

Создание компьютерной игры «Сказочное путешествие: геометрические фигуры» на Unity

Голубева Евгения Павловна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Цель статьи – создание компьютерной игры «Сказочное путешествие: геометрические фигуры» на Unity. Для создания игры использована кроссплатформенная среда для разработки интерактивных 2D- и 3D-приложений Unity, которая позволяет создавать 2D-игры, используя мощный физический движок и визуальные инструменты. В результате работы создана готовая компьютерная игра «Сказочное путешествие: геометрические фигуры».

Ключевые слова: Unity, игра, фигуры, геометрия.

Creating a computer game "Fairy Tale Journey: Geometric Shapes" using Unity

Golubeva Evgeniya Pavlovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The purpose of the article is to create a computer game "Fairy Tale Journey: Geometric Shapes" using Unity. To create the game, we used the cross-platform environment for developing 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 have created a finished computer game "Fairy Tale Journey: Geometric Shapes".

Keywords: Unity, game, shapes, geometry.

1 Введение

1.1 Актуальность

Современная образовательная парадигма претерпевает фундаментальные изменения в условиях цифровизации. Возникает объективная потребность в научно обоснованной интеграции технологических решений в образовательный процесс. Интерактивные образовательные приложения и игровые технологии становятся не просто дополнением к традиционным методам обучения, а самостоятельным эффективным инструментом формирования учебных компетенций. Это создаёт благоприятные предпосылки для разработки и внедрения новых форм обучающих компьютерных игр.

Обучение геометрическим фигурам на начальных этапах сталкивается с рядом системных трудностей. Исследования показывают, что у учащихся наблюдается низкий уровень концептуального понимания геометрических форм, а традиционные методы обучения не обеспечивают достаточной эффективности в формировании устойчивых геометрических представлений. Учащиеся испытывают сложности с активным включением в процесс изучения геометрии и глубоким осмыслением абстрактных понятий. Кроме того, наблюдается ограниченная доступность интерактивных обучающих средств по данной тематике, что требует поиска новых подходов к визуализации и освоению геометрического материала.

Игровой движок Unity представляет собой мощный инструмент для создания образовательных игр благодаря своей универсальности и широкому функционалу. Unity широко используется для разработки 2D- и 3D-игр для различных платформ, а также позволяет визуализировать учебный контент и способствует развитию мышления учащихся. Мини-игры на базе Unity, изначально воспринимаемые как средство развлечения, способны выполнять полноценные образовательные функции. Существующие разработки демонстрируют высокую валидность (до 96-98%) и практичность образовательных игр на Unity, что подтверждает целесообразность выбора данной платформы для создания обучающих продуктов.

Несмотря на имеющиеся разработки в области образовательных игр, в современной научной литературе отмечается недостаток системной интеграции стратегий геометрической геймификации. Существующие исследования указывают на проблемы неполной совместимости между игровыми механиками и учебными задачами, а также на необходимость дальнейших исследований в области разработки и апробации инновационных дидактических игр. В этом контексте создание компьютерной игры «Сказочное путешествие: геометрические фигуры» на платформе Unity представляет собой актуальное исследование, направленное на восполнение указанного пробела через разработку методически обоснованного игрового продукта, сочетающего современные технологии, сказочный нарратив и эффективные педагогические подходы к формированию геометрических представлений у обучающихся.

Таким образом, актуальность представленного исследования обусловлена совокупностью факторов: цифровой трансформацией образования, недостаточной эффективностью традиционных методов обучения геометрии, доказанным дидактическим потенциалом игровых технологий, технологическими возможностями Unity для создания образовательного контента и наличием нереализованного научно-практического пробела в области системной интеграции геометрической геймификации. Разработка игры «Сказочное путешествие: геометрические фигуры» отвечает на вызовы современного образования и вносит вклад в развитие теории и практики игрового обучения математике.

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 «Сказочное путешествие: геометрические фигуры» при помощи кроссплатформенной среды для создания интерактивных 2D и 3D приложений Unity.

2 Материалы и методы

В качестве базовой платформы для реализации игры «Сказочное путешествие: геометрические фигуры» используется Unity. Предпочтение отдано ей благодаря лёгкости в изучении, эргономичному пользовательскому интерфейсу и нативной интеграции с C#. Кодовая часть проекта разрабатывается и тестируется в среде Visual Studio Community.

3 Результаты

Разработка начинается с создания нового проекта в Unity. Для этого в Unity Hub выбирается шаблон «Universal 2D», который подразумевает использование универсального конвейера рендеринга, после чего нажимается кнопка «Создать проект» (см. рис. 1).

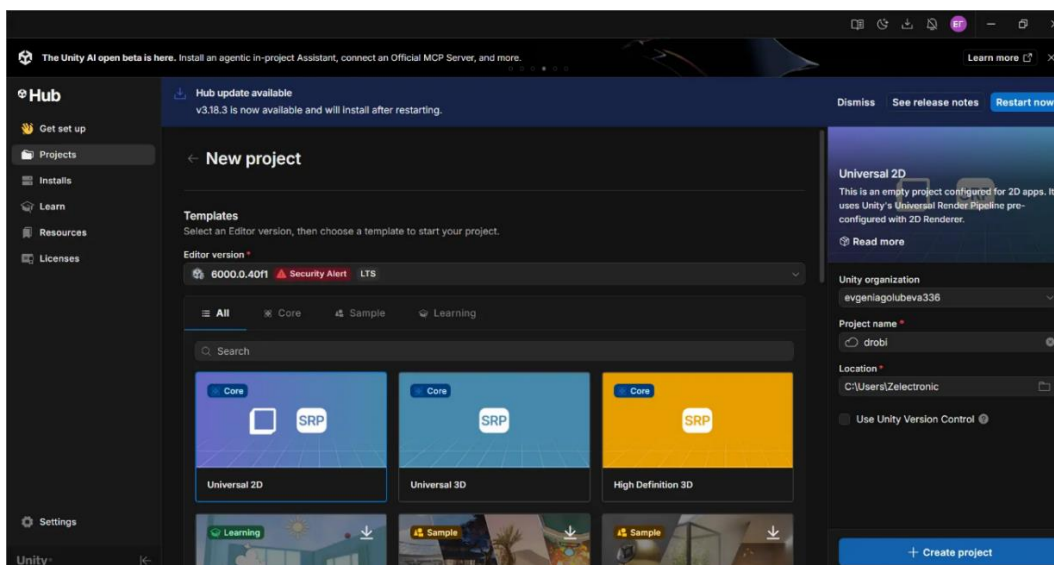


Рисунок 1 – Создание проекта

После того как проект на основе шаблона Universal 2D создан, требуется выполнить его первичную настройку под целевую ПК-платформу. Для этого необходимо перейти в меню File -> Build Profiles и выбрать в качестве целевой платформы Windows (см. рис. 2).

