

Создание компьютерной игры «Продуктовый магазин» на Unity

Голубева Евгения Павловна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Цель данной статьи – создание компьютерной игры «Продуктовый магазин» на Unity. 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 компьютерная игра «Продуктовый магазин».

Ключевые слова: Unity, игра, продукты, магазин.

Creating a "Grocery Store" computer game using Unity

Golubeva Evgeniya Pavlovna

Sholom-Aleichem Priamursky State University

Student

Abstract

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

Keywords: Unity, game, products, store.

1 Введение

1.1 Актуальность

Современное образование сталкивается с вызовом «цифрового поколения»: обучающиеся, выросшие в медиа-насыщенной среде, требуют новых интерактивных форматов подачи материала, где визуальная составляющая и вовлечённость доминируют над статичными текстами. В этих условиях традиционные педагогические подходы уступают место игровым технологиям. Видеоигры из нишевого развлечения превратились в мощный культурный феномен и серьёзный образовательный ресурс. Как отмечается в современных исследованиях, мини-игры, изначально воспринимаемые как средство развлечения, способны выполнять полноценные образовательные функции. Именно обучение на основе игр обладает наибольшим дидактическим потенциалом, поскольку помещает ученика в центр событий и создаёт мощную эмоциональную вовлечённость.

Симуляторы повседневных профессиональных и социальных практик представляют собой особо перспективное направление образовательных игр. Видеоигровой симулятор магазина позволяет игрокам погрузиться в виртуальный мир работы в розничной торговле, развивая навыки управления бизнесом, клиентского обслуживания и маркетинга, а также повышая понимание соответствующих процессов и стратегий. Такие симуляции находят применение не только в профессиональной подготовке, но и в формировании у детей финансовой грамотности, навыков классификации продуктов, здорового образа жизни и коммуникативных компетенций.

Выбор игрового движка Unity в качестве инструмента разработки имеет самостоятельное научно-практическое обоснование. Unity остаётся доминирующей платформой среди разработчиков независимых и средних студий благодаря своей универсальности, простоте использования и кроссплатформенности. Как подчёркивается в исследовательских работах, применение Unity для создания образовательных игр позволяет расширить возможности восприятия и запоминания материала, а сам движок обладает рядом преимуществ перед другими мультиплатформенными средами разработки. Современные тенденции в разработке на Unity - переход к более лёгким и быстрым проектам, активное внедрение AI-инструментов и автоматизации - делают эту платформу особенно привлекательной для образовательных и экспериментальных проектов.

Дополнительным значимым аспектом является необходимость стимулирования критического мышления, повышения креативности и вовлечённости обучающихся, а также усиления мотивации к познанию в условиях активной цифровизации образовательного пространства. Создание компьютерной игры «Продуктовый магазин» на Unity представляет собой практический вклад в решение этой задачи, предлагая готовый инструмент, который может быть адаптирован для различных образовательных целей - от дошкольного обучения до профессиональной ориентации и формирования цифровых компетенций.

Таким образом, исследование, направленное на разработку компьютерной игры-симулятора «Продуктовый магазин» на платформе Unity, отвечает на актуальный запрос современной педагогики и цифровой дидактики, сочетая в себе образовательный потенциал игровых технологий, практическую значимость симуляционного моделирования и техническую обоснованность выбранного инструментария.

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]. Максимов Н.В. 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 компьютерной игры «Продуктовый магазин» при помощи кроссплатформенной среды для создания интерактивных 2D и 3D приложений Unity.

2 Материалы и методы

В качестве основного инструмента для разработки игры «Продуктовый магазин» выбран Unity. Это решение обусловлено его доступностью для новичков, понятным интерфейсом и встроенной поддержкой языка C#. Для работы со скриптами использовалась среда разработки Visual Studio Community.

3 Результаты

Первый шаг в разработке игры - создание нового проекта в Unity. В Unity Hub нужно выбрать шаблон «Universal 2D» (рендер-пайплайн) и нажать кнопку «Создать проект» (см. рис. 1).

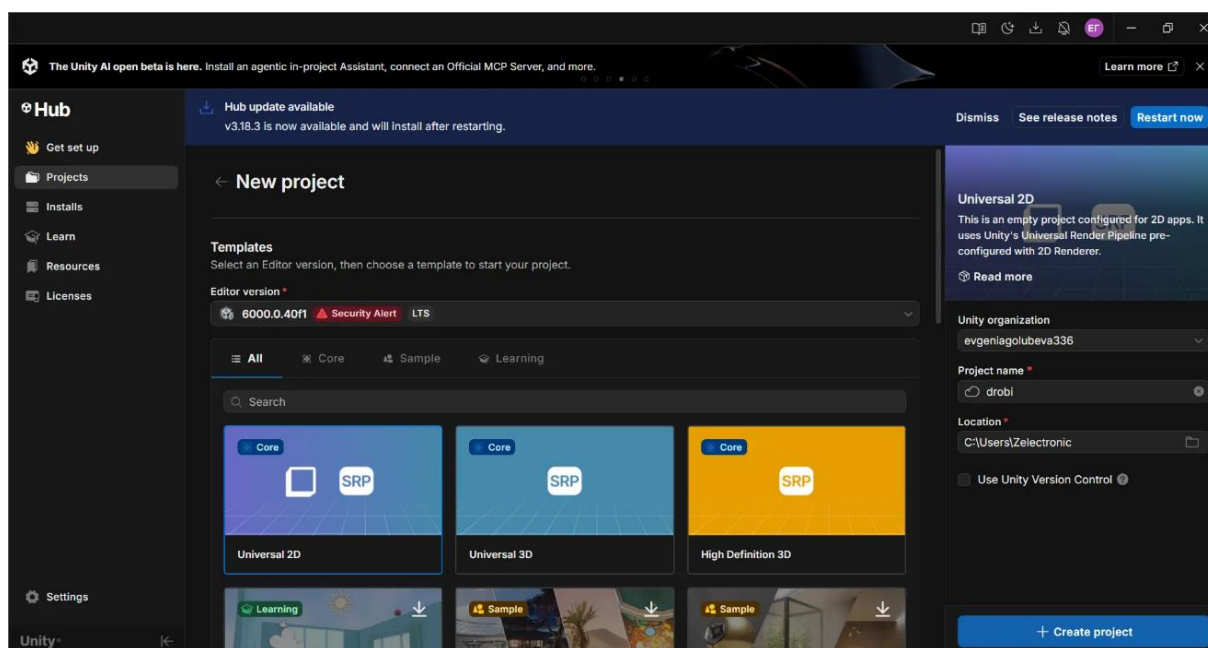


Рисунок 1 – Создание проекта

После того как проект на базе Universal 2D создан, необходимо выполнить его начальную настройку для целевой платформы - ПК. Первым делом в меню File -> Build Profiles нужно сменить целевую платформу на Windows (см. рис. 2).

