Лабораторная работа № 3. Проект – разработка простого AR-Приложения для Android-устройства (смартфон, планшет и пр.). Создание в графическом редакторе Unity 3D сцены дополненной реальности: Часть 3. Добавление в плоское меню, разработанное в Части 2. элементов управления (кнопок) для перемещения 3D-Модели по одной из осей и выхода из Приложения ДР.

Предметом исследования **во второй части ЛР №3** стала разработка элемента управления плоского меню Приложения ДР (кнопка **Button**) для старта/остановки разработанной в **первой части ЛР№3** анимации **3D модели**.

В данной, **третьей части ЛР№3**, предлагается дополнить разработанное в **ЛР№3, Часть 2** плоское меню Приложения ДР элементами управления **(кнопками – Button)** для визуализации перемещения **3D модели** по одной из осей, а также создать кнопку для выхода из Приложения.

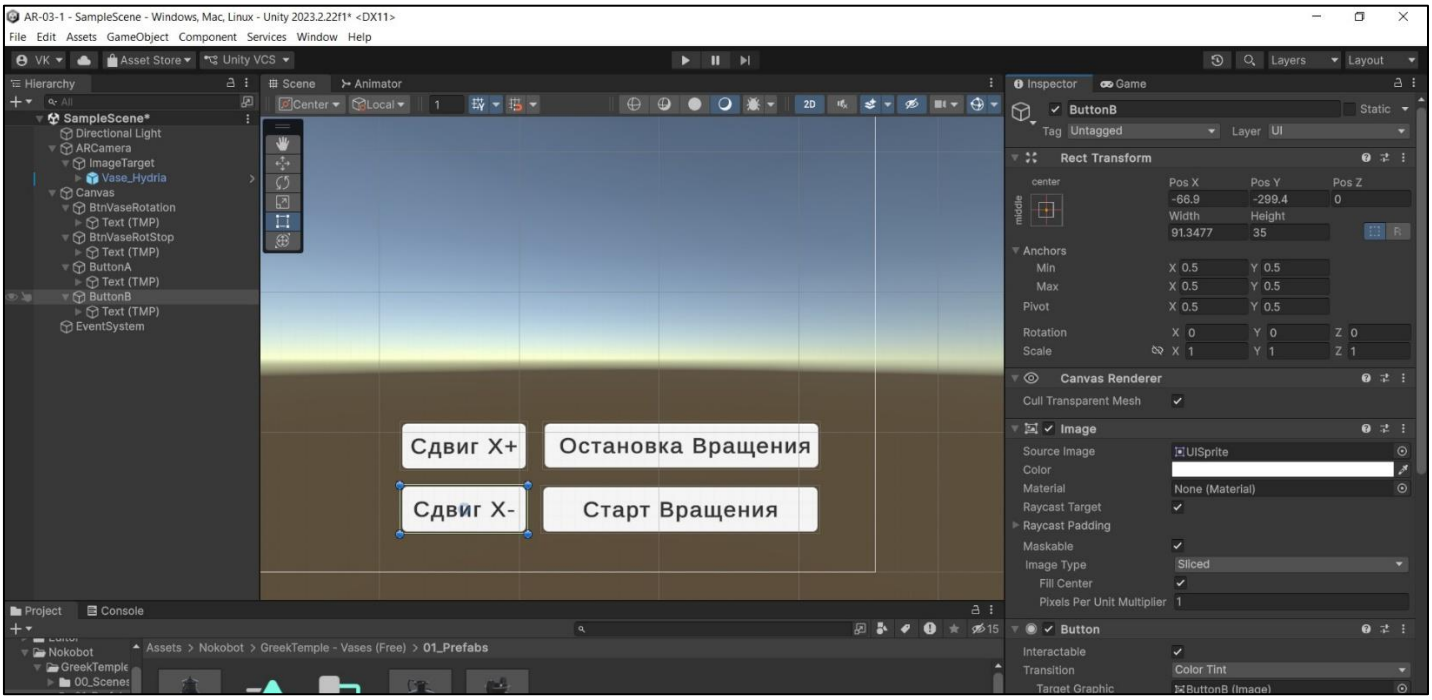
Итак, основной задачей является установление связей между поведением (**behaviour**) виртуальной **3D-модели** (у нас - **«перемещение греческой вазы по одной из осей»**) и состоянием элемента управления **«кнопка»**, а также **завершения/выхода из Приложения** с помощью управляющего элемента, входящегов состав плоского меню.

В данной, 3 части ЛР№3, для установления указанных связей в Unity 3D предлагается использовать скриптинг (программирование на языке **C#)**.

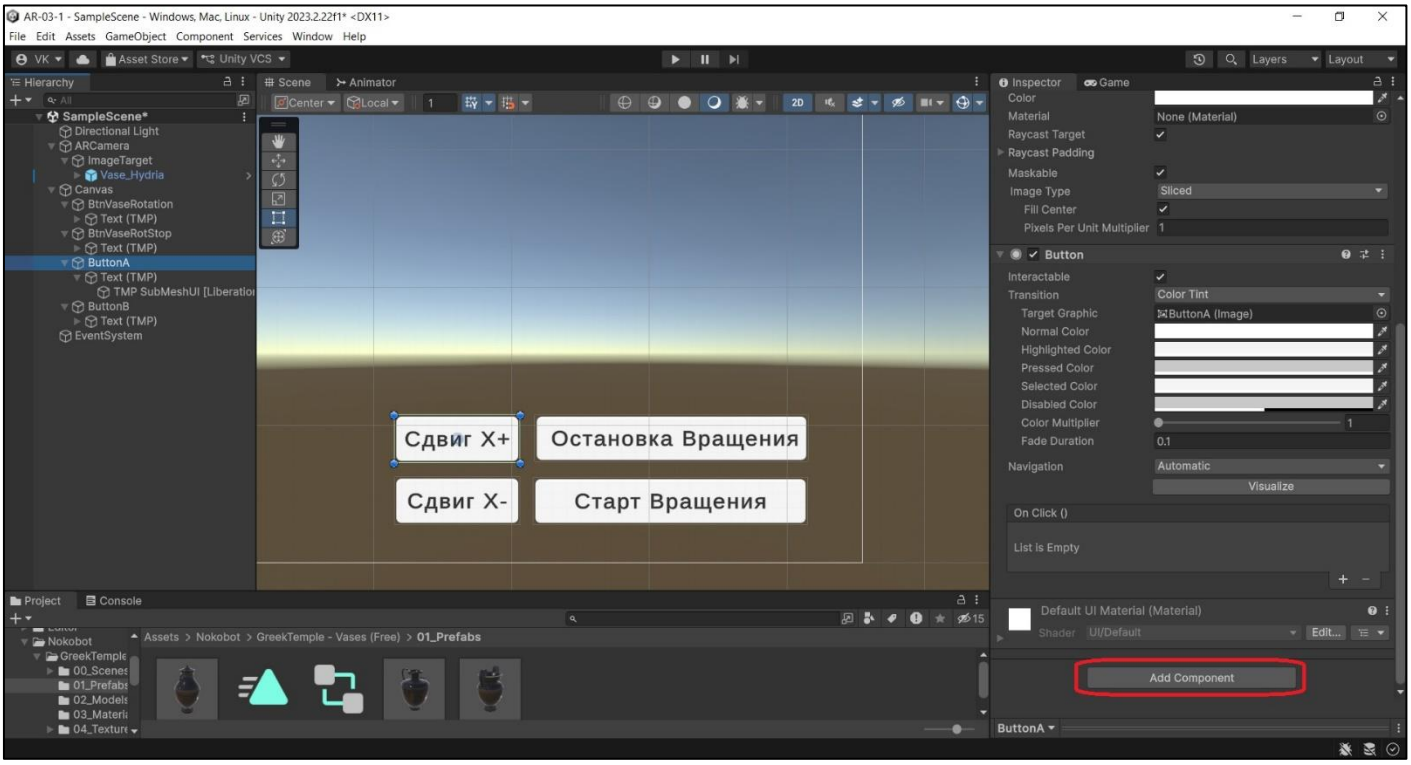
Разработанные элементы управления (кнопки) должны быть добавлены в плоское меню, разработанное в **ЛР№3, Часть 2**.

1. Создание элемента управления плоского меню - Кнопки для управления перемещением объекта контента (трехмерная модель греческой вазы - Vase_Hydria в Иерархии) по одной из осей (ось X) в положительном и отрицательном ее направлении.

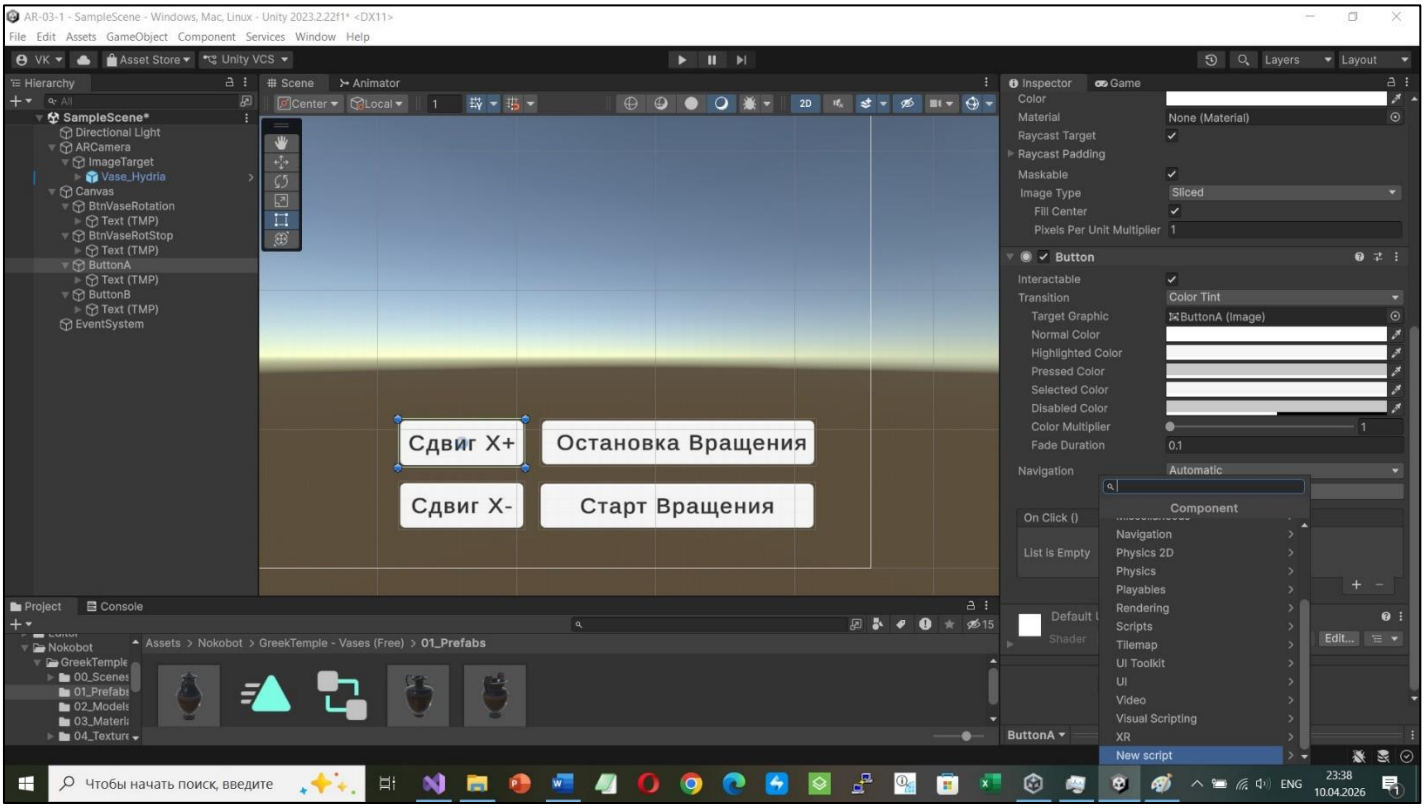
Входим в проект **ЛР№3, Части 1,2** и добавляем в разработанное плоское меню две кнопки **ButtonA - «Сдвиг X+»** и **ButtonB - «Сдвиг X-»**. Обращайте внимание на корректность расположения объектов в **Иерархии:**



