Лабораторная работа № 2. Проект – разработка простого AR-Приложения для Android-устройства (смартфон, планшет и пр.). Создание в графическом редакторе Unity 3D сцены дополненной реальности: *визуализация объекта контента - видеоклипа*.

Объекты контента ДР – это объекты проекта, создаваемого с помощью средств платформы Vuforia.

Введение.

Работа по созданию приложений ДР заключается в заведении проекта и объектов проекта (Контент) в **Vuforia**, а разработка **3D**-сцен для объектов этого проекта осуществляется в **Unity 3D**. При этом **Vuforia** отвечает за идентификацию проекта через **License key** (см. ниже), а привязка к будущей сцене виртуального **2D-** или **3D-**объекта (например, **3D-**модели, плоских изображений, видеоклипов и пр.) будет осуществляться через определяемую в **Vuforia** метку (**Target**). Допустимые в используемой в Лабораторном практикуме версии **Vuforia Engine** типы таргетов будут подробно рассмотрены ниже.

ВАЖНО!! → вся работа с **Vuforia** (с проектом, объектами) осуществляется через **web**-интерфейс, иными словами, **Vuforia** является облачным приложением. А работа с **Unity-3D** осуществляется непосредственно на компьютере разработчика, т.е. локально.

Связь между облачным ведением проекта (в **Vuforia**) и локальной проработкой сцен Приложения ДР должна быть выполнена за счет импорта подготовленных объектов проекта из облака **Vuforia** в среду редактора **Unity-3D**.

Рассмотрим типовую процедуру создания простого (игрового) **AR Приложения**.

Предлагается разработать приложение ДР для Android-устройств, в котором при наведении камеры устройства на реальную метку (таргет – изображение, например, на бумаге, или на дисплее) пользователь в области воспроизведения на экране мобильного устройства (МУ) увидит другое, заранее подготовленное 2D – изображение (лист инструкции по эксплуатации, пояснение, другую картинку, видеоклип и т.п.)

Предварительные условия для начала работы:

- Интернет-соединение локального компьютера;
- Наличие аккаунта пользователя **Vuforia** (результат успешного выполнения **ЛР №1**);
- Установленная на компьютере разработчика система **Unity 3D** (результат успешного выполнения **ЛР №1**);
- Заранее подготовленные изображения для метки (таргета) и контент (видеоклип);

В результате успешного выполнения **ЛР №1,** получены из облака **Vuforia** и сохранены в локальной файловой системе для локальной работы в **Unity 3D** следующие ресурсы:

- **установочный** специфический объект **Vuforia** – файл в формате **.unitypackage**
- **лицензия**
- **БД таргетов -** файл в формате **.unitypackage:**

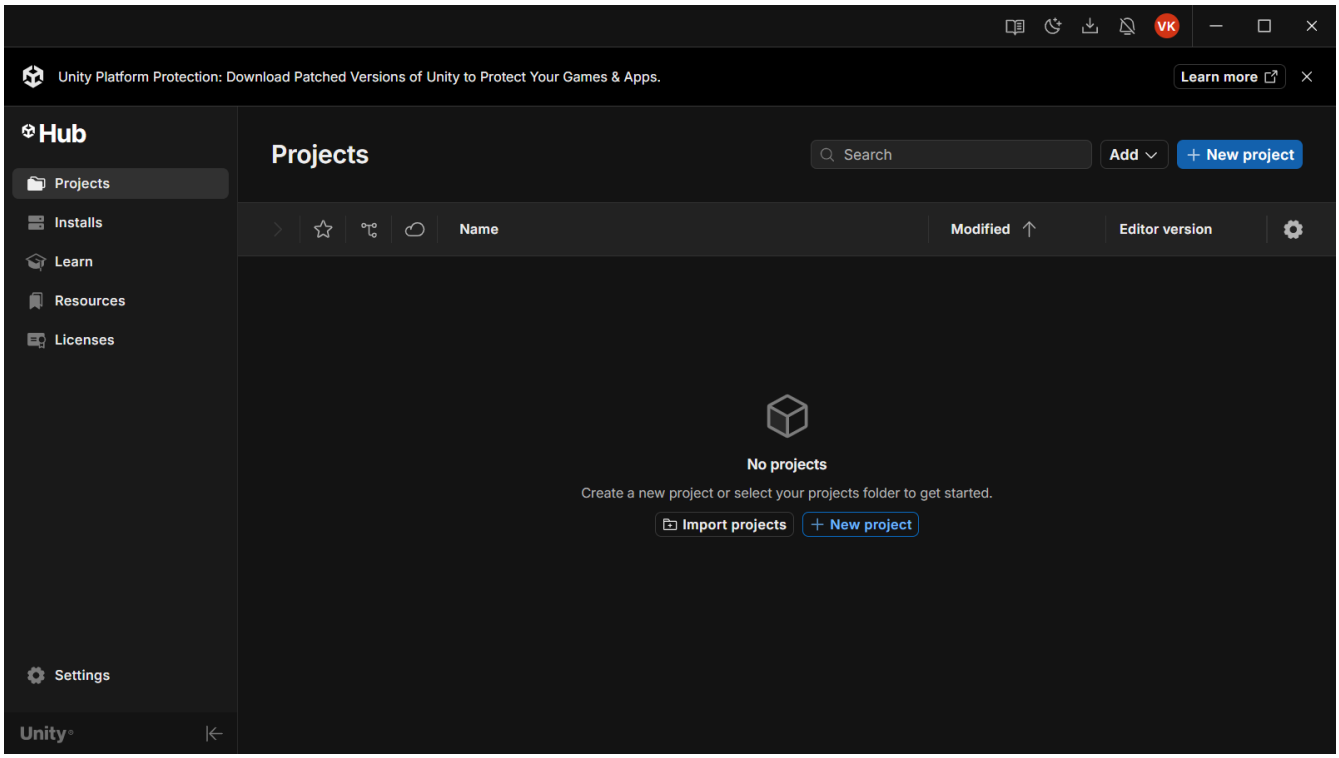
Кроме того, д. б. заранее подготовлены элементы контента ДР для воспроизведения их на экране **Android**-устройства в Приложении ДР: в данном описании первой части ЛР №2 это видеоклип в формате **.mp4** – файл **AR-001Video.mp4.**

1. Настройка среды разработки Приложений ДР – Vuforia Engine & Unity 3D.

Стартуем приложение **Unity 3D** (для загруженной в ЛРN№1 версии **Unity** это рекомендуется делать через **Unity Hub**) и выполняем настройки, необходимые для начала работы с **Vuforia**.

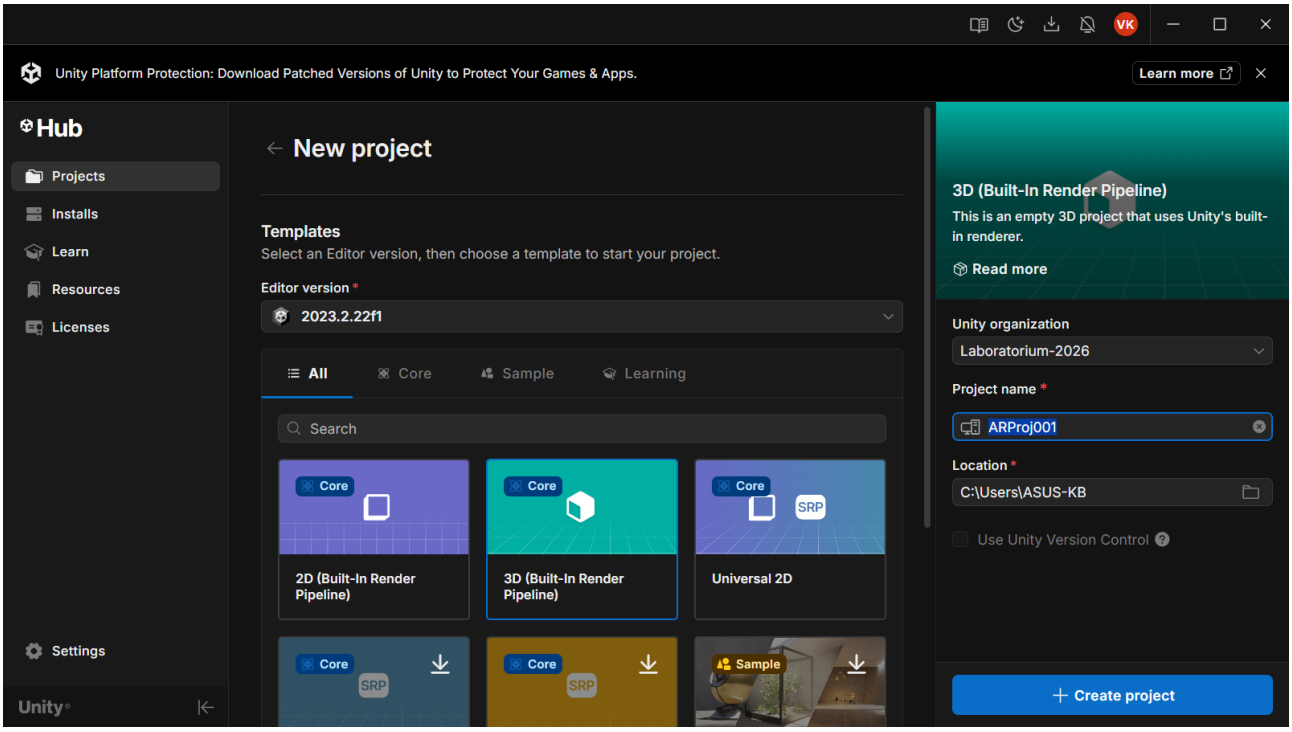
Если Первая ЛР выполнена корректно, вы увидите в открывшемся окне заведенный ранее проект. Можно повторить процедуру создания нового проекта еще раз.

Стартуем **Unity Hub** и переходим в раздел **Projects:**



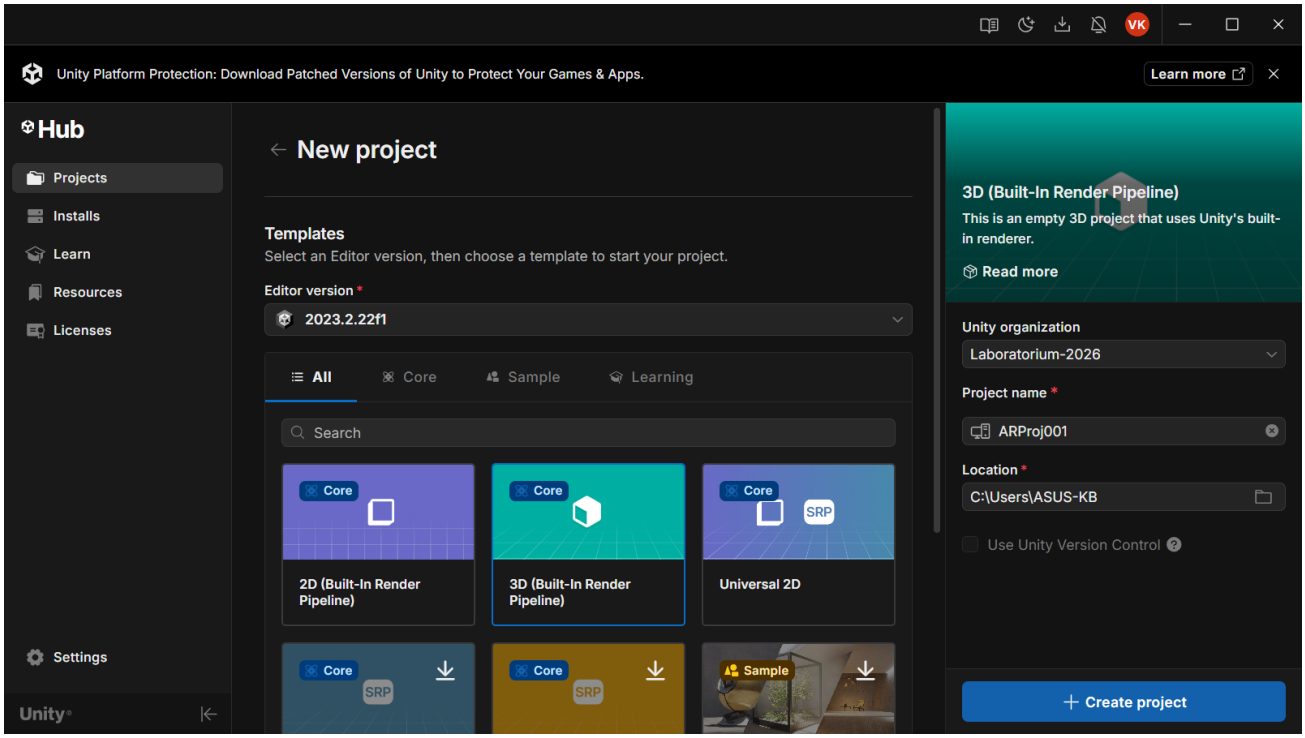
Переходим по ссылке **+New Project →**

- Выбираем нужную версию **Edit**,
- тип приложения: **Core – 3D (Build in Render Pipeline)**

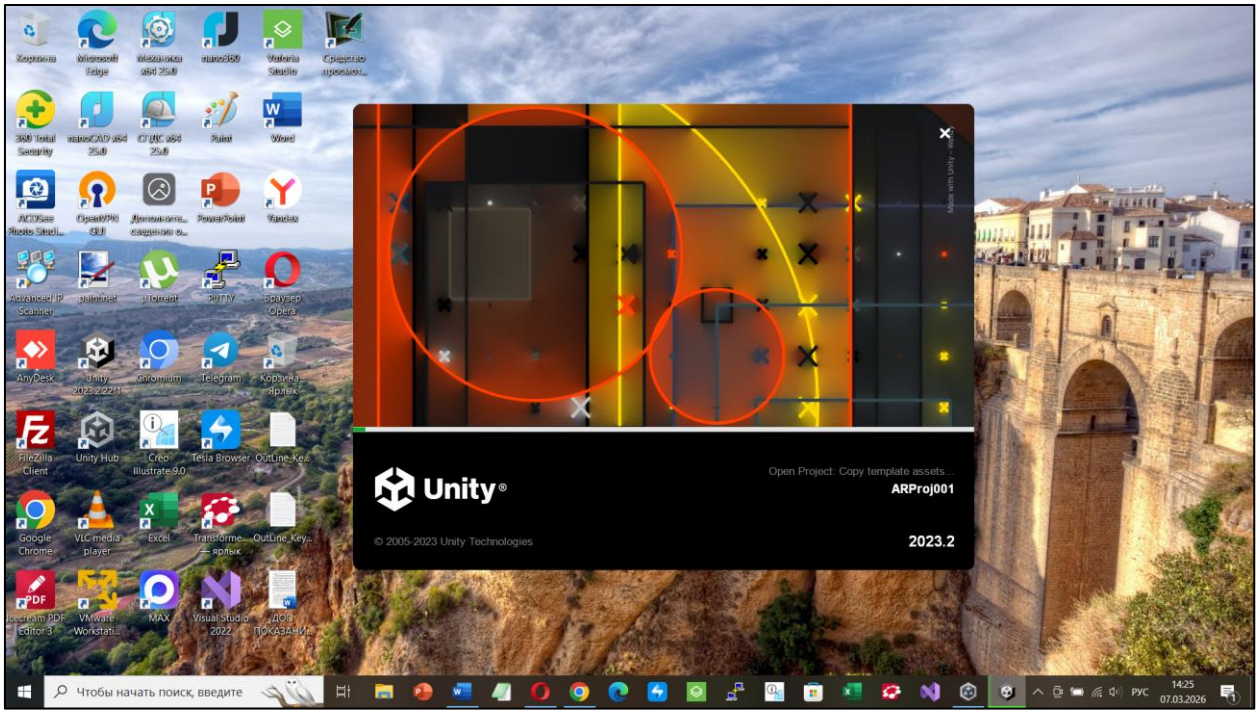


Убедитесь, что ваш проект будет размещен не в облаке, а в локальной файловой системе.

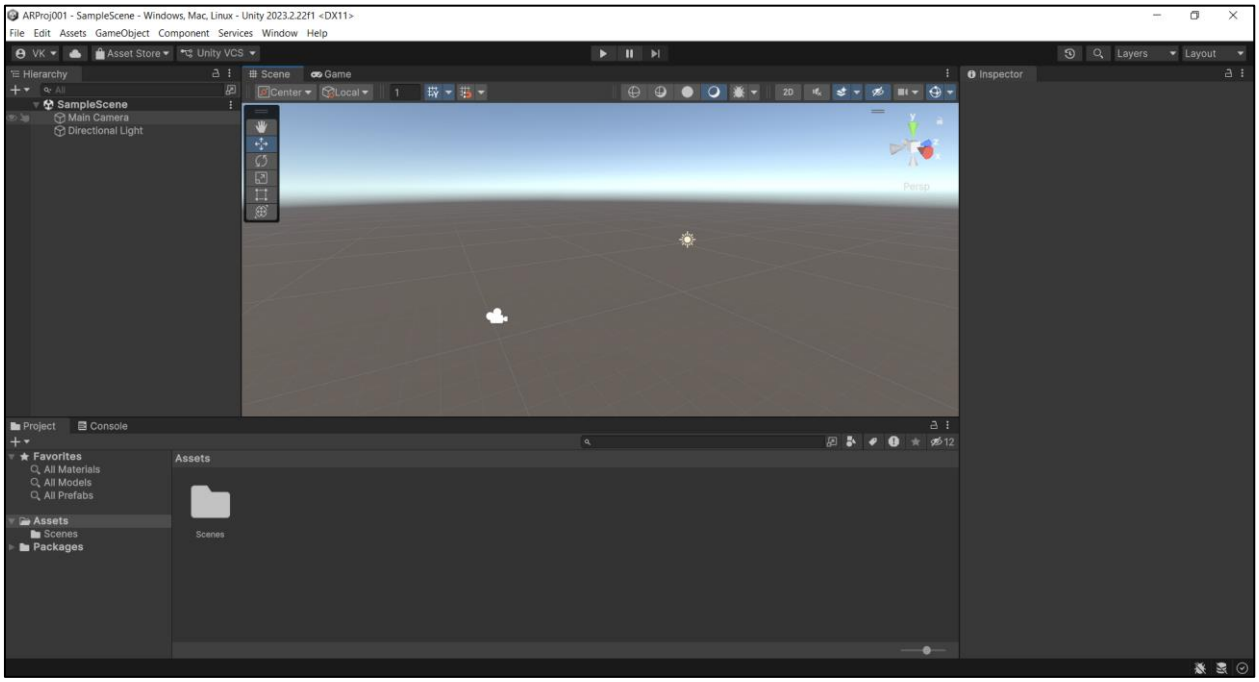
Далее необходимо заполнить поля **Unity Orgfnization** (любое имя), **Project name** (любое имя, например, **ARProj001**) и, если необходимо, изменить локацию проекта в локальной файловой системе (или оставить локацию «по умолчанию»)



Выбираем опцию **+Create Project**



Через несколько минут появляется окно редактора Unity 3D



Документацию по работе в редакторе – см. по ссылке:

Unity - Manual: Unity's interface

В зависимости от версии **Unity**, структура его редактора, представленная ниже, может изменяться. Однако стандартный состав полей редактора остается неизменным.

Позиции на рисунке – см. по ссылке **Unity - Manual: Unity's interface**

Обратите внимание - начальные действия по организации среды игрового движка **Unity 3D** для нашего проекта (Приложение ДР) происходят с использованием закладки **Scene** (область **Scene View**), группы элементов линейки **Toolbar**, и элементов закладки **GameObject**, входящей в состав ленты основного меню (**File Edit Assets GameObject …**).

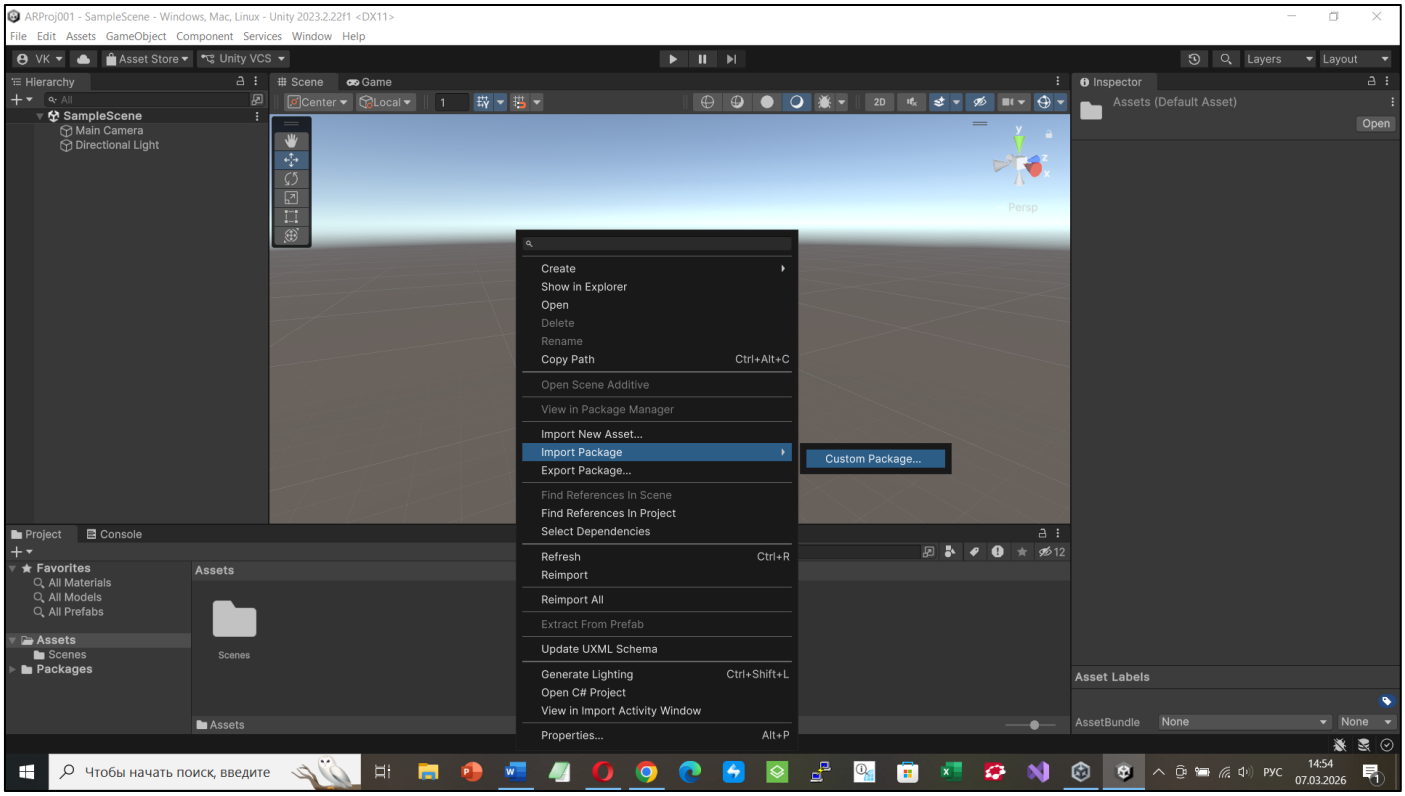
Без дополнительных настроек редактор **Unity 3D** работает в режиме **Виртуальной реальности**. По условиям Задания **ЛР №2** наше **Android**-устройство в Проекте должно работать в режиме **Дополненной реальности**. Как отмечалось выше, за режим ДР отвечает **Vuforia**. Это означает, что в список объектов **Unity 3D (GameObject)** необходимо добавить специфические объекты **Vuforia**. А работу с ними поддержать действующими лицензиями **Vuforia**.

Для этого необходимо файлы (**пакеты → packages**) типа **.unitypackage** загрузить в **Unity 3D.** Эти файлы (пакеты) были заранее подготовлены и загружены из облака **Vuforia** на локальную машину (**см. ЛР1**).

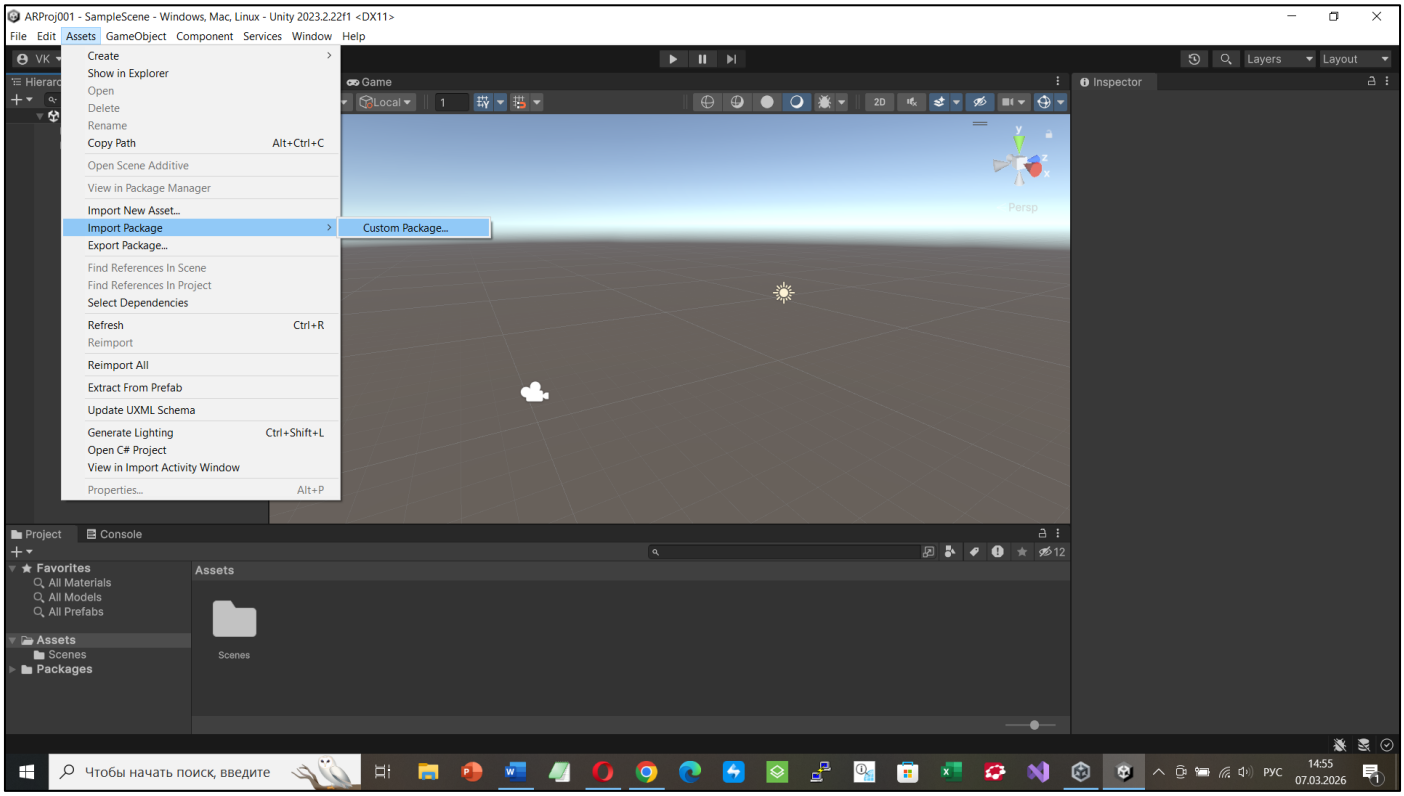
Выполнить данную операцию необходимо с помощью пакетного менеджера редактора **Uniy – Package Manager**.

