AR using Unity and Vuforia

Building a simple Augmented Reality (AR) App using Unity and Vuforia

Workshop
Part 2 - AR Marker Interaction

Linnéuniversitetet

4ME306

Cross-Media Design and Production

Part 2 - AR Marker Interaction : Description

Description

During this workshop, you will extend your application based on the progress made in the previous workshop Part 1: Setup. You will implement a simple AR marker logic, enabling two AR markers to interact. You will get familiar using the C# programming language.

Software (versions)

Unity3D (2019.2.17f1 from18 Dec, 2019) using Unity Hub (2.2.2)

Visual Studio IDE

Windows users: Visual Studio (Community 2019)

Mac OS users: Visual Studio 2019 for Mac (Community, v.8)

Vuforia (integrated natively in Unity3D 2019.2.17)

Attention

This activity is intended to be completed in self-study.

Part 2 - AR Marker Interaction : Outline

This workshop activity is divided into 3 milestones.

Milestone 1: Material and Color

Create material and color for the virtual object of your AR content marker.

Milestone 2: AR Marker Interaction - Scale up

Setup of the interaction between the AR content and AR plus markers according to the provided example and scripts.

Milestone 3: Implementation Challenge
(AR Marker Interaction - Scale down)

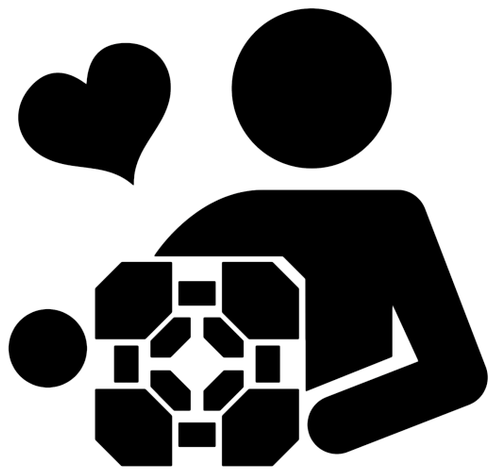
Implementation of interaction between AR content and AR minus markers, according to the logic presented in milestone 2.

Part 2 - AR Marker Interaction : Deliverable

Submit a **.zip archive** named **4me306vt20-ar2-LASTNAME_FIRSTNAME.zip** in the corresponding assignment submission activity on the course's Moodle page.

The submitted .zip folder should contain **three deliverables**:

1. A **screenshot** named **4me306vt20-ar2-LASTNAME_FIRSTNAME.jpg** showing your Unity3D environment, the AR application running, and all three AR markers.
2. A short **movie clip** named **4me306vt20-ar2-LASTNAME_FIRSTNAME.mp4** presenting a screen recording of your Unity3D environment, the AR application running, and the implemented interaction (milestones 2 and 3) between the AR markers.
3. The **ImageTargetDistanceScaleManager.cs C# script** from your project's Assets/_scripts folder containing your implementation of milestone 3.



Attention

Follow precisely the above stated file naming conventions.
The submitted .zip folder should not exceed a maximum file size of 10 MB.

Part 2 - AR Marker Interaction : Deliverable Hints

Example submission

Feel free to download and inspect the following .zip archive (~ 2.5 MB) representing an example submission:

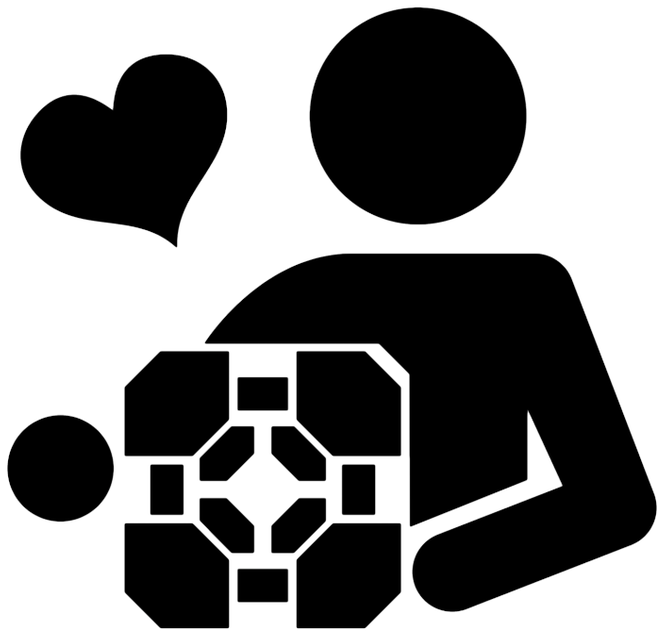
https://vrxar.lnu.se/downloads/4me306/4me306vt20-ar2-RESKI_NICO.zip

Helpful screen recording resources

(Mac OS X) Screen recording via QuickTime Player

(Win) Windows 10 built-in screen capture tool (via PCMag UK)

(Win & Mac OS X) Open Broadcaster Software



Helpful video converter resources

(Win & Mac OS X) Handbrake*

* For your submission it is sufficient to use e.g. the Handbrake preset **General** - **Fast 480p30** using the **video encoder H.264 (x264)**.

Part 2 - AR Marker Interaction : Assets Download

During this workshop, you will need some additional assets.

Please **download** the provided **4me306_ar_scripts.zip** archive using the following url:

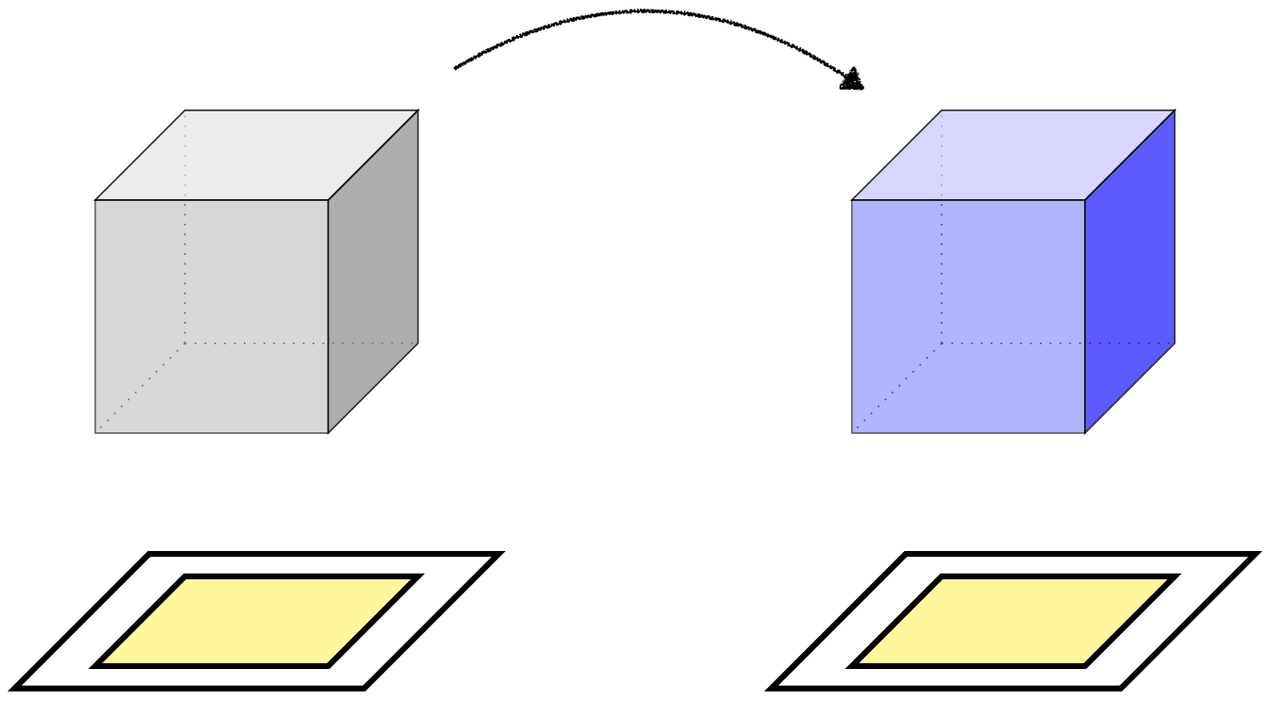
https://vrxar.lnu.se/downloads/4me306/4me306_ar_scripts.zip

