Комментарии к Лабораторной работе №4

Пошаговая инструкция по созданию плоских кнопок «на стекле» в Приложении ДР для управления:

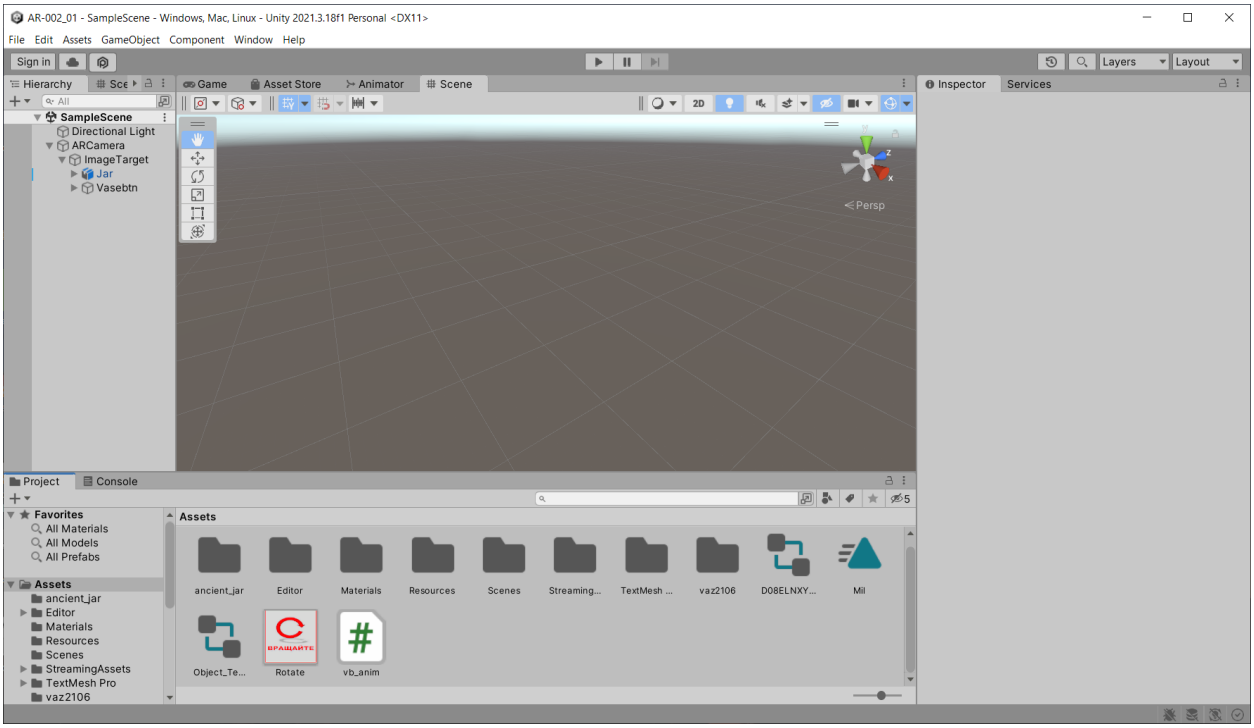
I. **перемещением объекта контента (трехмерная модель) по одной из осей в положительном (или отрицательном ее направлении),**
II. **выходом из Приложения,**
III. **ранее разработанной анимацией (анимациями) этой же модели.**

Ниже каждая из кнопок создается в отдельном проекте, базирующемся на ранее созданном – с анимацией трехмерной поверхностной модели античной вазы с использованием виртуальной кнопки.

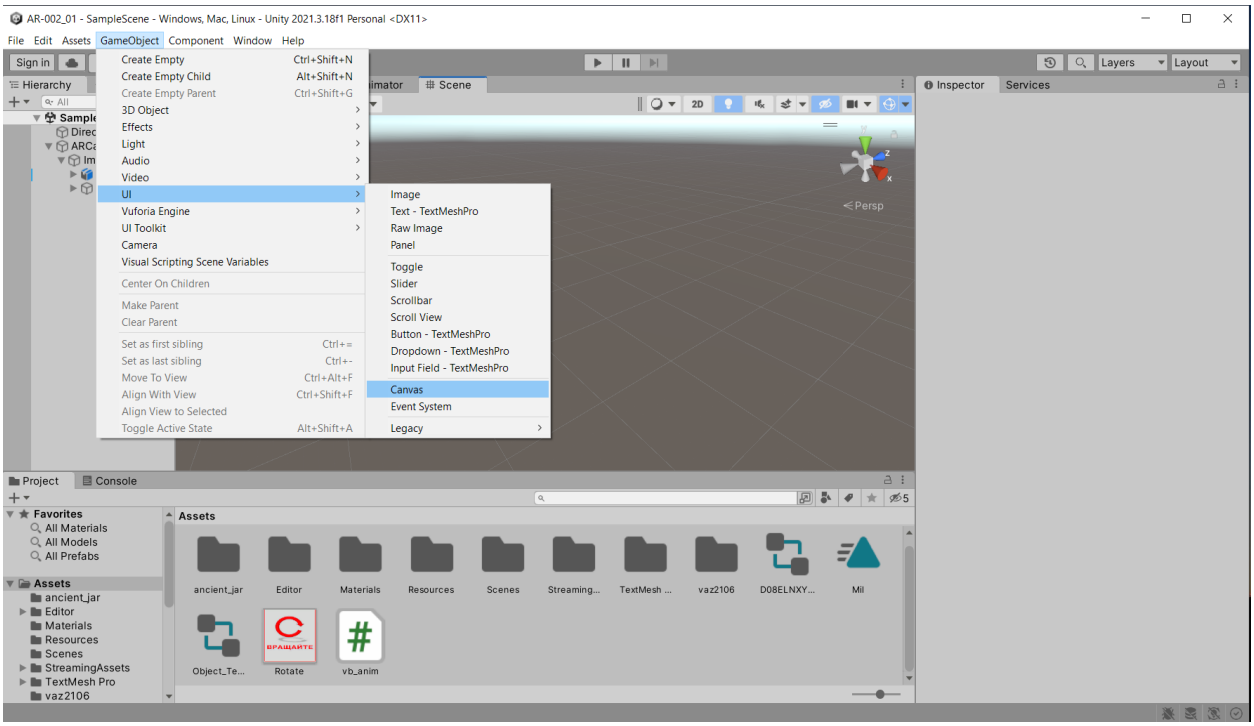
В индивидуальных заданиях (ИЗ) для ЛРN№4 все кнопки должны быть размещены в одном Проекте. В качестве базового можно использовать любой с анимациями трехмерной модели.

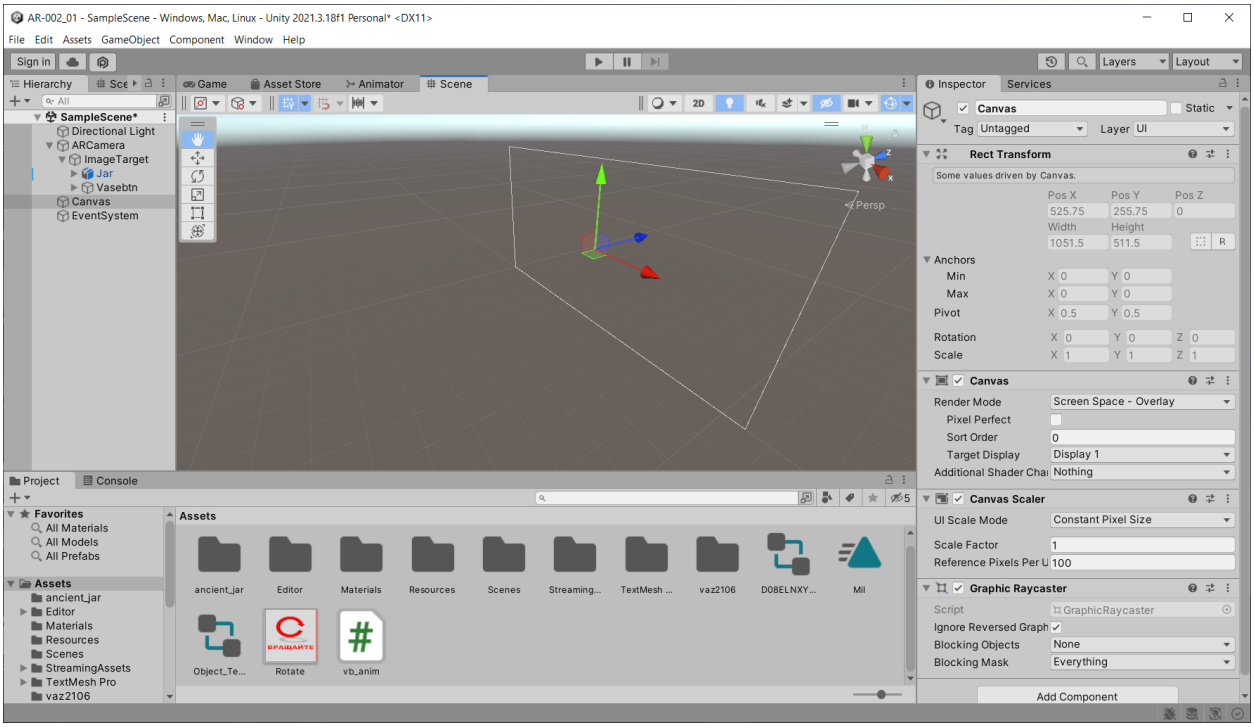
I. Перемещение трехмерной модели по оси X

Входим в проект с виртуальной кнопкой, управляющей вращением трехмерной модели с помощью виртуальной кнопки (**AR-002_01**). В данном проекте создана и сохранена одна анимация – вращение вазы (**модель Jar в Иерархии**):

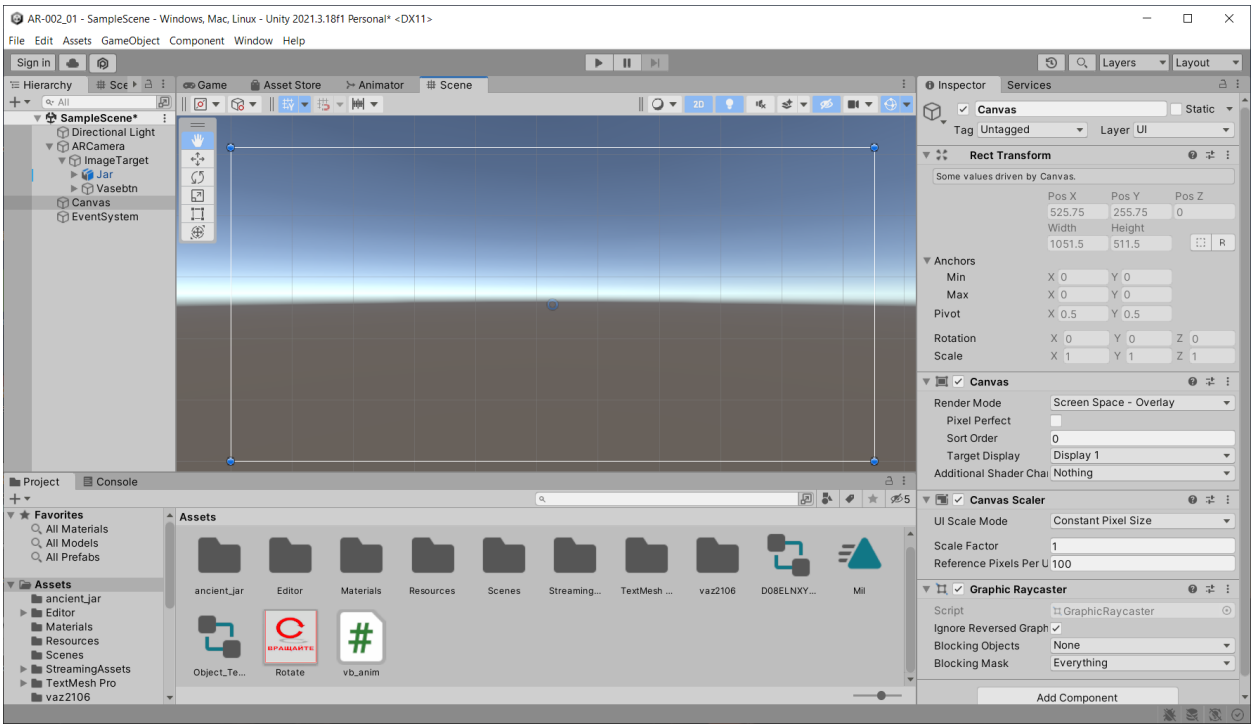


В **GameObject** вызываем канву:

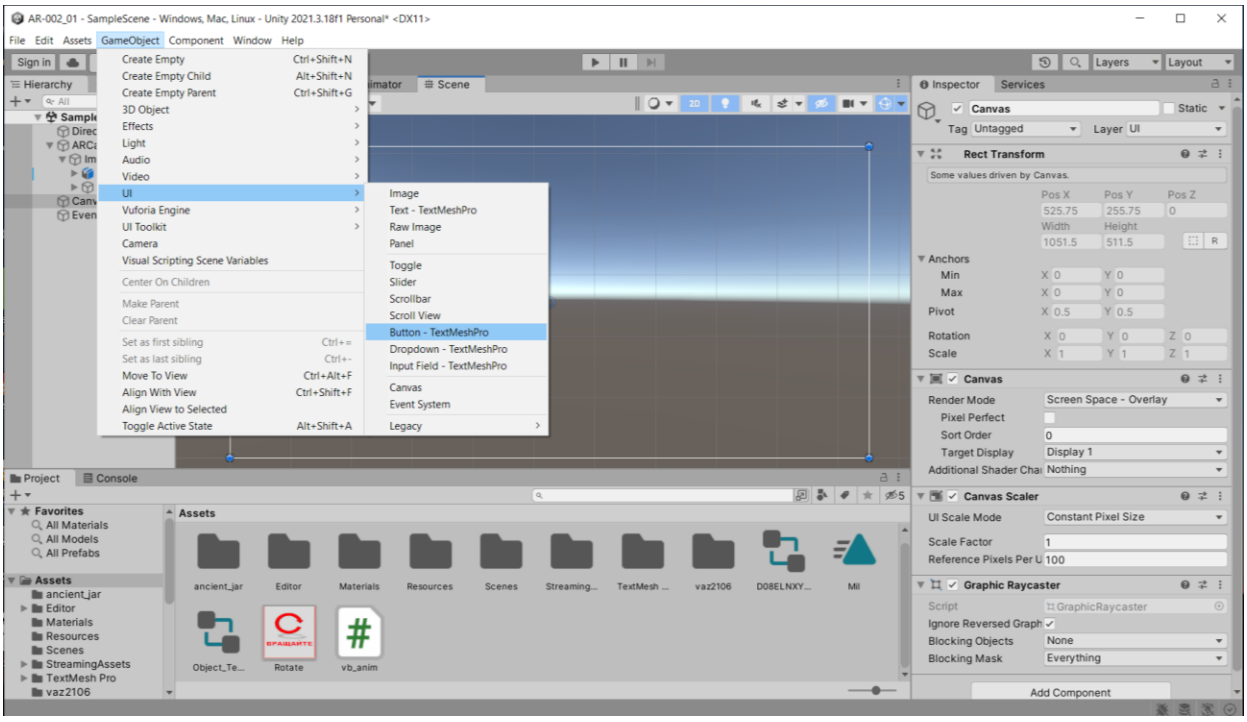




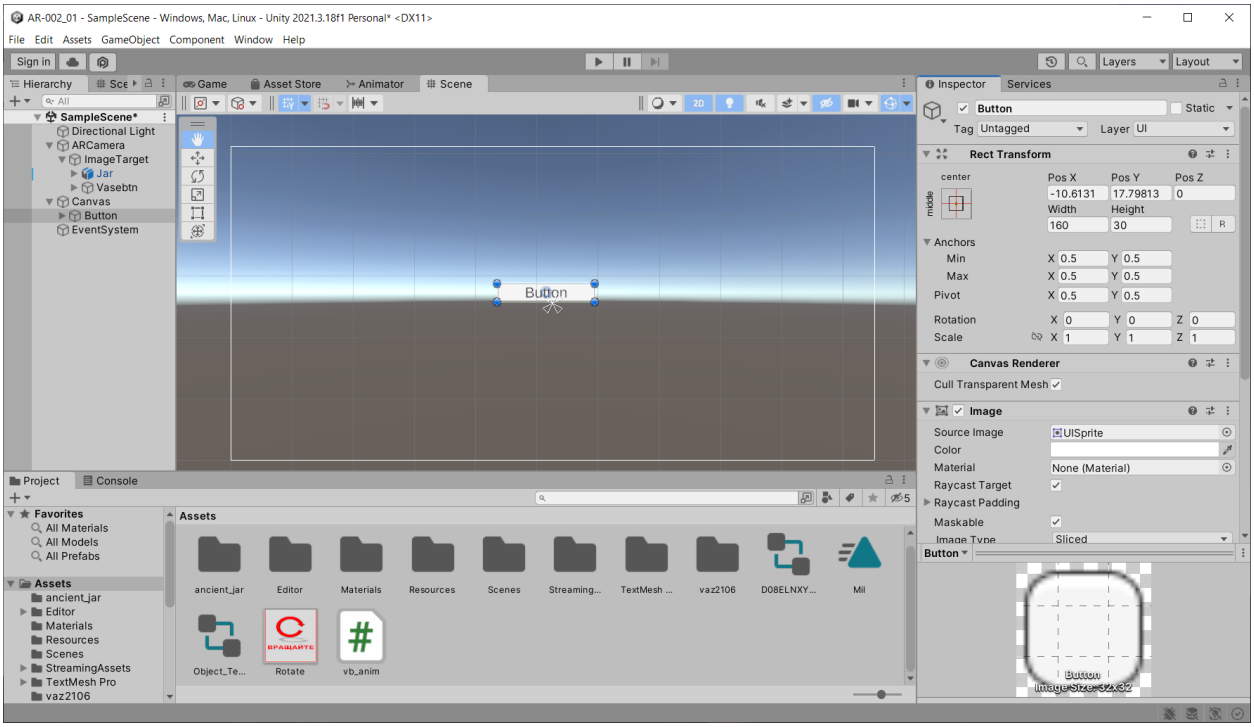
Изменения в интерфейсе и в иерархии после перехода в **2D** и манипуляций для совмещения канвы с экраном (юстировка – средняя кнопка - и перемещение – зажатая средняя кнопка):