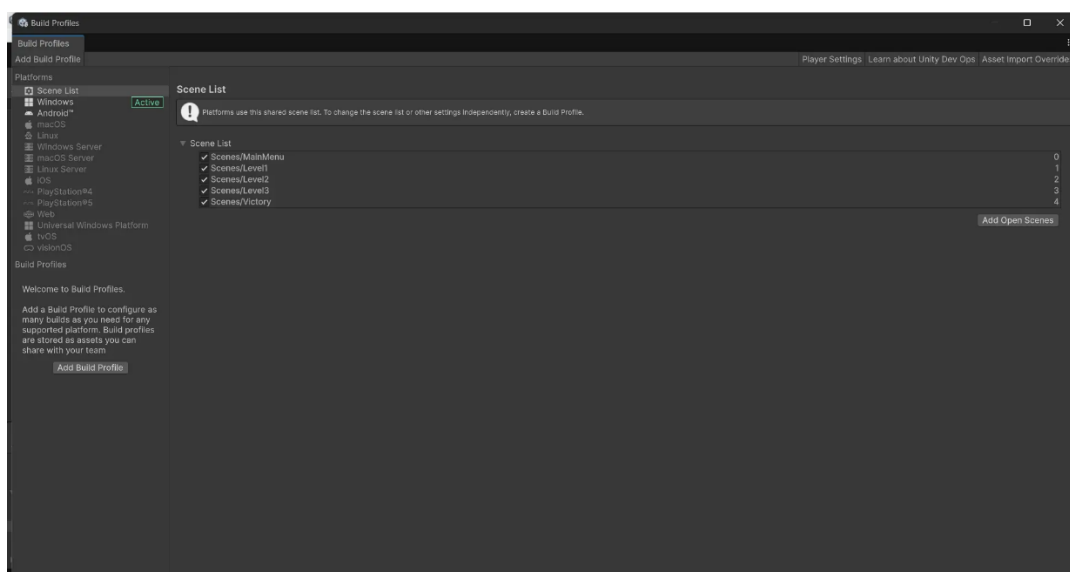


Рисунок 2 – Настройка проекта

Все заранее подготовленные графические материалы, задействованные в игре, необходимо скопировать в папку «image» (см. рис. 3).

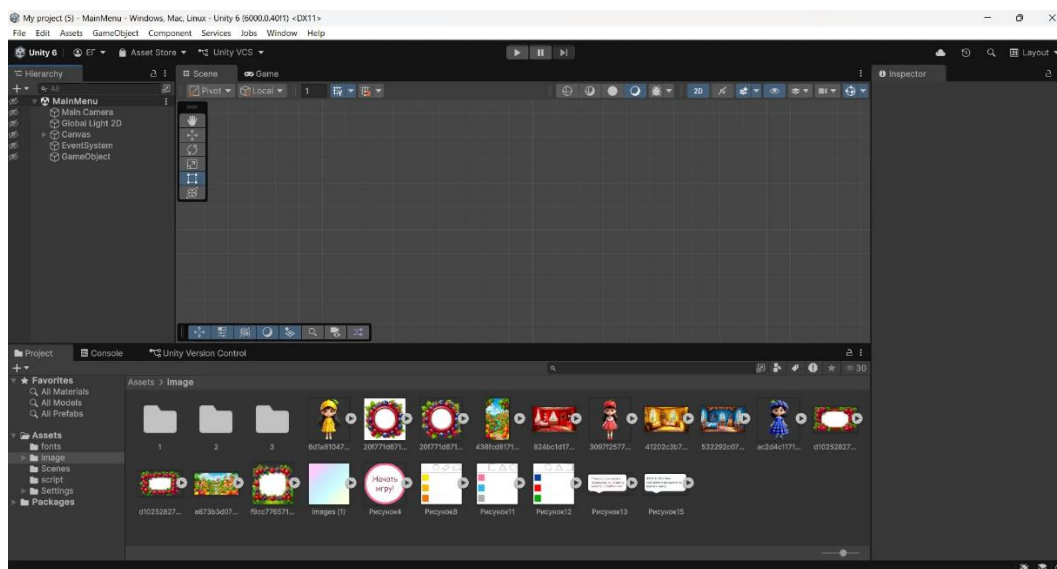


Рисунок 3 – Добавление изображений

Далее необходимо создать четыре сцены. Так как при создании проекта одна сцена уже генерируется автоматически, нужно добавить лишь три сцены. Для этого в папке Scenes вызывается контекстное меню правой кнопкой мыши, Create -> Scene и создаём недостающие сцены. После этого четыре сцены необходимо переименовать: MainMenu, Level1, Level2, Level3 и Victory (см. рис. 4).

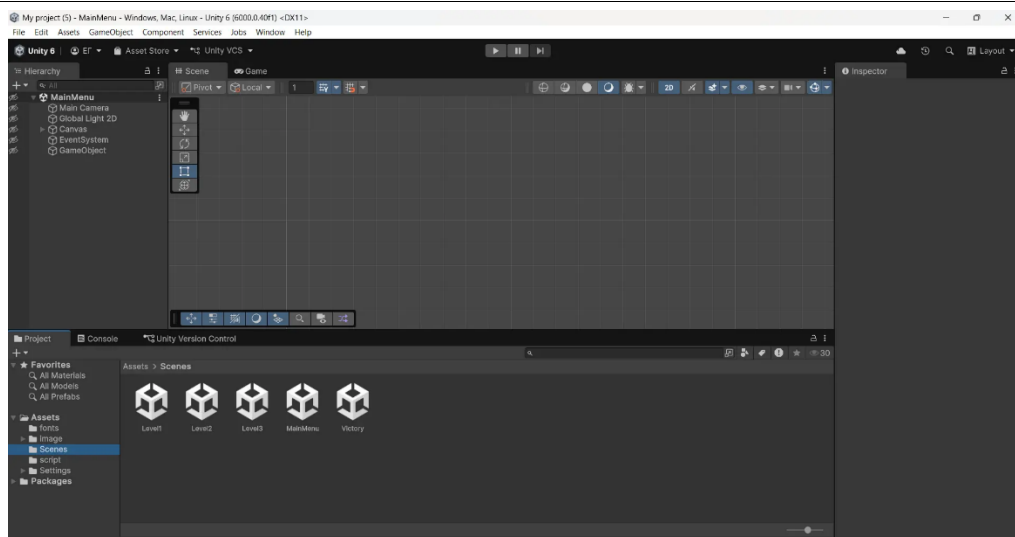


Рисунок 4 – Создание сцен игры

На следующем этапе открывается сцена MainMenu, выполняющая функцию главного меню игры. Прежде всего создаётся объект Canvas, который служит базой для всех элементов пользовательского интерфейса. Для этого в окне Hierarchy вызывается контекстное меню (правой кнопкой мыши), где выбирается UI -> Canvas. Чтобы интерфейс отображался корректно, необходимо настроить созданный Canvas. В окне Inspector выполняются следующие действия: в поле Render Mode устанавливается значение Screen Space – Camera; в поле Render Camera перетаскивается камера из иерархии; параметру Order in Layer присваивается значение 1; в поле Reference Resolution задаётся разрешение 1920×1080.