Upon **extraction** of this .zip archive, you will find the following C# scripts:

- ImageTargetDistanceScaleManager.cs
- ImageTargetTracker.cs

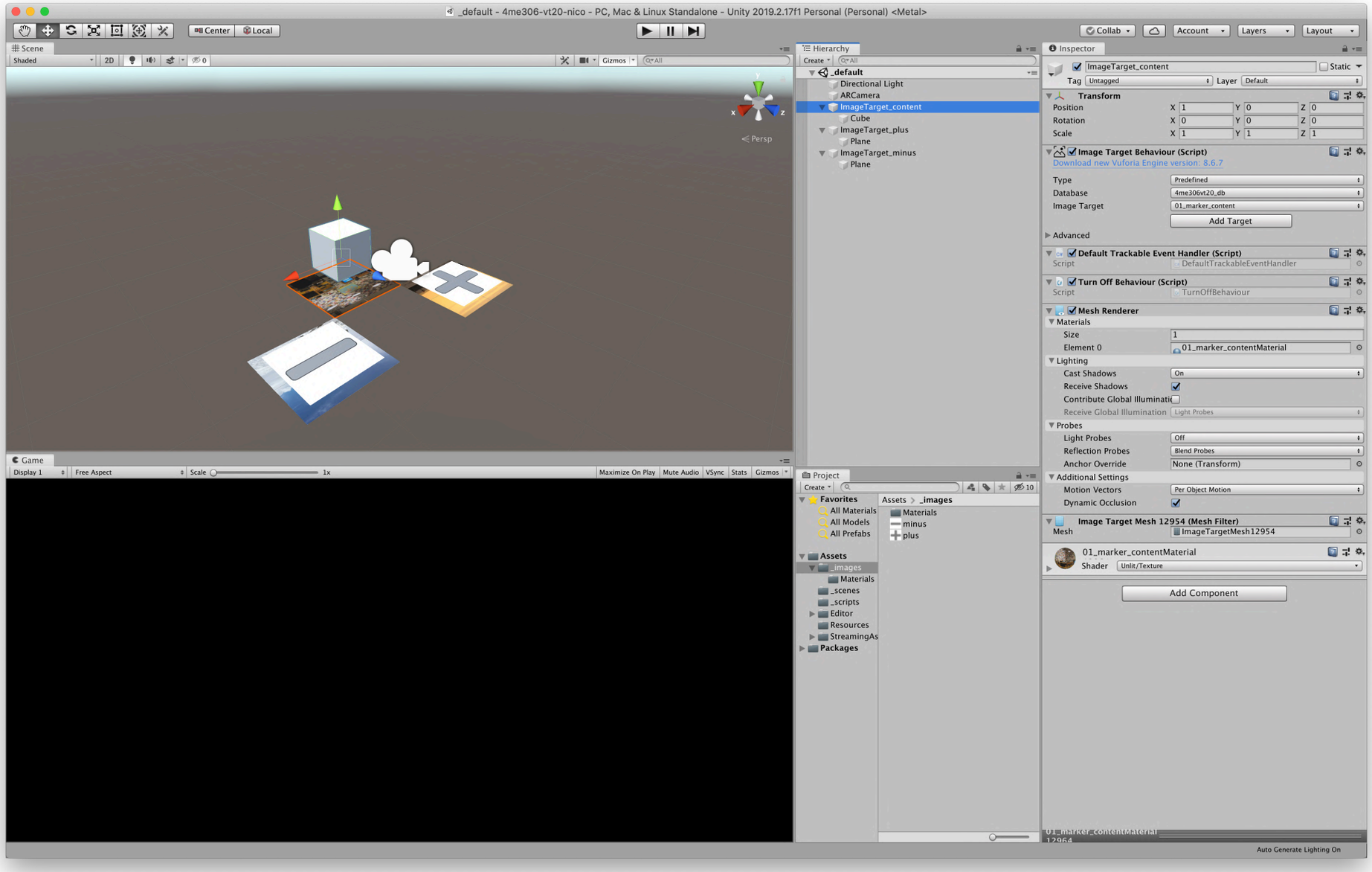
Milestone 1

Material and Color



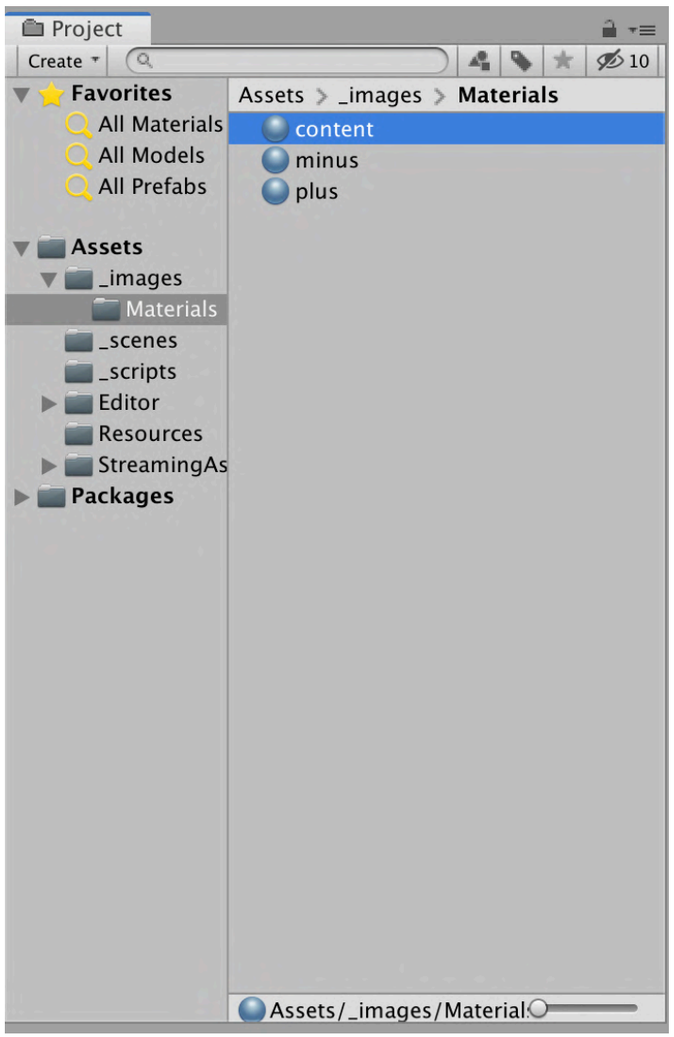
1.1 Open Unity Project

Diving right into the work of workshop 2, please open Unity and load your Unity project based on your completion of the previous workshop (Part 1: Setup).



