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 2D-Изображения.

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

Введение.

Работа по созданию приложений ДР заключается в создании 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 были подробно рассмотрены в Описнии ЛР№2, Часть 1.

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.

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 (изображение);

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 **Части 2 ЛР №2** это изображение в одном из следующих форматов **.jpg/.png/.bmp/.gif.**

Для выполнения данной части ЛР№2 предлагается в среде Unity завести новый проект ARProj002.

ВАЖНО!!! Все шаги и настройки, выполнявшиеся в ЛР №2 Часть 1 до начала размещения контента, связанного с меткой, в сцене ДР – повторяются при выполнении данной Части 2 (до стр. 22 описания ЛР №2, Часть1).

1. Контент в данной ЛР – это подготовленный файл - изображение в одном из перечисленных выше форматов, в нашем случае **Plan_MPEI_2026.png**. Находится в файловой структуре на локальной машине.

При этом - метка (таргет), объект **Image Target**, уже размещен в сцене.

Теперь необходимо загрузить тот плоский объект (2D-изображение), который должен появиться на экране устройства на фоне транслируемой реальности на месте, определяемом таргетом.

2D-Изображение (контент) в данной ЛР – это заранее подготовленный файл -

Plan_MPEI_2026.png. Находится в файловой структуре на локальной машине.

Для размещения контента в **Unity 3D** воспользуемся функционалом **Unity 3D**. Для этого в системе существует большое количество собственных объектов. В частности, 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, это – **Quad**.

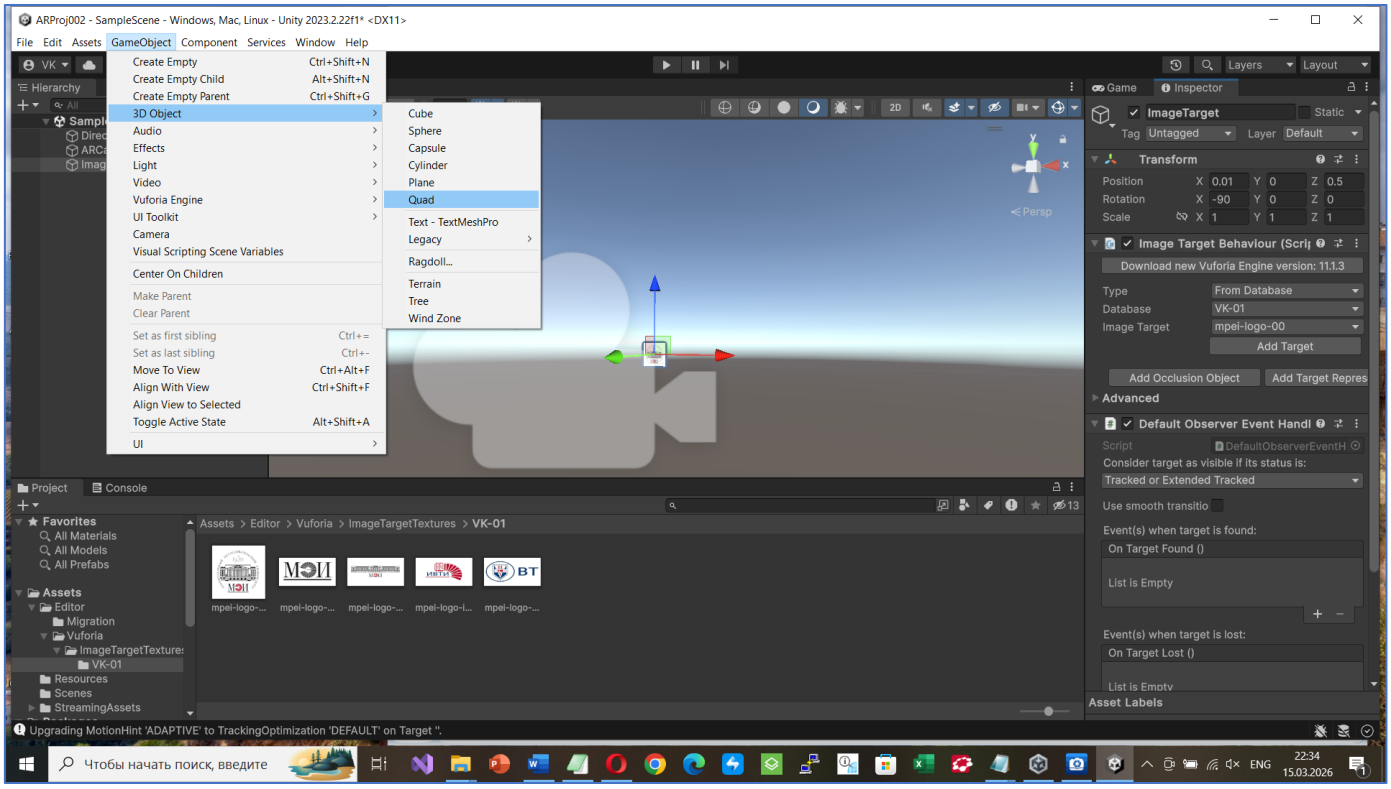
Т.о. в **Unity 3D** связываем **Image Target** с размещаемым контентом через объект (шаблон, контейнер) **Quad.**

Сцену формируем не обращая внимания на подчинение в иерархии. В дальнейшем, корректная работа Приложения ДР обеспечивается в том числе и подчиненным расположением **Quad** относительно **Image Target** в иерархии **(Hierarchy)**. А **Image Target** занимает подчиненное положение относительно **ARCamera** в иерархии **(Hierarchy)**. Это означает, что контент, связанный с **Quad** будет визуализироваться только после распознавания камерой таргета.

