

Создание компьютерной игры «Математическая цепочка» на Unity

Голубева Евгения Павловна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Цель статьи – создание компьютерной игры «Математическая цепочка» 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 «Математическая цепочка».

Ключевые слова: Unity, игра, математика, цепочка.

Creating a computer game "Mathematical Chain" using Unity

Golubeva Evgeniya Pavlovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The purpose of the article is to create a computer game "Mathematical Chain" using Unity. 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 computer game called "Mathematical Chain."

Keywords: Unity, game, mathematics, chain.

1 Введение

1.1 Актуальность

Современное образование переживает этап глубокой цифровой трансформации, которая предполагает применение в обучении цифровых и смешанных технологий, разработку интерактивных форм обучения и проектирование новых цифровых средств обучения. В контексте преподавания математики цифровизация открывает практически неограниченные возможности для визуализации абстрактных понятий, делая образовательный процесс более интересным и доступным. Однако, несмотря на наличие широкого спектра цифровых инструментов - электронных курсов, игровых платформ, образовательных квестов и обучающих программ, - сохраняется острая потребность в разработке новых, специализированных средств обучения, которые одновременно способствовали бы развитию

математических умений учащихся и соответствовали запросам цифрового поколения к форме подачи информации.

Противоречие между традиционными методами преподавания математики и ожиданиями современных учащихся является одним из ключевых вызовов образования. Математика часто воспринимается школьниками как сложный и неинтересный предмет, а учащиеся нередко осваивают арифметические операции, но испытывают трудности с их применением в реальных жизненных ситуациях. В этих условиях геймификация обучения математике выступает эффективной технологией реализации цифровой трансформации, позволяющей развить у школьников математические умения посредством игровых цифровых технологий. Как справедливо отмечается в современных исследованиях, игровые механики в образовании - это не инфантилизация, а его актуализация.

Особую значимость приобретает разработка цифровых дидактических игр, поскольку они стимулируют когнитивную активность, развивают логическое мышление и повышают мотивацию к обучению. Исследования подтверждают потенциал таких игр в формировании у школьников математических и логических умений, а также познавательной мотивации. Смена видов учебной деятельности в рамках игрового процесса позволяет расширить спектр формируемых умений, сделать обучение более динамичным и интересным, что способствует непроизвольному усвоению знаний.

Среди инструментов для создания образовательных игр платформа Unity зарекомендовала себя как одно из наиболее эффективных решений. Современные средства Unity дают возможность «оживить» задания на развитие логики, смекалки, памяти и пространственного мышления, а продуманная адаптивность и индивидуализация уровней сложности позволяют использовать игры для детей с разным уровнем математической подготовки. Разработанные на Unity образовательные игры показывают критерии валидности, практичности и эффективности при обучении математике.

Таким образом, актуальность настоящего исследования обусловлена необходимостью разработки и внедрения новых цифровых средств обучения математике, отвечающих вызовам цифровой трансформации образования и запросам современных учащихся. Создание компьютерной игры «Математическая цепочка» на платформе Unity представляет собой решение, направленное на преодоление указанного противоречия между традиционными методами преподавания и потребностями цифрового поколения, а также на повышение мотивации и эффективности обучения математике через использование современных игровых технологий.

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 «Математическая цепочка» при помощи кроссплатформенной среды для создания интерактивных 2D и 3D приложений Unity.

2 Материалы и методы

Разработка игры «Признаки делимости» ведется на платформе Unity. Данный выбор продиктован простотой освоения инструмента, интуитивно понятным интерфейсом и наличием встроенной поддержки языка C#. Для написания и отладки скриптов задействована среда Visual Studio Community.

3 Результаты

Начальный этап разработки включает создание проекта в Unity. Для этого в Unity Hub выбирается шаблон «Universal 2D» (рендеринг по универсальному конвейеру), после чего нажимается кнопка «Создать проект» (см. рис. 1).

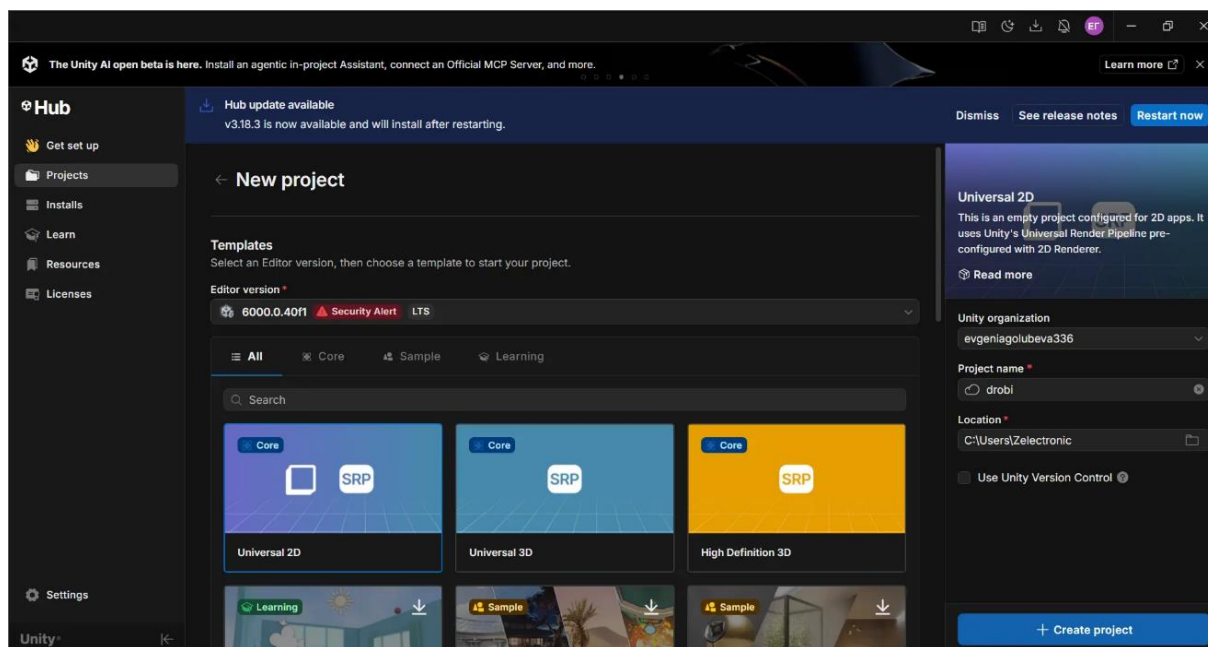


Рисунок 1 – Создание проекта

Когда проект с шаблоном Universal 2D уже создан, следует провести его первичную настройку под целевую платформу компьютер. Для начала необходимо перейти в меню File -> Build Profiles и изменить платформу на Windows (см. рис. 2).

