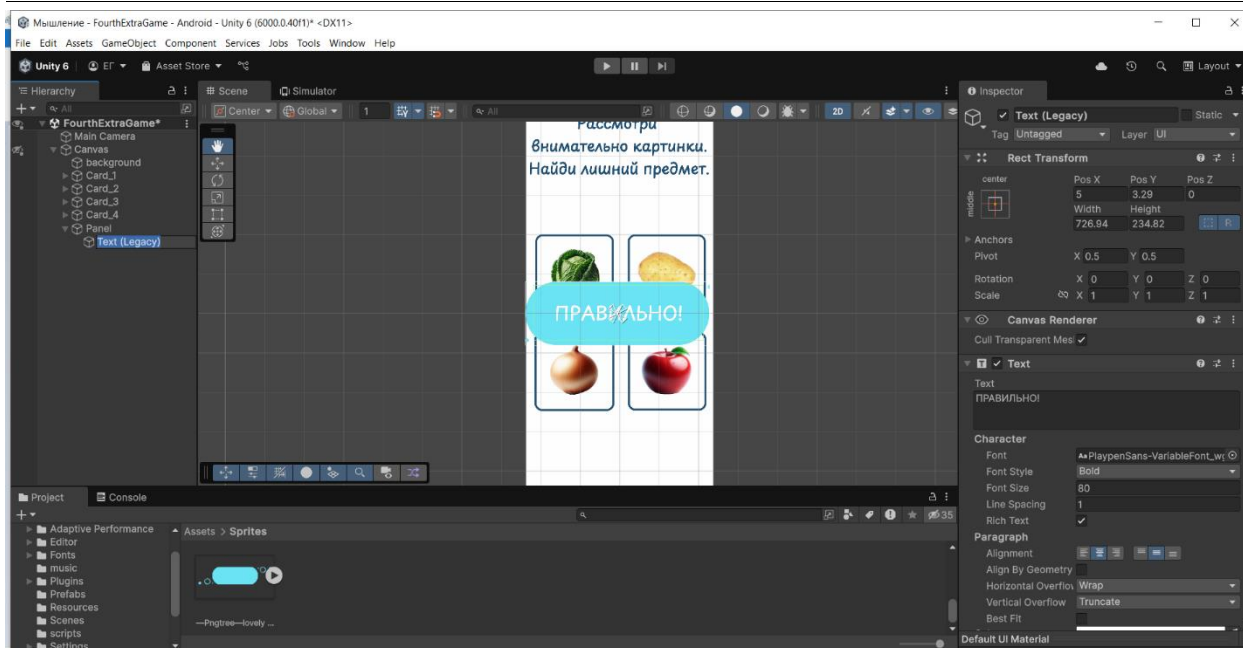


Рисунок 8 – Настройка текстового элемента панели сообщений

Для текстового элемента настраиваем шрифт Pangolin-Regular с жирным начертанием и размером 65 пунктов с жирным начертанием Bold, выравнивание по центру, включаем поддержку Rich Text для форматирования, а в Rect Transform задаем точные размеры текстового поля: Width 1039, Height 399,3 и устанавливаем якоря Anchors с центром в точке (0.5, 0.5) (см.рис.9).

