**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет Информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

**Дисциплина:** Шаблоны проектирования

**Тема:** Система игровых событий

**Выполнил: студент группы** *231-338*

*Богослов Илья Максимович*

(Фамилия И.О.)

**Дата, подпись** \_\_\_*22.05.2025*\_\_\_  ***\_\_\_*** ***\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2025**

**Цель**: Создайте систему событий, в которой различные игровые объекты могут подписываться и реагировать на игровые события с использованием определенного шаблона проектирования.

**Описание**: В играх часто происходят различные события: от перемещения игрока до завершения задания. Для управления такими событиями и реакцией на них различных компонентов игры необходима эффективная система. Использование подходящего шаблона проектирования может сделать эту систему более гибкой и удобной.

**Шаги**:

1. **Определение игровых событий**:
   * Определите набор игровых событий, которые вы хотите реализовать. Это могут быть события, связанные с движением игрока, завершением уровня, сбором предметов и т. д.
2. **Реализация системы событий**:
   * Создайте "издателя" событий, который будет генерировать события.
   * Разработайте механизм подписки, который позволит другим игровым объектам "подписываться" на интересующие их события.
   * Игровые объекты, "подписанные" на события, должны реагировать на них соответствующим образом.
3. **Тестирование**:
   * Запустите вашу игру и проверьте, как различные компоненты реагируют на игровые события в реальном времени.

**Ход работы:**

Было решено реализовать набор событий в качестве квеста по сбору монет. На поверхности разместились игрок и 5 объектов-монет.

Для реализации системы событий был выбран паттерн Observer.

Была создана библиотека событий, реализующая механизмы подписки и отписки на событие и для их «издания»:

using System;  
using System.Collections.Generic;  
  
public static class EventBus  
{  
 private static readonly Dictionary<Type, Delegate> Map = new();  
  
 public static void Subscribe<T>(Action<T> listener)  
 {  
 if (listener == null)  
 return;  
 Map.TryGetValue(typeof(T), out var dlg);  
 Map[typeof(T)] = Delegate.Combine(dlg, listener);  
 }  
  
 public static void Unsubscribe<T>(Action<T> listener)  
 {  
 if (listener == null)  
 return;  
 if (!Map.TryGetValue(typeof(T), out var dlg))  
 return;  
 var newDlg = Delegate.Remove(dlg, listener);  
 if (newDlg == null)  
 Map.Remove(typeof(T));  
 else  
 Map[typeof(T)] = newDlg;  
 }  
  
 public static void Publish<T>(T evt)  
 {  
 if (Map.TryGetValue(typeof(T), out var dlg))  
 (dlg as Action<T>)?.Invoke(evt);  
 }  
}

В качестве издателя событий будут выступать монеты. Для них создаётся скрипт, который издаёт событие ItemCollected при соприкосновении с ним игрока:

using UnityEngine;  
  
[RequireComponent(typeof(Collider))]  
public sealed class PickupItem : MonoBehaviour  
{  
 [SerializeField]  
 private string itemId = "coin";  
  
 [Min(1)]  
 [SerializeField]  
 private int amount = 1;  
  
 [SerializeField]  
 private AudioClip sfx;  
  
 private void OnTriggerEnter(Collider other)  
 {  
 if (!other.CompareTag("Player"))  
 return;  
  
 EventBus.Publish(new ItemCollected(itemId, amount));  
  
 if (sfx)  
 AudioSource.PlayClipAtPoint(sfx, transform.position);  
  
 Destroy(gameObject);  
 }  
}

Событие ItemCollected:

public readonly struct ItemCollected  
{  
 public readonly string ItemId;  
 public readonly int Amount;  
  
 public ItemCollected(string itemId, int amount = 1)  
 {  
 ItemId = itemId;  
 Amount = amount;  
 }  
}

Так же создаются подписчики на это событие, которые будут обрабатывать это событие каждый по своему.