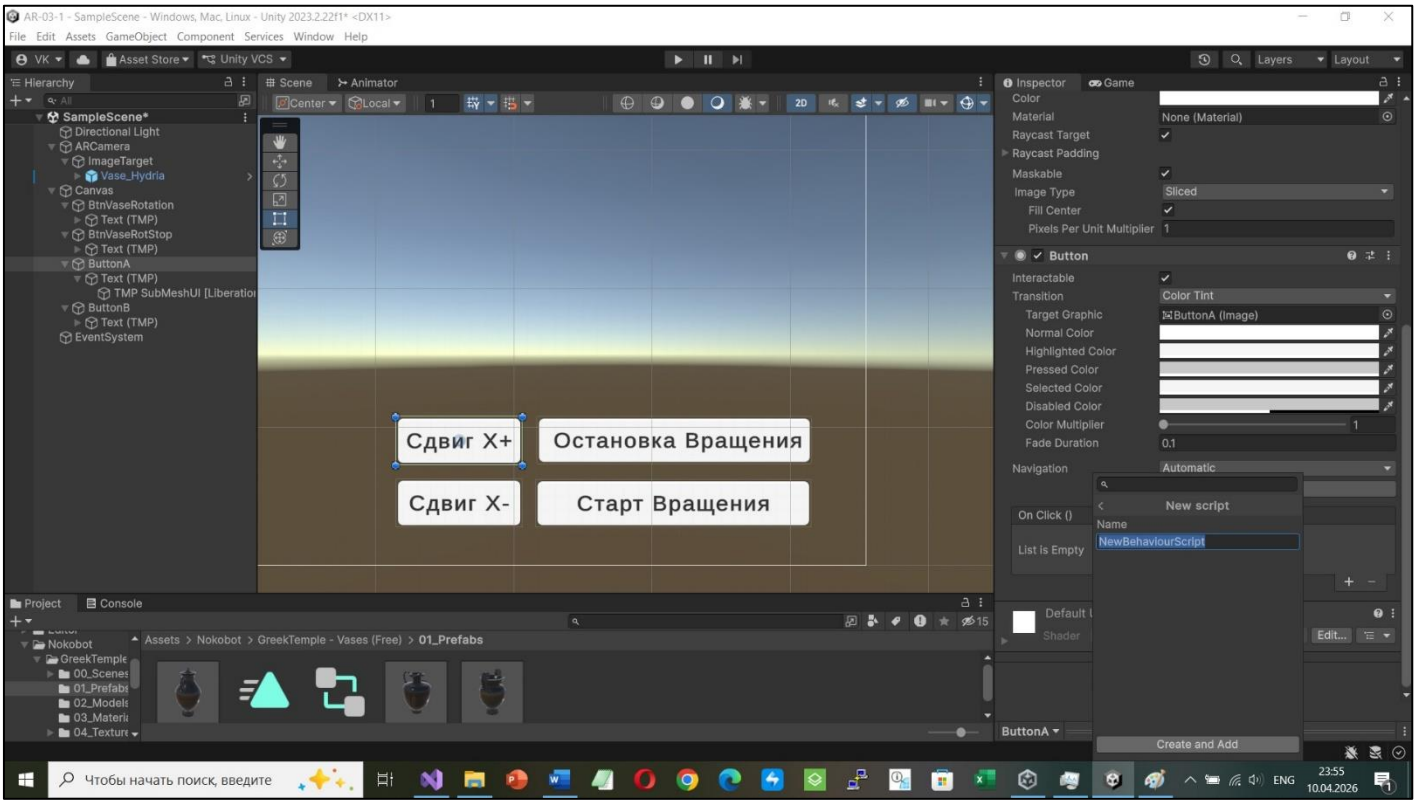
Рассмотрим связывание кнопки **ButtonA (Сдвиг X+)** с перемещением объекта **Vase_Hydria** с помощью скрипта. В иерархии выбираем кнопку **ButtonA** и в инспекторе находим позицию **Add Component:**



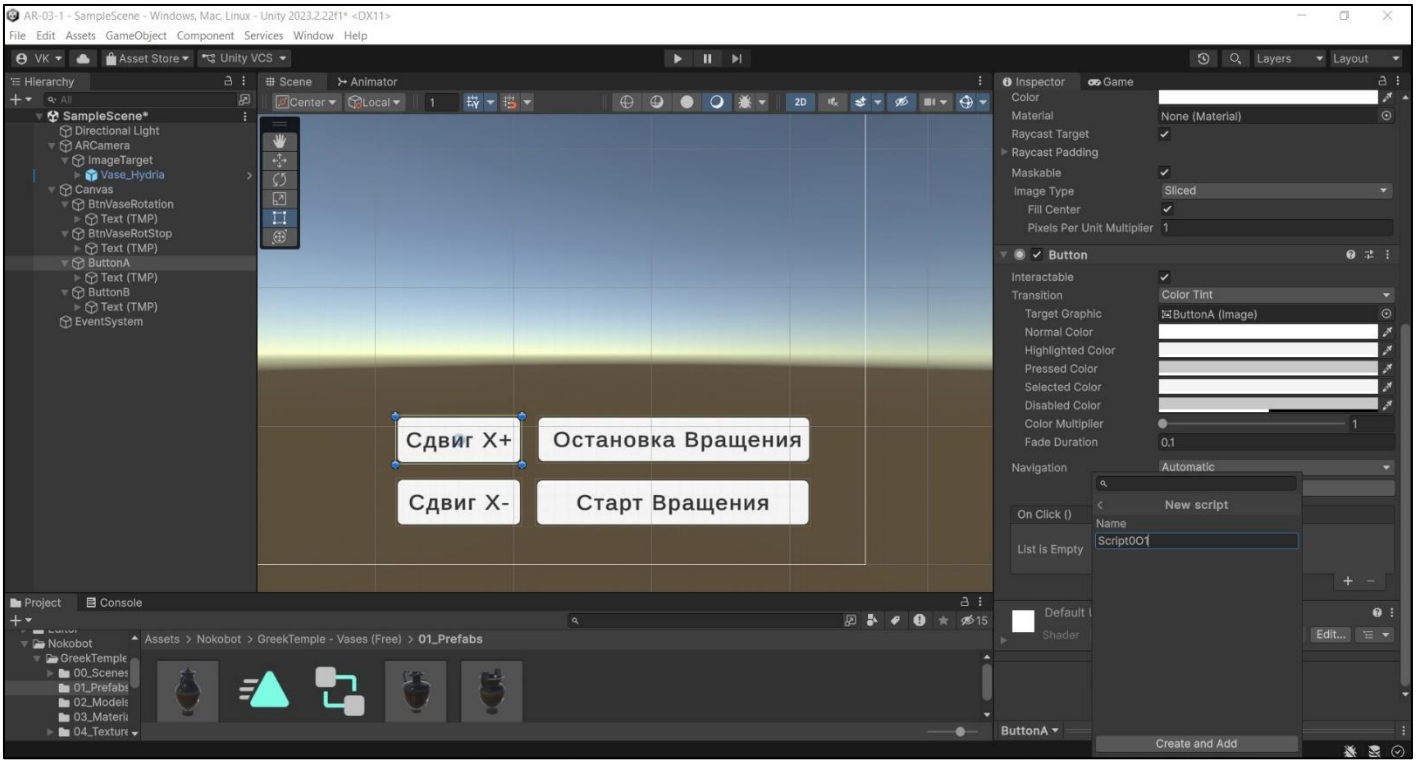
В появляющемся списке возможных компонентов для связывания с кнопкой выбираем **New script**, если он еще не создан (в нашем случае), или **Script**, если он уже находится в **Assets:**



Переходим по **New Script** в поле имени скрипта и прямо поверх **«NewBehaviourScript»** вводим пользовательское имя скрипта, например **Script001**:



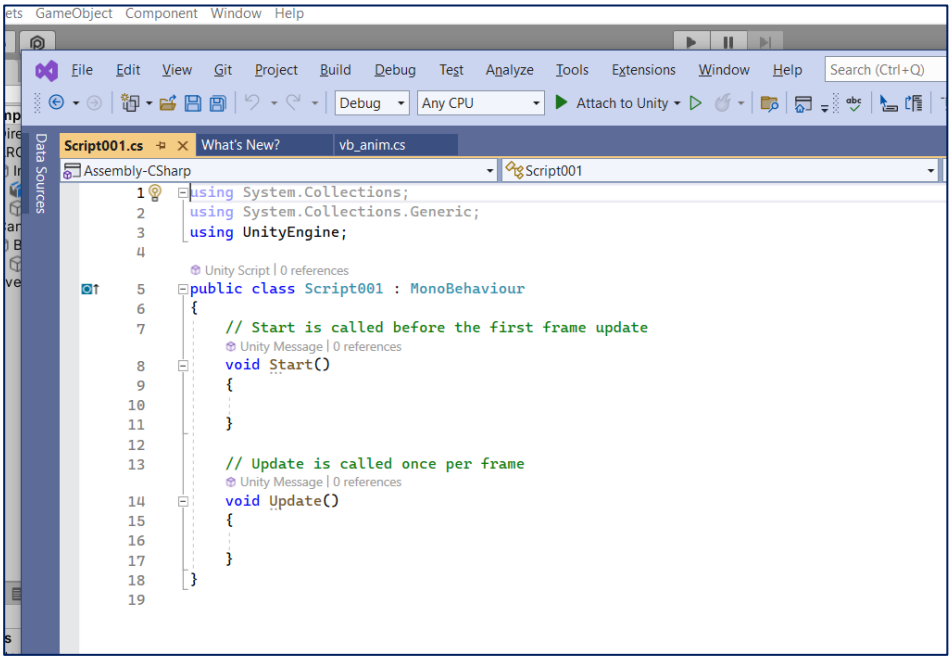
Результат:



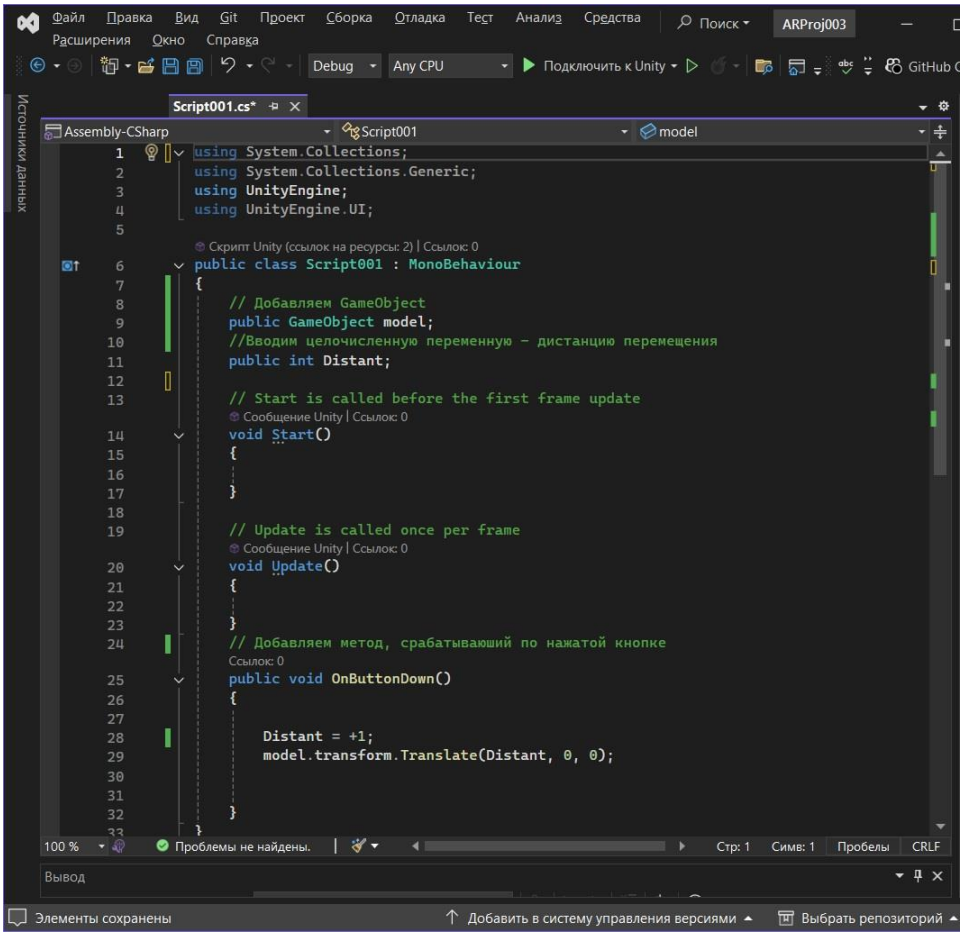
При выборе опции **Create and Add** открывается окно **Visual Studio** с шаблоном скрипта, на базе которого мы будем создавать скрипт для перемещения объекта **Vase_Hydria** в

положительном направлении по оси **X.** При этом в области **Assets** появляется иконка шаблона скрипта, а в **инспекторе** на кнопку **(Button)** появляются поля работы со скриптом.