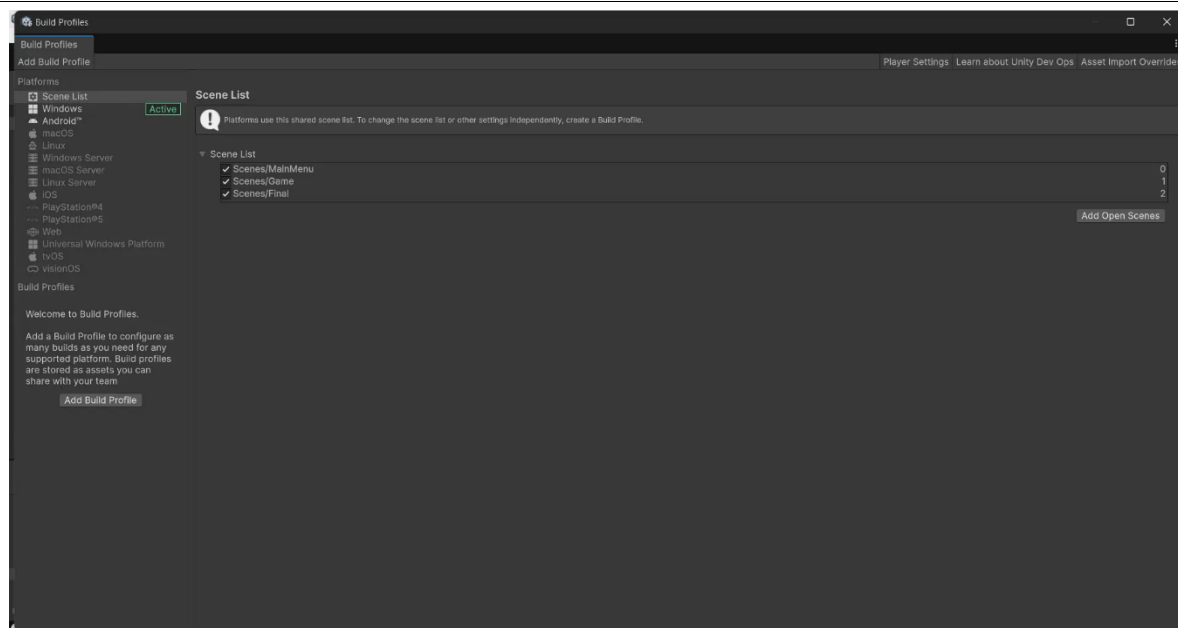


Рисунок 2 – Настройка проекта

Следующий шаг - подготовка трех сцен. Так как при создании проекта одна сцена уже генерируется автоматически, нужно добавить лишь две сцены. Для этого в папке Scenes вызывается контекстное меню правой кнопкой мыши, Create -> Scene и создаём недостающие сцены. После этого три сцены необходимо переименовать: существующую называем MainMenu, вторую Game и третью Final (см. рис. 3).

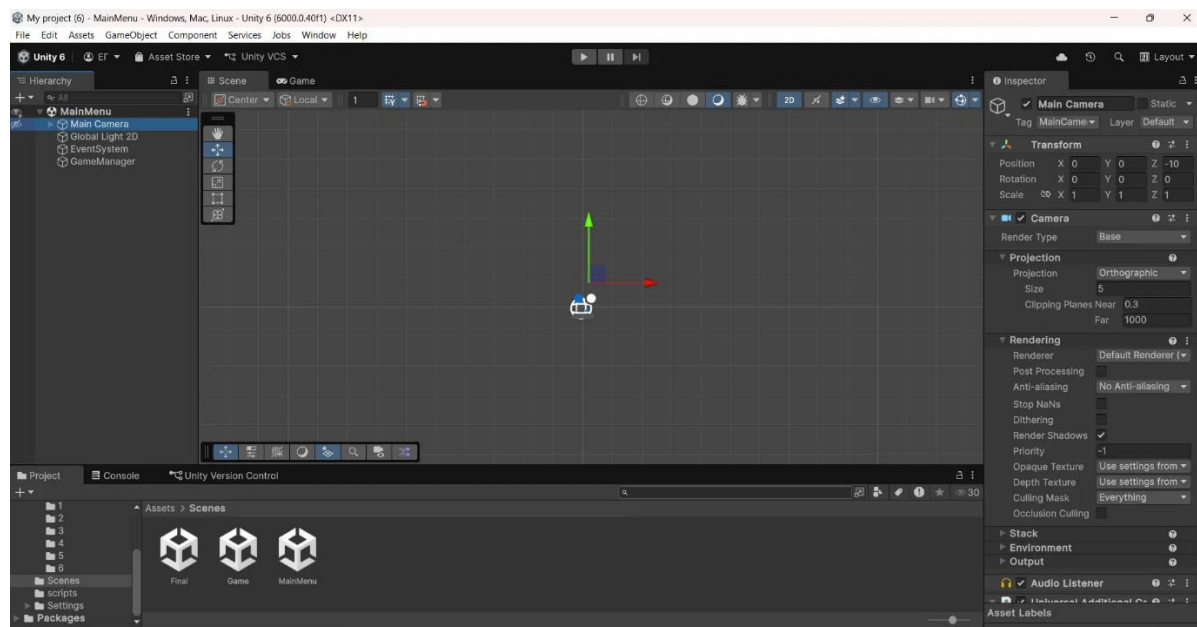


Рисунок 3 – Создание сцен игры

В папку «image» следует поместить предварительно подготовленные графические ресурсы, которые будут использоваться в игре (см. рис. 4).