1.2 Create Material

Navigate to your **Assets/_images/Materials** folder in the **Project** view, **[right-click]** somewhere in the empty space and select **Create ▸ Material** in order to create a new material. Rename the newly created material file **content**.

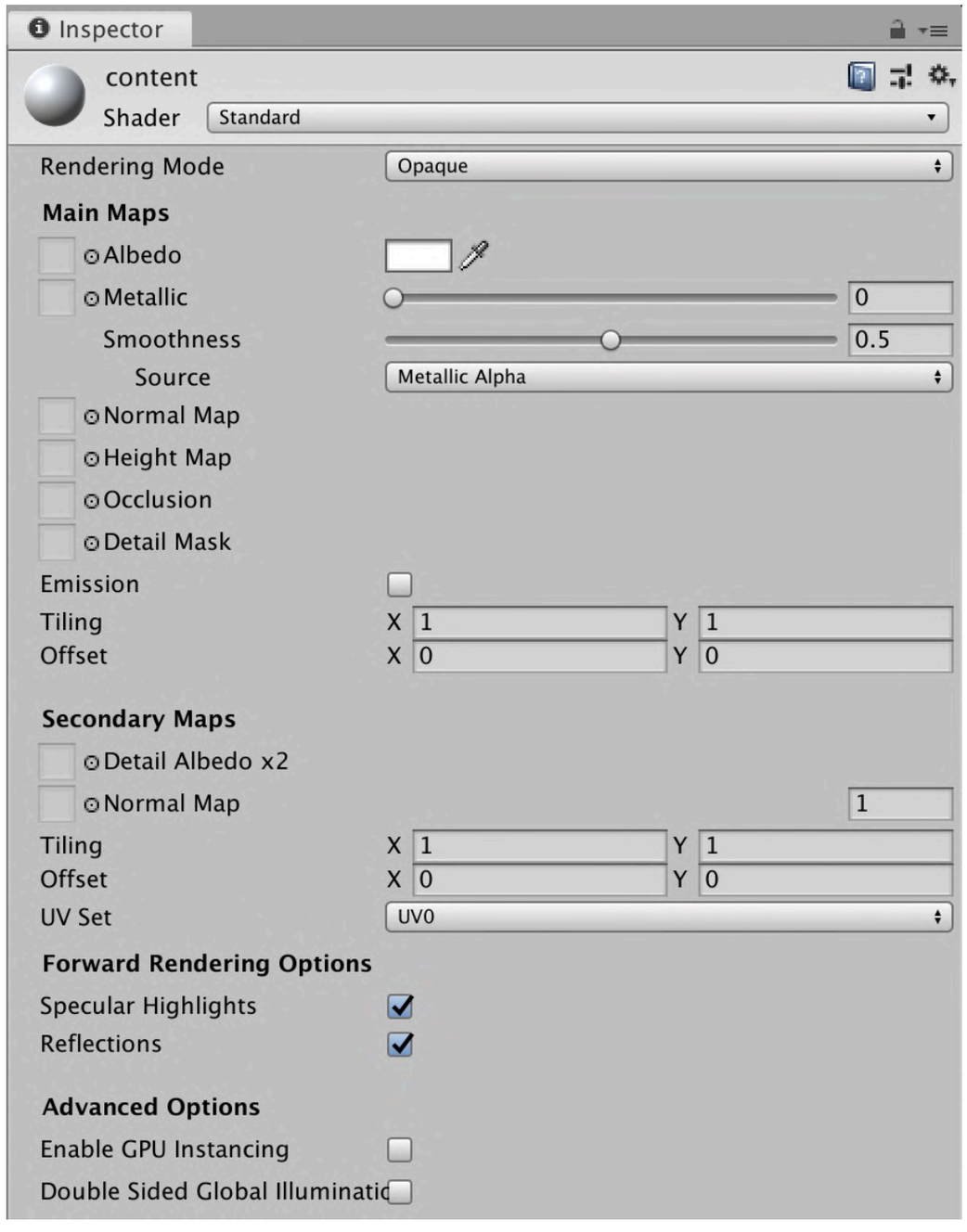


Further Reading

Materials determine how your virtual object (or other GameObjects) will be rendered, meaning how it will “look like”. If you want to learn more about materials, please start investigating at https://docs.unity3d.com/Manual/Materials.html.

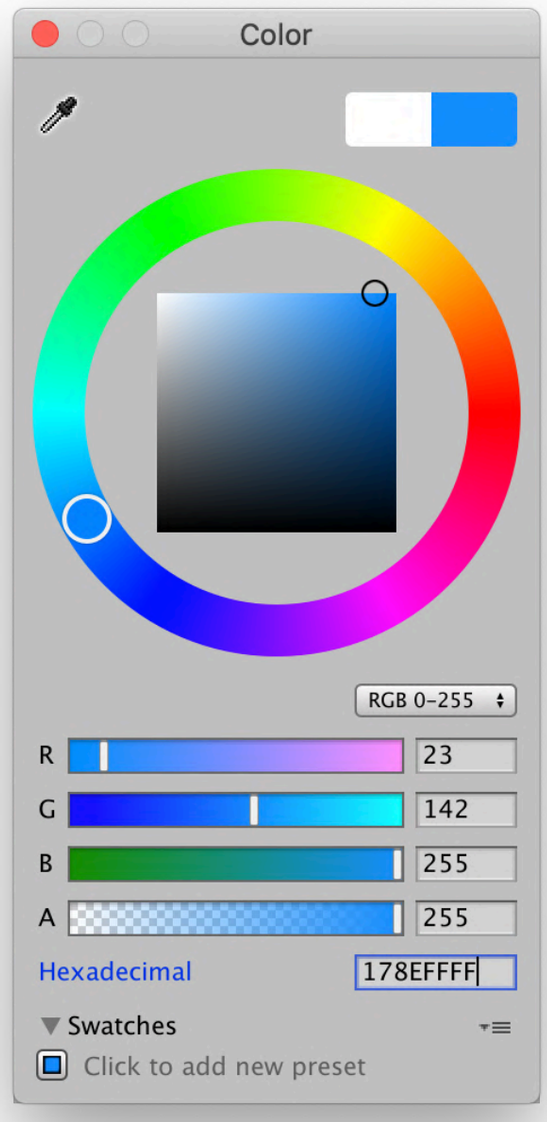
1.2 Create Material

With the **content** material file selected in the **Project** view, have a look at its properties using the **Inspector** view. Although these are the default settings, make sure your **content** material file has the same properties as shown in the screenshot:



1.3 Edit Material Color

Still inspecting the properties of the **content** material file using the **Inspector** view, click in the **white (color) box** next to the **Albedo** entry in the **Main Maps** section. A small pop-up window will appear, enabling you to set the color for the material.