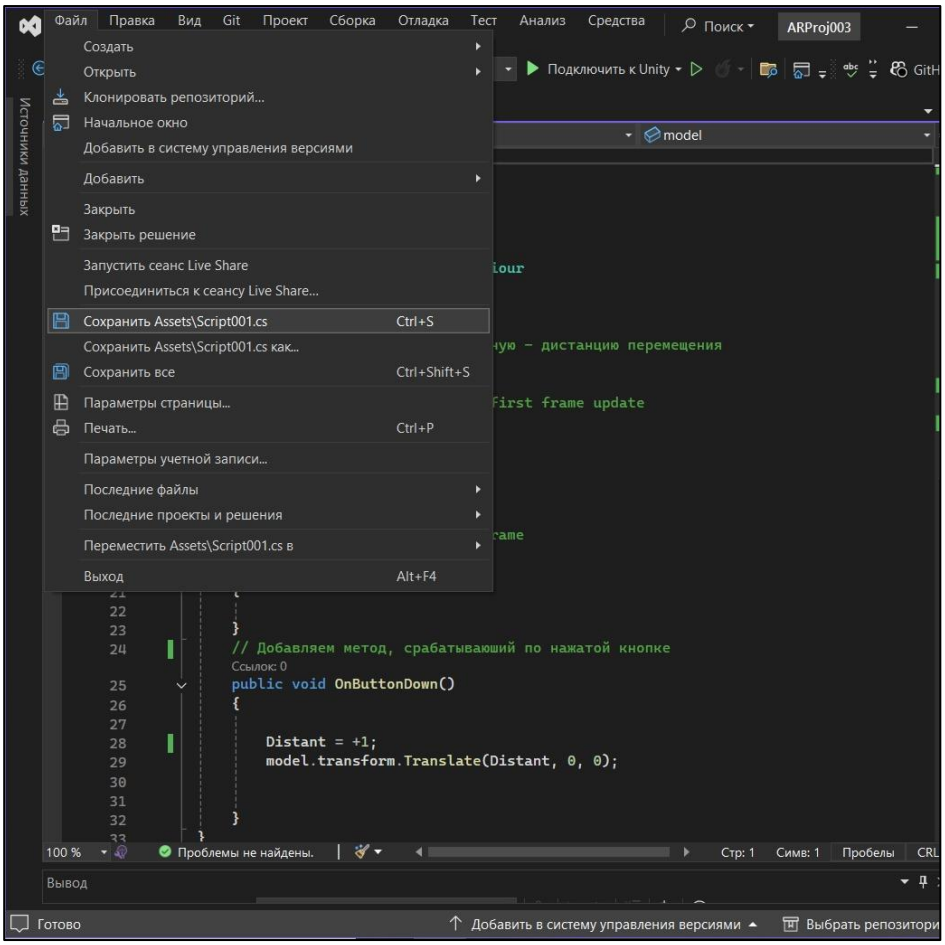
Двойной клик по иконке уже созданного или редактируемого скрипта в **области Assets** также открывает **Visual Studio.** В случае создания нового скрипта, шаблон всегда выглядит т.о.



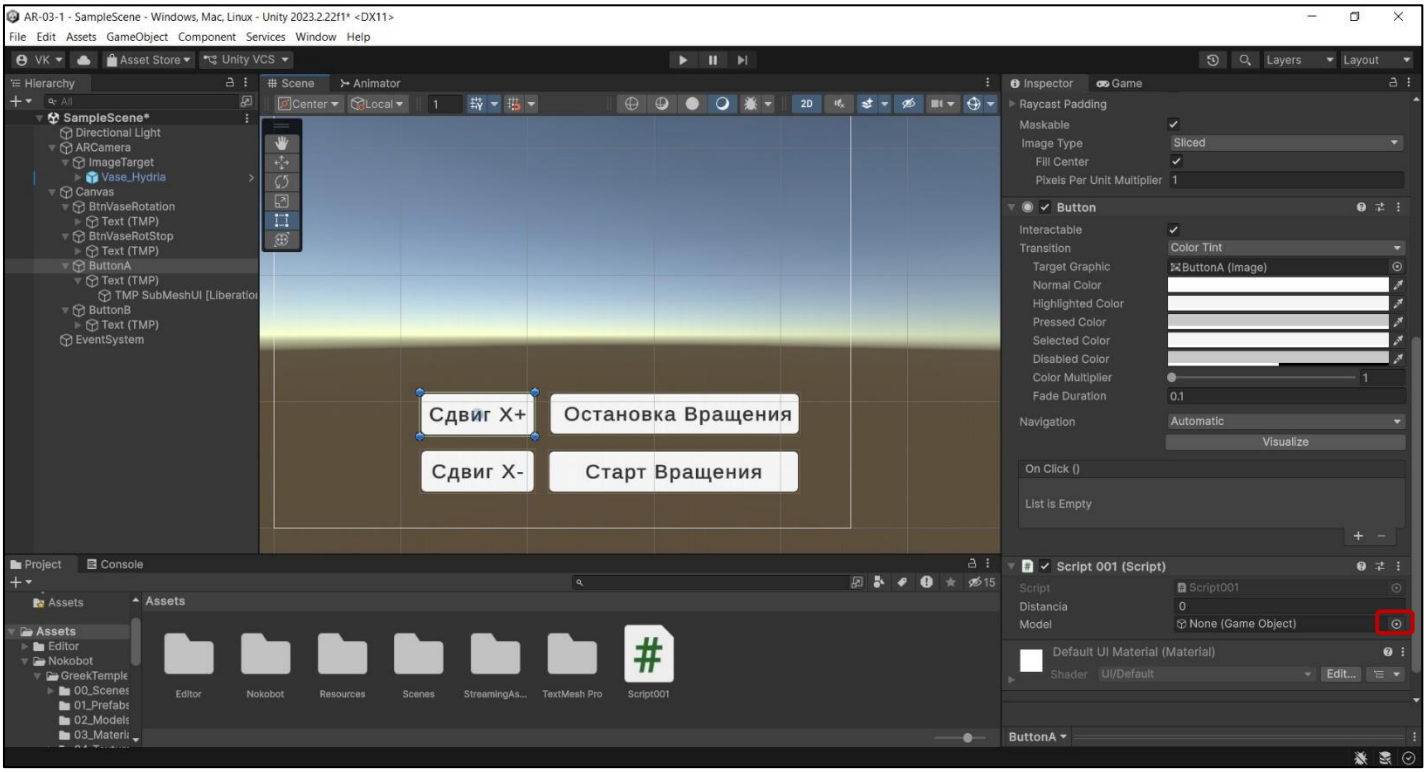
Результат редактирования шаблона скрипта для перемещения **Сдвиг X+** (см. русскоязычные комментарии в теле скрипта):



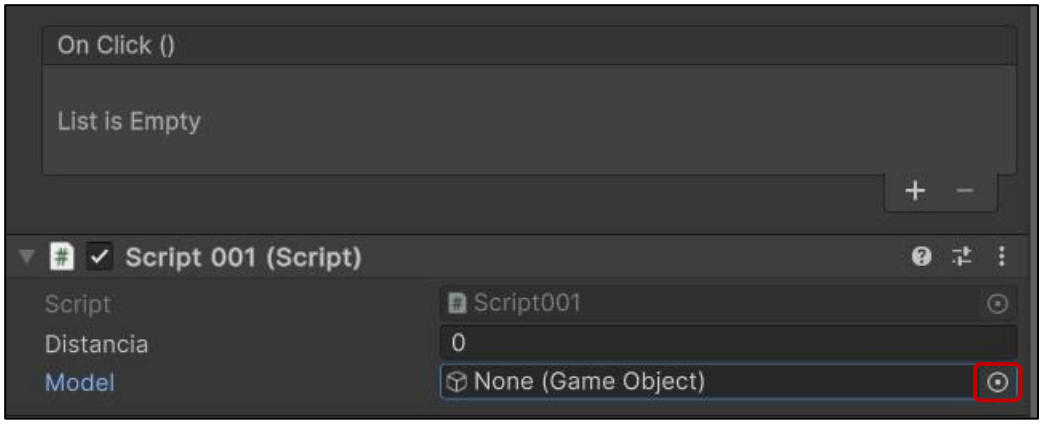
Сохраняем скрипт в Assets:



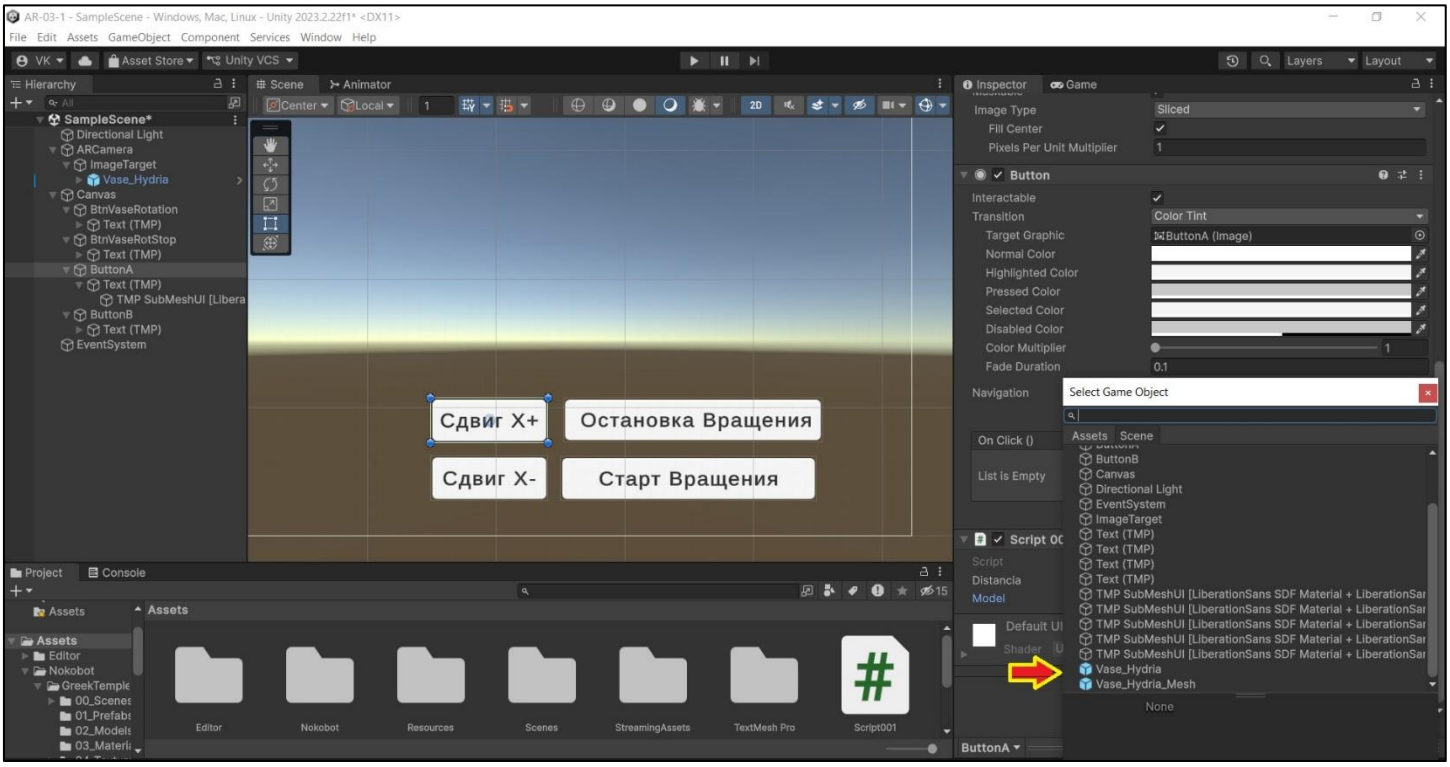
Результат:



Теперь объект (Game Object) model из скрипта надо привязать к нашей **3D - модели** в сцене. Сделаем это через интерфейс **инспектора** по **Button** (если ранее все было сделано правильно): **Позиция Script001 (Script) → Model →выбираем из списка доступных в проекте моделей нашу вазу → Vase_Hydria:**

