

## Создание мобильной игры «бусы» на Unity

*Голубева Евгения Павловна*

*Приамурский государственный университет имени Шолом-Алейхема*

*Студент*

### Аннотация

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

**Ключевые слова:** Unity, игра, бусы.

## Creating a mobile game "Beads" on Unity

Golubeva Evgeniya Pavlovna

Sholom-Aleichem Priamursky State University

Student

### Abstract

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

**Keywords:** Unity, game, beads.

## 1 Введение

### 1.1 Актуальность

В современном мире мобильные игры прочно вошли в повседневную жизнь миллионов людей, став не только способом развлечения, но и мощной индустрией с многомиллиардными оборотами. Среди всего разнообразия жанров особой популярностью пользуются казуальные головоломки, которые предлагают простой, но увлекательный геймплей для широкой аудитории.

Движок Unity, являясь одним из лидеров в среде разработки игр, предоставляет мощные и в то же время доступные инструменты для создания кроссплатформенных проектов. Это делает его идеальным выбором как для опытных студий, так и для начинающих геймдев-специалистов, желающих реализовать свои идеи.

Таким образом, разработка мобильной игры «Бусы» на Unity представляет собой актуальную и практически значимую задачу, сочетающую в себе образовательную ценность, соответствие рыночным запросам и применение современных технологий игровой индустрии.

## 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 мобильной игры «Бусы» для ОС Android при помощи кроссплатформенной среды для создания интерактивных 2D и 3D приложений Unity.

## 2 Материалы и методы

Для создания игры «Бусы» в качестве игрового движка был выбран Unity, поскольку он отличается низким порогом входа, интуитивным интерфейсом и возможностью написания кода на C#. Для редактирования скриптов использовалась интегрированная среда разработки visual studio community.

## 3 Результаты

Первый шаг в создании игры «Бусы» — это создание нового проекта в Unity. Для этого необходимо выбрать в unity hub рендер-пайплайн Universal 2D и создать проект (см. рис. 1).

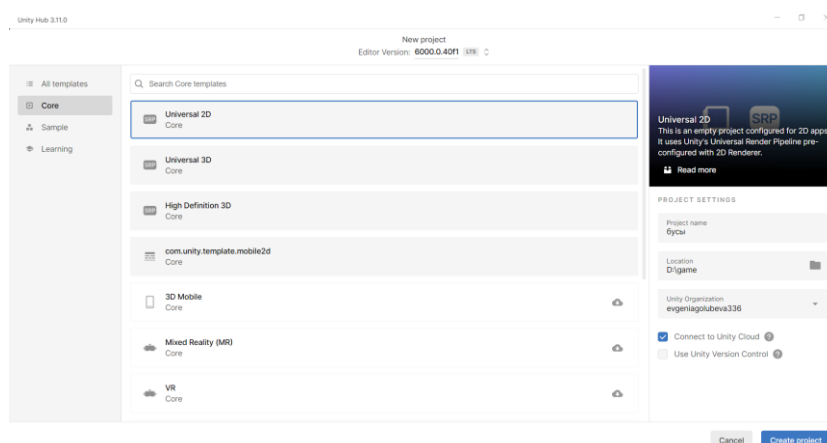


Рисунок-1 Создание проекта

После создания проекта на основе шаблона Universal 2D, необходимо провести базовую настройку под мобильную платформу. В первую очередь, в

настройках построения (File -> Build Profiles) следует переключить платформу на Android (см. рис. 2).