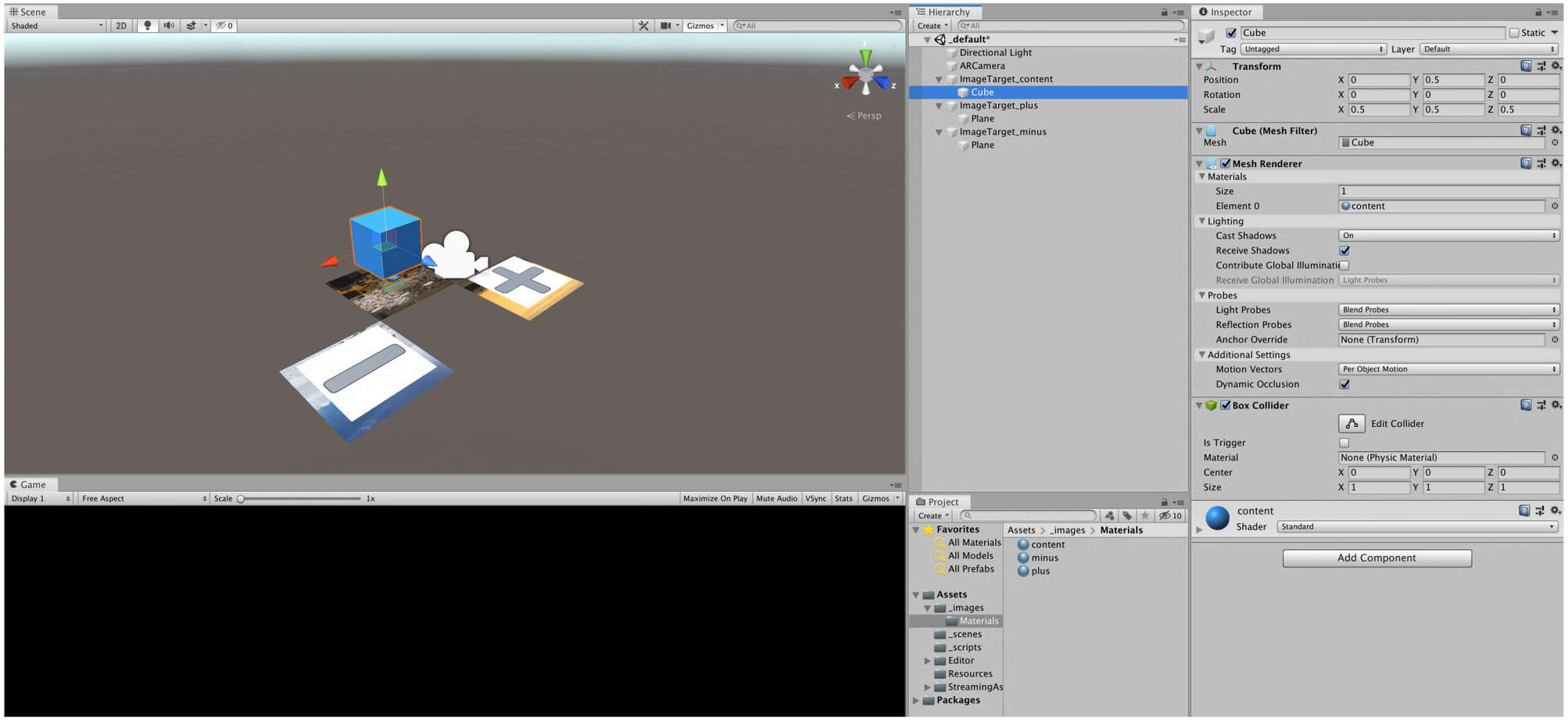


In this example, the color will be set according to the Hex Color code 178EFFFF *.

*** Attention:** 4ME306 students, please have a careful look at the additional information provided in Moodle. Everyone of you will have to set an individual color for this material based on the provided hex color code (instead of the color shown above).

1.4 Set Virtual Object's Material

Next, you will tell your virtual object attached to your **ImageTarget_content** GameObject to use the created material. From the **Project** view, simply **[drag & drop]** the **content** material file onto the **child GameObject** (representing your individual virtual object) of the **ImageTarget_content** GameObject in the **Hierarchy** view. Your virtual object should now be colored accordingly in the **Scene** view.

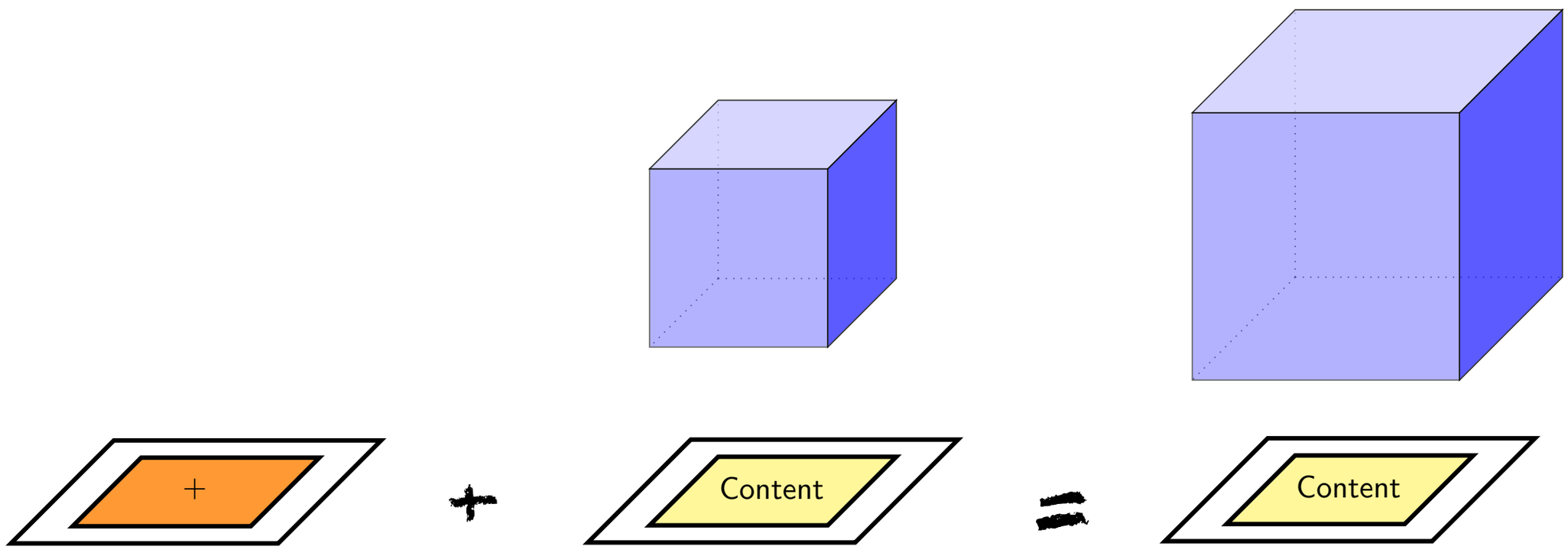


Milestone 1: Complete

Congratulations, you have successfully completed milestone 1 by creating a new material, setting its color, and applying it to your AR marker's virtual object.