На игрока прикрепился скрипт PlayerInventory, который выводит в консоль кол-во монет в инвентаре:

using System.Collections.Generic;  
using UnityEngine;  
  
public sealed class PlayerInventory : MonoBehaviour  
{  
 private readonly Dictionary<string, int> \_items = new();  
  
 private void OnEnable() => EventBus.Subscribe<ItemCollected>(OnItem);  
  
 private void OnDisable() => EventBus.Unsubscribe<ItemCollected>(OnItem);  
  
 private void OnItem(ItemCollected evt)  
 {  
 \_items.TryGetValue(evt.ItemId, out var cur);  
 \_items[evt.ItemId] = cur + evt.Amount;  
 Debug.Log($"Picked up {evt.Amount}×{evt.ItemId}. Now: {\_items[evt.ItemId]}");  
 }  
}

На UI создано текстовое поле счётчика монет, на которое повешен соответствующий скрипт:

using TMPro;  
using UnityEngine;  
  
public sealed class CoinCounterUI : MonoBehaviour  
{  
 [SerializeField]  
 private TMP\_Text label;  
 private int \_coins;  
  
 private void OnEnable() => EventBus.Subscribe<ItemCollected>(OnItem);  
  
 private void OnDisable() => EventBus.Unsubscribe<ItemCollected>(OnItem);  
  
 private void OnItem(ItemCollected evt)  
 {  
 if (evt.ItemId != "coin")  
 return;  
 \_coins += evt.Amount;  
 label.text = \_coins.ToString("N0");  
 }  
}

В менеджер квестов, расположенный на сцене, помещён скрипт квеста с монетками:

using UnityEngine;  
  
public sealed class CollectCoinsQuest : MonoBehaviour  
{  
 [SerializeField] private int target = 5;  
 private int \_progress;  
  
 private void OnEnable() => EventBus.Subscribe<ItemCollected>(OnItem);  
 private void OnDisable() => EventBus.Unsubscribe<ItemCollected>(OnItem);  
  
 private void OnItem(ItemCollected evt)  
 {  
 if (evt.ItemId != "coin") return;  
  
 \_progress += evt.Amount;  
 if (\_progress < target) return;  
 Debug.Log("Quest complete: Gather Coins!");  
 OnDisable();  
 }  
}

Как итог: когда Вы наступаете на предмет, PickupItem публикует ItemCollected, а все подписчики получают собственное уведомление и реагируют независимо.