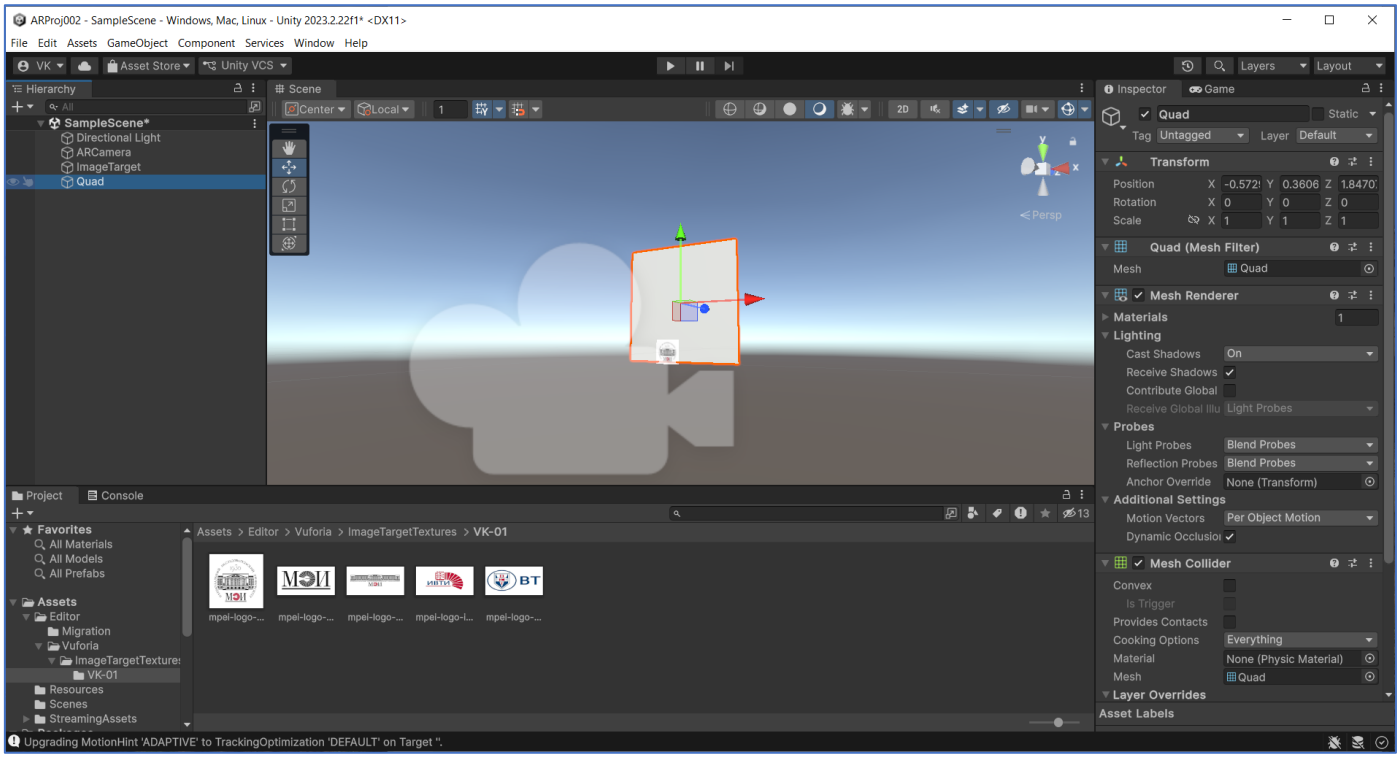
Последовательность операций для получения результата:

- В **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**:

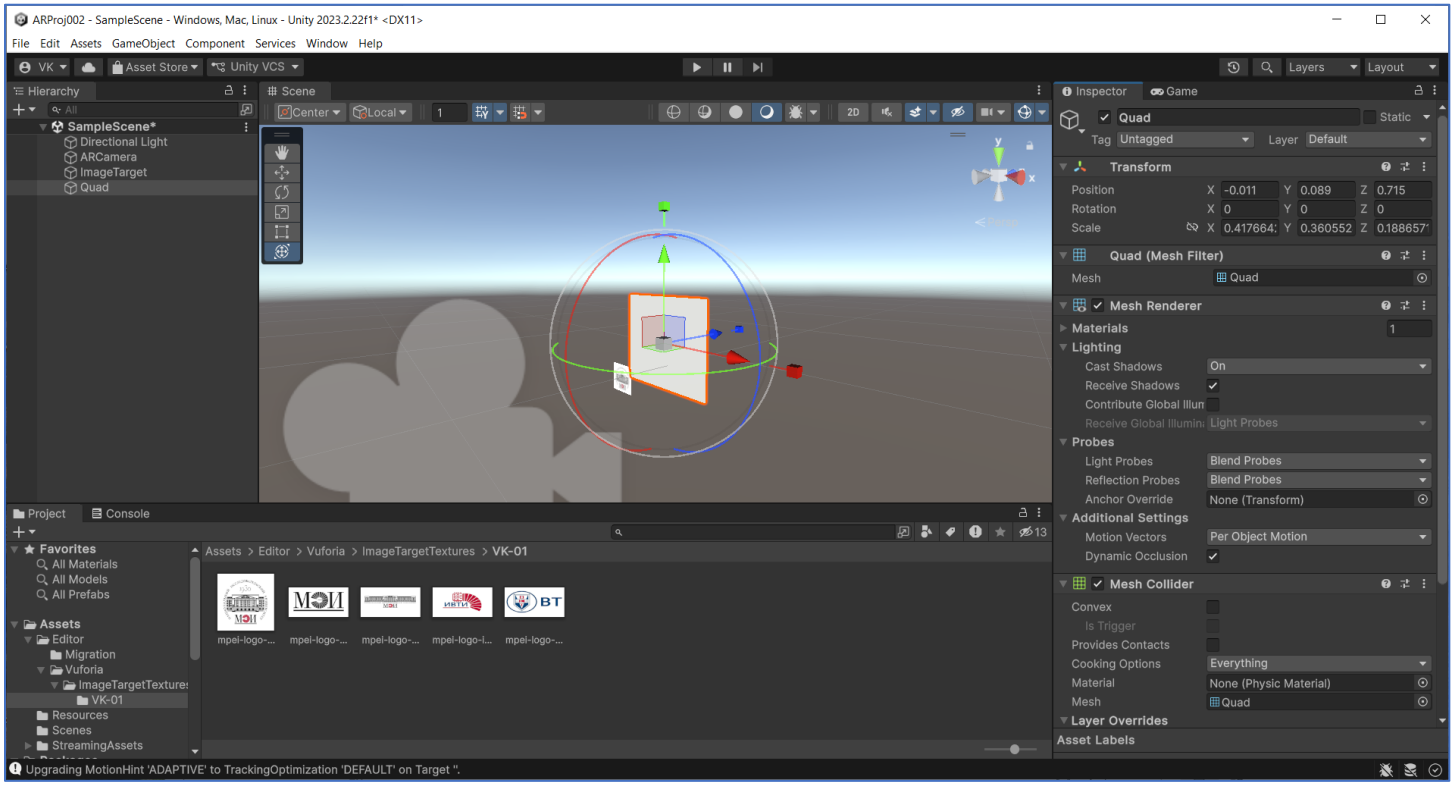


В результате в сцене появляется новый объект **Quad**, а в **Inspector'e**– информация о нем:

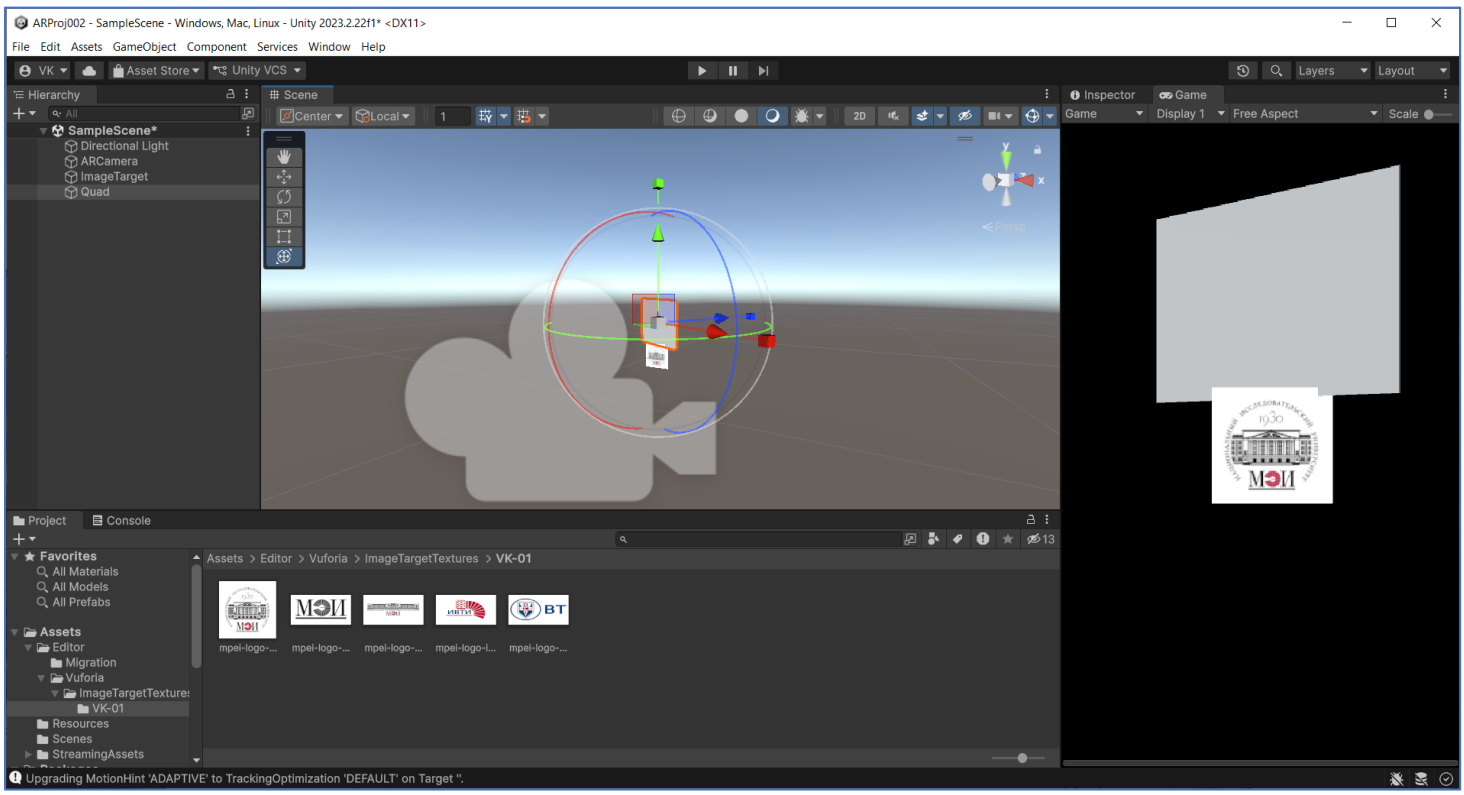


Не меняя расположения камеры и таргета, найдем наиболее удобное положение для контейнера 2D-Изображения – **Quad'a**. Например, экран с изображением (**Quad** с изображением) будет с небольшим разворотом по вертикали (ось «Y»). Вы можете выбрать расположение контейнера самостоятельно (**Toolbar** или **Transform** в **Inspector'e**).

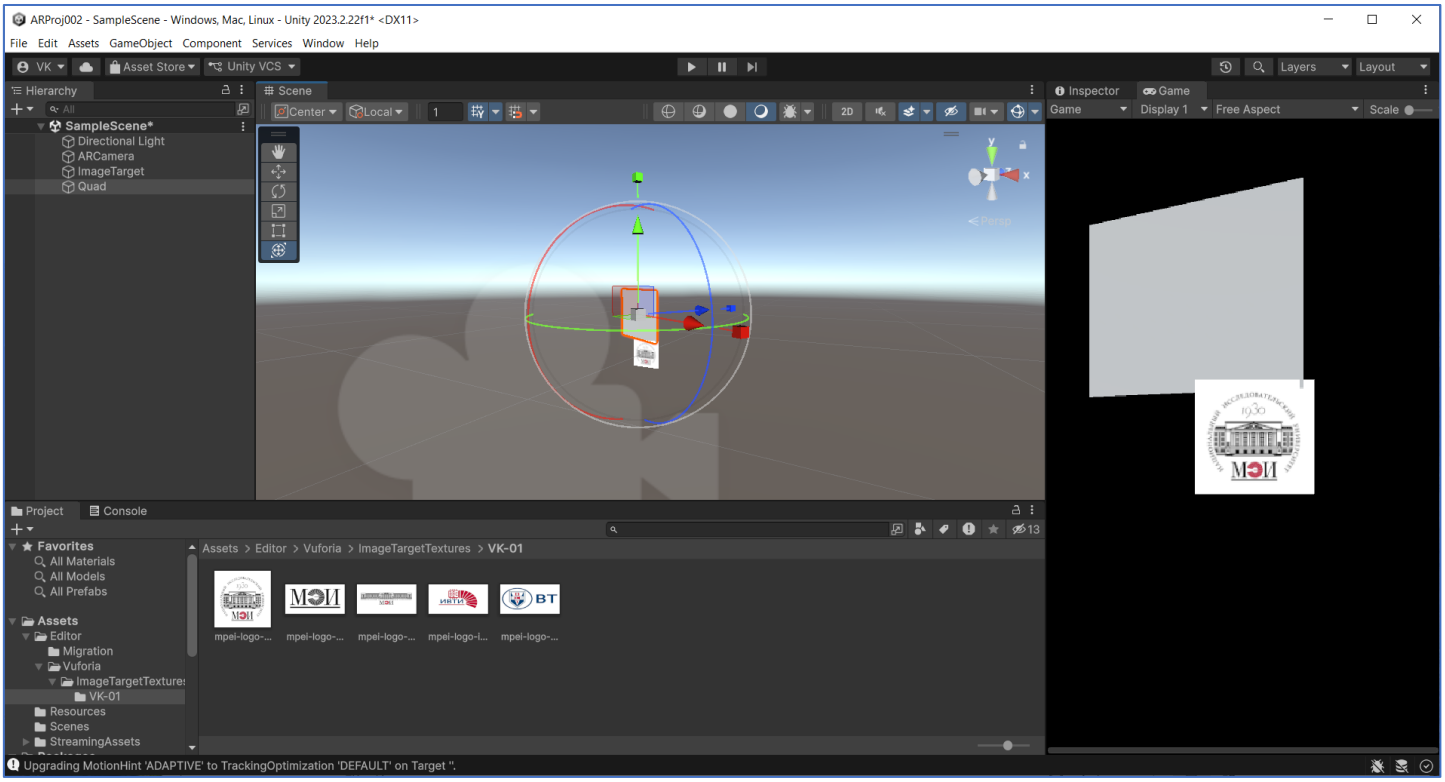
- **Image Target** и **Quad**:



- То, что видит **ARCamera**:

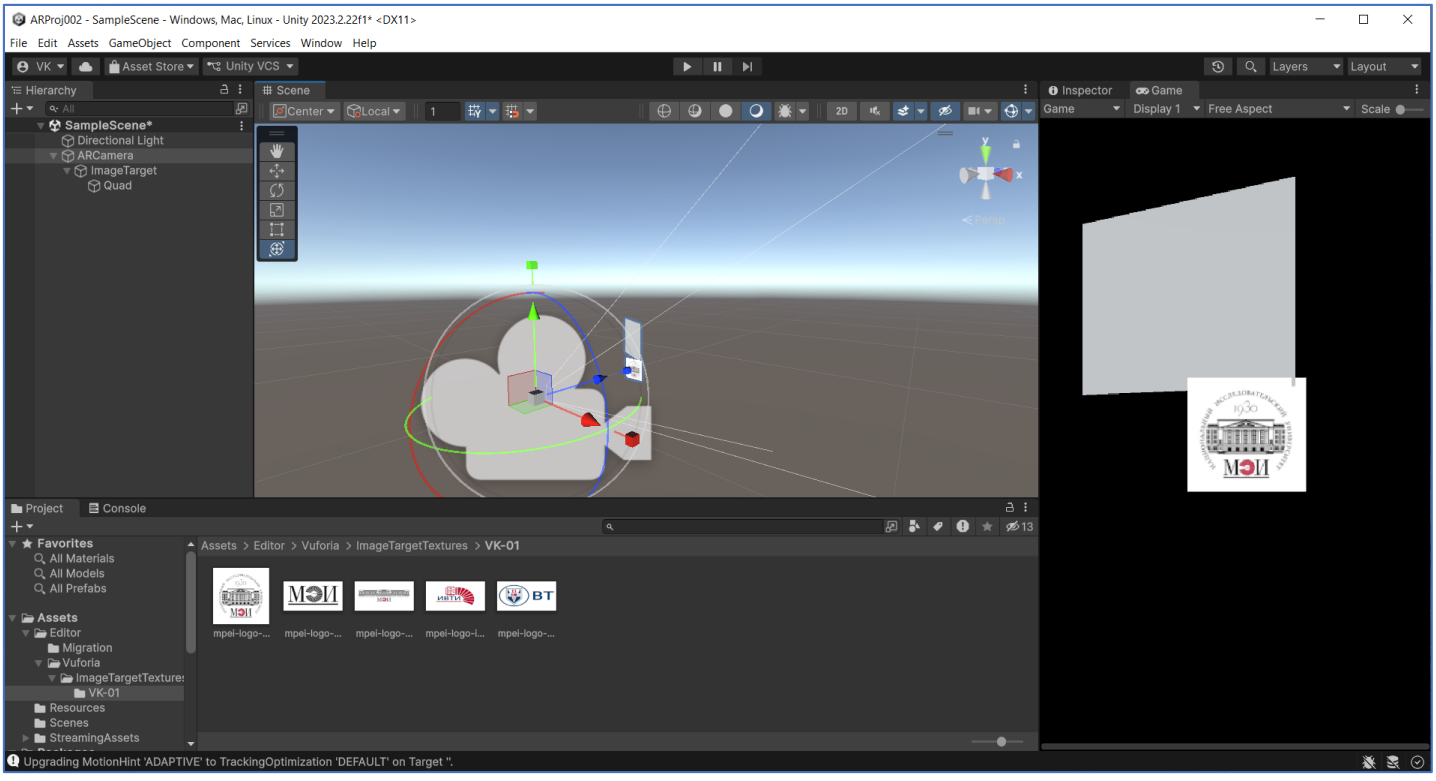


То, что вы видите означает, что **Quad** расположен перпендикулярно оси камеры, в поле зрения камеры, но немного дальше или от камеры, чем таргет. Добейтесь такого расположения квада и таргета, чтобы **Quad** для изображения был расположен на таком же расстоянии от камеры, что и таргет, но с небольшим разворотом по вертикали (ось «Y») и сдвинут в сторону, но в поле зрения камеры (см. ниже):



Таким образом, в сцене размещены **ARCamera**, таргет и контейнер для 2D-контента: Изображения.

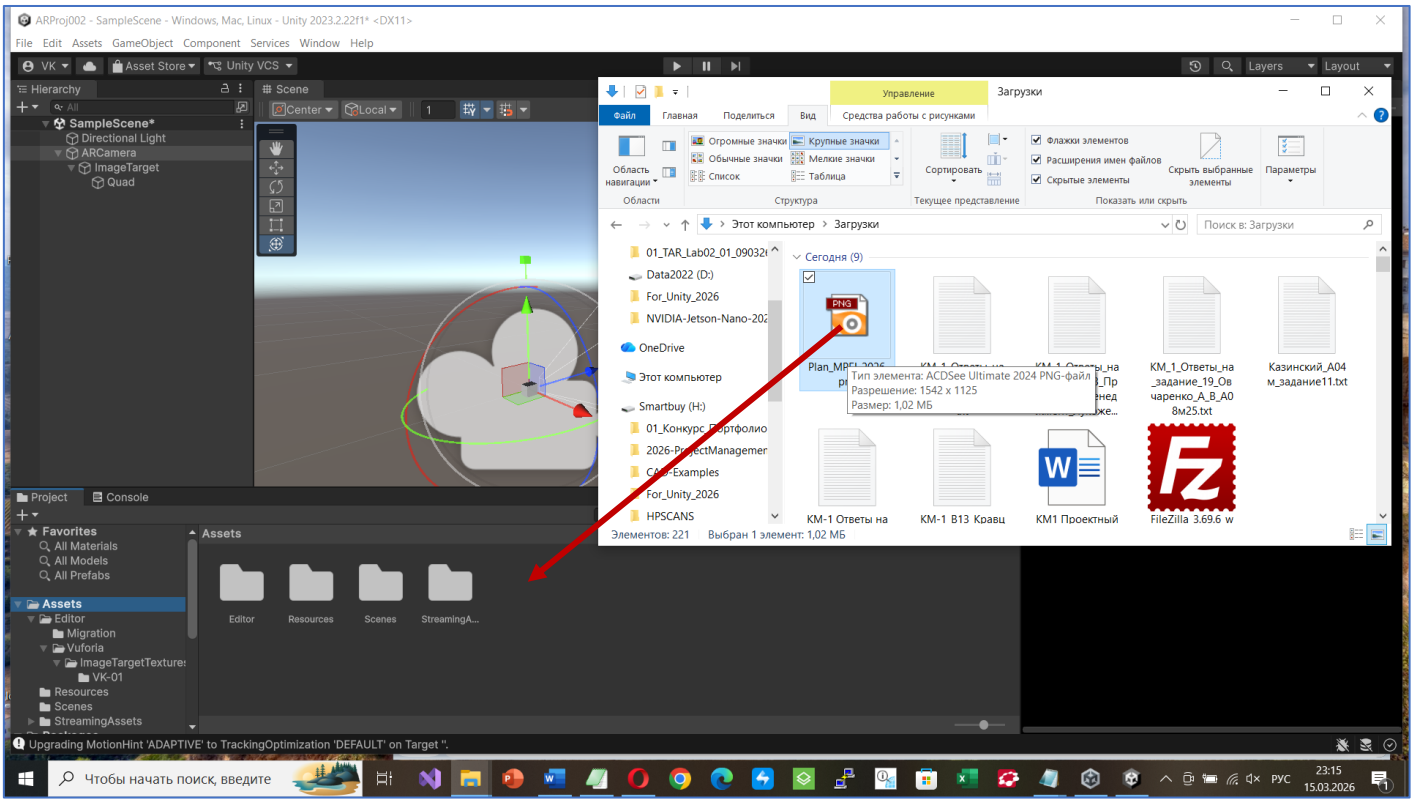
ВАЖНО!! Теперь необходимо корректно сформировать иерархию - корректная работа Приложения ДР обеспечивается в том числе и подчиненным расположением **Quad** относительно **Image Target** в иерархии **(Hierarchy)**. А **Image Target** занимает подчиненное положение относительно **ARCamera** в иерархии **(Hierarchy)**. Это означает, что контент, связанный с **Quad** будет визуализироваться только после распознавания камерой таргета.