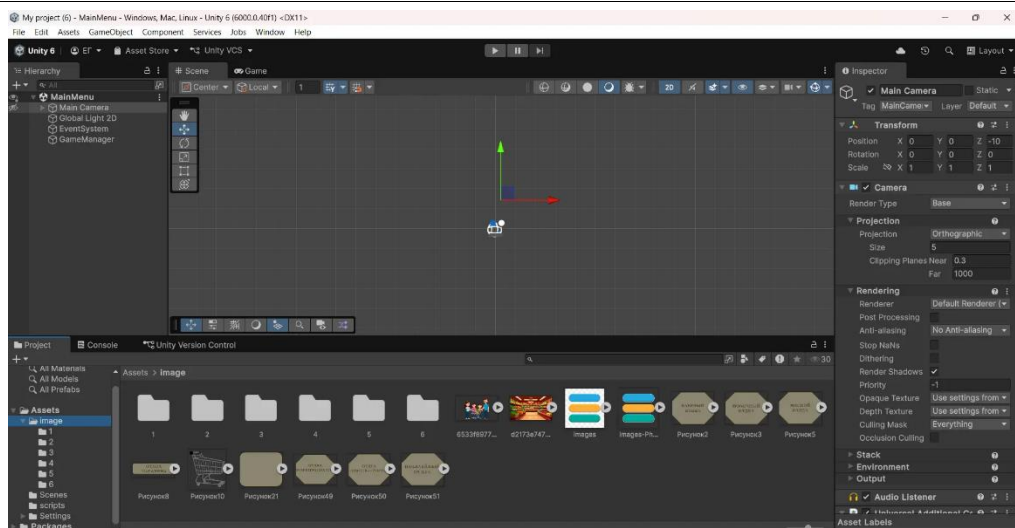


Рисунок 4 – Добавление изображений

Следующий этап — это открытие сцены MainMenu, которая служит главным меню игры. Первым делом создаём объект Canvas - основу для всех UI-элементов. Для этого в окне Hierarchy вызываем контекстное меню правой кнопкой мыши и выбираем UI -> Canvas. Чтобы интерфейс отображался корректно, необходимо настроить только что созданный Canvas. В инспекторе выполняем следующие действия: в поле Render Mode выбираем Screen Space – Camera; в поле Render Camera перетаскиваем камеру из иерархии; в Order in Layer выставляем значение 1; в Reference Resolution задаём разрешение 1920×1080 (см. рис. 5).

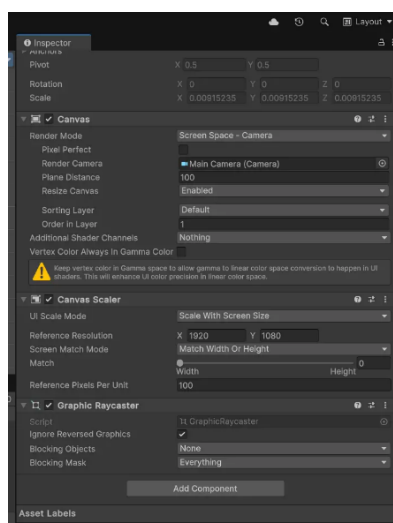


Рисунок 5 – Настройка Canvas

Внутри созданного Canvas необходимо разместить объект Image, который будет использоваться в качестве фонового изображения игры. После добавления в инспекторе следует отрегулировать его позицию, масштаб и привязки (якоря) таким образом, чтобы фон полностью перекрывал область обзора камеры. Затем в разделе Image инспектора нужно назначить заранее подготовленное фоновое изображение (см. рис. 6).

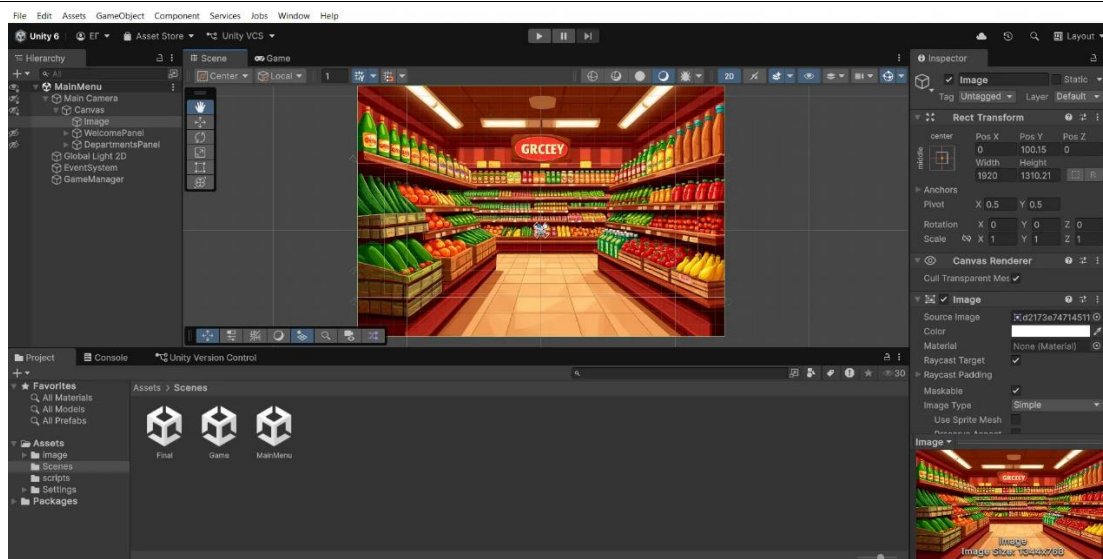


Рисунок 6 – Добавление фона

После создание фона, необходимо добавить объект Panel и переименовать в WelcomePanel. На созданную панель приветствия необходимо разместить объект Text для названия игры, Text для правил игры и объект Button для кнопки «начать игру!». Для объекта Text для названия игры, необходимо произвести следующие настройки в инспекторе: в окне Text набрать текст «Продуктовый магазин», в Font Style выбрать Bolt, размер текста 60, цвет текста синий. Для объекта Text правила игры в окне Text набрать текст «Правила игры 1. В игре 6 отделов магазина. 2. В каждом отделе лежат 6 продуктов. 3. Среди них есть лишние – не подходящие для этого отдела. 4. Перетаскивай в корзину только те продукты, которые относятся к данному отделу. 5. Собери все правильные продукты, чтобы пройти отдел. 6. Пройди все 6 отделов!», в Font Style выбрать Normal, размер текста 40, цвет текста синий. Также необходимо в панель добавить объект image для оформления панели (см. рис. 7).