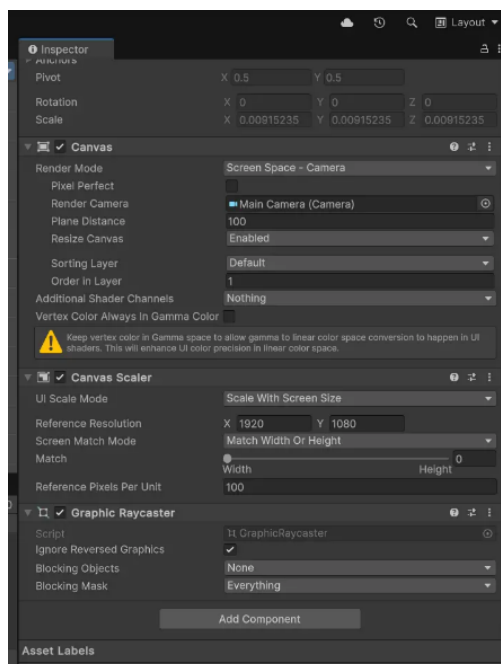


Рисунок 5 – Настройка Canvas

На созданном Canvas необходимо разместить объект Image, который будет выполнять роль фонового изображения игры. После добавления объекта

в инспекторе требуется настроить его позицию, масштаб и якоря таким образом, чтобы фон полностью перекрывал область видимости камеры. Затем в разделе Image инспектора следует назначить заранее подготовленное фоновое изображение (см. рис. 6).

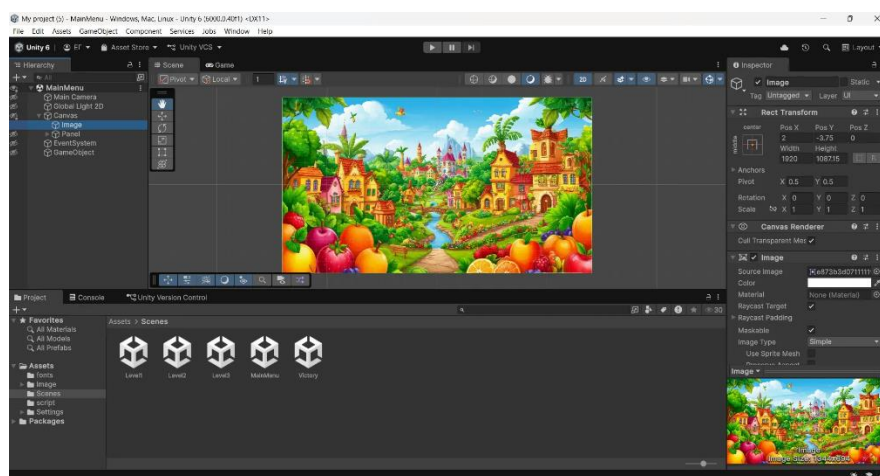


Рисунок 6 – Добавление фона

После создание фона, необходимо добавить объект Panel. На созданную панель приветствия необходимо разместить объект Text для вступления 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 «Далеко-далеко, в солнечной Фруктовой стране, живут весёлые друзья: весёлая Лимончик, добрая Клубничка и умница Черничка. У каждого из них есть любимые игрушки и вещи. Но однажды налетел озорной Ветер-Проказник и перепутал все предметы на их волшебных полочках! Теперь вещи лежат в куче, а друзья никак не могут их разобрать. Нужен смелый и внимательный помощник, который вернёт порядок! Поможешь?», в Font Style выбрать Normal, размер текста 45, цвет текста черный (см. рис. 7).

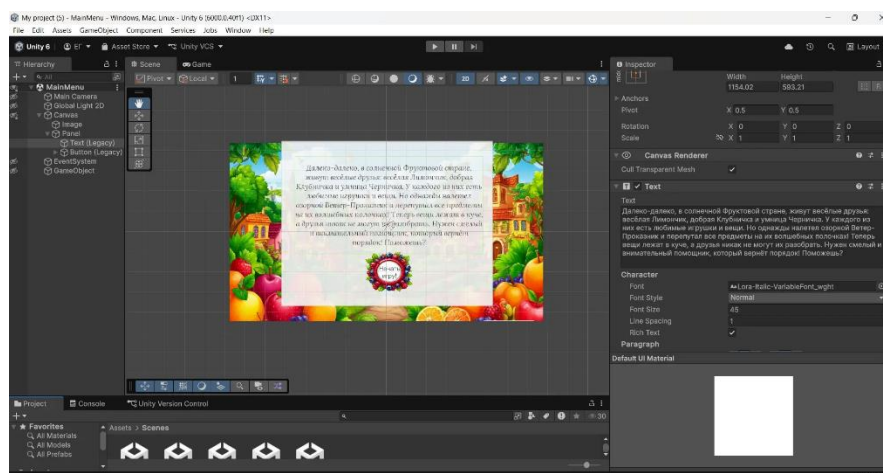


Рисунок 7 – Настройка панели

Для того, чтобы кнопка была активной необходимо создать скрипт GameManager который управляет глобальным состоянием игры: загружает сцены, переключает уровни.

Скрипт GameManager:

```
using UnityEngine;
using UnityEngine.SceneManagement;
public class GameManager : MonoBehaviour
{
    public static GameManager Instance;
    [Header("Audio")]
    public AudioClip correctSound;
    public AudioClip wrongSound;
    public AudioClip victorySound;
    private const int totalLevels = 3; // Уровни: Level1, Level2, Level3
    private void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else
        {
            Destroy(gameObject);
        }
    }
    public void LoadLevel(int index)
    {
        if (index >= 0 && index < totalLevels)
        {
            SceneManager.LoadScene("Level" + (index + 1));
        }
        else
        {
            SceneManager.LoadScene("Victory");
            PlayVictorySound();
        }
    }
    public void NextLevel()
    {
        string currentScene = SceneManager.GetActiveScene().name;
        int currentIndex = -1;
        if (currentScene == "Level1")
            currentIndex = 0;
        else if (currentScene == "Level2")
            currentIndex = 1;
        else if (currentScene == "Level3")
            currentIndex = 2;
        else
        {
            LoadLevel(0);
            return;
        }
        LoadLevel(currentIndex + 1);
    }
    public void PlayCorrectSound()
    {
        if (correctSound != null)
            AudioSource.PlayClipAtPoint(correctSound, Camera.main.transform.position);
    }
    public void PlayWrongSound()
    {
        if (wrongSound != null)
            AudioSource.PlayClipAtPoint(wrongSound, Camera.main.transform.position);
    }
    public void PlayVictorySound()
    {
        if (victorySound != null)
            AudioSource.PlayClipAtPoint(victorySound, Camera.main.transform.position);
    }
}
```

Далее необходимо создать пустой объект и переместить на него скрипт GameManager. Созданный объект необходимо перенести на кнопку в событие OnClick () и выбрать GameManager.LoadLevel (см. рис. 8).