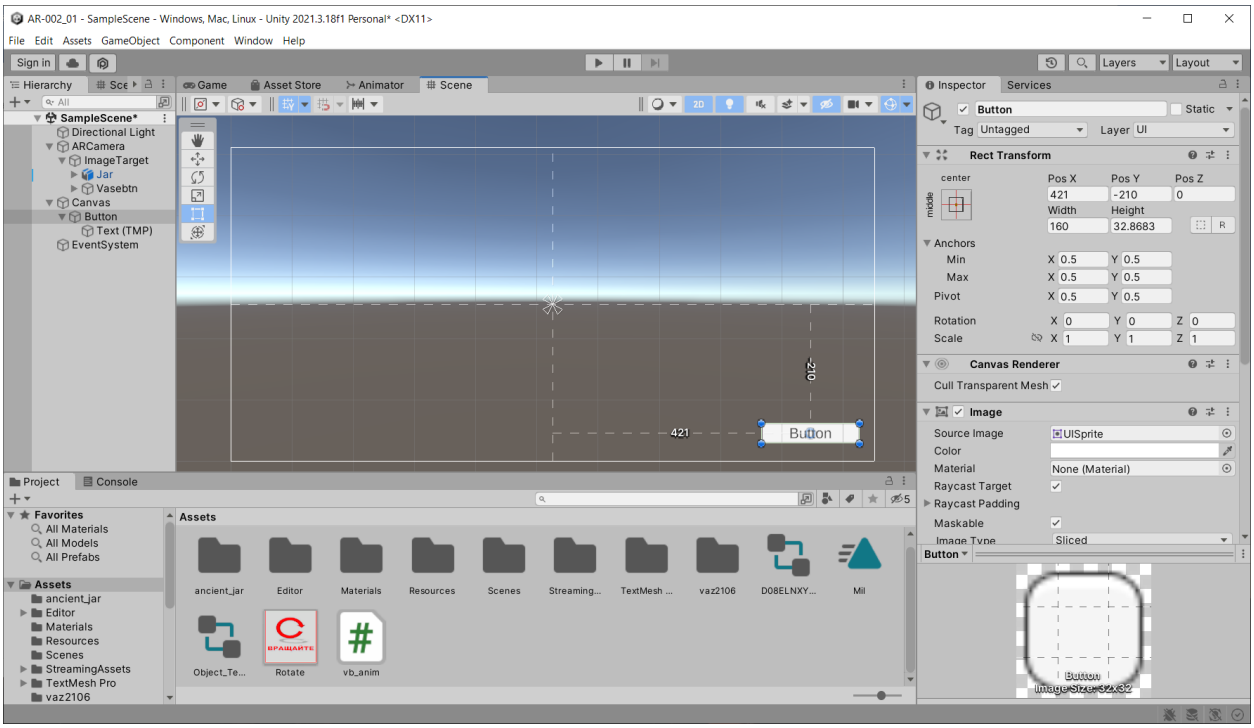
Добавляем кнопку:



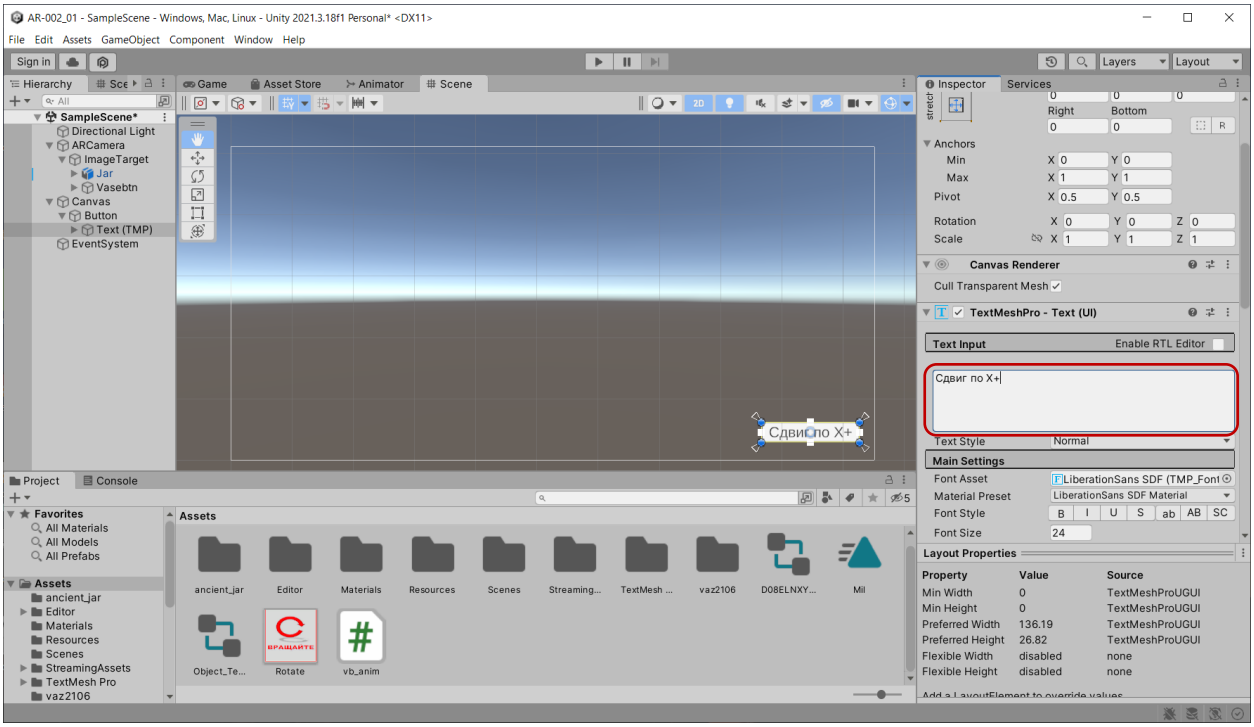
Результат:



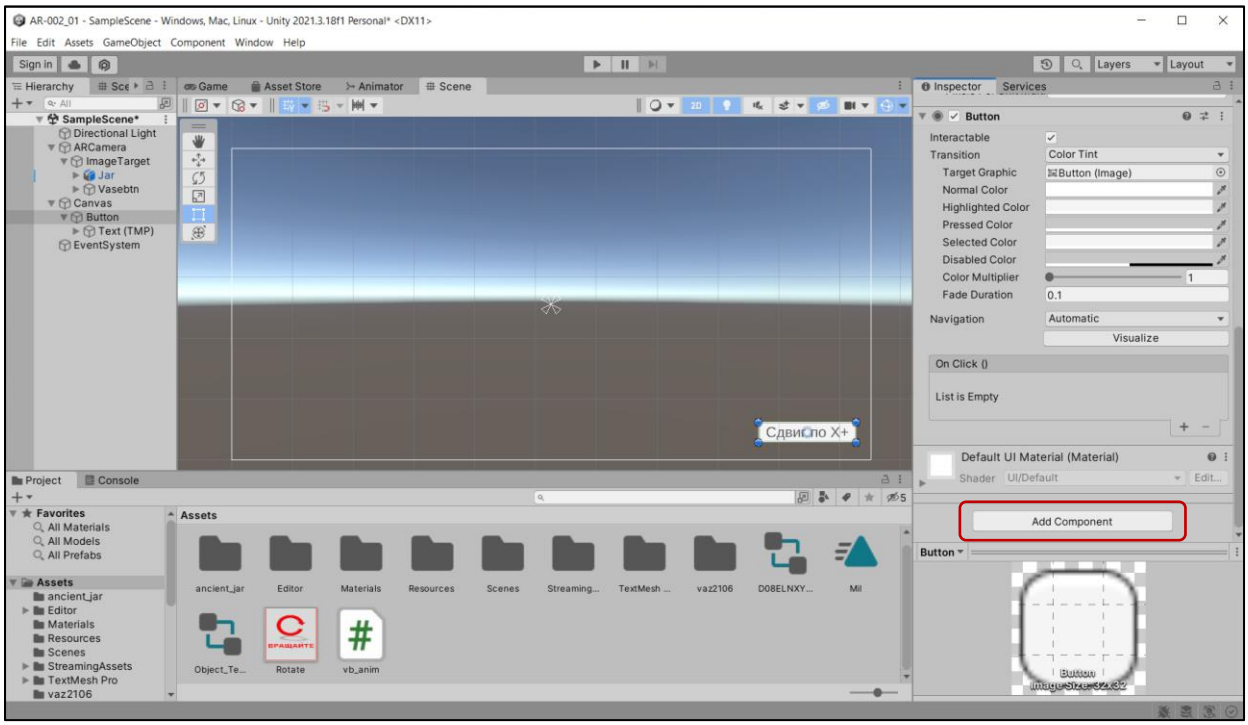
Перемещение кнопки в нужное место – есть в видео ЛР №4 (зажатая левая клавиша)



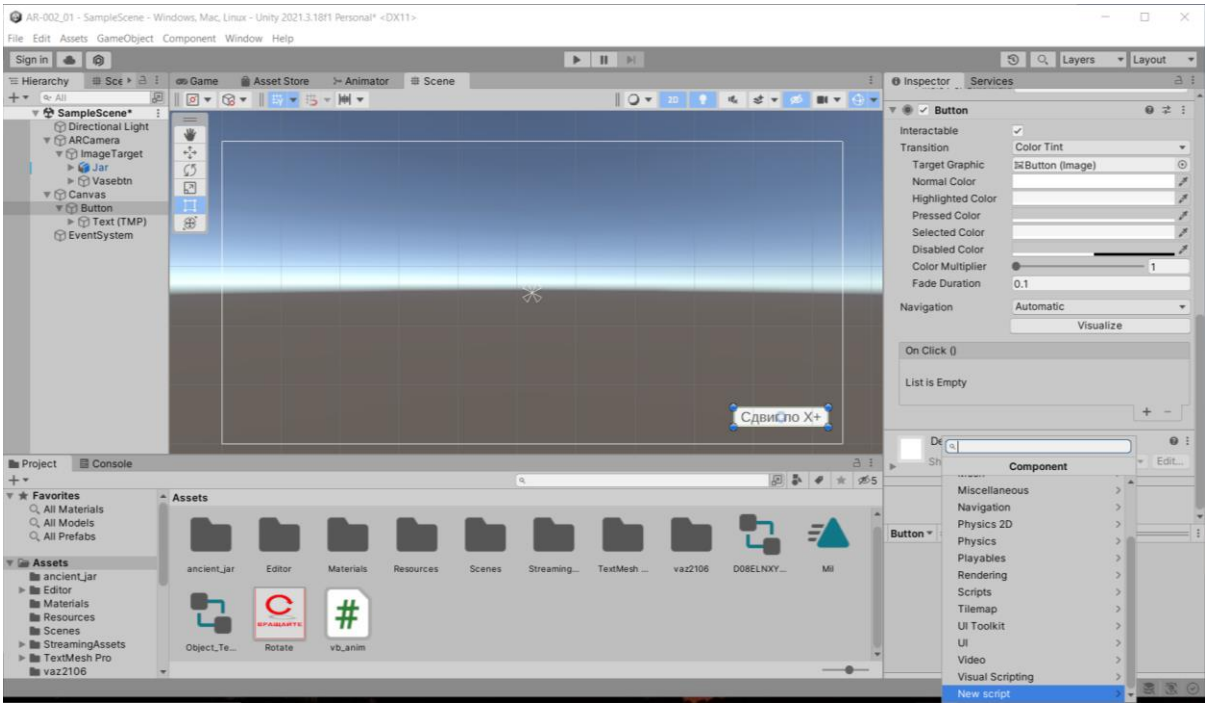
Вводим текст **TMP (TextMeshPro)**:

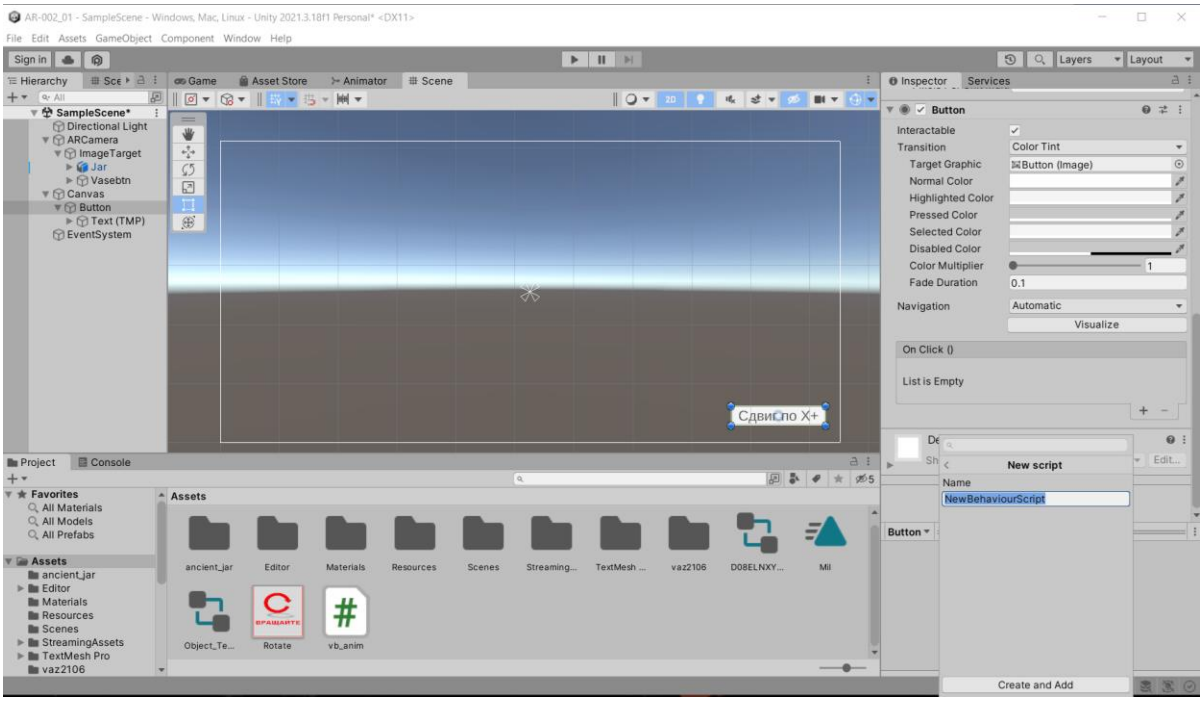


Связываем кнопку с функционалом (скриптинг):

