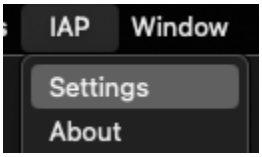
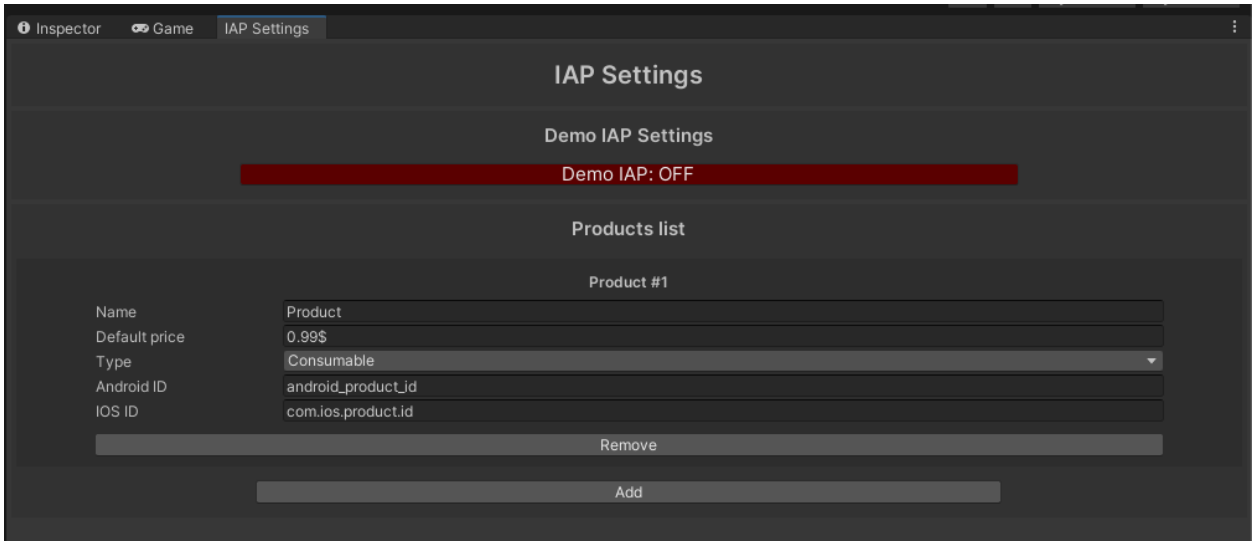
Як додати покупки

Щоб додати покупки потрібно відкрити вікно “IAP/Settings”:



Відкриється наступне вікно:



Demo IAP Settings - для того щоб у білді працювали тестові покупки треба включати кнопку Demo IAP, якщо вона горить зеленим то тестові покупки включені, якщо червоним то виключені.

Products list - це список того що буде продаватися у грі.

Для налаштування продукту потрібно заповнити наступні поля:

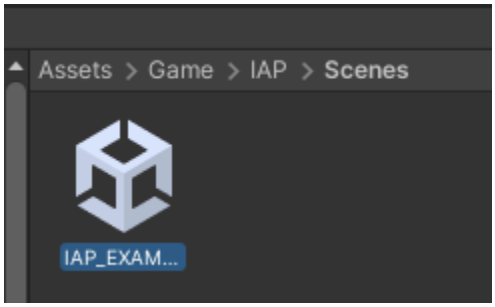
- Name - назва продукту, використовується для того щоб удобно вибирати який продукт продавати.
- DefaultPrice - це ціна яка буде відображатися якщо гра впреше була запущена без інтернету, і ніяк не змогла підтянути ціну, якщо колись ціна уже підтягувалася з маркета то буде остання відома ціна відображатися.

- Type - це тип покупки, може бути Consumable (ресурс, як монетки або ще щось що можна купляти багато разів) і NotConsumable (одноразові покупки, як відключення реклами або скіни і контент).
- Android ID - це ід покупки який треба брати з консолі гугла і вставляти в це поле.
- IOS ID - це ід покупки який треба брати з стора епла і вставляти в це поле.