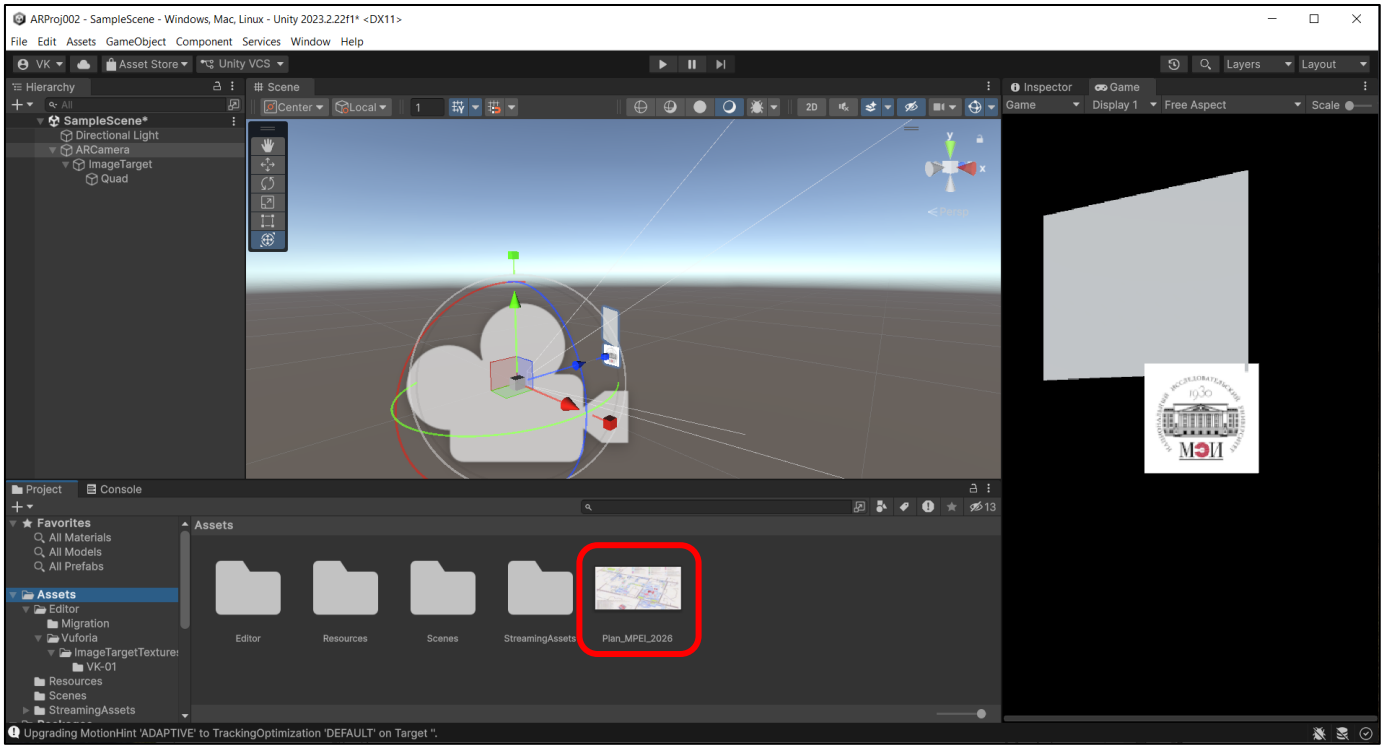


2. Переходим к подготовке контента для помещения его в контейнер Quad.

Контент в данном случае - это визуализируемый объект, помещаемый в контейнер **Quad**. В данной ЛР в качестве визуализируемого объекта будем использовать **плоское изображение** в одном из допустимых для **Unity 3D** форматов **- .png**.

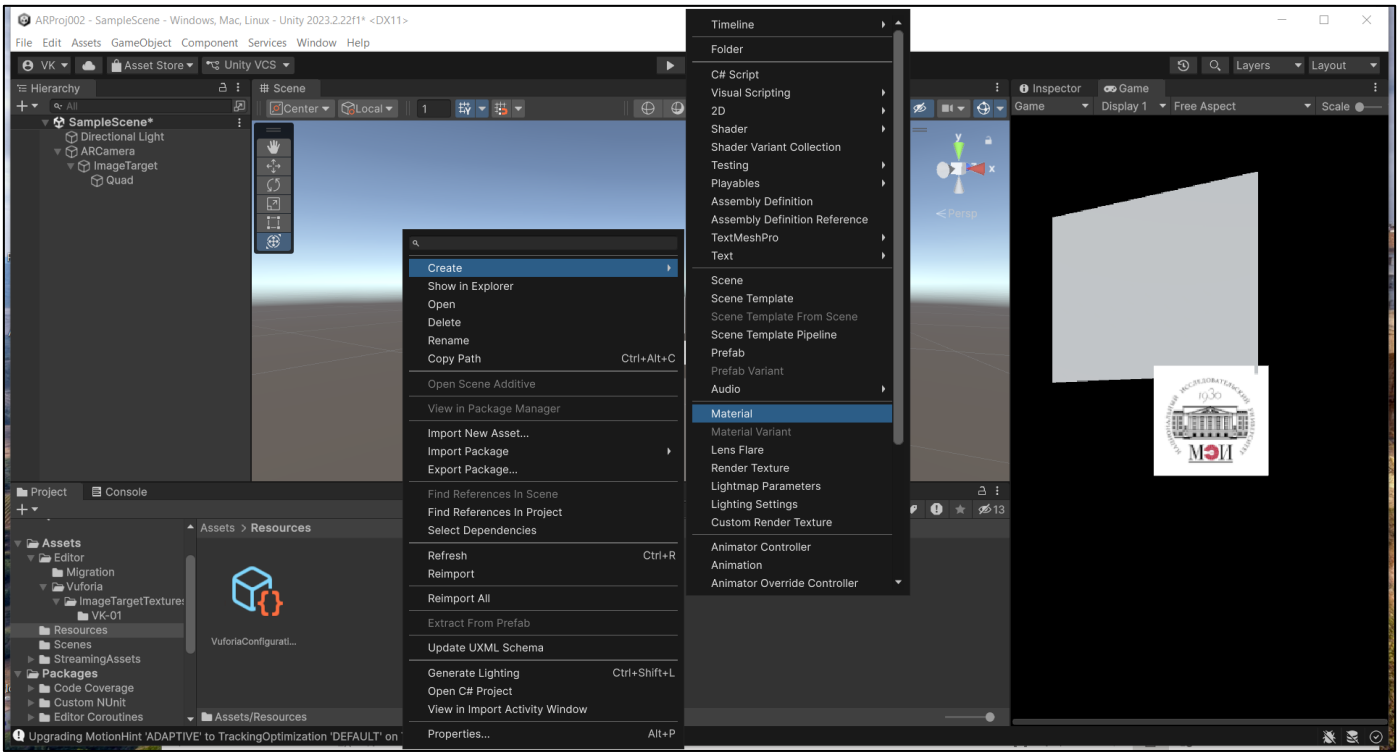
Загрузим наше **2D** Изображение (у нас – это **Plan_MPEI_2026.png**) из локальной файловой структуры (ЛФС) в область **Project**, в раздел **Assets**. Загружаемое изображение в логике Unity 3D – это ресурсы. Исходя из этой логики рекомендуется загружать изображения из ЛФС в раздел **Assets** в папку **Resources** (с помощью операции **Drag-n-Drop**). Но можно воспользоваться другими папками раздела **Assets**, уже содержащими контент такого формата – см. поясняющие рисунки ниже. В данном случае мы размещаем **2D**-изображение в папку нашего проекта, содержащую изображения для **Image Target.**