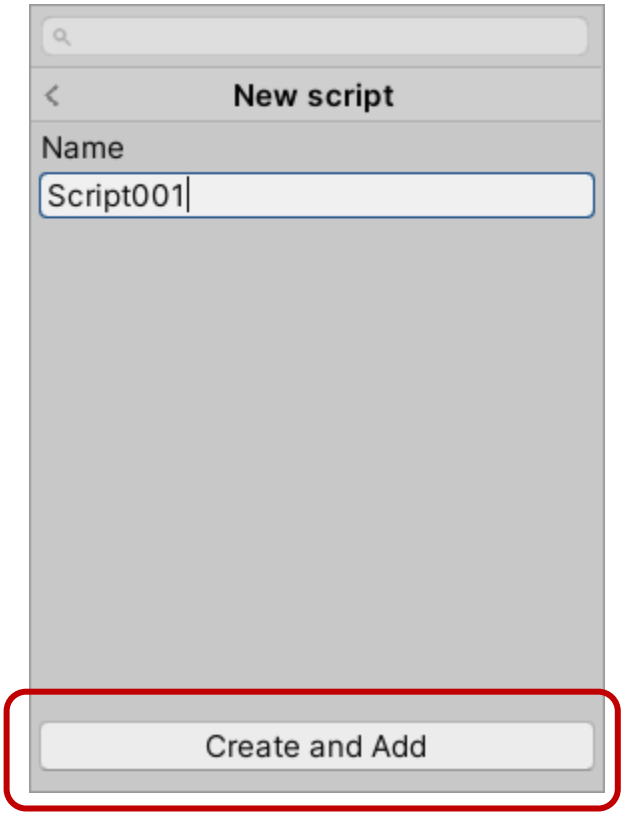


Добавляем компонент →

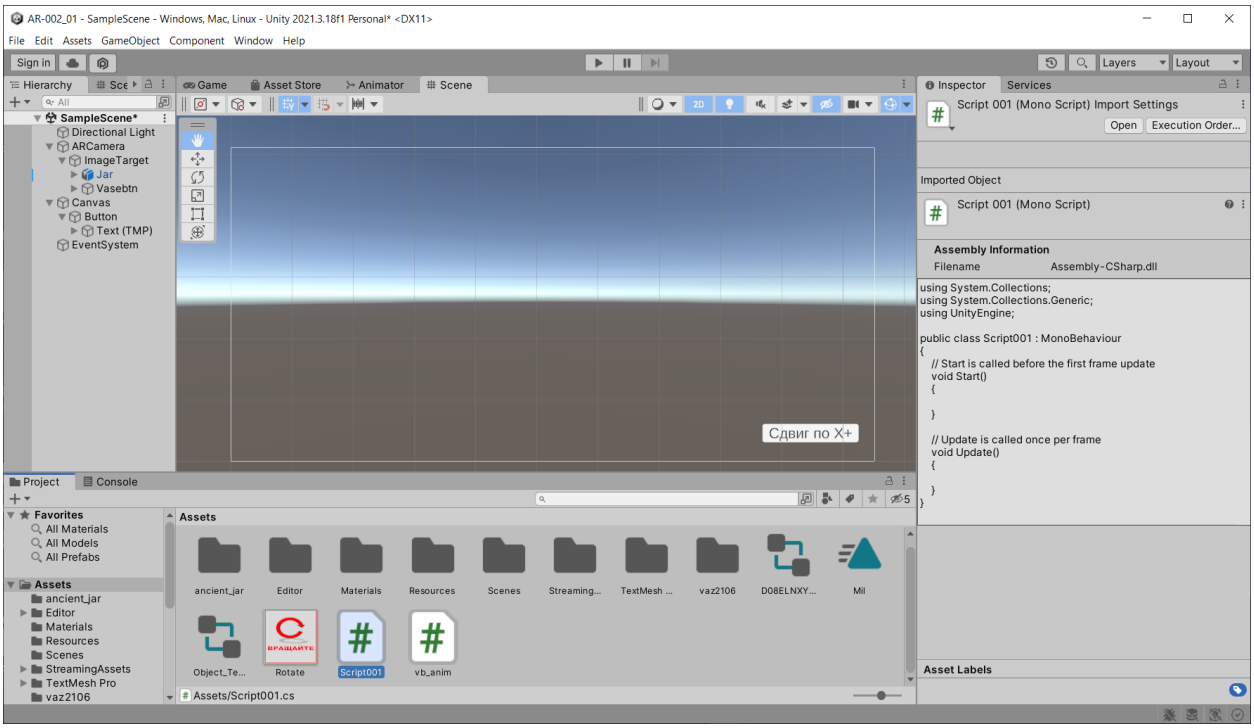




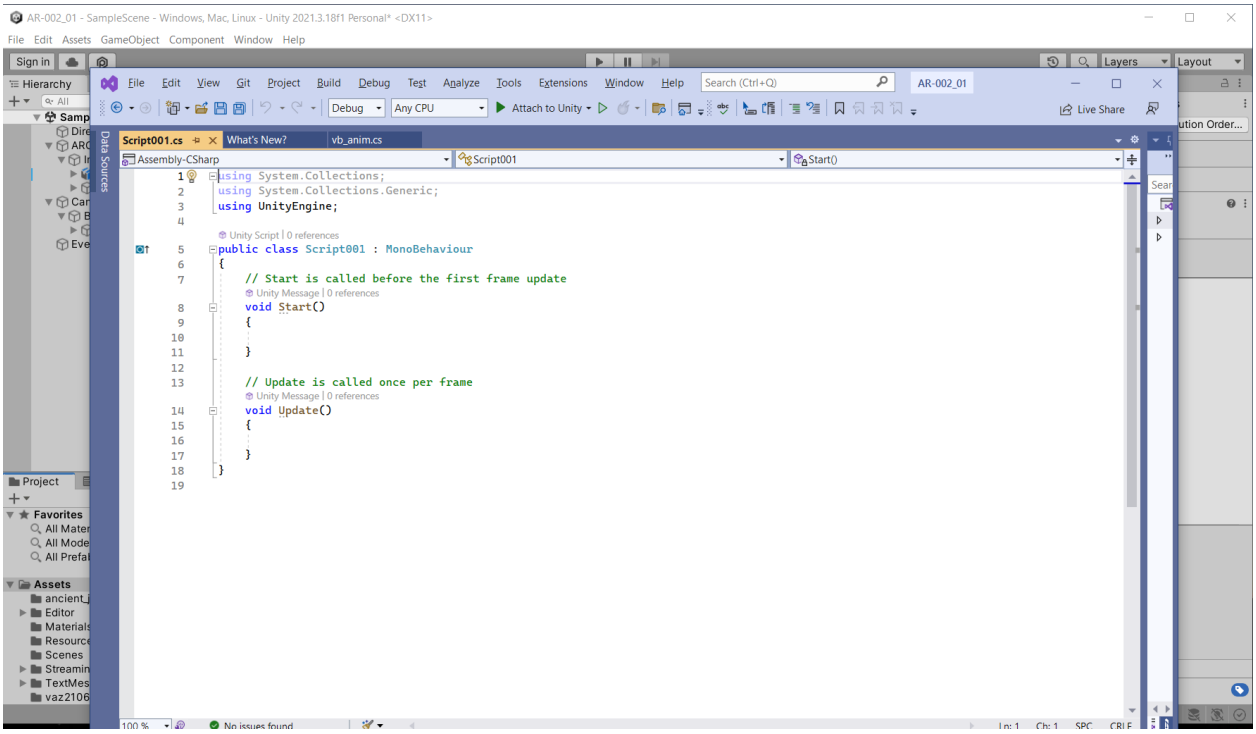
Переназываем скрипт – Script001 →



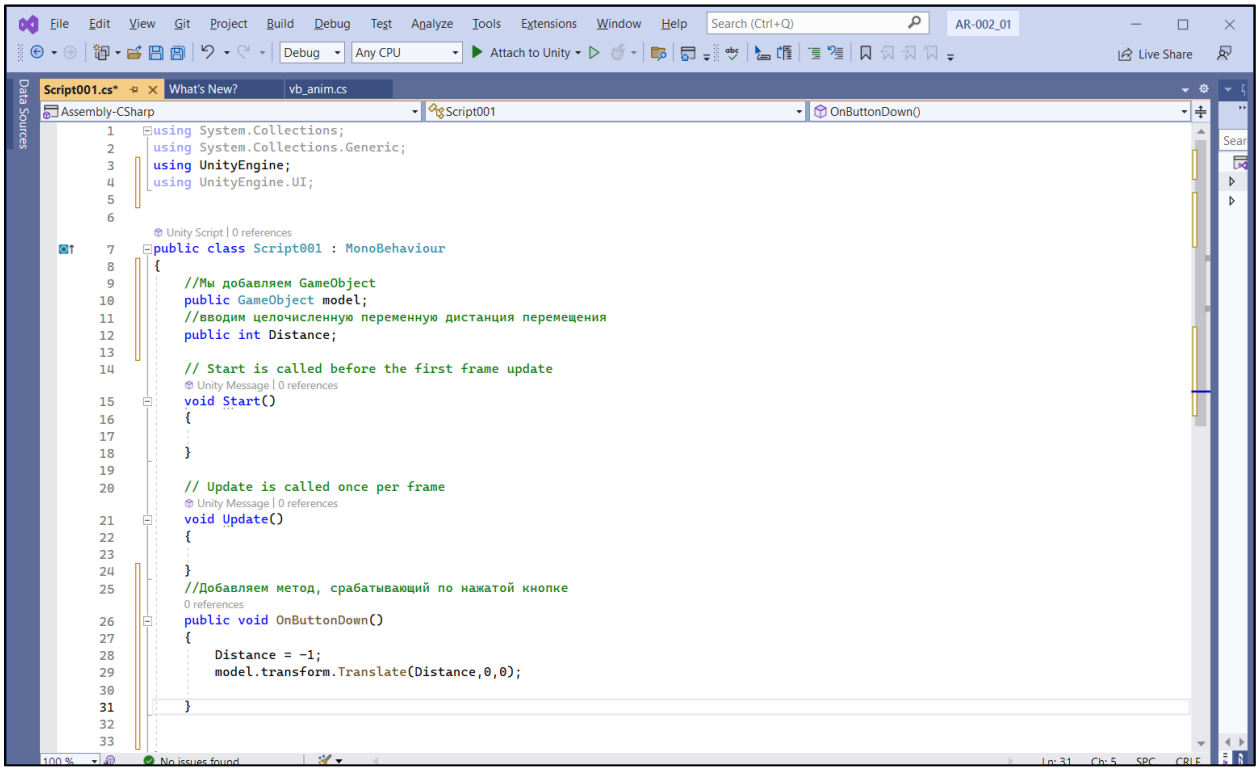
В результате в области Инспектора для **Script001** появляется шаблон скрипта:



Двойной клик – открывается **Visual Studio**, и если в предыдущей ЛР все для установлено правильно – открывается окно, в котором можно редатикровать шаблон:

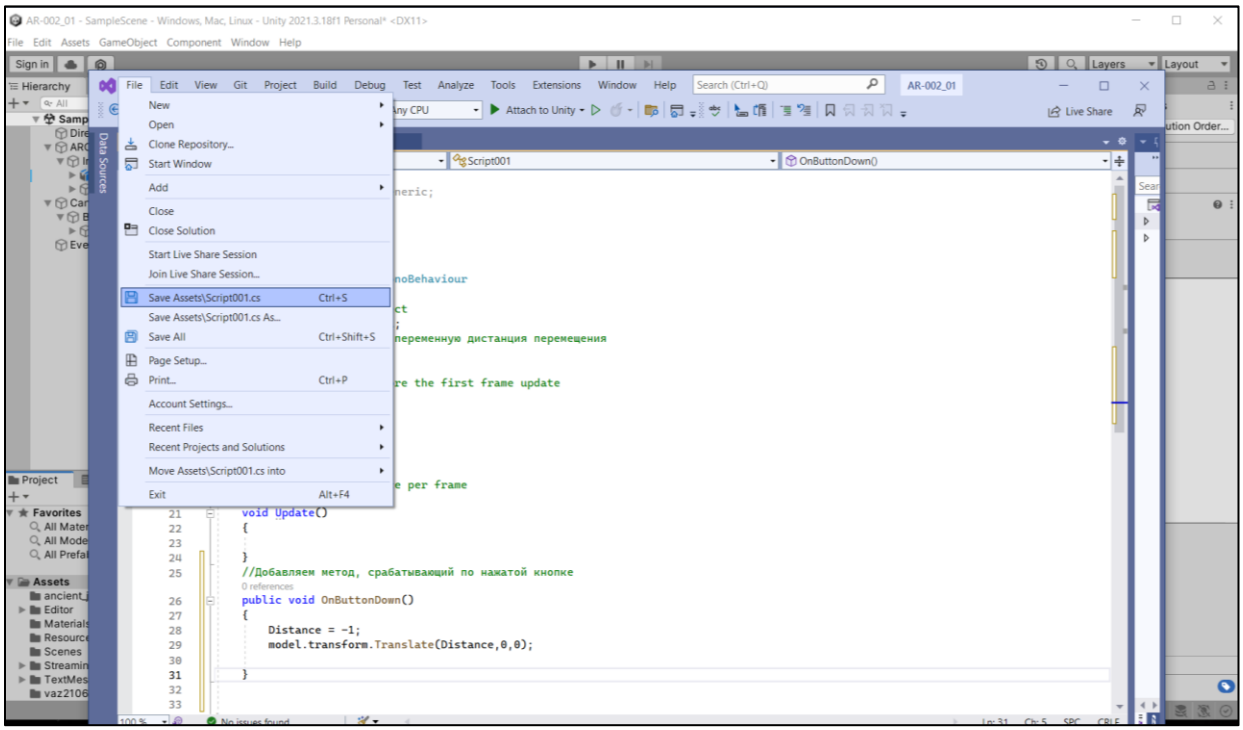


Результат (см. русскоязычные комментарии в теле скрипта):

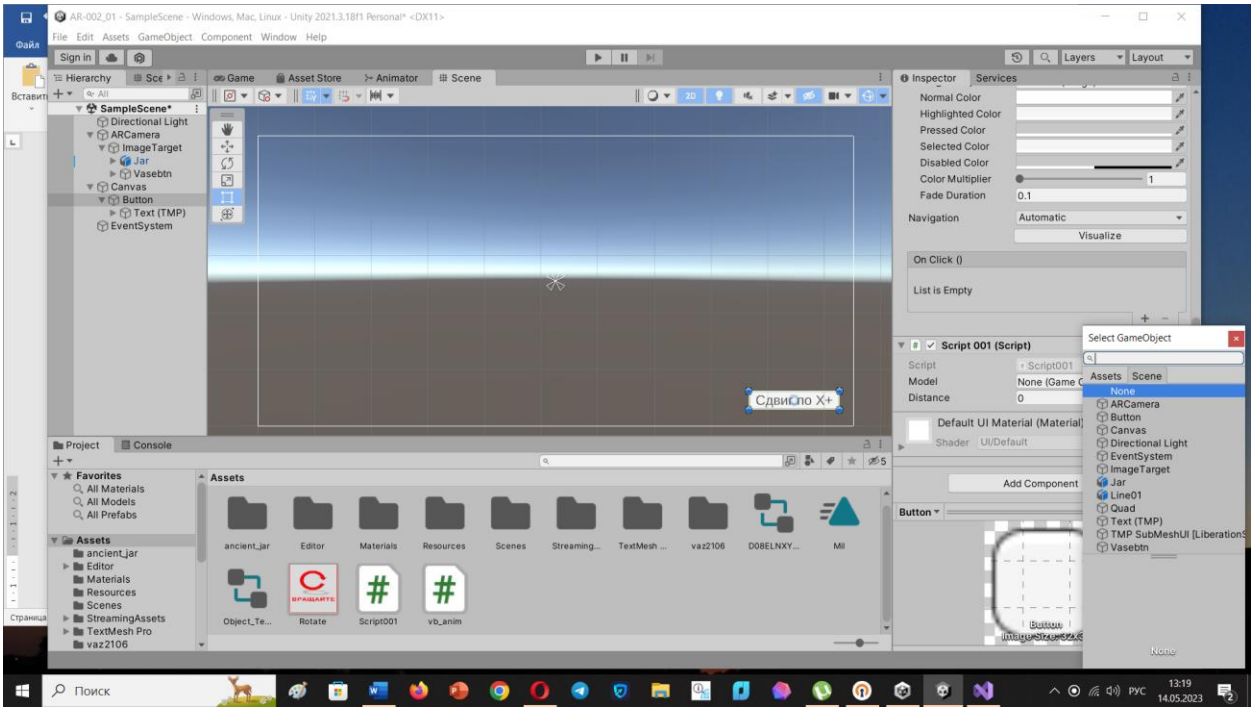


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;


Unity Script | 0 references
public class Script001 : MonoBehaviour
{
    //Мы добавляем GameObject
    public GameObject model;
    //вводим целочисленную переменную дистанция перемещения
    public int Distance;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {

    }
    //Добавляем метод, срабатывающий по нажатой кнопке
    0 references
    public void OnButtonDown()
    {
        Distance = -1;
        model.transform.Translate(Distance,0,0);