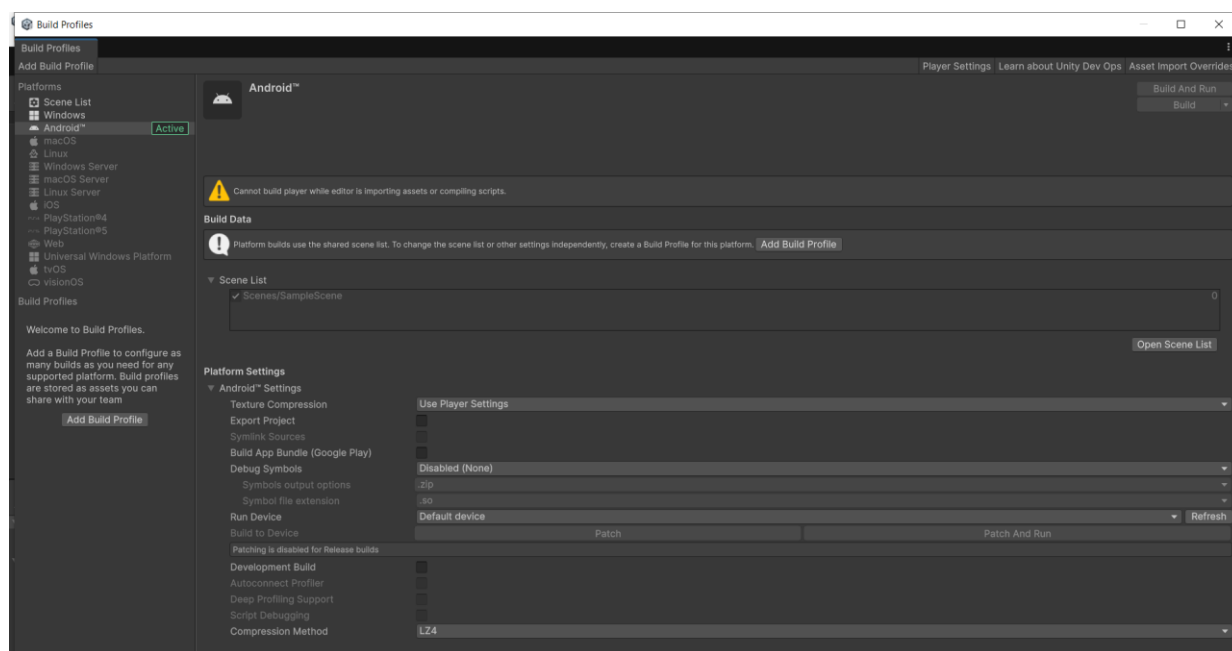


Рисунок- 2 Настройка проекта

Далее приступаем к созданию сцены игры. Первым шагом необходимо настроить камеру (см. рис. 3).