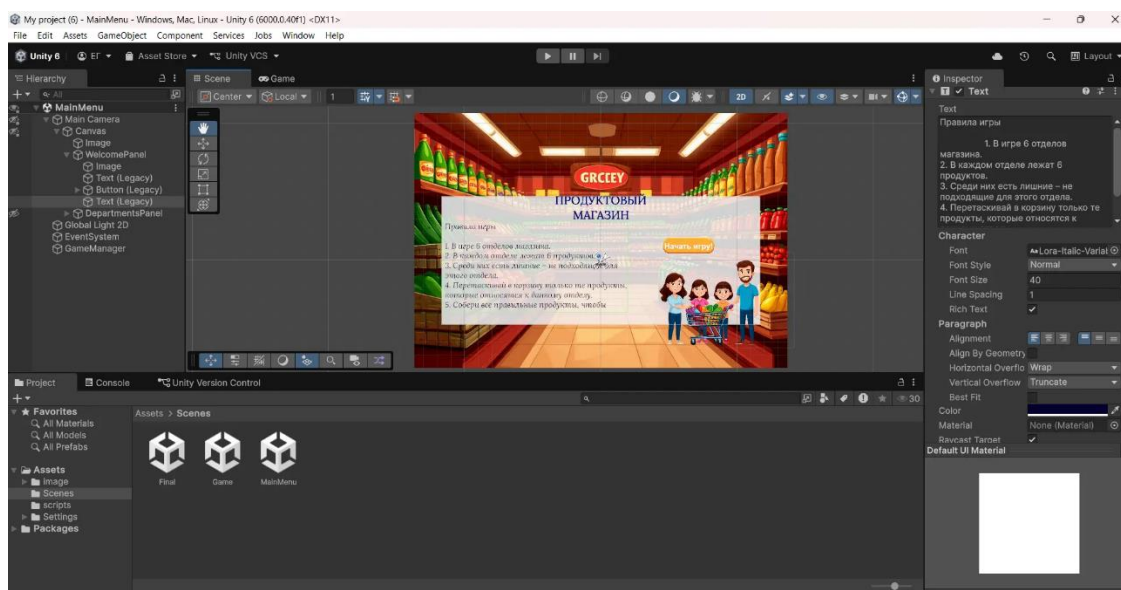


Рисунок 7 – Настройка панели

После создание приветственной панели необходимо создать панель с отделами продуктового магазина. Для этого создается объект Panel и переименовывается в DepartmentsPanel. В панели добавляется изображение с надписью «Отдел магазина» и создается шесть объектов Button, и в инспекторе кнопок добавляются уже подготовленные изображения кнопок с надписями отделов (см. рис. 8).

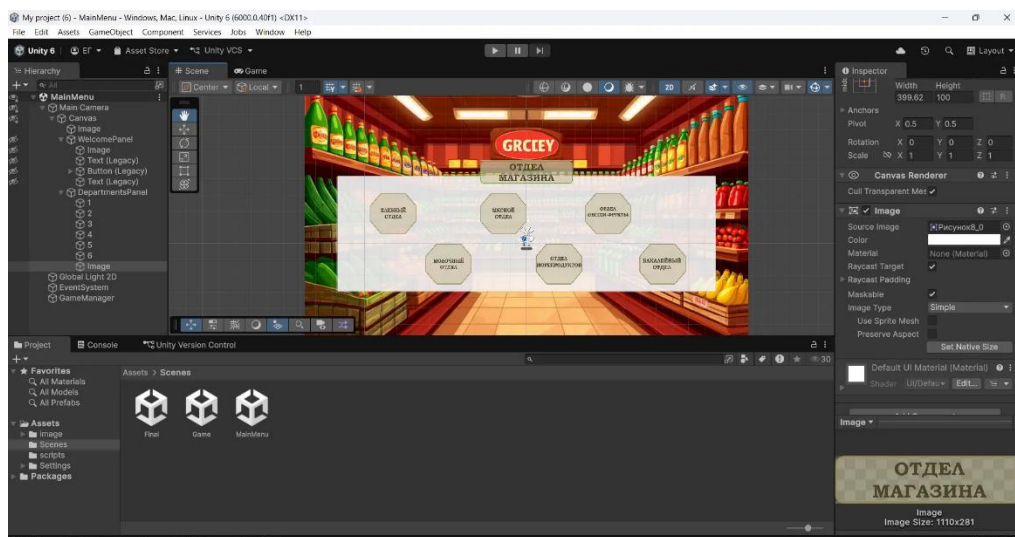


Рисунок 8 – Настройка панели

После оформления главного меню, можно приступать к созданию ее механики. Необходимо создать скрипт MainMenuManager, для управления главным меню.

Скрипт MainMenuManager:

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class MainMenuManager : MonoBehaviour
{
    public GameObject welcomePanel;
    public GameObject departmentsPanel;
    public Button[] departmentButtons;
    public Color completedColor = Color.gray;
    public Color defaultColor = Color.white;
    public static bool forceShowDepartments = false;
    void Start()
    {
        if (forceShowDepartments)
        {
            welcomePanel.SetActive(false);
            departmentsPanel.SetActive(true);
            forceShowDepartments = false; // сброс
        }
        else
        {
            welcomePanel.SetActive(true);
            departmentsPanel.SetActive(false);
        }
        UpdateDepartmentsUI();
    }
    public void OnStartButtonClick()
    {
        welcomePanel.SetActive(false);
        departmentsPanel.SetActive(true);
        UpdateDepartmentsUI();
    }
}
```