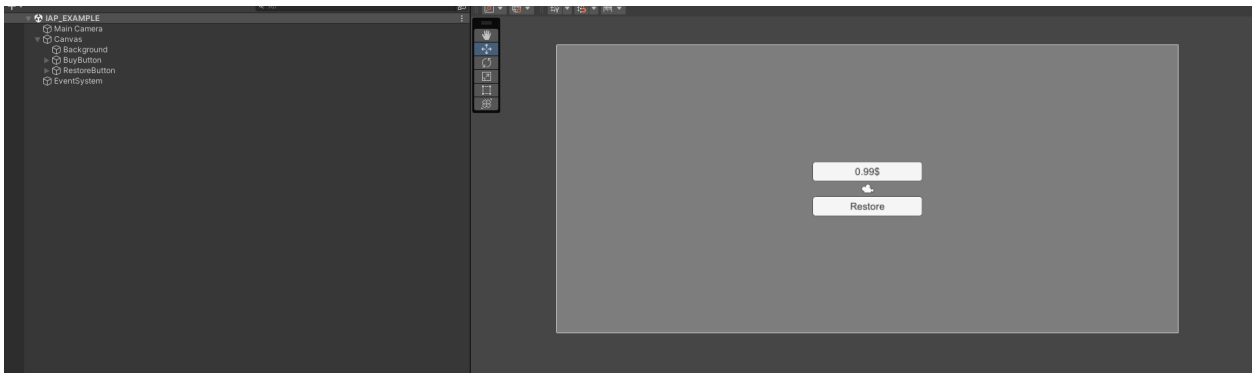
ID покупок спеціально розділені, і будуть вибиратися в залежжності від платформи.

Додавати і видаляти продукти можна по кнопкам.

Як робити покупки

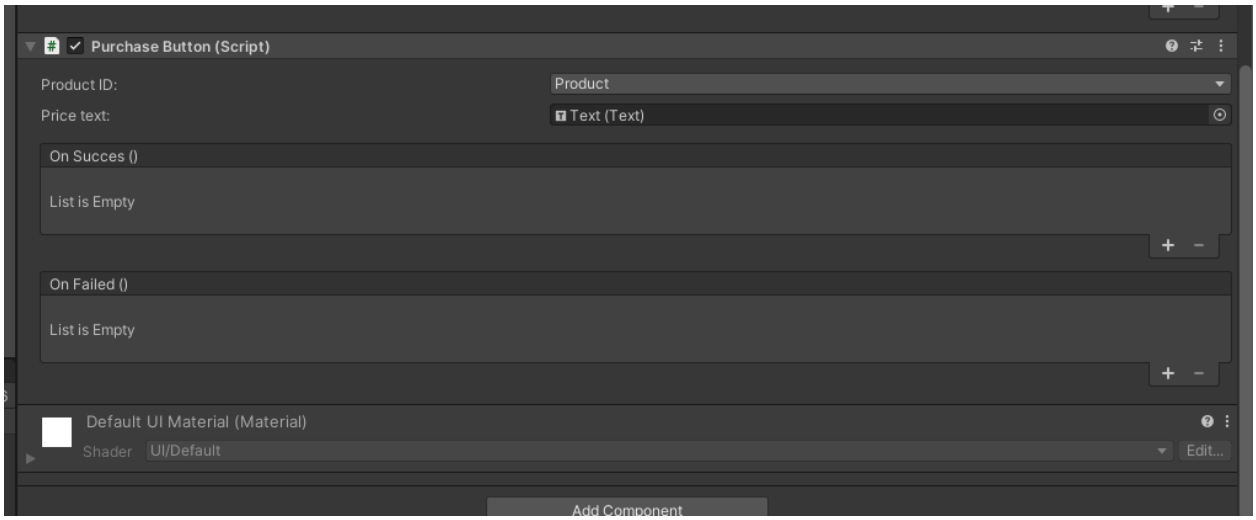


У пакеті є сцена приклад використання покупок.