    }
```

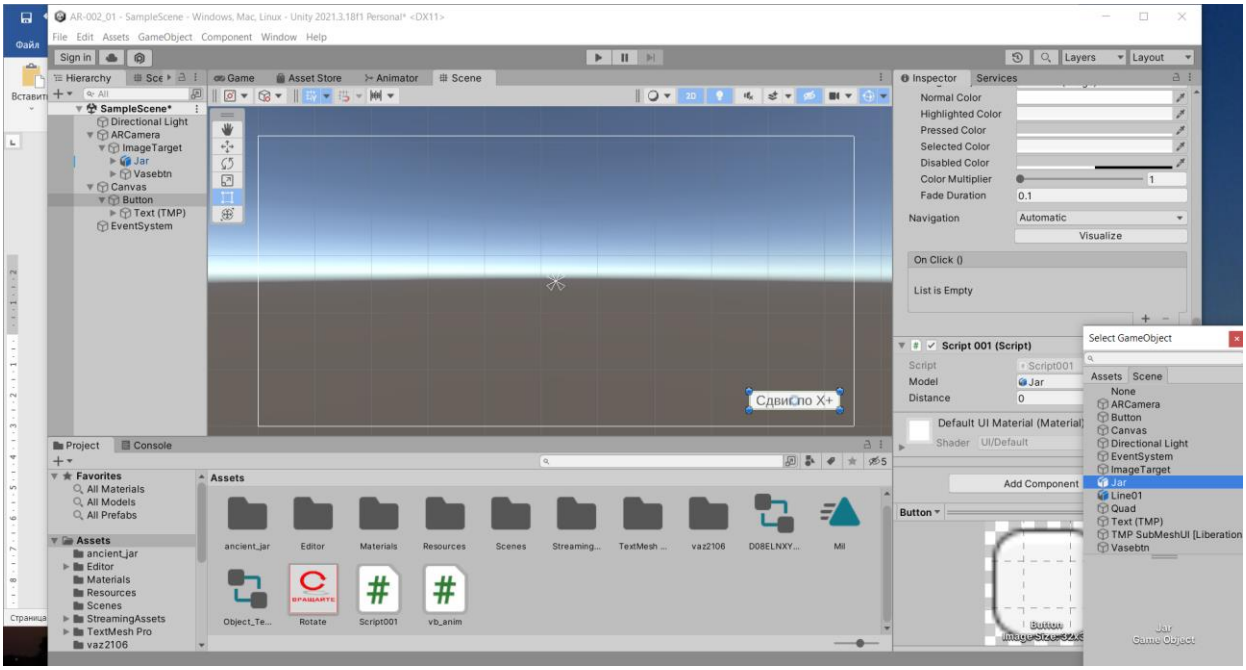
Сохраняем скрипт в ассетах:



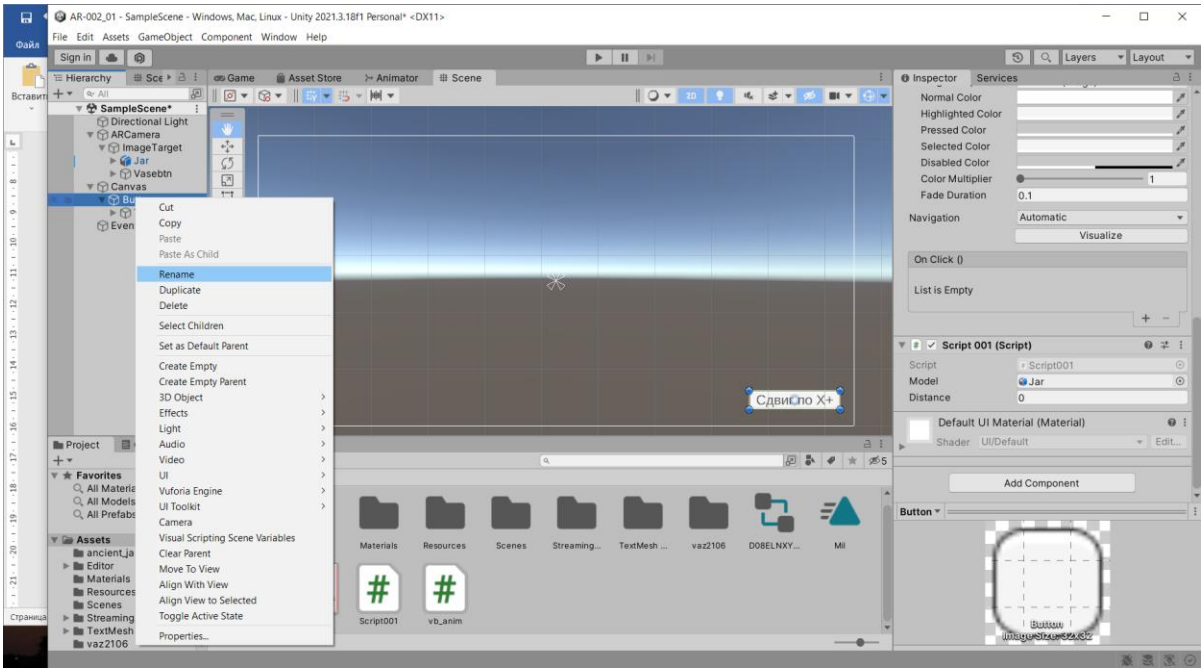
Теперь объект из скрипта **model** надо привязать к модели в сцене. Сделаем это через интерфейс инспектора по **button** (если ранее все было сделано правильно), а не с помощью **«drag-and-drop»**, как это показано в видео для ЛРN№4 для предыдущей версии **Unity**:



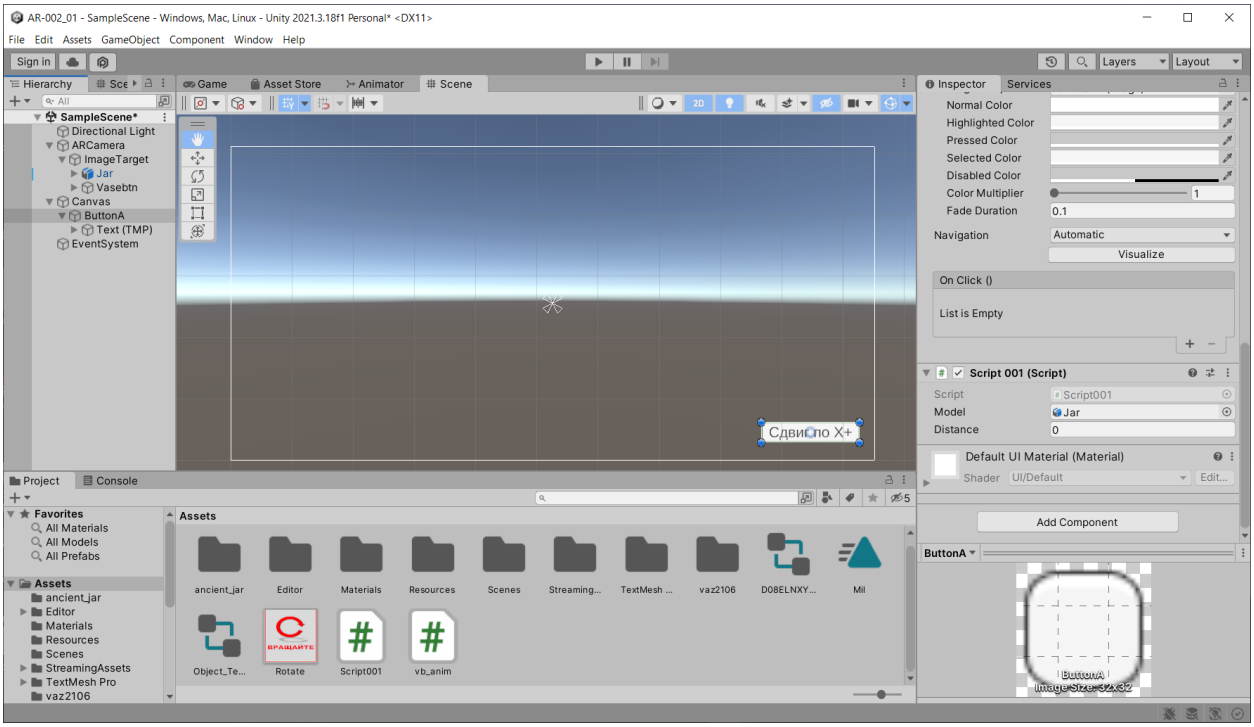
В открывшемся списке ищем и выбираем наш объект – **Jar** (ваза)



В Иерархии переименуем кнопку (вдруг у нас их будет несколько в одном пространстве Проекта):

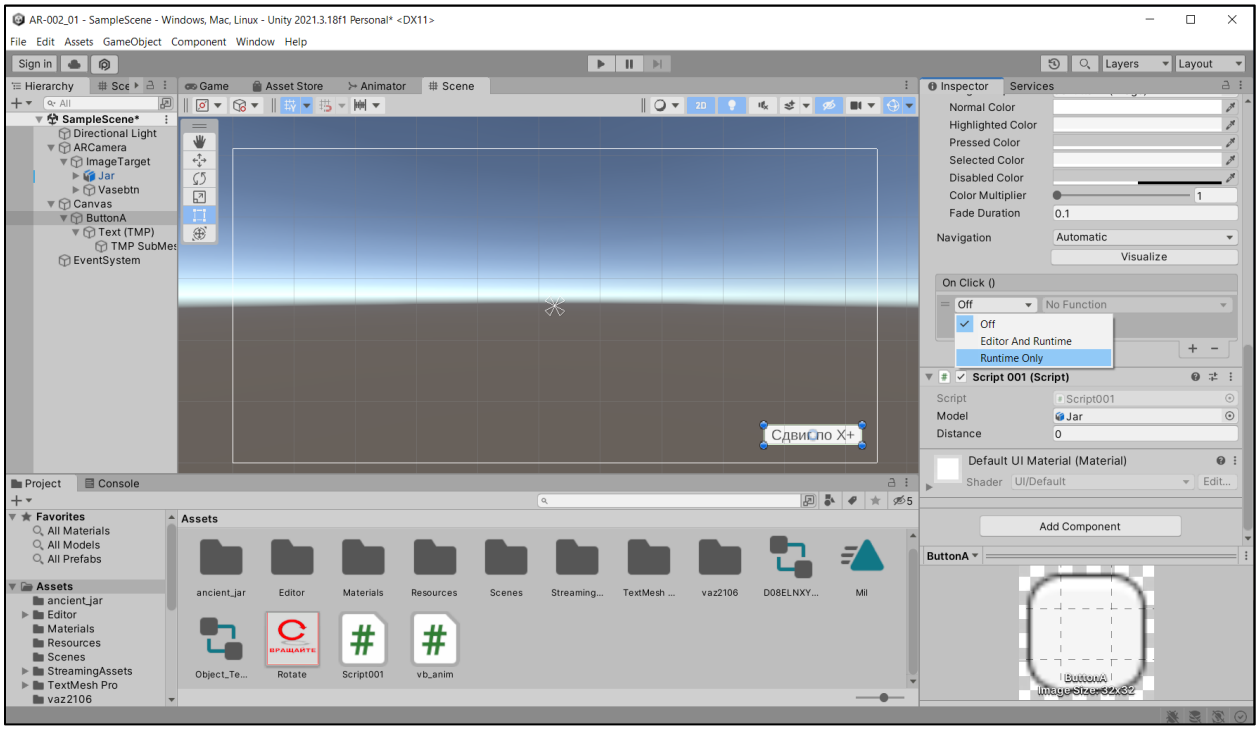


Назовем эту кнопку – **ButtonA →**

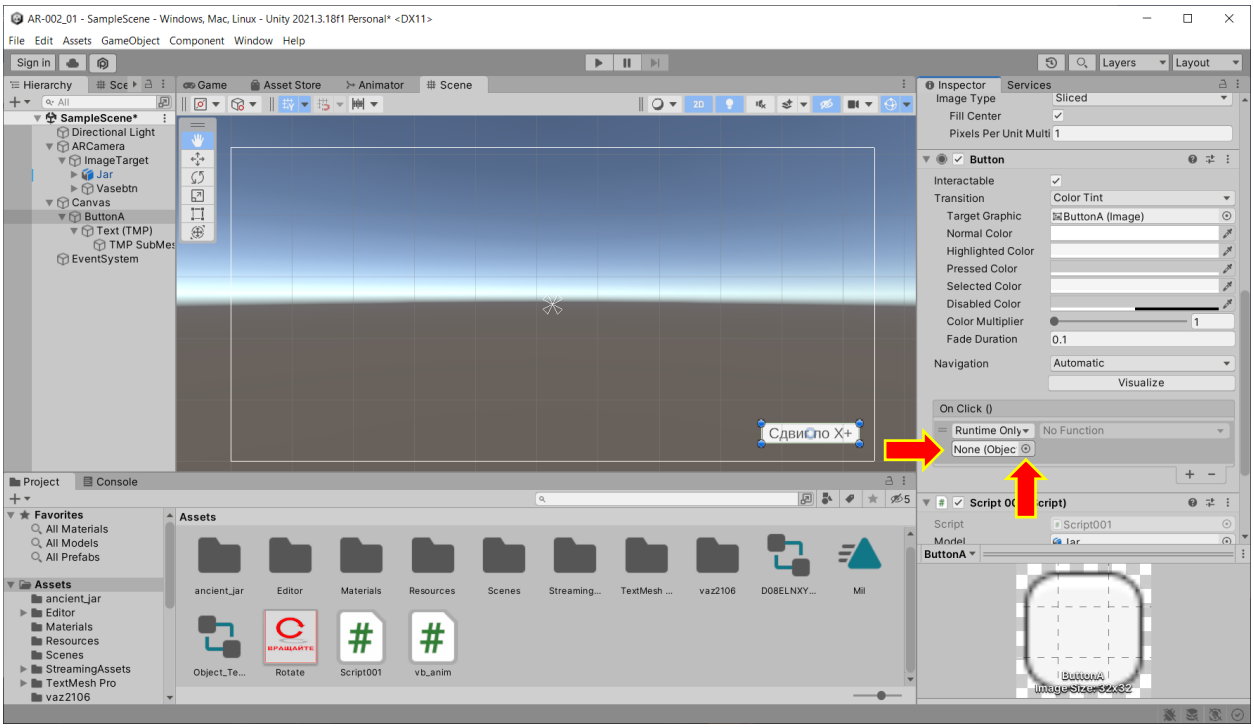


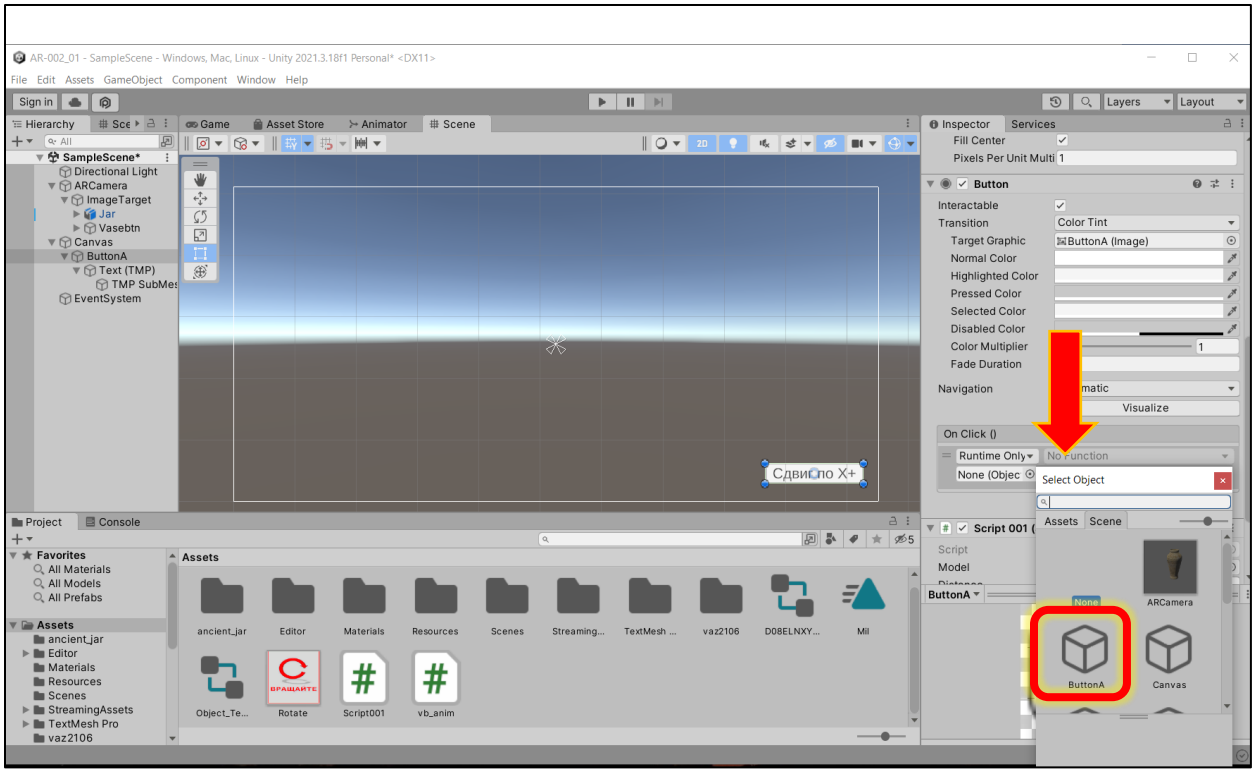
В инспекторе для **ButtonA** находим свойство **OnClick** и настраиваем его на действие, которое прописали в скрипте **Script001**.

Для этого приписываем свойству **OnClick** режим работы **Runtime Only** →

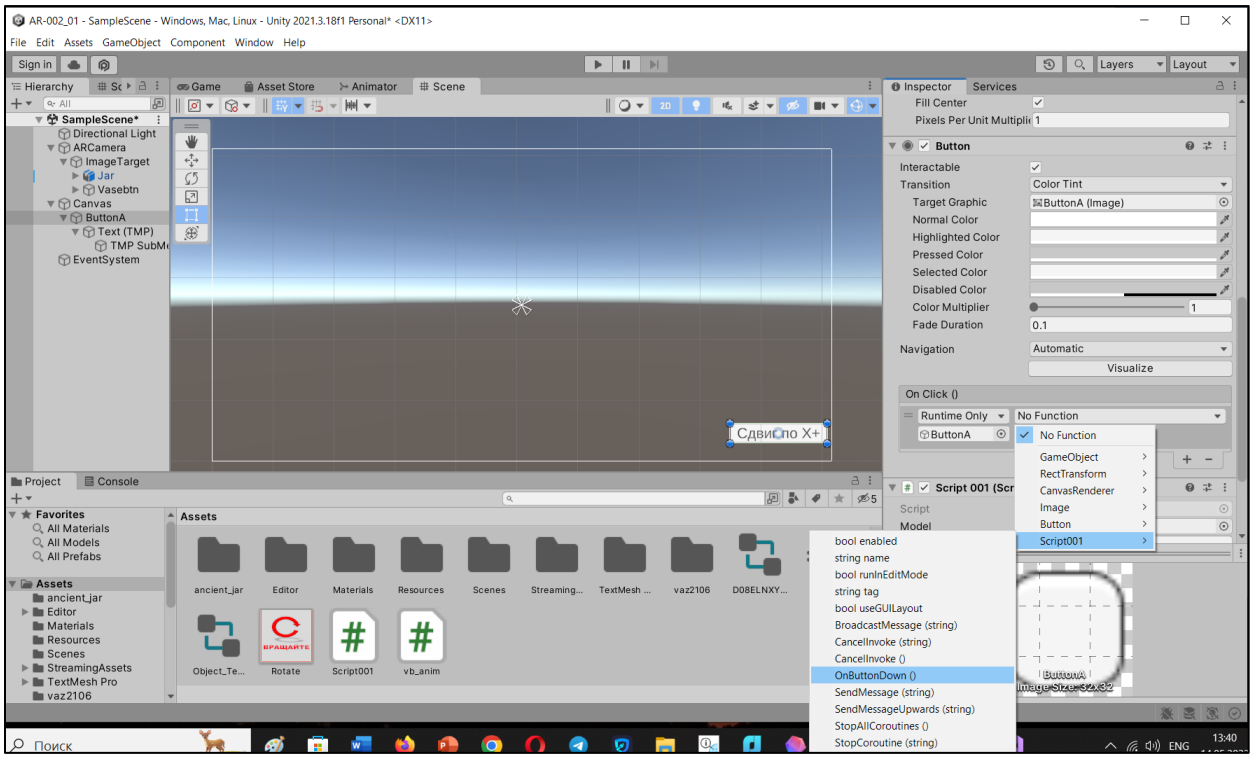


Рядом в поле ввода объекта вызываем доступный список объектов и ищем нашу кнопку **ButtonA**→