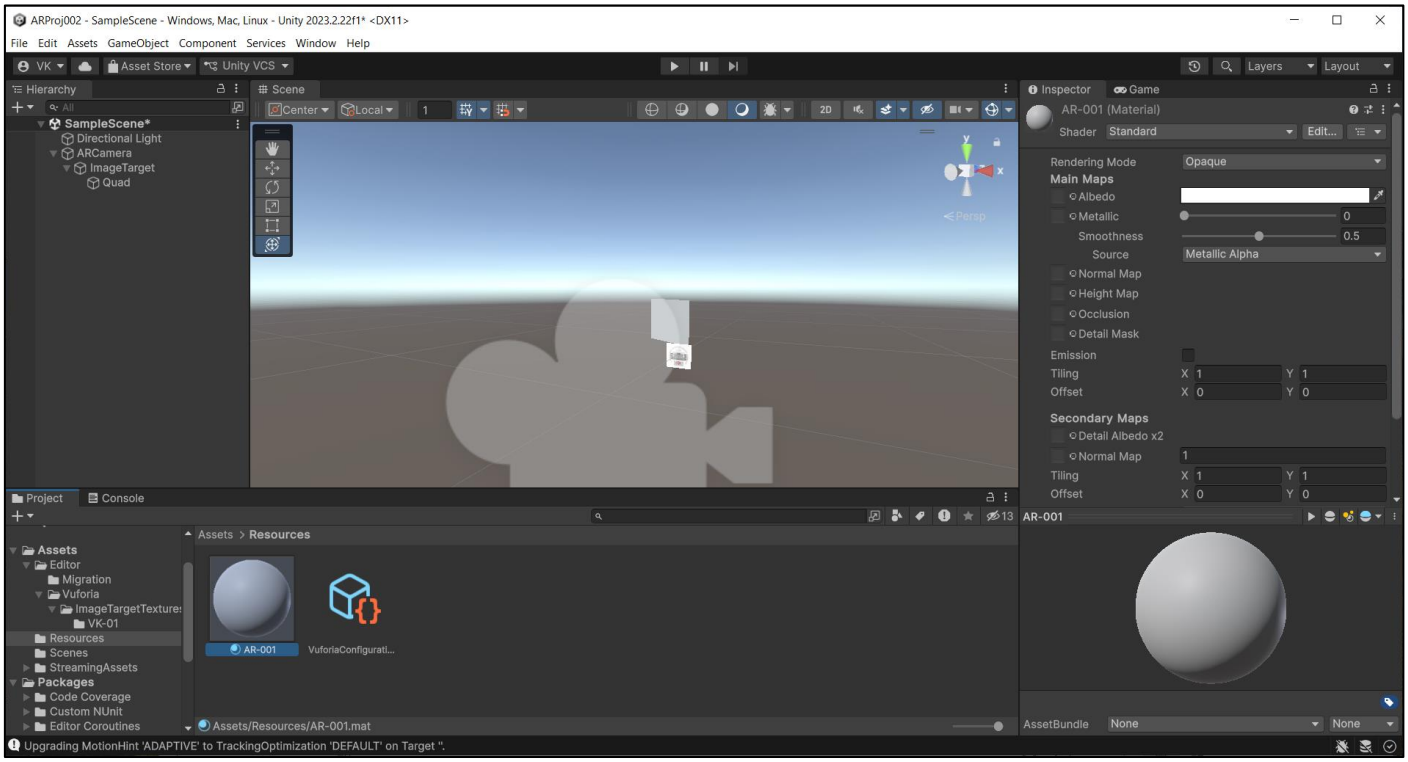


В отличие от первой части ЛР № 2 (замещающий виртуальный объект – видеоклип), при заполнении контейнера **Quad** конкретным значением из файла **.png**, необходимо использовать особый тип **asset'a** (актива) **Unity 3D**, который называется **Material** («материал»). Именно он будет являться контентом в данном случае.

В поле **Project** редактора **Unity 3D** выбираем папку **Resources**, переходим в поле **Assets-Resources**, в пустом пространстве кликаем правой клавишей мыши и «создаем» (элемент меню **Create**) новый **Material**:

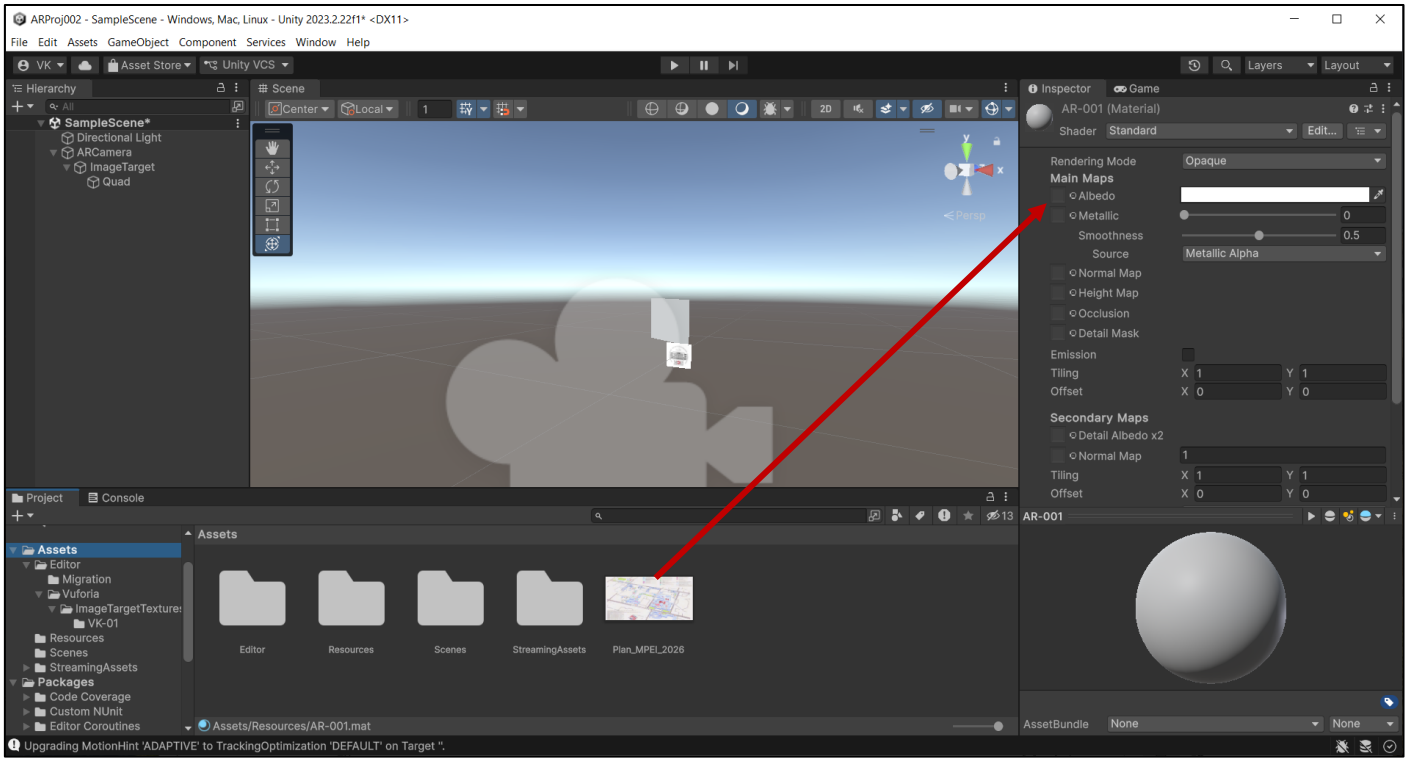


В результате в **Resources** будет добавлен новый **Material**, которому в поле **Assets** – **Resources**, под его иконкой, присвоим уникальное имя, например – **AR-001** (или **AR-002**) вместо **New Material**:



При этом в поле **Inspector** отображается вся информация, связанная с данным «материалом». Воспользуемся полем **Inspector,** для того, чтобы **ассоциировать** «материал» с **2D**-изображением, хранящимся в файле **.png,** т.е. получить требуемый в данном случае контент.