Milestone 2

AR Marker Interaction - Scale up



2.1 Marker Interaction Concept

In this milestone, you will learn how to setup a simple interaction between your **ImageTarget_content** and **ImageTarget_plus** GameObjects based on the **provided C# scripts**.

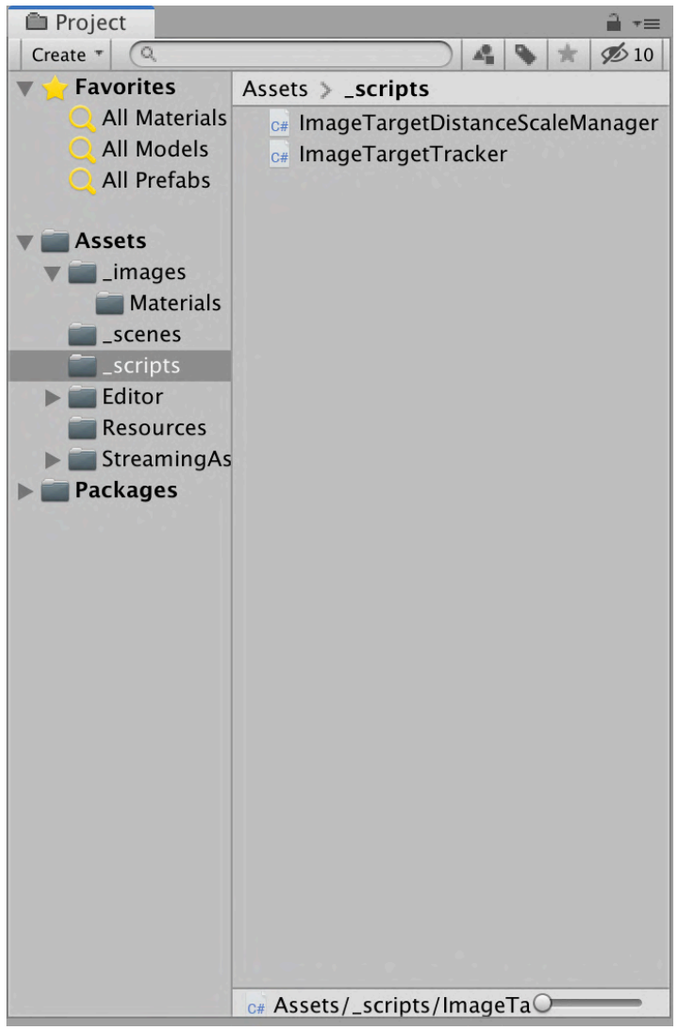
The **concept of the interaction** between these two markers is straight-forward:

1. Once **both AR markers are detected**, the **plus marker** will make the **virtual object** of your content marker **grow** (i.e., increase its size, scale up).
2. Based on the **proximity** (distance) **between the two markers**, the **virtual object** will **grow** either **slower** (both markers are far away from each other, distance = big) or **faster** (both markers are closer to each other, distance = small).

2.2 Import Scripts

Navigate to the **Assets/_scripts** folder in the **Project** view. **[Right-click]** on the **MyFirstUnityScript** file created during workshop Part 1: Setup and **Delete** it (you don't need this anymore).

Next, **[right-click]** somewhere into the empty space of the **Assets/_scripts** folder in the **Project** view, select **Import New Asset...** and add the C# scripts **ImageTargetTracker.cs** and **ImageTargetDistanceScaleManager.cs** from the downloaded **4me306_ar_scripts.zip** archive.



2.3 ImageTargetTracker.cs

[Double-click] the **ImageTargetTracker.cs** script in the **Project** view, which will open your **Visual Studio** editor, enabling you to examine the script in more detail.

Following, some simplified explanations regarding the **ImageTargetTracker.cs** C# script are stated. Once this script is attached to an **ImageTarget** GameObject in your project, it will **register and identify** itself as part of **Vuforia's platform tools**. Particularly, the script will **check** if the **ImageTarget** was **successfully detected** or if its **detection was lost** by the camera. A public boolean property named **isTracked** indicates this status at runtime.

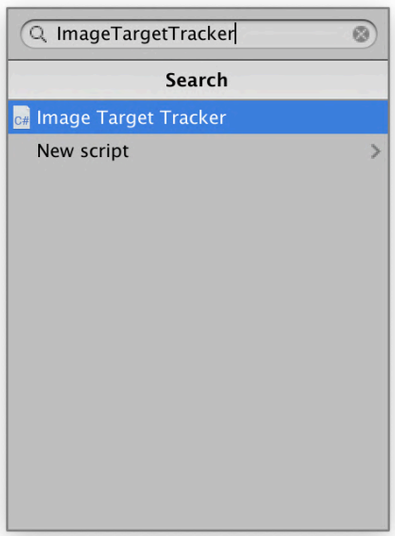
Take a moment to read through the script. The comments in the source code, indicated by a starting **//**, should help you to understand what is going on.

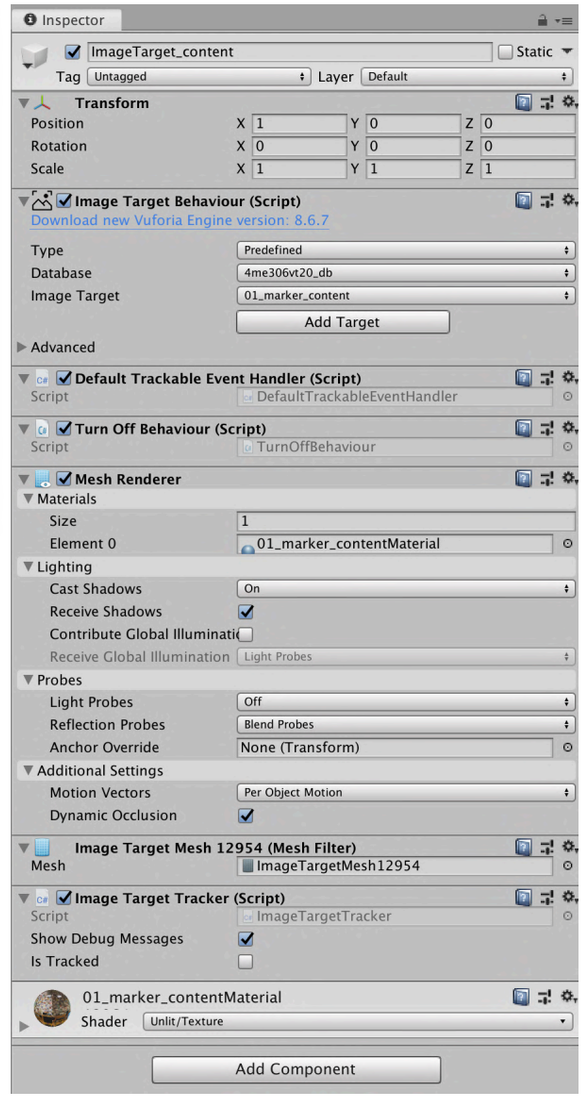
Further Reading
If you want to learn more about C# and scripting (programming) in Unity3D, please start investigating at https://learn.unity.com/project/beginner-gameplay-scripting.