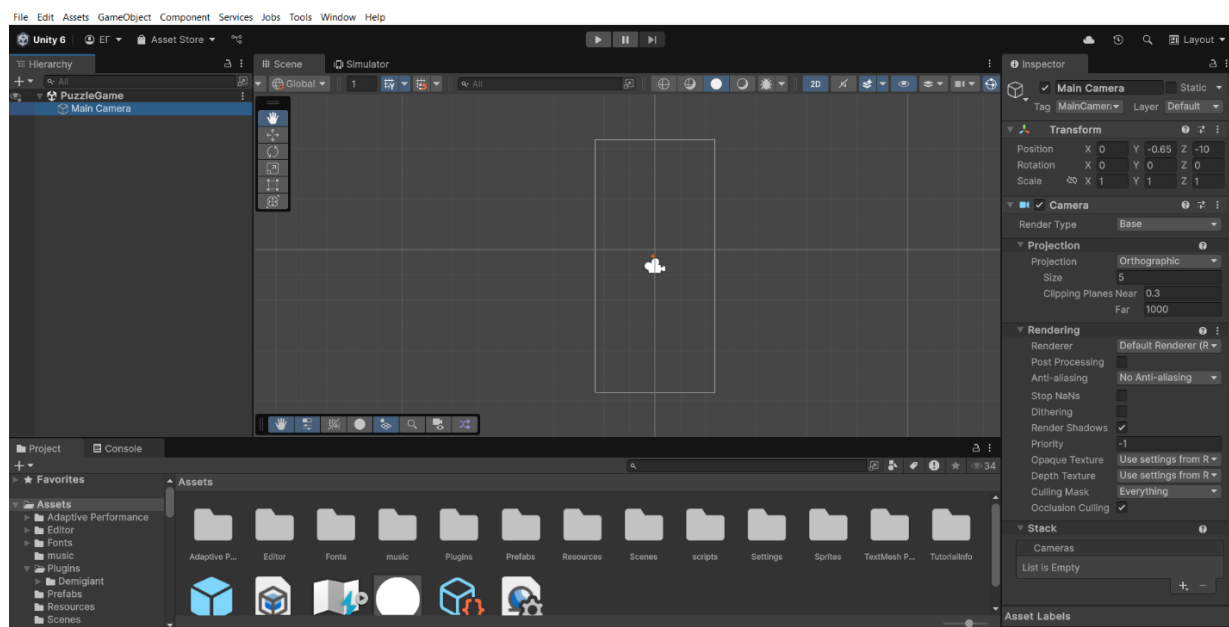


Рисунок-3 Настройка камеры

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

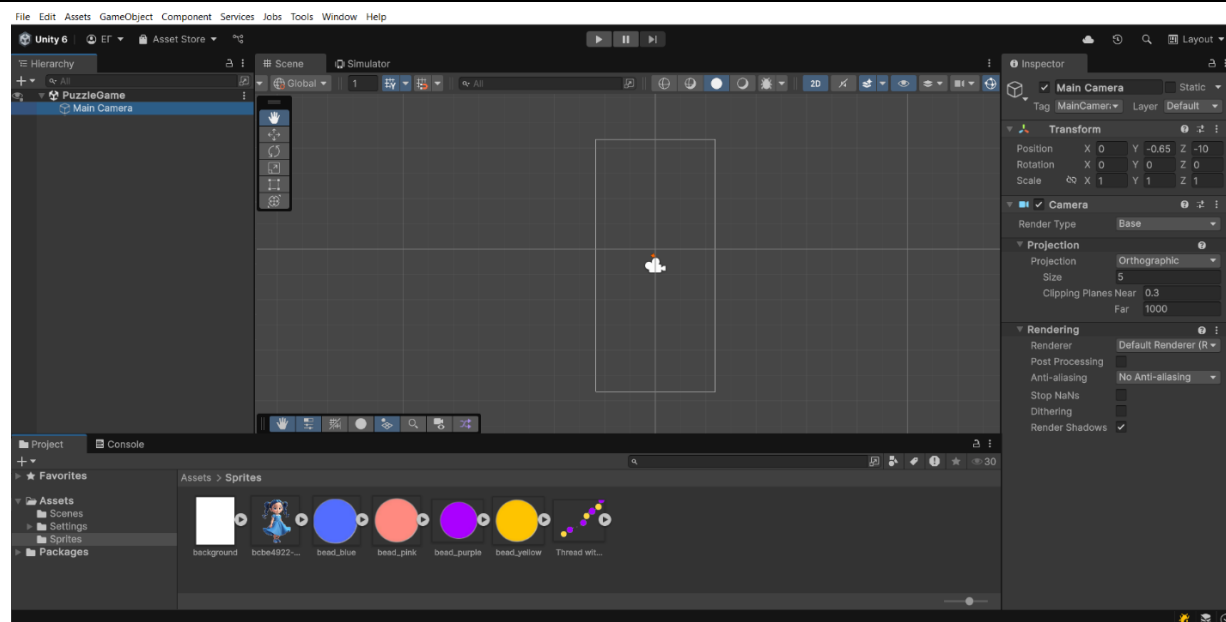


Рисунок 4 - Добавление изображений

Далее добавляем из папки Sprites изображение background для фона игры, и настраиваем изображение (см.рис.5).

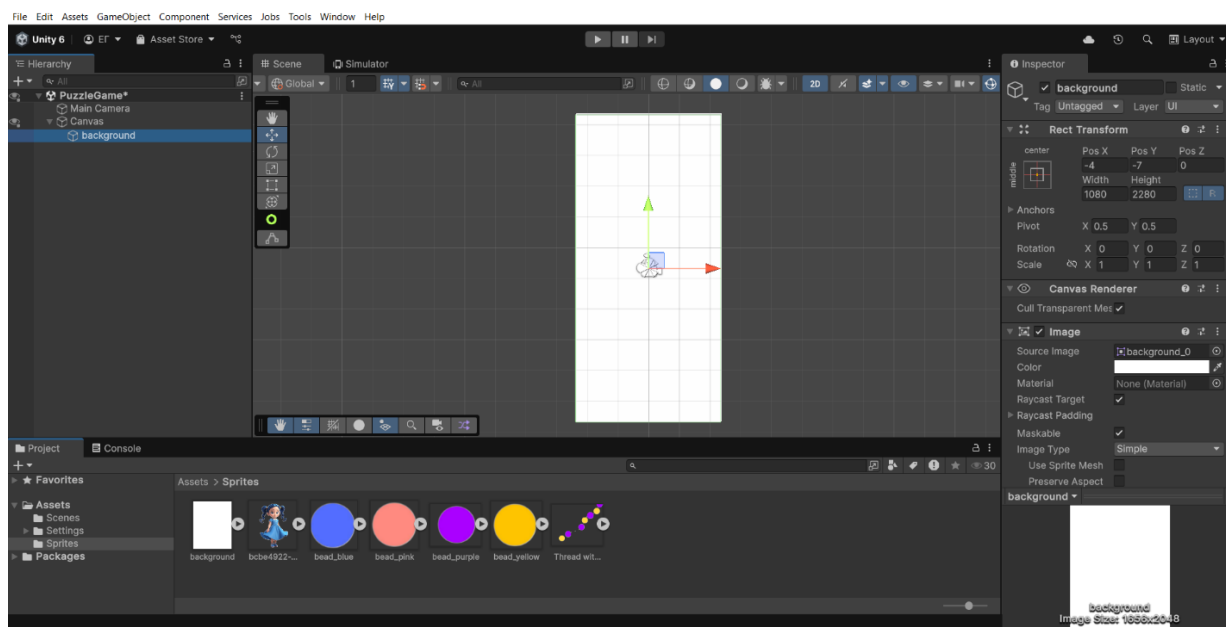


Рисунок 5 – Настройка фона игры

Следующим шагом добавляем изображения нитки с бусами и девочки на сцену (см.рис.6).