В более ранних версиях его вызов осуществлялся из меню **Window**, но теперь для задачи импорта пользовательского пакета **.unitypackage** используются более простые способы:

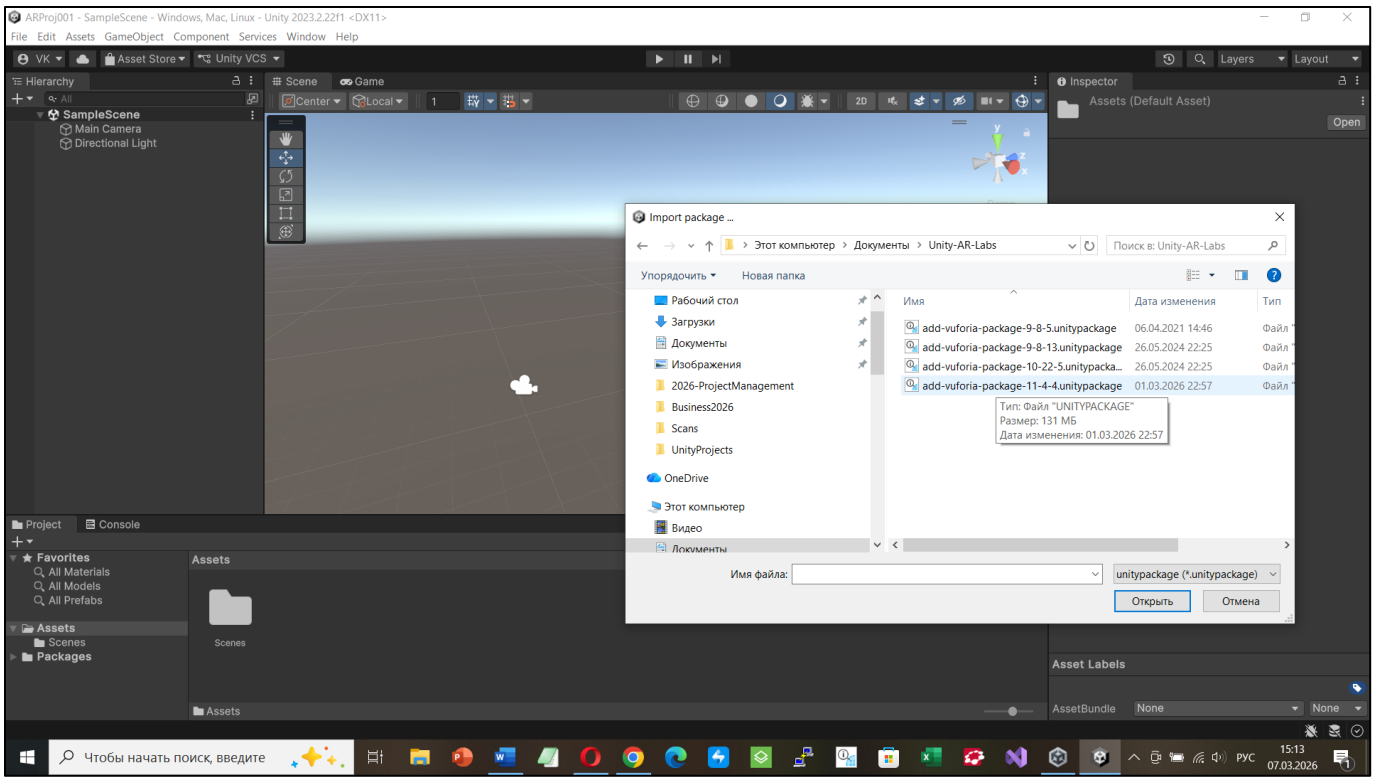
- либо путем вызова процесса **Custom Package →** клик RMB в поле **Assets:**

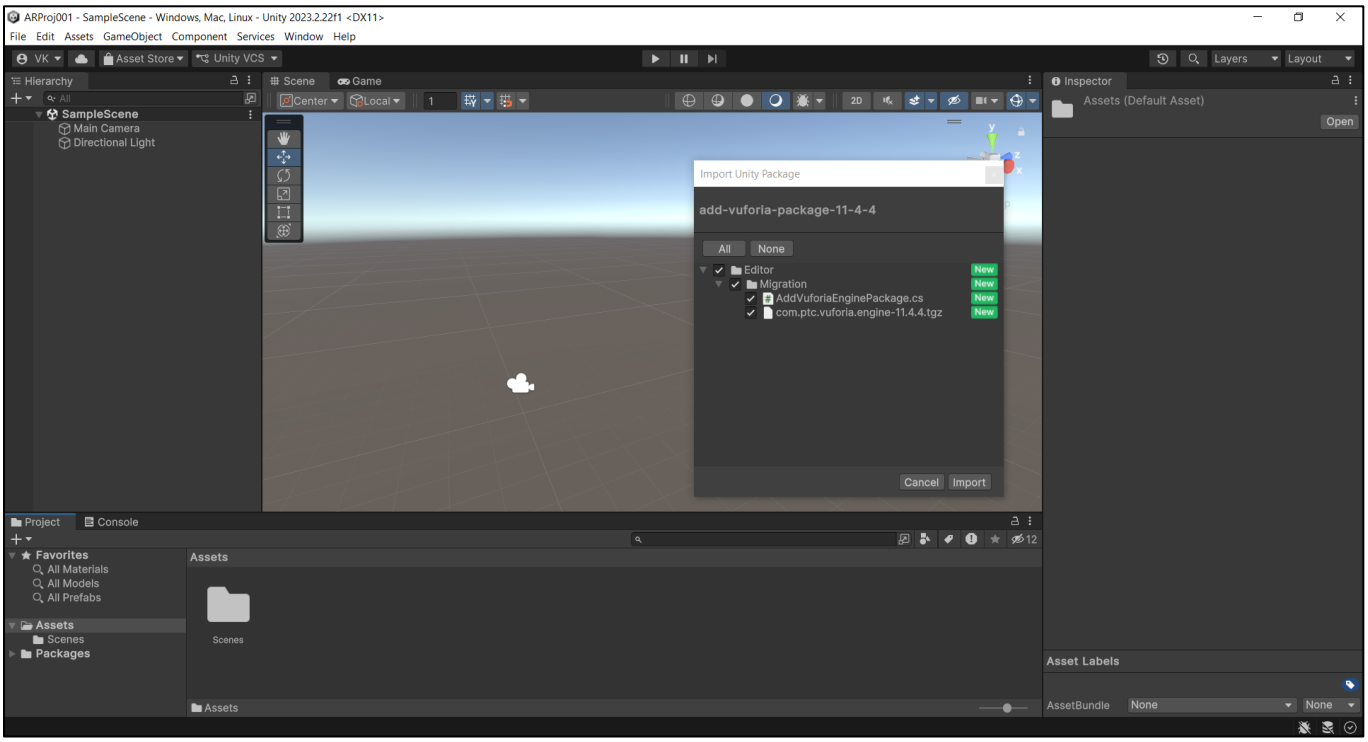


➢ либо, что тоже самое, через меню ленты **Assets:**

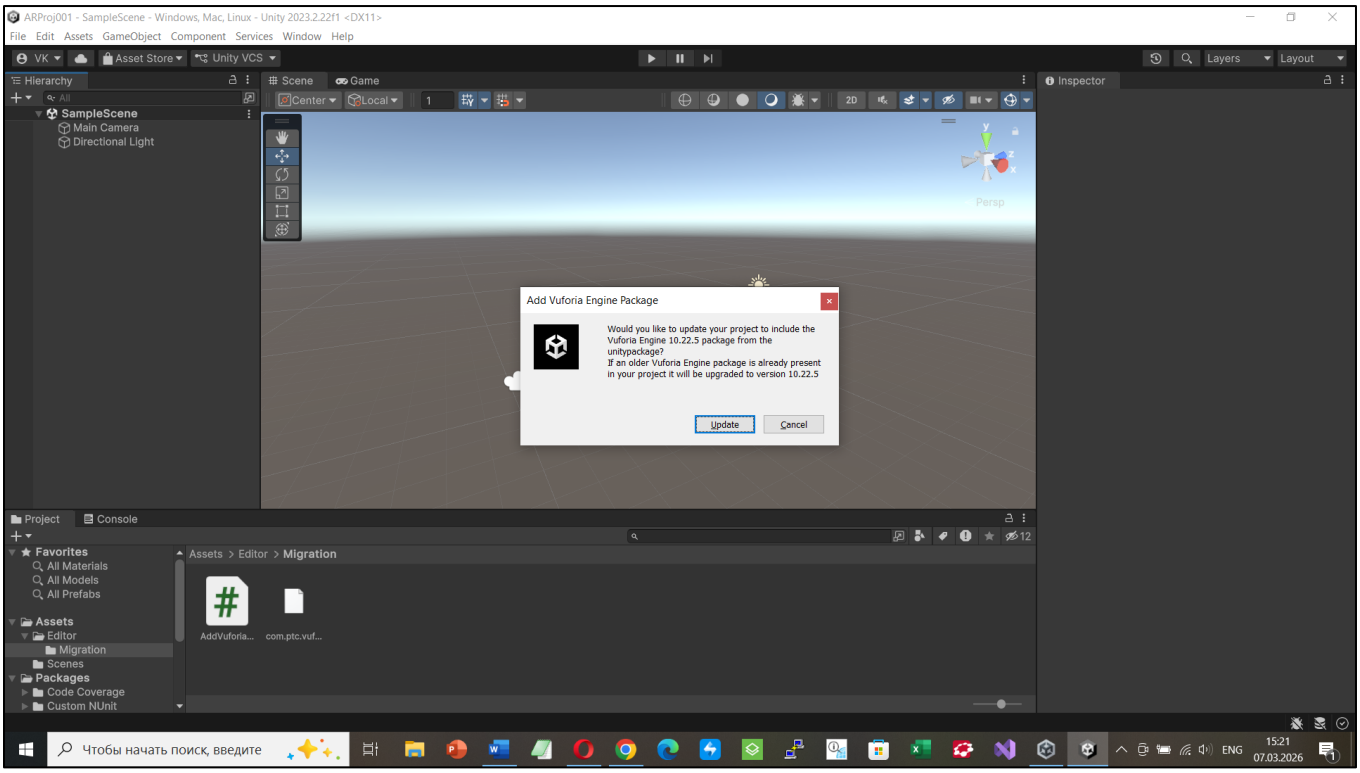


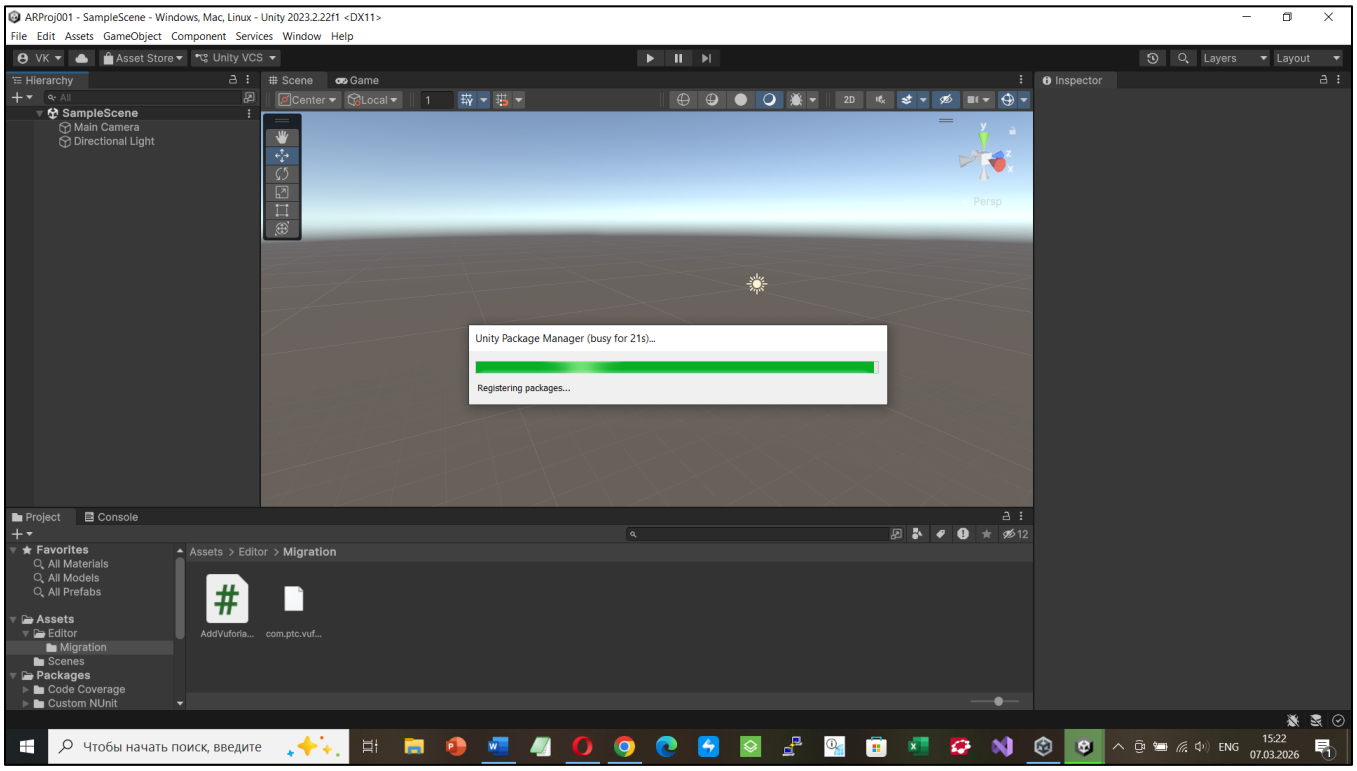
В результате вы получаете доступ к локальной файловой системе, ищете там папку, где ранее был сохранен специфический объект - установочный файл работы с объектами **Vuforia Engine, и «открываете»** необходимый:



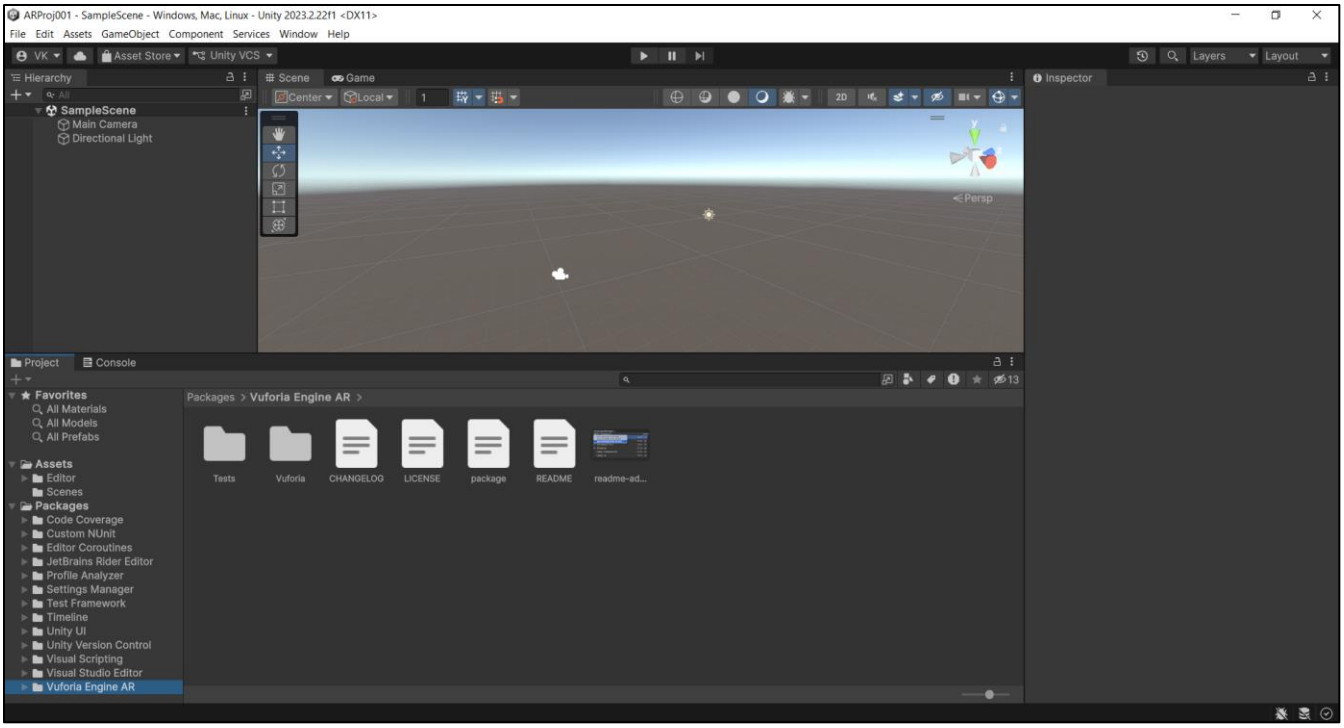


В появившемся меню проверяем состав пакета и нажимаем **«Import».** Обратите внимание – на некоторых этапах выполнения проекта система может предложить вам обновить версию **Vuforia Engine** на более новую**. Выполните данную рекомендацию!**

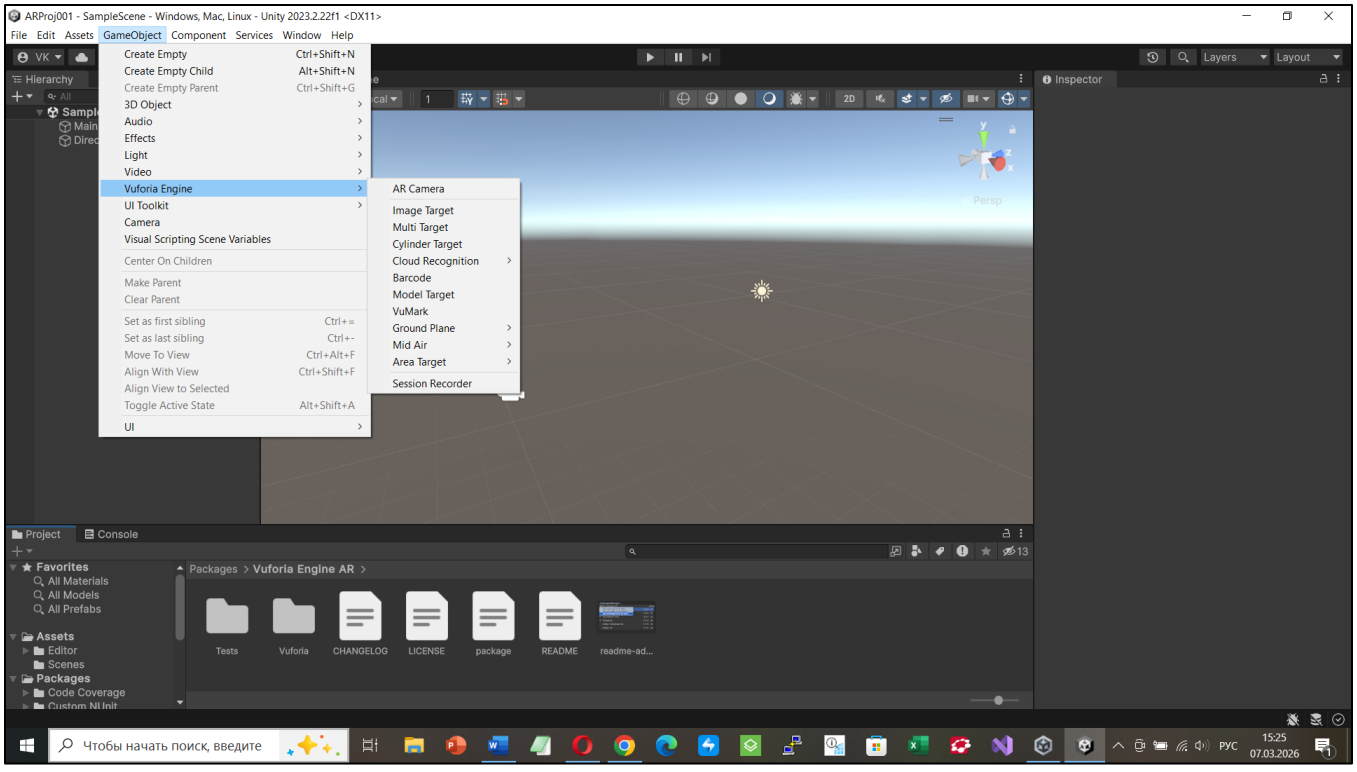




В результате в **Assets-Packages** появляется новый установленный пакет **Vuforia Engine AR** для дополненной реальности →



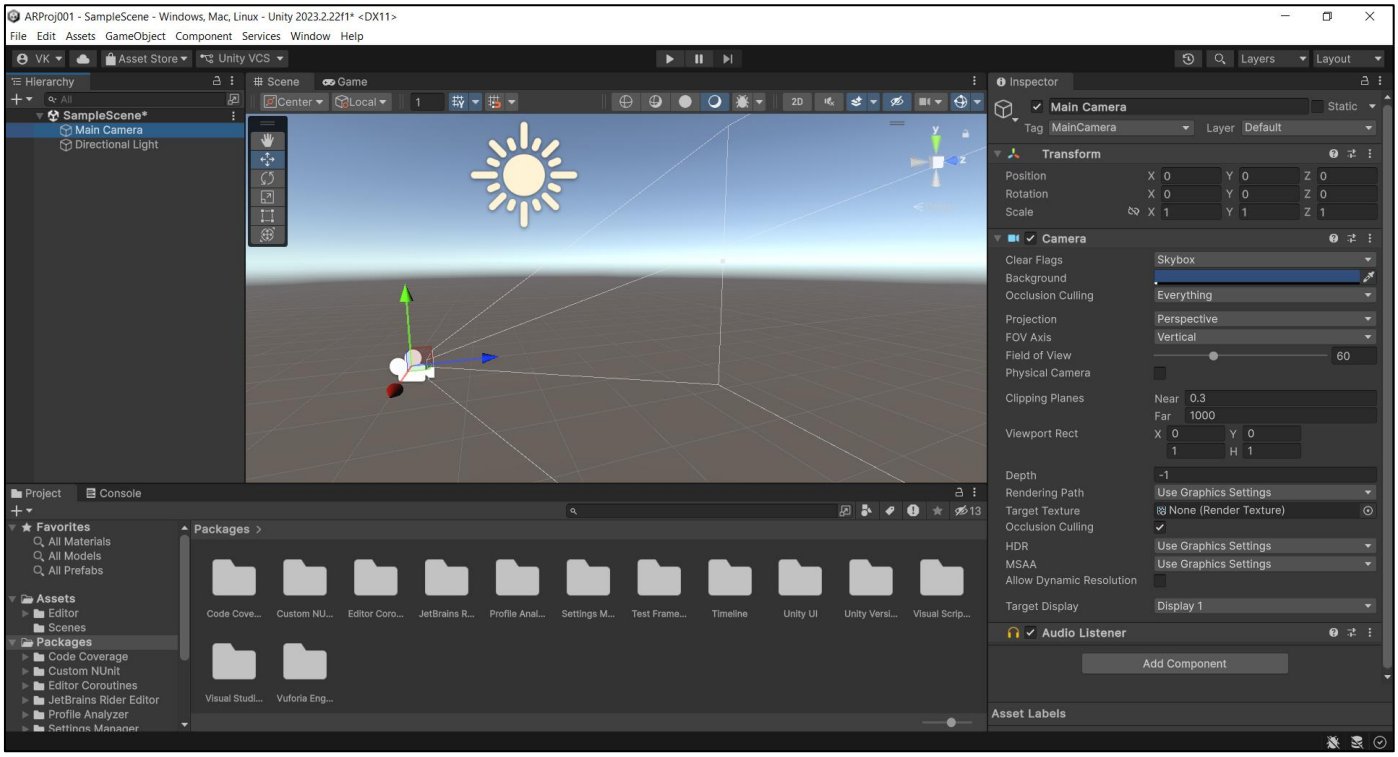
Убеждаемся, что в закладке **Game Object** в редакторе **Unity** появилась новая группа объектов – **Vuforia**:



На этом инсталляция пакета **.unitypackage** для работы с **Vuforia** в **Unity 3D** завершена. Базовые настройки для работы **Unity 3D** с **Vuforia** выполнены. Переходим к созданию Приложения ДР.

2. Создание Приложения ДР в среде Vuforia + Unity 3D для просмотра видеоклипа.

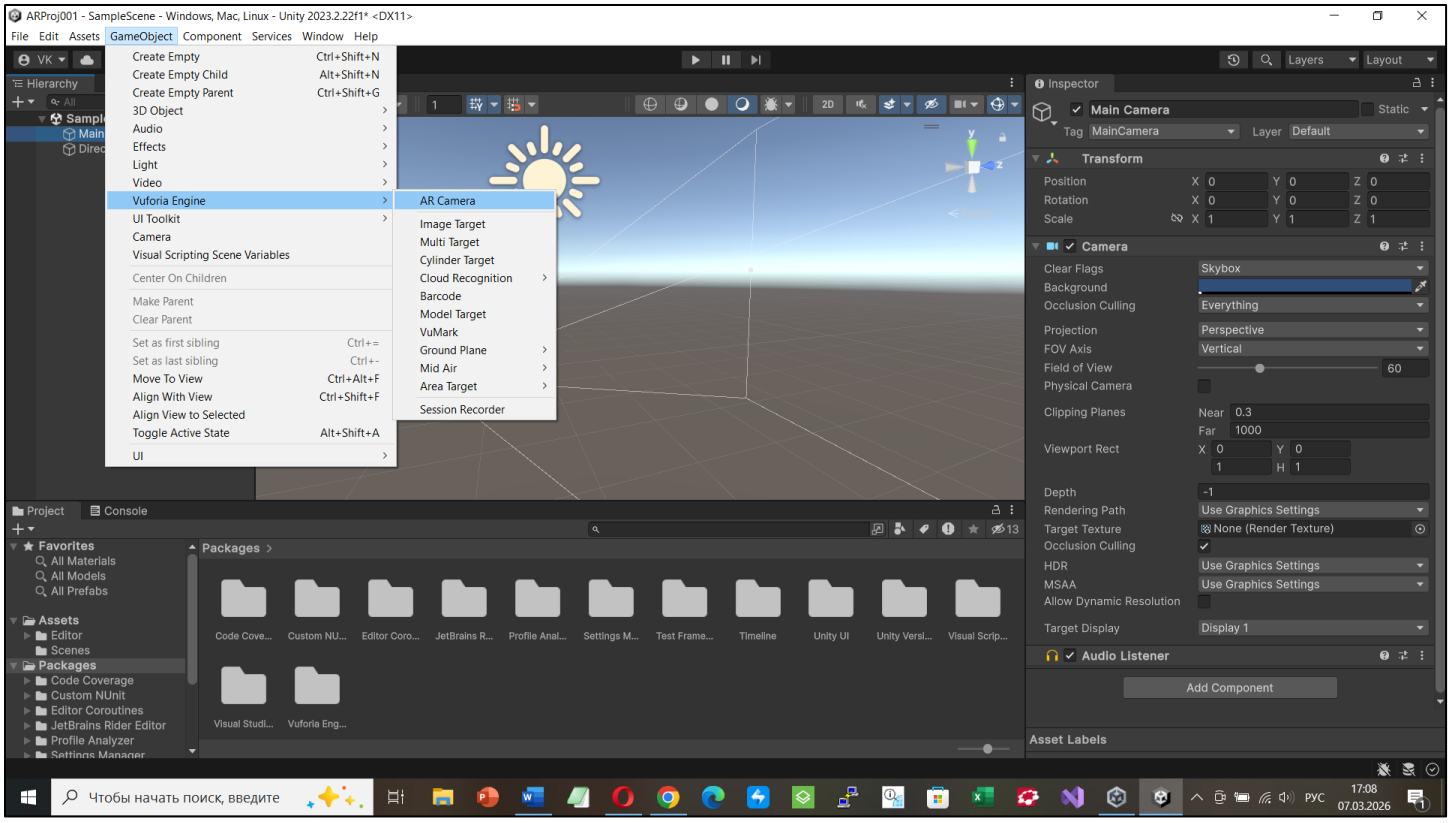
В стандартном режиме разработки Приложений в редакторе **Unity 3D** **Main Camera** не позволяет выполнять визуализацию реального окружения, транслируемого камерой МУ, а работает только с **виртуальными объектами сцены в виртуальном пространстве (режим ВР)**.



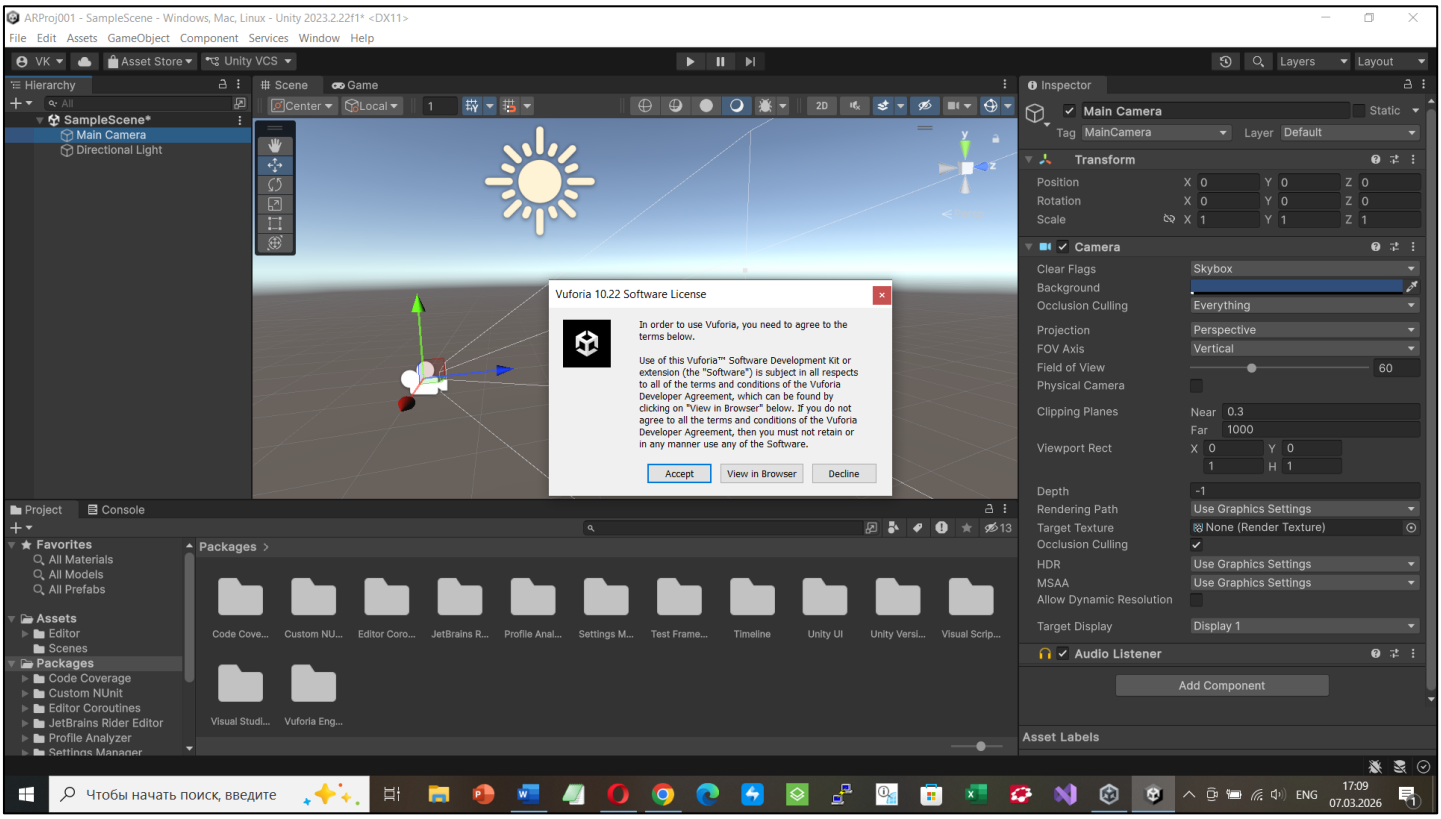
Для работы в другом режиме - **режиме ДР** - необходима «камера», способная выполнять визуализацию **виртуальных** объектов сцены (элементы контента) на фоне **реального окружения**. Для переключения нашего проекта из режима **виртуальной реальности** в режим **Дополненной реальности** необходимо заменить в наборе иерархических объектов (область экрана редактора **Hierarchy**) камеру **Main Camera** на камеру дополненной реальности из набора настроек **Vuforia**.