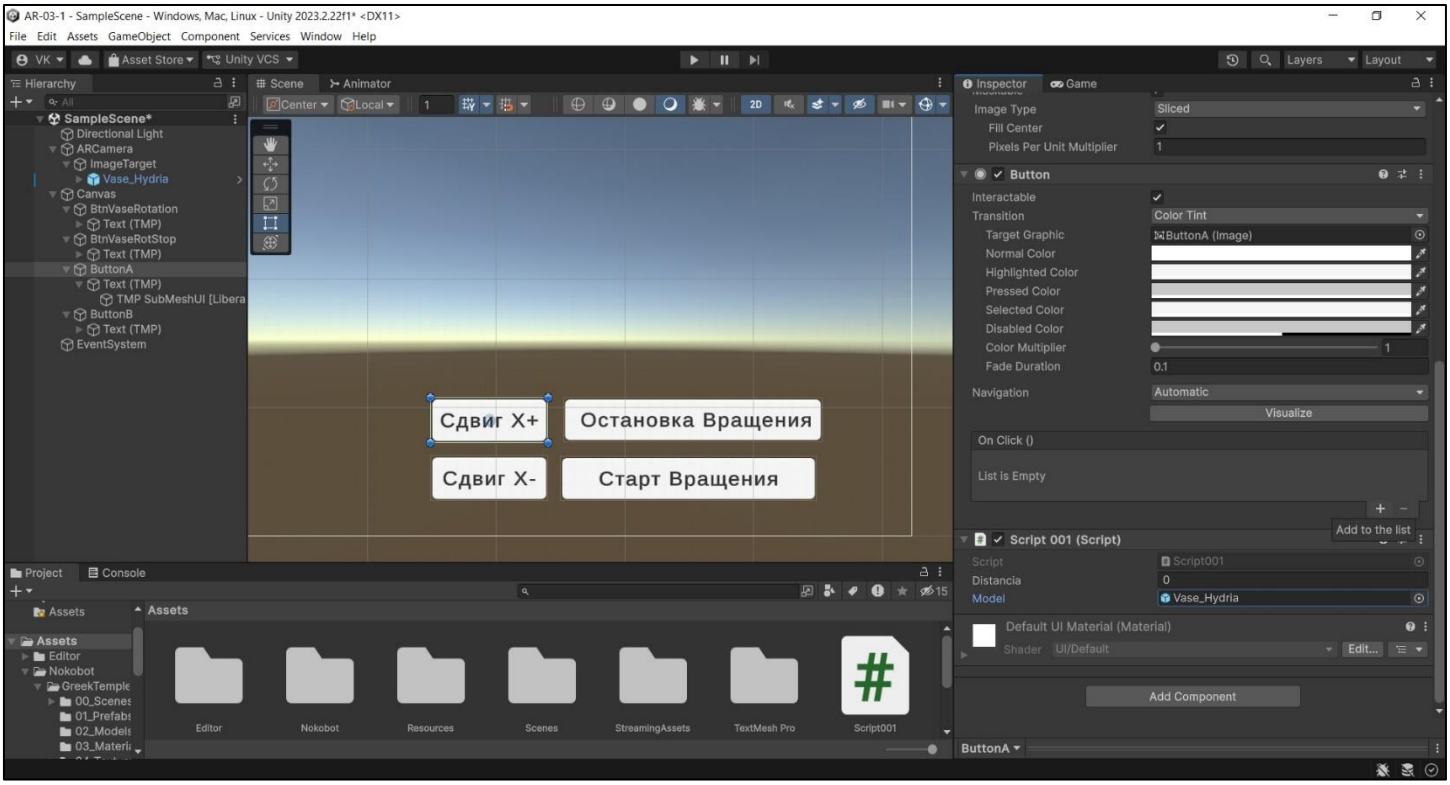


В открывшемся списке ищем и выбираем наш объект – **Vase_Hydria:**



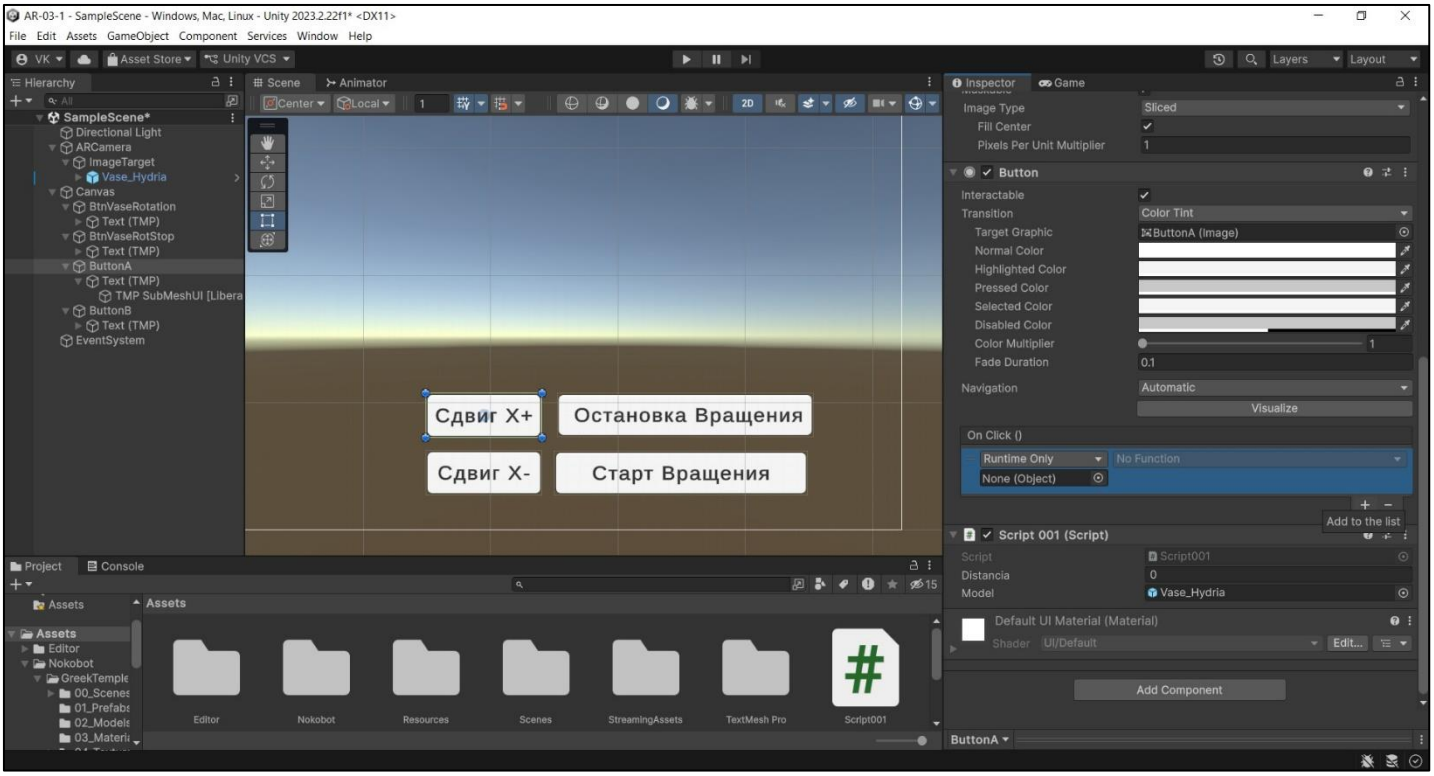
ВАЖНО!! Обратите внимание – мы опять работаем с позицией On Click (), но в данном случае – со скриптом: настраиваем/связываем объект, выбираем режим и функционал!!

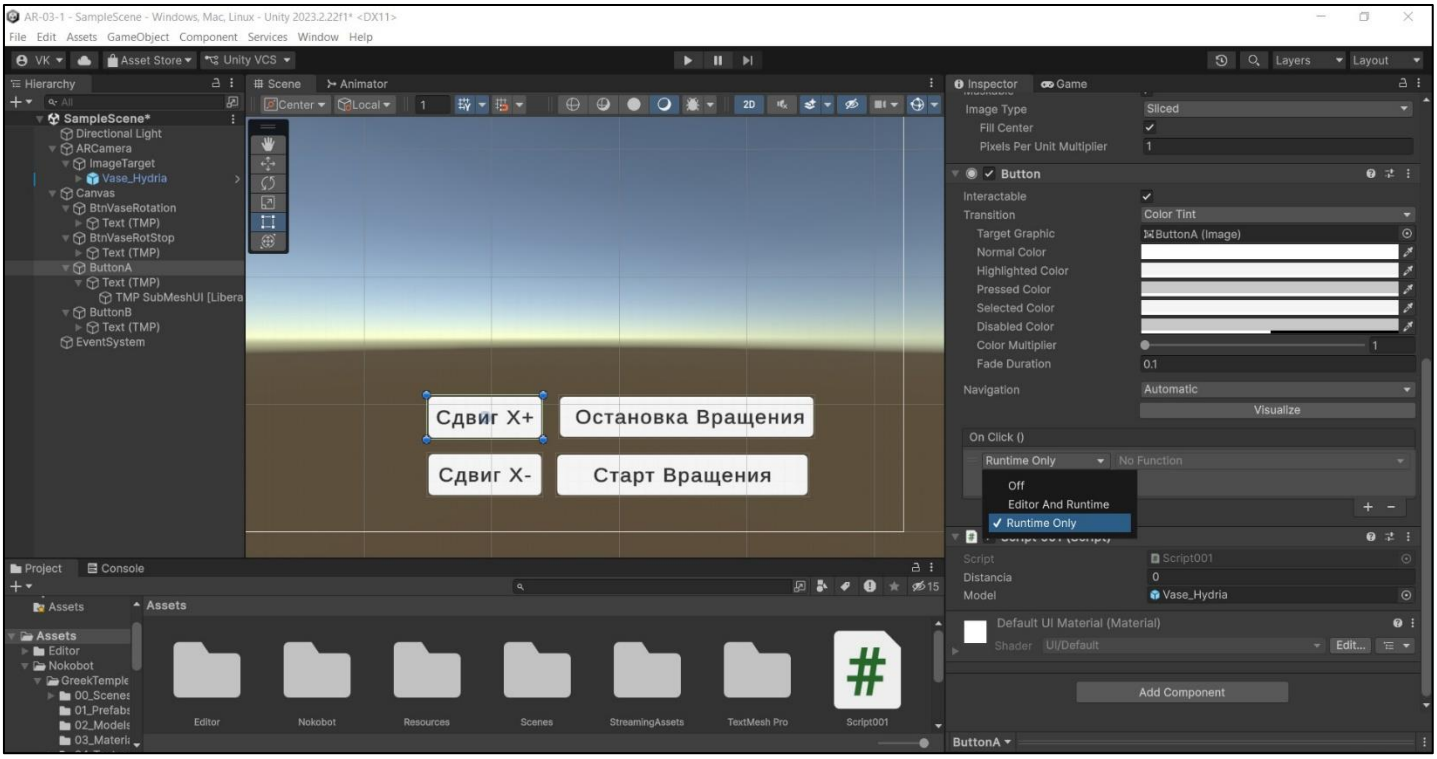
Последовательность шагов:



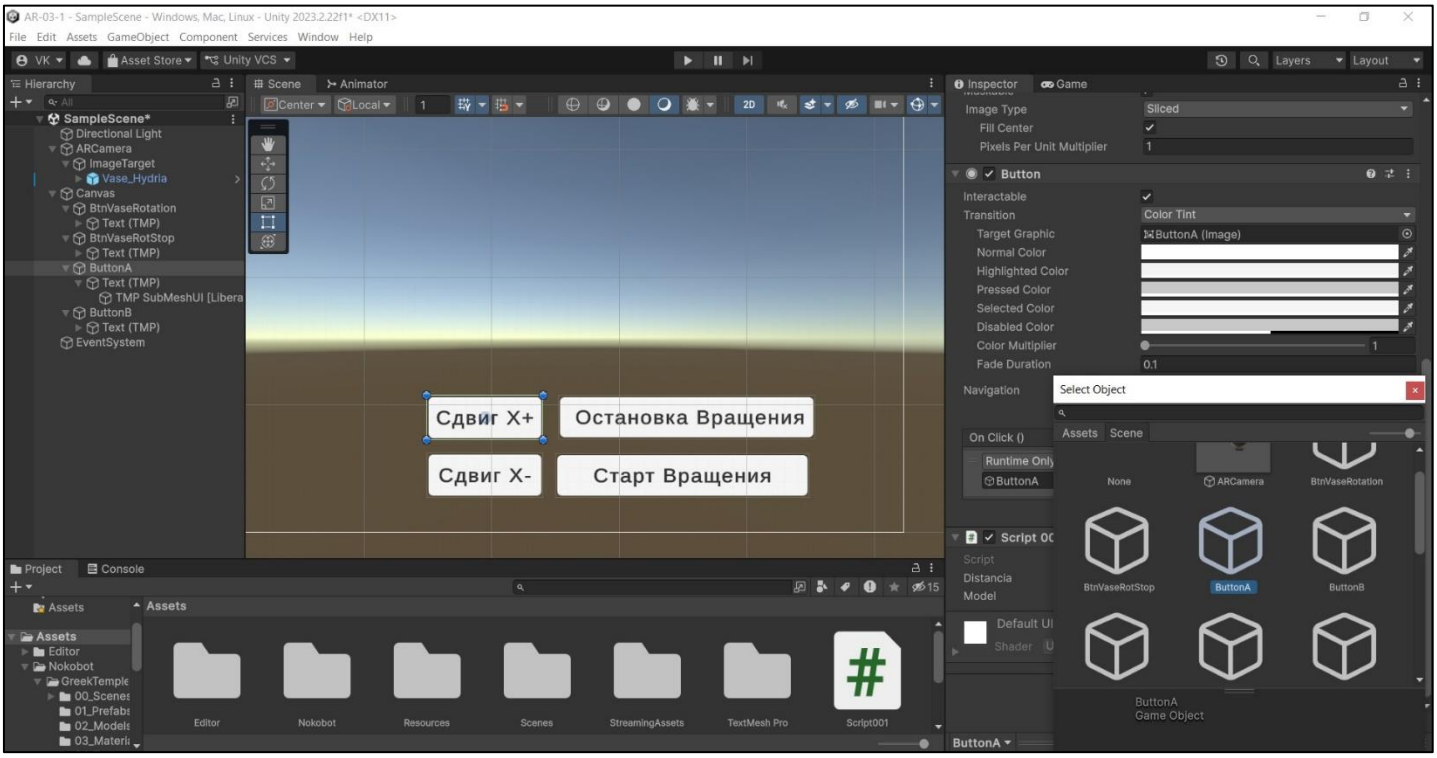
В инспекторе для **ButtonA** находим свойство **On Click ()** и настраиваем его на действие, которое прописали в скрипте **Script001**. Для этого используем кнопку **«+»** поля **On Click ()**