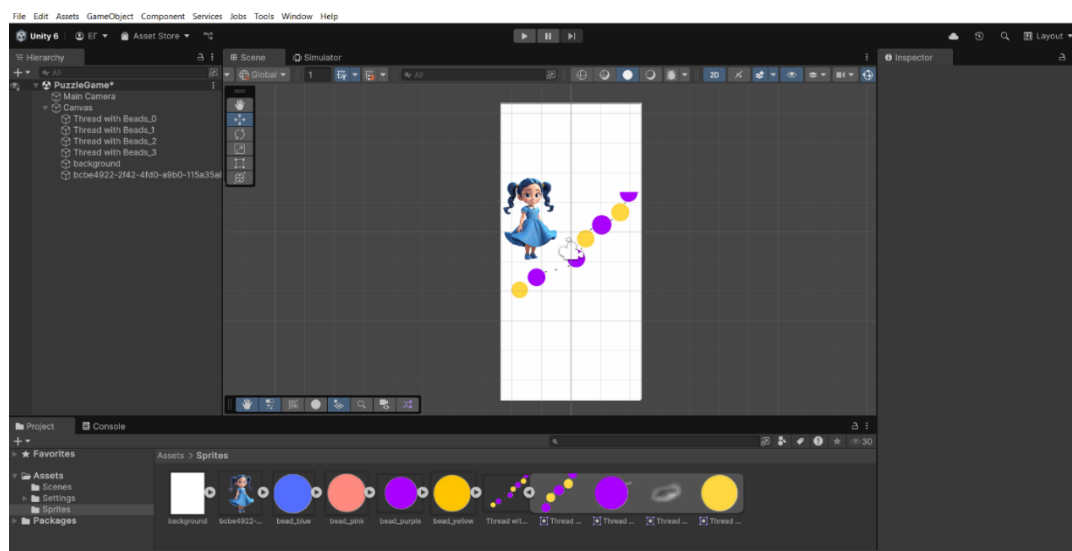


Рисунок 6 – Добавление изображений нити бус и девочки на сцену

Далее необходимо добавить инструкцию игры и панель для отображения уведомления «Правильно!». Для этого в Hierarchy необходимо создать новый UI-объект. Правой кнопкой мыши вызываем меню в пространстве иерархии и выбираем UI -> Panel. Далее для инструкции добавляем текст UI -> Text.

После добавления панели и текста, необходимо их настроить. Для панели устанавливаем якоря (Anchors) в центр (0.5, 0.5) для правильного позиционирования относительно разрешения экрана, задаем точные координаты Position (555, 1240) и размеры Width/Height (1115, 29.41). В компоненте Image выбираем подходящий спрайт для фона, настраиваем цвет и прозрачность, а также выставляем Image Type в значение Simple для оптимального отображения (см.рис.7).

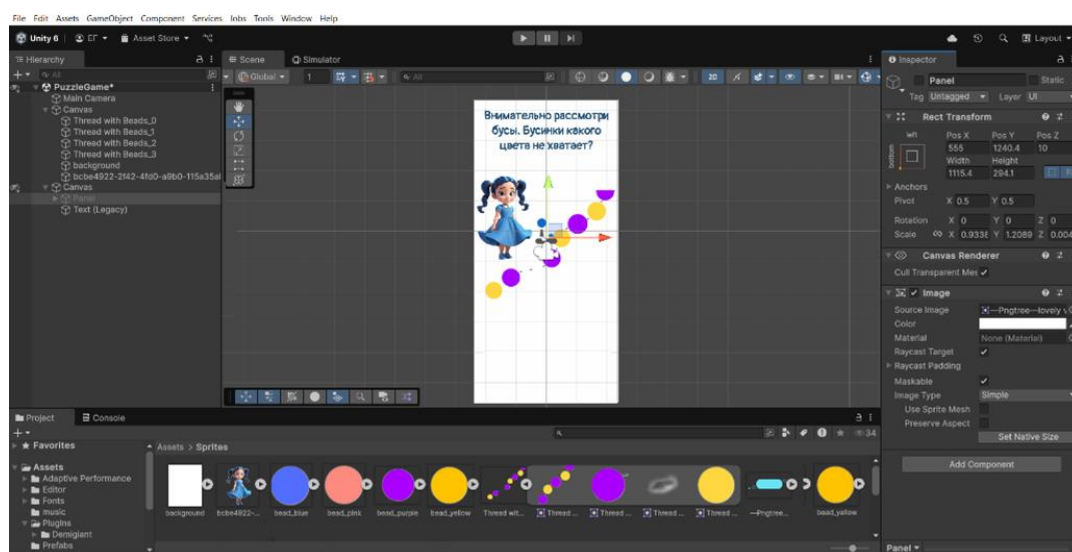


Рисунок 7 – Настройка панели

Также необходимо настроить текстовый элемент панели сообщений. Для текста "Правильно!" в компоненте Text (Legacy) устанавливается

шрифт PlaypenSans-VariableFont\_wght с жирным начертанием (Bold) и размером 150 пунктов для обеспечения хорошей читаемости. В разделе Rect Transform задаются точные размеры текстового поля (Width: 1220.7, Height: 313.4) и позиционирование относительно родительской панели (Pos X: -15, Pos Y: 32). Особое внимание уделяется настройке якорей (Anchors) с центром в точке (0.5, 0.5), что обеспечивает правильное отображение текста на разных разрешениях экрана (см.рис. 8).