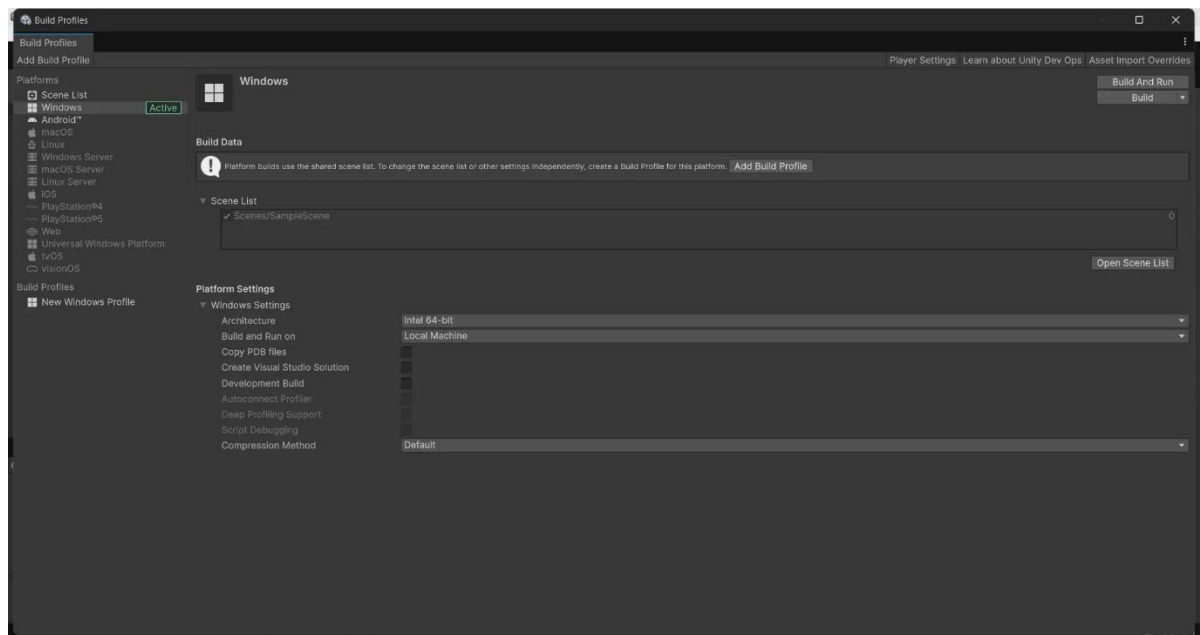


Рисунок 2 – Настройка проекта

В папку «image» следует поместить предварительно подготовленные графические ресурсы, которые будут использоваться в игре (см. рис. 3).

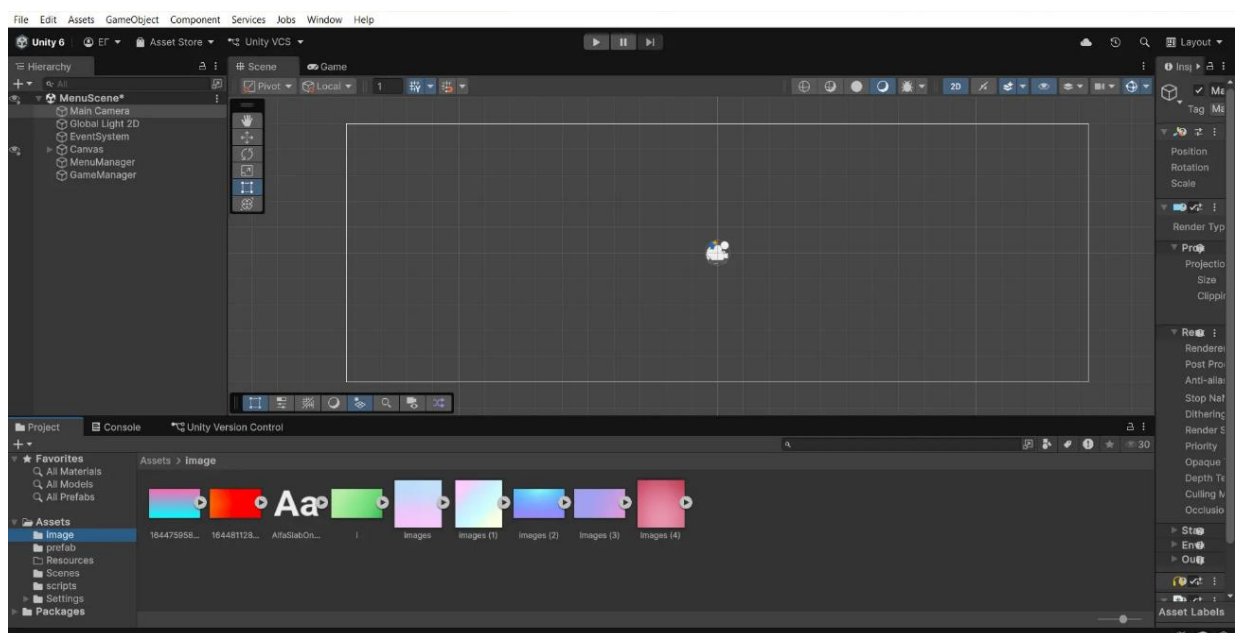


Рисунок 3 – Добавление изображений

В проекте уже присутствует одна сцена. Первым делом требуется создать объект Canvas - базовый контейнер для всех элементов интерфейса. Для этого в окне Hierarchy вызывается контекстное меню правой кнопкой мыши и выбирается UI -> Canvas. Чтобы интерфейс отображался корректно, необходимо настроить созданный Canvas. В инспекторе выполняются следующие действия: в поле Render Mode выбирается Screen Space - Camera; в поле Render Camera перетаскивается камера из иерархии; в Order in Layer

устанавливается значение 1; в Reference Resolution задаётся разрешение 1920×1080 (см. рис. 4).