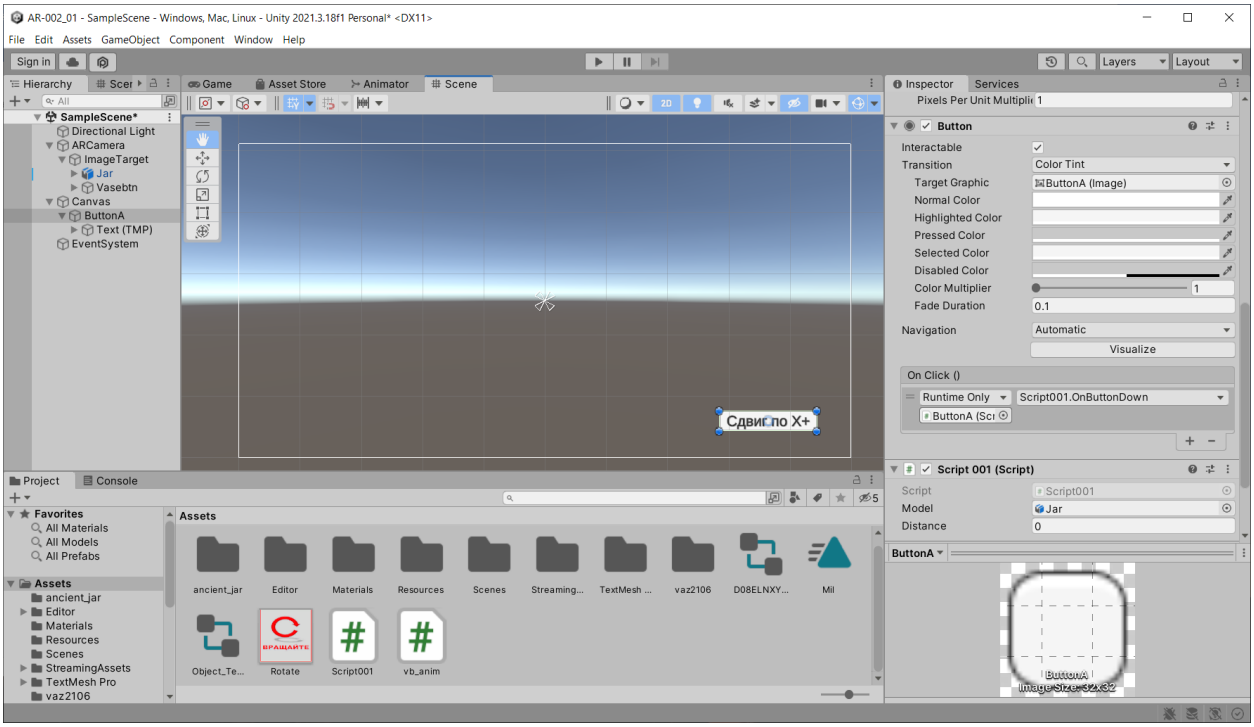




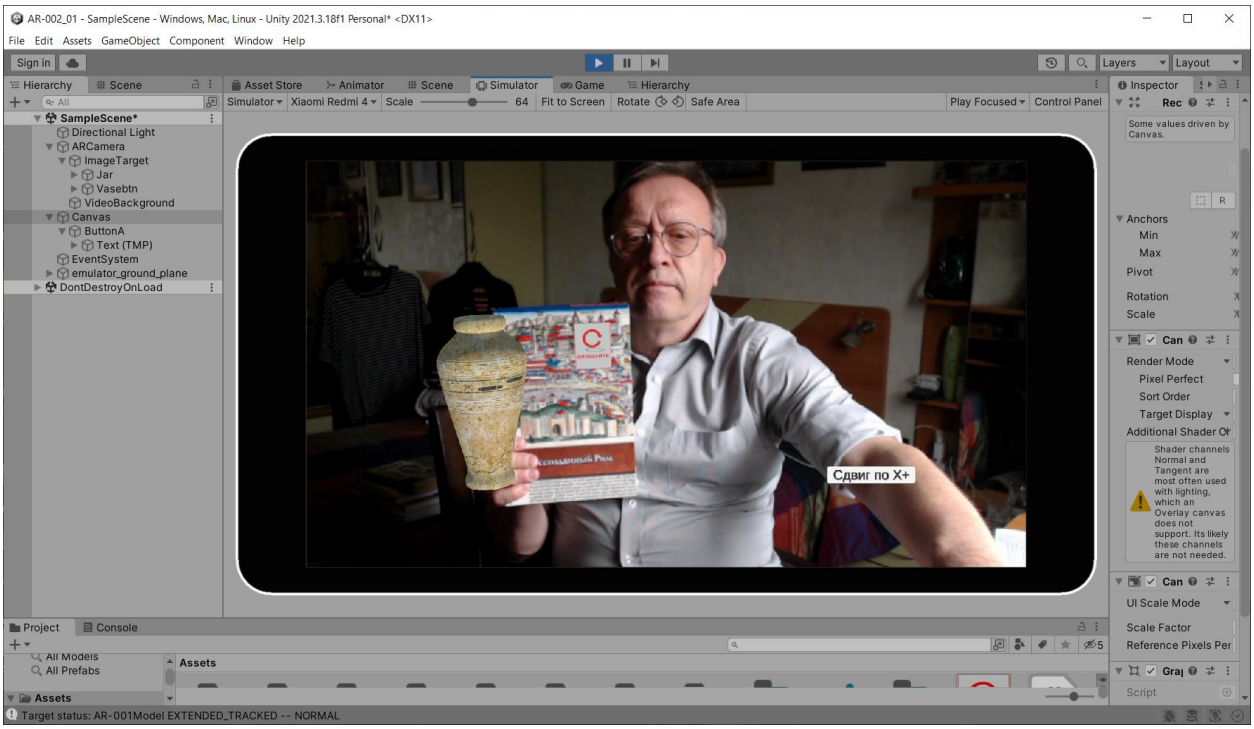
И выбираем метод, который мы добавили в шаблон скрипта для собственно действия по перемещению →

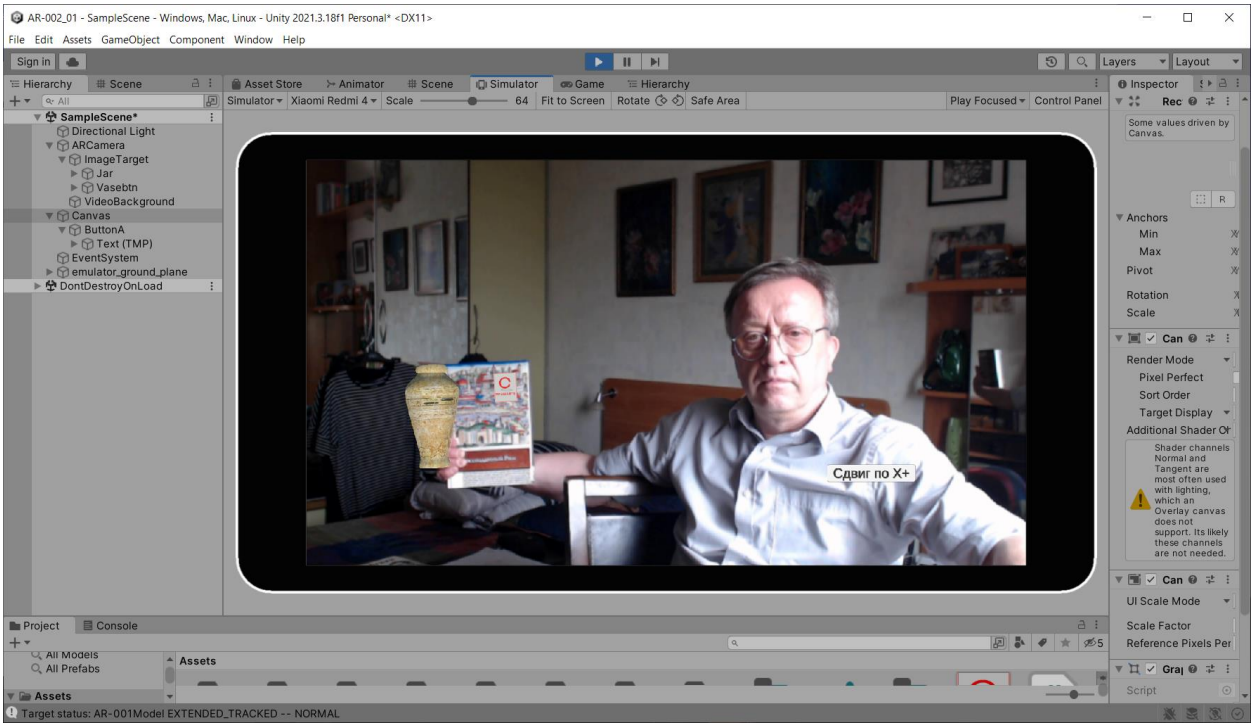


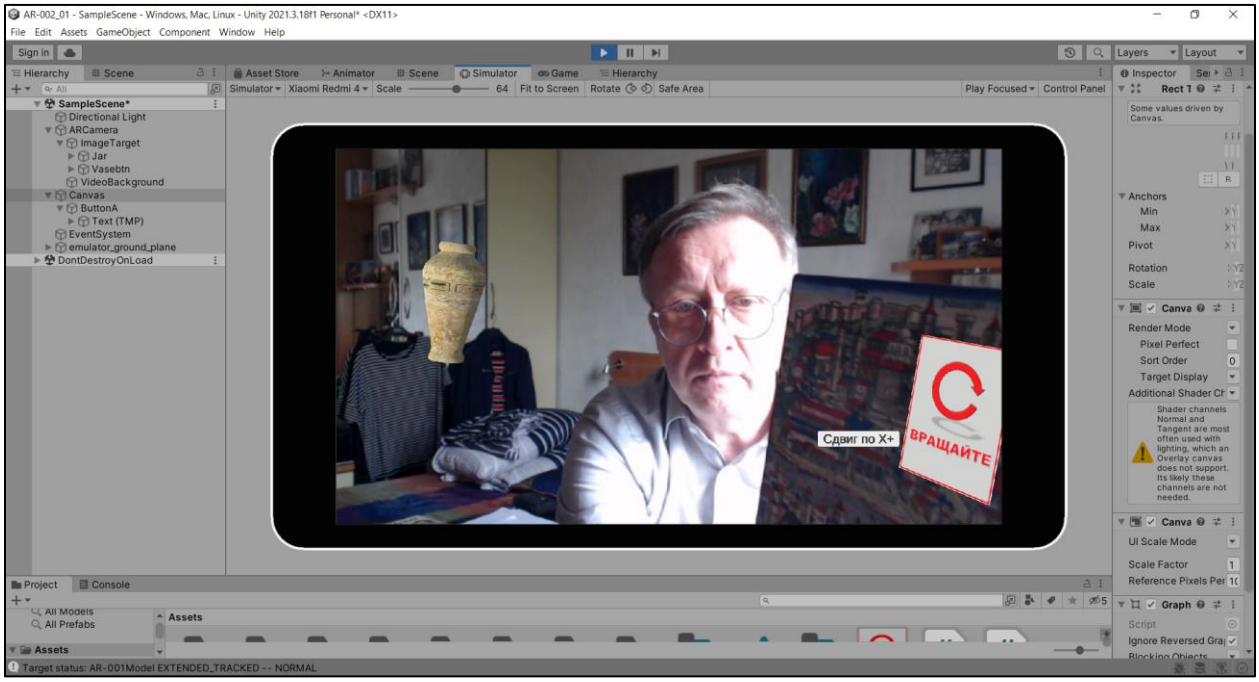
Результат:



Работу кнопки можно проверить в режиме **Game (→ Play)**:

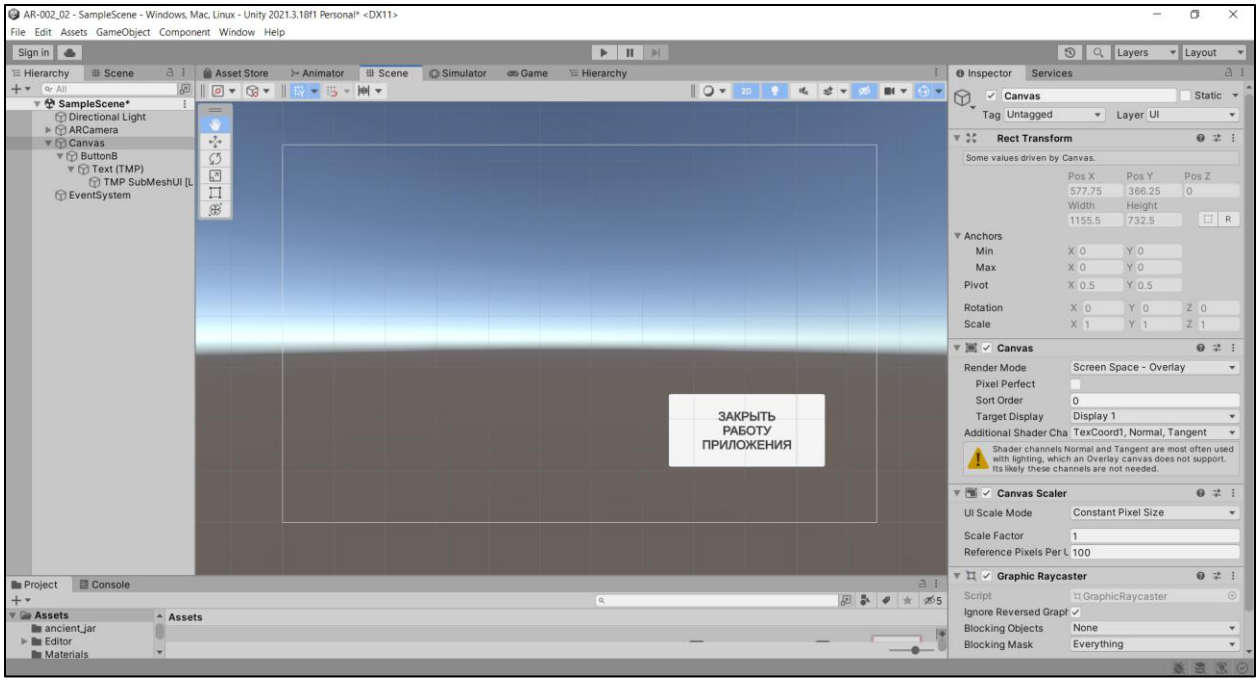




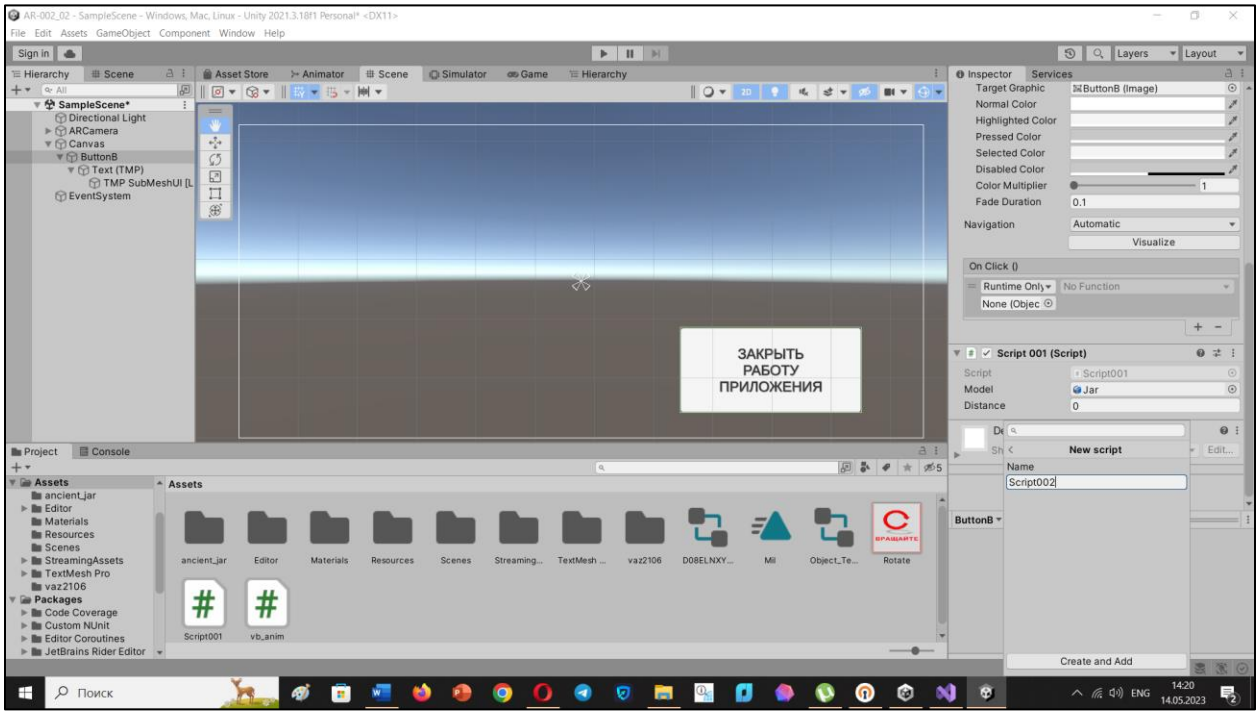


II. Кнопка выхода из Приложения

По аналогии с шагами в предыдущем примере в готовой сцене (Проект с виртуальной кнопкой) добавляем канву **2D** и **Button** на этой канве для выхода из разработанного приложения («сцены»):

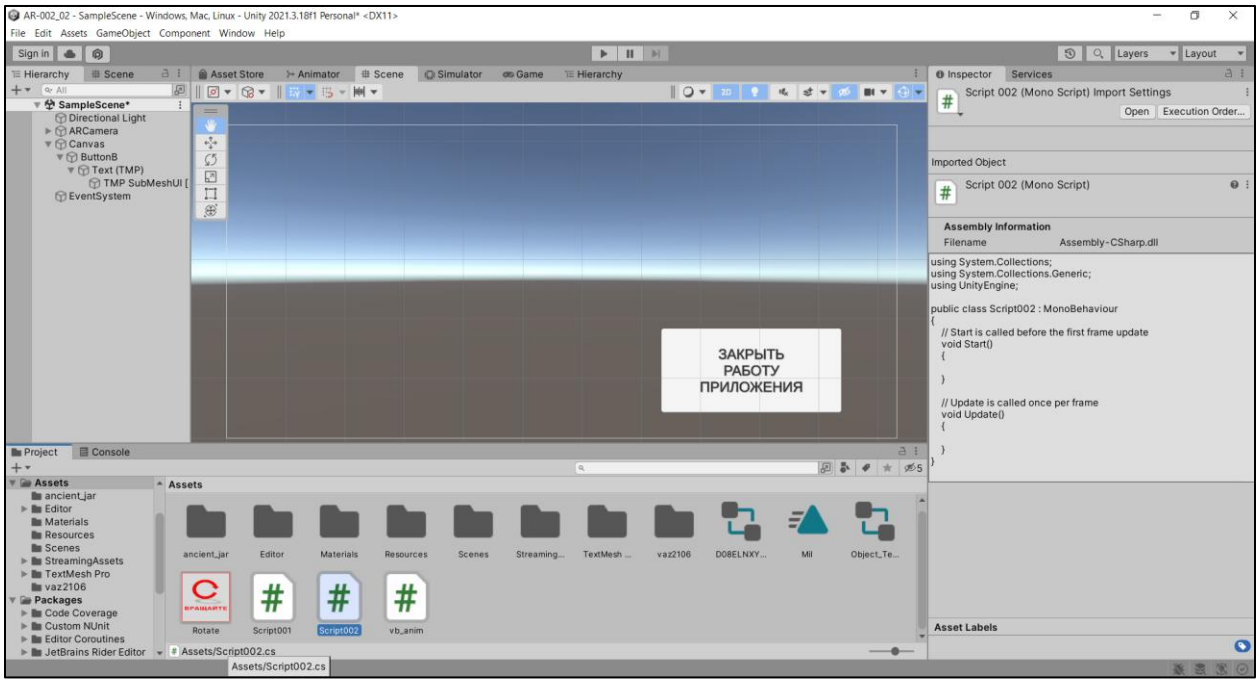


Последовательность шагов аналогична **пункту I.**

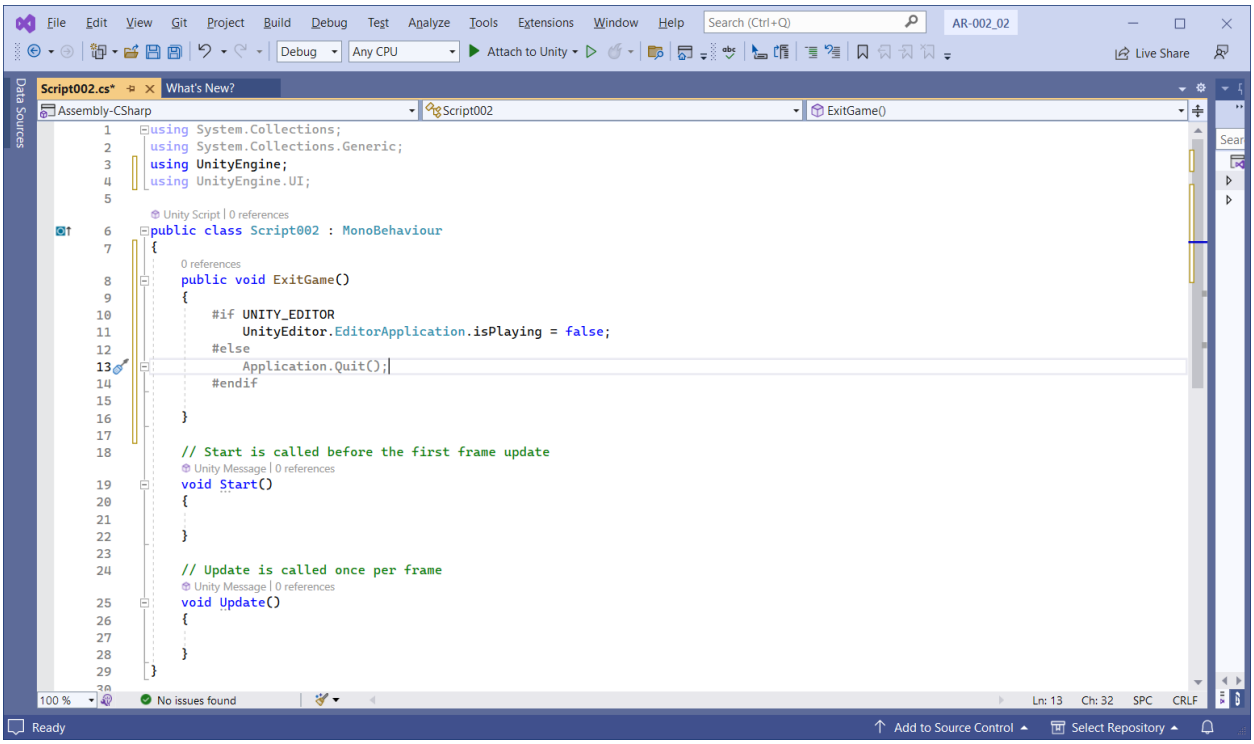


Для этой кнопки добавляем скрипт, который бы закрывал работу приложения.

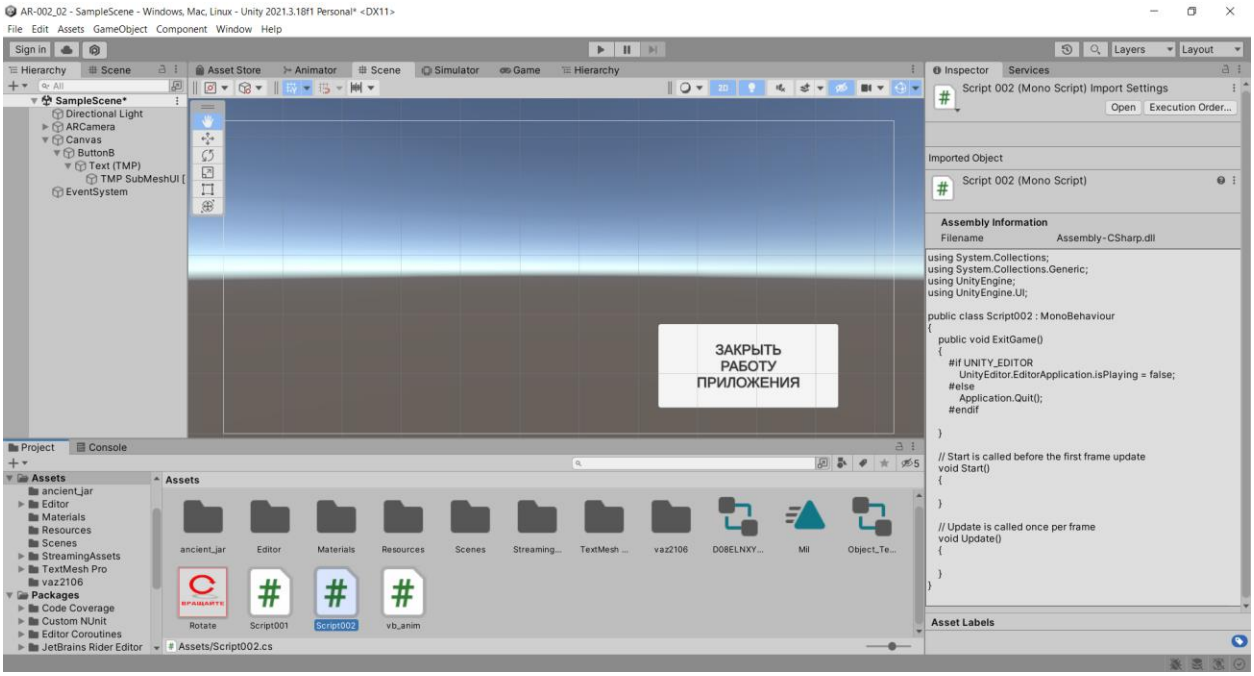
Назовем его **Script002** →



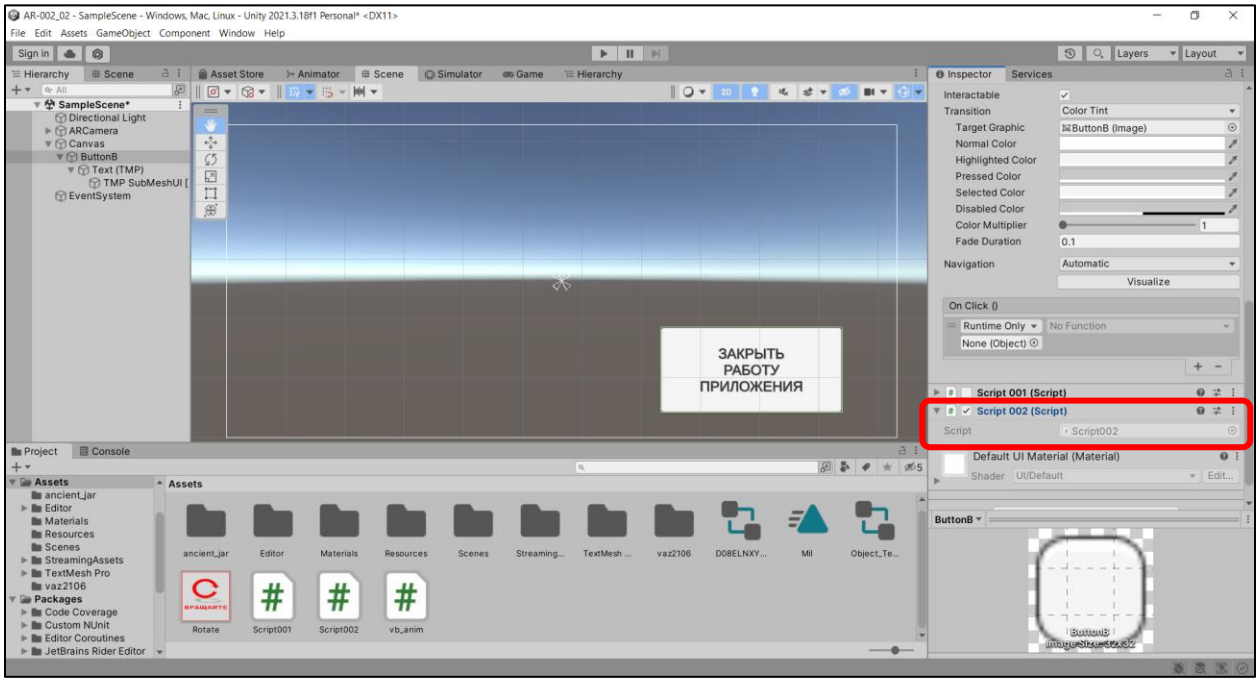
Переходим в **Visual Studio**. Отредактированный скрипт для данной операции - ниже:

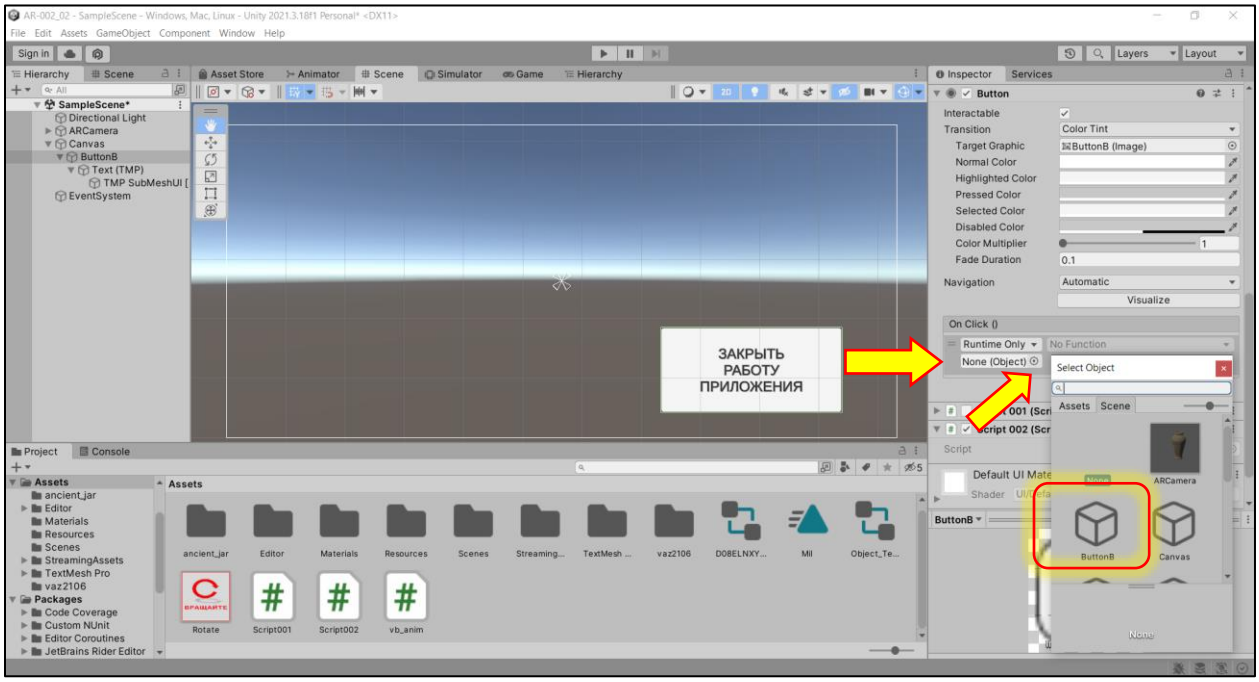