2.3 ImageTargetTracker.cs

In order to start establishing the logic between your AR content and plus markers, you need to know about each marker's tracking state at runtime. Luckily, you have the **ImageTargetTracker.cs** script to take care of this task.

Click on your **ImageTarget_content** GameObject in the **Hierarchy** view, so its properties are displayed in the **Inspector** view. In the **Inspector** view, click **[Add Component]** and type **ImageTargetTracker**, which will show your imported **ImageTargetTracker** C# script in the **Search results**. Attach the script by clicking on the according entry in the **Search results**.

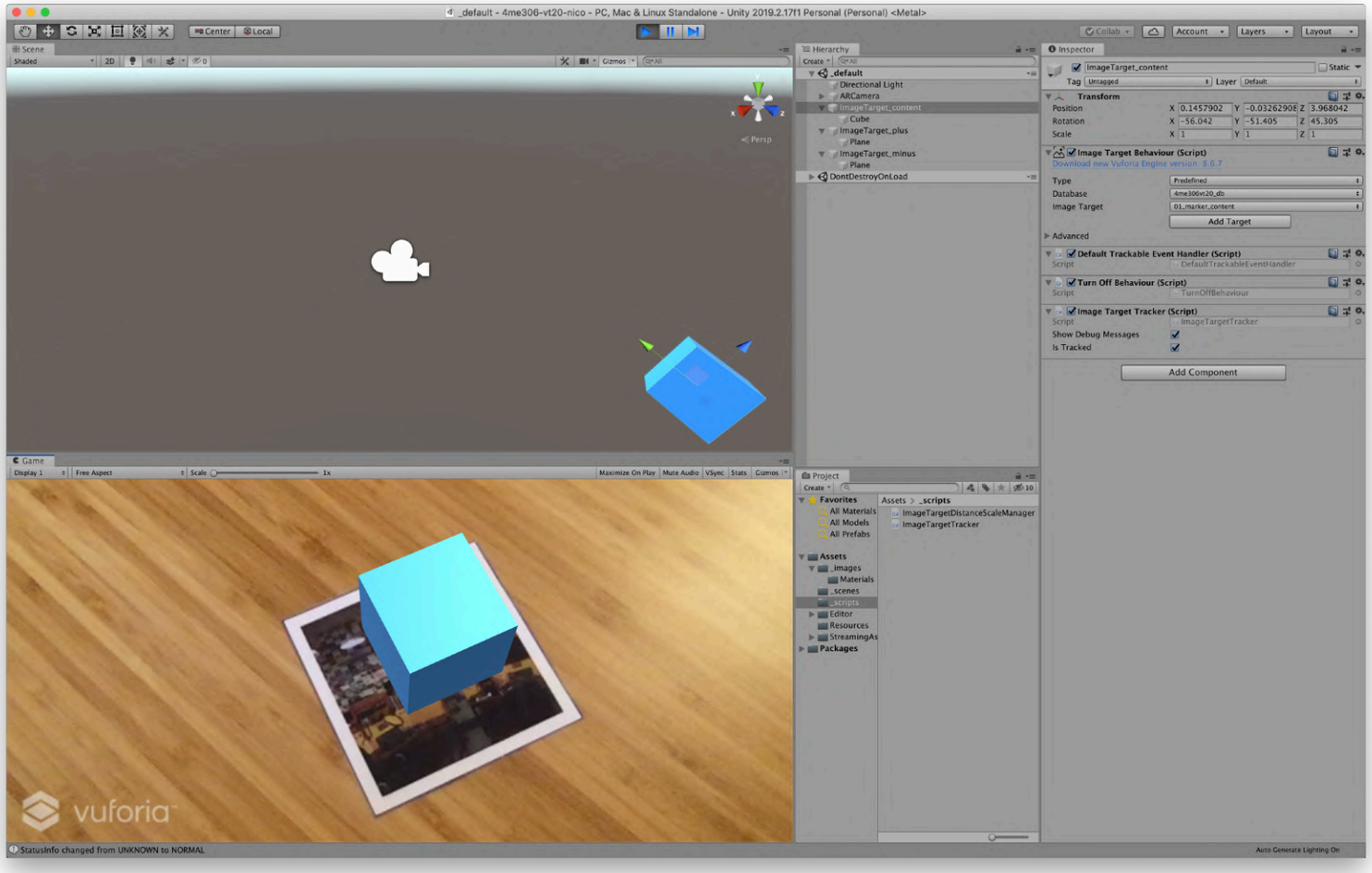




Repeat this step to attach the **ImageTargetTracker** C# script to your **ImageTarget_plus** GameObject.

2.3 ImageTargetTracker.cs

Let’s check if the script works and does its job. With the **ImageTarget_content** GameObject selected in the **Hierarchy** view, click the **[Play]** button (▶) in the top center of the interface to **run** your application. Once your **physical 01_marker_content AR marker** is detected, the **isTracked check box** in the **Inspector** view should automatically change to **being checked**, while it is **unchecked** if the AR marker is not detected (anymore).



2.4 ImageTargetDistanceScaleManager.cs

[Double-click] the **ImageTargetDistanceScaleManager.cs** script in the **Project** view, which will open your **Visual Studio** editor, enabling you to examine the script in more detail.

This script is a bit more advanced compared to the previous one. However, some simplified explanations regarding the **ImageTargetDistanceScaleManager.cs** C# script are stated in the next slides.

Take a moment to read through the script. The comments in the source code, indicated by a starting **//**, should help you to understand what is going on.

Further Reading

If you want to learn more about C# and scripting (programming) in Unity3D, please start investigating at https://learn.unity.com/project/beginner-gameplay-scripting.

2.4 ImageTargetDistanceScaleManager.cs

In the beginning of the script, you will find **three public properties** (2x of type **ImageTargetTracker**; 1x of type **Transform**). These will establish references (“links”) to the **content** and **plus AR markers** as well as the **virtual object** in your scene.

Furthermore, you will find a **Vector3** property, which will save the **original scale** of your **virtual object** at start, as well as a property of type **float** indicating the **maximum scale** of your **virtual object** (you don’t want it to grow “infinitively”). The value of **2.0** indicates that the virtual object can **grow twice as big as its original scale**, but not beyond that.

The **Start()** function takes care of checking if all references are set up properly, and saves the virtual object’s original scale into the private Vector3 property.

The **Update()** function checks if the **ImageTarget_content** AR marker got detected. If it was detected, it will further check if also the **ImageTarget_plus** AR marker was detected, and in that case also call the functions in order to make the virtual object grow.

2.4 ImageTargetDistanceScaleManager.cs

The **DISTANCE** region of the script contains a function called **distanceContentToPlus()**. This function is responsible for **calculating the distance** between your **content** and **plus AR markers**. If both markers are detected by the camera, it will retrieve each marker's position in the 3D space (indicated by a Vector3 type variable). With both positions located, the function then calculates the distance between these two AR markers and returns the result as a **float** type value.