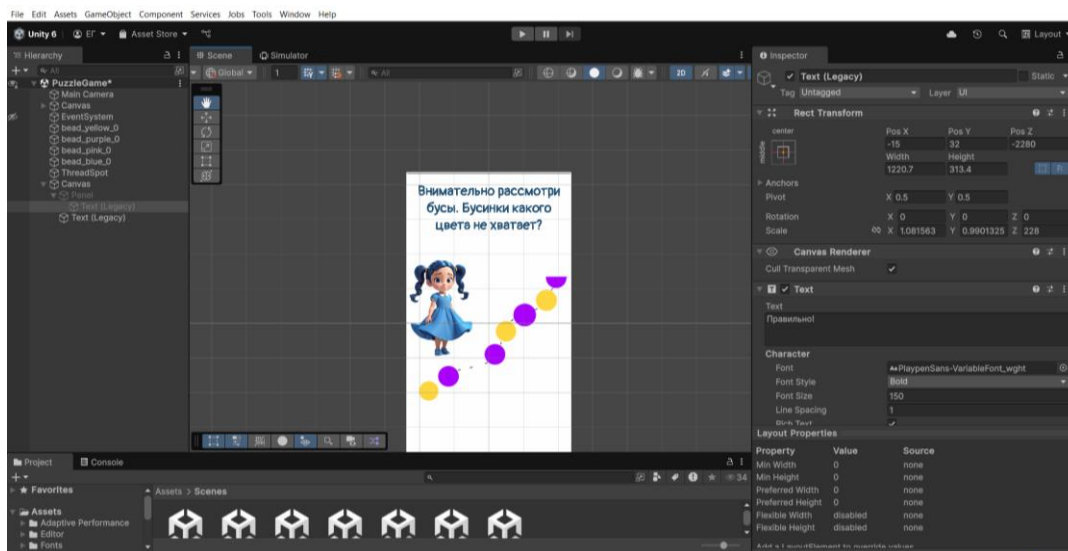


Рисунок 8 – Настройка текстового элемента панели сообщений

Для текстового элемента настраиваем шрифт Pangolin-Regular с жирным начертанием и размером 90 пунктов, выравнивание по центру, включаем поддержку Rich Text для форматирования, а в Rect Transform задаем соответствующие размеры области отображения и позицию относительно родительской панели. После базовой настройки рекомендуется сделать панель изначально неактивной, чтобы она появлялась только в нужные моменты игры (см.рис.9).

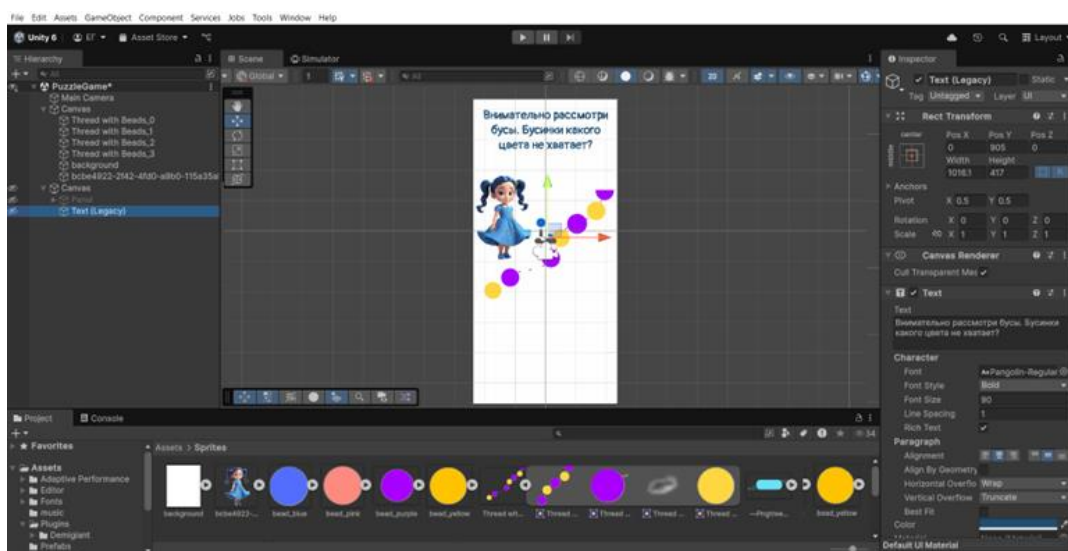


Рисунок 9 – Настройка инструкции игры

Далее добавляем изображения бусинок на сцену и настраиваем их, чтобы можно было передвигать бусины. Для этого в Inspector бусин добавляем компонент Sprite Renderer, где назначается визуальный спрайт бусины, а в качестве материала выбирается Sprite-Lit-Default для совместимости с URP. Далее добавляем Rigidbody 2D, и выбираем режим Kinematic, который позволяет управлять ее движением через код. Для оптимизации замораживаем позицию по осям X и Y в Constraints, чтобы бусина не падала под действием гравитации. Также добавляем компонент Circle Collider 2D, где задается радиус, точно соответствующий границам спрайта (см.рис.10).