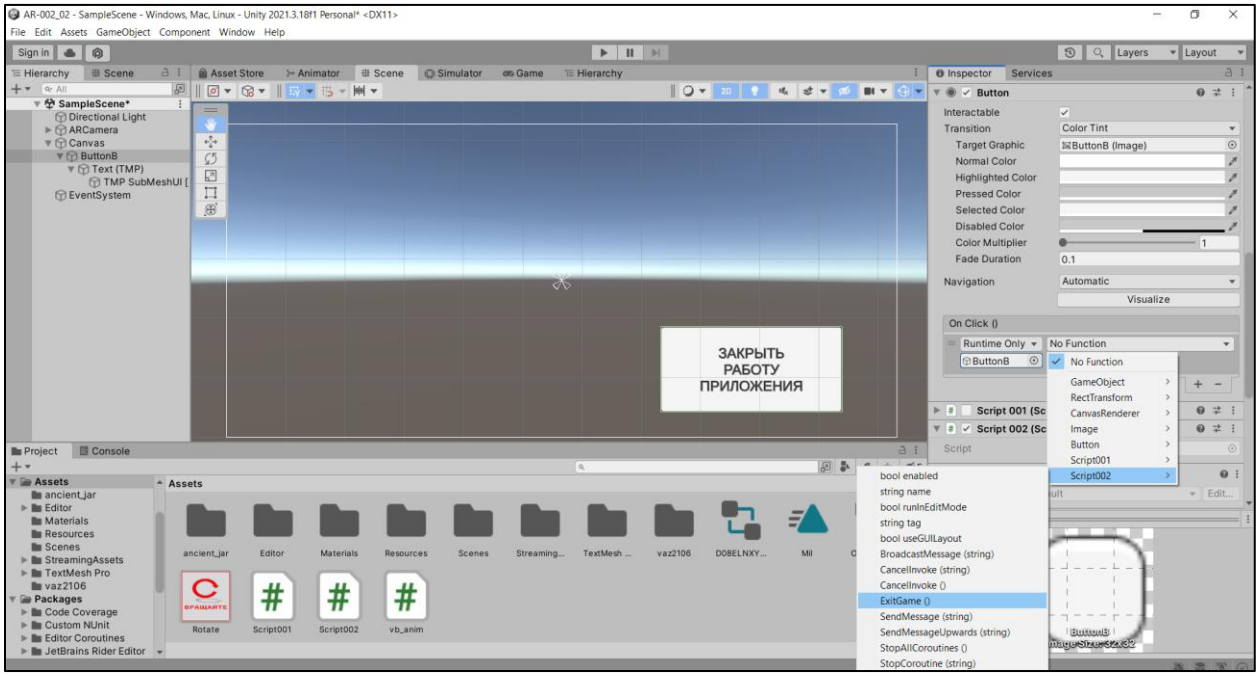
Сохраняем скрипт в **Assets →**



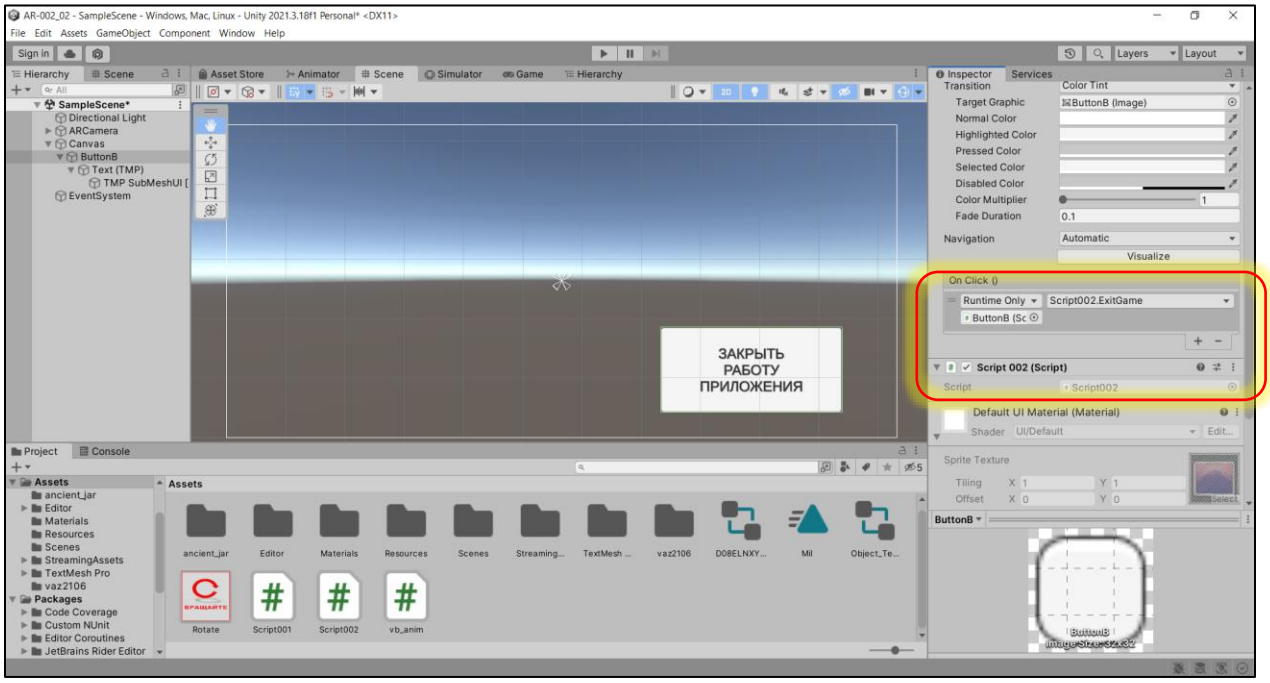
Дорабатываем кнопку **ButtonB** для работы по этому скрипту:



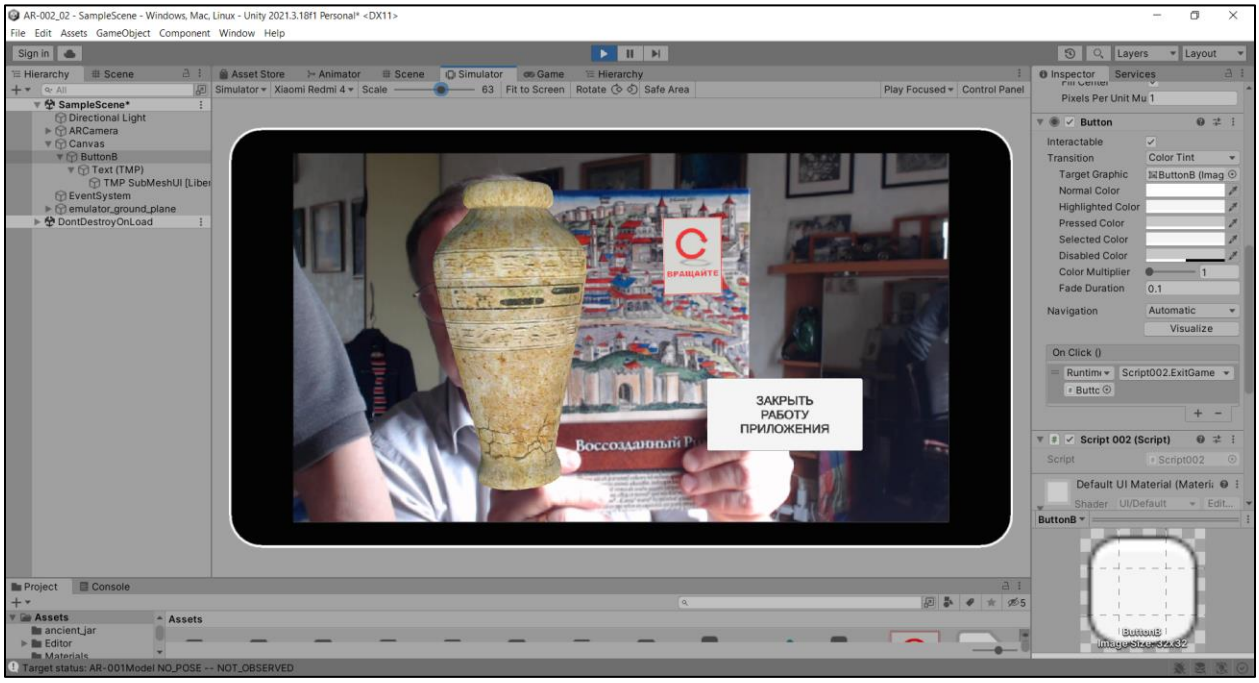

ЗАКРЫТЬ
РАБОТУ
ПРИЛОЖЕНИЯ

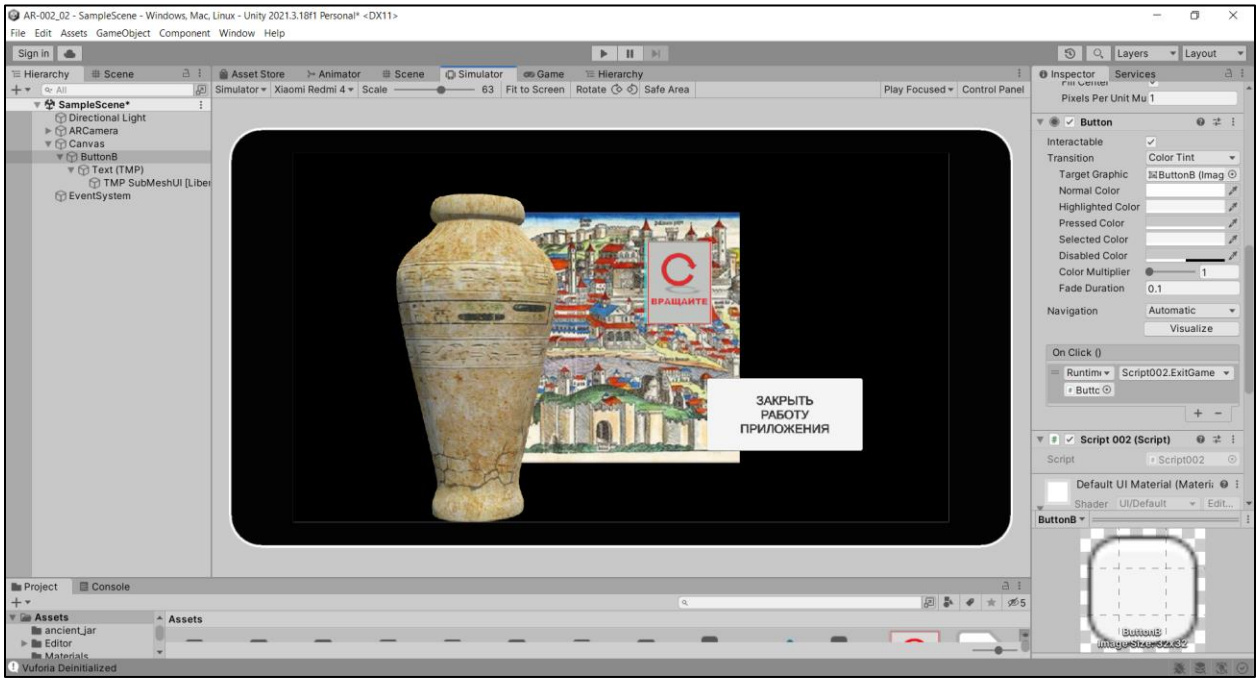

ЗАКРЫТЬ
РАБОТУ
ПРИЛОЖЕНИЯ

В результате в инспекторе для кнопки **ButtonB**:



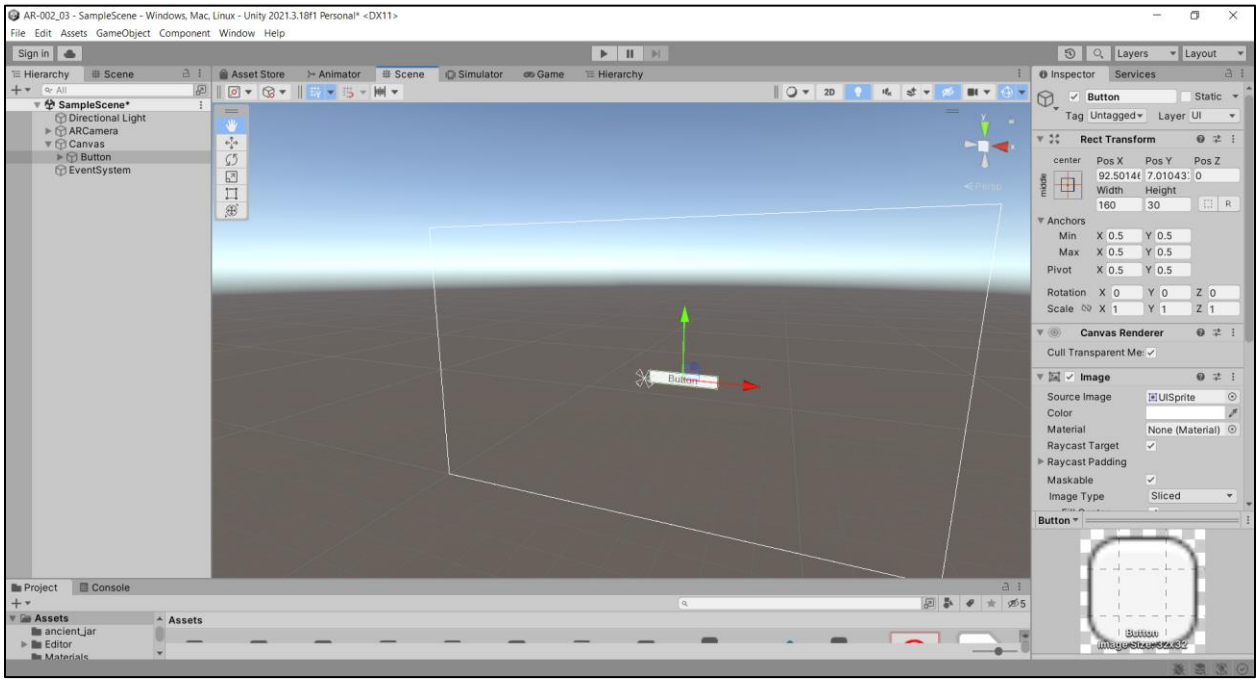
Работу кнопки можно проверить в режиме **Game (→ Play):**



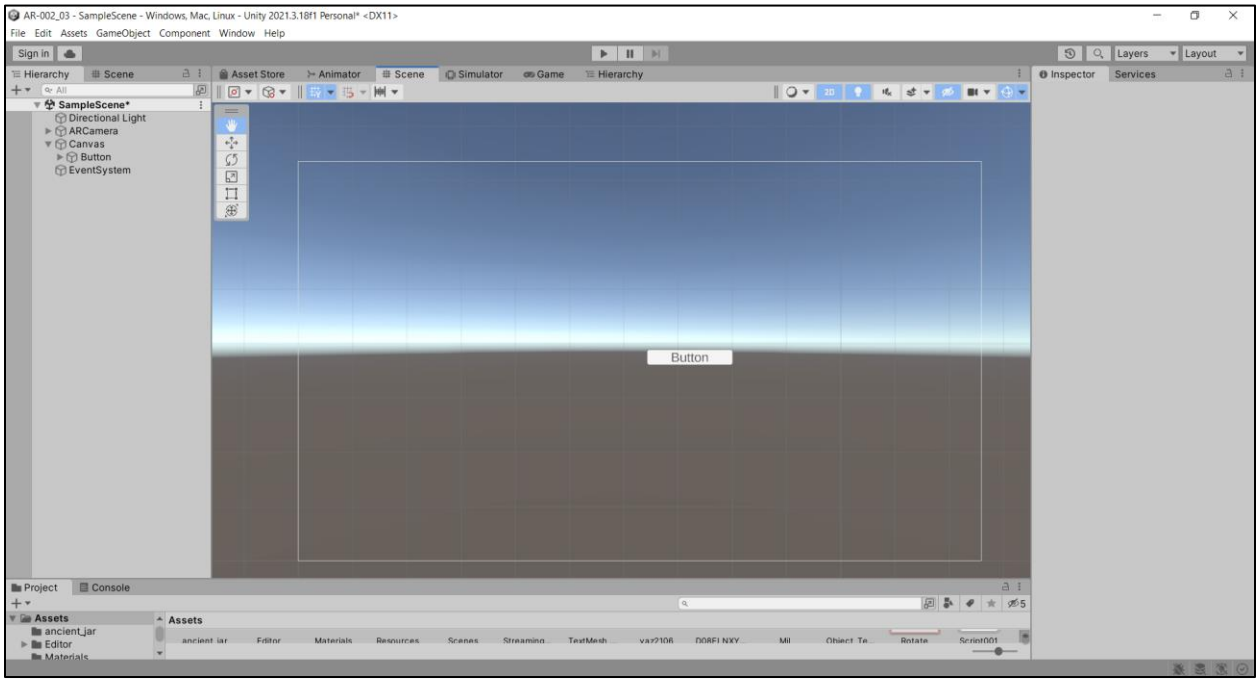

AR-002_02 - SampleScene - Windows, Mac, Linux - Unity 2021.3.18f1 Personal* <DX11>
File Edit Assets GameObject Component Window Help
Sign in
Layers
Layout
Hierarchy
Scene
SampleScene*
Directional Light
ARCamera
Canvas
ButtonB
Text (TMP)
TMP SubMeshUI [Liber
EventSystem
Asset Store
Animator
Scene
Simulator
Game
Hierarchy
Simulator
Xiaomi Redmi 4
Scale
63
Fit to Screen
Rotate
Safe Area
Play Focused
Control Panel
ВРАЩАЙТЕ
ЗАКРЫТЬ
РАБОТУ
ПРИЛОЖЕНИЯ
Inspector
Services
Pixels Per Unit Mu
1
Button
Interactable
Transition
Color Tint
Target Graphic
ButtonB (Imag
Normal Color