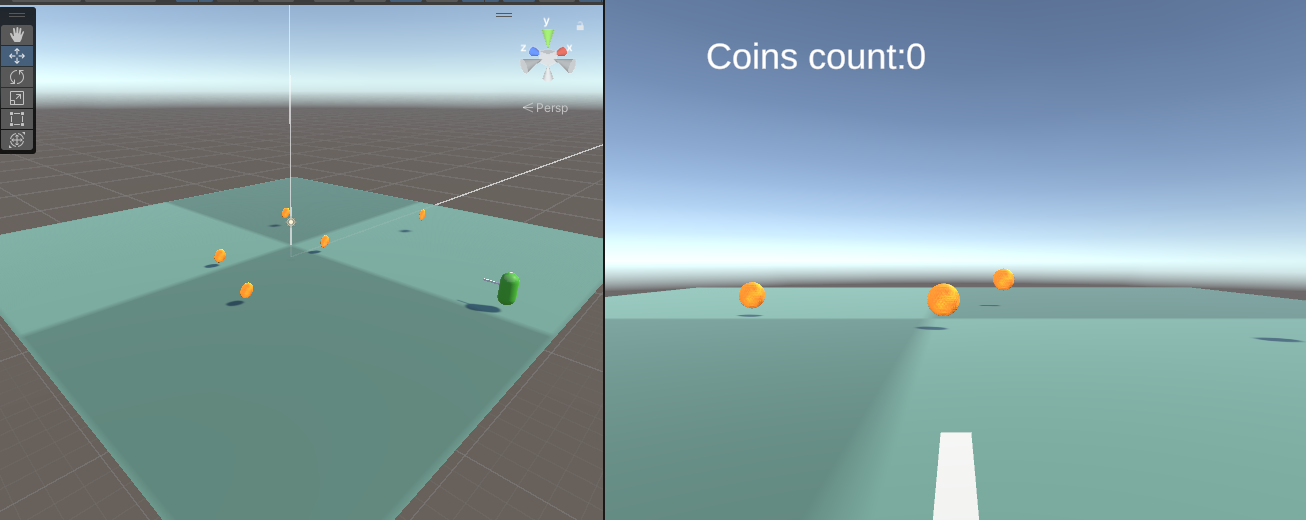


Рисунок 1 – Начальный экран

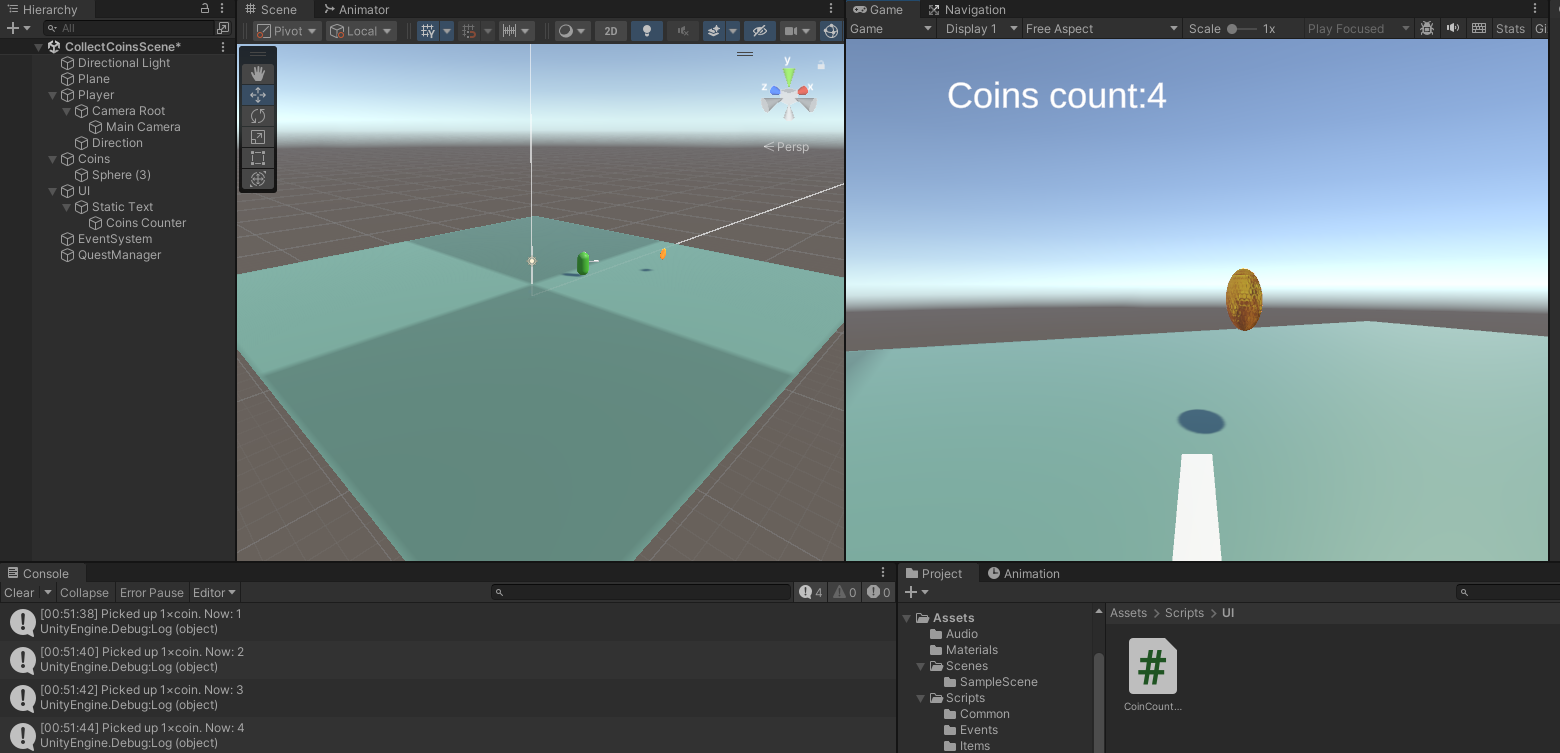