Для этого выполним следующие вызовы:

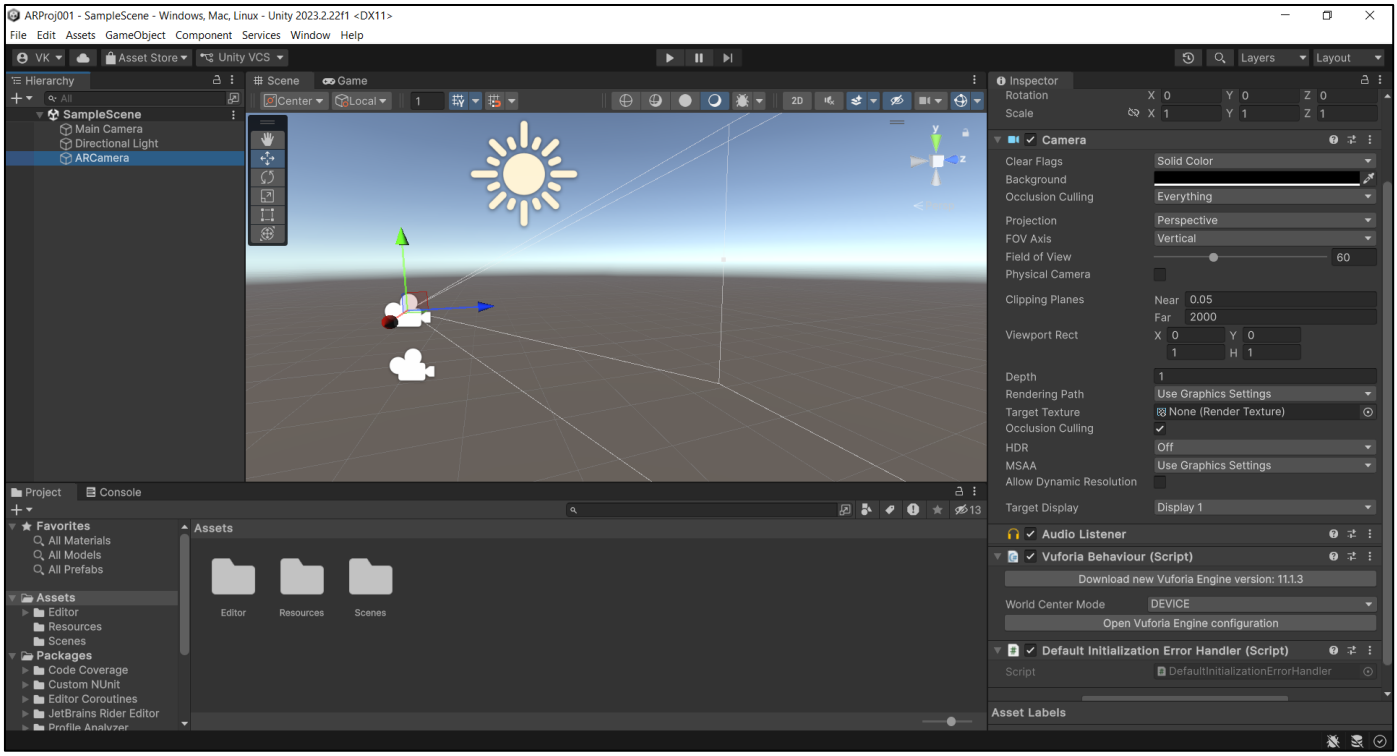
Game Object → Vuforia → AR Camera



При этом у вас может появиться запрос на подтверждение использования **Vuforia Engine**:

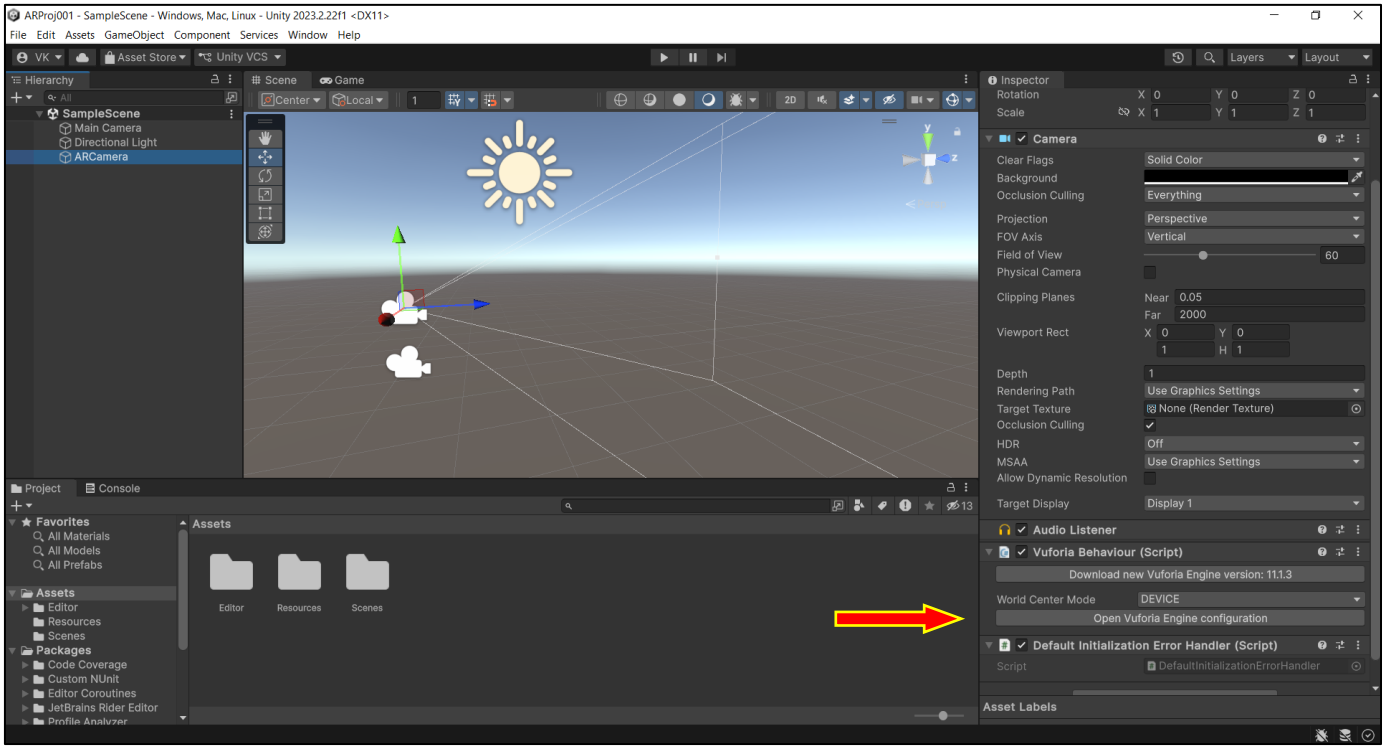


В области **Hierarchy** появляется **Vuforia**–объект **ARCamera.**

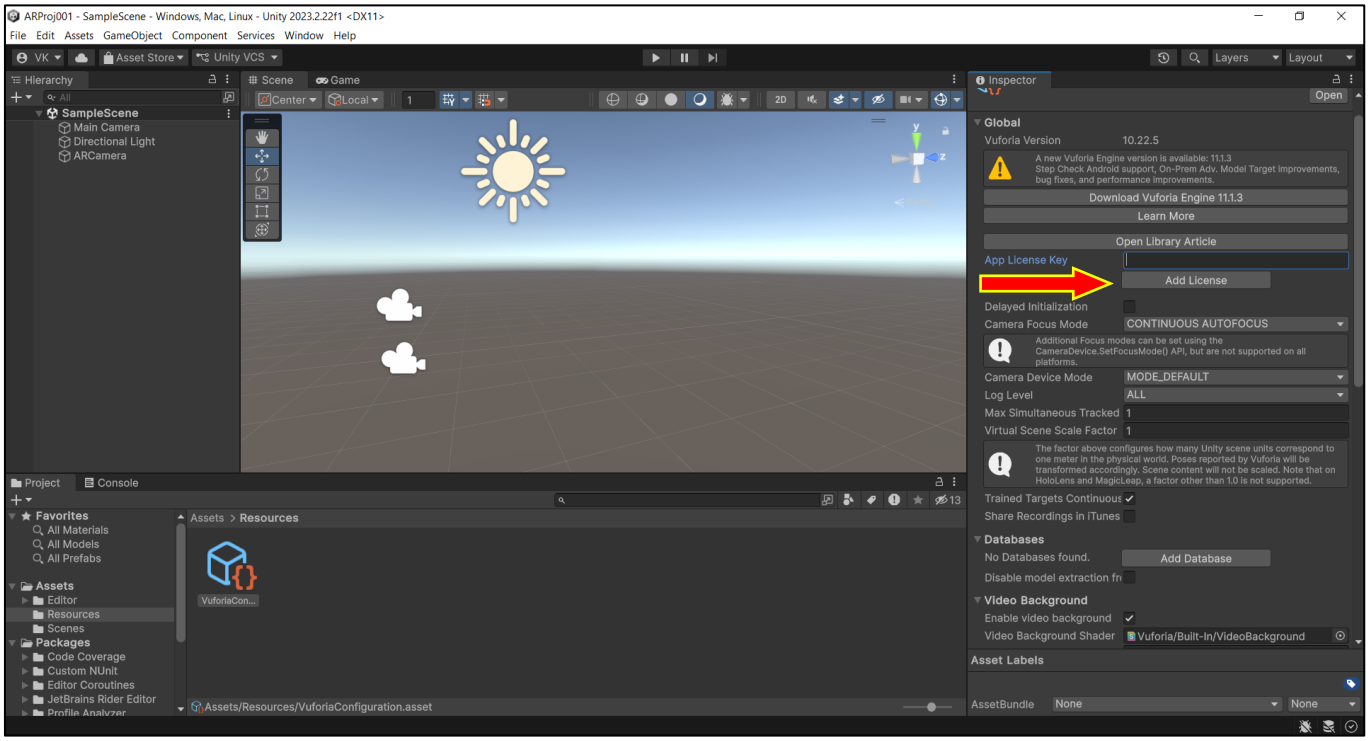


Для работы с объектами **Vuforia** необходимо перевести их под действие сгенерированной вами выше лицензии. Делать мы это будем через верхний уровень объектов **Vuforia**. Установленный объект **ARCamera** является верхним уровнем иерархии всех объектов **Vuforia** разрабатываемой сцены →

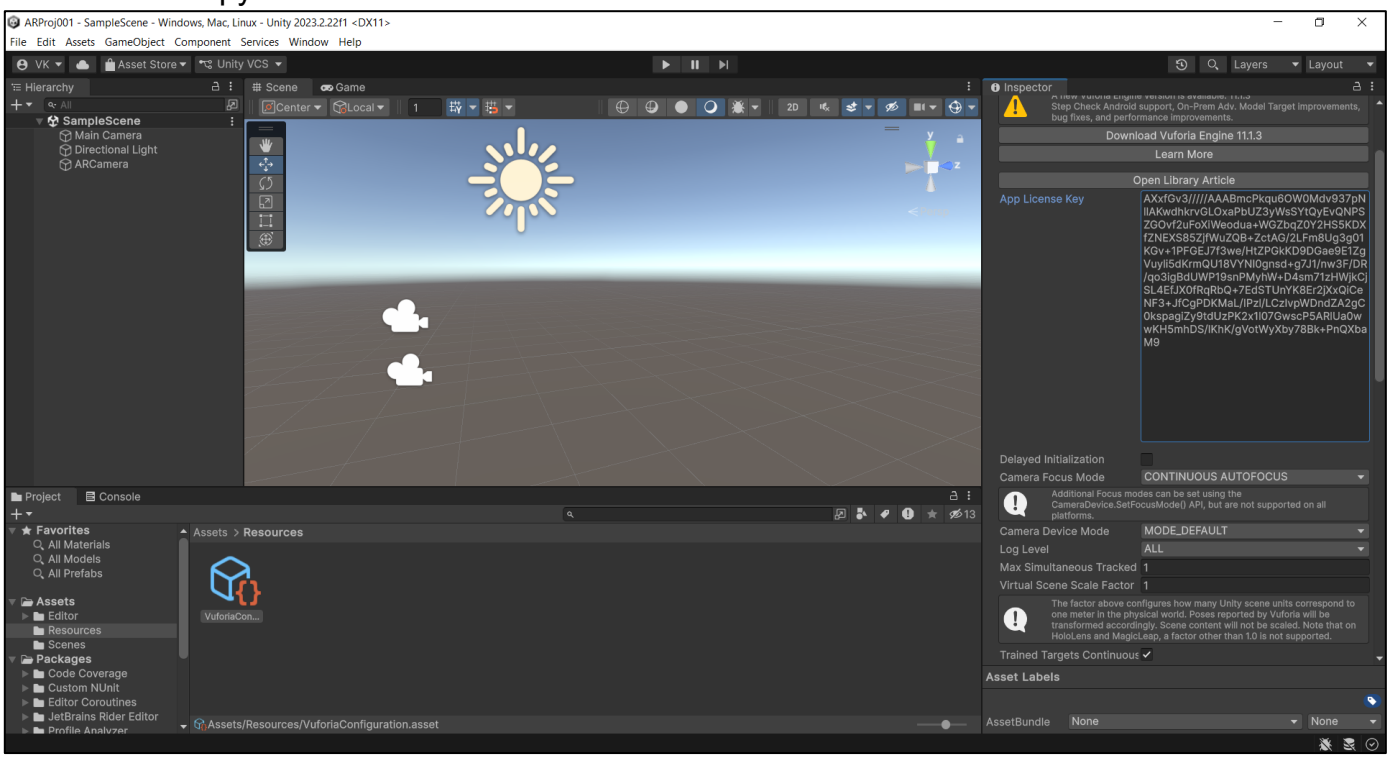
- В области **Hierarchy** выбираем **ARCamera** → в области **Inspector** находим и выбираем поле **Open Vuforia Engine configuration.**



➢ Теперь в **Inspector'е** появилось окно ввода лицензии **App License Key:**



➢ Ранее на локальной машине вы сохранили текст лицензии (**license key**). Находим этот текст и копируем в это поле.

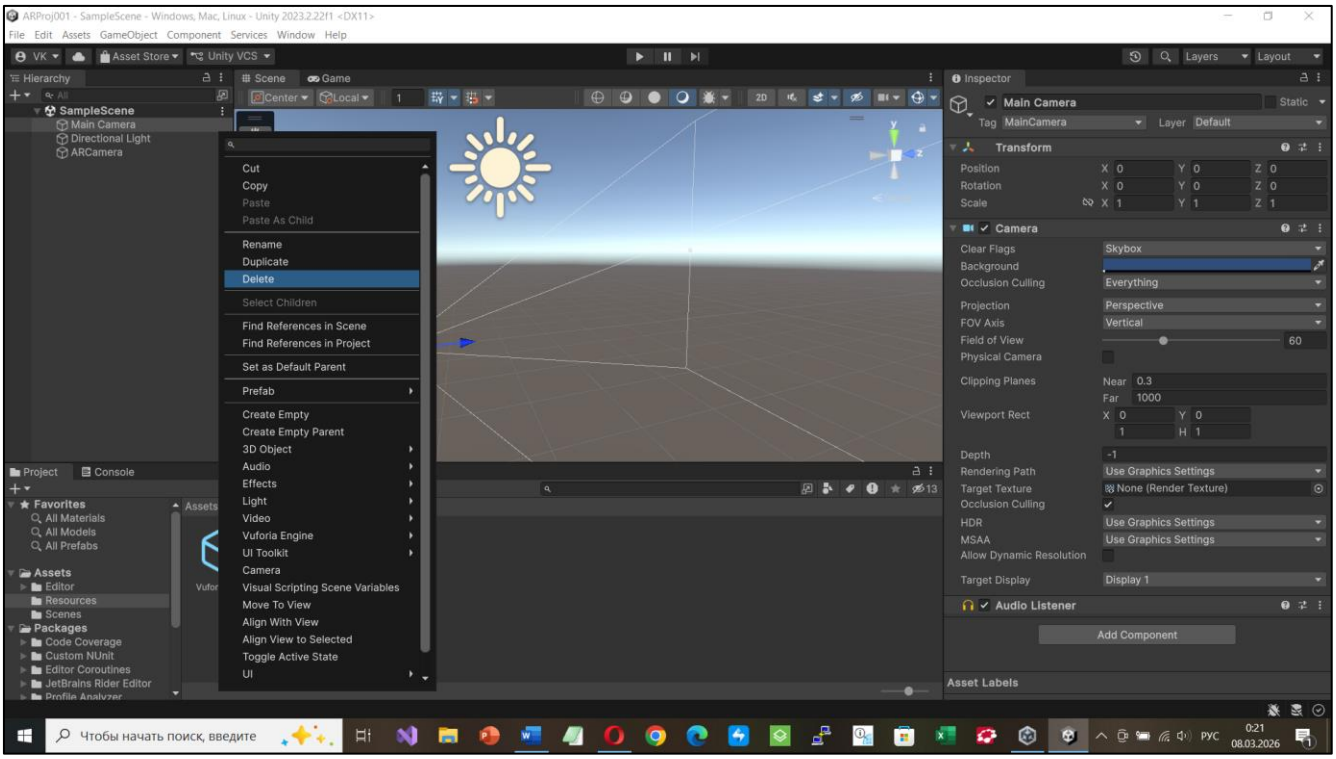


ВАЖНО!! Если Вы выберете на этом этапе поле **Add License**, то вновь попадете на этап ее генерации, который в данной ЛР был выполнен в первую очередь. Иными словами, лицензия автоматом не устанавливается при конфигурировании, ее необходимо скопировать в поле **App License Key**, предварительно сгенерировав.

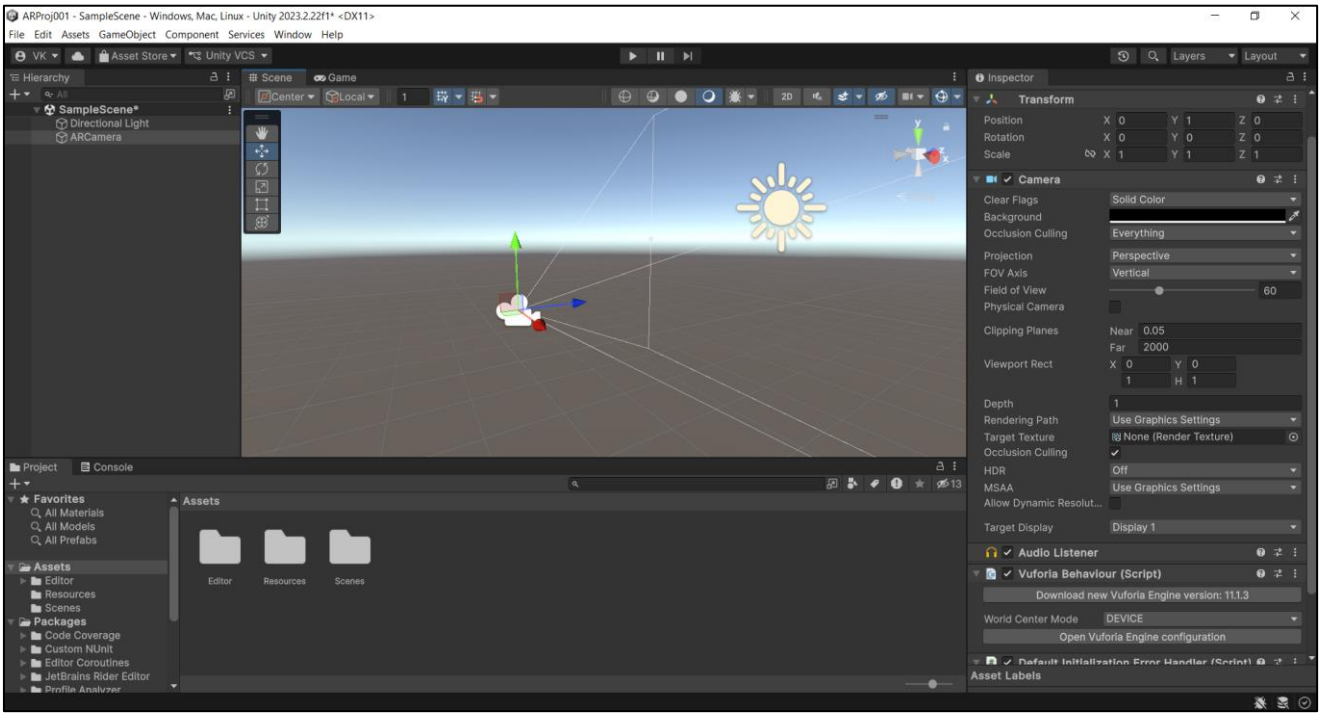
Но если вы забыли это сделать в свое время – здесь у вас есть возможность исправить эту ошибку.

Итак, лицензию вы установили.

Теперь выполняем замену камеры: оставляем **AR Camera** и удаляем **Main Camera**. **Main Camera** должна быть удалена обычным способом – маркируем **Main Camera** в меню иерархии и выбираем в падающем контекстном меню (**RMB**) функцию **Delete**.

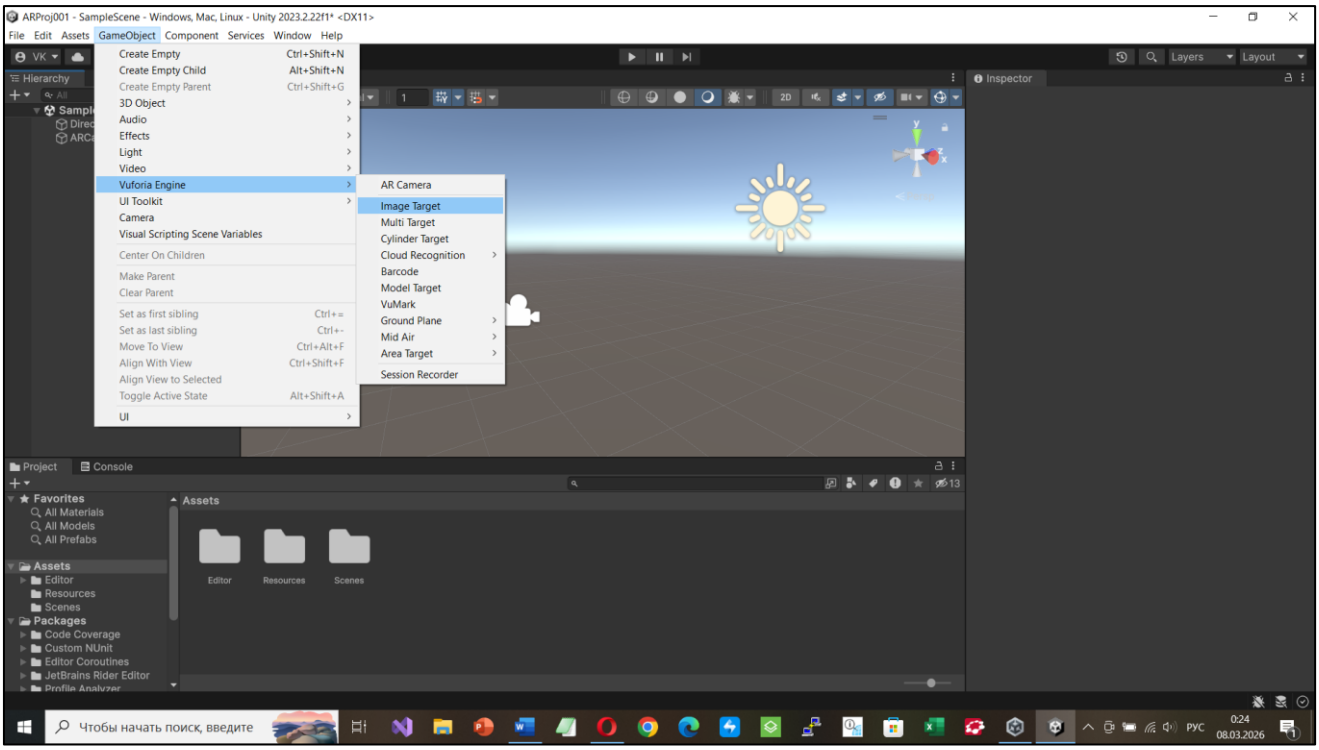


В результате в **Hierarchy** остается только одна камера, свойства которой отображены в области **Inspector**.

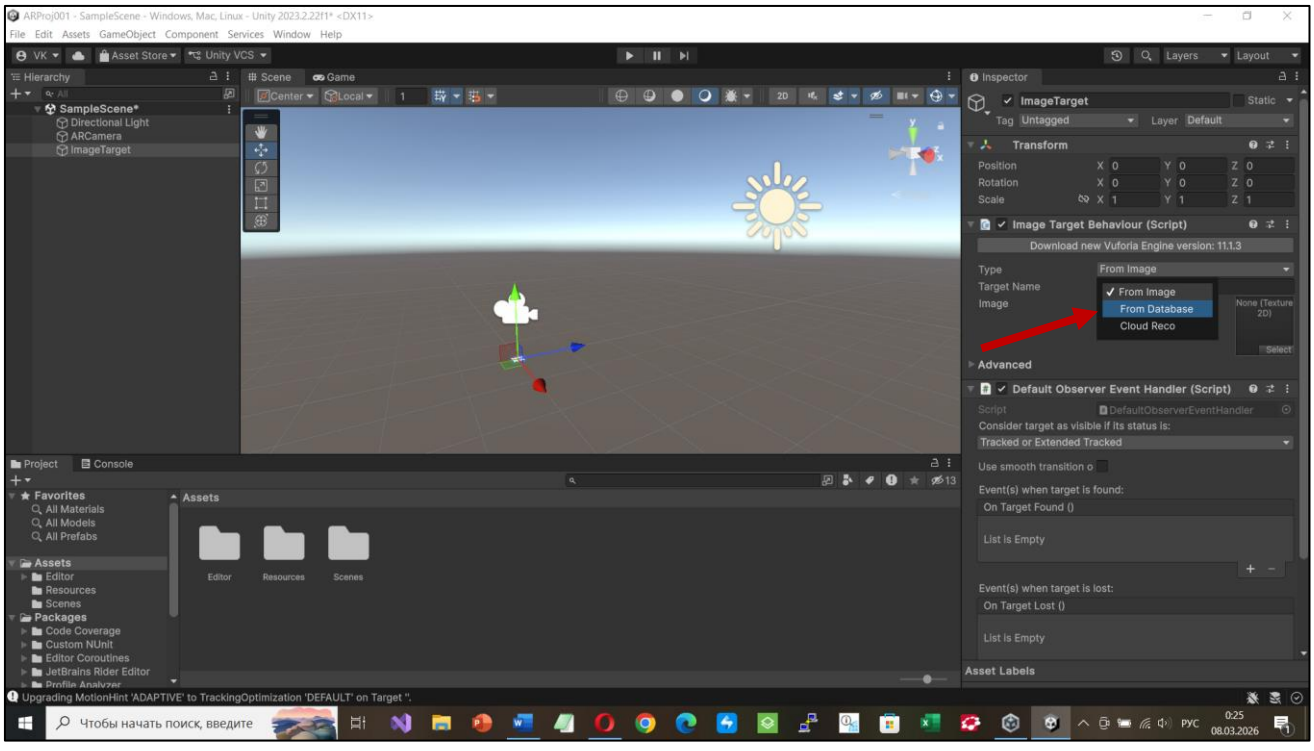


- **Далее необходимо загрузить Базу данных таргетов.**

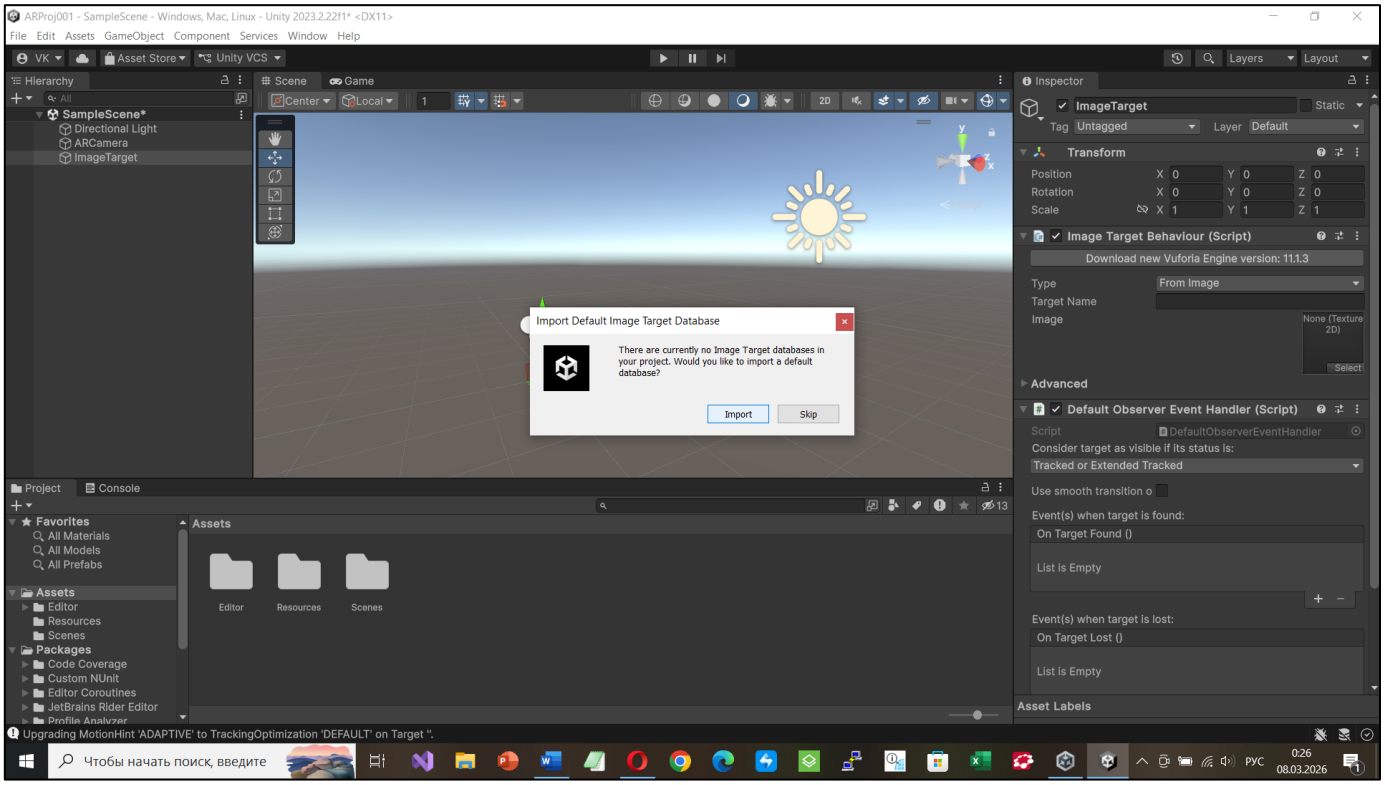
Для этого осуществляем вызовы в закладке **Game Object → Vuforia Engine → Image Target**:



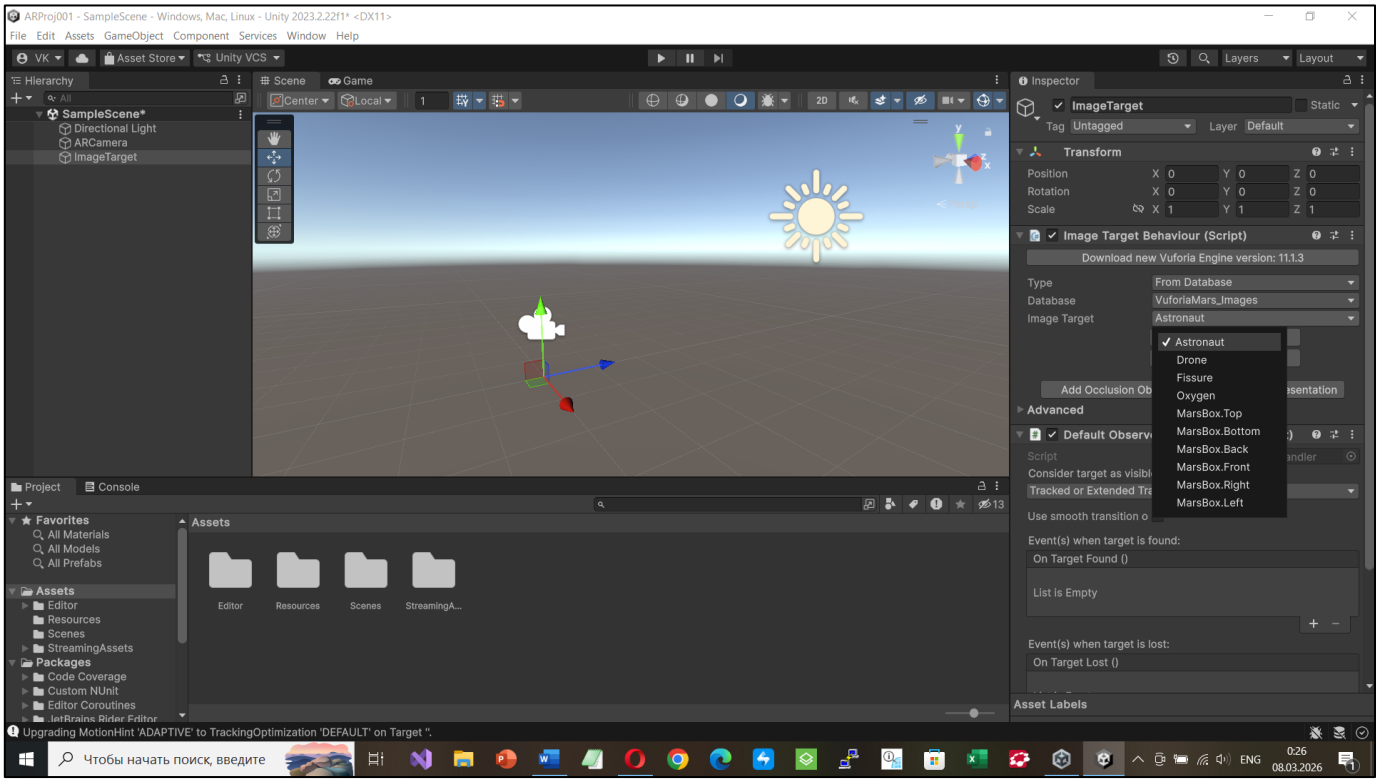
В результате в области **Hierarchy** появляется объект **Image Target**. Выберем его и в области **Inspector** определим источник изображения, которое будет ассоциировано с таргетом → в разделе **Image target Behaviour (Script)**. По умолчанию таким источником является простое изображение – **Image**. Нам же необходимо в качестве источника указать одну из записей сформированной ранее **базы данных таргетов**, сохраненной вами в локальной ФС в виде специфического объекта в формате **.unitypackage**.



- Для этого, в падающем меню (см. рисунок выше) выбираем позицию **From Database.** При попытке работы с базой данных таргетов (особенно при первом обращении) появляется запрос на подключение к какой-либо конкретной базе данных (через структуру, отображаемую обычно в формате **.unitypackage**)

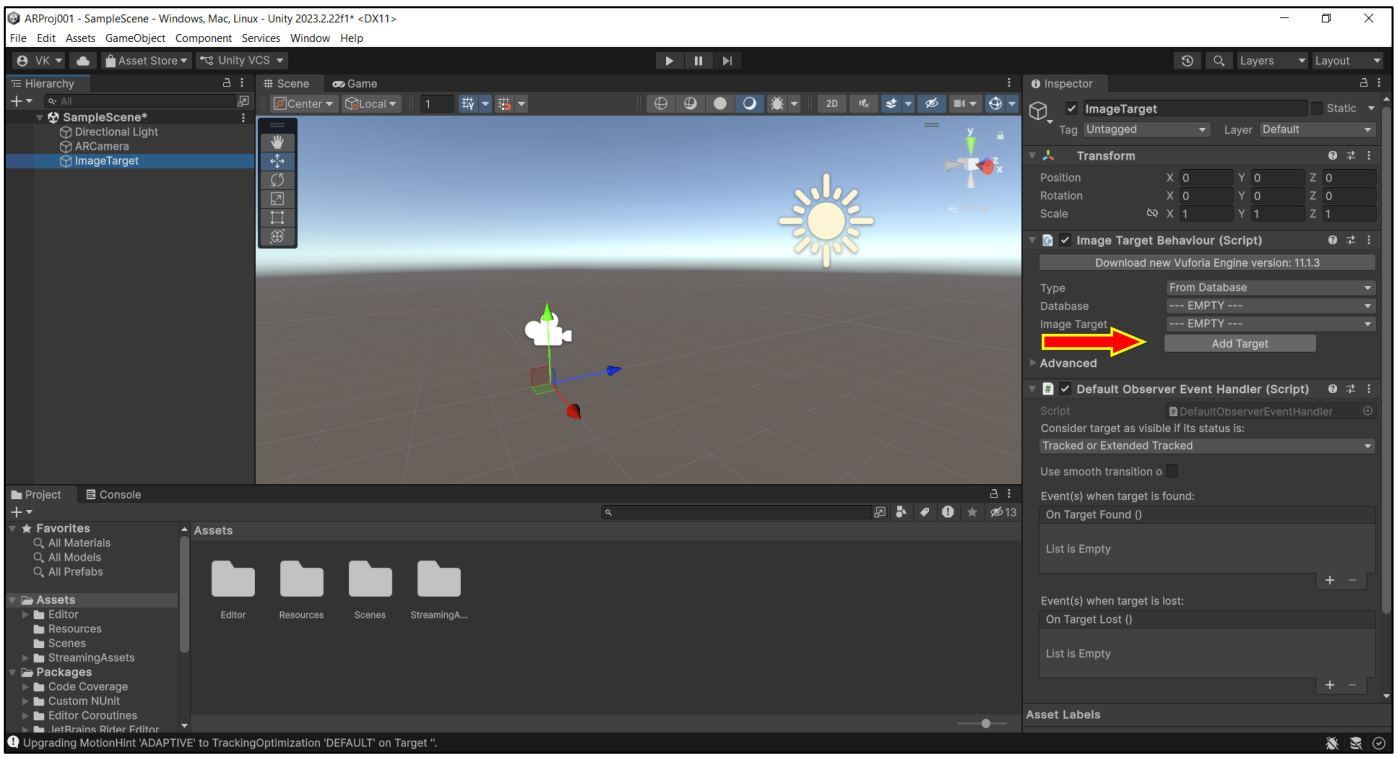


В этом случае выбираем **Import**



➢ В результате в области **Inspector**, в разделе **Image target Behaviour (Script)** появляется не только база данных таргетов образцов **Vuforia Engine** («Марсианская» серия карточек), но также и **возможность** добавить нашу собственную базу данных таргетов → **Add Target**:

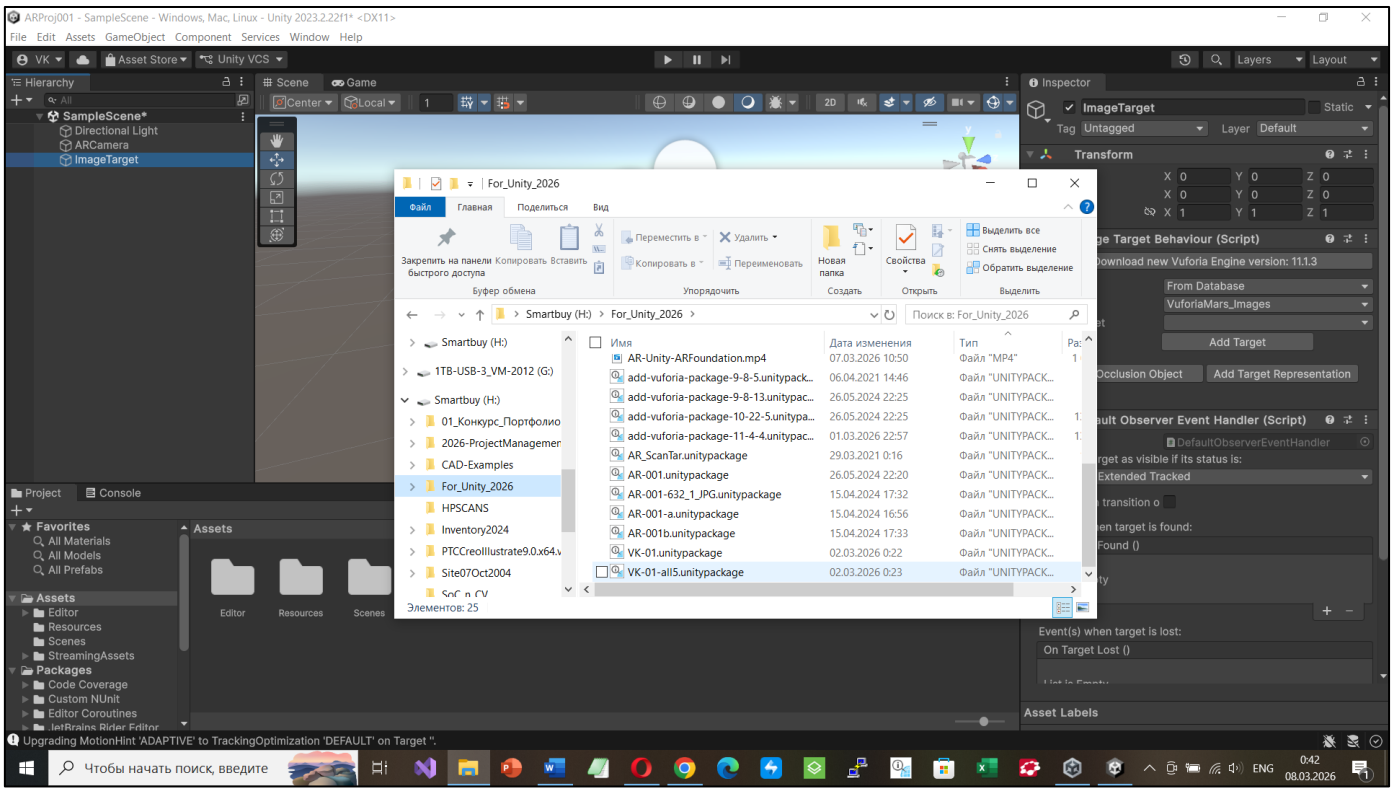
РАССМОТРИМ вариант локальной загрузки подготовленной БД таргетов в формате **.unitypackage** из локальной файловой области



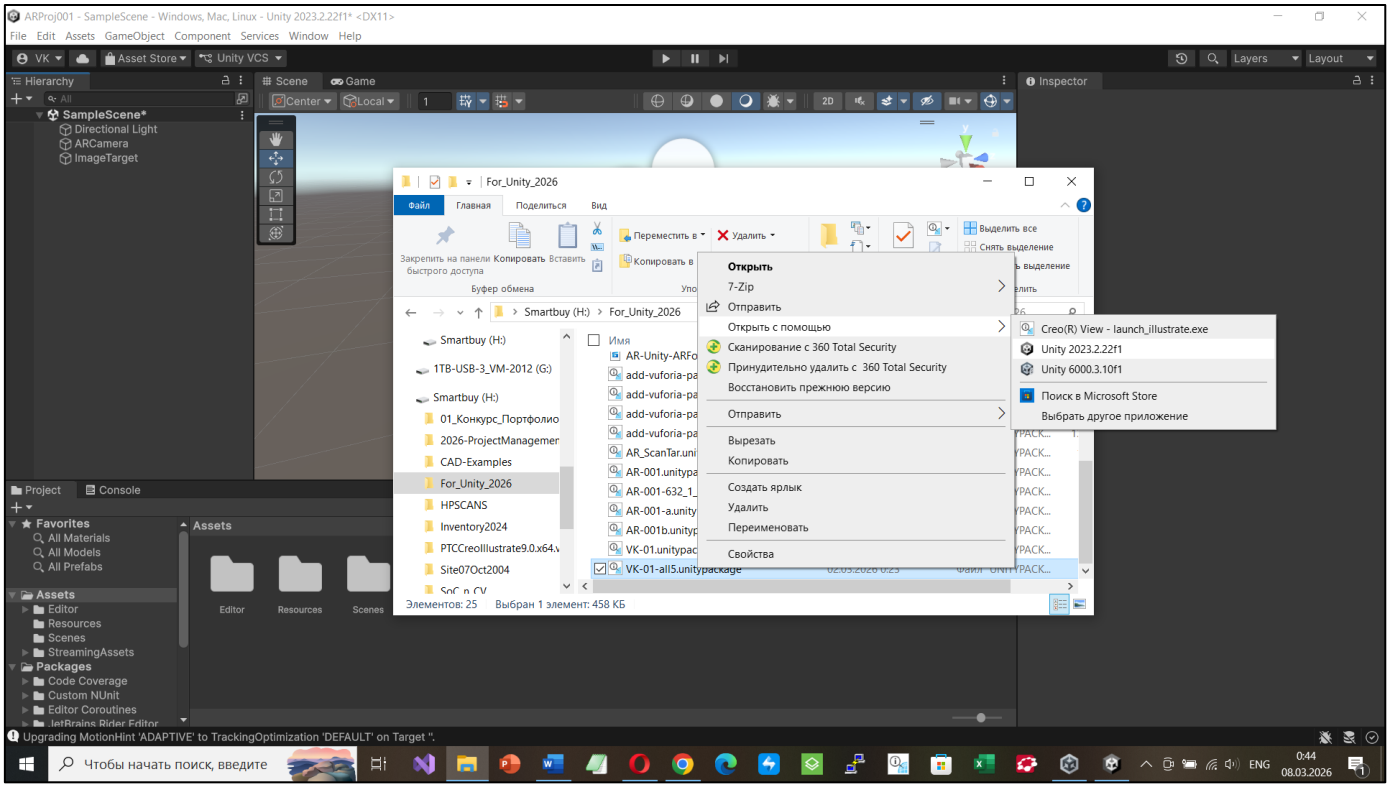
Использование кнопки **Add Target** приводит к обращению к активному порталу https://developer.vuforia.com/ что может быть затруднено. Предлагается использовать другой способ загрузки БД таргетов →

➢ Добавляем подготовленную заранее БД таргетов в сцену нашего проекта.

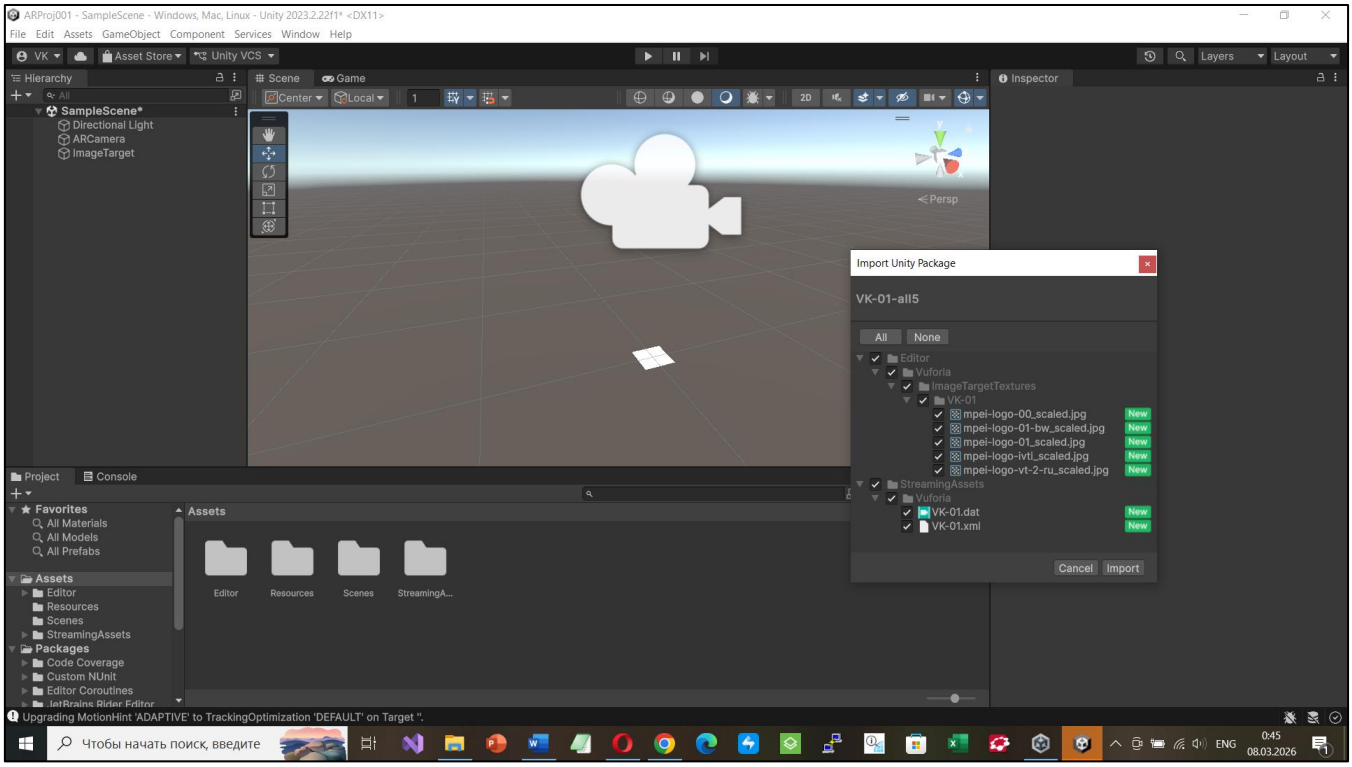
Для этого, не закрывая сессию **Unity Editor**, в локальной файловой структуре находим эту ранее подготовленную БД таргетов,



по правой клавише мыши вызываем контекстное меню работы с БД и выбираем действие, связанное с импортом БД в формате **.unitypackage** в **Unity Editor**:

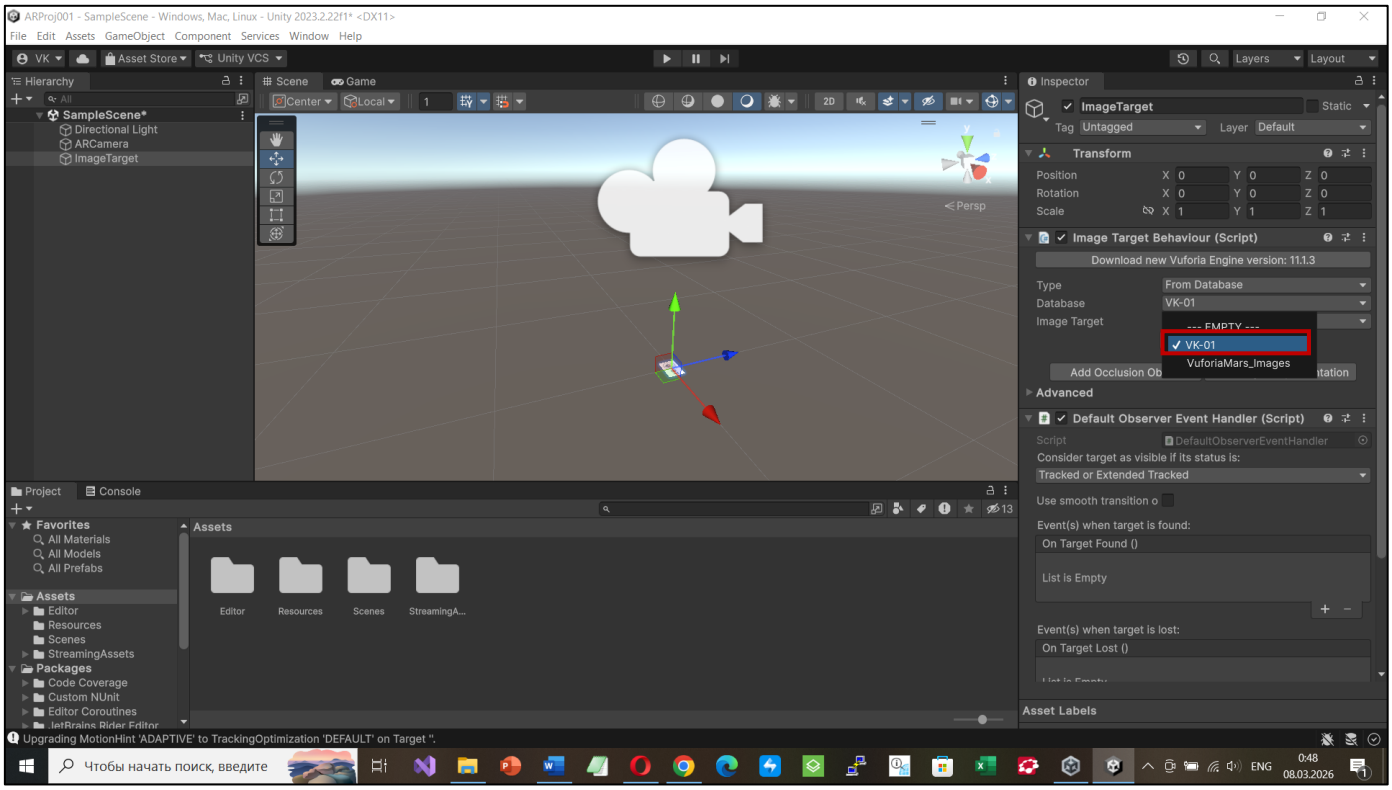


В результате в окне среды **Unity 3D** появляется диалоговое окно импорта БД таргетов:

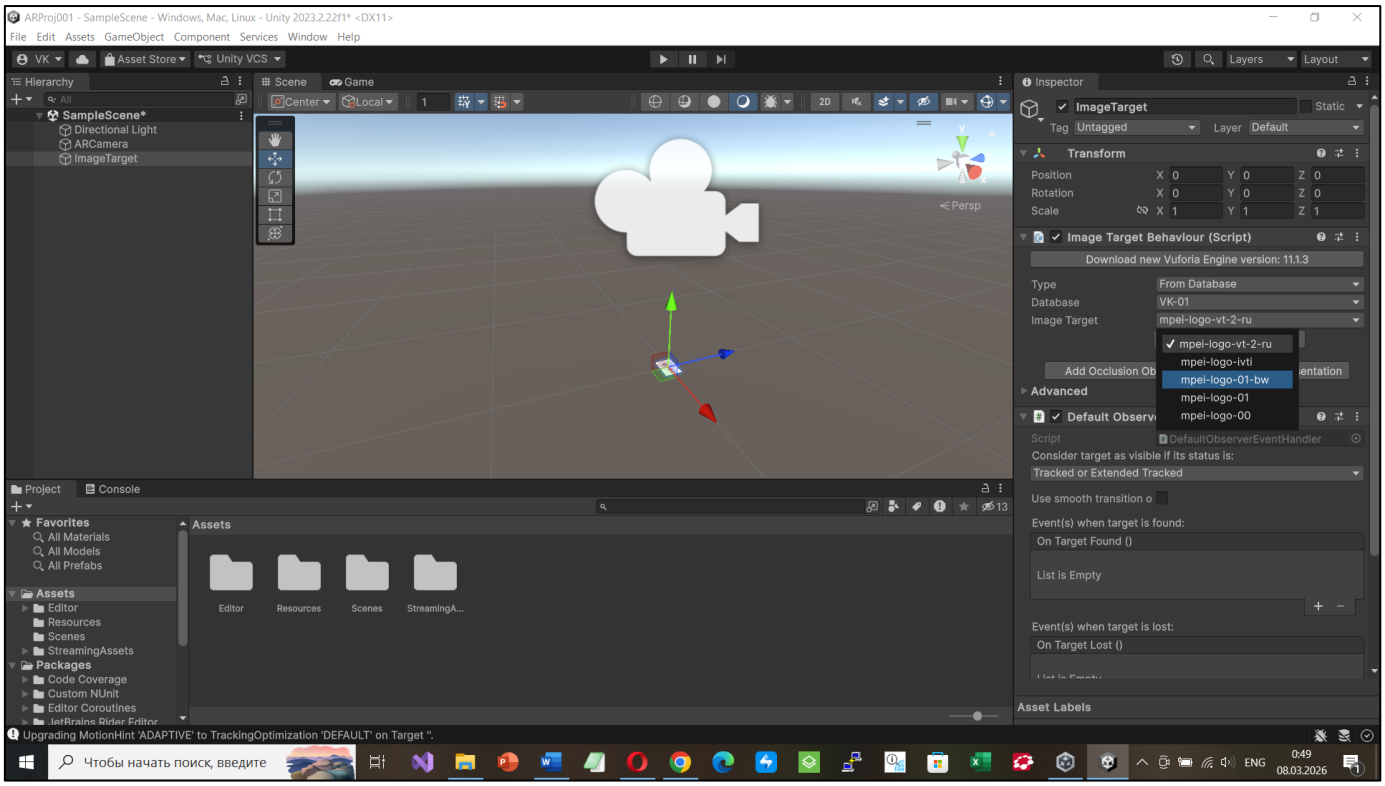


➢ Выполняем импорт нашей БД таргетов **VK-01** путем нажатия клавиши **Import**.