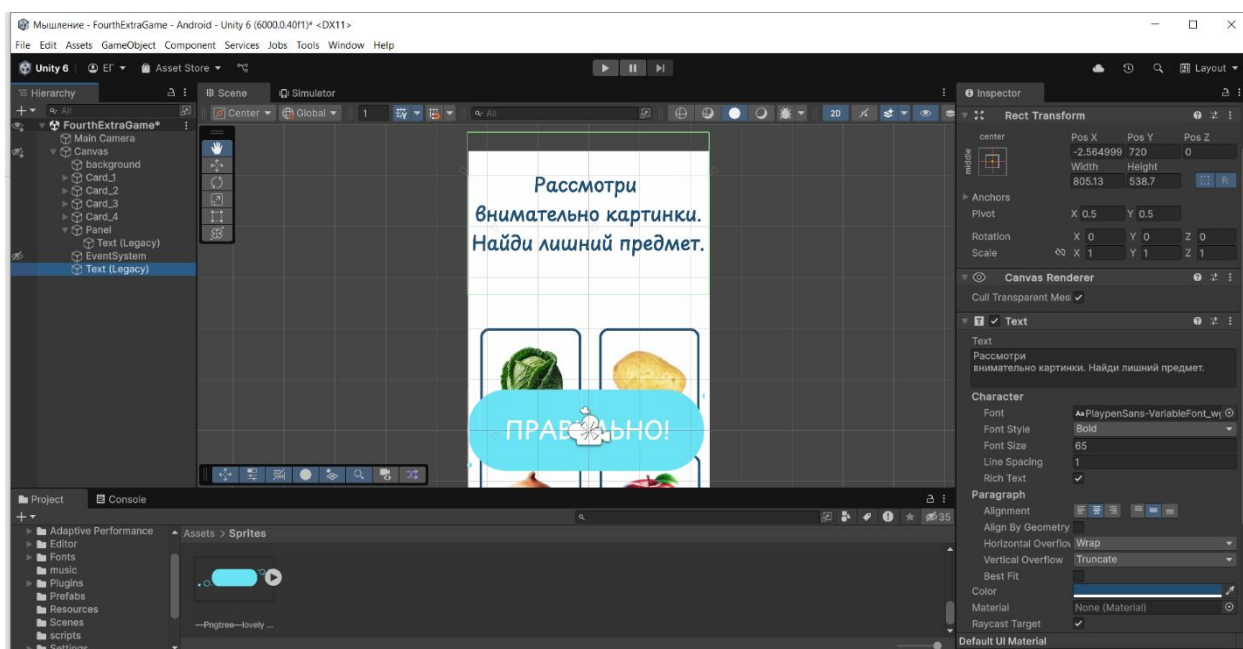


Рисунок 9 – Настройка инструкции игры

После добавления всех необходимых компонентов необходимо создать скрипт под названием GameManager. Скрипт GameManager.cs является центральным управляющим компонентом игры, ответственным за координацию всей игровой логики, взаимодействие с интерфейсом и обратную связь с пользователем.

В начале скрипта располагается блок объявления и настройки всех необходимых переменных. Для удобства работы в редакторе Unity они сгруппированы с помощью атрибутов Header. В разделе Game Elements объявлены массив кнопок `imageButtons`, представляющих собой игровые карточки, и переменная `correctIndex`, хранящая индекс правильного ответа. Раздел Notification Panel содержит ссылки на объекты панели уведомления (`notificationPanel`), текстового поля (`messageText`) и его фона (`panelBackground`), а также параметры для настройки цвета сообщений, времени их отображения и размера шрифта. В Audio Settings находятся ссылки на компонент `AudioSource` и аудиоклипы для различных игровых событий: звука клика, правильного и неправильного ответа.

Основная инициализация происходит в методе `Start ()`. При старте игры панель уведомлений скрывается, для текстового поля устанавливается заданный размер шрифта. Также выполняется проверка наличия компонента `AudioSource` на объекте, и он добавляется при необходимости. Важнейшей частью инициализации является динамическое назначение обработчиков события `onClick` для каждой кнопки в массиве. Это гарантирует, что при нажатии на любую карточку будет вызван метод `OnImageClick` с передачей её индекса в качестве аргумента.

Ядром игровой механики является метод `OnImageClick(int index)`. Он обрабатывает выбор игрока: сначала воспроизводится звук клика, а затем индекс нажатой кнопки сравнивается со значением `correctIndex`. В случае совпадения запускается звук победы, отображается панель уведомления с сообщением «Правильно!» и зелёным фоном, а через заданную задержку автоматически вызывается метод завершения раунда. Если ответ неверный, воспроизводится звук ошибки, а на экране появляется красная панель с соответствующим сообщением, которая автоматически скрывается через несколько секунд.

Управление визуальной обратной связью осуществляется методами `ShowNotification` и `HideNotification`. Метод `ShowNotification` активирует панель, устанавливает переданный текст, задаёт цвет фона в зависимости от успешности ответа и управляет логикой её автоматического скрытия для неправильных ответов. Метод `HideNotification` просто деактивирует объект панели.

Для работы со звуком создан вспомогательный метод `PlaySound(AudioClip clip)`. Он обеспечивает безопасное воспроизведение аудиоклипов, проверяя наличие компонента `AudioSource` и самой звуковой дорожки перед вызовом `PlayOneShot`. Это централизует управление аудио и предотвращает возможные ошибки.

Завершающим блоком логики является метод `EndGame()`. Он отвечает за финальное действие после правильного ответа — переход на сцену с главным меню с помощью встроенного менеджера сцен `Unity SceneManager.LoadScene`.

Таким образом, скрипт `GameManager` выполняет роль дирижёра, который интегрирует пользовательский ввод, игровые правила, визуальный

интерфейс и звуковое сопровождение в единый, последовательный и отзывчивый игровой процесс.