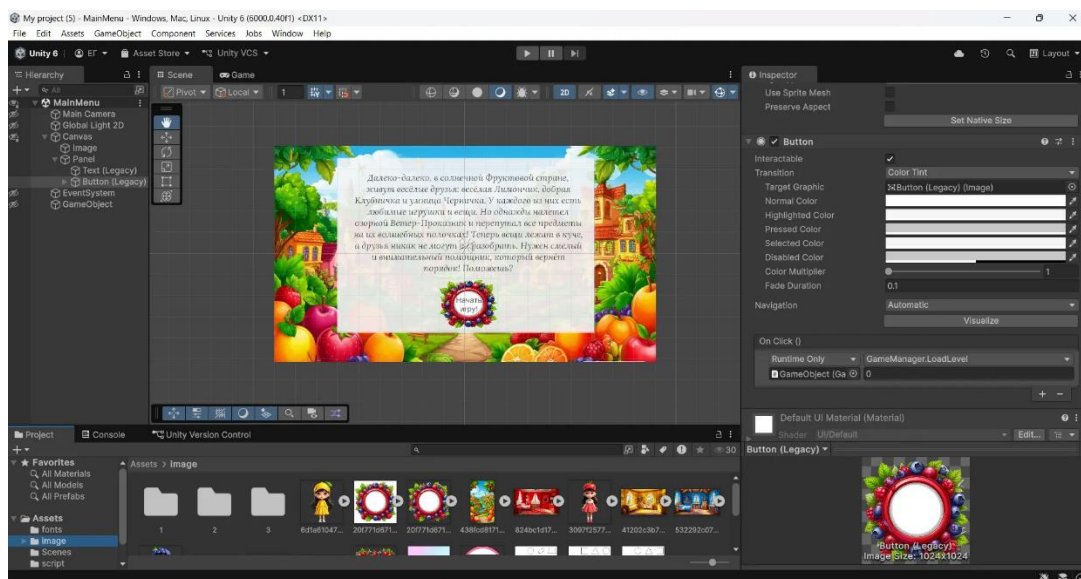


Рисунок 8 – Настройка кнопки

Настройка меню завершена, можно приступать к настройке самой игры. Для начала необходимо перейти на сцену Level1. Настройки Canvas и фона такие же, как и у сцены меню. Первым шагом необходимо добавить два объекта image и добавить к ним изображения персонажа игры и надписи с заданием игры (см. рис. 9).

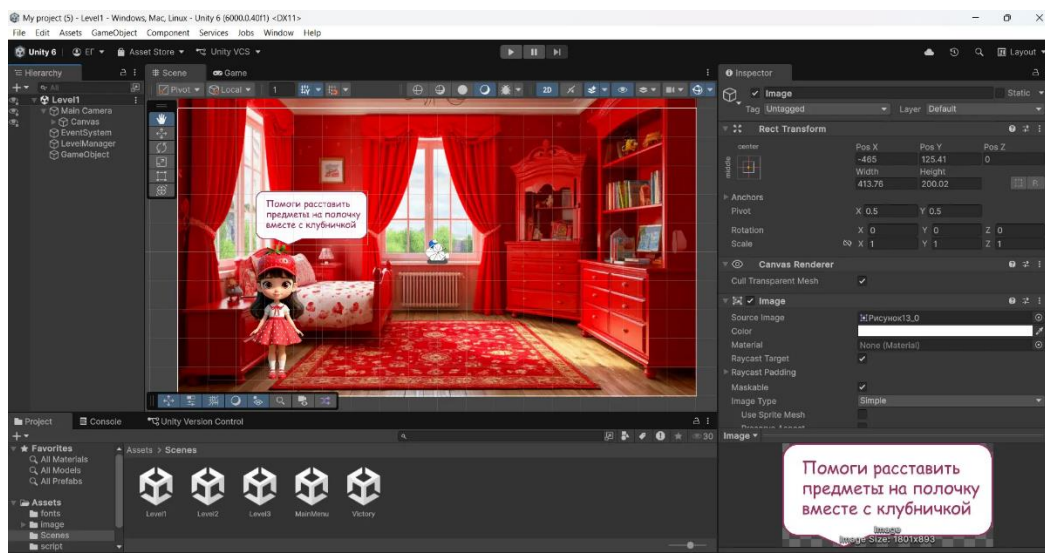


Рисунок 9 – Настройка сцены

Далее необходимо создать пустой объект для создания сетки, в которую будут перемещаться предметы в соответствующие ячейки. Объект необходимо переименовать в CellsContainer. В объект необходимо добавить изображение сетки. Для того, чтобы ячейки были активными необходимо

создать скрипт CellSlot который отвечает за приём перетаскиваемого предмета в ячейку, проверяет, совпадает ли он с целевой ячейкой, и визуально отображает предмет на полке.

Скрипт CellSlot:

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
public class CellSlot : MonoBehaviour, IDropHandler
{
    public bool isOccupied = false;
    private Image cellImage;
    private Image itemImage;
    private LevelManager levelManager;
    private void Awake()
    {
        cellImage = GetComponent<Image>();
        Transform itemTransform = transform.Find("ItemImage");
        if (itemTransform != null)
        {
            itemImage = itemTransform.GetComponent<Image>();
        }
        else
        {
            GameObject newItemImage = new GameObject("ItemImage");
            newItemImage.transform.SetParent(transform);
            RectTransform rect = newItemImage.AddComponent<RectTransform>();
            rect.anchorMin = Vector2.zero
;
            rect.anchorMax = Vector2.one
;
            rect.offsetMin = Vector2.zero
;
            rect.offsetMax = Vector2.zero
;

            itemImage = newItemImage.AddComponent<Image>();
            itemImage.raycastTarget = false; // Не мешаем событиям
            itemImage.enabled = false;
        }
        levelManager = FindObjectOfType<LevelManager>();
        if (cellImage != null)
        {
            Color c = cellImage.color;
            c.a = 0.01f;
            cellImage.color = c;
            cellImage.raycastTarget = true;
        }
    }
    public void SetItem(Sprite sprite)
    {
        isOccupied = true;
        if (itemImage != null)
        {
            itemImage.sprite = sprite;
            itemImage.enabled = true;
        }
    }
    public void OnDrop(PointerEventData eventData)
    {
        if (isOccupied) return;
        if (levelManager == null)
            levelManager = FindObjectOfType<LevelManager>();
        if (levelManager == null)
        {
            Debug.LogError("LevelManager не найден!");
            return;
        }
        GameObject draggedObject = eventData.pointerDrag;
        if (draggedObject == null) return;
        DraggableItem draggable = draggedObject.GetComponent<DraggableItem>();
        if (draggable == null) return;
        if (draggable.targetCell == this)
        {
            Sprite itemSprite = draggable.GetComponent<Image>().sprite;
            levelManager.OnItemPlacedCorrectly(itemSprite);
            SetItem(itemSprite);
        }
    }
}
```