Проверить результат данного действия можно, выбрав в окне **Hierarchy** позицию **ImageTarget**. Далее, в проявившемся окне **Inspector** можно убедиться в наличии БД таргетов проекта, выбрав выпадающий список в области **Image Target Behaviour → Database**:



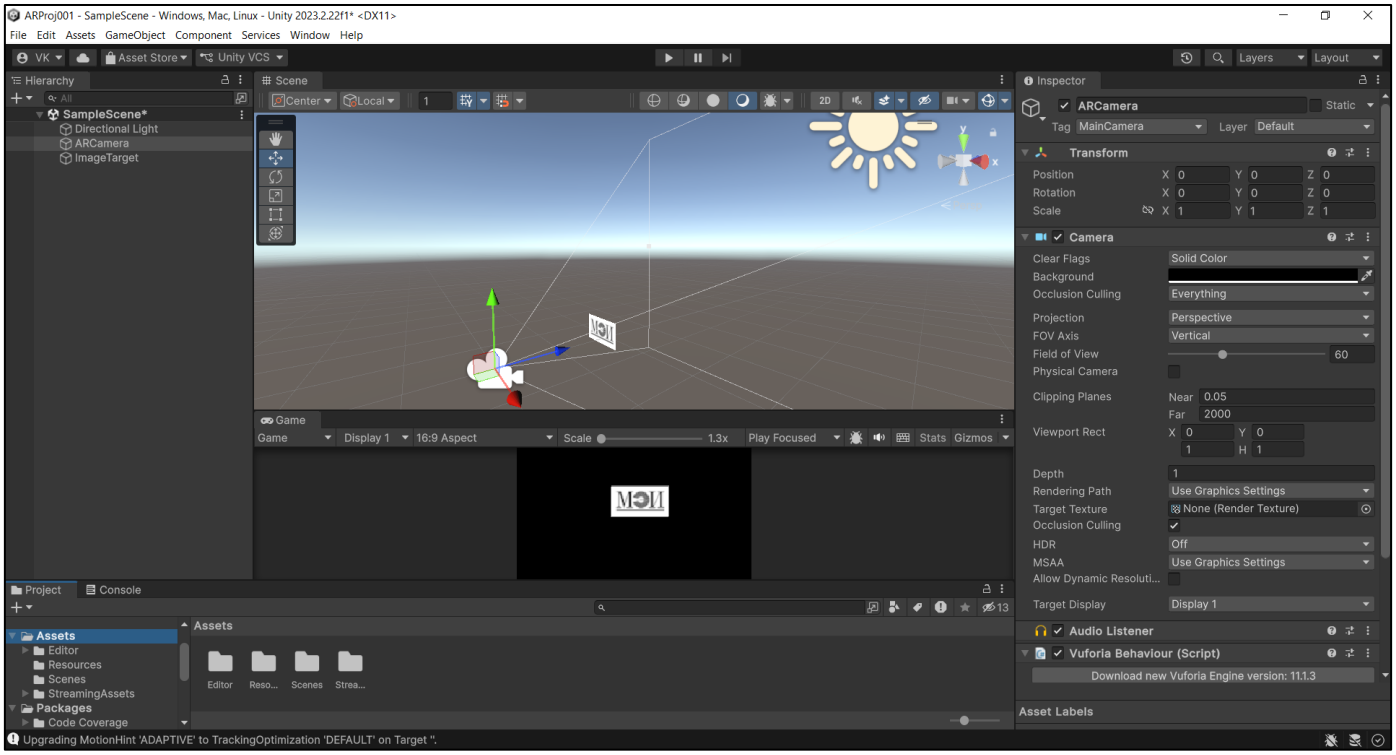
Из БД таргетов **VK-01** выбираем нужный нам **Image Target**, например **mpei-logo-01-bw**:



Теперь в поле сцены имеются два интересующих нас объекта **Vuforia**:

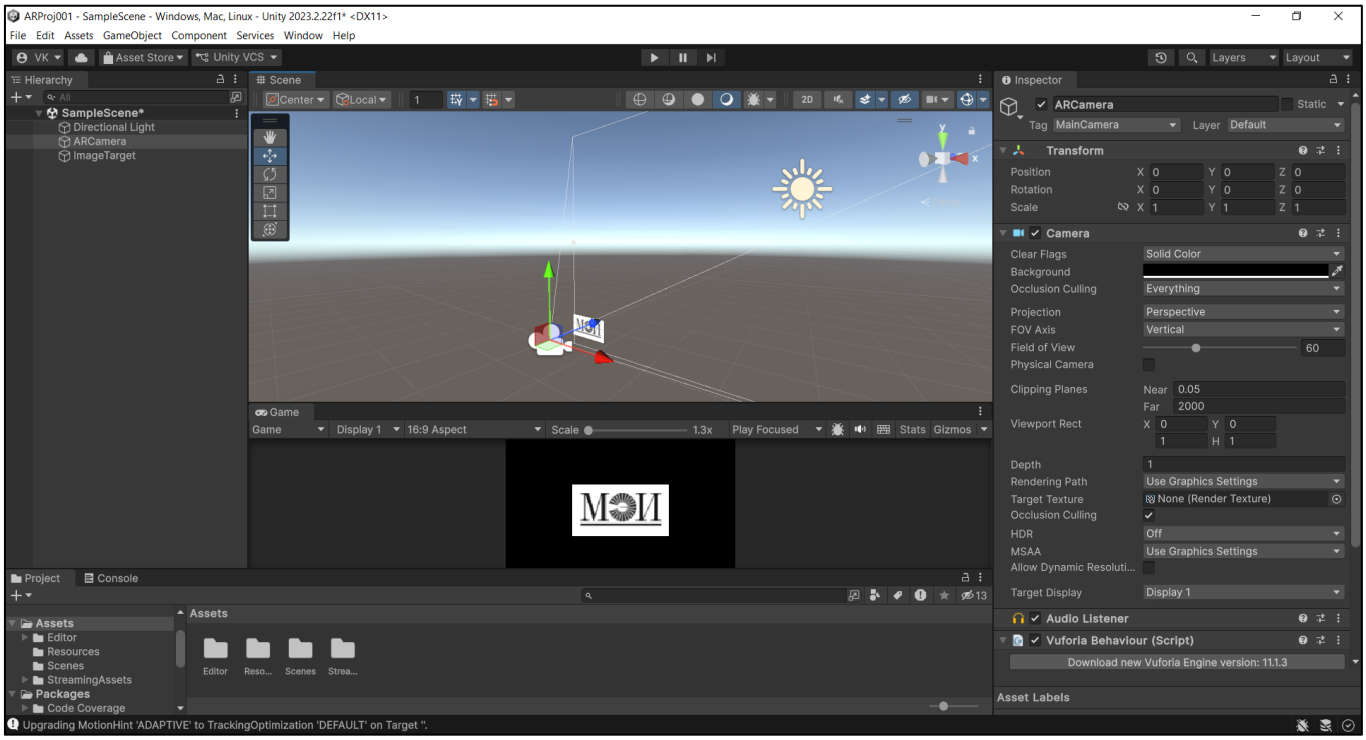
- **AR Camera** и
- **Image Target**.

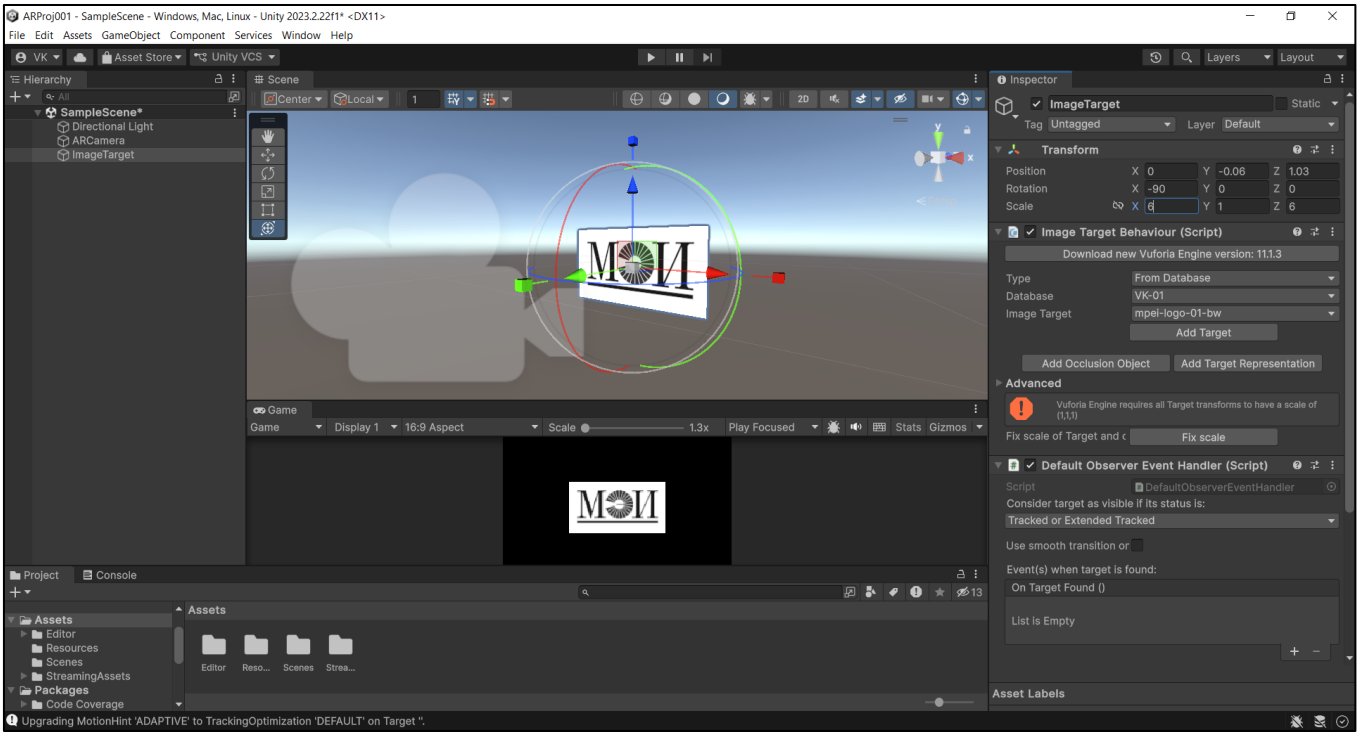
Нам необходимо добиться правильного взаимного расположения камеры и таргета: например, в соответствии с вашим видением сцены, фронтально перед камерой, горизонтально на столе или на полу и пр. Контролировать правильность выполнения этого действия можно в окне предварительного просмотра (выбрать **AR Camera** в области **Hierarchy**) в нижней правой части области **Scene View** – окно **Camera Preview**. Используйте ручки **Transform (Position, Rotation, Scale)** в области **Inspector.** Удобнее всего воспользоваться инструментом **Gismo**  , который размещен в области **Scene View** и позволяет визуально изменять взаимное расположение камеры и таргета, наблюдая результат наиболее удобным способом. Объекты для манипуляций - **AR Camera** и **Image Target** - выбираются в **Hierarchy**. Рекомендуется не менять положение камеры (оставьте ее в положении начала координат - 0,0,0 по трем осям), а манипулировать таргетом (вращать, поднимать, отодвигать от камеры, масштабировать), при этом контролировать сцену, одновременно наблюдая положение объектов в закладке **Scena** и видимый результат в закладке **Game**. В состоянии «по умолчанию» закладка **Scene** перекрывает закладку **Game,** для удобства контроля работы со сценой вы можете разнести эти закладки по разным частям основного окна **Unity Editor**.



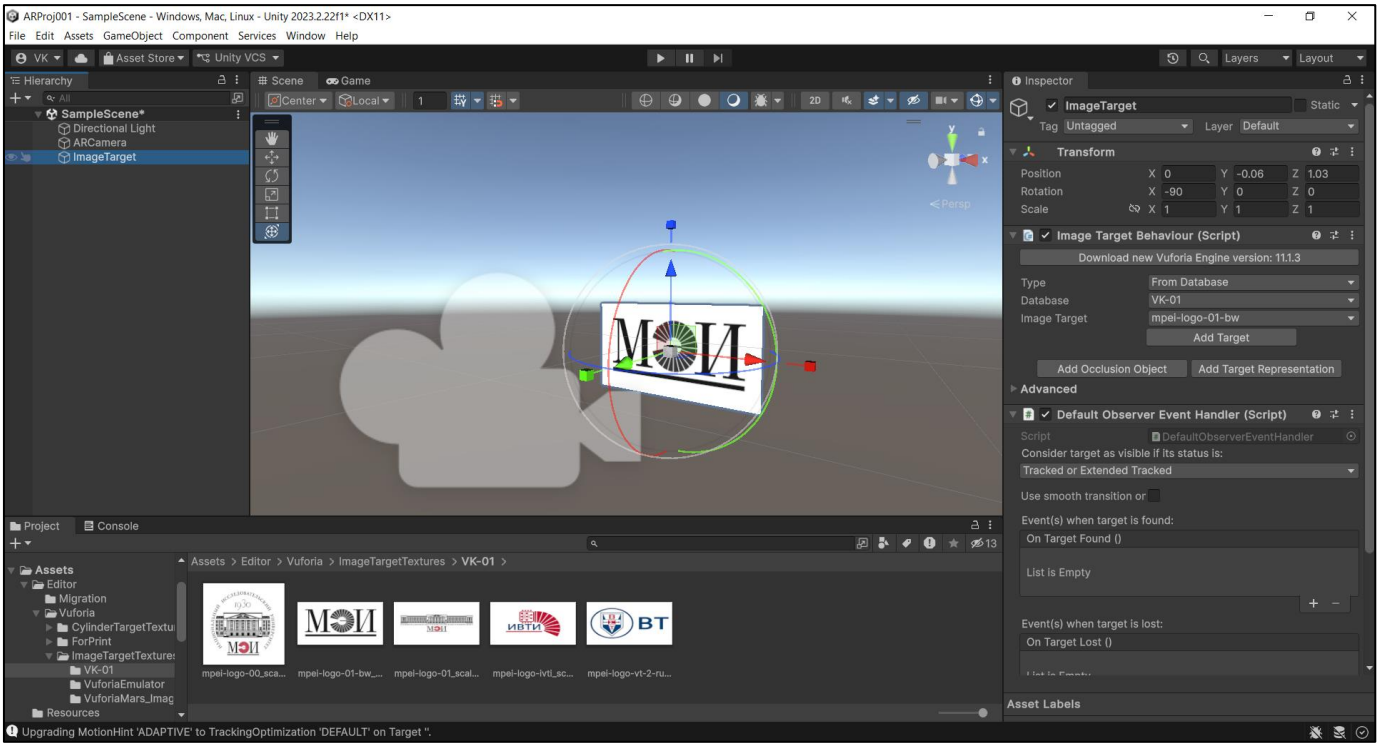
Хорошим результатом начального размещения таргета на сцене считается такой, при котором изображение **Image Target** занимает от 30% до 60% окна **Game**. Добейтесь этого.

Обратите внимание на положение камеры и таргета в **Hierarchy!! Image Target** и **AR Camera** в процессе этих манипуляций находятся на одном уровне. После достижения требуемого результата объект **Image Target** в **Hierarchy** должен занять подчиненное по отношению к **AR Camera** положение – выполняется т.н. **Parenting** (перетаскиванием одного объекта на другой), установление связей **«Родитель-потомок»**. Т.о. закрепляется связка камера-таргет.





По мере выполнения всех этих действий следите за изменением наполнения области **Project** редактора **Unity 3D** и содержанием появляющихся там папок, в том числе и тех, которые имеют отношение к нашему проекту:



Теперь надо разместить контент, связанный с меткой, в сцене ДР.

В нашем случае метка (таргет), объект **Image Target**, уже размещен в сцене.

Контент в данной ЛР – это подготовленный файл, видеоклип – **AR-001Video.mp4**. Находится в файловой структуре на локальной машине.

Фактически в проектируемой сцене мы хотим разместить виртуальный экран, на котором будет демонстрироваться видеоклип **AR-001Video.mp4**. Демонстрация начинается только после того, как на **Мобильном Устройстве (МУ)** с установленным на нем **Приложением ДР**, происходит распознавание физического объекта – в нашем случае плоского изображения - **Image Target,** попавшего в обзор видеокамеры МУ. На данном этапе мы будем заниматься размещением наилучшим образом в **3D**-пространстве сцены виртуального **2D**-экрана и выводом на этот экран элемента контента ДР – видеоклипа (**AR-001Video.mp4**).

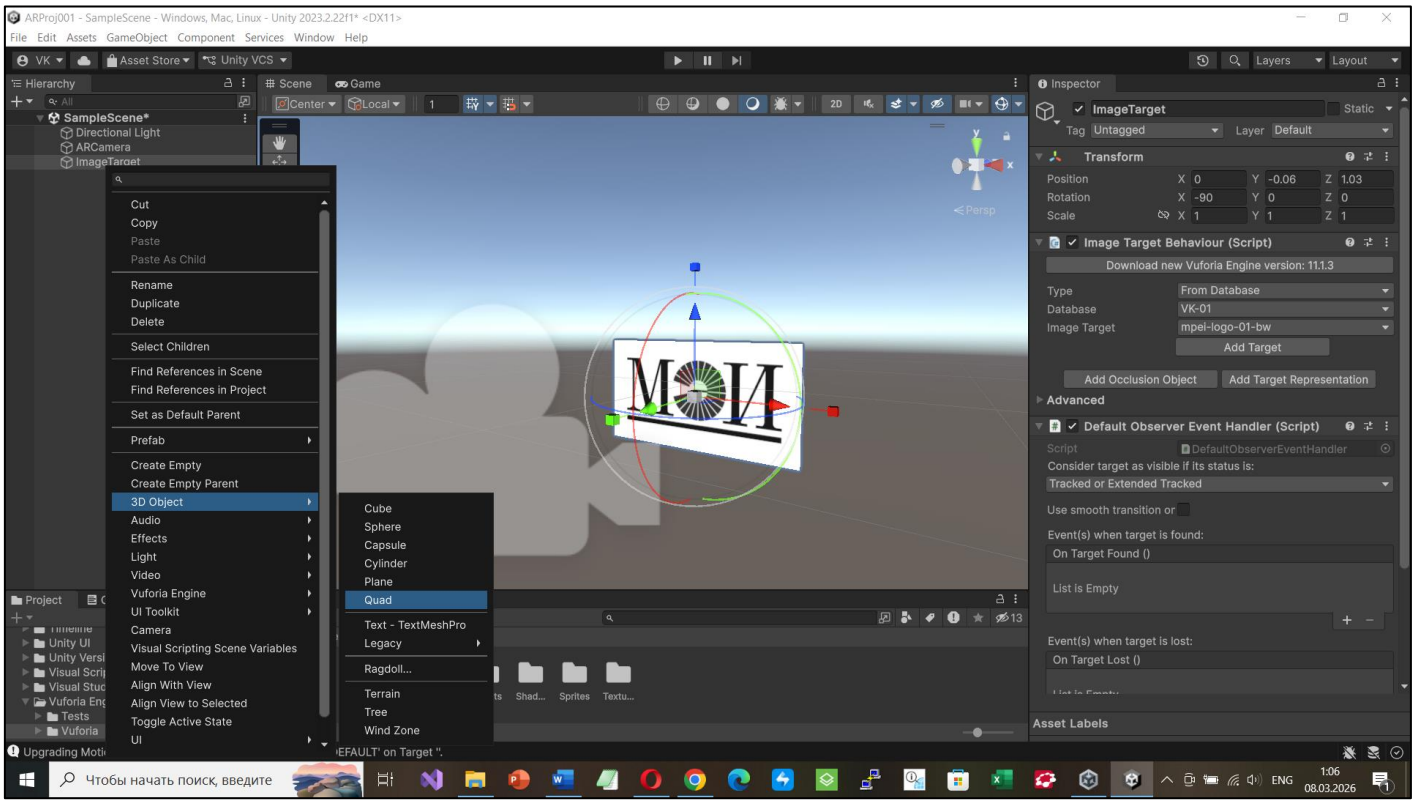
Для размещения контента в сцене **Unity 3D** воспользуемся функционалом редактора. Для этого в системе существует большое количество собственных объектов. Эти объекты можно увидеть в основной панели меню **Unity 3D** в группе **Game Object**. Один из них, чаще всего используемый для размещения **2D**-контента в трехмерном пространстве, – **Quad**.

Т.о. в **Unity 3D** связываем **Image Target** с размещаемым контентом через объект (шаблон, контейнер) **Quad.** Для такого связывания в иерархии (области **Hierarchy**) **Quad** доложен располагаться под **Image Target**.

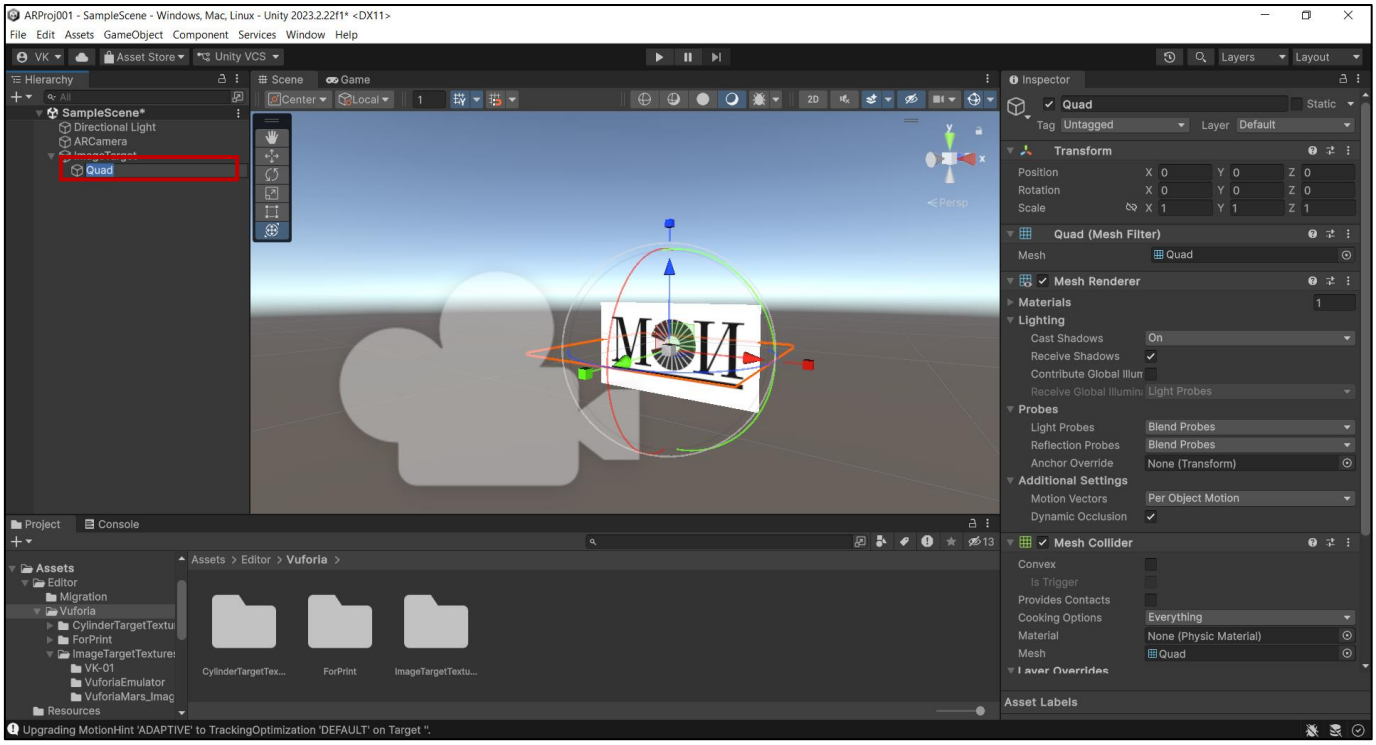
Последовательность вызовов для получения результата:

- В области **Hierarchy** выбираем **Image Target;**
- По правой клавише мыши в выпадающем меню находим строчку **3D Object;**
- В связанном с ней списке альтернатив выбираем **Quad.**

Unity 3D: 3D Object → Quad, контейнер **2D-объекта**:



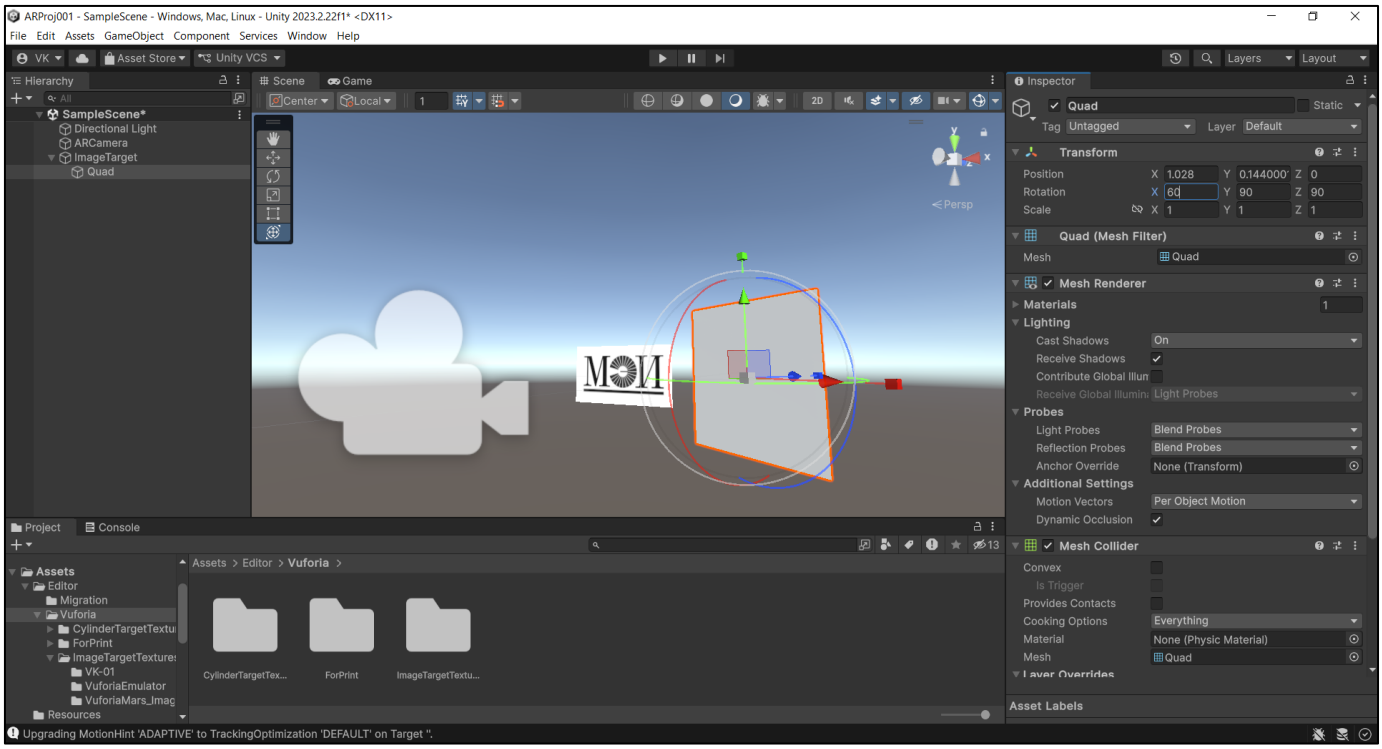
В результате в сцене появляется новый объект **Quad**, а в **Inspector'е** – информация о нем:

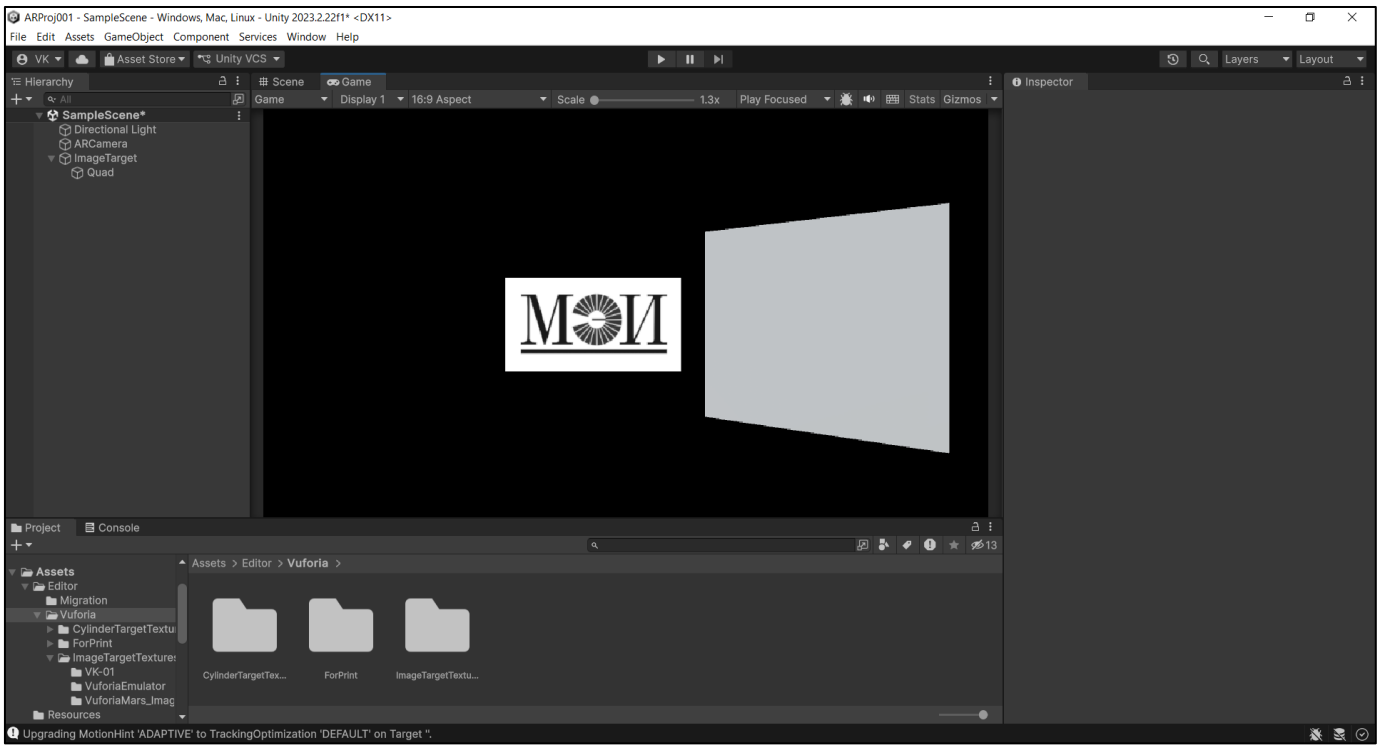


Для удобства работы со сценой можно воспользоваться привычными функциями средней клавиши мыши (уменьшить/увеличить).

Теперь в сцене видны все объекты контента.

Не меняя расположения камеры и таргета, найдем наиболее удобное положение для контейнера видеоклипа – **Quad'а**. Например, экран с видео (**Quad** с видео) будет расположен на таком же расстоянии от камеры, что и таргет, но с небольшим разворотом по вертикали (**ось «Y»**). Одновременно уменьшим и размеры экрана. Вы можете выбрать расположение самостоятельно (с помощью инструмента **Gismo** или ручек **Transform** в **Inspector'е**).