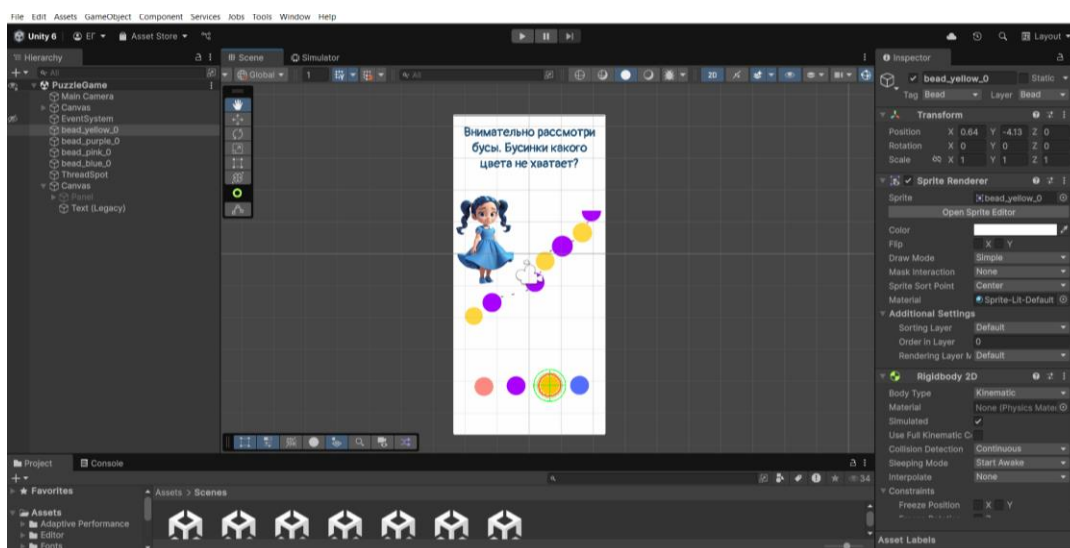


Рисунок 10 – Добавление бусин на сцену и настройка параметров

Также необходимо создать область, куда игрок должен перетащить недостающую бусину. Для этого в Hierarchy создаем пустой игровой объект через правый клик и выбираем Create Empty. Этот объект необходимо переименовать в ThreadSpot. Его основное предназначение — отмечать на сцене точку, где должна находиться бусина правильного цвета. Для реализации этого функционала к ThreadSpot необходимо добавить компонент Box Collider 2D. В настройках коллайдера обязательно надо включить опцию Is Trigger, чтобы объект не создавал физического препятствия, но мог регистрировать пересечения с другими коллайдерами. Размер коллайдера Size настраивается так, чтобы он визуально соответствовал пустому месту на нитке бус. Через компонент Transform задается точное положение объекта на сцене, обеспечивая его выравнивание с уже существующими бусинами (см.рис.11).

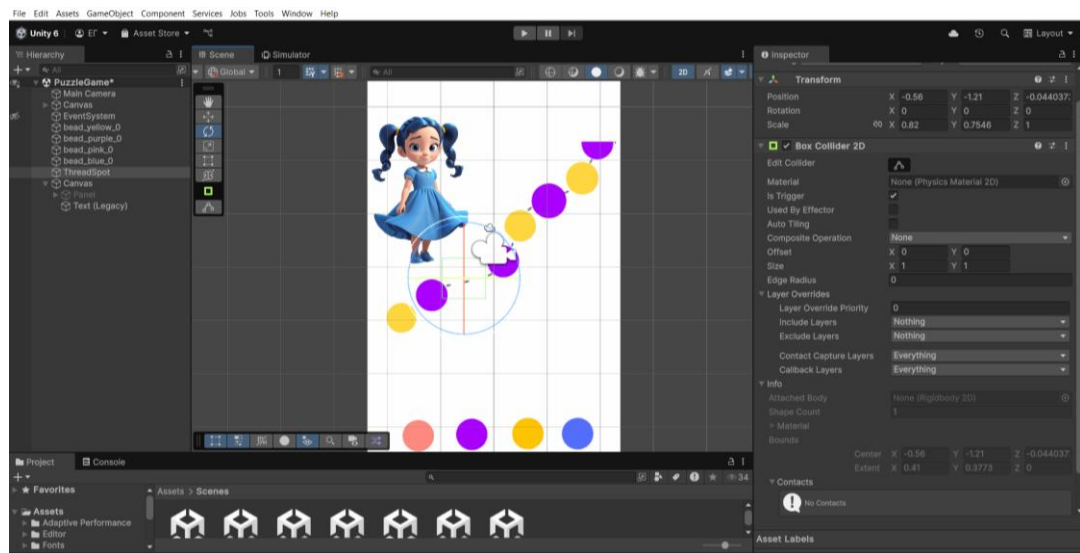


Рисунок 11 – Добавление области недостающей бусины на сцену и настройка параметров

После добавления всех необходимых компонентов необходимо создать скрипт под названием `BeadDragHandler`, который реализует основную игровую механику перетаскивания бусин. Данный скрипт содержит несколько ключевых функциональных блоков:

Основные настройки бусины включают параметры цвета текущей бусины (`beadColor`), правильного цвета (`correctBeadColor`) и радиуса размещения (`placementRadius`), который определяет зону допустимого размещения бусины. Визуальная обратная связь реализована через систему сообщений. При успешном размещении активируется панель с текстом "Правильно!", которая анимируется через последовательное изменение масштаба и прозрачности с использованием `AnimationCurve` для плавности.