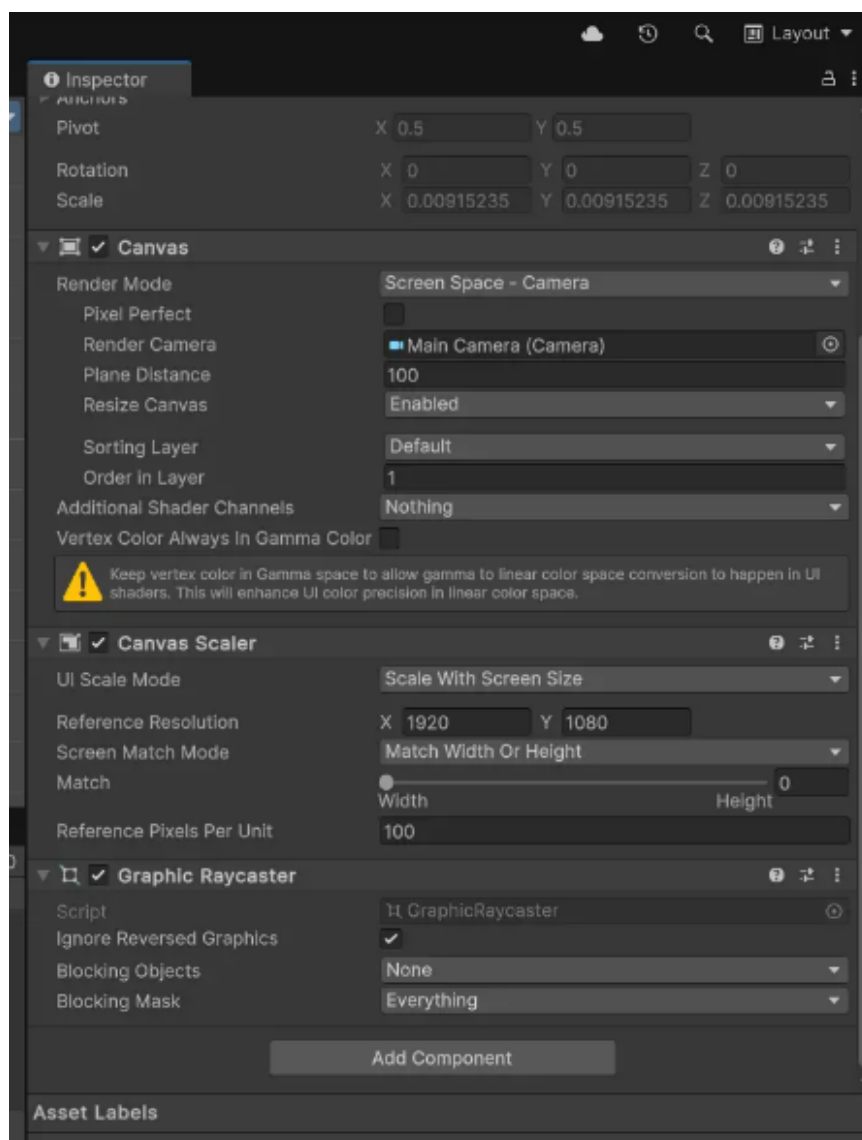


Рисунок 4 – Настройка Canvas

На созданном Canvas размещается объект Image, который будет выполнять функцию фонового изображения игры. После его добавления в инспекторе необходимо настроить позицию, масштаб и привязки (якоря) так, чтобы фон полностью покрывал область видимости камеры. Далее в разделе Image инспектора следует назначить заранее подготовленный графический файл фона (см. рис. 5).

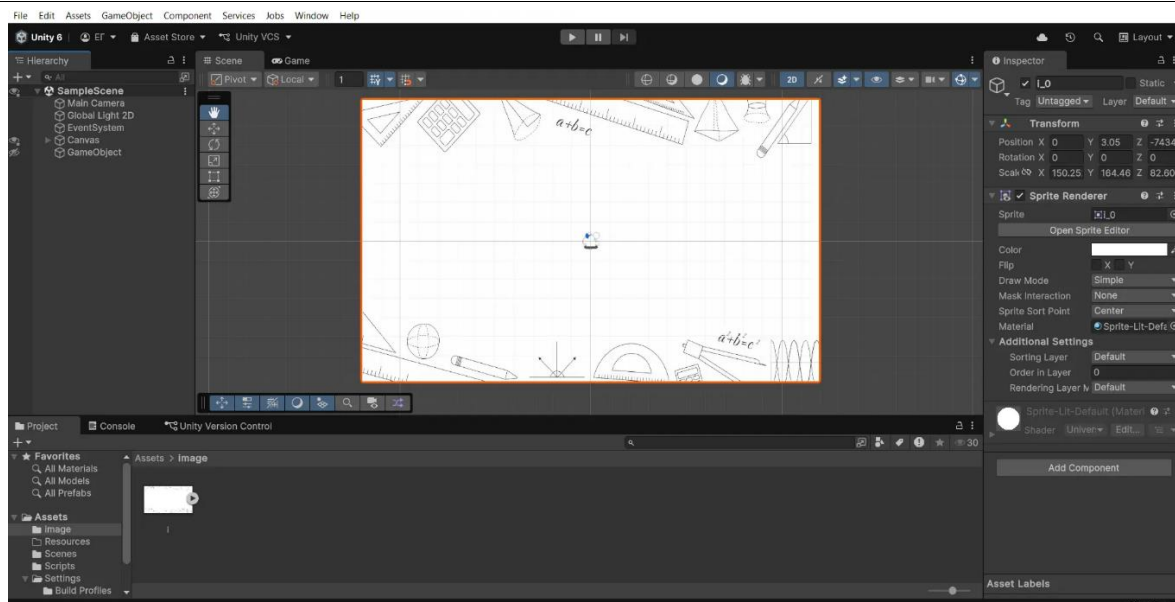


Рисунок 5 – Добавление фона

После создание фона, необходимо создать две панели, для этого в Hierarchy необходимо нажать правой кнопкой мыши для вызова меню в пространстве иерархии и выбрать UI -> Panel, и дать название панелям HeaderPanel и GamePanel. В HeaderPanel необходимо добавить объект Text и в инспекторе произвести следующие настройки: в окне Text набрать текст «Математическая цепочка», в Font Style выбрать Normal, размер текста 80, цвет текста черный (см. рис. 6).

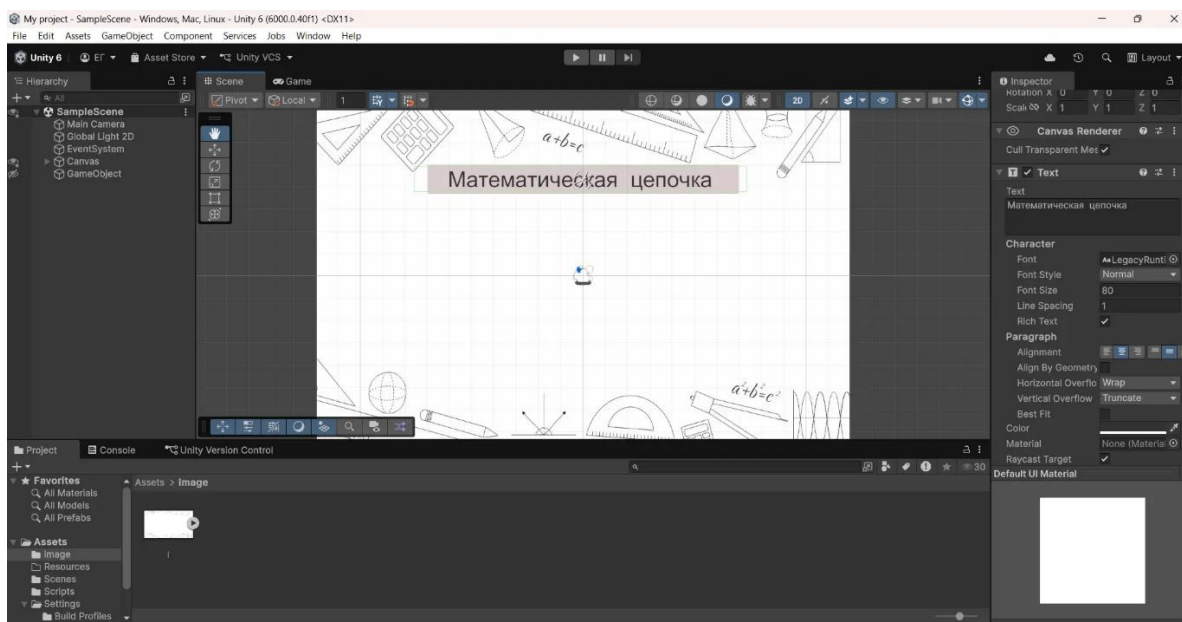


Рисунок 6 – Настройка объекта Text

В панель GamePanel необходимо добавить два объекта Text, один объект Button и один объект Dropdown. В первом объекте Text в инспекторе произвести следующие настройки: в окне Text набрать текст «Счет:», в Font Style выбрать Normal, размер текста 70, цвет текста черный. Во втором объекте

Text в инспекторе произвести следующие настройки: в окне Text не надо набирать текст, т. к. автоматически там будет появляться счет игры, в Font Style выбрать Normal, размер текста 70, цвет текста черный. В объекте Button есть автоматически созданный объект Text, и в этом объекте надо произвести настройки в инспекторе: в окне Text набрать текст «Новая цепочка», в Font Style выбрать Normal, размер текста 34, цвет текста черный. В Dropdown необходимо в инспекторе в пункте Options создать варианты выпадающего списка (см. рис. 7).