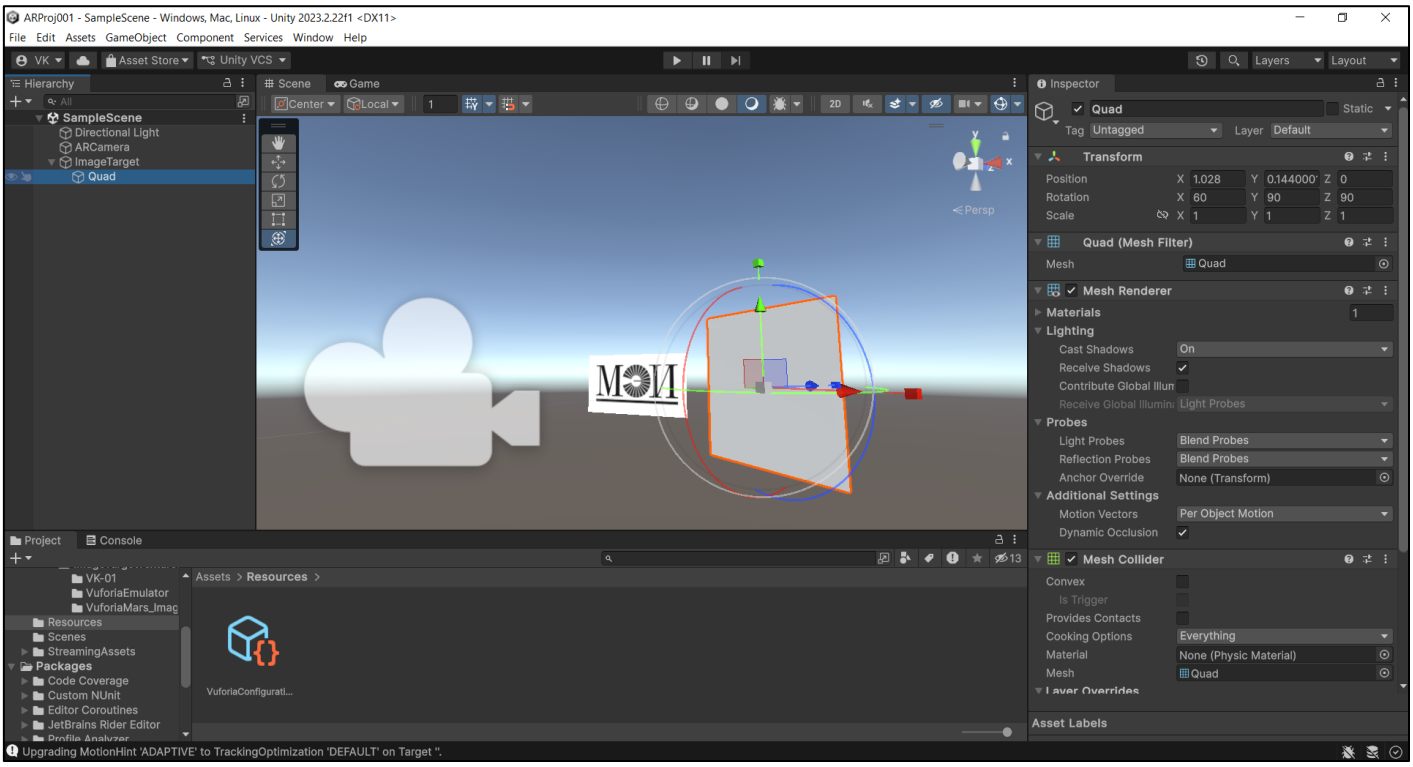




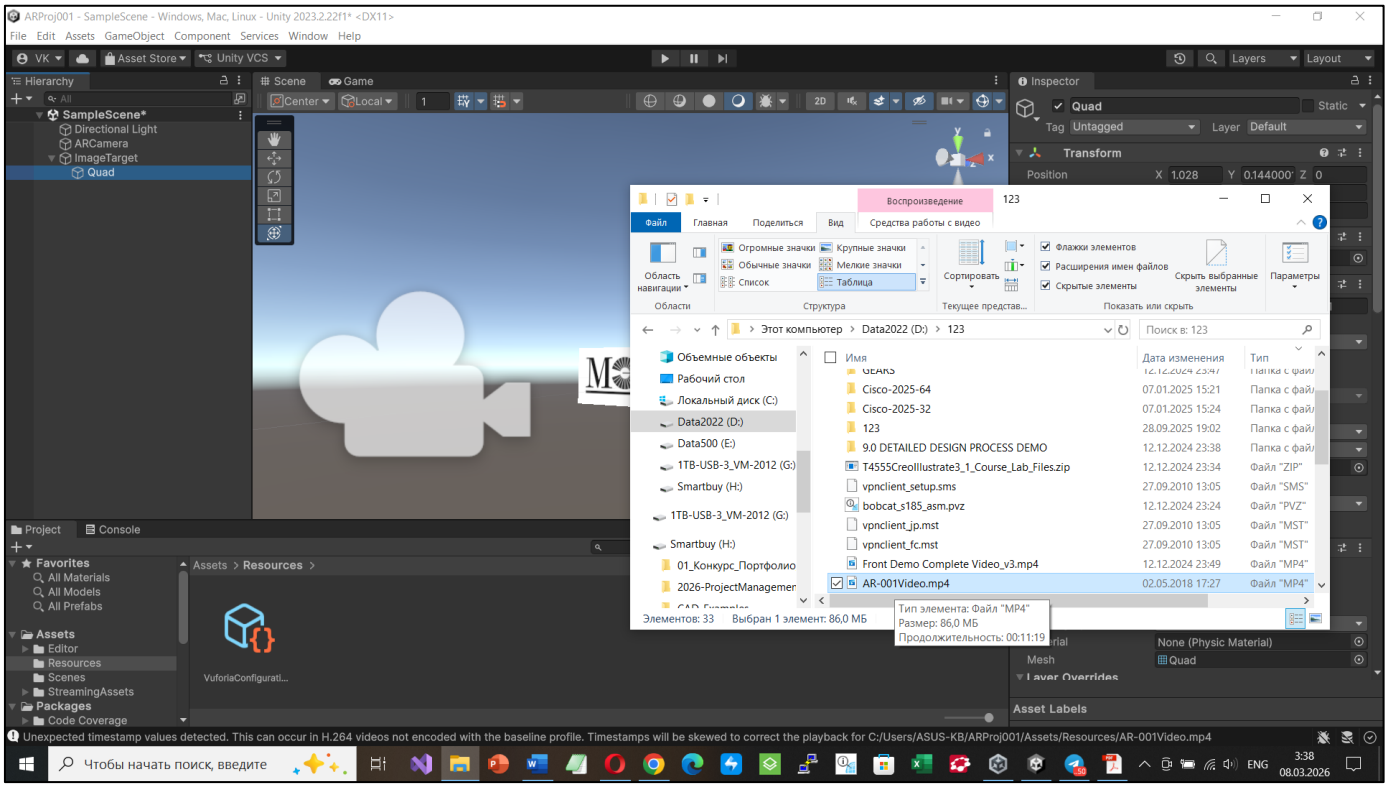
Рекомендуем манипулировать только **Quad**'ом, а **ARCamera** использовать для контроля результата в окне **PreView**. Если вы смотрите на экран фронтально, он не прозрачен!!

В результате сцену можно считать предварительно сформированной – в ней размещены **ARCamera**, таргет и контейнер для **2D**-контента – видеоклипа.

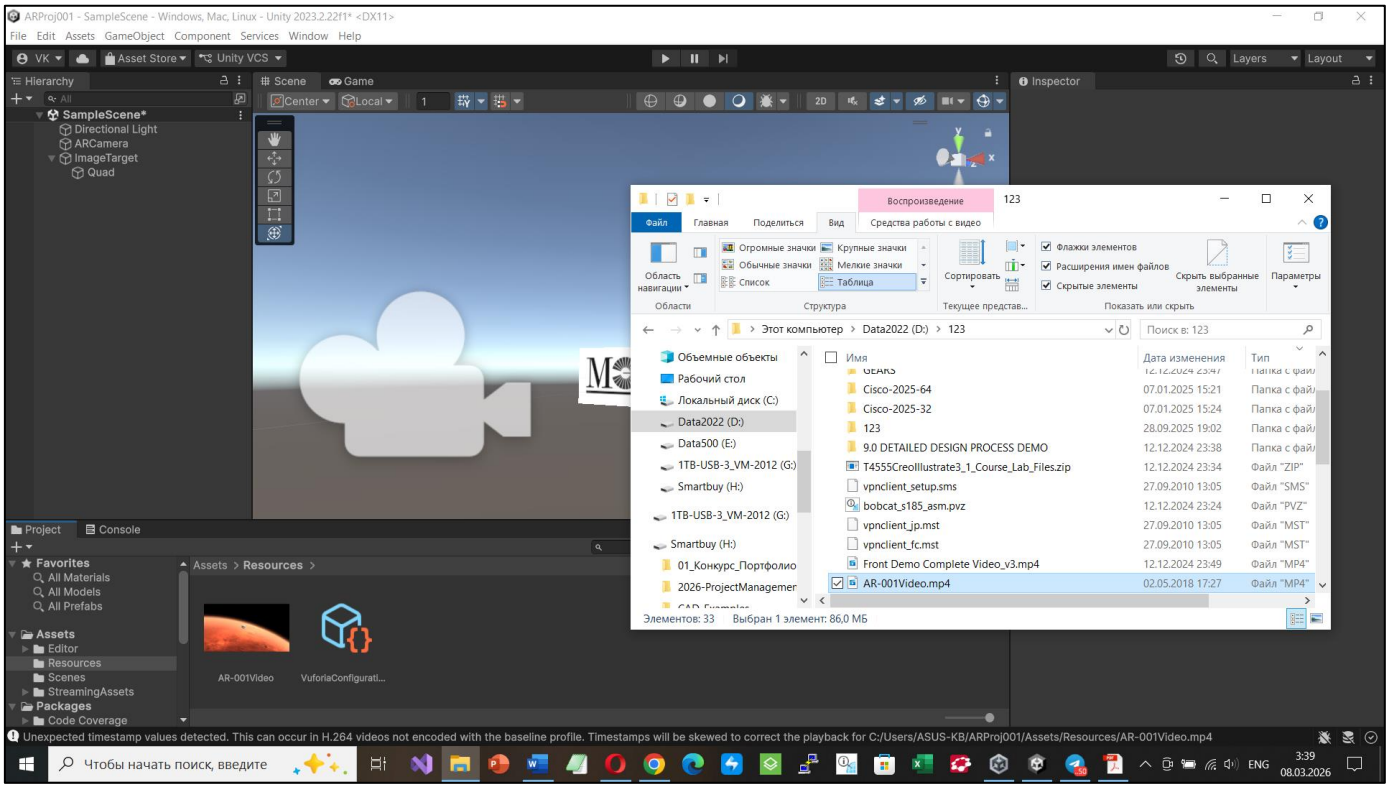
ВАЖНО!! Элементы контента такого типа (пользовательские видеоклипы, изображения и пр.) нашего Приложения ДР при работе с **Unity** должны быть помещены в раздел ресурсы (**Resources**) библиотеки активов (**Assets**). Пока этот раздел пустой:



В локальной ФС находится заранее подготовленный файл в формате **.mp4** для использования его в Приложении ДР - **AR-001Video.mp4**:

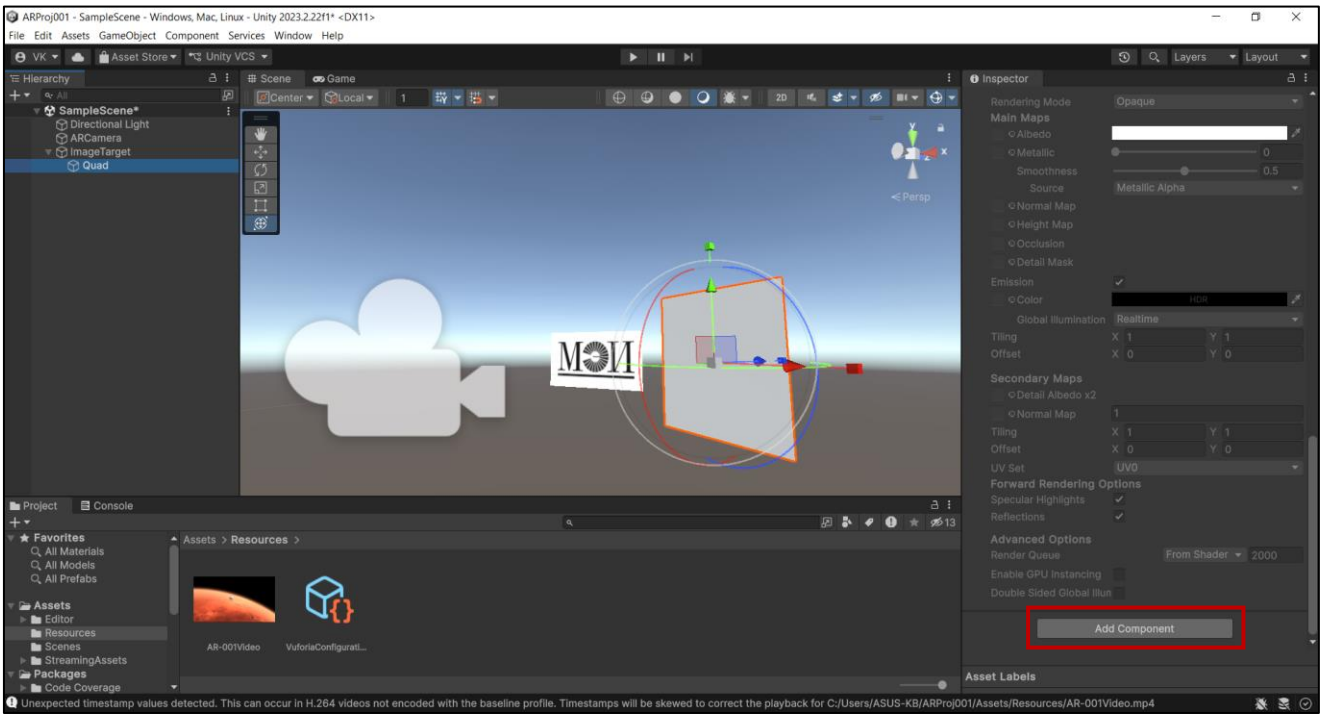


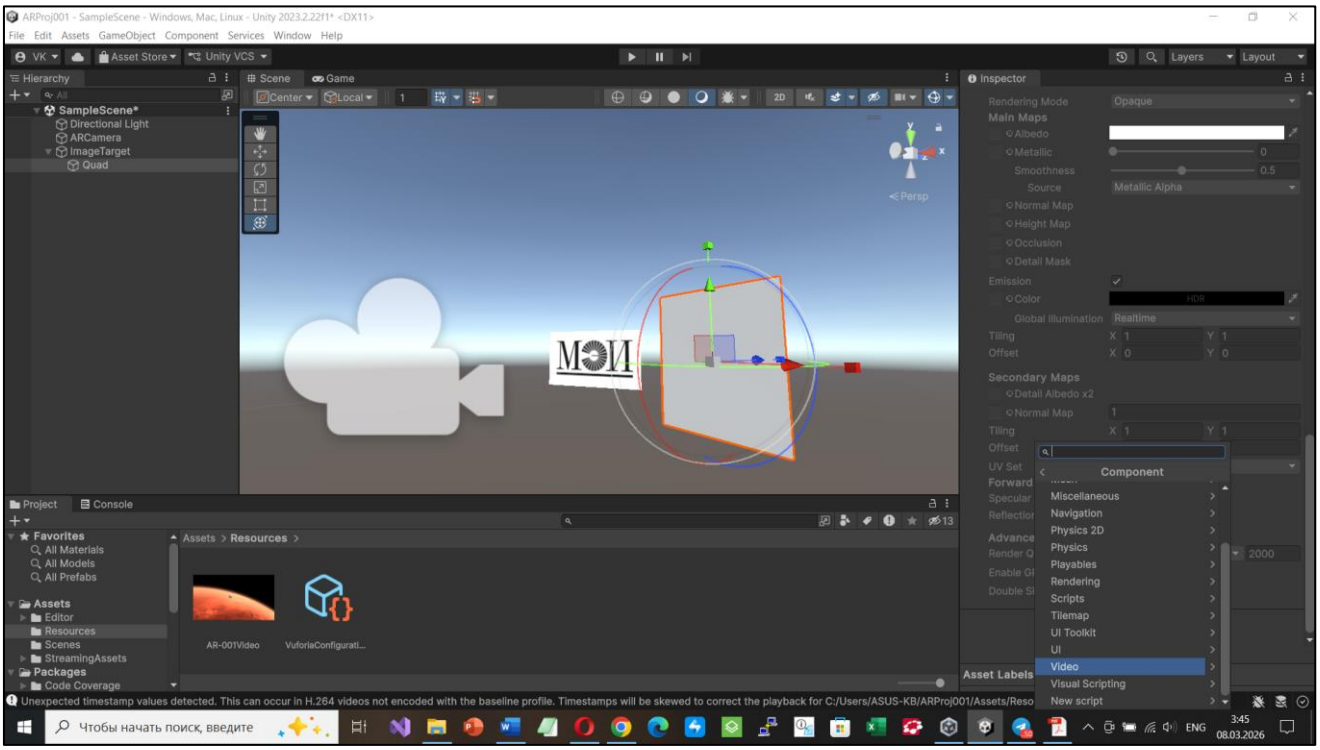
Импортируем (методом **Drag-and-Drop**) выбранный файл, в результате чего он (например) появляется в разделе ресурсы (**Resources**) библиотеки активов (**Assets**):



Следует иметь в виду, что по логике работы **Unity 3D** **контентом является как визуализируемый объект (в нашем случае видеоклип), так и «устройство», этот объект визуализирующее.** У нас это – заранее подготовленный файл с видеоклипом - **AR-001Video.mp4,** и компонент **Unity 3D**, отвечающий за визуализацию объектов типа «видео» - **Video Player. Оба эти объекта (AR-001Video.mp4 и Video Player) необходимо связать с контейнером Quad.**

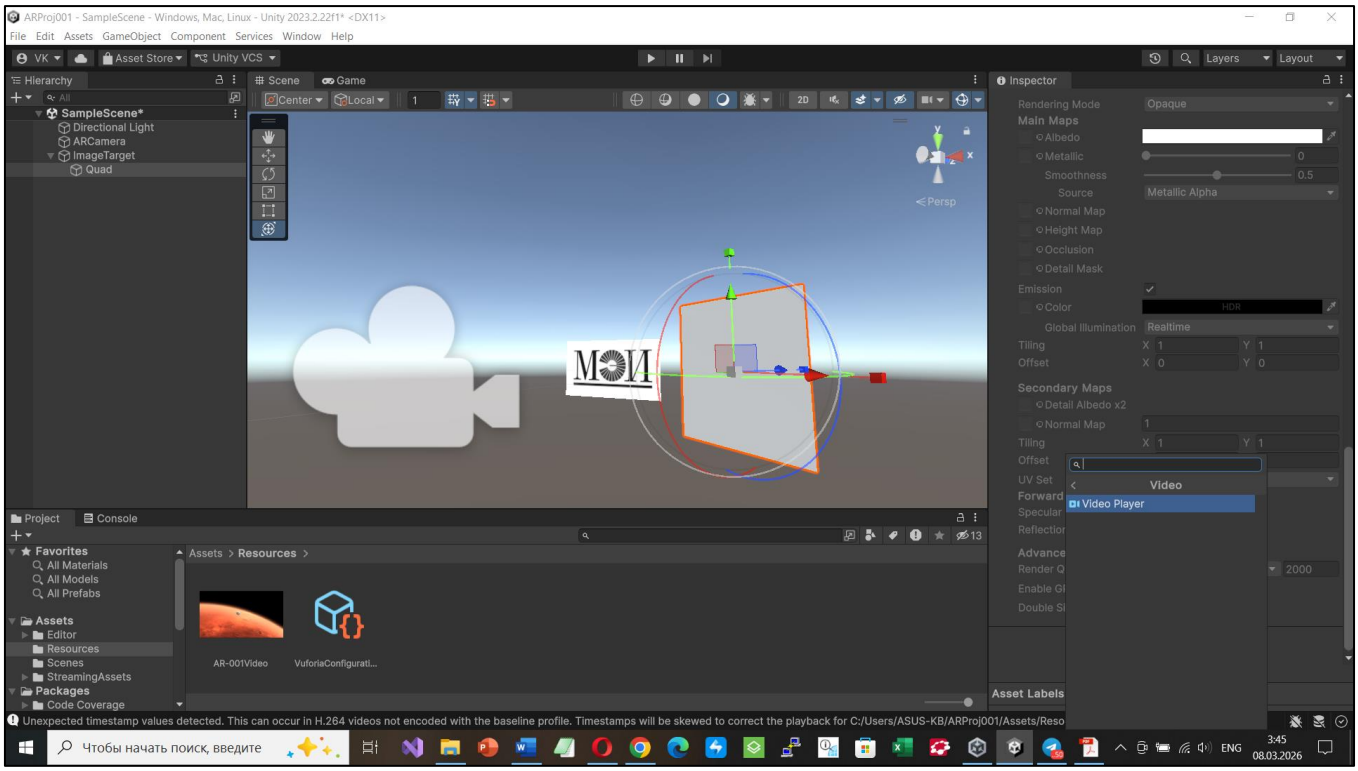
Начинаем наполнять контентом контейнер **Quad**. В первую очередь помещаем в контейнер компонент **Video Player**. Для этого в области редактора **Hierarchy** выбираем **Quad**, а в появившемся **Inspector'е** – кнопку **Add Component**:



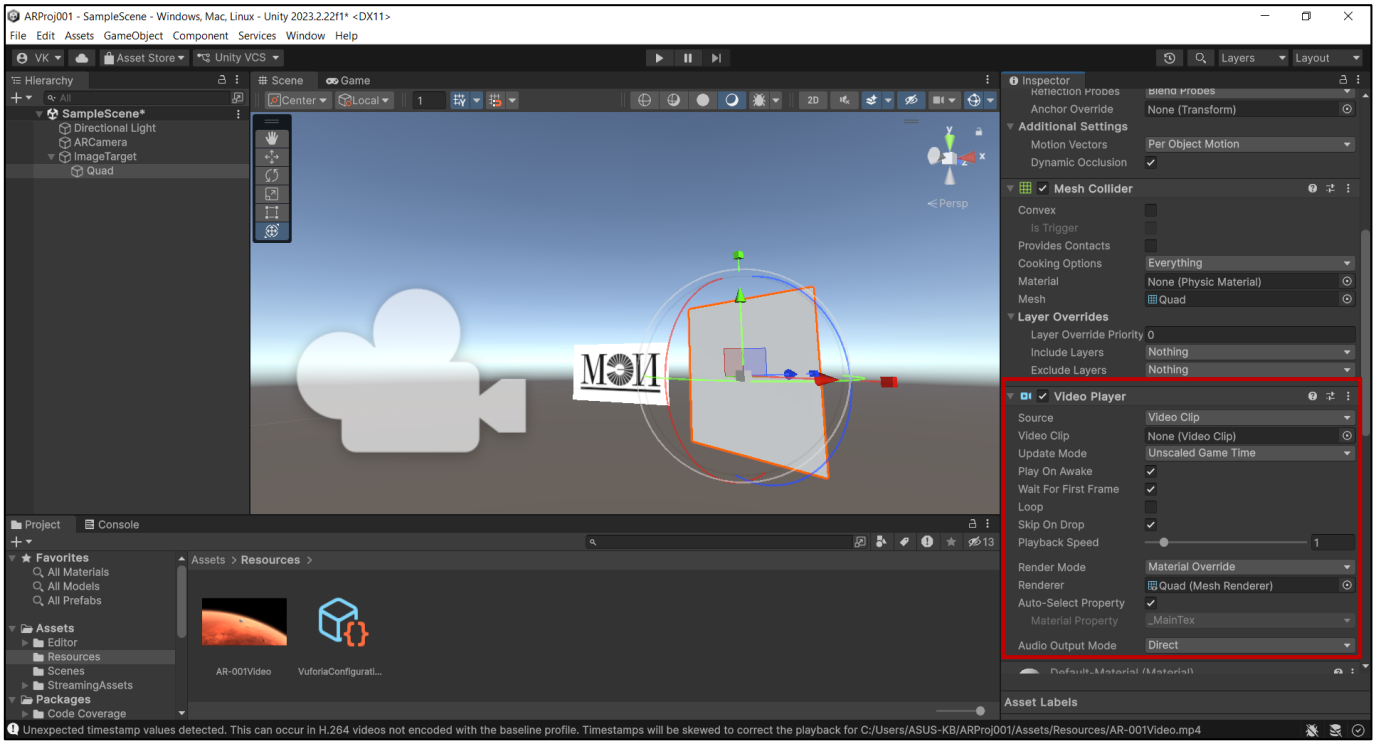


В выпадающем списке находим компонент **Video Player** по двухступенчатому пути выбора:

Add Component → Video → Video Player:

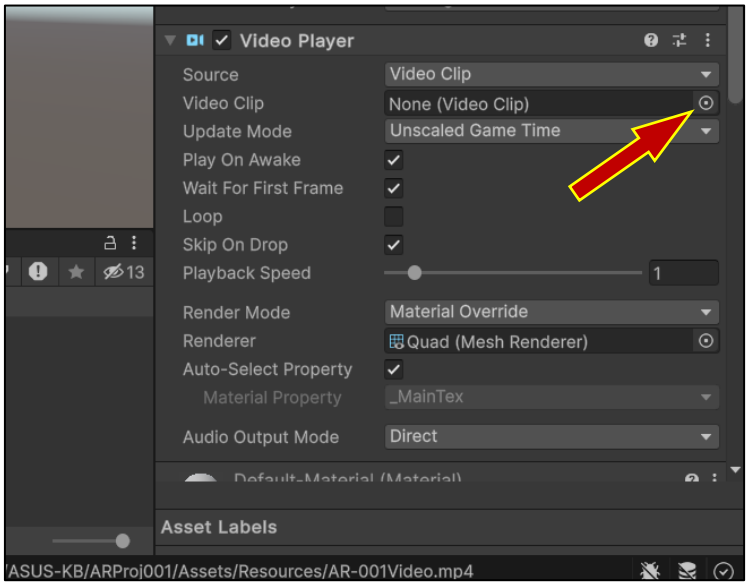


Выбираем Video Player → В результате в **Inspector'е** для **Quad** появится компонент **Video Player** и область его настройки:

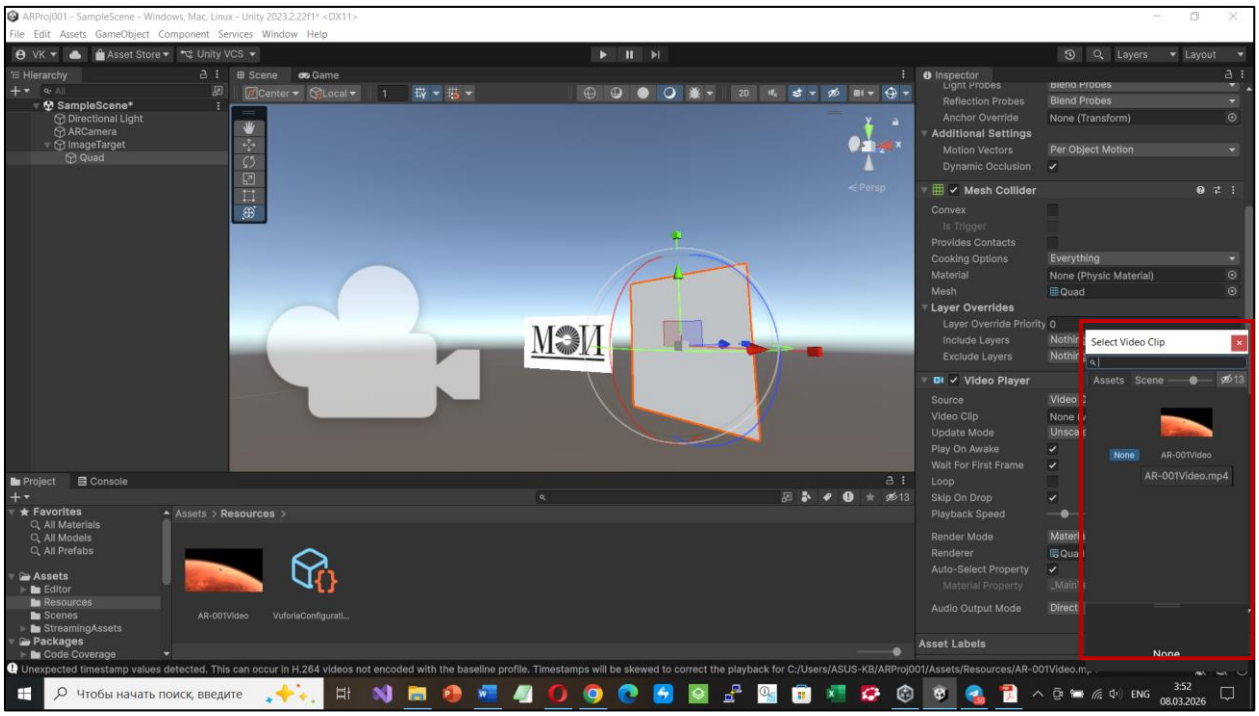


ВАЖНО!! Обратите внимание на выбранные и установленные параметры **Video Player.**

Для того, чтобы выбрать нужный нам видеоклип (файл **AR-001Video.mp4),** в компоненте **Video Player** вызываем меню доступных в проекте клипов, нажав кнопку выбора в поле **Video Clip**:



В выпадающем списке выбираем нужный нам клип:

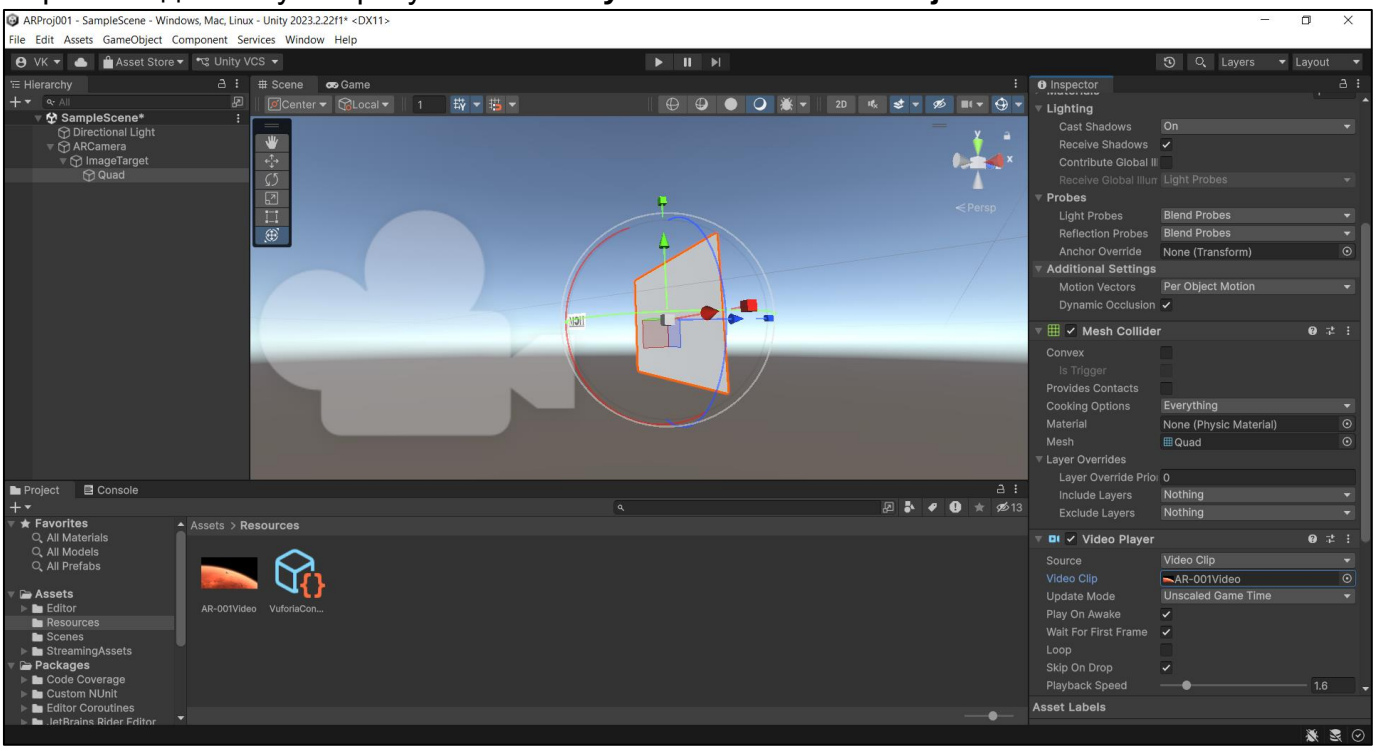


ВАЖНО!! Таким обрзом в **Unity 3D** сформирована сцена Приложения ДР:

- В сцену включена **ARCamera** вместо **Main Camera Unity 3D;**
- Импортирован таргет – **Image Target;**
- С таргетом связан контейнер **Quad** для размещения видеоклипа;
- В **Quad** в качестве контента загружен **визуализатор – плеер** – и подготовленный видеоклип.