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class GameManager : MonoBehaviour
{
    [Header("Game Elements")]
    public Button[] imageButtons; // Массив кнопок с изображениями
    public int correctIndex = 0; // Индекс правильного ответа

    [Header("Notification Panel")]
    public GameObject notificationPanel; // Панель уведомления
    public Text messageText; // Текст внутри панели
    public Image panelBackground; // Фон панели
    public Color successColor = Color.green; // Цвет для успеха
    public Color failColor = Color.red; // Цвет для ошибки
    public float panelShowTime = 2f; // Время отображения панели
    public int fontSize = 24; // Размер шрифта для уведомлений

    [Header("Audio Settings")]
    public AudioSource audioSource; // Компонент AudioSource для
    // воспроизведения звуков
    public AudioClip buttonClickSound; // Звук при нажатии на любую иконку
    public AudioClip winSound; // Звук при правильном ответе
    public AudioClip wrongAnswerSound; // Звук при неправильном ответе

    void Start()
    {
        // Инициализация панели
        notificationPanel.SetActive(false);

        // Устанавливаем размер шрифта
        if (messageText != null)
        {
            messageText.fontSize = fontSize;
        }

        // Проверяем наличие AudioSource
        if (audioSource == null)
        {
            audioSource = gameObject.AddComponent<AudioSource>();
        }

        // Назначаем обработчики кликов для всех кнопок
        for (int i = 0; i < imageButtons.Length; i++)
        {
            int index = i;
            imageButtons[i].onClick.AddListener(() => OnImageClick(index));
        }
    }

    void OnImageClick(int index)
    {
        // Воспроизводим звук нажатия на кнопку
        PlaySound(buttonClickSound);

        if (index == correctIndex)
        {
            // Воспроизводим звук победы
        }
    }
}
```

```

        PlaySound(winSound);
        ShowNotification("Правильно!", true);
        Invoke("EndGame", panelShowTime);
    }
    else
    {
        // Воспроизводим звук неправильного ответа
        PlaySound(wrongAnswerSound);
        ShowNotification("Неправильно, попробуй еще!", false);
    }
}

void ShowNotification(string message, bool isSuccess)
{
    // Настраиваем панель
    messageText.text = message;
    panelBackground.color = isSuccess ? successColor : failColor;

    // Устанавливаем размер шрифта (на случай если он изменился)
    messageText.fontSize = fontSize;

    // Показываем панель
    notificationPanel.SetActive(true);

    // Скрываем панель через время, если это не победа
    if (!isSuccess)
    {
        Invoke("HideNotification", panelShowTime);
    }
}