```

    }
    public void OnDepartmentClick(int index)
    {
        if (GameState.IsDepartmentCompleted(index)) return;
        GameState.CurrentDepartmentIndex = index;
        SceneManager.LoadScene("Game");
    }
    void UpdateDepartmentsUI()
    {
        for (int i = 0; i < departmentButtons.Length; i++)
        {
            bool completed = GameState.IsDepartmentCompleted(i);
            departmentButtons[i].interactable = !completed;
            ColorBlock cb = departmentButtons[i].colors;
            cb.normalColor = completed ? completedColor : defaultColor;
            departmentButtons[i].colors = cb;
        }
        if (GameState.AllDepartmentsCompleted())
            Invoke("LoadFinalScene", 1.5f);
    }
    void LoadFinalScene() => SceneManager.LoadScene("Final");
}

```

Для механики необходимо создать пустой объект и к нему добавить скрипт `MainMenuManager`, и в инспекторе добавить созданные панели (см. рис. 9).

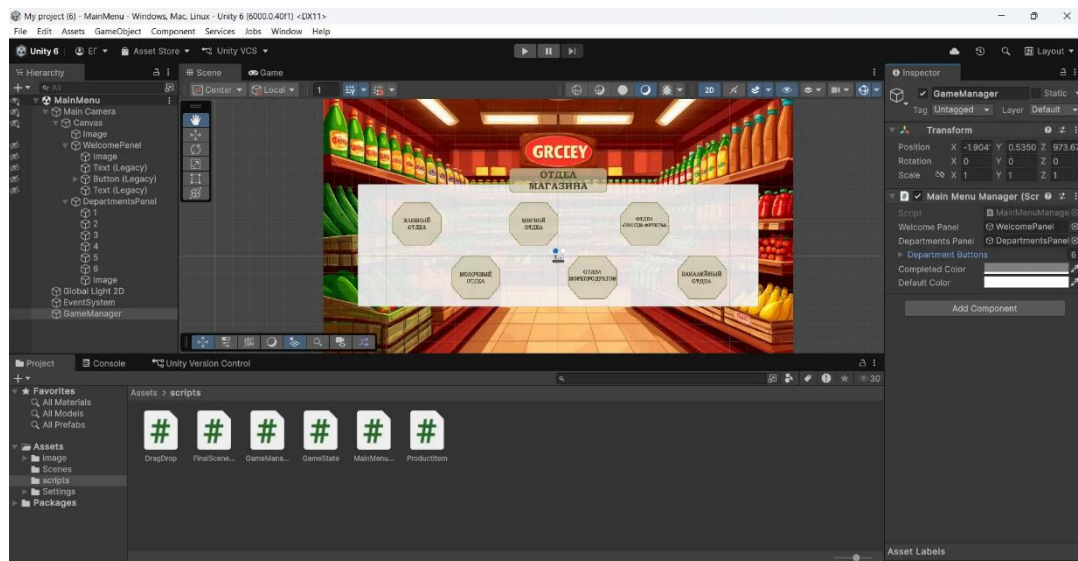


Рисунок 9 – Настройка механики меню

Далее можно приступать к сцене `Game`. На сцене также настраивается объект `Canvas` и фон игры, как и на первой сцене. После создания фона необходимо создать объект `Panel` и переименовать в `TopPanel`. В панели необходимо создать объект `image` и переименовать в `departmentNameImage`, на данном объекте будет появляться название отдела. Также в панели создается объект `Text` для вывода прогресса игры, и объект `Button` для кнопки в меню (см. рис. 10).

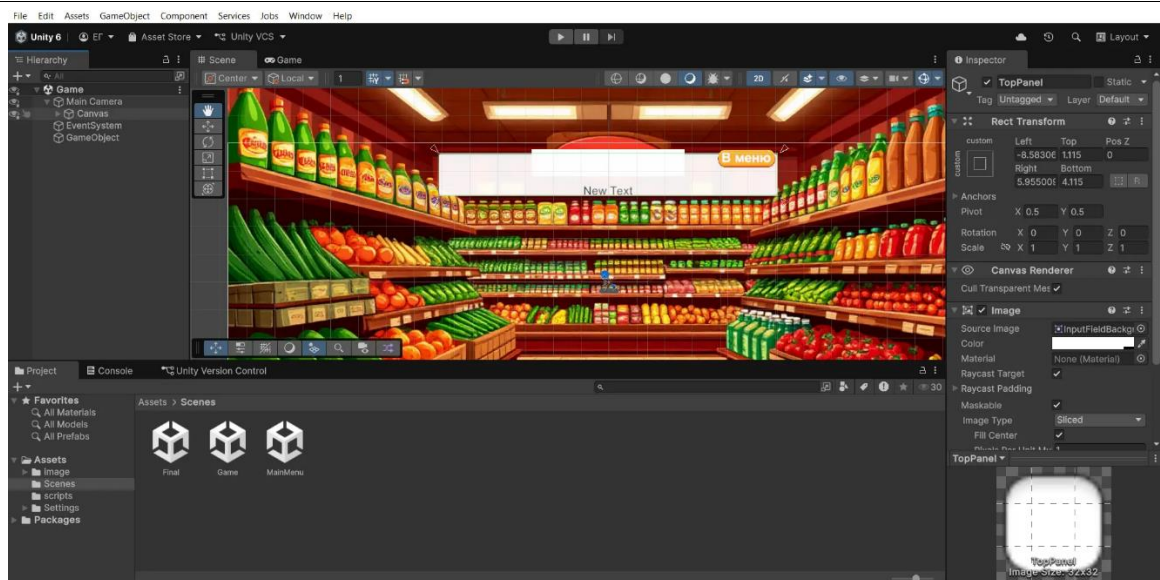


Рисунок 10 – Настройка объекта Text

Далее создается объект Panel для панели, где будут размещаться продукты. Для того, чтобы продукты были в две колонки, надо добавить в инспекторе компонент GridLayoutGroup и настроить компонент, Cell Size 160 на 160, Spacing 70 на 70, и Constraint Count 2 (см. рис. 11).

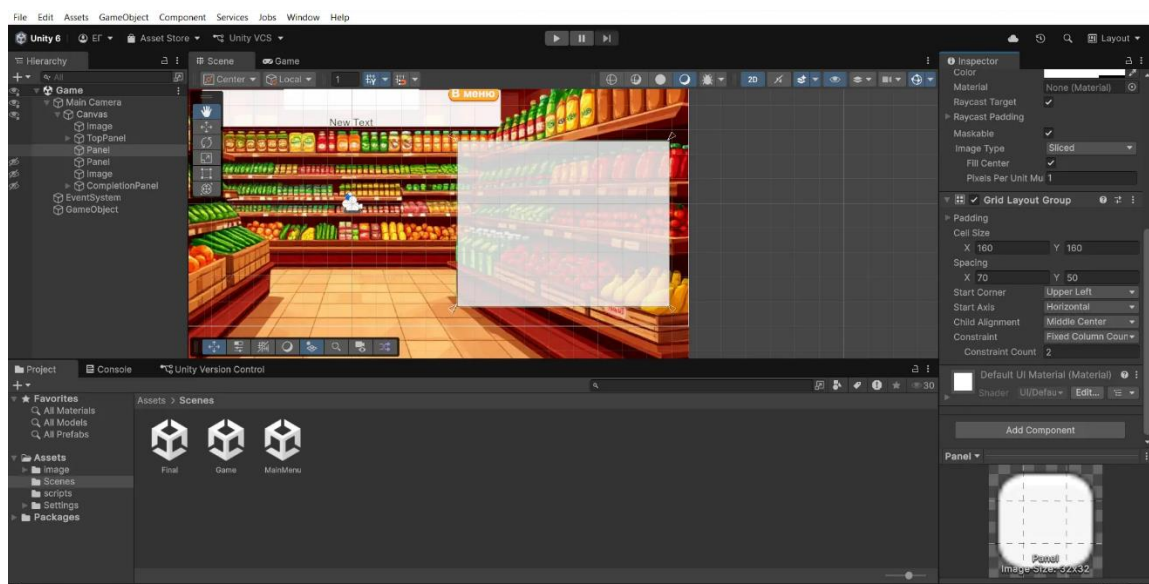


Рисунок 11 – Настройка панели продуктов

Дальше необходимо создать панель для продуктовой тележки. Создается объект Panel, и необходимо в инспекторе сделать панель прозрачной. После создания панели необходимо создать объект image и к объекту добавить изображение продуктовой тележки (см. рис. 12).

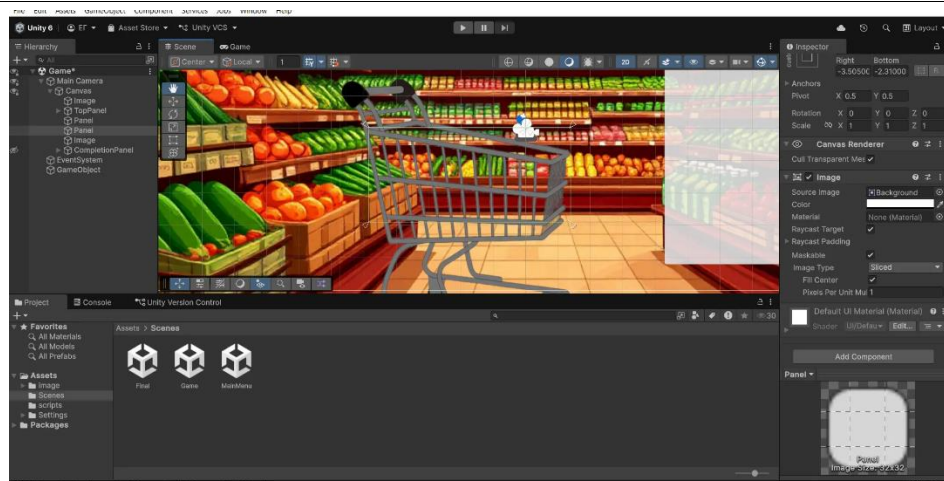


Рисунок 12 – Настройка панели продуктовой тележки

Последним шагом необходимо создать панель с поздравлением после завершения отдела. Также создается объект `Panel` и переименовывается в `CompletionPanel`. В панели необходимо создать объект `Text` для вывода текста поздравления (см. рис. 13).

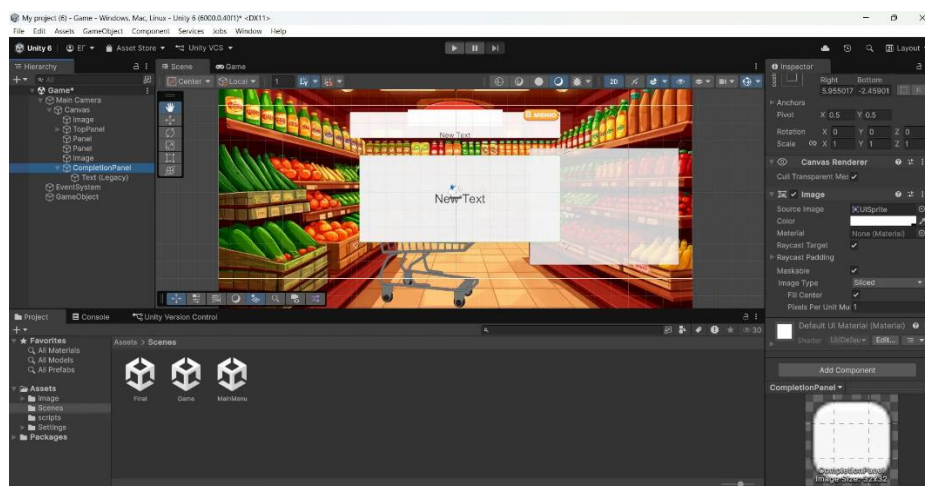


Рисунок 13 – Настройка панели поздравления

На сцене созданы все необходимые объекты, можно приступать к созданию механики. Необходимо создать скрипты. `GameState` – хранит глобальный прогресс игры и предоставляет методы для проверки, отметки и сброса состояния. `ProductItem` – служит маркером, указывающим, является ли данный продукт правильным для текущего отдела. `DragDrop` – реализует механизм перетаскивания продуктов мышью с проверкой попадания в зону корзины и возвратом в исходную позицию при ошибке. `GameManager` – создаёт игровое поле для выбранного отдела, обрабатывает логику сбора продуктов, отслеживает прогресс и переходит в меню или на финальную сцену.

Скрипт `GameState`:

```
public static class GameState
{
```

```

        private static bool[] completed = new bool[6];
        public static int CurrentDepartmentIndex { get; set; }
        public static bool IsDepartmentCompleted(int index) => index >= 0 && index < 6 &&
completed[index];
        public static void SetDepartmentCompleted(int index) { if (index >= 0 && index < 6)
completed[index] = true; }
        public static bool AllDepartmentsCompleted() { foreach (bool b in completed) if (!b)
return false; return true; }
        public static void Reset() { for (int i = 0; i < 6; i++) completed[i] = false;
CurrentDepartmentIndex = 0; }
    }

```

Скрипт ProductItem:

```

using UnityEngine;
public class ProductItem : MonoBehaviour
{
    public bool isCorrect;
}

```

Скрипт DragDrop:

```

using UnityEngine;
using UnityEngine.EventSystems;
public class DragDrop : MonoBehaviour, IBeginDragHandler, IDragHandler, IEndDragHandler
{
    private RectTransform rectTransform;
    private Canvas canvas;
    private CanvasGroup canvasGroup;
    private Vector2 originalPos;
    private Transform originalParent;
    private GameManager gameManager;
    private Transform basketTransform;
    public void Initialize(GameManager manager, Transform basket)
    {
        gameManager = manager;
        basketTransform = basket;
    }
    void Awake()
    {
        rectTransform = GetComponent<RectTransform>();
        if (rectTransform == null)
            rectTransform = gameObject.AddComponent<RectTransform>();
        canvas = GetComponentInParent<Canvas>();
        canvasGroup = GetComponent<CanvasGroup>();
        if (canvasGroup == null) canvasGroup = gameObject.AddComponent<CanvasGroup>();
        originalParent = transform.parent;
        originalPos = rectTransform.anchoredPosition;
    }
    public void OnBeginDrag(PointerEventData eventData)
    {
        originalPos = rectTransform.anchoredPosition;
        originalParent = transform.parent;
        canvasGroup.blocksRaycasts = false;
        transform.SetParent(canvas.transform);
    }
    public void OnDrag(PointerEventData eventData)
    {
        rectTransform.anchoredPosition += eventData.delta
/ canvas.scaleFactor;
    }
    public void OnEndDrag(PointerEventData eventData)
    {
        canvasGroup.blocksRaycasts = true;
        if (IsOverBasket(eventData))
            gameManager.OnProductDropped(gameObject);
        else
            ReturnToOriginalPosition();
    }
    public void ReturnToOriginalPosition()
    {

```

```

        transform.SetParent(originalParent);
        rectTransform.anchoredPosition = originalPos;
    }
    private bool IsOverBasket(PointerEventData eventData)
    {
        if (basketTransform == null) return false;
        RectTransform basketRect = basketTransform.GetComponent<RectTransform>();
        if (basketRect == null) return false;
        Vector2 localPoint;
        RectTransformUtility.ScreenPointToLocalPointInRectangle(
            basketRect, eventData.position, eventData.pressEventCamera, out
            localPoint);
        return basketRect.rect.Contains(localPoint);
    }
}

```

Скрипт GameManager:

```

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using System.Collections.Generic;
public class GameManager : MonoBehaviour
{
    [Header("UI References")]
    public Image departmentNameImage;
    public Text progressText;
    public Transform productsParent;    // родитель для слотов
    public Transform basketParent;
    public GameObject completionPanel;
    public Text feedbackText;
    [Header("Display Settings")]
    public Sprite backgroundSprite;    // фон слота
    public Vector2 slotSize = new Vector2(110, 110);
    public Vector2 productSize = new Vector2(90, 90);
    public float spacingX = 130f;
    public float spacingY = 130f;
    public float feedbackDuration = 1.5f;
    [System.Serializable]
    public class DepartmentData
    {
        public Sprite nameSprite;
        public Sprite[] correct = new Sprite[4]; // 4 правильных
        public Sprite[] wrong = new Sprite[2];  // 2 неправильных
    }
    public DepartmentData[] departments = new DepartmentData[6];
    private int correctCount = 0;
    private int totalCorrect = 4;
    private List<GameObject> productObjects = new List<GameObject>();
    private List<GameObject> slotObjects = new List<GameObject>();
    void Start()
    {
        int deptIndex = GameState.CurrentDepartmentIndex;
        if (departments == null || departments.Length == 0)
        {
            Debug.LogError("Массив departments пуст! Заполните в инспекторе.");
            return;
        }
        if (deptIndex < 0 || deptIndex >= departments.Length)
        {
            Debug.LogError("Некорректный индекс отдела: " + deptIndex);
            return;
        }
        if (departmentNameImage != null && departments[deptIndex].nameSprite != null)
            departmentNameImage.sprite = departments[deptIndex].nameSprite;
        else
            Debug.LogWarning("Нет спрайта названия для отдела " + deptIndex);
        completionPanel.SetActive(false);
        feedbackText.gameObject.SetActive(false);
        correctCount = 0;
        UpdateProgress();
    }
}

```

```

        // Проверка спрайтов продуктов
        Sprite[] correct = departments[deptIndex].correct;
        Sprite[] wrong = departments[deptIndex].wrong;
        if (correct == null || correct.Length != 4 || wrong == null || wrong.Length !=
2)
        {
            Debug.LogError("Ошибка: для отдела " + deptIndex + " нужно 4 правильных и 2
неправильных спрайта.");
            return;
        }
        List<Sprite> allSprites = new List<Sprite>();
        List<bool> flags = new List<bool>();
        foreach (Sprite s in correct) { allSprites.Add(s); flags.Add(true); }
        foreach (Sprite s in wrong) { allSprites.Add(s); flags.Add(false); }
        for (int i = 0; i < allSprites.Count; i++)
        {
            int r = Random.Range(i, allSprites.Count);
            Sprite temp = allSprites[i];
            allSprites[i] = allSprites[r];
            allSprites[r] = temp;
            bool f = flags[i];
            flags[i] = flags[r];
            flags[r] = f;
        }
        foreach (Transform child in productsParent)
            Destroy(child.gameObject);
        int cols = 2, rows = 3;
        float startX = -(cols - 1) * spacingX / 2f;
        float startY = (rows - 1) * spacingY / 2f;
        for (int i = 0; i < 6; i++)
        {
            int col = i % cols;
            int row = i / cols;
            float x = startX + col * spacingX;
            float y = startY - row * spacingY;
            Vector2 pos = new Vector2(x, y);
            GameObject slot = new GameObject("Slot_" + i);
            slot.transform.SetParent(productsParent, false);
            RectTransform slotRect = slot.AddComponent<RectTransform>()
;

            slotRect.sizeDelta = slotSize;
            slotRect.anchoredPosition = pos;
            Image slotImg = slot.AddComponent<Image>();
            if (backgroundSprite != null)
                slotImg.sprite = backgroundSprite;
            else
                slotImg.color = new Color(0.9f, 0.9f, 0.9f, 0.8f);
            slotObjects.Add(slot);
            GameObject product = new GameObject("Product_" + i);
            product.transform.SetParent(slot.transform, false);
            RectTransform prodRect = product.AddComponent<RectTransform>();
            prodRect.sizeDelta = productSize;
            prodRect.anchoredPosition = Vector2.zero
;

            Image prodImg = product.AddComponent<Image>();
            prodImg.sprite = allSprites[i];
            ProductItem item = product.AddComponent<ProductItem>();
            item.isCorrect = flags[i];
            DragDrop drag = product.AddComponent<DragDrop>();
            drag.Initialize(this, basketParent);
            productObjects.Add(product);
        }
    }
    public void OnProductDropped(GameObject product)
    {
        ProductItem item = product.GetComponent<ProductItem>();
        if (item == null) return;
        if (item.isCorrect)
        {
            correctCount++;
            UpdateProgress();
        }
    }
}

```

```

        product.transform.SetParent(basketParent);
        DragDrop drag = product.GetComponent<DragDrop>();
        if (drag) drag.enabled = false;
        ShowFeedback("Верно!", Color.green
    );

        if (correctCount >= totalCorrect)
        {
            completionPanel.SetActive(true);
            GameState.SetDepartmentCompleted(GameState.CurrentDepartmentIndex);
            if (GameState.AllDepartmentsCompleted())
            {
                Invoke("LoadFinalScene", 2.0f);
            }
            else
            {
                Invoke("GoBackToMenu", 2.0f);
            }
        }
    }
    else
    {
        DragDrop drag = product.GetComponent<DragDrop>();
        if (drag) drag.ReturnToOriginalPosition();
        ShowFeedback("Не подходит!", Color.red
    );
    }
}

void UpdateProgress()
{
    progressText.text = correctCount + " / " + totalCorrect;
}

void ShowFeedback(string message, Color color)
{
    feedbackText.text = message;
    feedbackText.color = color;
    feedbackText.gameObject.SetActive(true);
    CancelInvoke("HideFeedback");
    Invoke("HideFeedback", feedbackDuration);
}

void HideFeedback()
{
    feedbackText.gameObject.SetActive(false);
}

void GoBackToMenu()
{
    MainMenuManager.forceShowDepartments = true;
    SceneManager.LoadScene("MainMenu");
}