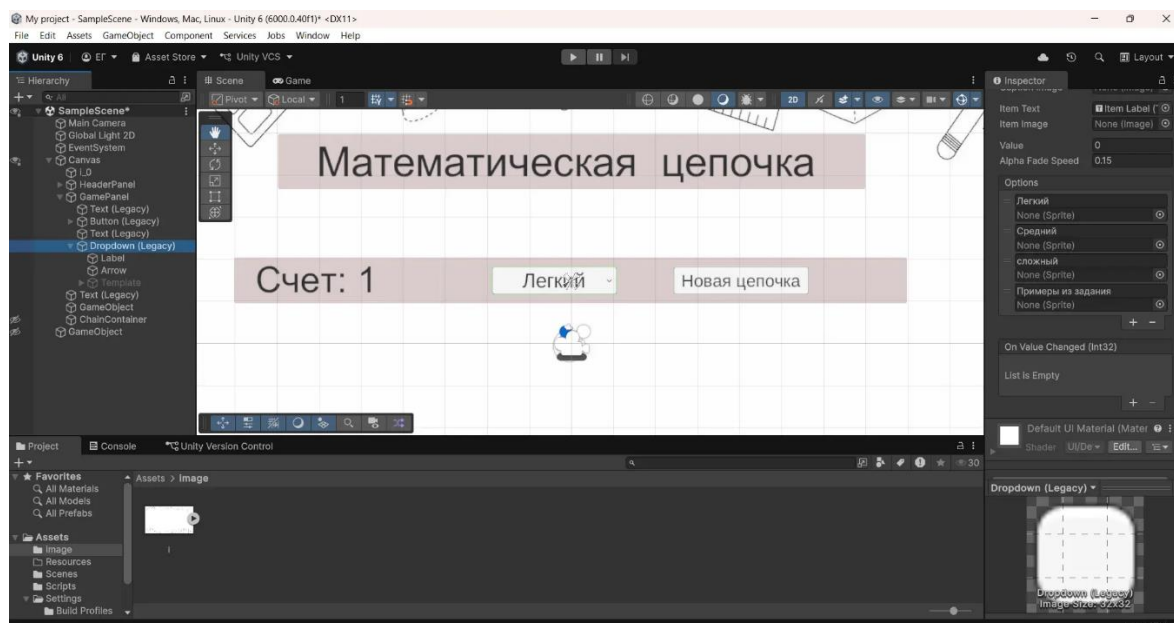


Рисунок 7 – Настройка панели

Далее необходимо создать скрипт для звукового сопровождения AudioManager.

Скрипт AudioManager:

```
using UnityEngine;
public class AudioManager : MonoBehaviour
{
    public static AudioManager Instance { get; private set; }
    private AudioSource audioSource;
    private void Awake()
    {
        if (Instance == null)
            Instance = this;
        else
            Destroy(gameObject);
        audioSource = GetComponent();
        if (audioSource == null)
            audioSource = gameObject.AddComponent();
    }
    private AudioClip CreateTone(float frequency, float duration, float amplitude = 0.3f)
    {
        int sampleRate = 44100;
        int sampleCount = Mathf.RoundToInt(sampleRate * duration);
        float[] samples = new float[sampleCount];
        for (int i = 0; i < sampleCount; i++)
        {
            samples[i] = amplitude * Mathf.Sin(2 * Mathf.PI * frequency * i / sampleRate);
        }
    }
}
```

```

        AudioClip clip = AudioClip.Create("Tone", sampleCount, 1, sampleRate, false);
        clip.SetData(samples, 0);
        return clip;
    }
    public void PlayCorrect()
    {
        AudioClip clip = CreateTone(523.25f, 0.15f);
        audioSource.PlayOneShot(clip);
        Invoke(nameof(PlayCorrectSecond), 0.1f);
    }
    private void PlayCorrectSecond()
    {
        AudioClip clip = CreateTone(1046.5f, 0.15f);
        audioSource.PlayOneShot(clip);
    }
    public void PlayError()
    {
        AudioClip clip = CreateTone(150f, 0.2f, 0.4f);
        audioSource.PlayOneShot(clip);
    }
    public void PlayWin()
    {
        float[] freqs = { 523.25f, 659.25f, 783.99f, 1046.5f };
        float duration = 0.15f;
        for (int i = 0; i < freqs.Length; i++)
        {
            int index = i;
            Invoke(nameof(PlayWinNote), i * 0.12f);
        }
    }
    private void PlayWinNote()
    {
    }
    public void PlayWinArpeggio()
    {
        float[] freqs = { 523.25f, 659.25f, 783.99f, 1046.5f };
        float duration = 0.15f;
        float gap = 0.1f;
        int sampleRate = 44100;
        int totalSamples = Mathf.RoundToInt(sampleRate * (duration + gap) *
freqs.Length);
        float[] samples = new float[totalSamples];
        for (int i = 0; i < freqs.Length; i++)
        {
            int startSample = Mathf.RoundToInt(sampleRate * (duration + gap) * i);
            int endSample = startSample + Mathf.RoundToInt(sampleRate * duration);
            for (int s = startSample; s < endSample && s < totalSamples; s++)
            {
                float t = (s - startSample) / (float)sampleRate;
                samples[s] = 0.3f * Mathf.Sin(2 * Mathf.PI * freqs[i] * t);
            }
        }
        AudioClip clip = AudioClip.Create("Win", totalSamples, 1, sampleRate, false);
        clip.SetData(samples, 0);
        audioSource.PlayOneShot(clip);
    }
}

```

Необходимо создать пустой объект `GameObject` для звукового сопровождения игры и добавить скрипт `AudioManager`, и через `add component` добавить `AudioSource` (см. рис. 8).

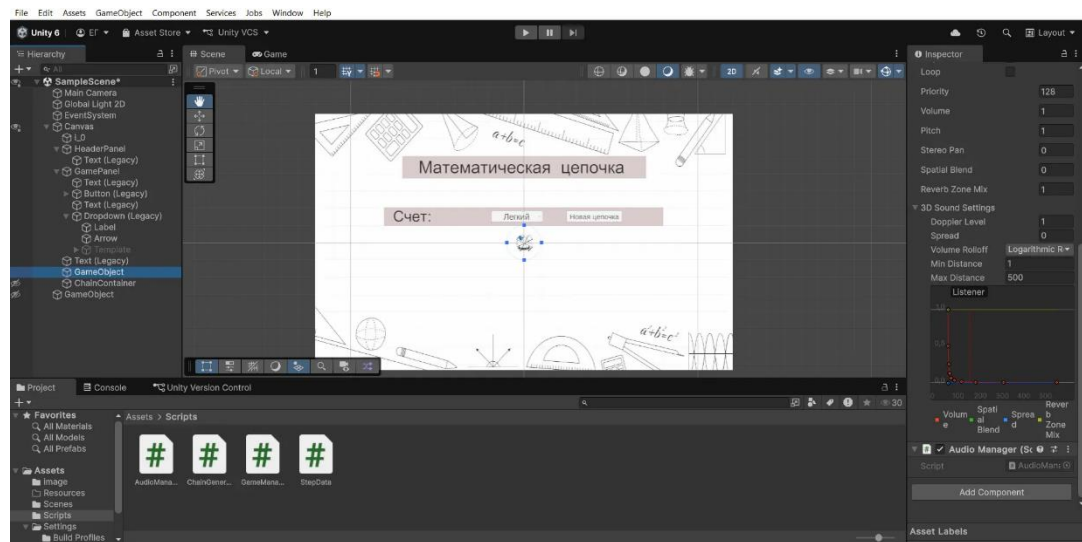


Рисунок 8 – Настройка объекта

Для математической цепочки необходимо создать пустой GameObject и переименовать в ChainContainer. Для механики игры необходимо создать три скрипта ChainGenerator отвечающий за генерацию цепочек, GameManager управляет всей логикой игры и StepData отвечает за модель данных.