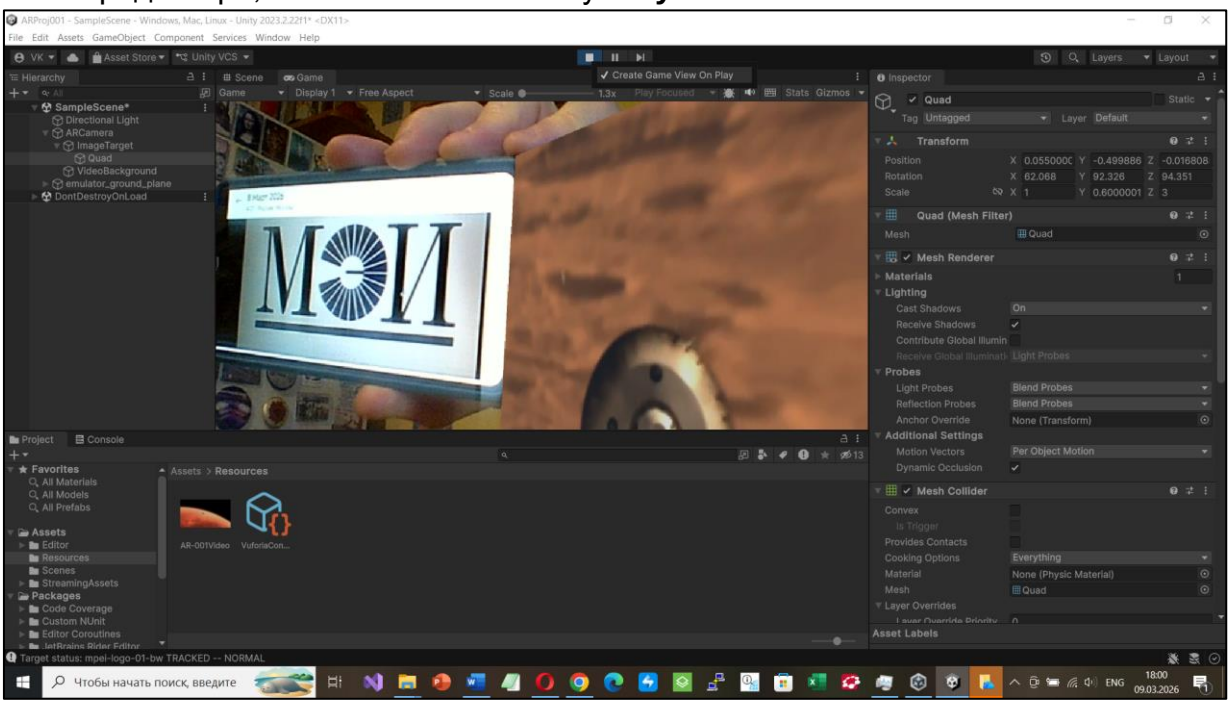
Теперь все объекты в сцене: **AR Camera**, отслеживаемый ею **Image Target** и позиционируемый относительно него контент (в нашем случае – видеоклип) в **Hierarchy** должны образовывать иерархию, где на верхнем уровне - **AR Camera,** на среднем - **Image Target,** на нижнем **Quad**, содержащий контент (видеоклип). Этого можно достичь вручную - перетаскиванием одного объекта на другой.

Сохраняем достигнутый результат в **Unity 3D: File → Save Project.**



Проверить работоспособность разрабатываемого Приложения ДР непосредственно в **Unity 3D** можно на локальной машине, если она снабжена видеокамерой.

Предпросмотр достигнутого результата выполняется в режиме **Game** (закладка) в области **Scene View** редактора, нажатием на клавишу **Play**:



Полученный результат показывает, что таргет и клип работают, но бывает так, что сцена сформирована некорректно – экран, скорее всего, развернут к зрителю тыльной стороной и взаимное расположение таргета и экрана возможно следует изменить таким образом, чтобы в область трансляции МУ попадал весь экран (**Quad**).
В случае обнаружения некорректно сформированной сцены ДР-приложения необходимо еще на стадии работы в **Unity Editor** добиться приемлемого результата → убедимся в том, что **Quad** развернут правильно, его размеры соответствуют размерам таргета, имеют верное соотношение сторон и т.д. Рекомендуется располагать камеру в точке начала координат (0,0,0), с углом поворота по трем осям – 0, и масштабным коэффициентом = 1 по всем трем осям →

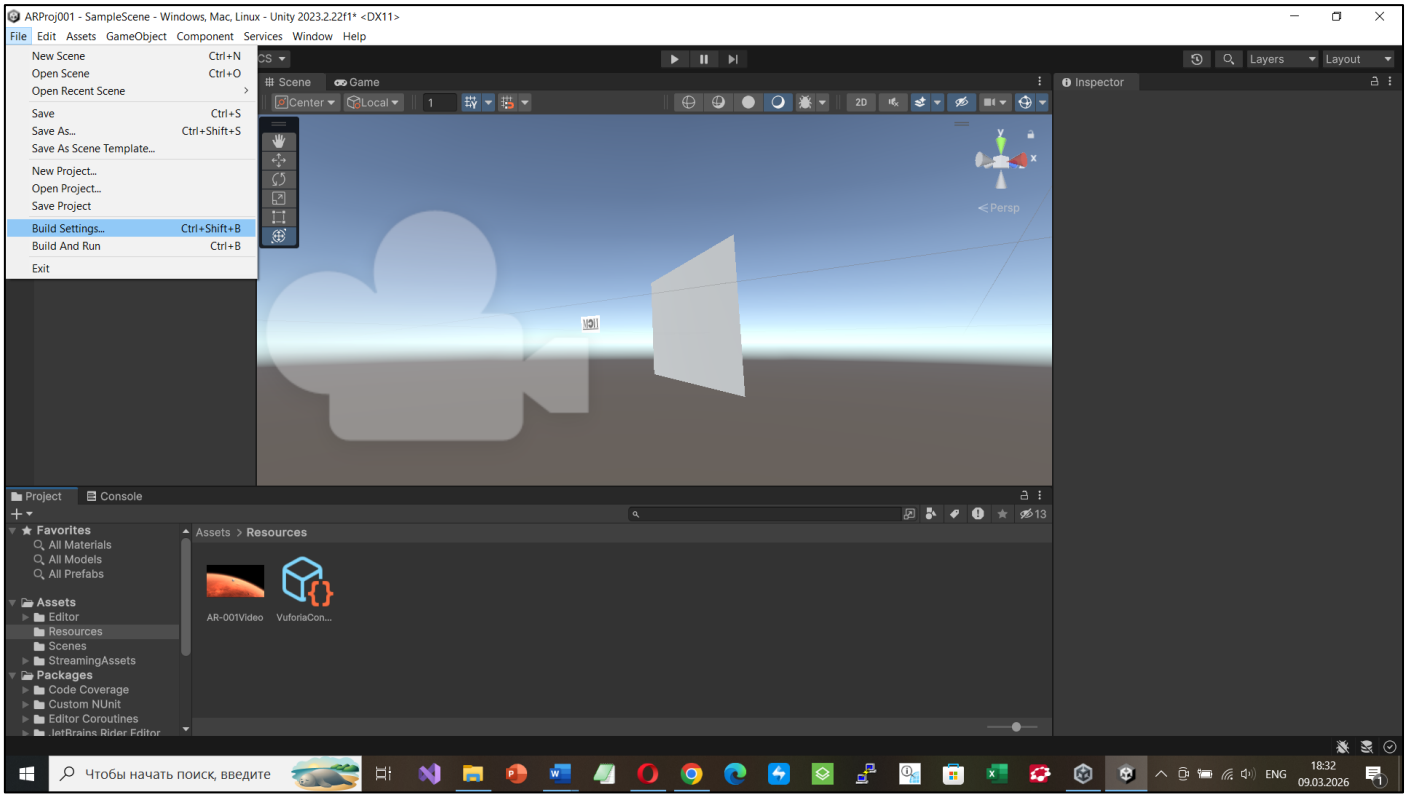
Разработка сцены закончена. Здесь можно сохранить полученный результат – сцену - не выходя из **Unity 3D**. Это может пригодиться в дальнейшем при доработке сцены, при прерывании сеанса работы в **Unity 3D** и т.д. Для сохранения сцены выполнить: **File→Save** или **File→Save as.** Сохранение производится в локальной ФС на вашей локальной машине.

3. Создание файла .apk для загрузки Приложения ДР на Android-МУ.

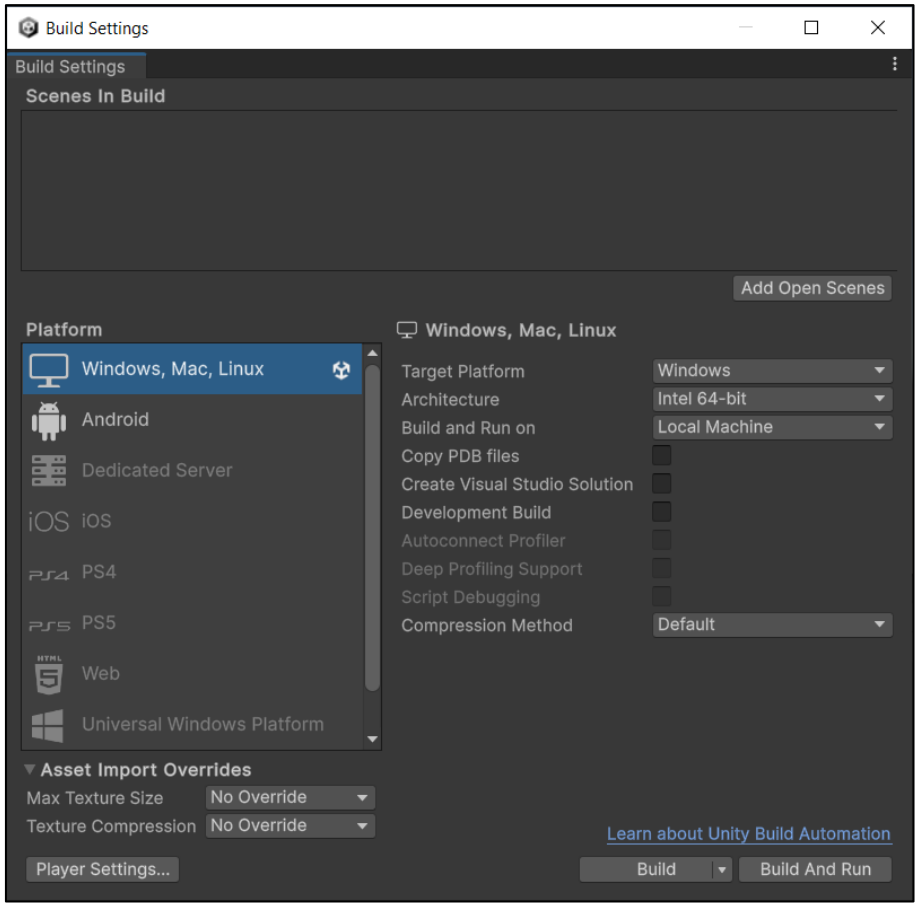
Разрабатываемое приложение должно будет использовать камеру конкретного **Android-**мобильного устройства. **Unity 3D** в нашем проекте только подготавливает универсальный драйвер **AR Camer'ы** для типового **Android**-устройства. Все особенности конкретных, передовых **Android**-устройств (стерео, 4K, и т.д.) требуют дополнительного программирования, что находится за пределами данной ЛР. Формат файла загружаемого Приложения ДР для **Android**-устройств – это **.apk**.

3.1. Для создания файла .apk включаем в редакторе Unity 3D режим Build:

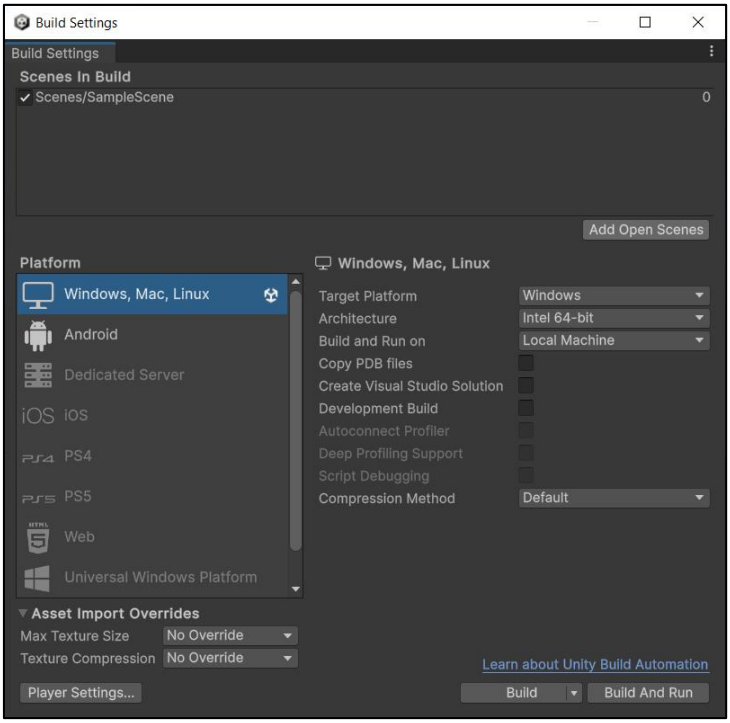
File→Build Settings:



Все предварительные настройки – выбор сцены, настройки ОС устройства – осуществляются в открывшемся окне **Build Settings**.



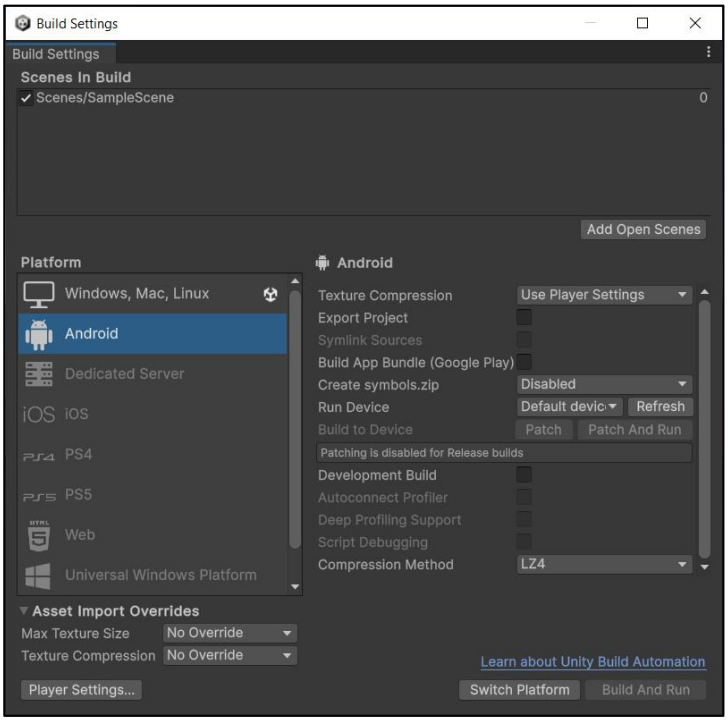
Для выбора сцены нужно использовать кнопку **Add Open Scene**, пометив в случае необходимости нужную из списка:



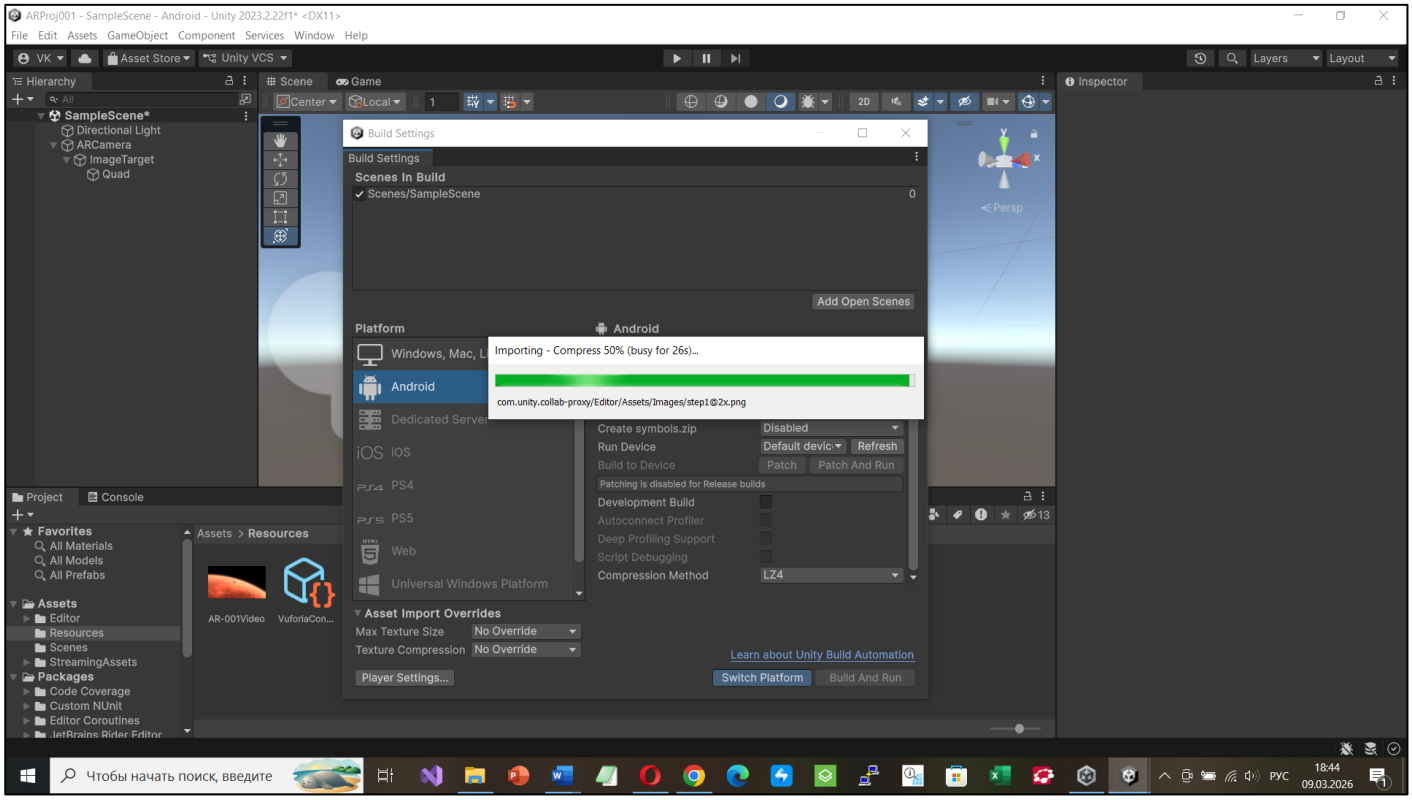
Пока в списке **Platform не выбрана** (не отмечена справа значком) **или может даже отсутствовать** нужная нам платформа – **Android**, под управлением которой должно работать устройство ДР – Мобильное Устройство.

3.2. Настройка Builder’а для платформы Android.

Для настройки Builder’а выбираем **Android** в списке платформ

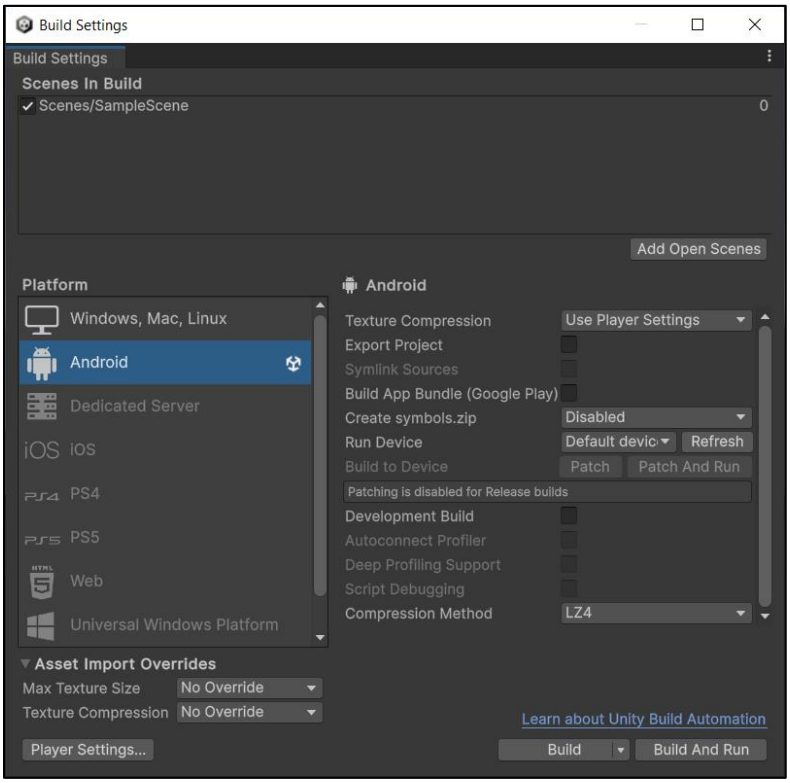


И нажимаем кнопку **Switch Platform**:

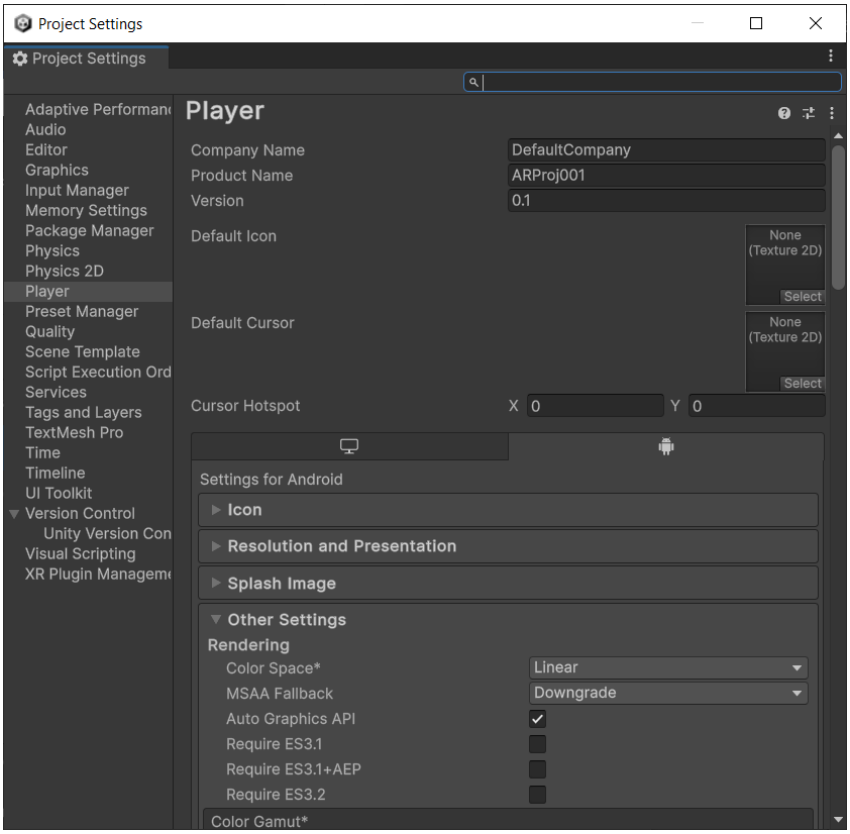


В результате:

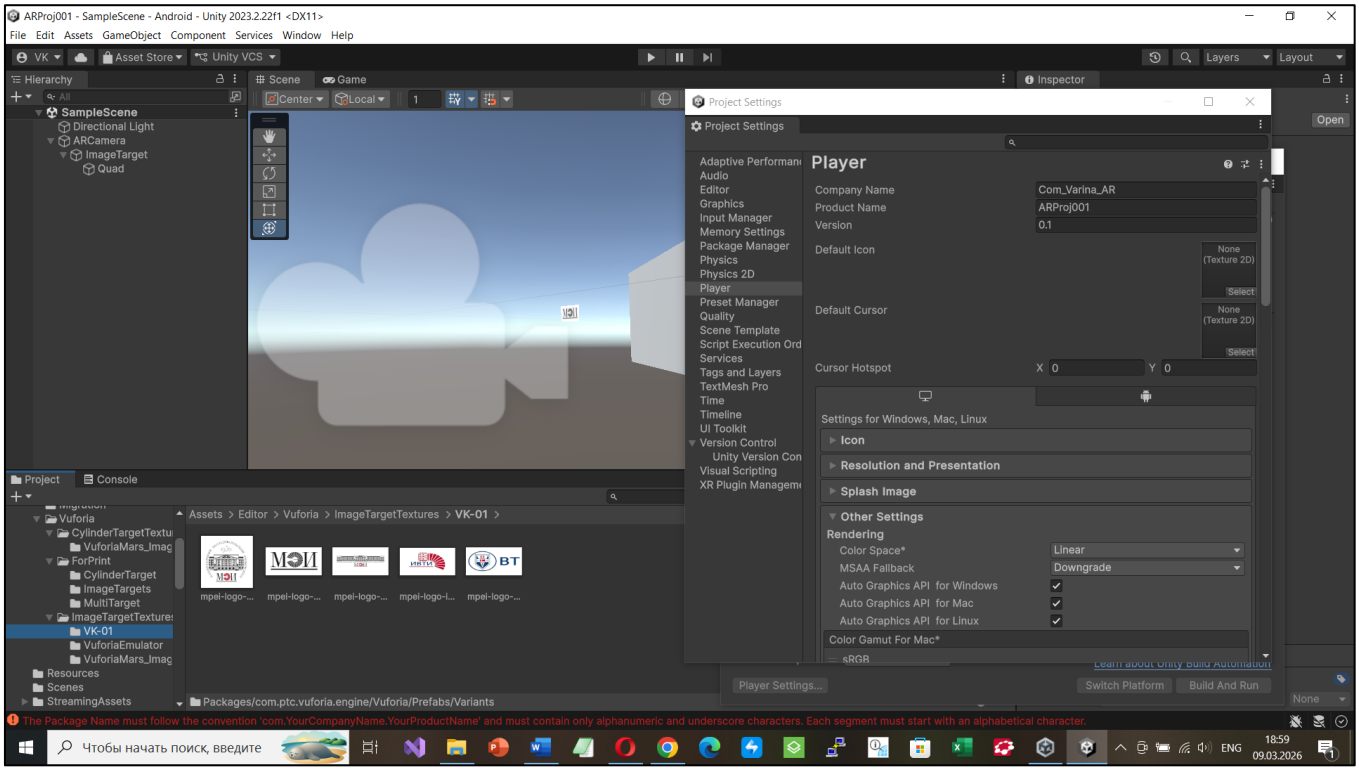
Позиция **Android** в окне **Build Settings** «отмерзает» и можно переходить к настройке работы оборудования в данной ОС → клавиша **Player Settings.**

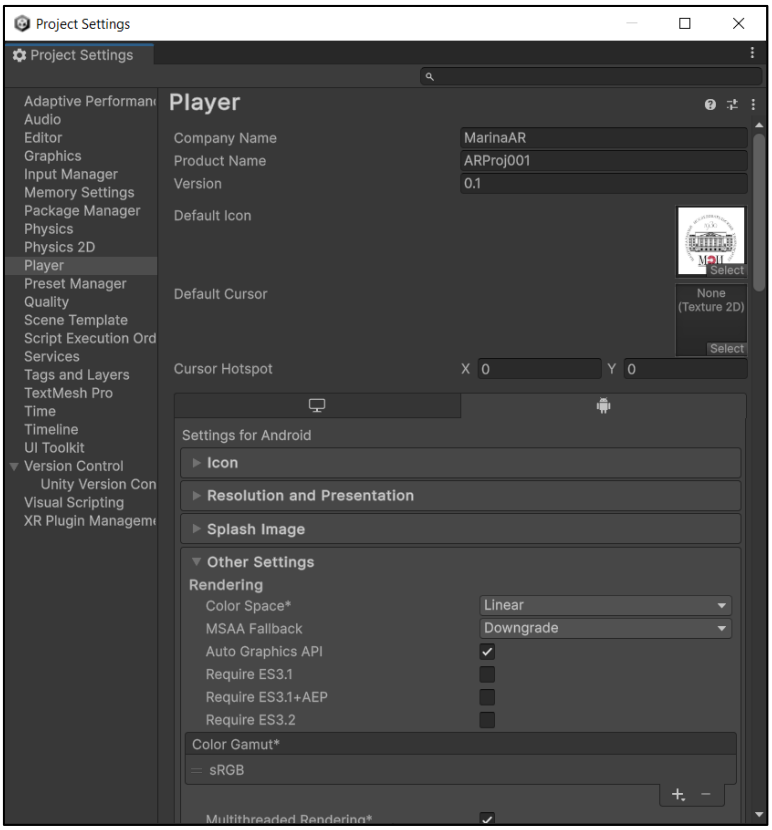


В результате в открывшемся окне **Project Settings** появляются настройки для плеера под **Android**:



Выполняем необходимые настройки. Заполните позицию **Company Name** (пофантазируйте), убедитесь в том, что остальные позиции заполняются по умолчанию (рекомендуется изменить имя **Product Name** для удобства проверки ваших Проектов преподавателем, добавить фамилию латинскими буквами), добавьте иконку для вашего Проекта → Режим **Select** в позиции **Default Icon**. Если поле для пиктограммы останется незаполненным, то для вашего Приложения ДР пиктограммой всегда будет логотип **Unity**.