void HideNotification()
{
    notificationPanel.SetActive(false);
}

void EndGame()
{
    SceneManager.LoadScene("MainMenu");
}

void PlaySound(AudioClip clip)
{
    if (audioSource != null && clip != null)
    {
        audioSource.PlayOneShot(clip);
    }
}
}

```

Далее необходимо создать пустой объект для скрипта. Для этого в Hierarchy нажимаем правой кнопкой мыши, и в контекстном меню выбираем Create empty и переименовываем в GameManager. Скрипт GameManager добавляем к созданному компоненту, и настраиваем параметры игры. Настройка включает заполнение всех полей, объявленных в скрипте, чтобы обеспечить корректную работу игровой механики, интерфейса и звукового сопровождения.

В первую очередь необходимо связать логические элементы игры с объектами на сцене. В поле Image Buttons следует добавить четыре элемента

(Element 0 – Element 3), перетаскив в них из иерархии объекты Card_1, Card_2, Card_3 и Card_4 соответственно. Это свяжет массив кнопок в скрипте с реальными интерактивными карточками. Далее в поле Correct Index указывается индекс правильного ответа, который будет сравниваться с выбором игрока.

Для работы системы обратной связи необходимо настроить элементы интерфейса. В поле Notification Panel перетаскивается объект Panel из иерархии. В поле Message Text назначается текстовый объект Text (Legacy), который будет отображать сообщения. В поле Panel Background указывается компонент Image самой панели, чтобы скрипт мог менять её цвет. Далее задаются визуальные параметры: Success Color зелёный и Fail Color красный для цветовой индикации результата. Panel Show Time определяет длительность показа сообщения (например, 2 секунды), а Font Size — размер шрифта в текстовом блоке.

Для добавления звукового сопровождения в поле Audio Source необходимо назначить компонент AudioSource, который можно добавить на объект GameManager через кнопку Add Component. Затем в соответствующие поля перетаскиваются аудиофайлы: Button Click Sound (звук клика), Win Sound (звук победы) и Wrong Answer Sound (звук ошибки) из папки проекта. Это обеспечит воспроизведение звуков при взаимодействии игрока с карточками.

После заполнения всех полей игра готова к тестированию. Правильная настройка параметров гарантирует, что выбор карточки будет обрабатываться корректно, игрок получит визуальную и звуковую обратную связь, а игра перейдёт на следующий уровень или в главное меню по завершению раунда (см. рис. 10).

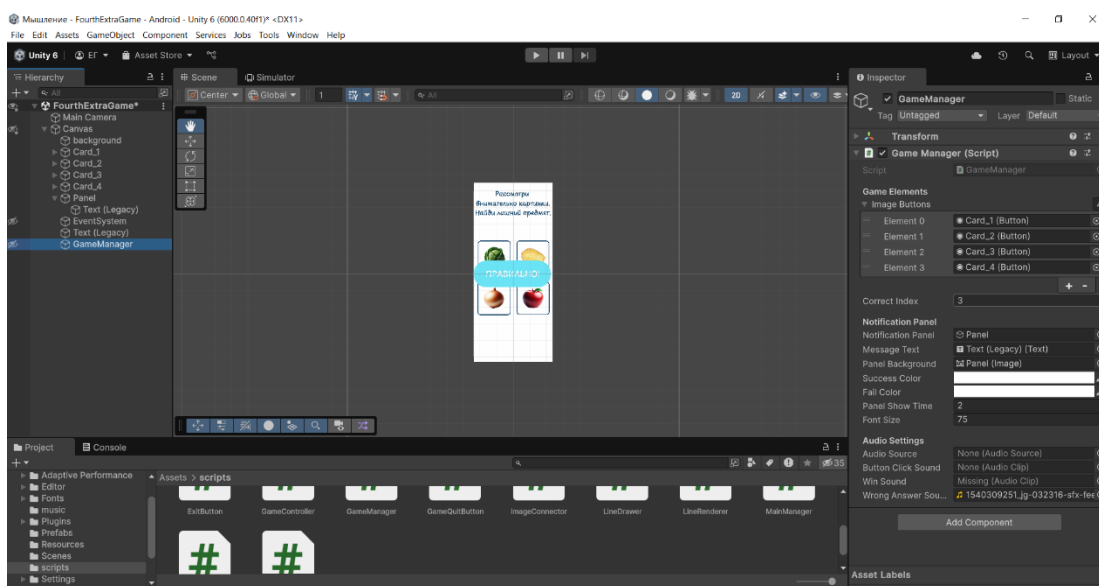


Рисунок 10 – Добавление скрипта к компоненту GameManager и настройка компонента

Итоговый результат создания игры «Четвертый лишний» (см. рис.11).

Рассмотри
внимательно картинки.
Найди лишний предмет.



Рисунок 11 – Готовая игра «Четвертый лишний»

4. Выводы

В данной работе был показан процесс создания игры «Четвертый лишний» с помощью кроссплатформенной среды для создания интерактивных 2D и 3D приложений. Статья может служить методическим пособием в учебной деятельности.

Библиографический список

1. Унарова А. Е. Разработка игры на платформе Unity // Технические и естественные науки: вчера, сегодня, завтра: Сборник научных статей. Волгоград: Общество с ограниченной ответственностью "Сфера", 2024. С. 22-25.
2. Фаст Я. А. Разработка 2D игр с элементами головоломок на платформе Unity // V Международная научная конференция по междисциплинарным исследованиям: сборник статей V Международной научной конференции по междисциплинарным исследованиям, Екатеринбург, 15 апреля 2024 года. Екатеринбург: Общество с ограниченной ответственностью "Институт Цифровой Экономики и Права", 2024. С. 266-269.
3. Максимов Н. В. Разработка мобильной 2D игры "Bootur Jump" на Unity // Фундаментальные и прикладные аспекты развития современной науки: Сборник трудов по материалам XV Международного конкурса научно-исследовательских работ, Уфа, 04 декабря 2023 года. Уфа: Общество с

ограниченной ответственностью "Научно-издательский центр "Вестник науки", 2023. С. 306-308.

4. Любич А. А. Создание 2D-игры «б-у-у-м» на движке Unity // Фундаментальные и прикладные исследования молодых ученых: Сборник научных трудов XV Всероссийской научно-практической конференции студентов, аспирантов и молодых учёных, Новокузнецк, 07–25 апреля 2025 года. – Новокузнецк: Кемеровский государственный университет, 2025. С. 167-169.