Рисунок 2 – Собранные 4 монеты

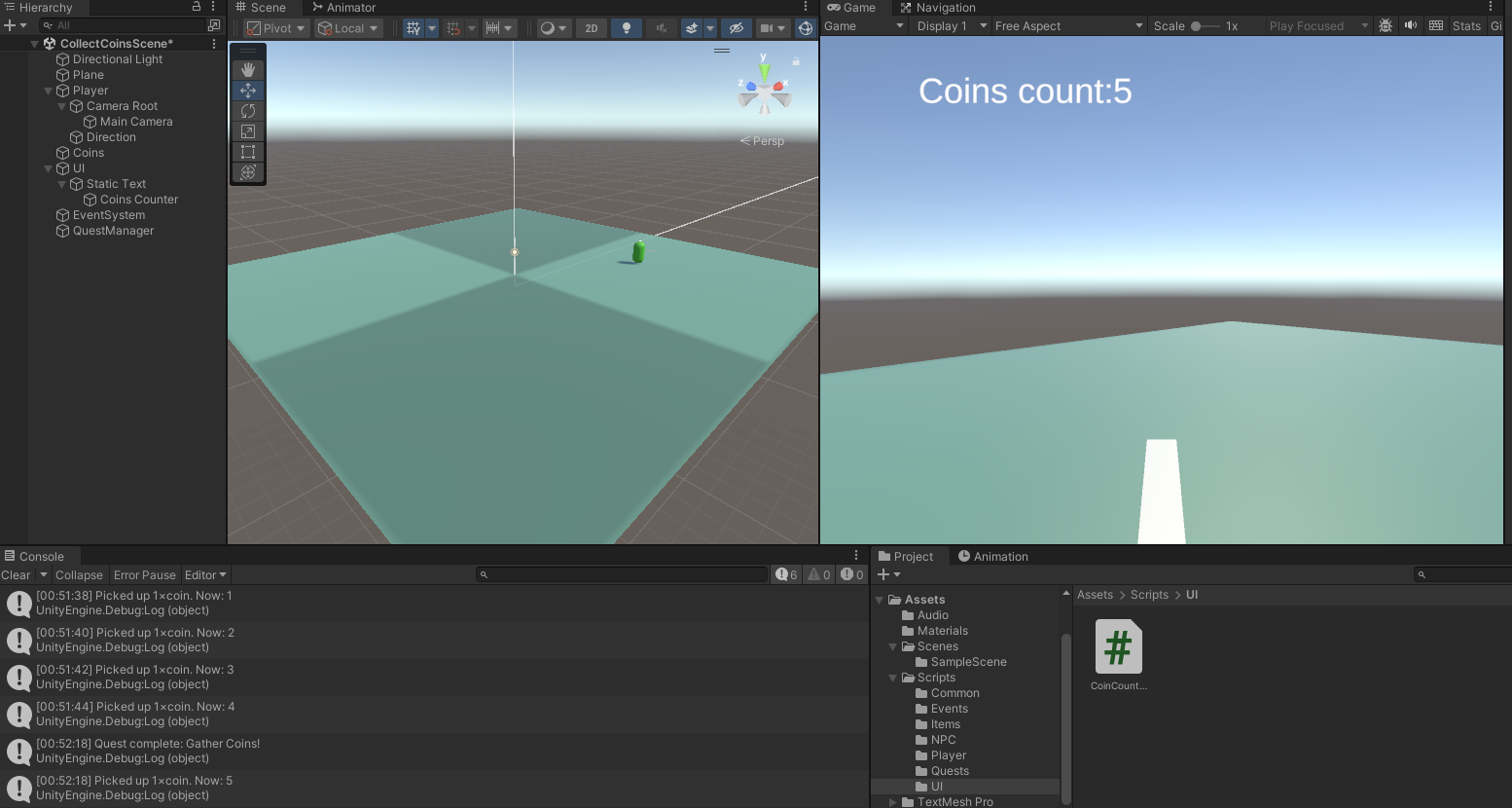


Рисунок 3 – Завершение квеста