Приписываем свойству **On Click** режим работы **Runtime Only →**



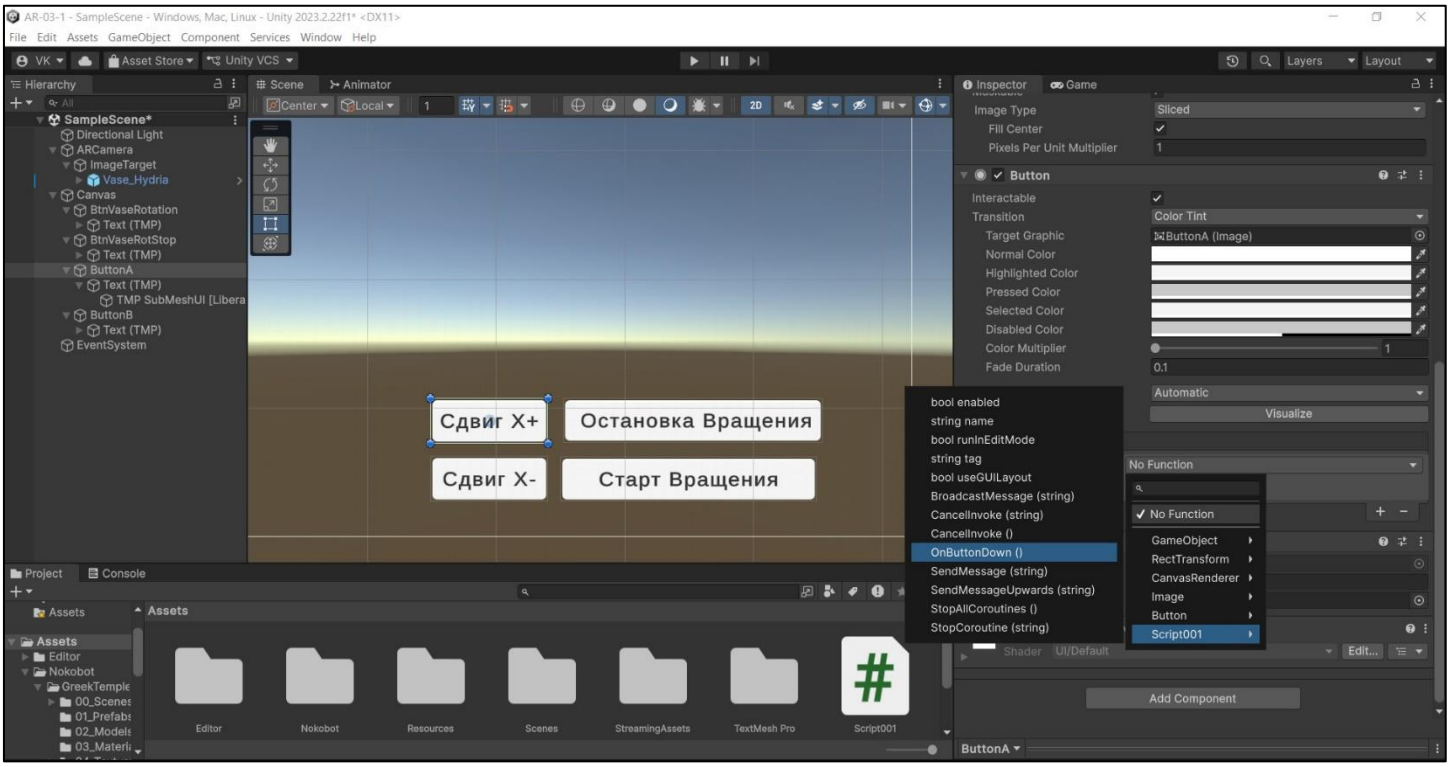


Рядом в поле ввода объекта **(None (Object))** вызываем доступный список объектов и ищем нашу кнопку **ButtonA** →

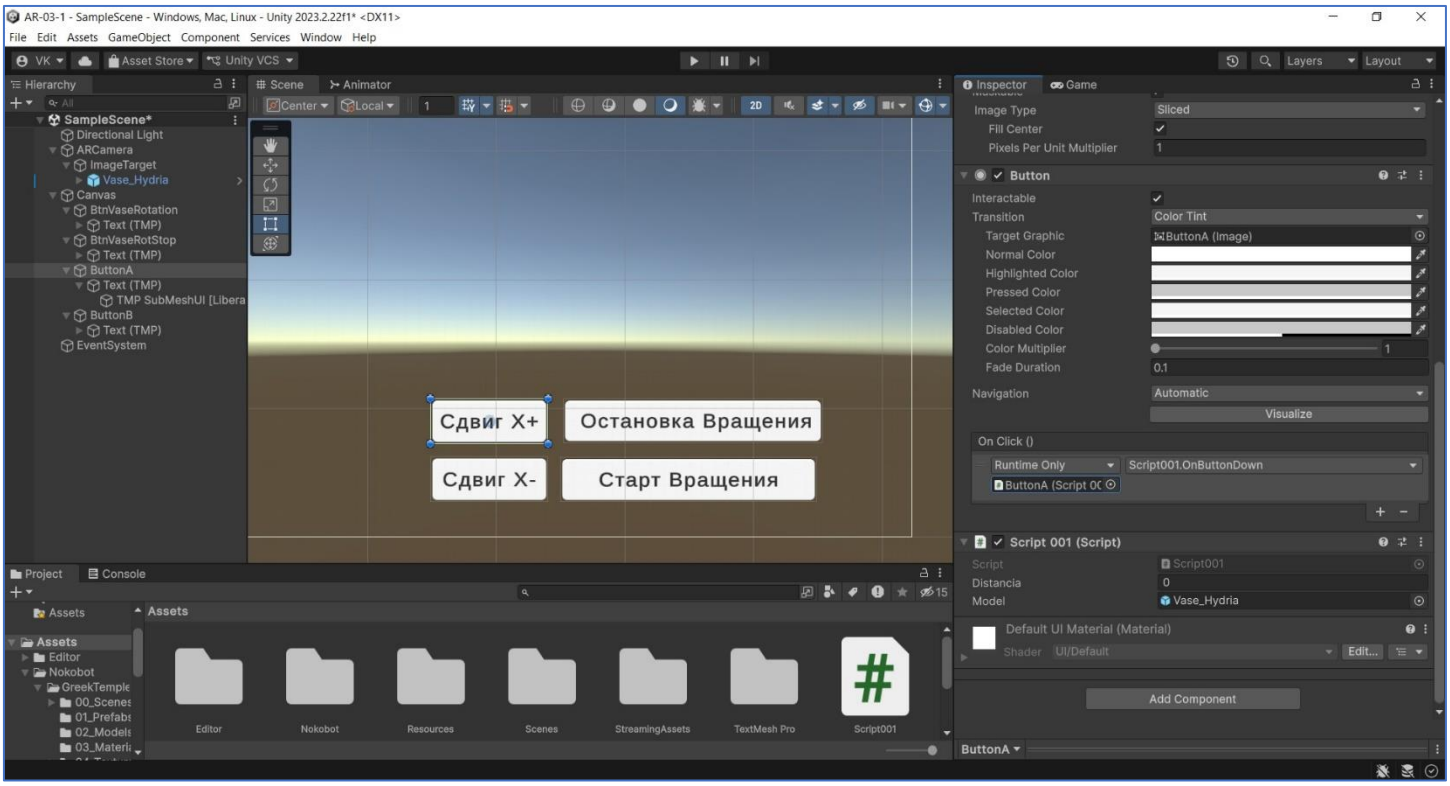


Обратите внимание – ищем в с цене!!!

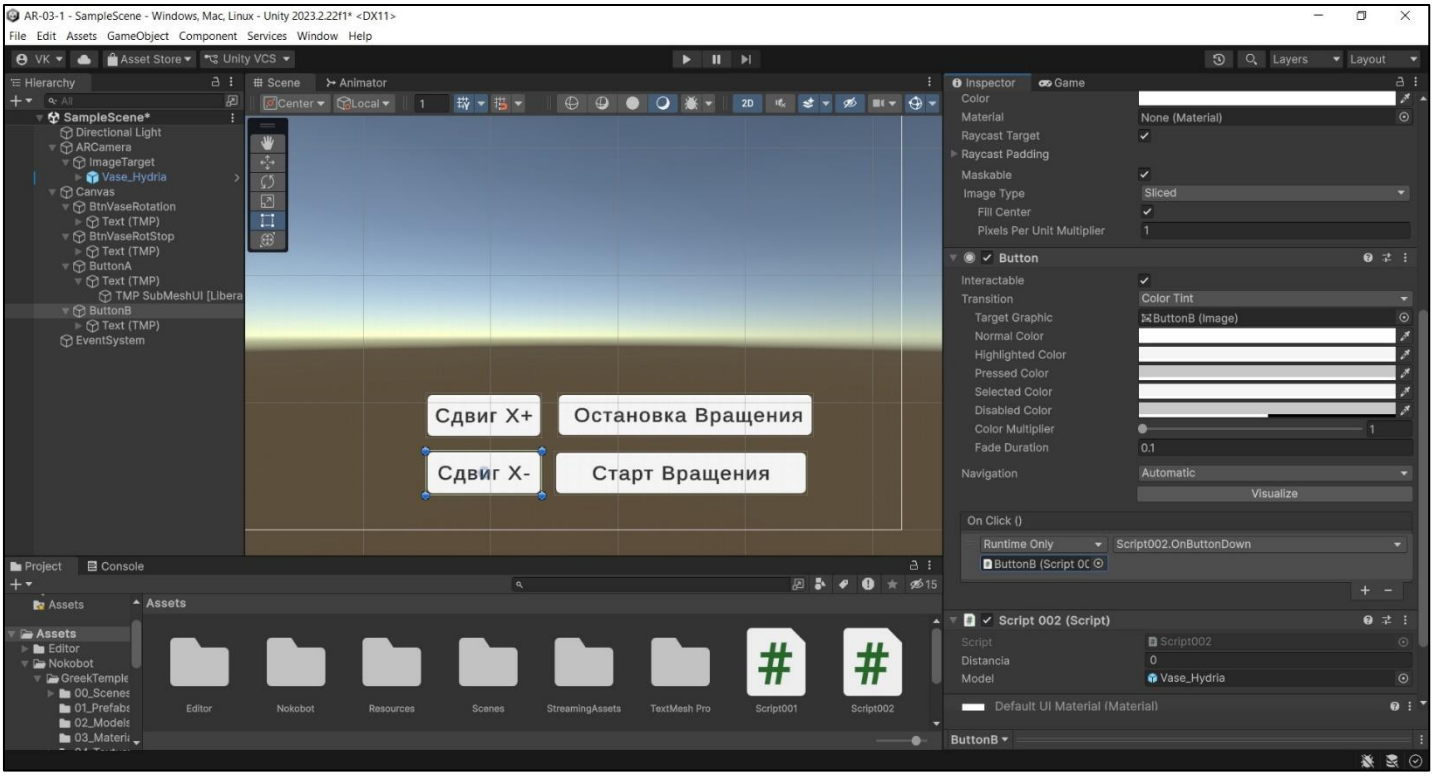
И выбираем метод (**в поле No Function**), который мы добавили в шаблон скрипта **Script001** для собственно действия по перемещению → **OnButtonDown ()**