Аудио-обратная связь включает два типа звуков: `dragSound` при начале перетаскивания бусин и `successSound` при правильном размещении бусины. Для воспроизведения используется компонент `AudioSource`.

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

public class BeadDragHandler : MonoBehaviour
{
    [Header("Bead Settings")]
    public string beadColor = "bead_blue";
    public string correctBeadColor = "bead_yellow";
    public float placementRadius = 0.5f;

    [Header("UI Settings")]
    public RectTransform messagePanel;
    public Text messageText;
    public Image panelBackground;
    public AudioClip successSound;
    public AudioClip dragSound; // Добавлен звук перетаскивания
```

```
[Header("Animation")]
public float fadeDuration = 0.5f;
public float showDuration = 2f;
public AnimationCurve scaleCurve = AnimationCurve.EaseInOut(0, 0, 1, 1);

[Header("Scene Management")]
public string menuSceneName = "MainMenu";

private Vector3 originalPosition;
private bool isPlacedCorrectly;
private bool isDragging;
private AudioSource audioSource;
private Canvas messageCanvas;
private CanvasGroup canvasGroup;

void Start()
{
    originalPosition = transform.position;
    audioSource = GetComponent<AudioSource>();

    // Если AudioSource не добавлен, добавляем его
    if (audioSource == null)
    {
        audioSource = gameObject.AddComponent<AudioSource>();
    }

    InitializeMessagePanel();
}

void InitializeMessagePanel()
{
    if (messagePanel != null)
    {
        messageCanvas = messagePanel.gameObject.AddComponent<Canvas>();
        messageCanvas.overrideSorting = true;
        messageCanvas.sortingLayerName = "UI";
        messageCanvas.sortingOrder = 9999;

        canvasGroup =
messagePanel.gameObject.AddComponent<CanvasGroup>();
        canvasGroup.alpha = 0;
        messagePanel.gameObject.SetActive(false);
    }
}

private void OnMouseDown()
{
    if (!isPlacedCorrectly && !IsPointerOverUI())
    {
        PlayDragSound(); // Воспроизводим звук при начале перетаскивания
        StartCoroutine(DragBead());
    }
}

private bool IsPointerOverUI()
{
    return EventSystem.current != null &&
EventSystem.current.IsPointerOverGameObject();
}

private IEnumerator DragBead()
{
    isDragging = true;
```

```
        while (Input.GetMouseButton(0))
        {
            Vector3 mousePos =
Camera.main.ScreenToWorldPoint(Input.mousePosition);
            transform.position = new Vector3(mousePos.x, mousePos.y, 0);
            yield return null;
        }

        isDragging = false;
        CheckPlacement();
    }

    // Метод для воспроизведения звука перетаскивания
    private void PlayDragSound()
    {
        if (dragSound != null && audioSource != null)
        {
            audioSource.PlayOneShot(dragSound);
        }
    }

    // Альтернативный метод - непрерывное воспроизведение звука во время
перетаскивания
    private IEnumerator PlayDragSoundContinuous()
    {
        if (dragSound == null || audioSource == null) yield break;

        while (isDragging)
        {
            if (!audioSource.isPlaying)
            {
                audioSource.PlayOneShot(dragSound);
            }
            yield return new WaitForSeconds(0.1f); // Интервал между
повторениями
        }
    }

    private void ReturnToOriginalPosition()
    {
        transform.position = originalPosition;
    }

    private void CheckPlacement()
    {
        Collider2D hit = Physics2D.OverlapCircle(transform.position,
placementRadius);
        if (hit != null && hit.CompareTag("ThreadZone"))
        {
            if (beadColor == correctBeadColor)
            {
                HandleCorrectPlacement(hit.transform.position);
            }
            else
            {
                ReturnToOriginalPosition();
            }
        }
        else
        {
            ReturnToOriginalPosition();
        }
    }
}
```

```
private void HandleCorrectPlacement(Vector3 targetPosition)
{
    isPlacedCorrectly = true;
    transform.position = targetPosition;

    // Отключаем физику если есть Rigidbody2D
    Rigidbody2D rb = GetComponent<Rigidbody2D>();
    if (rb != null)
    {
        rb.bodyType = RigidbodyType2D.Static;
    }
    ShowSuccessMessage();
}
private void ShowSuccessMessage()
{
    if (messagePanel == null) return;
    messagePanel.gameObject.SetActive(true);
    if (messageCanvas != null) messageCanvas.sortingOrder = 9999;
    if (messageText != null) messageText.text = "Правильно!";
    if (panelBackground != null) panelBackground.color = new Color(0.2f,
0.8f, 0.2f, 0.9f);