void LoadFinalScene()
{
    SceneManager.LoadScene("Final");
}

public void OnBackButton()
{
    SceneManager.LoadScene("MainMenu");
}
}

```

Для механики игры необходимо создать пустой объект и добавить к объекту скрипт GameManager. В инспекторе необходимо добавить все объекты с панели TopPanel, настроить размеры продуктов, и в компоненте Departments создать 6 компонентов, и настроить их под отделы, т.е. добавить в Name Sprite изображения названия отдела, в Correct добавить правильные продукты, а в Wrong не правильные продукты (см. рис. 14).

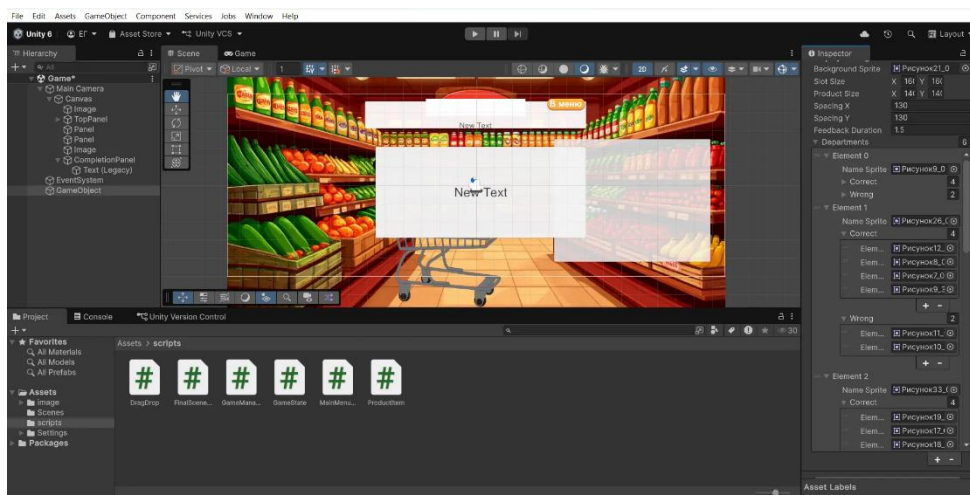


Рисунок 14 – Настройка механики игры

Сцена игры готова полностью, можно переходить к финальной сцене. В финальной сцене аналогично настраивается Canvas и фон игры. Далее необходимо создать панель и на панели разместить два объекта Text, два объекта Button и один объект image для оформления панели. В первом объекте Text необходимо настроить: в окне Text набрать текст «Поздравляем!», в Font Style выбрать Bolt, размер текста 70, цвет текста синий. Во втором объекте Text: в окне Text набрать текст «Все отделы пройдены!», в Font Style выбрать Bolt, размер текста 70, цвет текста синий. Для работоспособности кнопок необходимо создать скрипт FinalSceneManager который управляет финальной сценой, обеспечивая перезапуск игры или выход из приложения.

Скрипт FinalSceneManager:

```
using UnityEngine;
using UnityEngine.SceneManagement;
public class FinalSceneManager : MonoBehaviour
{
    public void OnPlayAgain()
    {
        GameState.Reset();
        MainMenuManager.forceShowDepartments = true;
        SceneManager.LoadScene("MainMenu");
    }
    public void OnExitGame()
    {
        #if UNITY_EDITOR
            UnityEditor.EditorApplication.isPlaying = false;
        #else
            Application.Quit();
        #endif
    }
}
```

Необходимо создать пустой объект и переместить скрипт на него. Созданный объект необходимо переместить в событие OnClick () двух кнопок. В кнопке Начать заново! Выбрать FinalSceneManager.OnPlayAgain, а для кнопки выход FinalSceneManager.OnExitGame (см. рис. 15).

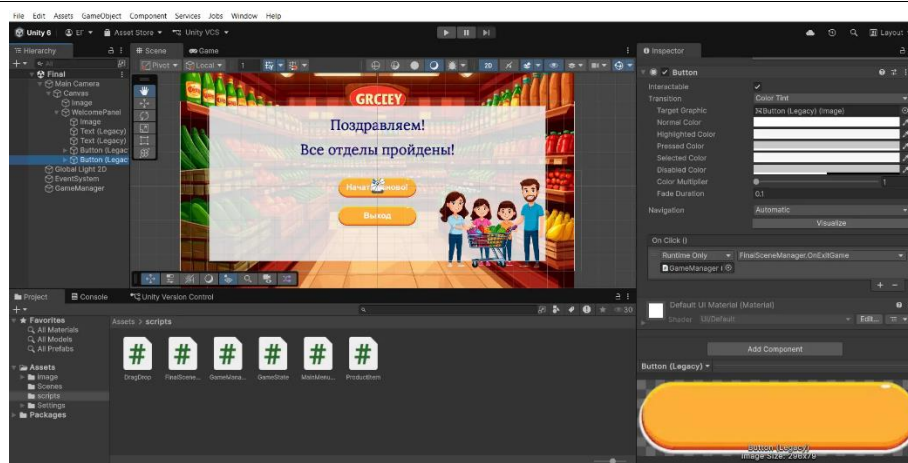


Рисунок 15 – Финальная сцена

Готовая игра (см. рис. 16. 1, 16. 2, 16. 3, 16. 4, 16. 5, 16. 6, 16. 7, 16. 8, 16. 9).



Рисунок 16.1 – Готовая игра



Рисунок 16.2 – Готовая игра



Рисунок 16.3 – Готовая игра



Рисунок 16.4 – Готовая игра



Рисунок 16.5 – Готовая игра



Рисунок 16.6 – Готовая игра



Рисунок 16.7 – Готовая игра



Рисунок 16.8 – Готовая игра



Рисунок 16.9 – Готовая игра

4 Выводы

В данной работе был показан процесс создания компьютерной игры «Продуктовый магазин» с помощью кроссплатформенной среды для создания интерактивных 2D и 3D приложений. Научная 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. Волгоград: Общество с ограниченной ответственностью "Сфера", 2024. С. 22-25. URL: <https://www.elibrary.ru/item.asp?id=68624300>
2. Фаст Я. А. Разработка 2D игр с элементами головоломок на платформе Unity // V Международная научная конференция по междисциплинарным исследованиям. Екатеринбург: Общество с ограниченной ответственностью "Институт Цифровой Экономики и Права", 2024. С. 266-269. URL: <https://www.elibrary.ru/item.asp?id=67143462>
3. Максимов Н. В. Разработка мобильной 2D игры "Bootur Jump" на Unity // Фундаментальные и прикладные аспекты развития современной науки. Уфа: Общество с ограниченной ответственностью "Научно-издательский центр "Вестник науки", 2023. С. 306-308. URL: <https://www.elibrary.ru/item.asp?id=59762421>
4. Любич А. А. Создание 2D-игры «б-у-у-м» на движке Unity // Фундаментальные и прикладные исследования молодых ученых. Новокузнецк: Кемеровский государственный университет, 2025. С. 167-169. URL: <https://www.elibrary.ru/item.asp?id=82653081>