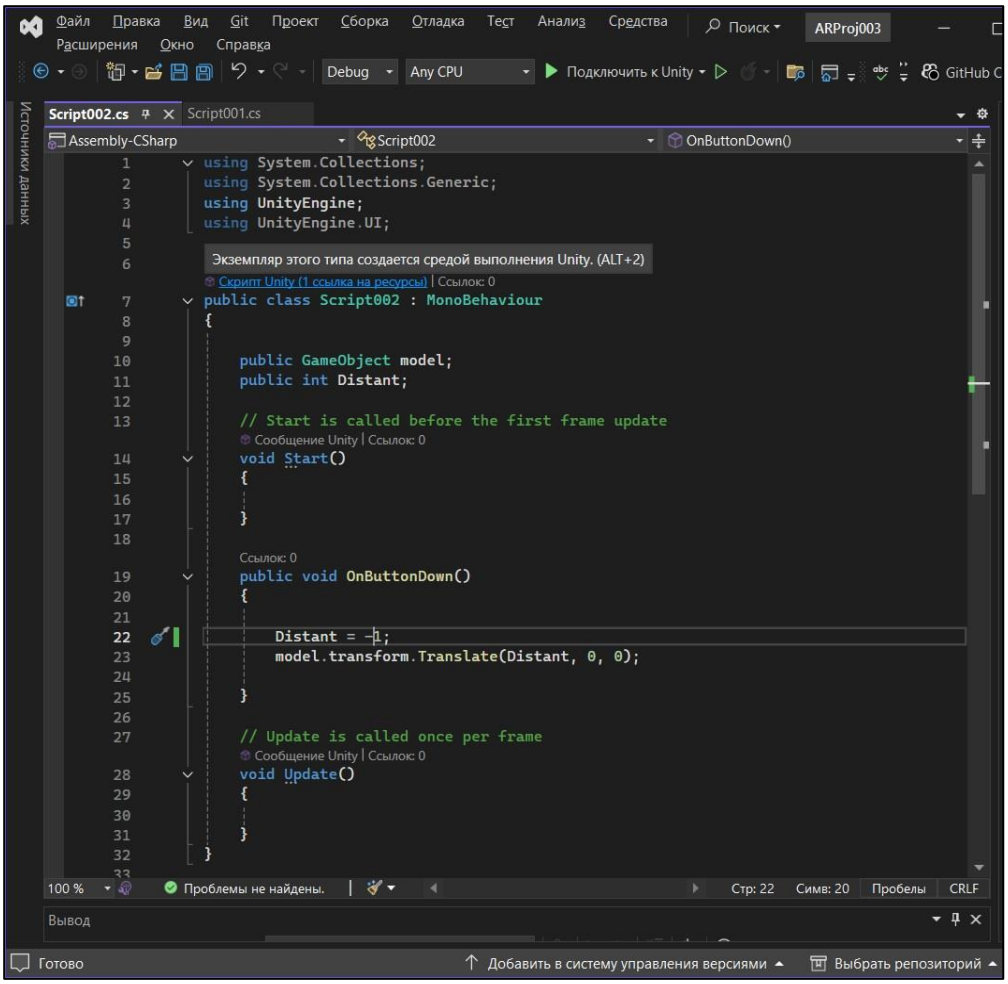
Результат:



Кнопку ButtonB определяем аналогично. Результат ее настроек приведен ниже:



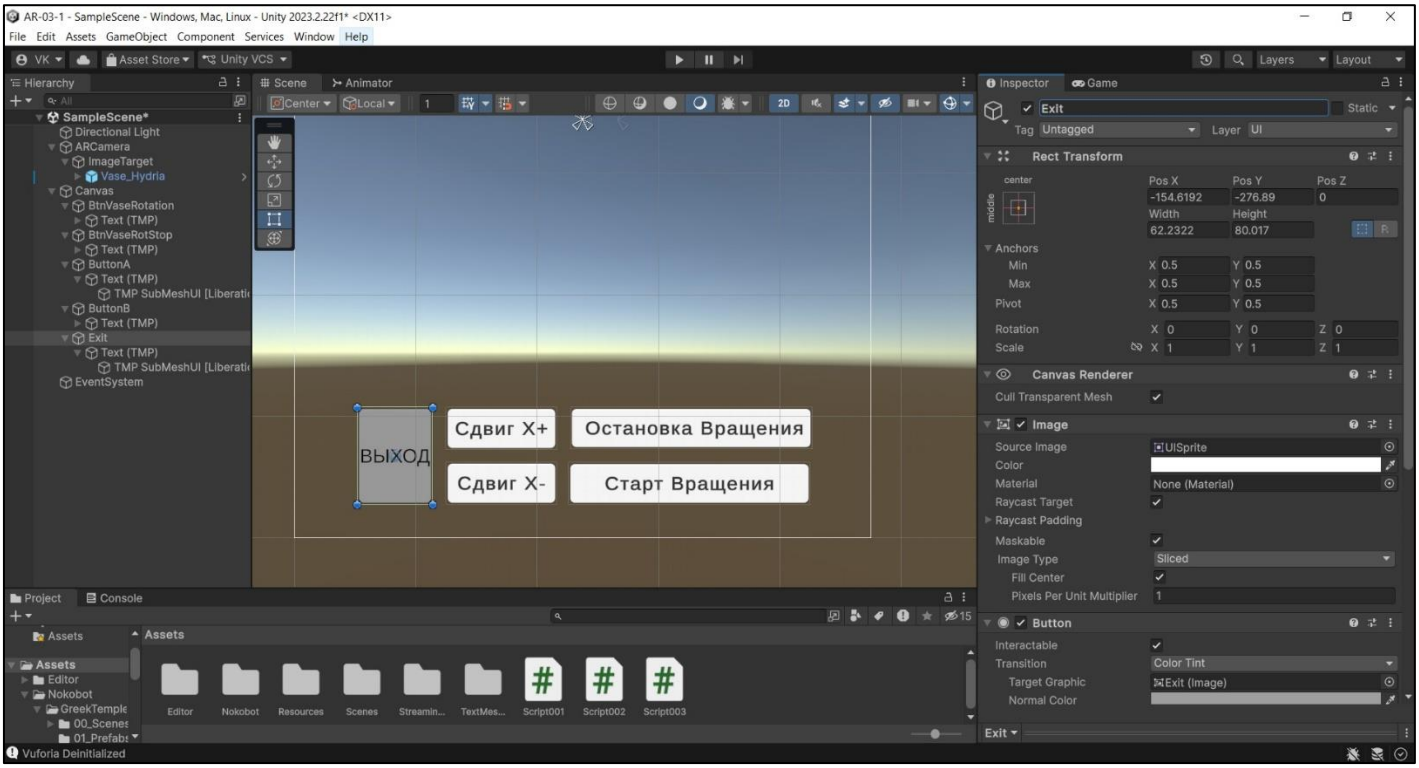
Скрипт Script002:



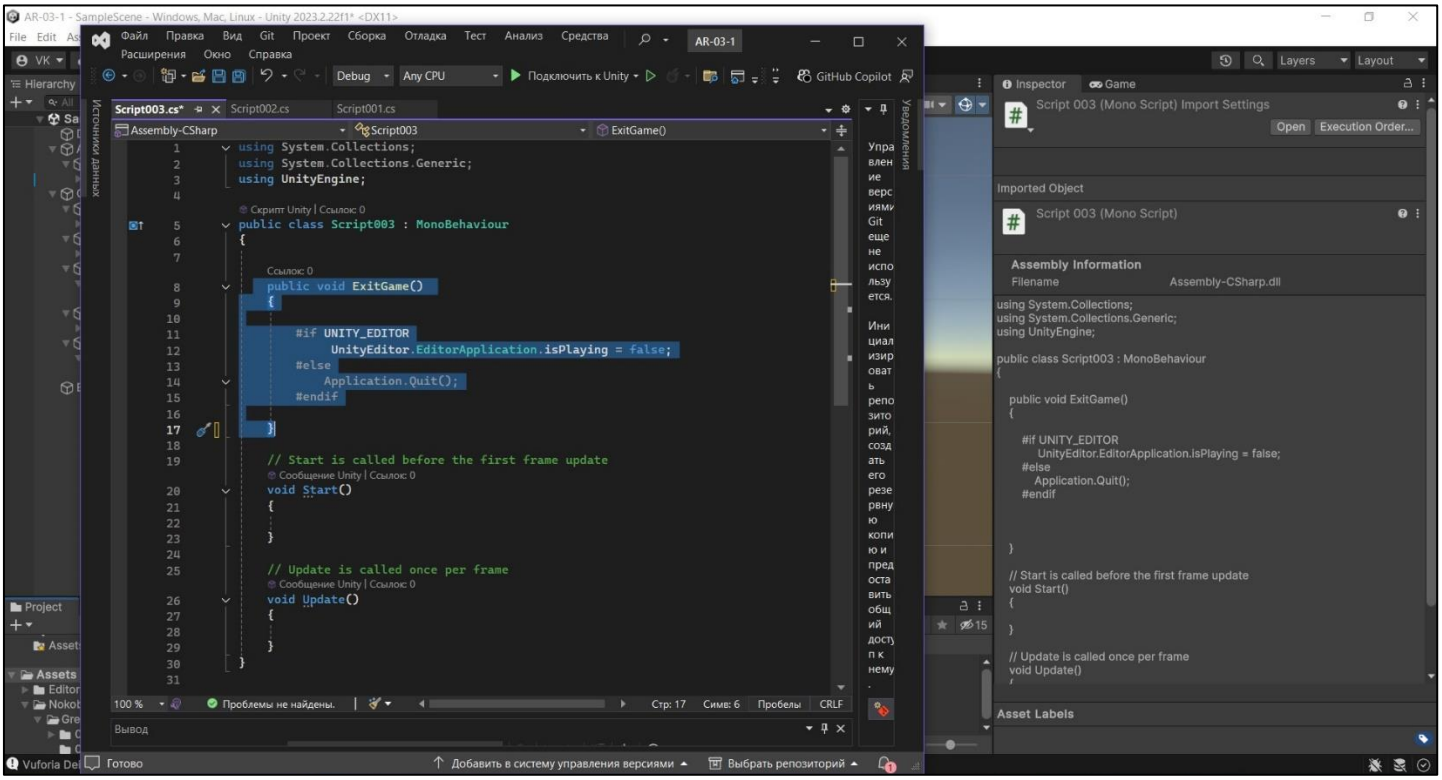
Работу обоих кнопок можно проверить в режиме **Game (→ Play)**.

2. Создание элемента управления плоского меню - кнопки для управления выходом из Приложения.

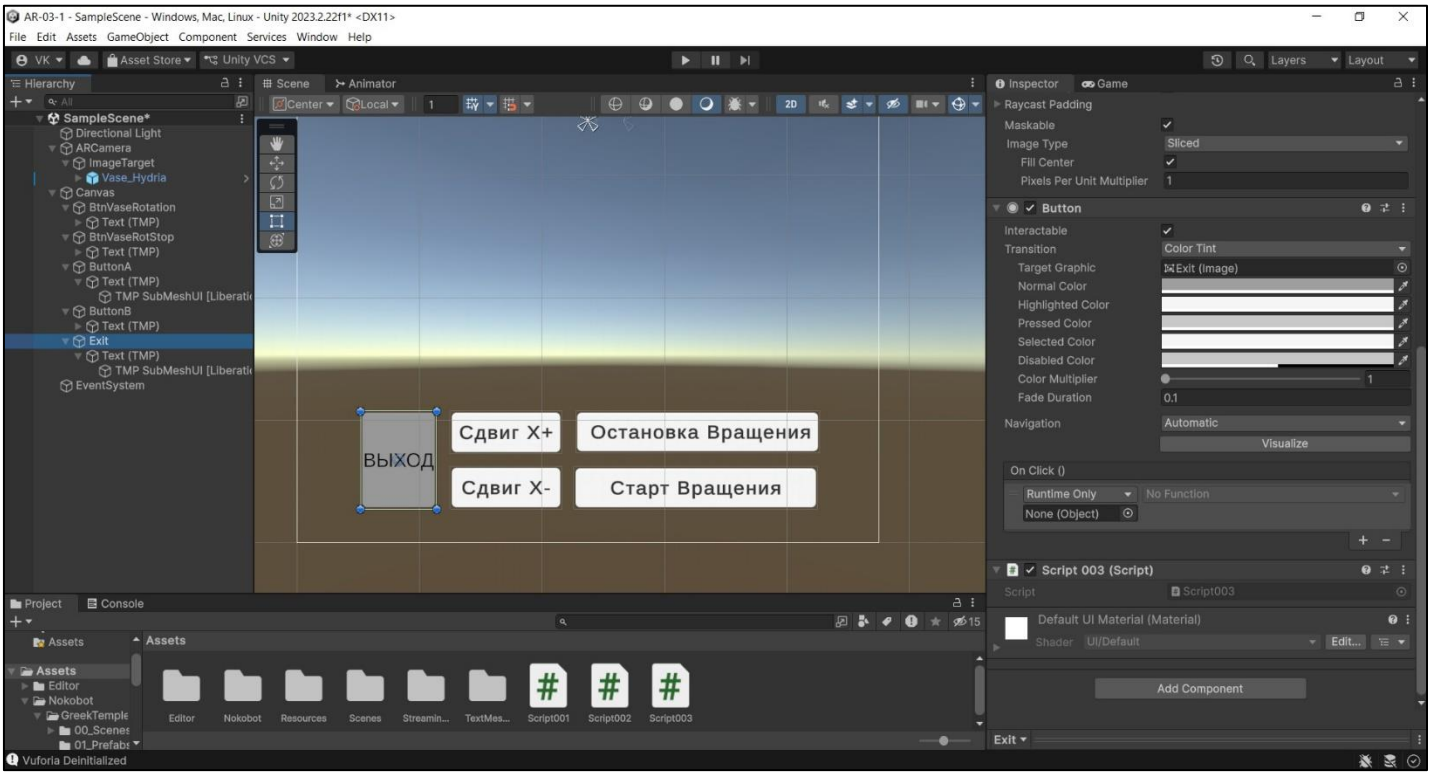
По аналогии с шагами в предыдущем пункте в готовой сцене добавляем в канву **2D Button** (переименуем ее в **иерархии** уже на данном этапе – **Exit**) для выхода из разработанного приложения («сцены»):