Скрипт ChainGenerator:

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

public static class ChainGenerator
{
    private static readonly List<List<StepData>> presetChains = new List<List<StepData>>()
    {
        new List<StepData>
        {
            new StepData { Type = StepData.NodeType.Start, Value = 5 },
            new StepData { Type = StepData.NodeType.Step, Operation = "x60", Value = 300 },
            new StepData { Type = StepData.NodeType.Step, Operation = ":100", Value = 3 },
            new StepData { Type = StepData.NodeType.Step, Operation = "x140", Value = 420 },
            new StepData { Type = StepData.NodeType.Step, Operation = ":2", Value = 210 },
            new StepData { Type = StepData.NodeType.Step, Operation = "x4", Value = 840 },
            new StepData { Type = StepData.NodeType.Step, Operation = ":8", Value = 105 },
            new StepData { Type = StepData.NodeType.Step, Operation = "x120", Value = 12600 },
            new StepData { Type = StepData.NodeType.Step, Operation = ":24", Value = 525 }
        },
        new List<StepData>
        {
            new StepData { Type = StepData.NodeType.Start, Value = 30 },
            new StepData { Type = StepData.NodeType.Step, Operation = "+20", Value = 50 },
            new StepData { Type = StepData.NodeType.Step, Operation = ":2", Value = 25 },
            new StepData { Type = StepData.NodeType.Step, Operation = ":5", Value = 5 },
            new StepData { Type = StepData.NodeType.Step, Operation = "+19", Value = 24 }
        },
        new List<StepData>
        {
            new StepData { Type = StepData.NodeType.Start, Value = 6 },
            new StepData { Type = StepData.NodeType.Step, Operation = "x50", Value = 300 },
            new StepData { Type = StepData.NodeType.Step, Operation = ":3", Value = 100 },
            new StepData { Type = StepData.NodeType.Step, Operation = "x24", Value = 2400 },
            new StepData { Type = StepData.NodeType.Step, Operation = ":12", Value = 200 },
            new StepData { Type = StepData.NodeType.Step, Operation = "x18", Value = 3600 },
            new StepData { Type = StepData.NodeType.Step, Operation = ":120", Value = 30 },
            new StepData { Type = StepData.NodeType.Step, Operation = "x170", Value = 5100 },
            new StepData { Type = StepData.NodeType.Step, Operation = ":100", Value = 51 }
        },
        new List<StepData>
        {
            new StepData { Type = StepData.NodeType.Start, Value = 91 },
            new StepData { Type = StepData.NodeType.Step, Operation = "-73", Value = 18 },
            new StepData { Type = StepData.NodeType.Step, Operation = ":9", Value = 2 },

```

```

        new StepData { Type = StepData.NodeType.Step, Operation = "×7", Value = 14 },
        new StepData { Type = StepData.NodeType.Step, Operation = "+58", Value = 72 },
        new StepData { Type = StepData.NodeType.Step, Operation = ":8", Value = 9 },
        new StepData { Type = StepData.NodeType.Step, Operation = "×3", Value = 27 },
        new StepData { Type = StepData.NodeType.Step, Operation = "+56", Value = 83 },
        new StepData { Type = StepData.NodeType.Step, Operation = "-47", Value = 36 },
        new StepData { Type = StepData.NodeType.Step, Operation = ":6", Value = 6 },
        new StepData { Type = StepData.NodeType.Step, Operation = "×5", Value = 30 },
        new StepData { Type = StepData.NodeType.Step, Operation = "-14", Value = 16 },
        new StepData { Type = StepData.NodeType.Step, Operation = ":4", Value = 4 },
        new StepData { Type = StepData.NodeType.Step, Operation = "×7", Value = 28 },
        new StepData { Type = StepData.NodeType.Step, Operation = "+35", Value = 63 },
        new StepData { Type = StepData.NodeType.Step, Operation = "-24", Value = 39 },
        new StepData { Type = StepData.NodeType.Step, Operation = "+10", Value = 49 },
        new StepData { Type = StepData.NodeType.Step, Operation = ":7", Value = 7 }
    }
};

public static List<StepData> GetPresetChain()
{
    int index = Random.Range(0, presetChains.Count);
    return presetChains[index];
}

public static List<StepData> GenerateRandomChain(string difficulty)
{
    int length = 5;
    switch (difficulty)
    {
        case "easy": length = Random.Range(3, 5); break; // 3-4
        case "medium": length = Random.Range(5, 8); break; // 5-7
        case "hard": length = Random.Range(8, 13); break; // 8-12
        default: length = 5; break;
    }

    List<StepData> chain = new List<StepData>();
    int currentVal = Random.Range(2, 22); // 2..21
    chain.Add(new StepData { Type = StepData.NodeType.Start, Value = currentVal });
    string[] ops = { "+", "-", "×", ":" };
    for (int i = 0; i < length; i++)
    {
        string op = ops[Random.Range(0, ops.Length)];
        int nextVal = currentVal;
        string opStr = "";
        if (op == "+")
        {
            int add = Random.Range(1, 51);
            nextVal = currentVal + add;
            opStr = $"{add}";
        }
        else if (op == "-")
        {
            int sub = Random.Range(1, currentVal);
            nextVal = currentVal - sub;
            opStr = $"{sub}";
        }
        else if (op == "×")
        {
            int mul = Random.Range(2, 11);
            nextVal = currentVal * mul;
            opStr = $"{mul}";
        }
        else if (op == ":")
        {
            // Поиск делителей
            List<int> divisors = new List<int>();
            for (int d = 2; d <= currentVal; d++)
            {
                if (currentVal % d == 0) divisors.Add(d);
            }
            if (divisors.Count > 0)
            {
                int div = divisors[Random.Range(0, divisors.Count)];
                nextVal = currentVal / div;
                opStr = $"{div}";
            }
            else
            {
                // если нет делителей (простое) - меняем на сложение
                int add = Random.Range(1, 21);
                nextVal = currentVal + add;
                opStr = $"{add}";
            }
        }
    }
}

```