2.4 ImageTargetDistanceScaleManager.cs

The **SCALING** region of the script contains three functions.

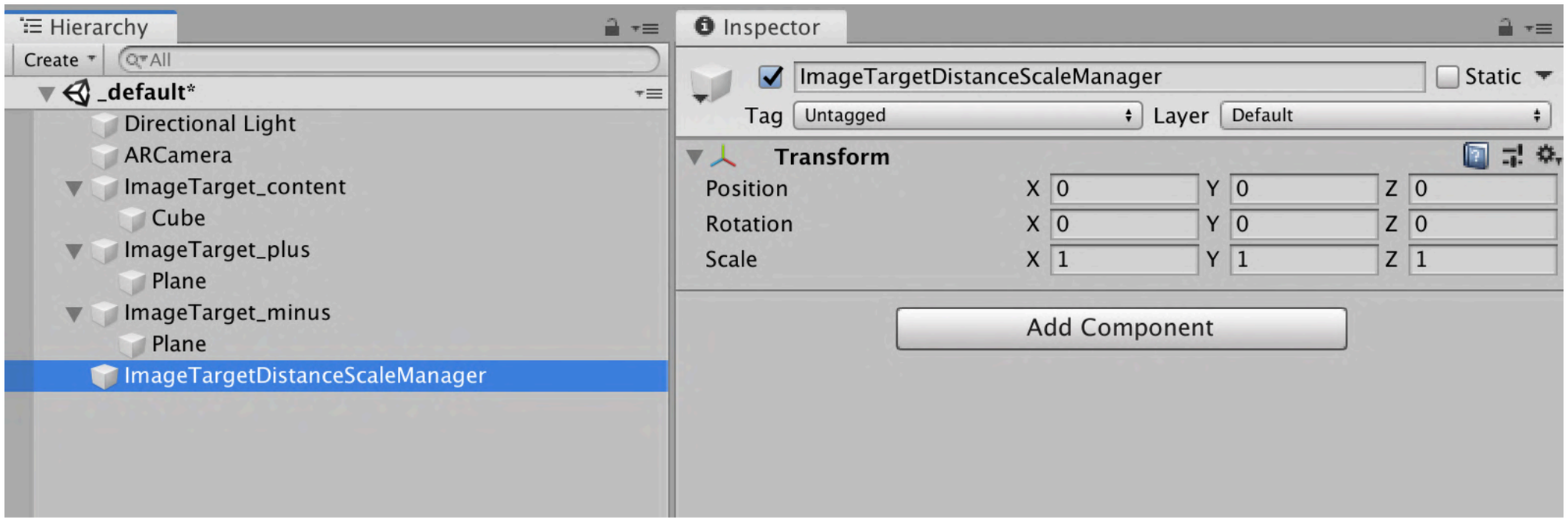
The **calculateScaleMultiplier(float distance)** function is a **helper method**. Based on a provided distance value, it decides an **appropriate scaling factor** that you can use in order to make the virtual object grow (frame-by-frame).

The **scaleUpContent(float distance)** function takes care of the actual **growing** (scaling up) of your virtual object.

The **repositionVirtualObject()** function is another **helper method**. After the scaling of your virtual object, its **position** needs to be **translated** (“moved”), so it will appear to stick to its original position, i.e., floating slightly over the AR marker as set in workshop Part 1: Setup.

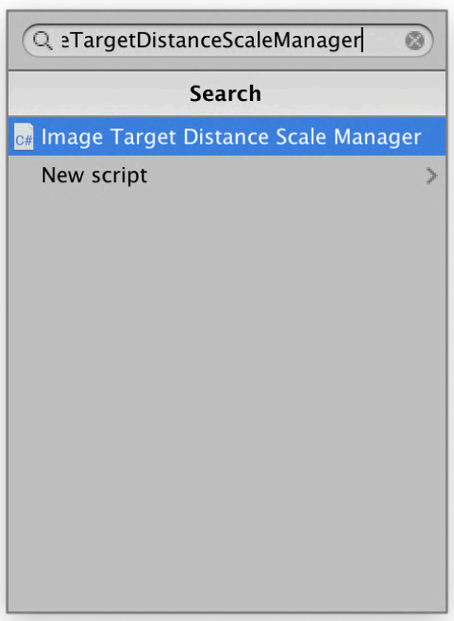
2.4 ImageTargetDistanceScaleManager.cs

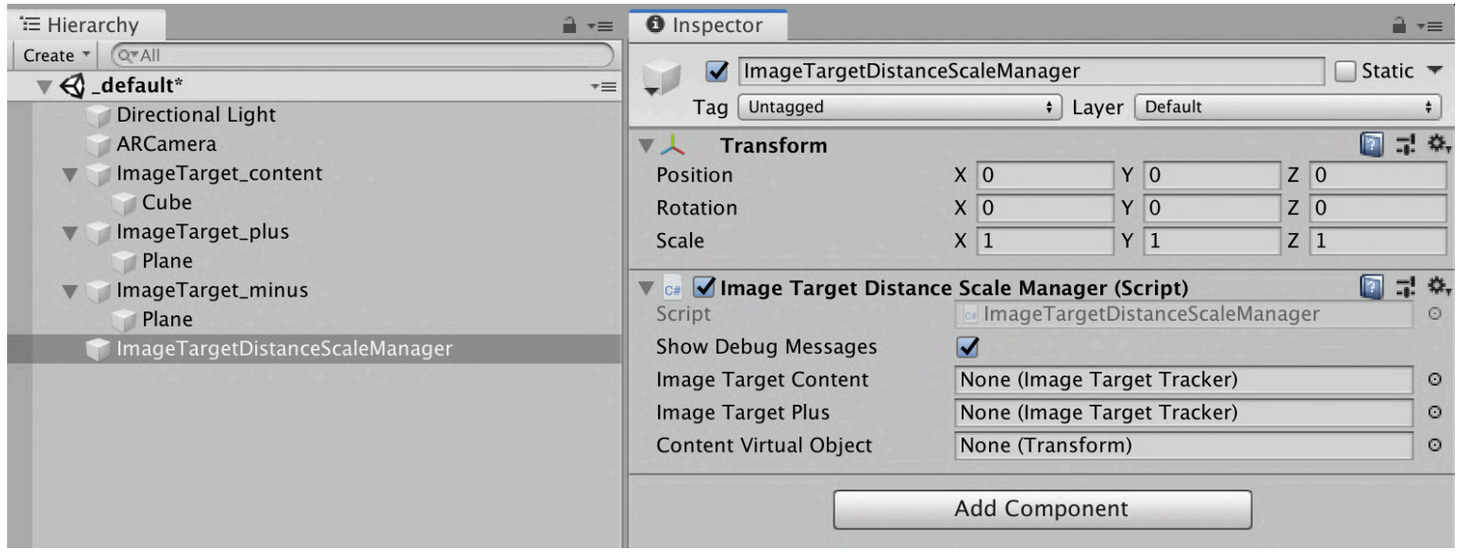
Back in Unity, **[right-click]** somewhere in the empty space of the **Hierarchy** view and select **Create Empty** in order to create a new empty GameObject with nothing attached.
Rename the newly created GameObject to **ImageTargetDistanceScaleManager**.
Furthermore, with this GameObject selected, click on the **small settings wheel (icon)** in its **Transform** section in the **Inspector** view and select **Reset** to reset its **Transform** values.