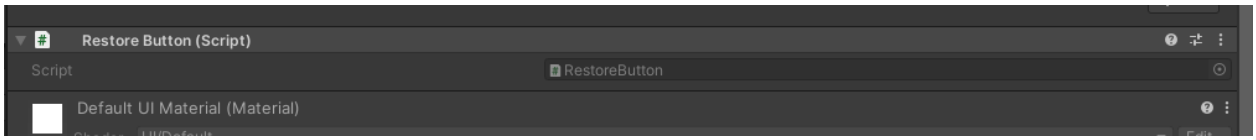
На сцені є дві кнопки - одна кнопка це кнопка для покупки, а інша це Restore кнопка.

Restore кнопка це кнопка яка на IOS повертає тобі в гру твої одноразові покупки якщо ти переустановив гру або граєш на іншому девайсі.

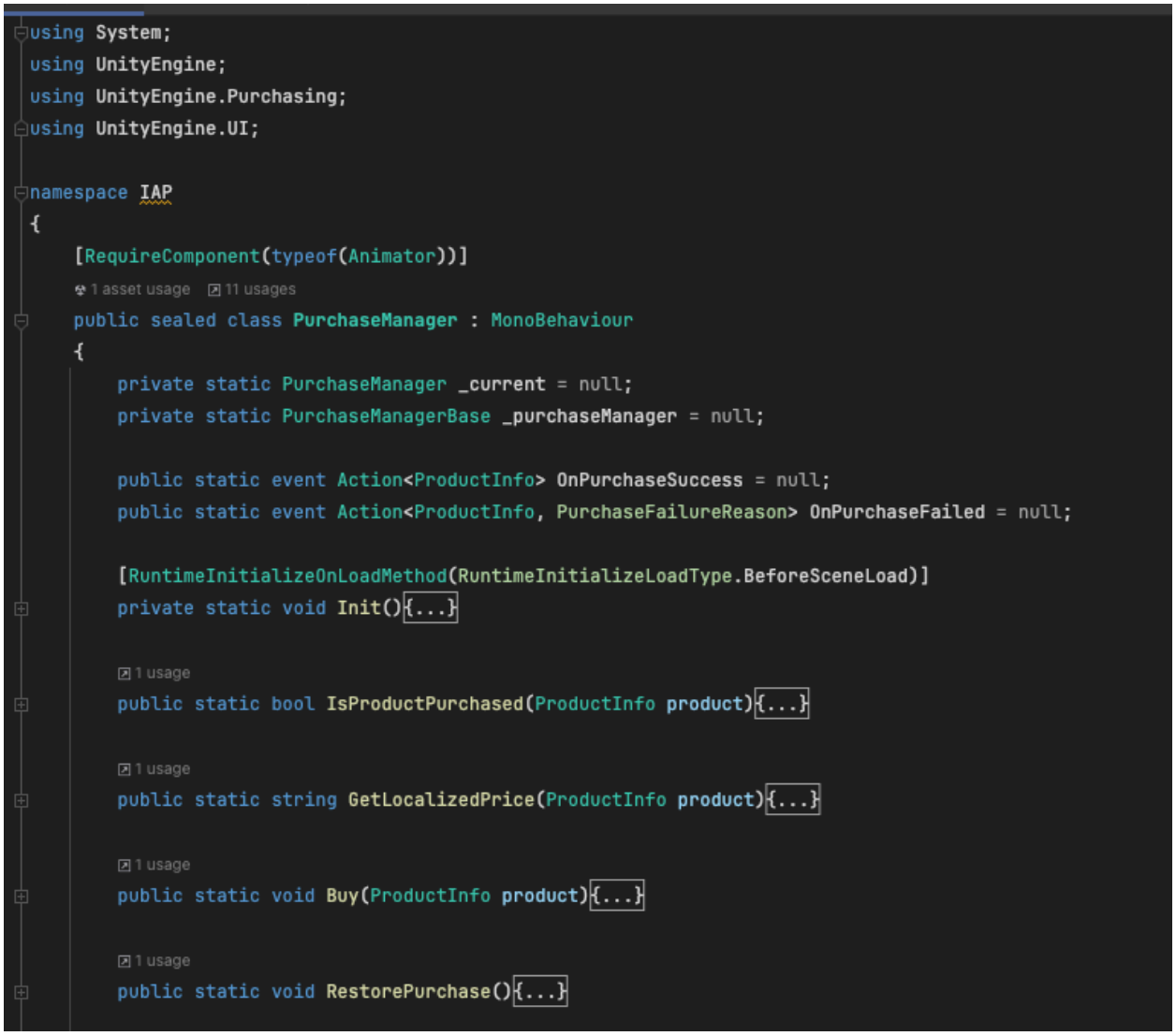


На кнопці покупки висить скрипт PurchaseButton, його достатньо просто закинути, вибрати назву продукта і задати текст для виводу ціни. Якщо текст не задати то помилки не буде.

Івент OnSucces викликається при успішній покупці, а івент OnFailed якщо покупка не вдалася, або юзер відмінив дію.



На кнопці відновленн покупки висить скрипт PurchaseButton, його достатньо просто закинути.



```
using System;
using UnityEngine;
using UnityEngine.Purchasing;
using UnityEngine.UI;

namespace IAP
{
    [RequireComponent(typeof(Animator))]
    1 asset usage  11 usages
    public sealed class PurchaseManager : MonoBehaviour
    {
        private static PurchaseManager _current = null;
        private static PurchaseManagerBase _purchaseManager = null;

        public static event Action<ProductInfo> OnPurchaseSuccess = null;
        public static event Action<ProductInfo, PurchaseFailureReason> OnPurchaseFailed = null;

        [RuntimeInitializeOnLoadMethod(RuntimeInitializeLoadType.BeforeSceneLoad)]
        private static void Init(){...}

        1 usage
        public static bool IsProductPurchased(ProductInfo product){...}

        1 usage
        public static string GetLocalizedPrice(ProductInfo product){...}

        1 usage
        public static void Buy(ProductInfo product){...}

        1 usage
        public static void RestorePurchase(){...}
```

Для того щоб зробити покупки в ручну можна скористатися скриптом PurchaseManager:

- PurchaseManager.IsProductPurchased(ProductInfo productInfo) - це метод для провірки чи продукт куплений (якщо він NotConsumable).
- PurchaseManager,GetLocalizedPrice(ProductInfo productInfo) - це метод для отримання ціни продукту.
- PurchaseManager.Buy(ProductInfo productInfo) - це метод для виклику покупки.
- PurchaseManager.RetorePurchase() - це метод для виклику відновлення покупки.

- Action<ProductInfo> OnPurchaseSuccess - це івент який викликається при успішній покупці.
- Action<ProductInfo, PurchaseFailureReazon> - це івент який викликається при помилці або відміні покупки.

Для того щоб отримати ProductInfo можна скористатися IAPSettings.Data.GetProductInfo(string name).

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

namespace IAP
{
    [CreateAssetMenu(fileName = "IAPSettingsData", menuName = "Settings/IAPSettingsData", order = 0)]
    1 asset usage  4 usages  1 exposing API
    public class IAPSettingsData : ScriptableObject
    {
        [SerializeField] private bool _usDemoIAP = false;  "true"

        [SerializeField] private List<ProductInfo> _products = new List<ProductInfo>();  Serializable

        6 usages
        public bool UsDemoIAP => _usDemoIAP || Application.isEditor;

        2 usages
        public ProductInfo GetProductInfo(string name)
        {
            return _products.FirstOrDefault(p :ProductInfo => p.Name == name);
        }

        2 usages
        public ProductInfo GetProductInfoById(string id)
        {
            return _products.FirstOrDefault(p :ProductInfo => p.GetCurrentId() == id);
        }

        Frequently called  3 usages
        public IReadOnlyList<ProductInfo> GetAllProductInfos()
        {
            return _products;
        }
    }
}
```