Highlighted Color
Pressed Color
Selected Color
Disabled Color
Color Multiplier
1
Fade Duration
0.1
Navigation
Automatic
Visualize
On Click ()
Runtim
Script002.ExitGame
Buttc
Script 002 (Script)
Script
Script002
Default UI Material (Materi
Shader
UI/Default
Edit...
ButtonB
Project
Console
Assets
ancient_jar
Editor
Materials
Assets
Vuforia Deinitialized

III. Кнопка для управления ранее созданной анимацией – вращение вазы

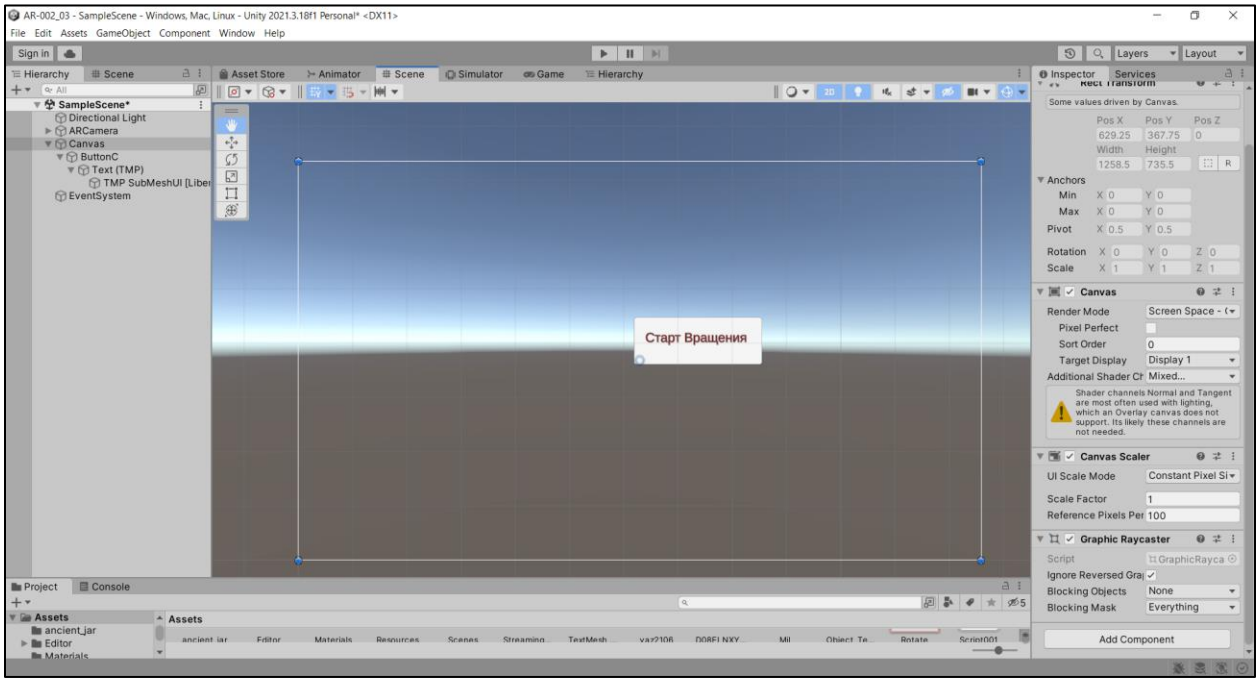
Все начальные шаги аналогичны двум предыдущим примерам.



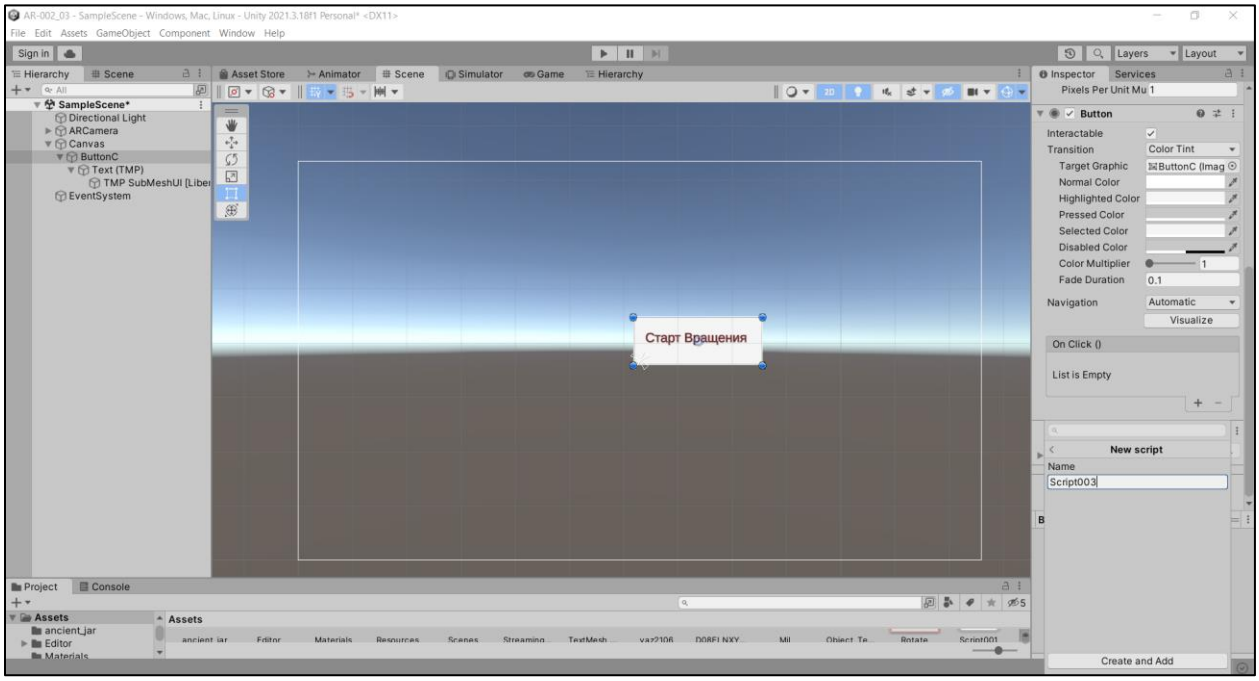
Переходим в 2D →

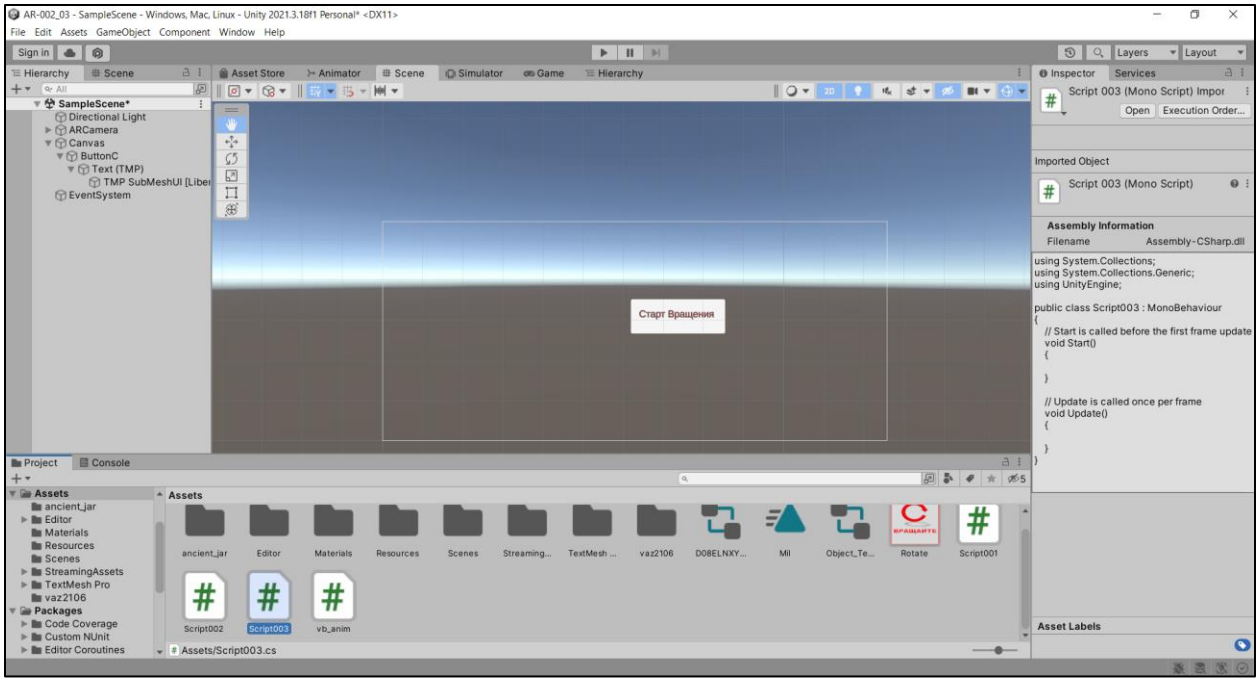


На кнопке задаем текст **«Старт Вращения»**, переименовываем ее в иерархии в **ButtonC**:



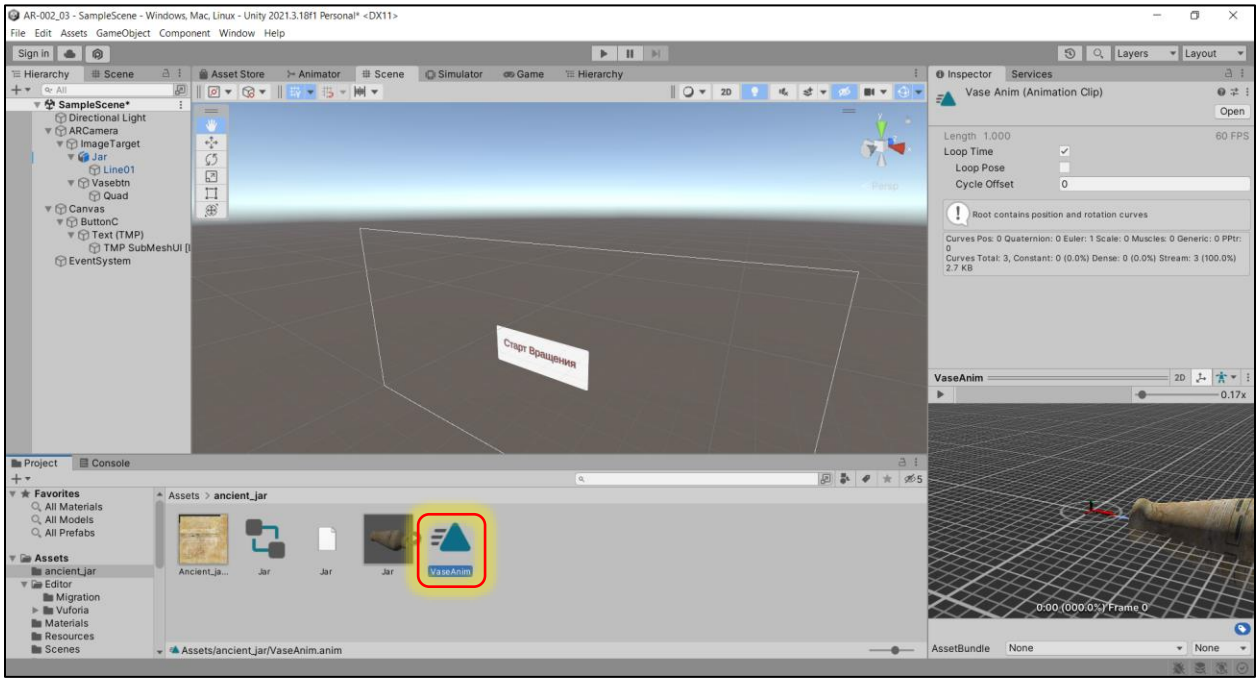
Добавим новый скрипт – **Script003 →**



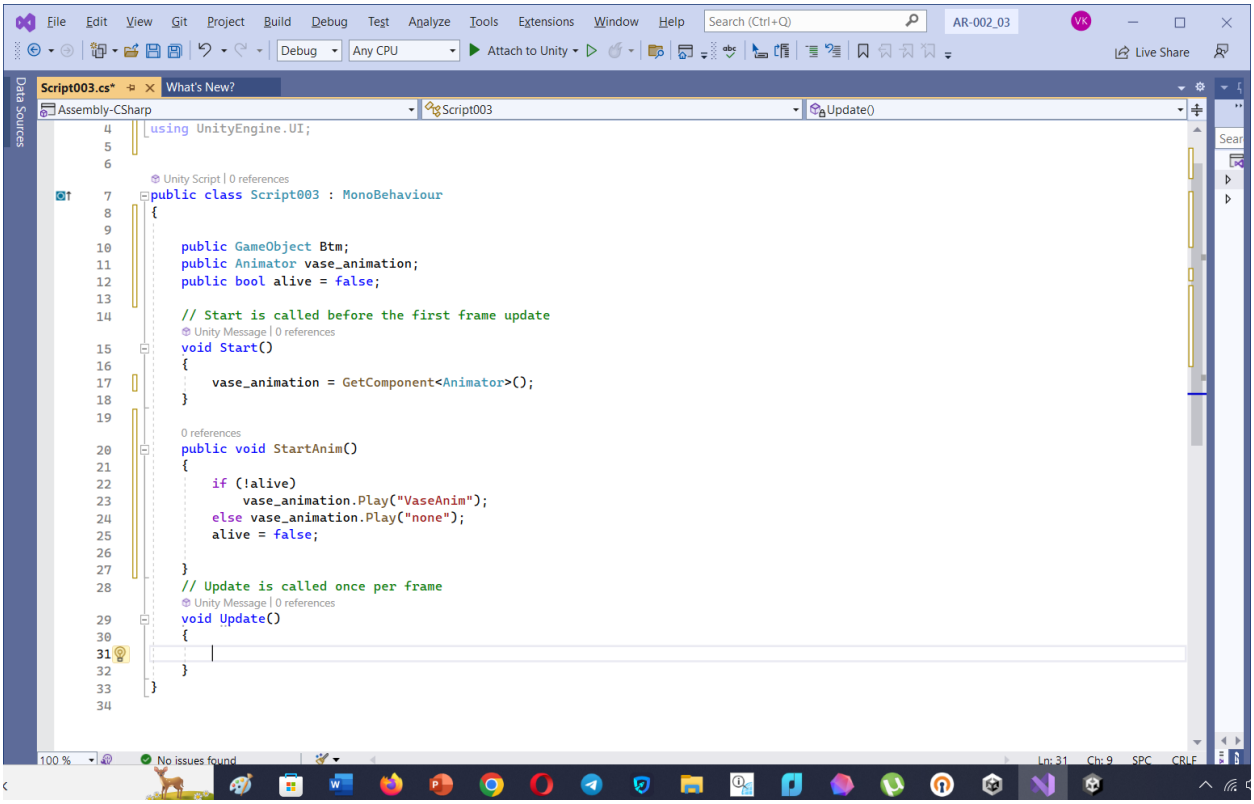


Модифицируем шаблон для вызова уже готовой анимации →

Где найти анимацию (как ассет)? Она была создана ранее в рамках ЛРN№3:

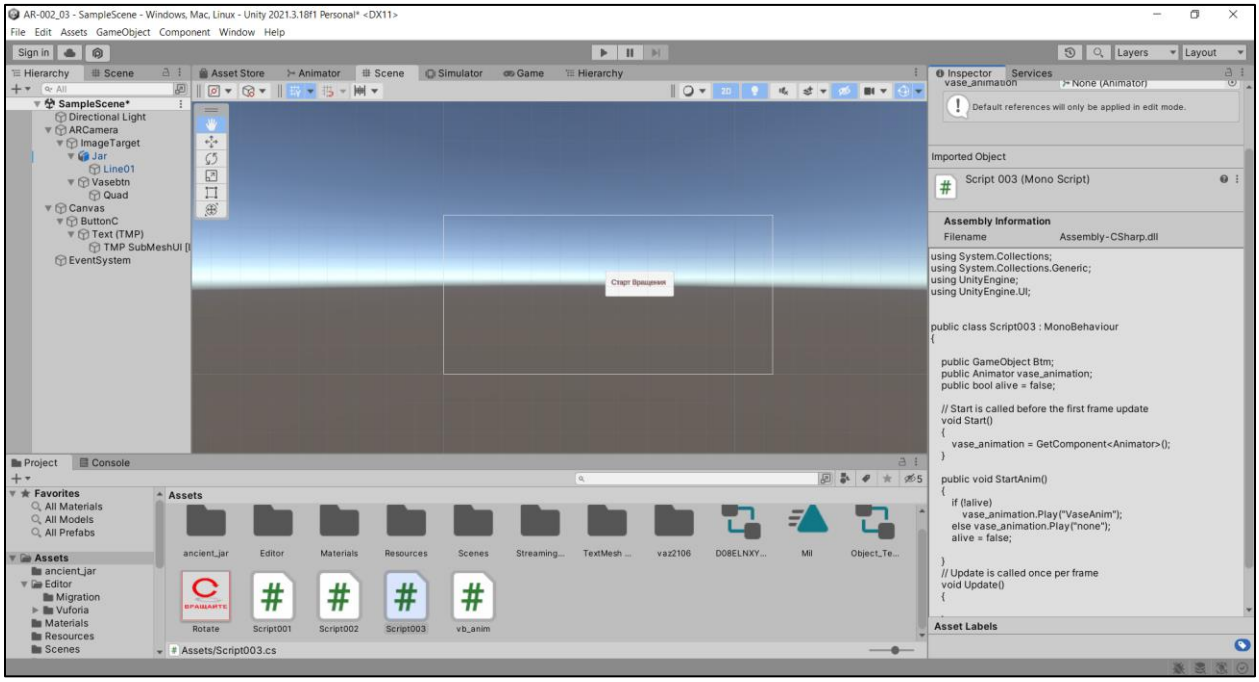


Свяжем вызов этой анимации с кнопкой через скрипт.

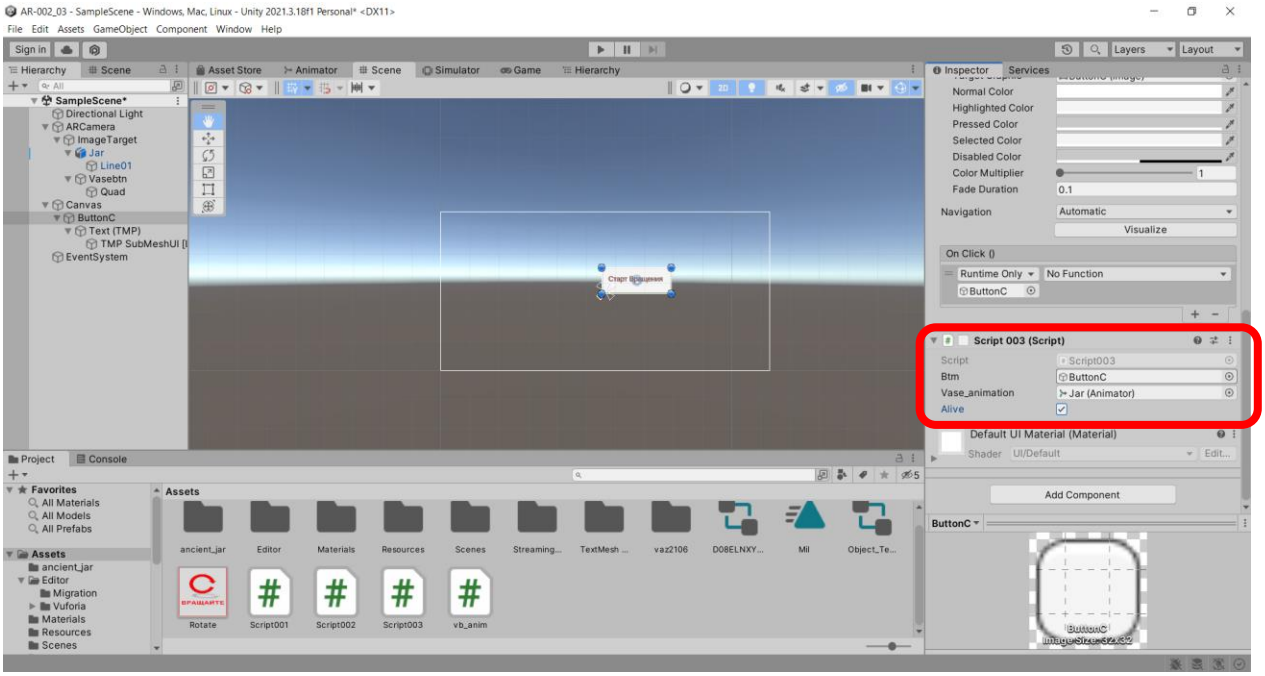