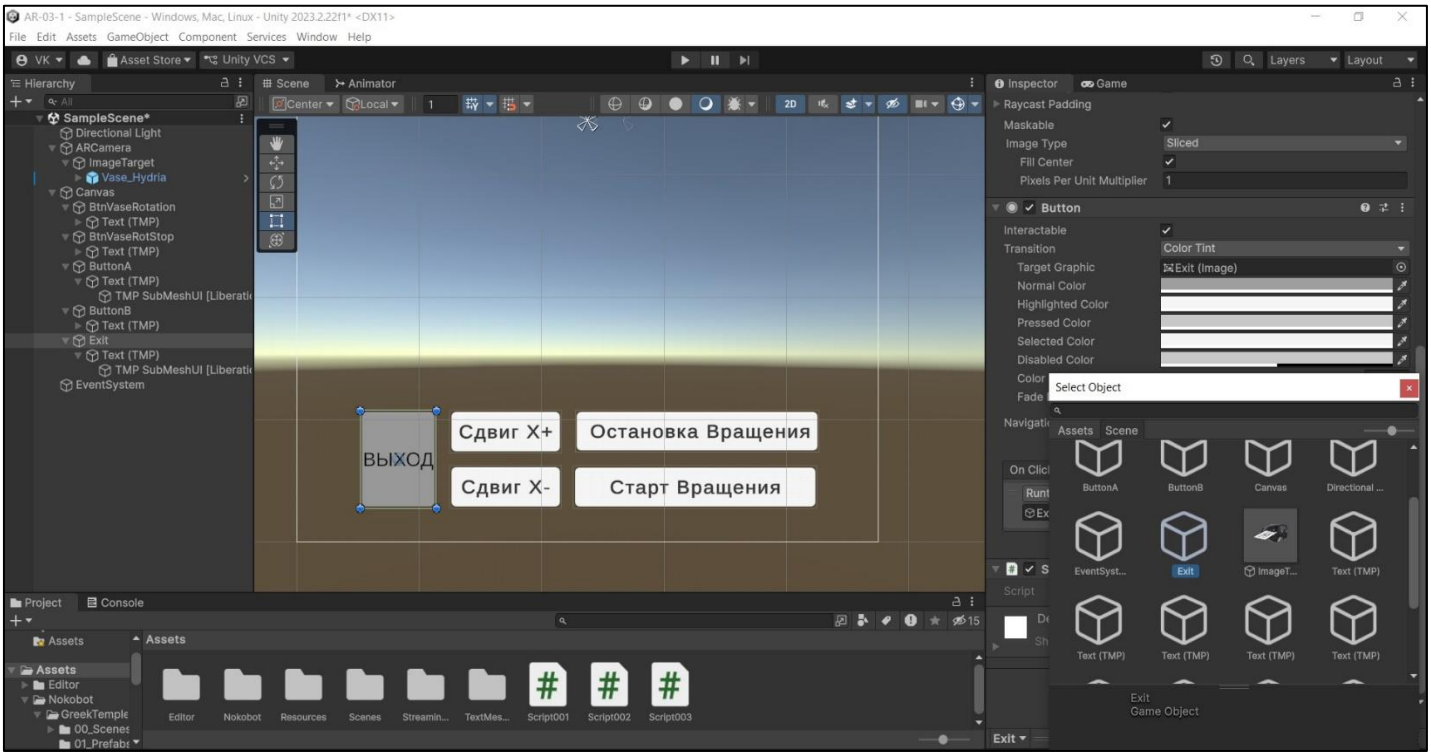
Script003 для кнопки Exit создаем аналогично пункту 1.



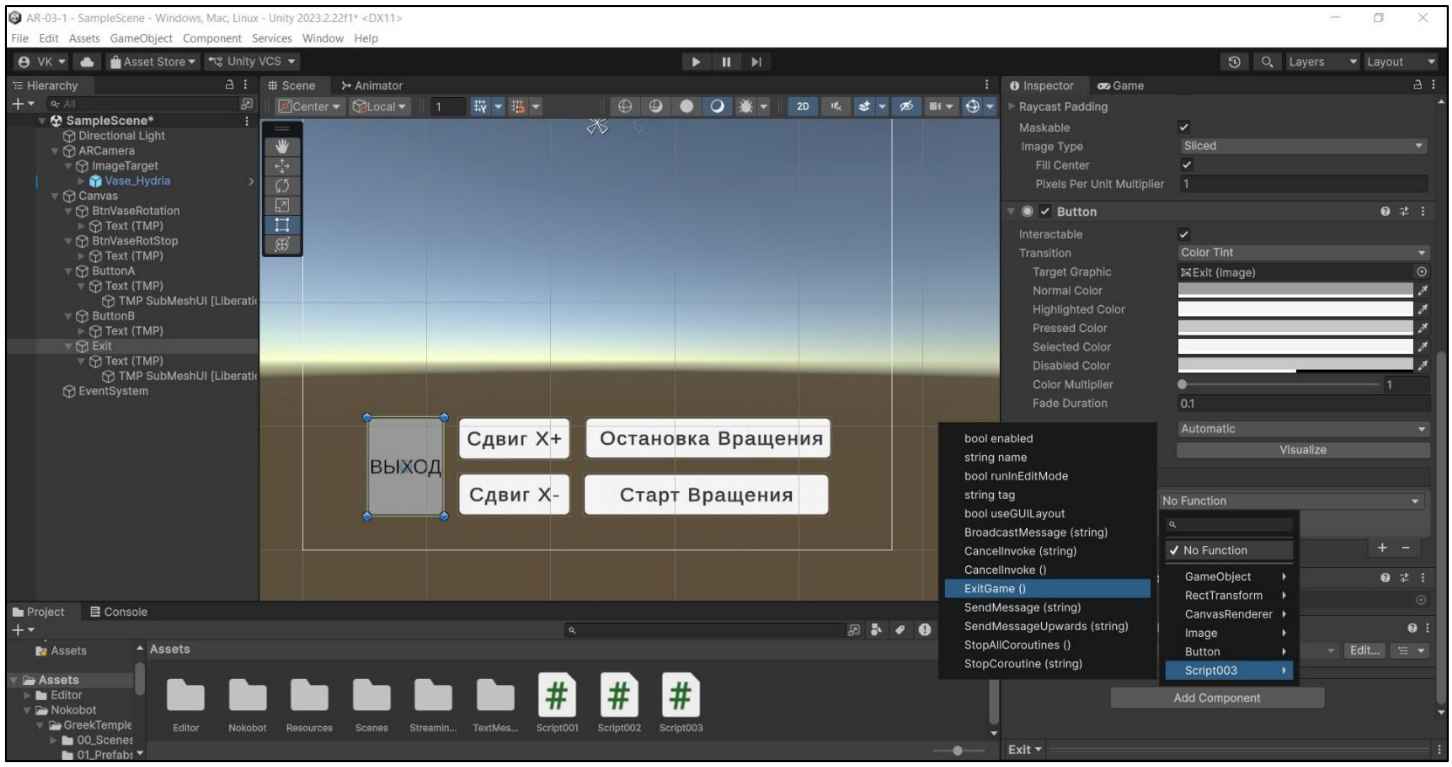
Сохраняем скрипт **Script003**. В отличие от предыдущих кнопок **ButtonA и ButtonB** в скрипте нет ссылок на **Game Object**. Поэтому сразу переходим в настройки **On Click ():**



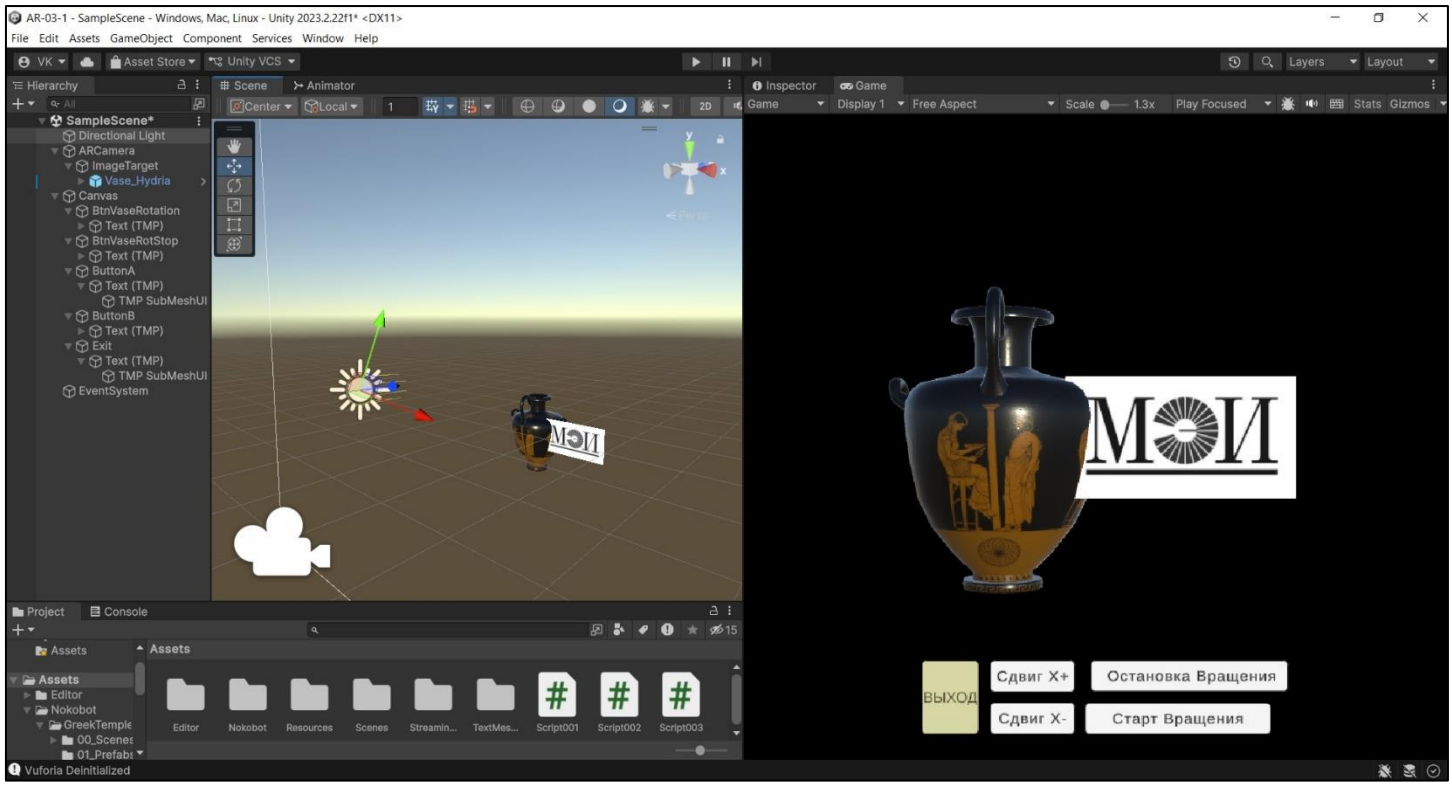
Как обычно выбираем режим **Runtime Only,** в позиции **None (Object)**, в закладе **Scene**, выбираем кнопку **Exit:**