    PlaySuccessSound();
    StartCoroutine(AnimateMessage());
}

private void PlaySuccessSound()
{
    if (successSound == null) return;

    if (audioSource != null)
    {
        audioSource.PlayOneShot(successSound);
    }
    else
    {
        AudioSource.PlayClipAtPoint(successSound,
Camera.main.transform.position);
    }
}

private IEnumerator AnimateMessage()
{
    yield return AnimateScale(Vector3.zero, Vector3.one, fadeDuration);
    yield return AnimateAlpha(0, 1, fadeDuration);
    yield return new WaitForSeconds(showDuration);
    yield return AnimateAlpha(1, 0, fadeDuration);
    messagePanel.gameObject.SetActive(false);
    LoadMenuScene();
}

private IEnumerator AnimateScale(Vector3 from, Vector3 to, float
duration)
{
    float elapsed = 0f;
    while (elapsed < duration)
    {
        float t = elapsed / duration;
        messagePanel.localScale = Vector3.Lerp(from, to,
scaleCurve.Evaluate(t));
        elapsed += Time.deltaTime;
        yield return null;
    }
    messagePanel.localScale = to;
}

private IEnumerator AnimateAlpha(float from, float to, float duration)
```

```

{
    float elapsed = 0f;
    while (elapsed < duration)
    {
        float t = elapsed / duration;
        canvasGroup.alpha = Mathf.Lerp(from, to, t);
        elapsed += Time.deltaTime;
        yield return null;
    }
    canvasGroup.alpha = to;
}
private void LoadMenuScene()
{
    if (!string.IsNullOrEmpty(menuSceneName))
    {
        SceneManager.LoadScene(menuSceneName);
    }
}
void OnDrawGizmosSelected()
{
    Gizmos.color = Color.green;
    Gizmos.DrawWireSphere(transform.position, placementRadius);
}
}

```

После создания скрипта `BeadDragHandler` необходимо добавить его к объектам-бусинам в сцене. Для этого необходимо перетащить скрипт на каждую бусину в Hierarchy. После добавления скрипта требуется настроить его параметры: укажите цвет бусины (`Bead Color`) и соответствующий правильный цвет (`Correct Bead Color`), установите радиус размещения (`Placement Radius`) около 0.5. В Message Panel добавляем панель сообщений, а в Message Text текстовое поле панели, а также добавляем звуковые файлы для перетаскивания в `Drag Sound`, а для успешного размещения в `Success Sound` (см. рис. 12).

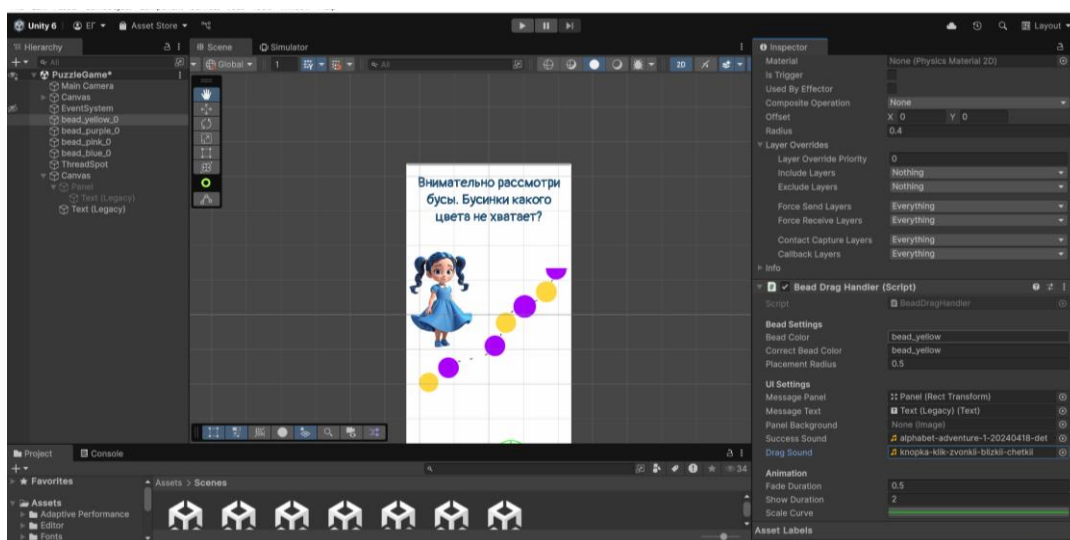


Рисунок 12 – Добавление скрипта к бусинам

Итоговый результат создания игры «Бусы» (см. рис.13).



Рисунок 12 – Готовая игра «Бусы»

#### 4. Выводы

В данной работе был показан процесс создания игры «бусы» 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: Сборник научных статей. Волгоград: Общество с ограниченной ответственностью "Сфера", 2024. С. 22-25.
2. Фаст Я. А. Разработка 2D игр с элементами головоломок на платформе Unity // V Международная научная конференция по междисциплинарным исследованиям: сборник статей V Международной научной конференции по междисциплинарным исследованиям, Екатеринбург, 15 апреля 2024 года. Екатеринбург: Общество с ограниченной ответственностью "Институт Цифровой Экономики и Права", 2024. С. 266-269.
3. Максимов Н. В. Разработка мобильной 2D игры "Bootur Jump" на Unity // Фундаментальные и прикладные аспекты развития современной науки: Сборник трудов по материалам XV Международного конкурса научно-исследовательских работ, Уфа, 04 декабря 2023 года. Уфа: Общество с ограниченной ответственностью "Научно-издательский центр "Вестник науки", 2023. С. 306-308.

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