```
using UnityEngine.UI;


Unity Script | 0 references
public class Script003 : MonoBehaviour
{

    public GameObject Btm;
    public Animator vase_animation;
    public bool alive = false;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        vase_animation = GetComponent<Animator>();
    }

    0 references
    public void StartAnim()
    {
        if (!alive)
            vase_animation.Play("VaseAnim");
        else vase_animation.Play("none");
        alive = false;

    }
    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {

    }
}
```

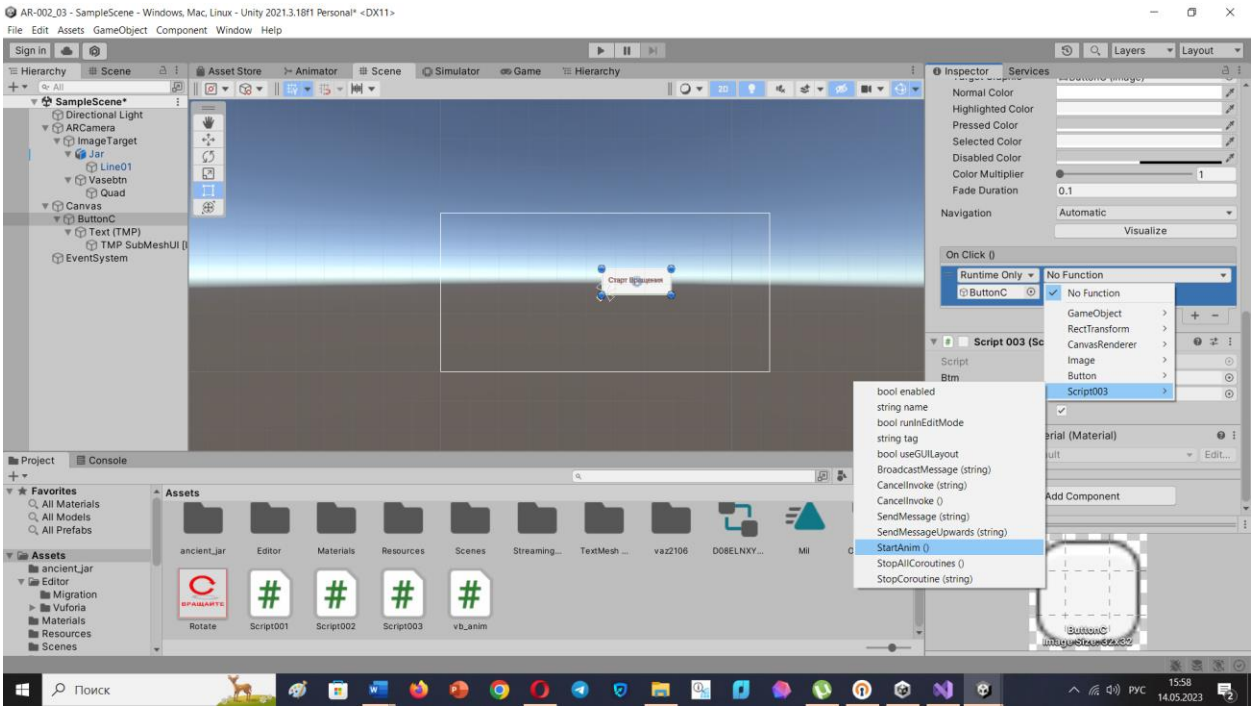
Скрипт – в ассетах:



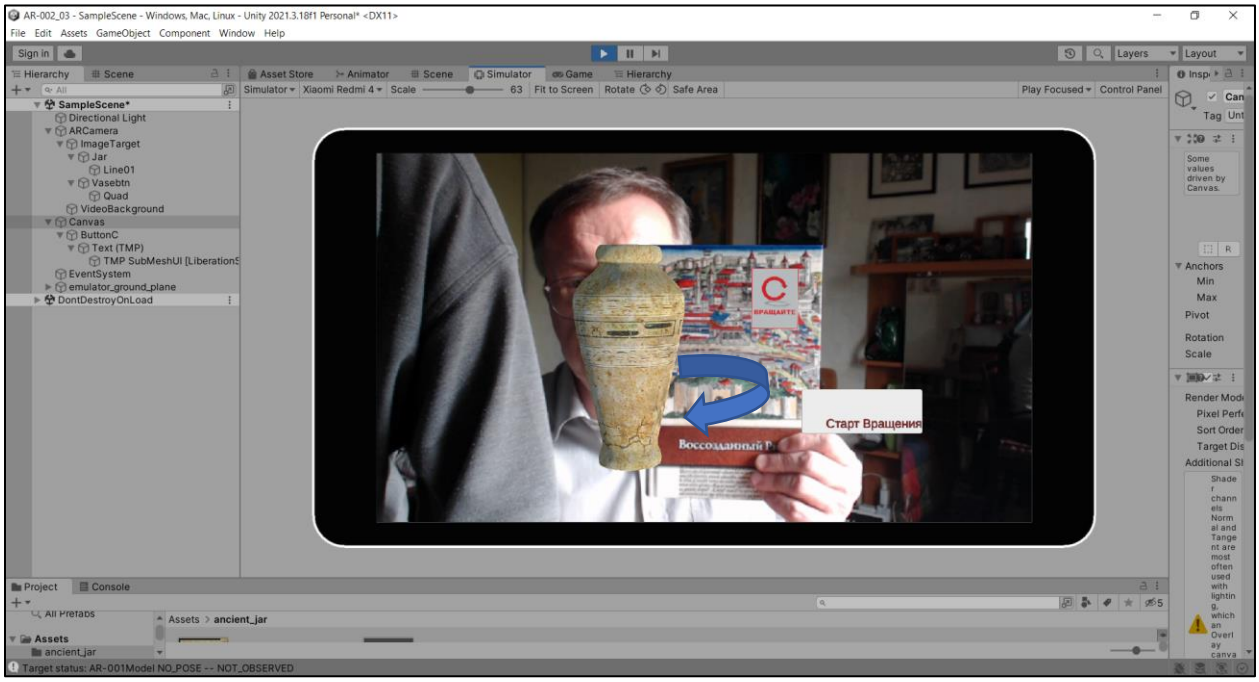
Теперь привязываем кнопку к этому скрипту →



и к функционалу (в **«On Click»**) этого скрипта →



Работу кнопки можно проверить в режиме **Game (→ Play):**



Таким образом были разработаны и добавлены на стекло Приложения ДР три управляющие кнопки для управления:

I. **перемещением объекта контента (трехмерная модель) по одной из осей в положительном (или отрицательном ее направлении),**
II. **выходом из Приложения,**
III. **ранее разработанной анимацией (анимациями) этой же модели.**

Разработка велась в среде Unity 2021.3.18 и Visual Studio 2022