```

        Destroy(draggedObject);
    }
    else
    {
        levelManager.OnItemPlacedWrong();
        draggable.ReturnToPanel();
    }
}
}

```

В объекте CellsContainer необходимо создать пустой объект, дать название Cell_1_1 и в инспекторе добавить компонент image, сделать его прозрачным, а также добавить скрипт CellSlot. В объекте Cell_1_1 необходимо создать объект image и переименовать в itemImage. Объект Cell_1_1 необходимо продублировать 8 раз, чтобы получилось 9 ячеек (см. рис. 10).

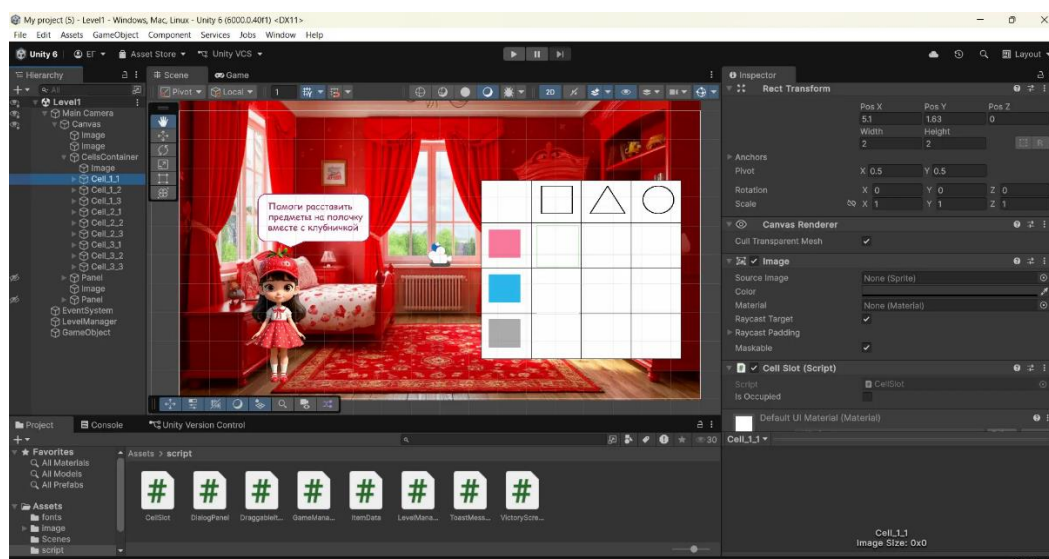


Рисунок 10 – Создание игрового поля

Далее можно приступить к созданию панели с предметами. Для этого создается объект Panel. В данном объекте необходимо создать объект image и его настроить в инспекторе: добавить изображение предмета, добавить компонент canvas Group, Box Collider 2D, Graphic Raycaster. Для работоспособности ячейки необходимо создать скрипт DraggableItem который реализует механику перетаскивания предмета мышью и возвращает его на панель при ошибке или по окончании перетаскивания.

Скрипт DraggableItem:

```

using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
public class DraggableItem : MonoBehaviour, IBeginDragHandler, IDragHandler, IEndDragHandler
{
    [Header("Целевая ячейка")]
    public CellSlot targetCell;
    private CanvasGroup canvasGroup;
    private RectTransform rectTransform;
    private Transform startParent;
    private Vector3 startPosition;
    private void Awake()
    {
        canvasGroup = GetComponent<CanvasGroup>();
        rectTransform = GetComponent<RectTransform>();
        startParent = transform.parent;
    }
}

```

```

        startPosition = rectTransform.position;
        if (canvasGroup == null)
            canvasGroup = gameObject.AddComponent<CanvasGroup>();
    }
    public void OnBeginDrag(PointerEventData eventData)
    {
        canvasGroup.blocksRaycasts = false;
        canvasGroup.alpha = 1f;
        transform.SetAsLastSibling(); // Поверх других
    }
    public void OnDrag(PointerEventData eventData)
    {
        RectTransform parentRect = (RectTransform)startParent;
        RectTransformUtility.ScreenPointToLocalPointInRectangle(
            parentRect,
            eventData.position,
            eventData.pressEventCamera,
            out Vector2 localPoint);
        rectTransform.localPosition = localPoint;
    }
    public void OnEndDrag(PointerEventData eventData)
    {
        canvasGroup.blocksRaycasts = true;
        canvasGroup.alpha = 1f;
        ReturnToPanel();
    }
    public void ReturnToPanel()
    {
        rectTransform.position = startPosition;
    }
}

```

Скрипт `DraggableItem` необходимо добавить в изображения предмета, и в инспекторе назначить предмету подходящую ячейку. Чтобы не создавать предметы по отдельности и их настраивать, можно клонировать объект с предметов 8 раз, и поменять только изображения предмета, и ячейку в скрипте (см. рис. 11).

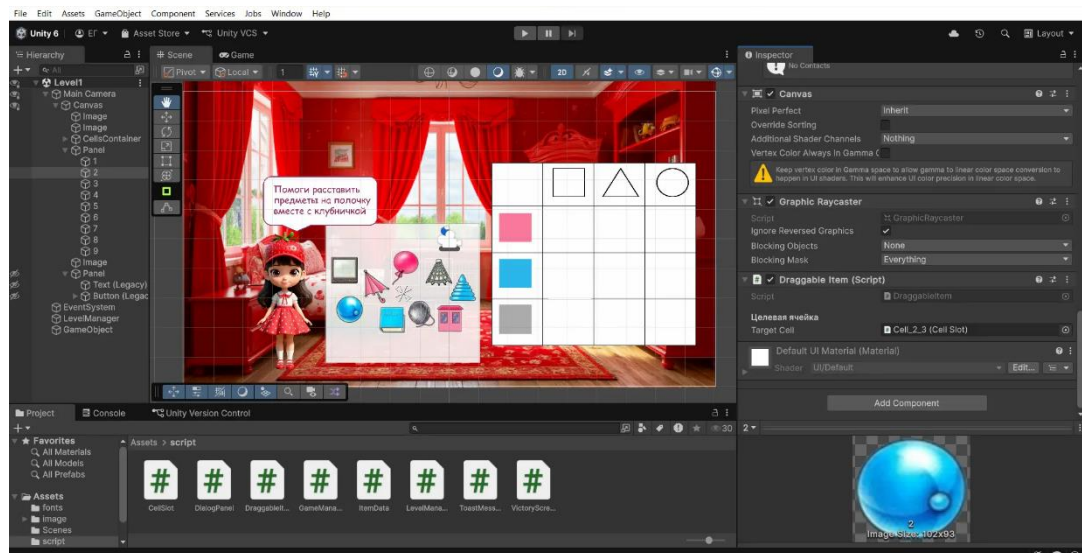


Рисунок 11 – Настройка сцены

Также необходимо создать панель поздравления. Для этого также создается объект `Panel`, и в инспекторе необходимо выбрать изображение подготовленной панели. Для того чтобы панель работала необходимо создать скрипт `DialogPanel`.

Скрипт `DialogPanel`:

```

using UnityEngine;
using UnityEngine.UI;
using System;
public class DialogPanel : MonoBehaviour
{
    public Text messageText;
    public Button actionButton;
    public Action OnButtonClick;
    private void Start()
    {
        if (actionButton != null)
            actionButton.onClick.AddListener(() => OnButtonClick?.Invoke());
    }
    public void SetText(string text)
    {
        if (messageText != null) messageText.text = text;
    }
    public void SetButtonText(string text)
    {
        if (actionButton != null)
        {
            Text btnText = actionButton.GetComponentInChildren<Text>();
            if (btnText != null) btnText.text = text;
        }
    }
    public void Show() => gameObject.SetActive(true);
    public void Hide() => gameObject.SetActive(false);
}

```

На панели необходимо разместить объект Text для вывода текста поздравление от персонажа, и объект button для кнопки продолжить. Также необходимо добавить скрипт DialogPanel, и в инспекторе в окне скрипта добавить объект Text и button панели (см. рис. 12).

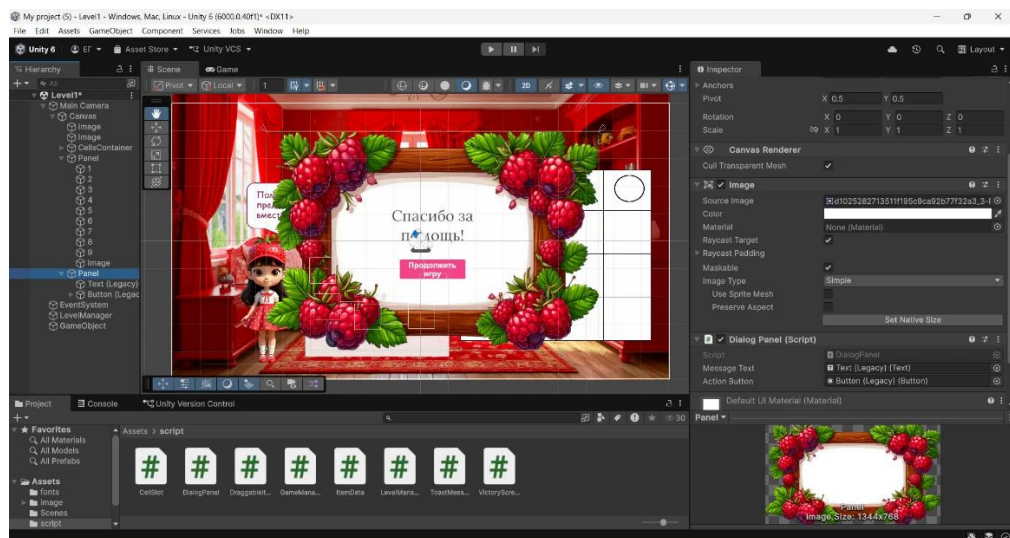


Рисунок 12 – Настройка панели поздравления

Следующий этап – создание механики уровня. Для этого необходимо создать скрипт LevelManager который контролирует логику текущего уровня, отслеживает количество правильно установленных предметов, перемешивает их на панели и вызывает панель победы после завершения уровня.

Скрипт LevelManager:

```

using UnityEngine;
using UnityEngine.UI;
using System.Collections;
public class LevelManager : MonoBehaviour
{

```

```

[Header("Level Data")]
public int levelIndex;           // 0, 1 или 2
public Sprite characterAvatar;
public string characterName;
[Header("UI References")]
public Transform itemPanel;
public Image characterImage;
public Text characterNameText;
public DialogPanel winPanel;    // Панель победы (спасибо за помощь)
private int correctPlaced = 0;
private const int totalItems = 9;
private void Start()
{
    if (characterImage != null) characterImage.sprite = characterAvatar;
    if (characterNameText != null) characterNameText.text = characterName;
    if (GameManager.Instance == null)
        Debug.LogWarning("GameManager не найден!");
    ShuffleItems();
    if (winPanel != null)
        winPanel.Hide();
}
private void ShuffleItems()
{
    if (itemPanel == null) return;
    int count = itemPanel.childCount;
    for (int i = 0; i < count; i++)
    {
        int newIndex = Random.Range(0, count);
        itemPanel.GetChild(i).SetSiblingIndex(newIndex);
    }
}
public void OnItemPlacedCorrectly(Sprite itemSprite)
{
    correctPlaced++;
    if (GameManager.Instance != null)
        GameManager.Instance.PlayCorrectSound();
    if (correctPlaced == totalItems)
    {
        ShowWinPanel();
    }
}
public void OnItemPlacedWrong()
{
    if (GameManager.Instance != null)
        GameManager.Instance.PlayWrongSound();
    Debug.Log("Не та ячейка!");
}
private void ShowWinPanel()
{
    if (winPanel != null)
    {
        string message = "Спасибо за помощь! 🍓 ";
        if (levelIndex == 0) message = "Клубничка благодарит! 🍓 🍓 ";
        else if (levelIndex == 1) message = "Спасибо, Лимоне рада! 🍓 ";
        else if (levelIndex == 2) message = "Черничка говорит спасибо! ☐ 🍓 ";
        winPanel.SetText(message);
        winPanel.SetButtonText("Далее");
        winPanel.OnButtonClick = () =>
        {
            winPanel.Hide();
            if (GameManager.Instance != null)
                GameManager.Instance.NextLevel();
            else
            {
                UnityEngine.SceneManagement.SceneManager.LoadScene("Victory");
            }
        };
        winPanel.Show
    }
}
else
{
    StartCoroutine(DelayedNextLevel());
}
}
private IEnumerator DelayedNextLevel()
{
    yield return new WaitForSeconds(0.8f);
    if (GameManager.Instance != null)

```