Для этого начинаем работать с основными параметрами материала, которые отображены в **Inspector'е** в различных полях. В нашем случае, когда «материал» - это всего лишь изображение, нам важен только один его параметр (см. документацию по **Unity 3D**) - **отражательная способность материала**. Этому параметру соответствует поле **Albedo.** С помощью операции **Drag-n-Drop** помещаем в это поле ассет с файлом **Plan_MPEI_2026.png.**

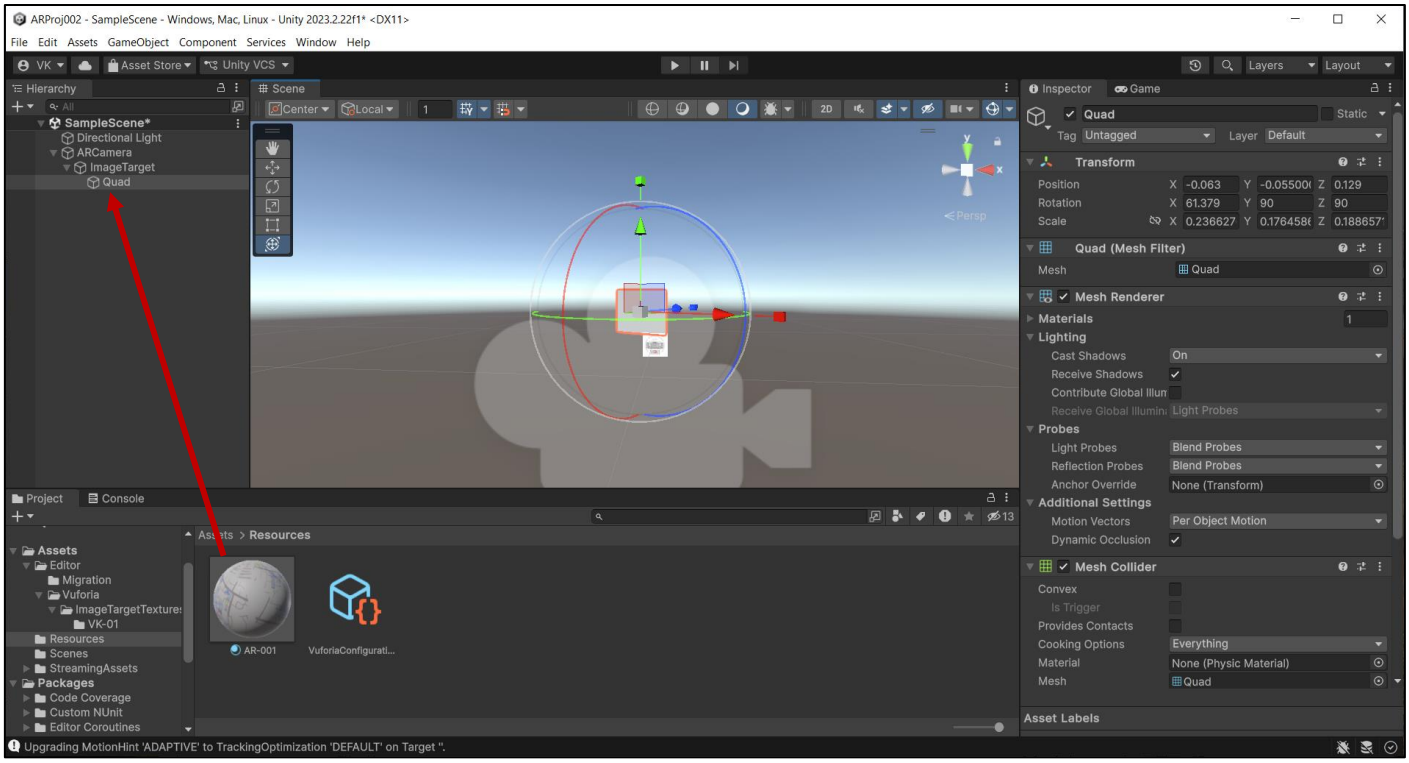


В результате в нижней части **Inspector'a** появляется интерактивный индикатор (Шарик), который свидетельствует об успешности ассоциирования ассета «материал» с ассетом «изображение».

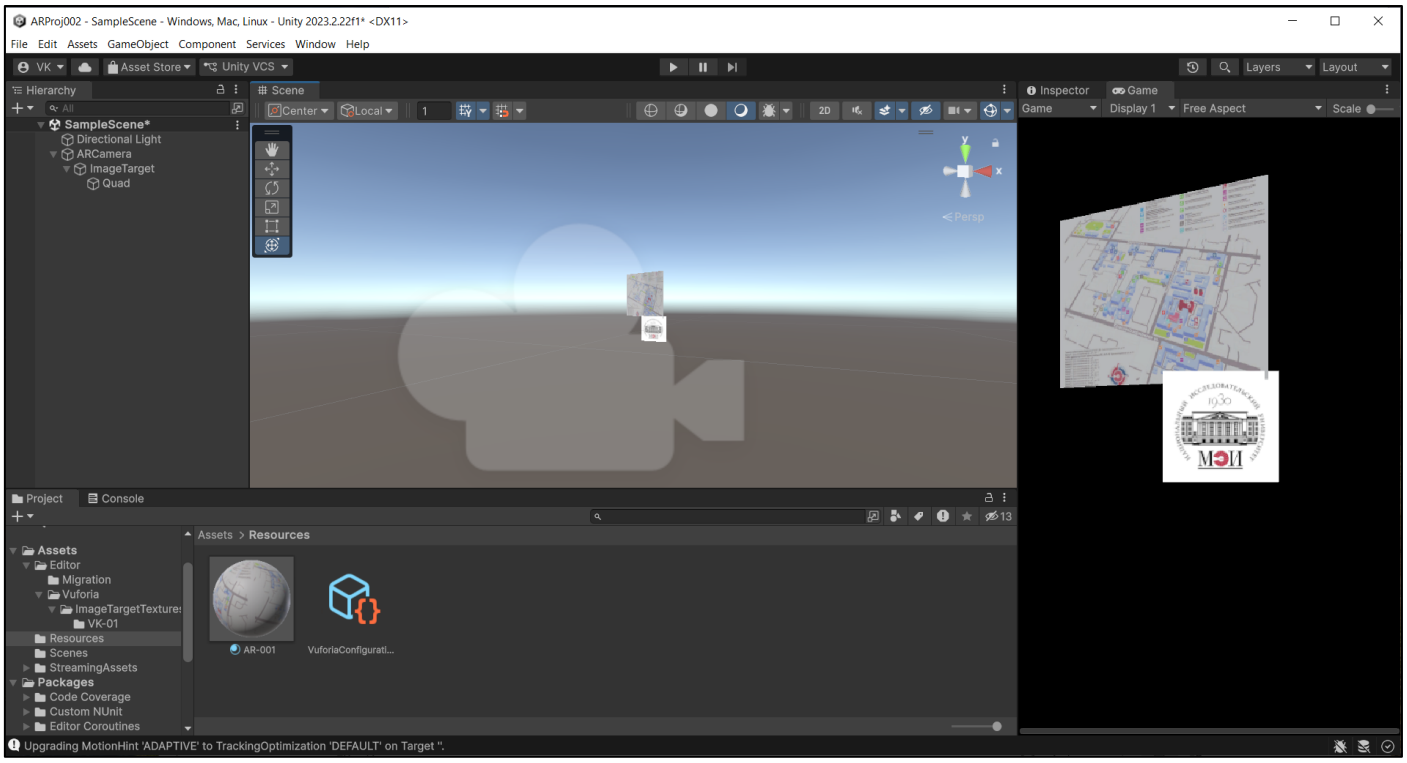
Переходим к загрузке контента (файл .png) в контейнер (Quad).