```

    }
    chain.Add(new StepData { Type = StepData.NodeType.Step, Operation = opStr, Value =
nextVal });
    currentVal = nextVal;
    }
    return chain;
}
}
}

```

Скрипт GameManager:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
public class GameManager : MonoBehaviour
{
    private const int NODE_SIZE = 120;
    private const int ARROW_WIDTH = 100; // ширина стрелки
    private const int ARROW_HEIGHT = 80; // высота стрелки
    private const int FONT_SIZE_START = 48; // шрифт в стартовом узле
    private const int FONT_SIZE_INPUT = 44; // шрифт в поле ввода
    private const int FONT_SIZE_PLACEHOLDER = 44; // шрифт placeholder
    private const int FONT_SIZE_ARROW = 36; // шрифт в стрелке
    private const float CONTAINER_WIDTH = 2000; // ширина контейнера
    private const float CONTAINER_HEIGHT = 180; // высота контейнера
    private const float SPACING = 12f; // отступ между элементами
    public static GameManager Instance { get; private set; }
    [Header("UI References")]
    public Text scoreText;
    public Dropdown difficultyDropdown;
    public Button newGameButton;
    public Text messageText;
    public Transform chainContainer;
    private List<StepData> currentChain = new List<StepData>();
    private int score = 0;
    private List<InputField> inputFields = new List<InputField>();
    private Font _font;
    private Dictionary<InputField, (Image bg, Image border)> nodeData = new Dictionary<InputField,
(Image bg, Image border)>();
    private Sprite circleSprite;
    private Sprite roundedRectSprite;
    private void Awake()
    {
        if (Instance == null)
            Instance = this;
        else
            Destroy(gameObject);
        _font = Resources.GetBuiltinResource<Font>("LegacyRuntime.ttf");
        if (_font == null)
        {
            _font = Font.CreateDynamicFontFromOSFont("Arial", 32);
            Debug.LogWarning("LegacyRuntime.ttf не найден, используем системный Arial");
        }
        CreateSprites();
        SetupChainContainer();
    }
    private void CreateSprites()
    {
        Texture2D circleTex = new Texture2D(128, 128);
        Color[] pixels = new Color[128 * 128];
        Vector2 center = new Vector2(64, 64);
        float radius = 60;
        for (int y = 0; y < 128; y++)
        {
            for (int x = 0; x < 128; x++)
            {
                float dist = Vector2.Distance(new Vector2(x, y), center);
                pixels[y * 128 + x] = (dist <= radius) ? Color.white : Color.clear;
            }
        }
        circleTex.SetPixels(pixels);
        circleTex.Apply();
        circleSprite = Sprite.Create(circleTex, new Rect(0, 0, 128, 128), new Vector2(0.5f, 0.5f),
100);
        Texture2D rectTex = new Texture2D(128, 128);
        Color[] rectPixels = new Color[128 * 128];
        int cornerRadius = 20;
        for (int y = 0; y < 128; y++)

```

```

    {
        for (int x = 0; x < 128; x++)
        {
            bool inside = true;
            if (x < cornerRadius && y < cornerRadius)
                inside = (x - cornerRadius) * (x - cornerRadius) + (y - cornerRadius) * (y - cornerRadius) <= cornerRadius * cornerRadius;
            else if (x >= 128 - cornerRadius && y < cornerRadius)
                inside = (x - (128 - cornerRadius)) * (x - (128 - cornerRadius)) + (y - cornerRadius) * (y - cornerRadius) <= cornerRadius * cornerRadius;
            else if (x < cornerRadius && y >= 128 - cornerRadius)
                inside = (x - cornerRadius) * (x - cornerRadius) + (y - (128 - cornerRadius)) * (y - (128 - cornerRadius)) <= cornerRadius * cornerRadius;
            else if (x >= 128 - cornerRadius && y >= 128 - cornerRadius)
                inside = (x - (128 - cornerRadius)) * (x - (128 - cornerRadius)) + (y - (128 - cornerRadius)) * (y - (128 - cornerRadius)) <= cornerRadius * cornerRadius;
            else
                inside = true;
            rectPixels[y * 128 + x] = inside ? Color.white : Color.clear;
        }
        rectTex.SetPixels(rectPixels);
        rectTex.Apply();
        roundedRectSprite = Sprite.Create(rectTex, new Rect(0, 0, 128, 128), new Vector2(0.5f, 0.5f), 100);
    }
    private void SetupChainContainer()
    {
        if (chainContainer == null) return;
        HorizontalLayoutGroup layout = chainContainer.GetComponent<HorizontalLayoutGroup>();
        if (layout == null)
            layout = chainContainer.gameObject.AddComponent<HorizontalLayoutGroup>();
        layout.childAlignment = TextAnchor.MiddleCenter;
        layout.spacing = SPACING;
        layout.childForceExpandWidth = false;
        layout.childForceExpandHeight = false;
        layout.childControlWidth = false;
        layout.childControlHeight = false;
        RectTransform rect = chainContainer as RectTransform;
        if (rect != null)
        {
            rect.anchorMin = new Vector2(0.5f, 0.5f);
            rect.anchorMax = new Vector2(0.5f, 0.5f);
            rect.pivot = new Vector2(0.5f, 0.5f);
            rect.sizeDelta = new Vector2(CONTAINER_WIDTH, CONTAINER_HEIGHT);
        }
    }
    private void Start()
    {
        if (newGameButton == null)
            Debug.LogError("newGameButton не назначен!");
        else
            newGameButton.onClick.AddListener(StartNewGame);
        if (difficultyDropdown == null)
            Debug.LogError("difficultyDropdown не назначен!");
        else
            difficultyDropdown.onValueChanged.AddListener(delegate { StartNewGame(); });
        StartNewGame();
    }
    public void StartNewGame()
    {
        if (chainContainer == null)
        {
            Debug.LogError("chainContainer = null");
            return;
        }
        foreach (Transform child in chainContainer)
            Destroy(child.gameObject);
        inputFields.Clear();
        nodeData.Clear();
        HideMessage();
        string diff = "Средний";
        if (difficultyDropdown != null && difficultyDropdown.options.Count > 0)
            diff = difficultyDropdown.options[difficultyDropdown.value].text;
        if (diff == "Примеры из задания")
            currentChain = ChainGenerator.GetPresetChain();
        else
            currentChain = ChainGenerator.GenerateRandomChain(diff.ToLower());
        BuildChainUI();
        if (inputFields.Count > 0)