```

    GameManager.Instance.NextLevel();
}
}

```

Далее необходимо создать пустой компонент и дать название LevelManager и прикрепить скрипт LevelManager. В инспекторе объекта необходимо указать индекс уровня 0, добавить изображение персонажа, написать имя персонажа, добавить панель с предметами, и панель с поздравлением (см. рис. 13).

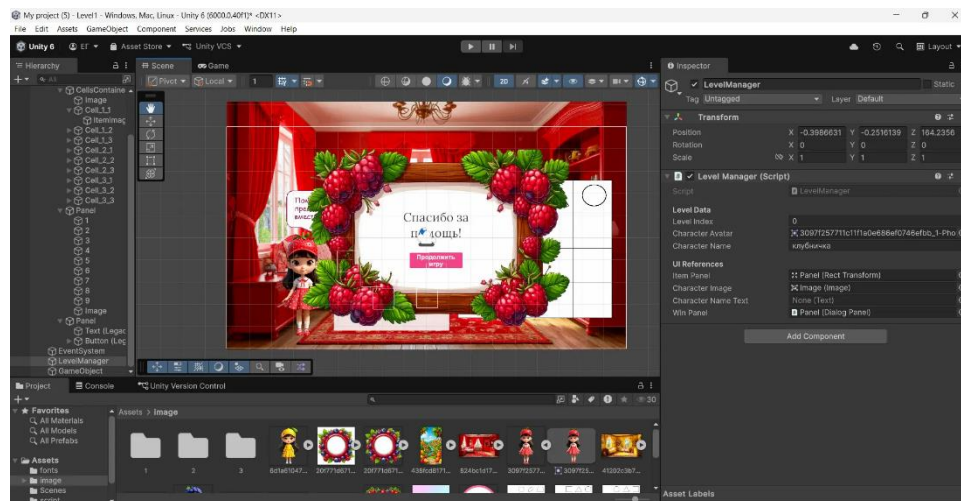


Рисунок 13 – Настройка сцены уровня

Две остальные сцены Level2 и Level3 создаются аналогично первой, но только меняются предметы и их ячейки в инспекторе, сетка куда необходимо перенести предметы, фон игры, персонаж и обращения персонажа. Также в объекте LevelManager необходимо указать индекс уровня 1 для второго уровня и индекс 2 для третьего уровня (см. рис. 14. 1, 14. 2).

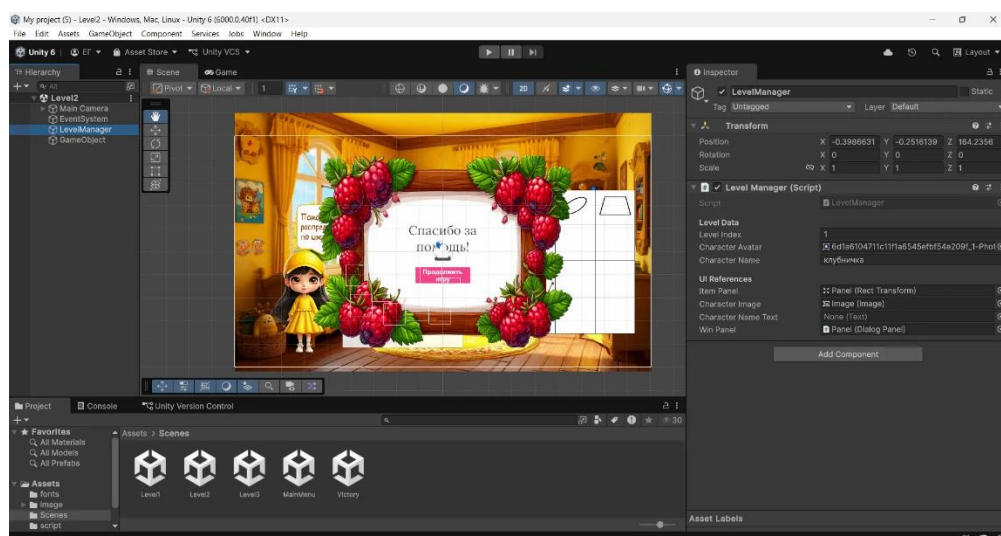


Рисунок 14.1 – Сцена Level2

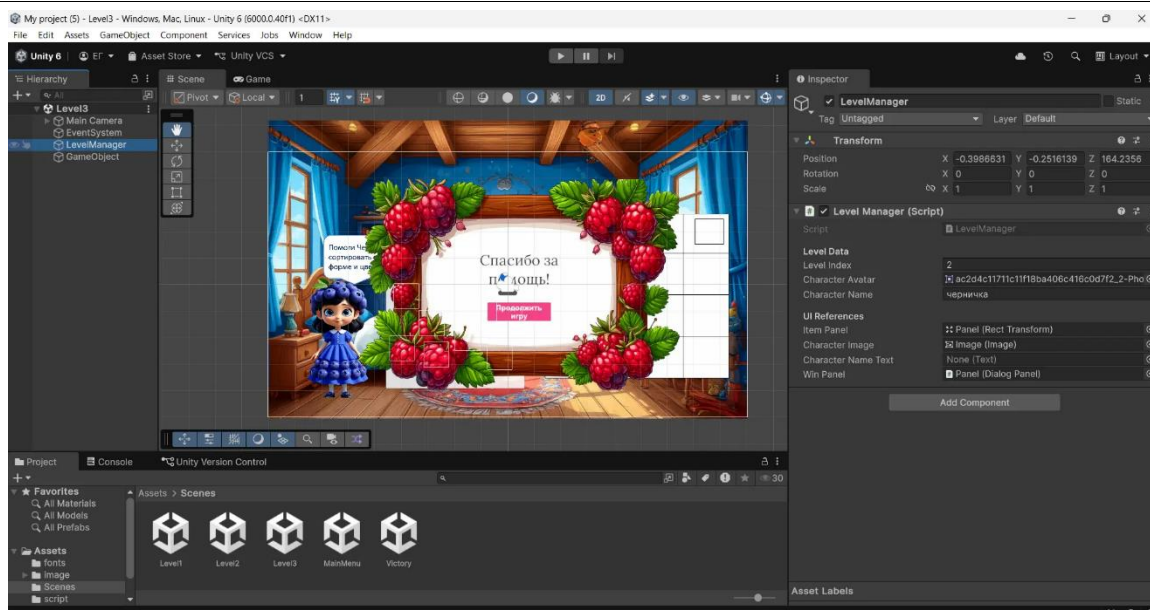


Рисунок 14.2 – Сцена Level3

После создания уровней игры можно приступить к сцене финиша, для этого необходимо открыть сцену Victory и настроить Canvas и фон как на первой сцене. На canvas необходимо разместить объект Panel. В панели надо создать объект Text, Button и три объекта image. Для того чтобы кнопка работала необходимо создать скрипт VictoryScreen который обрабатывает финальную сцену, проигрывает звук победы и по нажатию кнопки «Играть ещё раз» перезапускает игру с первого уровня.

Скрипт VictoryScreen:

```
using UnityEngine;
using UnityEngine.SceneManagement;
public class VictoryScreen : MonoBehaviour
{
    public void PlayAgain()
    {
        if (GameManager.Instance != null)
        {
            GameManager.Instance.LoadLevel(0);
        }
        else
        {
            SceneManager.LoadScene("MainMenu");
        }
    }
    public void QuitGame()
    {
        Application.Quit();
    }
}
```

Скрипт можно прикрепить к объекту Canvas. В инспекторе Button в событие OnClick() необходимо добавить объект Canvas, и выбрать VictoryScreen.PlayAgain (см. рис. 15).

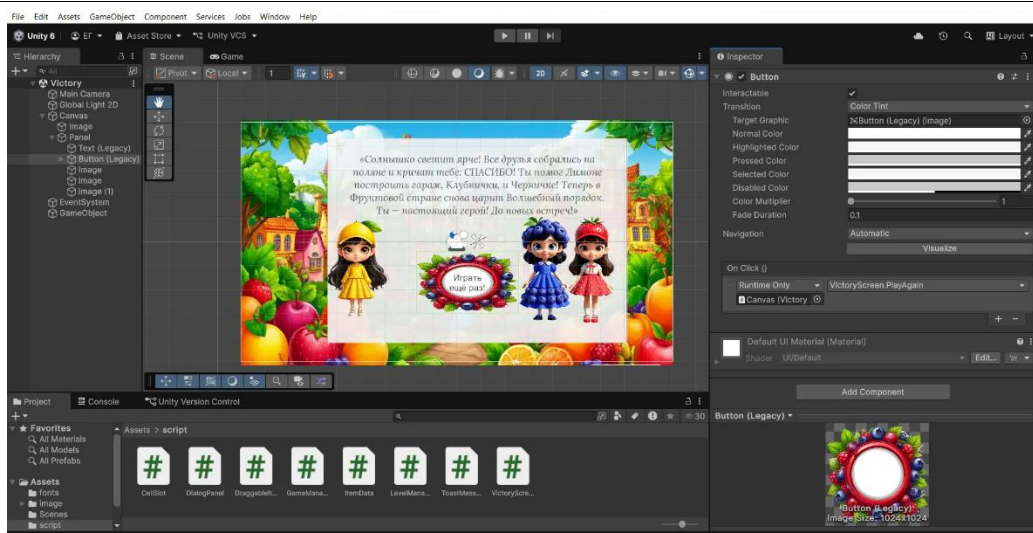


Рисунок 15 – Сцена финиша

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



Рисунок 16.1 – Готовая игра



Рисунок 16.2 – Готовая игра

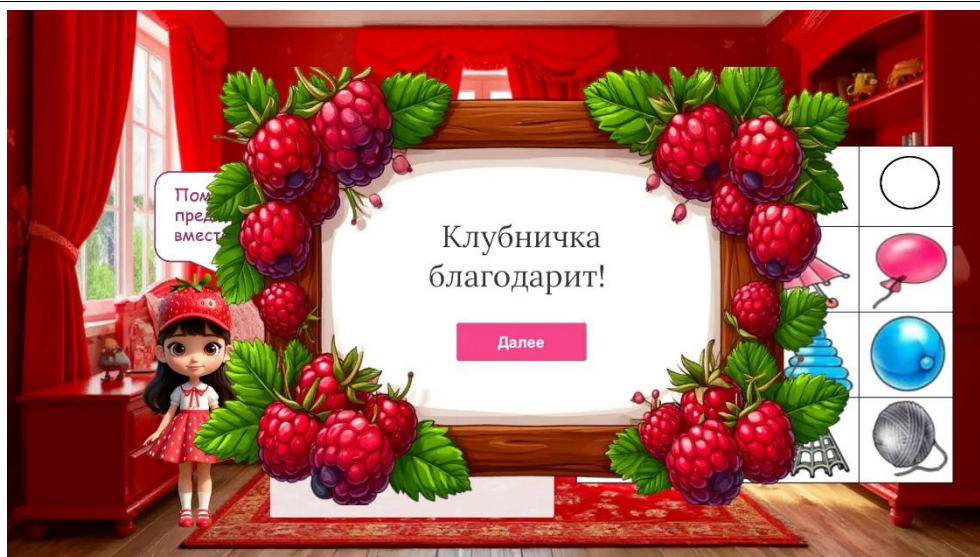


Рисунок 16.3 – Готовая игра



Рисунок 16.4 – Готовая игра

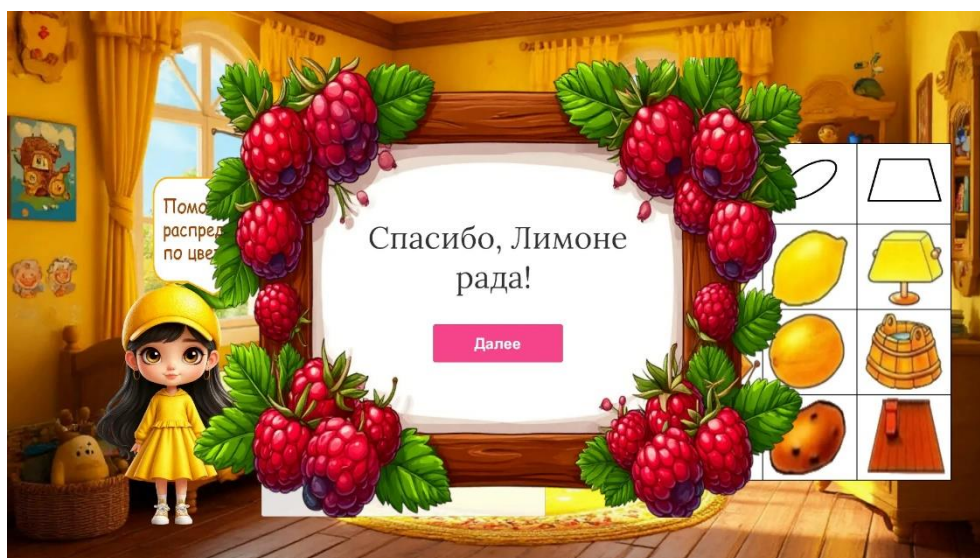


Рисунок 16.5 – Готовая игра

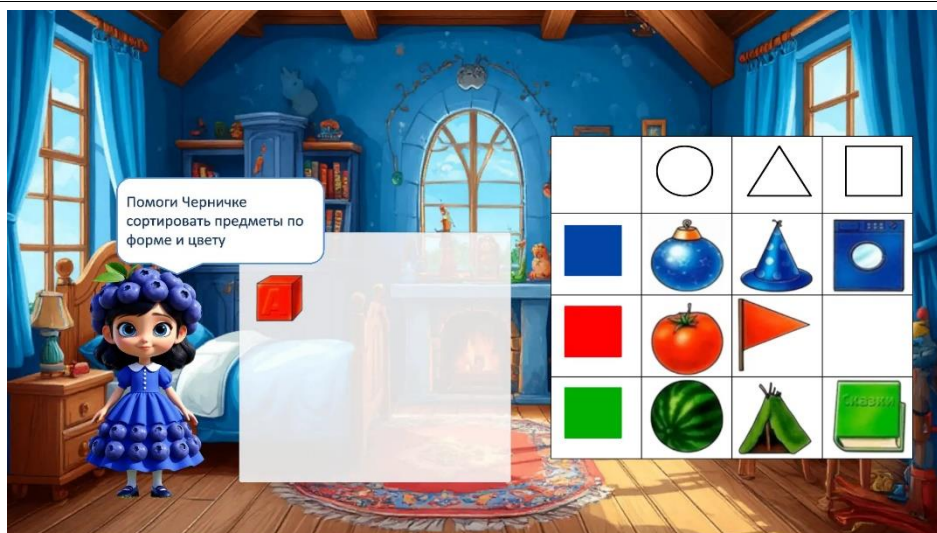


Рисунок 16.6 – Готовая игра

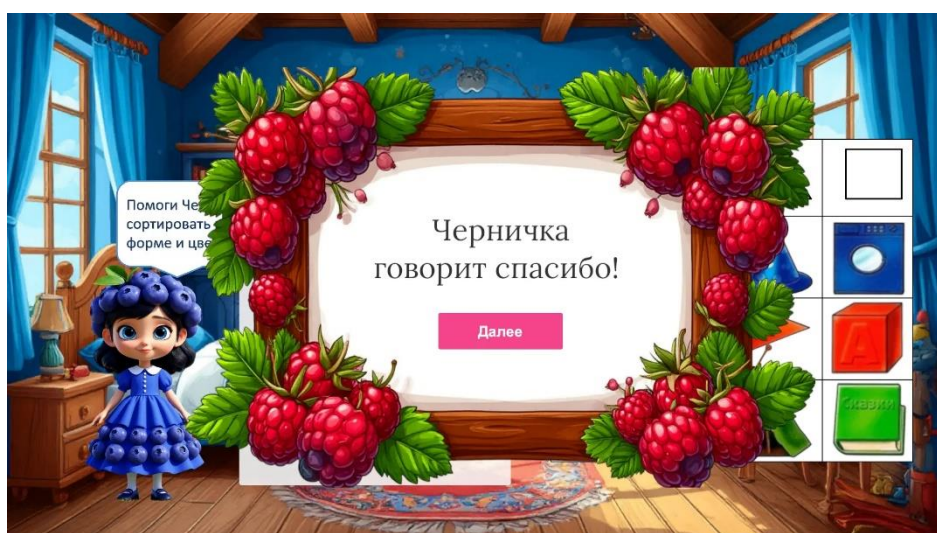


Рисунок 16.7 – Готовая игра



Рисунок 16.8 – Готовая игра

4 Выводы

В данной работе показан процесс создания компьютерной игры «Сказочное путешествие: геометрические фигуры» 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>