И **Company Name** и **Product Name** выбираются разработчиком произвольно, с учетом следующих ограничений: можно использовать только буквы латинского алфавита и цифры. Выбранные **Company Name** и **Product Name** размещаются в соответствующих полях в верхней части **Player Settings.**

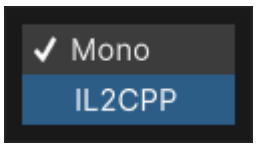
Познакомьтесь с остальными позициями **Project Settings**. Изменять заполненные позиции по умолчанию не следует.

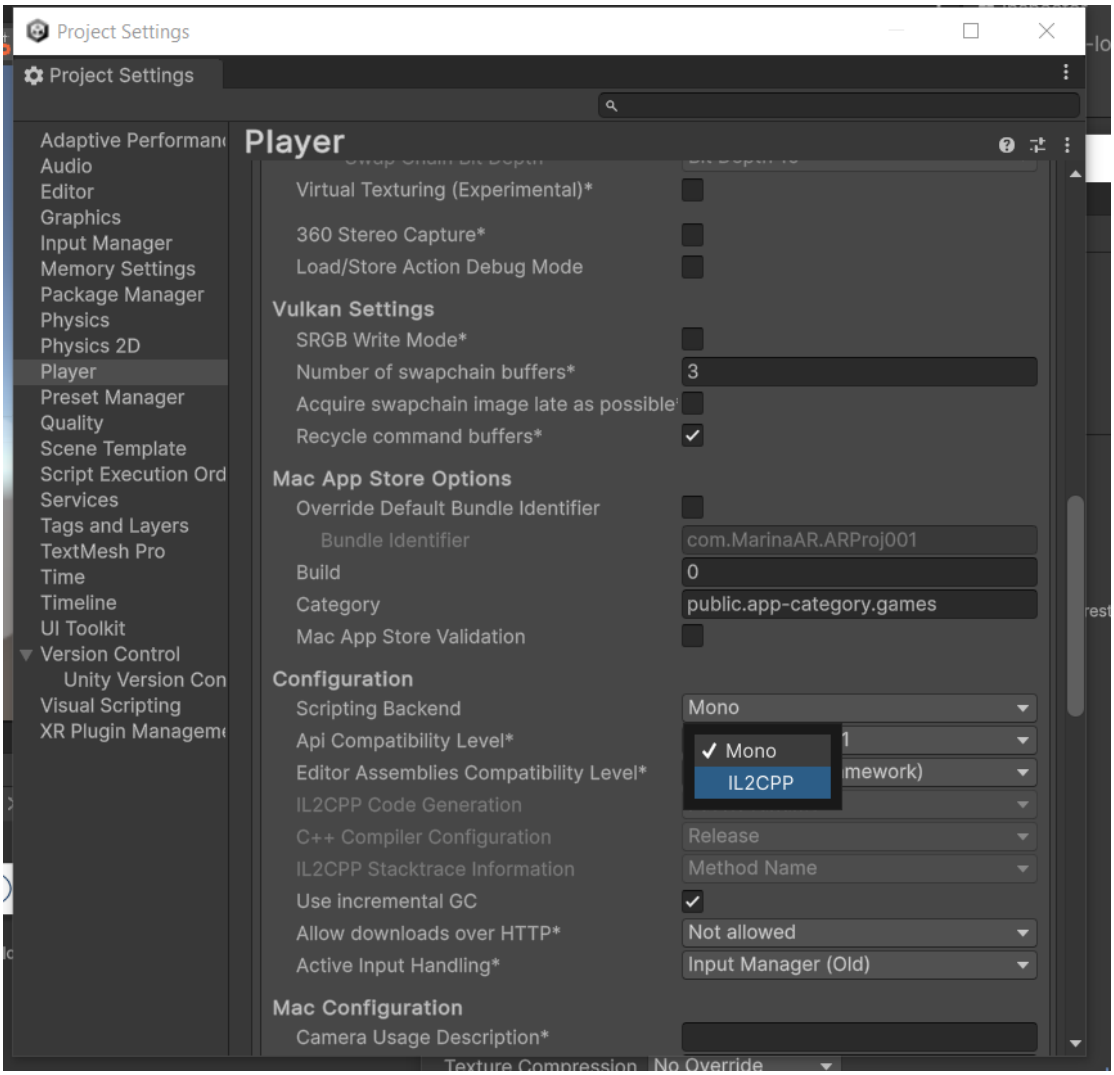
ВАЖНО!!! Обратите внимание, что для текущей версии **Unity:**

- минимальная версия **Android – 8**
- необходимо использовать **64-разрядную** версию

Для соответствия этим условиям выполняем в **Project Settings** следующее:

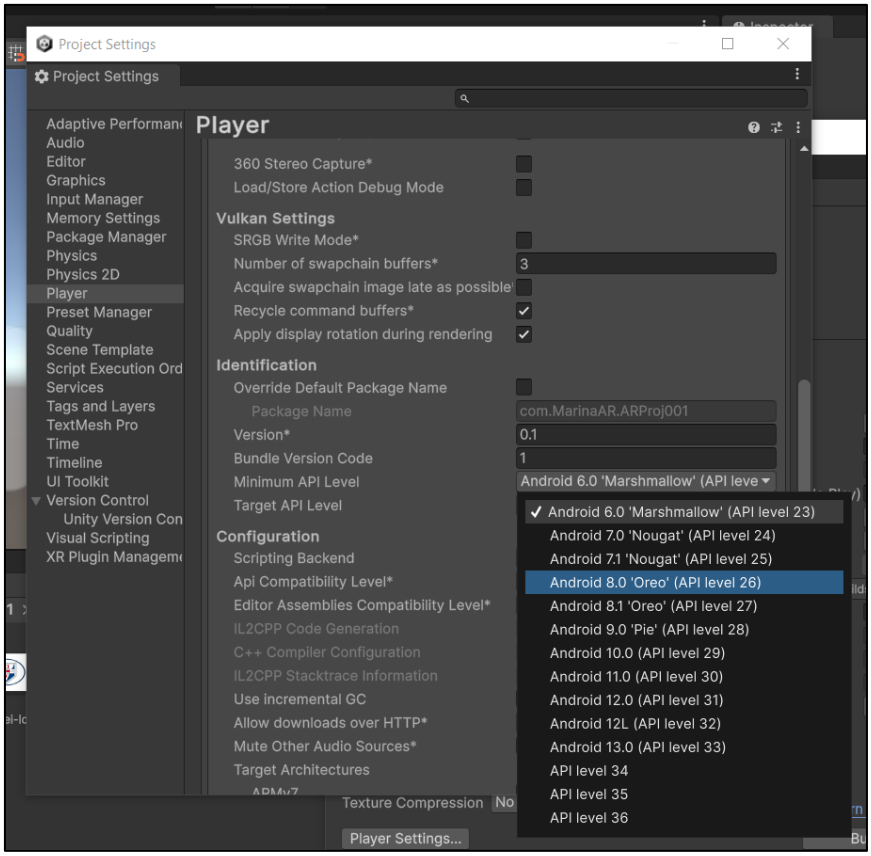
- переходим из режима **Mono** в режим **IL2CPP:**

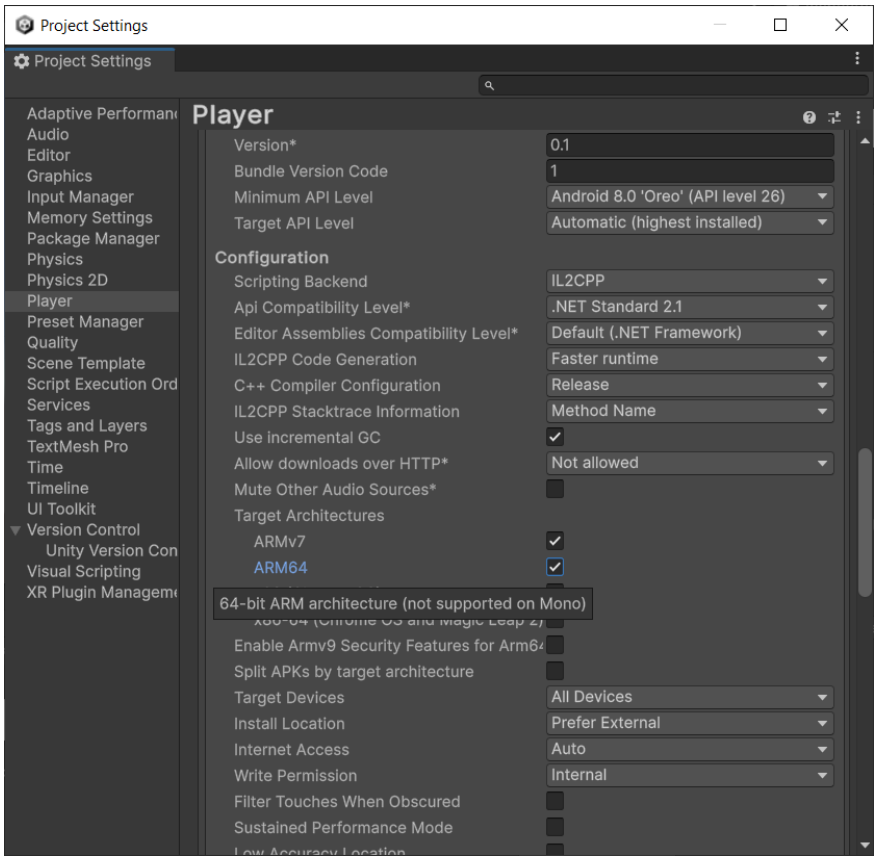




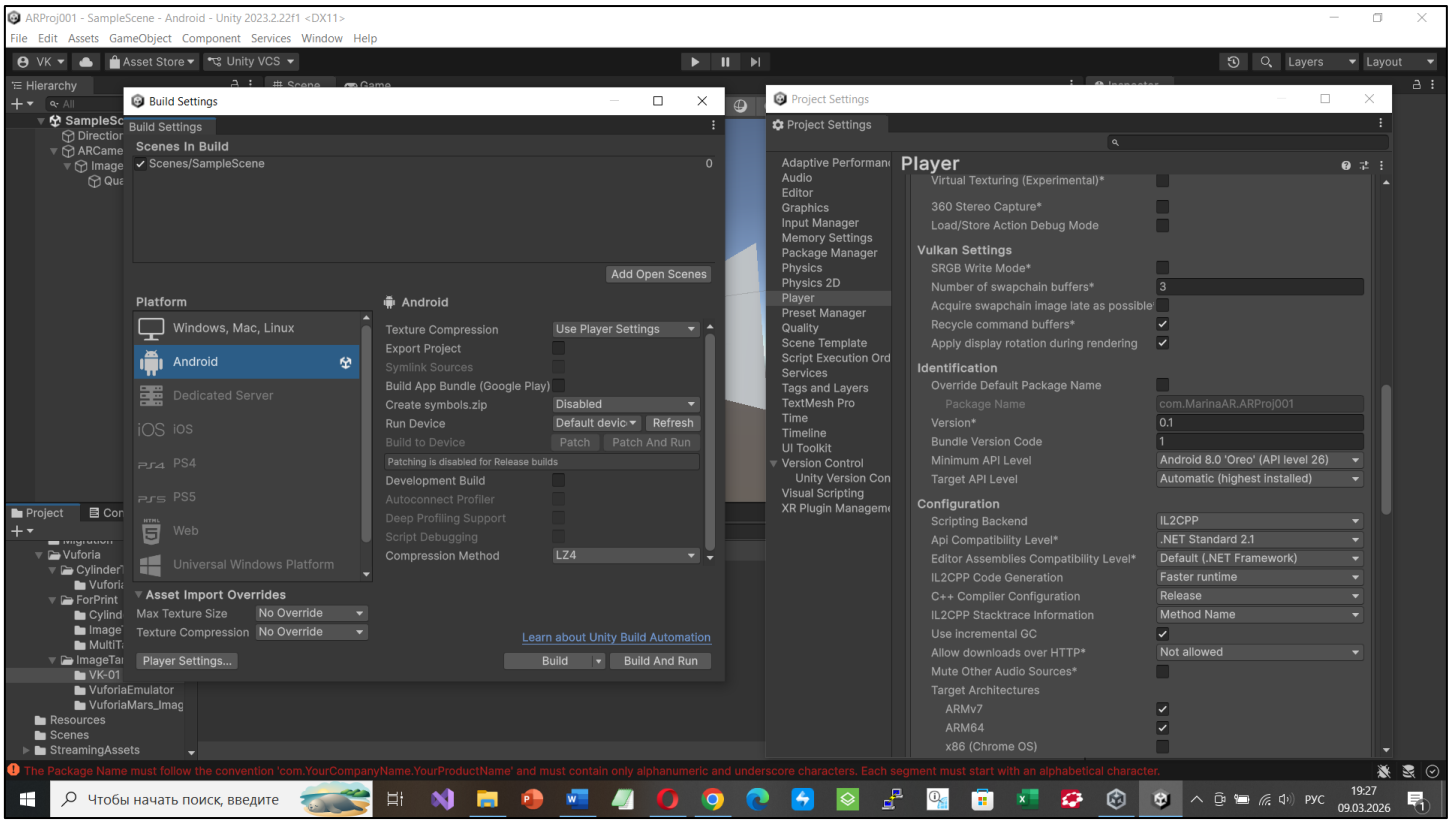
После этого выбираем нужный уровень (API Level) Android и разрядность процессора МУ:

Project Settings → Player → Identification → Minimum API Level



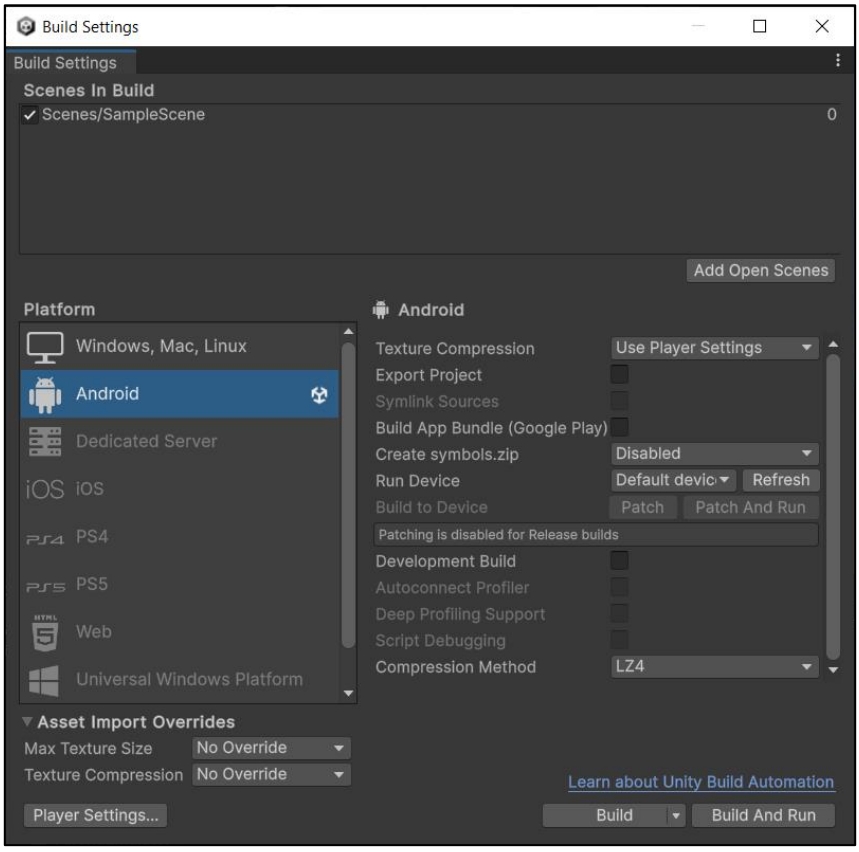


После выполнения всех этих настроек можно выполнить операцию **Build:**

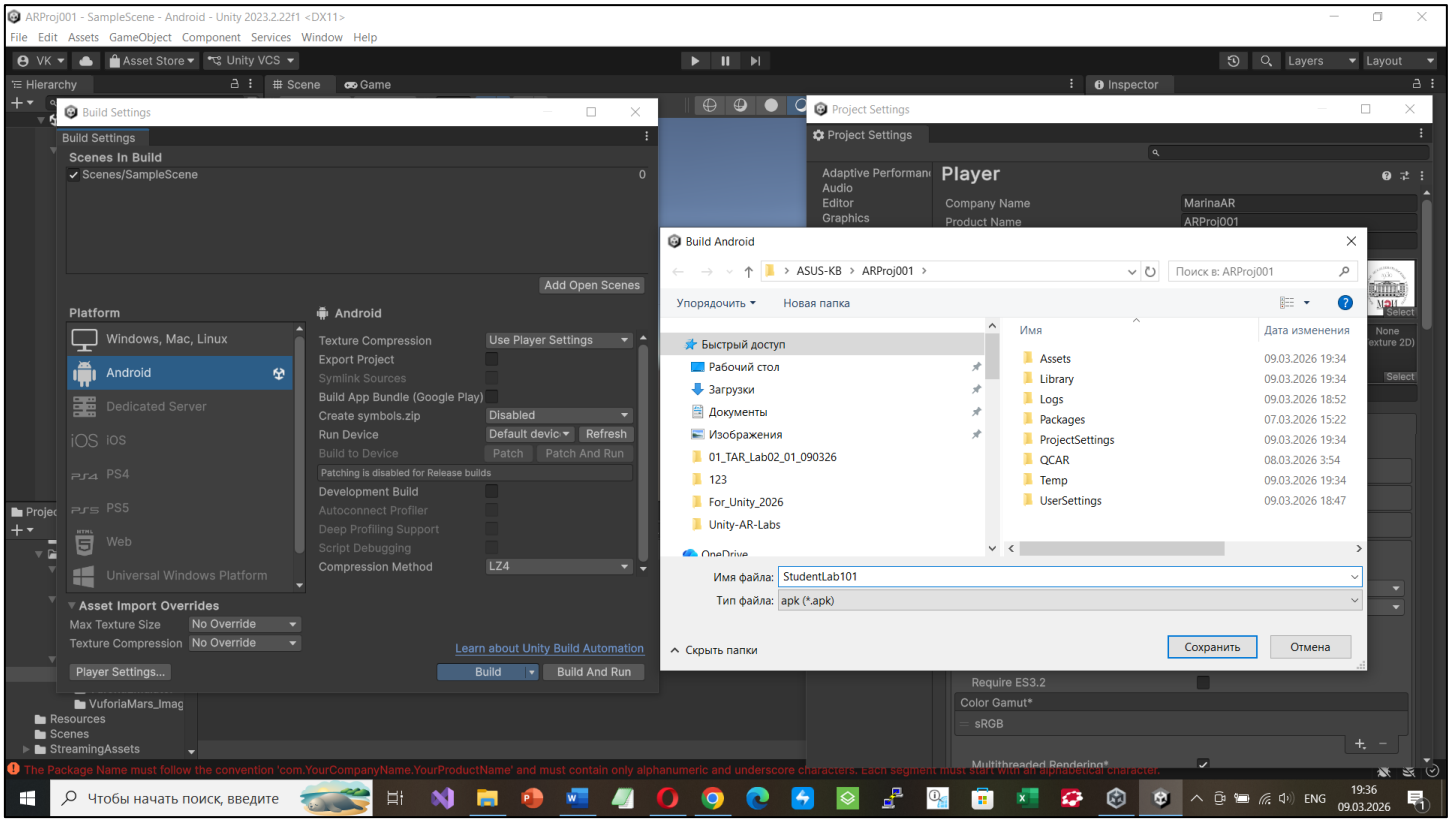


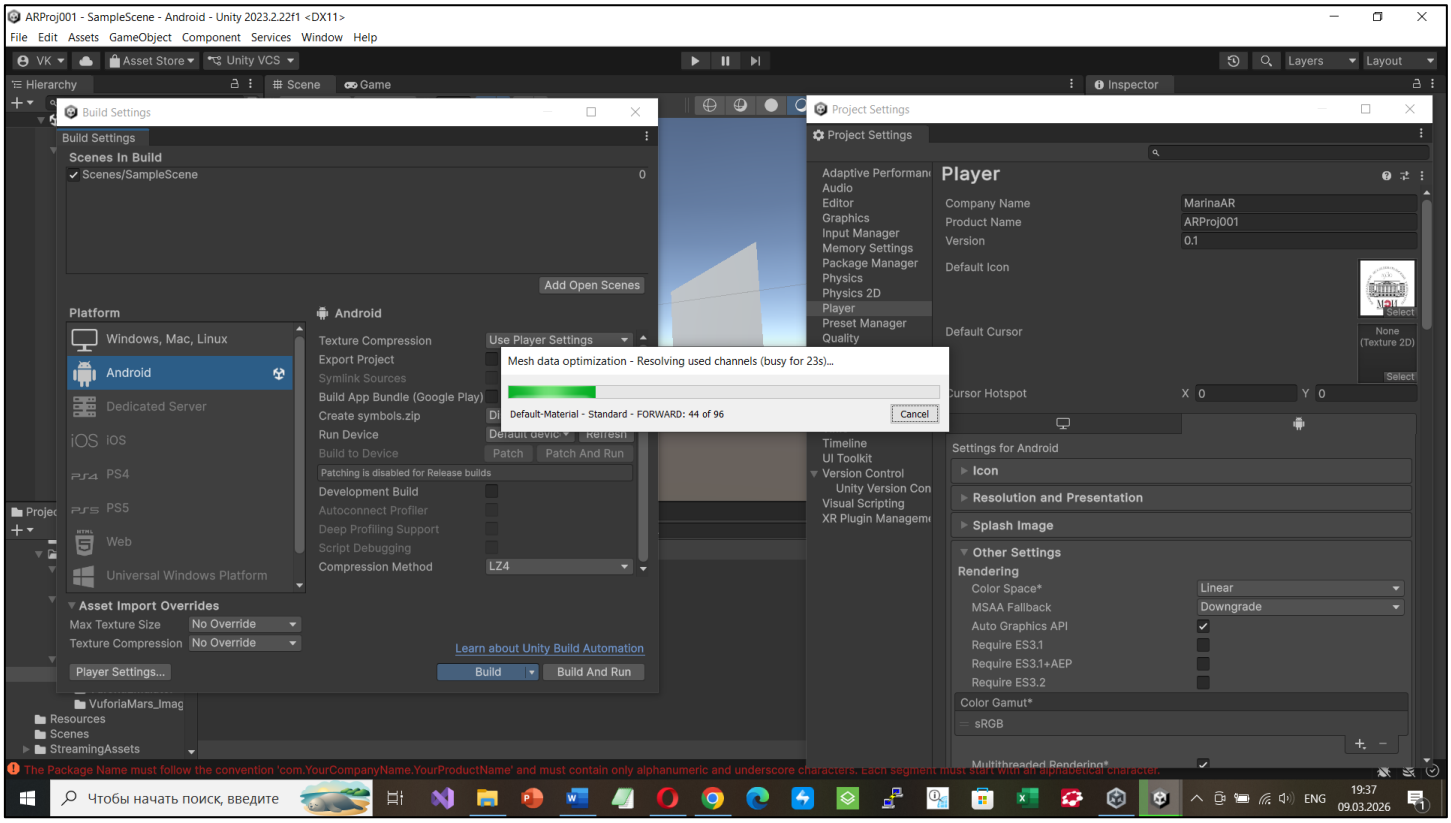
3.3. **Выполнение операции Build**.

В списке **Platform** доступна и выбрана платформа **Android**, доступна клавиша **Build**:



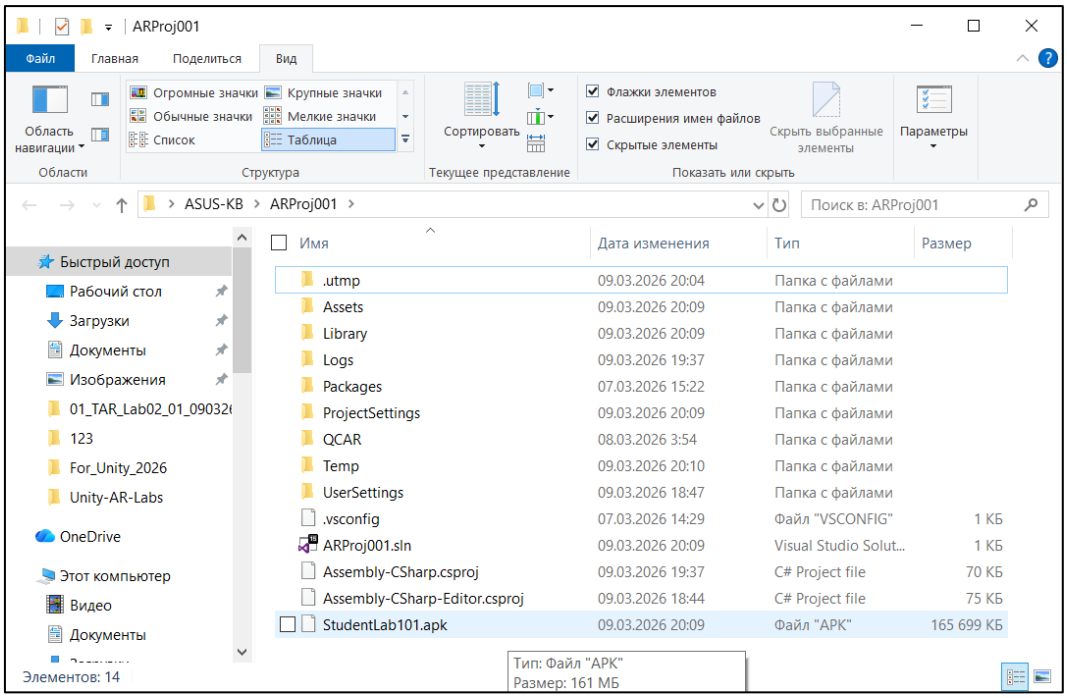
Нажимаем клавишу **Build** → выбираем рабочую директорию в локальной файловой системе разработчика для сохранения собранного файла **.apk**:





Процесс сборки файла **.apk** может длиться несколько минут.

Файл **StudentLab101.apk** разработанного в данном описании Приложения ДР сохранен.



Теперь его осталось загрузить (и далее – установить) в **Android - МУ** любым известным вам способом.

Отчет по ЛР№2

- **должен быть оформлен в соответствии с правилами, принятыми в МЭИ,**
- **содержать цель ЛР, краткое описание основных этапов разработки с необходимым количеством сканов экрана,**
- **содержать описание проблем, если они возникали,**
- **содержать результаты тестирования в режиме Play в редакторе Unity и сканы экрана МУ с результатами работающего Приложения.**

Создание коротких видеороликов и процесса тестирования и работающего Приложения ДР приветствуются.

Разработанное в рамках ЛР № 2 AR-Приложение необходимо продемонстрировать преподавателю. Для этого загрузите свой .apk на любой файлообменник или доступное облако и пришлите мне ссылку!

Большая просьба – формируйте имя загружаемого файла .apk так, чтобы было можно понять, кто его выполнял. Префикс Student в рассмотренном в описании примере – это ваши ФИО латинскими буквами!

Не забудьте про таргет – его тоже необходимо прислать мне отдельным файлом!