```

```

        StartCoroutine(SelectNextFrame(inputFields[0].gameObject));
        LayoutRebuilder.ForceRebuildLayoutImmediate(chainContainer as RectTransform);
    }
    private IEnumerator SelectNextFrame(GameObject obj)
    {
        yield return null;
        if (obj != null && obj.activeInHierarchy)
            EventSystem.current?.SetSelectedGameObject(obj);
    }
    private IEnumerator DisableInputFieldNextFrame(InputField inputField)
    {
        yield return null;
        if (inputField != null && inputField.isActiveAndEnabled)
            inputField.interactable = false;
    }
    private void BuildChainUI()
    {
        for (int i = 0; i < currentChain.Count; i++)
        {
            StepData step = currentChain[i];
            if (step.Type == StepData.NodeType.Start)
                CreateStartNode(step.Value);
            else
            {
                CreateArrow(step.Operation);
                CreateInputNode(step.Value, i, i == currentChain.Count - 1);
            }
        }
    }
    private void CreateStartNode(int value)
    {
        GameObject node = new GameObject("StartNode", typeof(RectTransform), typeof(Image),
        typeof(LayoutElement));
        node.transform.SetParent(chainContainer, false);
        RectTransform rect = node.GetComponent<RectTransform>();
        rect.sizeDelta = new Vector2(NODE_SIZE, NODE_SIZE);
        LayoutElement le = node.GetComponent<LayoutElement>();
        le.preferredWidth = NODE_SIZE;
        le.preferredHeight = NODE_SIZE;
        Image img = node.GetComponent<Image>();
        img.sprite = roundedRectSprite;
        img.color = new Color(0.29f, 0.56f, 0.89f);
        GameObject textObj = new GameObject("ValueText", typeof(RectTransform), typeof(Text));
        textObj.transform.SetParent(node.transform, false);
        RectTransform textRect = textObj.GetComponent<RectTransform>();
        textRect.anchorMin = Vector2.zero
;
        textRect.anchorMax = Vector2.one
;
        textRect.offsetMin = Vector2.zero
;
        textRect.offsetMax = Vector2.zero
;
        Text text = textObj.GetComponent<Text>();
        text.text = value.ToString();
        text.fontSize = FONT_SIZE_START;
        text.fontStyle = FontStyle.Bold;
        text.alignment = TextAnchor.MiddleCenter;
        text.color = Color.white;
        text.font = _font;
    }
    private void CreateArrow(string operation)
    {
        GameObject arrow = new GameObject("Arrow", typeof(RectTransform), typeof(Text),
        typeof(LayoutElement));
        arrow.transform.SetParent(chainContainer, false);
        RectTransform rect = arrow.GetComponent<RectTransform>();
        rect.sizeDelta = new Vector2(ARROW_WIDTH, ARROW_HEIGHT);
        LayoutElement le = arrow.GetComponent<LayoutElement>();
        le.preferredWidth = ARROW_WIDTH;
        le.preferredHeight = ARROW_HEIGHT;
        Text txt = arrow.GetComponent<Text>();
        txt.text = operation;
        txt.fontSize = FONT_SIZE_ARROW;
        txt.alignment = TextAnchor.MiddleCenter;
        txt.color = new Color(0.6f, 0.6f, 0.6f);
        txt.font = _font;
    }
    private void CreateInputNode(int expectedValue, int stepIndex, bool isFinal)
    {

```

```

GameObject node = new GameObject("InputNode", typeof(RectTransform), typeof(Image),
typeof(LayoutElement));
node.transform.SetParent(chainContainer, false);
RectTransform rect = node.GetComponent<RectTransform>();
rect.sizeDelta = new Vector2(NODE_SIZE, NODE_SIZE);
LayoutElement le = node.GetComponent<LayoutElement>();
le.preferredWidth = NODE_SIZE;
le.preferredHeight = NODE_SIZE;
Image bg = node.GetComponent<Image>();
if (isFinal)
{
    bg.sprite = roundedRectSprite;
    bg.color = new Color(0.29f, 0.56f, 0.89f); // ГОЛУБОЙ (как стартовый)
}
else
{
    bg.sprite = circleSprite;
    bg.color = Color.white;
}
GameObject borderObj = new GameObject("Border", typeof(RectTransform), typeof(Image));
borderObj.transform.SetParent(node.transform, false);
RectTransform borderRect = borderObj.GetComponent<RectTransform>();
borderRect.anchorMin = Vector2.zero
;
borderRect.anchorMax = Vector2.one
;
borderRect.offsetMin = new Vector2(-4, -4);
borderRect.offsetMax = new Vector2(4, 4);
Image borderImg = borderObj.GetComponent<Image>();
borderImg.sprite = isFinal ? roundedRectSprite : circleSprite;
borderImg.color = isFinal ? new Color(0.29f, 0.56f, 0.89f) : new Color(0.96f, 0.65f, 0.14f);
borderImg.raycastTarget = false;
GameObject inputObj = new GameObject("InputField", typeof(RectTransform), typeof(Image),
typeof(InputField));
inputObj.transform.SetParent(node.transform, false);
RectTransform inputRect = inputObj.GetComponent<RectTransfor
m>();
inputRect.anchorMin = Vector2.zero
;
inputRect.anchorMax = Vector2.one
;
inputRect.offsetMin = new Vector2(8, 8);
inputRect.offsetMax = new Vector2(-8, -8);
Image inputImage = inputObj.GetComponent<Image>();
inputImage.color = new Color(1, 1, 1, 0);
InputField inputField = inputObj.GetComponent<InputField>();
inputField.targetGraphic = inputImage;
inputField.contentType = InputField.ContentType.IntegerNumber;
inputField.lineType = InputField.LineType.SingleLine;
GameObject textObj = new GameObject("Text", typeof(RectTransform), typeof(Text));
textObj.transform.SetParent(inputObj.transform, false);
RectTransform textRect = textObj.GetComponent<RectTransform>();
textRect.anchorMin = Vector2.zero;
textRect.anchorMax = Vector2.one;
textRect.offsetMin = Vector2.zero;
textRect.offsetMax = Vector2.zero;
Text textComponent = textObj.GetComponent<Text>();
textComponent.fontSize = FONT_SIZE_INPUT;
textComponent.fontStyle = FontStyle.Bold;
textComponent.alignment = TextAnchor.MiddleCenter;
textComponent.color = Color.black;
textComponent.font = _font;
inputField.textComponent = textComponent;
GameObject placeholderObj = new GameObject("Placeholder", typeof(RectTransform),
typeof(Text));
placeholderObj.transform.SetParent(inputObj.transform, false);
RectTransform phRect = placeholderObj.GetComponent<RectTransform>();
phRect.anchorMin = Vector2.zero
;
phRect.anchorMax = Vector2.one
;
phRect.offsetMin = Vector2.zero
;
phRect.offsetMax = Vector2.zero
;
Text phText = placeholderObj.GetComponent<Text>();
phText.text = "?";
phText.fontSize = FONT_SIZE_PLACEHOLDER;
phText.alignment = TextAnchor.MiddleCenter;
phText.color = Color.gray;

```

```

        phText.font = _font;
        inputField.placeholder = phText;
        inputField.name
= $"Input_{stepIndex}";
        inputField.onEndEdit.AddListener((string value) => ValidateInput(inputField, stepIndex));
        inputFields.Add(inputField);
        nodeData[inputField] = (bg, borderImg);
    }
    private void ValidateInput(InputField inputField, int stepIndex)
    {
        if (string.IsNullOrEmpty(inputField.text)) return;
        int userValue;
        if (!int.TryParse(inputField.text, out userValue))
        {
            inputField.text = "";
            return;
        }
        int expected = currentChain[stepIndex].Value;
        if (userValue == expected)
        {
            AudioManager.Instance?.PlayCorrect();
            StartCoroutine(DisableInputFieldNextFrame(inputField));
            if (nodeData.TryGetValue(inputField, out var data))
            {
                StartCoroutine(AnimateCorrect(data.bg
, data.border, inputField.transform));
            }
            int currentIdx = inputFields.IndexOf(inputField);
            if (currentIdx + 1 < inputFields.Count)
            {
                StartCoroutine(SelectNextFrame(inputFields[currentIdx + 1].gameObject));
            }
            else
            {
                AudioManager.Instance?.PlayWinArpeggio();
                score++;
                if (scoreText != null) scoreText.text = score.ToString();
                ShowMessage("🎉 Отлично! Вся цепочка решена верно!", true);
                StartCoroutine(Celebrate());
            }
        }
        else
        {
            AudioManager.Instance?.PlayError();
            inputField.text = "";
            if (nodeData.TryGetValue(inputField, out var data))
            {
                St
vector2.one
Domainpark - Bitte den Rasen nicht betreten. Vielen Dank!
Diese Domain ist zur Zeit keinem Server oder Webhosting zugewiesen. Sie können die Zuordnung der
Domain in Ihrem Kundenbereich (CCP) verwalten.artCoroutine(AnimateError(data.border,
inputField.transform));
            }
            StartCoroutine(SelectNextFrame(inputField.gameObject));
        }
    }
    private IEnumerator AnimateCorrect(Image bg, Image border, Transform nodeTransform)
    {
        Color origBg = bg.color;
        bg.color = new Color(0.49f, 0.83f, 0.13f);
        Color origBorder = border.color;
        border.color = new Color(0.49f, 0.83f, 0.13f);
        Vector3 originalScale = nodeTransform.localScale;
        float duration = 0.5f;
        float elapsed = 0f;
        while (elapsed < duration)
        {
            float progress = elapsed / duration;
            float scale = 1f + 0.1f * Mathf.Sin(progress * Mathf.PI * 4);
            nodeTransform.localScale = originalScale * scale;
            elapsed += Time.deltaTime;
            yield return null;
        }
        nodeTransform.localScale = originalScale;
    }
    private IEnumerator AnimateError(Image border, Transform nodeTransform)
    {
        Vector3 originalPos = nodeTransform.localPosition;