На данном шаге мы имеем подготовленный контент, представляющий из себя **Material**, содержащий требуемое **2D**-Изображение, и контейнер – объект **Quad**. Для загрузки контента в контейнер переместим с помощью операции **Drag-n-Drop** наш **Material** по имени **AR-001**, находящийся в поле **Project → Assets → Resources,** в **Quad,** расположенный в поле **Hierarchy**.

ВАЖНО!! Для лучшего восприятия изображения позаботьтесь о том, чтобы соотношение сторон у **Quad** было таким же, как соотношение сторон у загружаемого изображения. В нашем случае:



В результате контейнер оказывается загружен нужным контентом, что можно проконтролировать в поле **Game** редактора **Unity 3D.**



Разрабатываемое приложение должно будет использовать камеру конкретного **Android-**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.

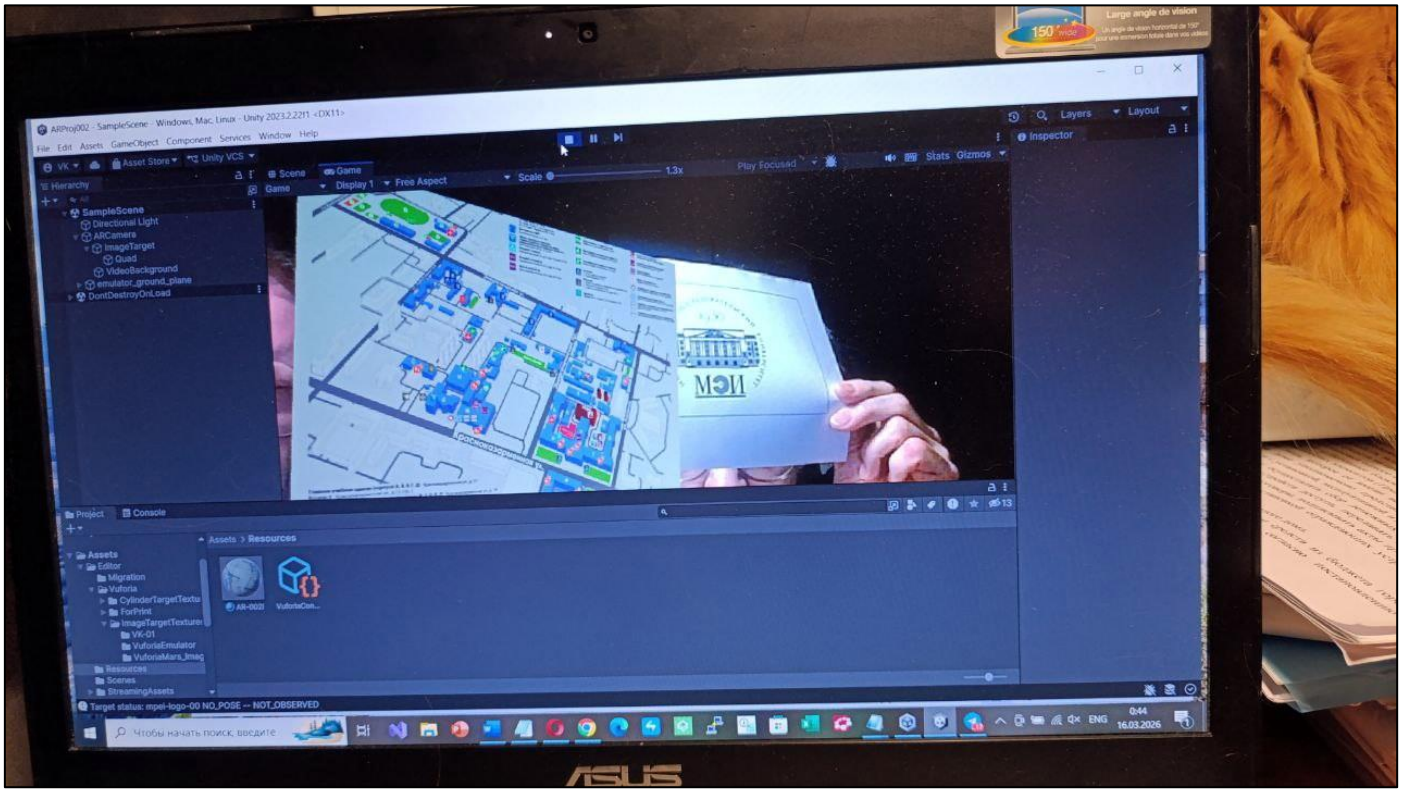
Таким образом, после проведенных манипуляций сцена нашего проекта содержит камеру ДР (**AR Camera**) для типового **Android**-устройства, метку (**Image Target**), контейнер (**Quad**) для 2D-контента, связанный с меткой.

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

Для этого в окне **Unity 3D** необходимо войти в режим **Play** – найти и нажать на клавишу проигрывания **(«Play»)** в верхней части окна, как показано на рисунке ниже:

Тест: в область просмотра камеры локальной машины помещается твердая копия выбранного для данной сцены таргета:

Тест: по выбранному таргету автоматически воспроизводится объект дополненной реальности. В данном случае – это изображение - **Plan_MPEI_2026.png**:



Разработка сцены закончена. 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.**

Все установки, настройки и манипуляции с окнами и функциями **Unity 3D** для создания файла **.apk** идентичны тем, что вы выполняли в ЛР№2, часть 1. Не забудьте снабдить свое загружаемое на МУ Приложение иконкой!

Сохраняем собранный файл **.apk** в локальной файловой системе разработчика, загружаем и устанавливаем его на **Android -** МУ любым известным вам способом.

Видео разработанного в данной ЛР одного из Приложений ДР (различный контент) доступно по ссылке:

https://youtu.be/XPAkxZAET98

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

Не забудьте про таргет!