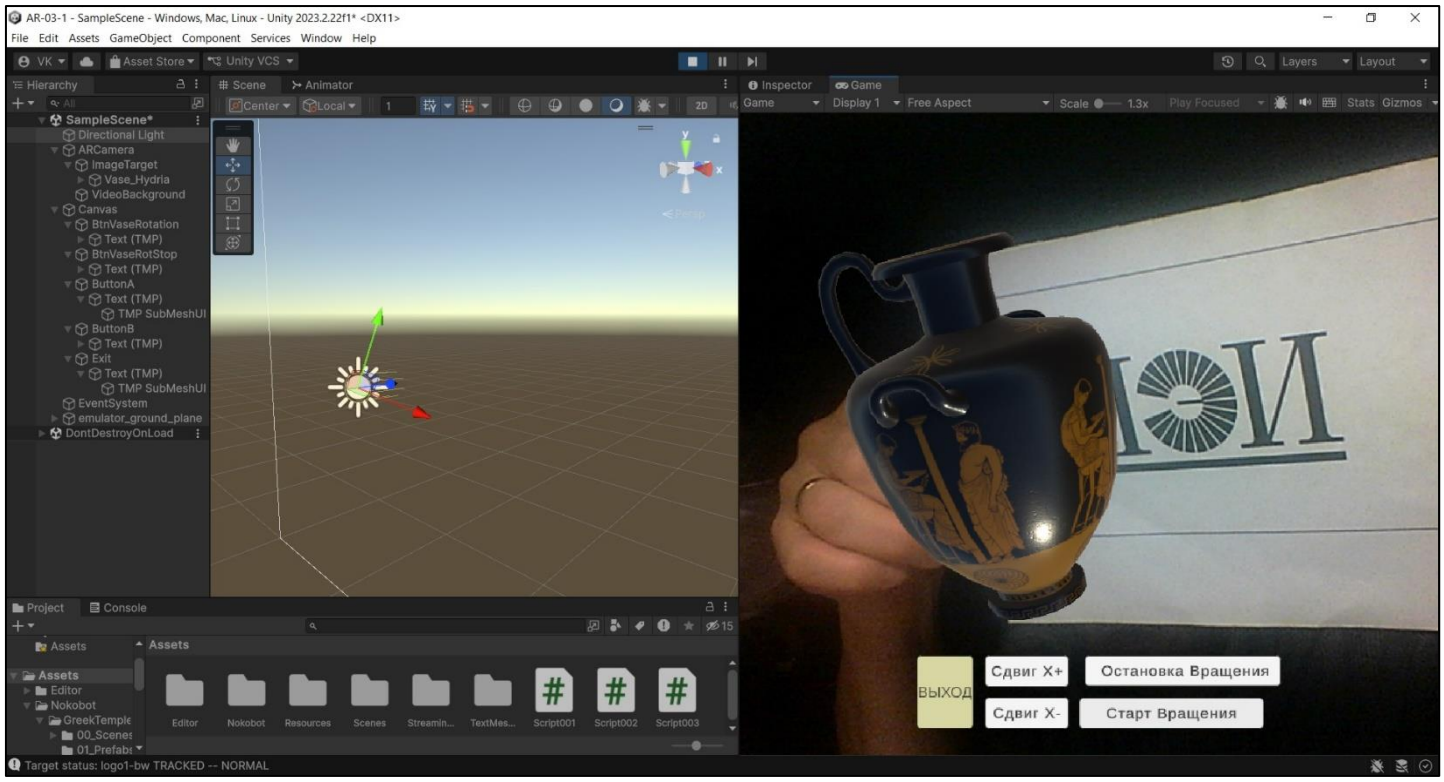


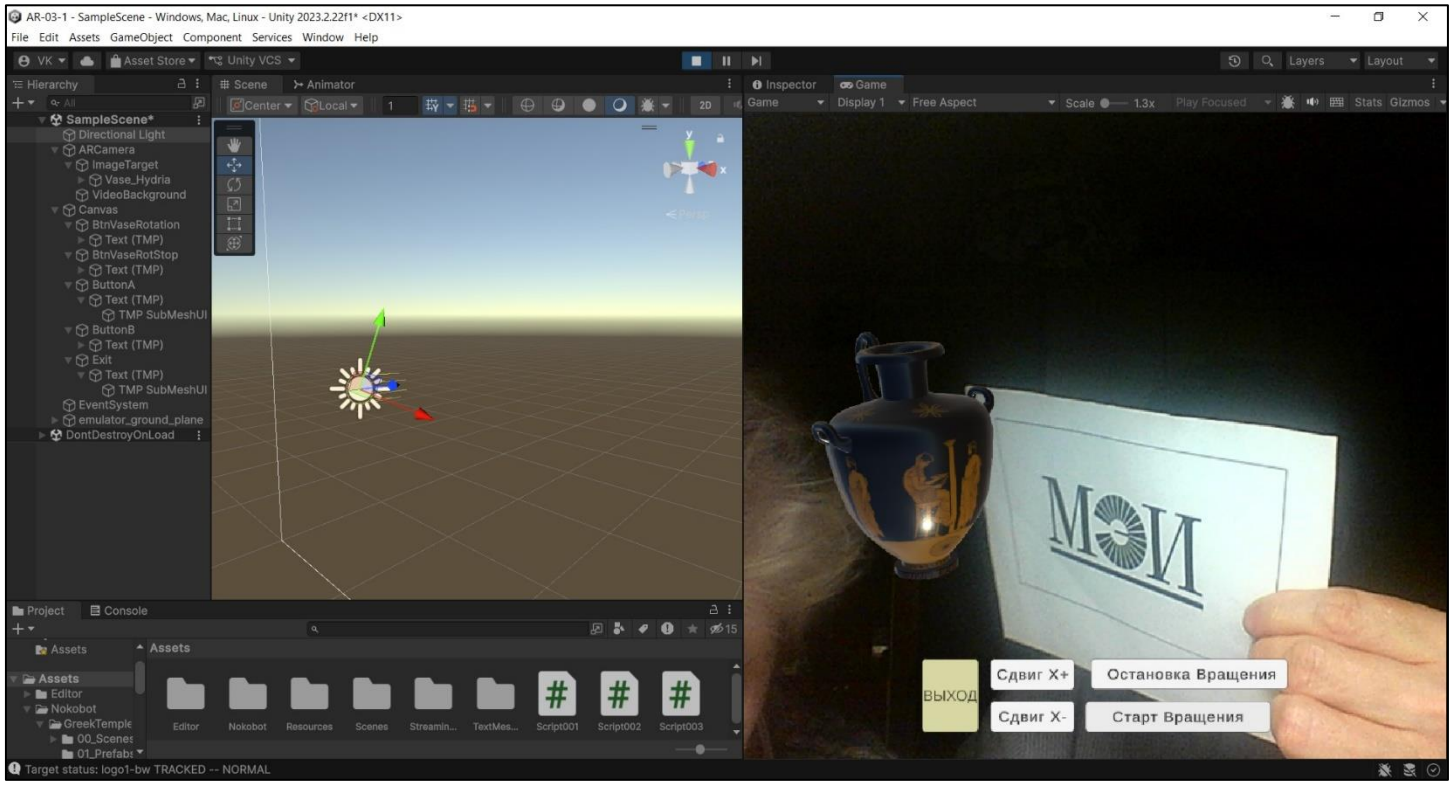
В поле функционала **(No Function)** выбираем метод **ExitGame ()** скрипта **Script003:**

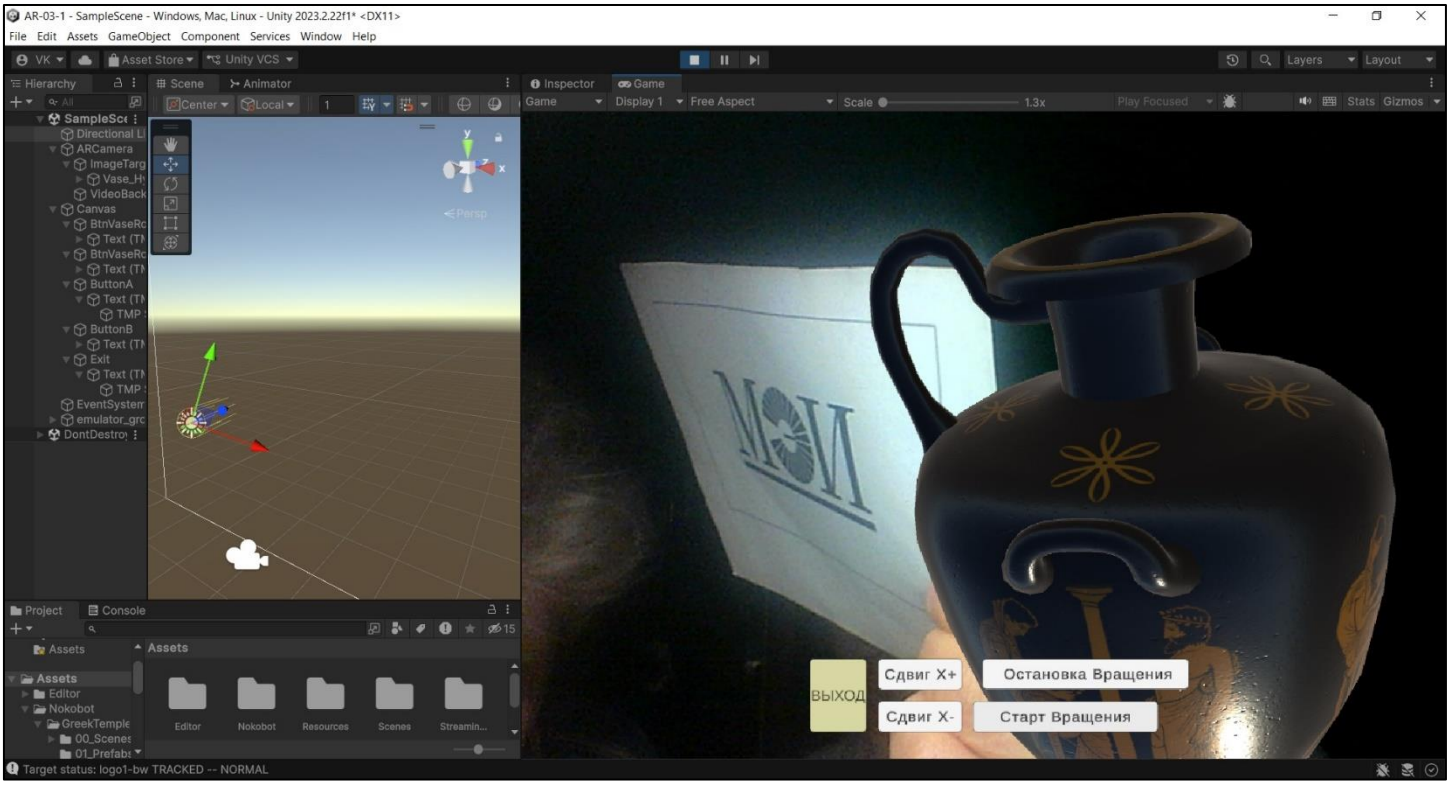


Теперь работу всех управляющих элементов плоского меню (Кнопки) можно проверить в режиме Game (→ Play):




AR-03-1 - SampleScene - Windows, Mac, Linux - Unity 2023.2.22f1* <DX11>
File Edit Assets GameObject Component Services Window Help
Hierarchy
SampleScene*
Directional Light
ARCamera
ImageTarget
Vase_Hydria
VideoBackground
Canvas
BtnVaseRotation
Text (TMP)
BtnVaseRotStop
Text (TMP)
ButtonA
Text (TMP)
TMP SubMeshUI
ButtonB
Text (TMP)
Exit
Text (TMP)
TMP SubMeshUI
EventSystem
emulator_ground_plane
DontDestroyOnLoad
Scene
Animator
Inspector
Game
Project
Console
Assets
Editor
Nokobot
Resources
Scenes
Script001
Script002
Script003
ВЫХОД
Сдвиг X+
Остановка Вращения
Сдвиг X-
Старт Вращения
Target status: logo1-bw TRACKED -- NORMAL


AR-03-1 - SampleScene - Windows, Mac, Linux - Unity 2023.2.22f1* <DX11>
File Edit Assets GameObject Component Services Window Help
Hierarchy
SampleScene*
Directional Light
ARCamera
ImageTarget
Vase_Hydria
VideoBackground
Canvas
BtnVaseRotation
Text (TMP)
BtnVaseRotStop
Text (TMP)
ButtonA
Text (TMP)
TMP SubMeshUI
ButtonB
Text (TMP)
Exit
Text (TMP)
TMP SubMeshUI
EventSystem
emulator_ground_plane
DontDestroyOnLoad
Scene
Animator
Inspector
Game
Project
Console
Assets
Editor
Nokobot
Resources
Scenes
Script001
Script002
Script003
ВЫХОД
Сдвиг X+
Остановка Вращения
Сдвиг X-
Старт Вращения
Target status: logo1-bw TRACKED -- NORMAL



Таким образом в одном проекте были разработаны и добавлены на «стекло» (плоское меню управления контентом) Приложения ДР пять кнопок для управления:

1. **ранее разработанной анимацией (анимациями) трехмерной модели – старт и остановка анимации;**
2. **перемещением объекта контента (трехмерная модель) по одной из осей в положительном и отрицательном ее направлении;**
3. **выходом из Приложения.**

Разработка велась в среде Unity 2023.2.22f1 и Visual Studio 2022

Полностью разработанное плоское меню (меню на стекле) необходимо интегрировать (если этого еще не было сделано) в один проект с остальным контентом и протестировать его в режиме Game.

Далее необходимо подготовить файл .apk – Приложение ДР - для загрузки на МУ – ТУ ДР.

Полученный результат продемонстрировать/отослать для демонстрации преподавателю вместе с коротким отчетом об основных этапах работы. Особое внимание в отчете обратить на проблемы, если они возникли.