```

```

float duration = 0.5f;
float elapsed = 0f;
while (elapsed < duration)
{
    float offsetX = Mathf.Sin(elapsed * 60f) * 8f;
    nodeTransform.localPosition = originalPos + new Vector3(offsetX, 0, 0);
    elapsed += Time.deltaTime;
    yield return null;
}
nodeTransform.localPosition = originalPos;
Color origBorder = border.color;
border.color = new Color(0.82f, 0.01f, 0.11f);
yield return new WaitForSeconds(0.2f);
border.color = origBorder;
}
private IEnumerator Celebrate()
{
    Transform gameArea = chainContainer.parent;
    if (gameArea != null)
    {
        Vector3 originalScale = gameArea.localScale;
        for (int i = 0; i < 3; i++)
        {
            gameArea.localScale = originalScale * 1.05f;
            yield return new WaitForSeconds(0.15f);
            gameArea.localScale = originalScale;
            yield return new WaitForSeconds(0.15f);
        }
    }
}
private void ShowMessage(string text, bool isSuccess)
{
    if (messageText != null)
    {
        messageText.text = text;
        messageText.color = isSuccess ? new Color(0.49f, 0.83f, 0.13f) : new Color(0.82f,
0.01f, 0.11f);
        messageText.gameObject.SetActive(true);
    }
}
private void HideMessage()
{
    if (messageText != null)
        messageText.gameObject.SetActive(false);
}
}

```

Скрипт StepData:

```

using System;
[Serializable]
public class StepData
{
    public enum NodeType { Start, Step }
    public NodeType Type;
    public int Value; // ожидаемое значение
    public string Operation; // для отображения, например "+20"
}

```

Для механики игры необходимо создать пустой объект, и перенести на него скрипт **GameManager** и в инспекторе произвести настройки (см. рис. 9).

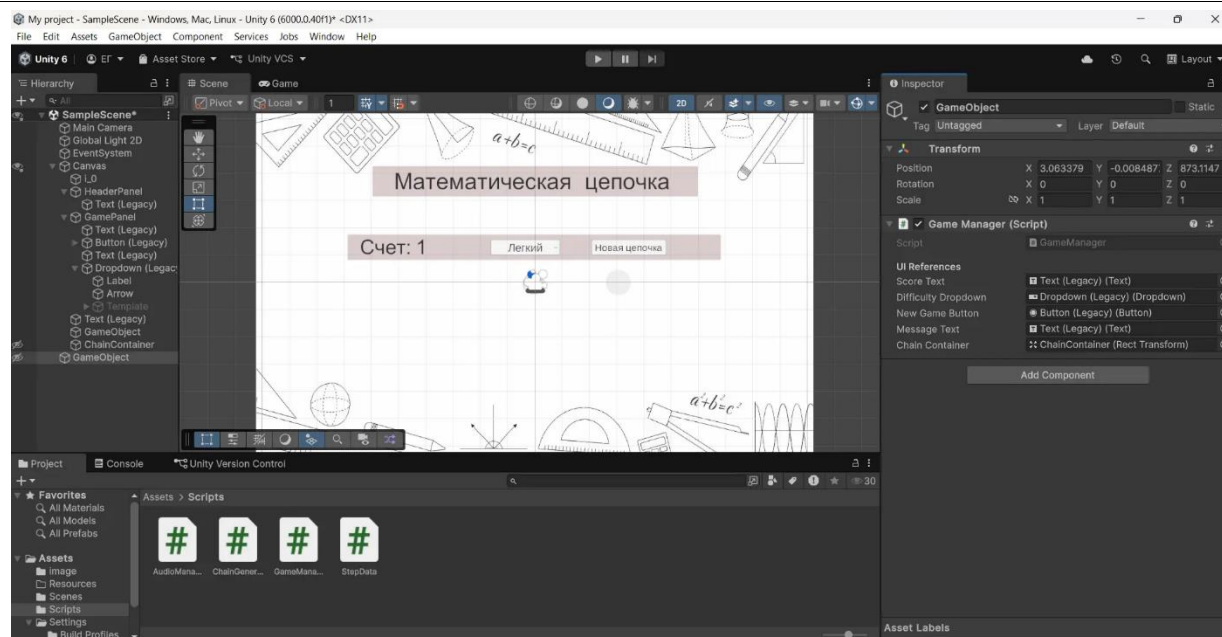
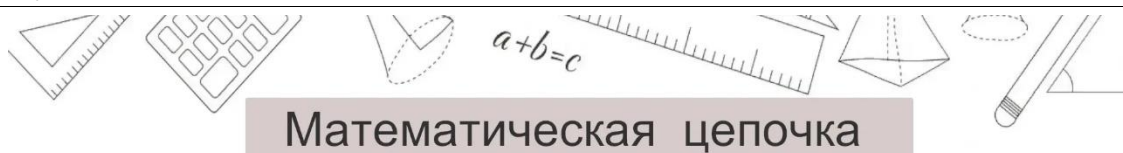


Рисунок 9 – Настройка механики игры

Готовая игра (см. рис. 10. 1, 10. 2).



Рисунок 10.1 – Готовая игра



Счет: 1

сложный ▾

Новая цепочка

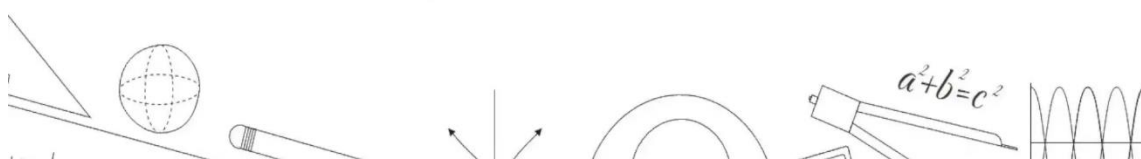


Рисунок 10.2 – Готовая игра

4 Выводы

В данной работе был показан процесс создания компьютерной игры «Математическая цепочка» 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>