2.4 ImageTargetDistanceScaleManager.cs

With the **ImageTargetDistanceScaleManager** GameObject selected, click **[Add Component]**, type **ImageTargetDistanceScaleManager** in the **Search** pop-up window and select the **ImageTargetDistanceScaleManager** C# script from the **Search results** in order to attach this script to the GameObject.





2.4 ImageTargetDistanceScaleManager.cs

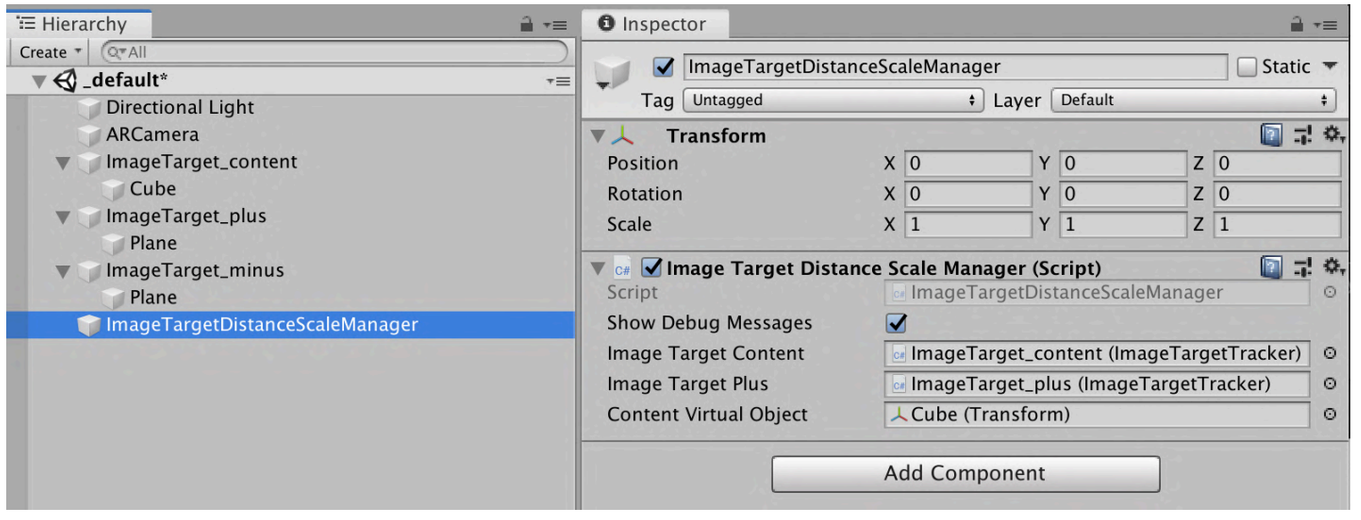
In the **Inspector** view, you will notice that the three public properties **Image Target Content**, **Image Target Plus**, and **Content Virtual Object** of the attached **ImageTarget DistanceScaleManager** script have the value **None**. Let's assign them.

From the **Hierarchy** view, **[drag & drop]** the corresponding **GameObject** onto the **property field** in the **Inspector** view in order to set the references:

ImageTarget_content → Image Target Content

ImageTarget_plus → Image Target Plus

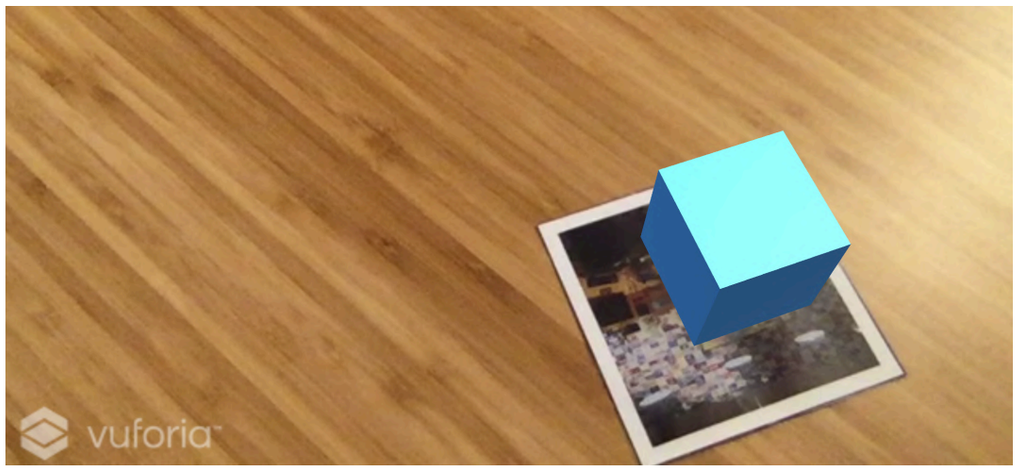
Cube * → Content Virtual Object



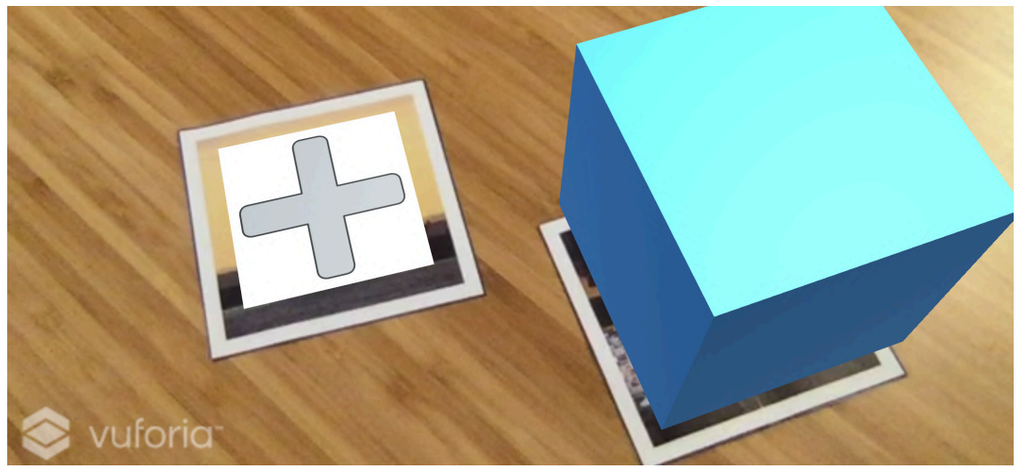
*** Attention:** 4ME306 students, the name of the virtual object varies based on the one you added within workshop Part 1: Setup. Choose the GameObject accordingly.

2.5 AR testing time

Finally, let's check if everything is set up correctly and working appropriately. **Run** your application by clicking the **[Play]** button (▸) in the top center of the interface.



1. Place your physical **01_marker_content AR marker** in the field of view of your camera and make sure it gets **successfully detected**.



2. Place your physical **02_marker_plus AR marker** in the field of view of your camera and make sure it gets **successfully detected**.

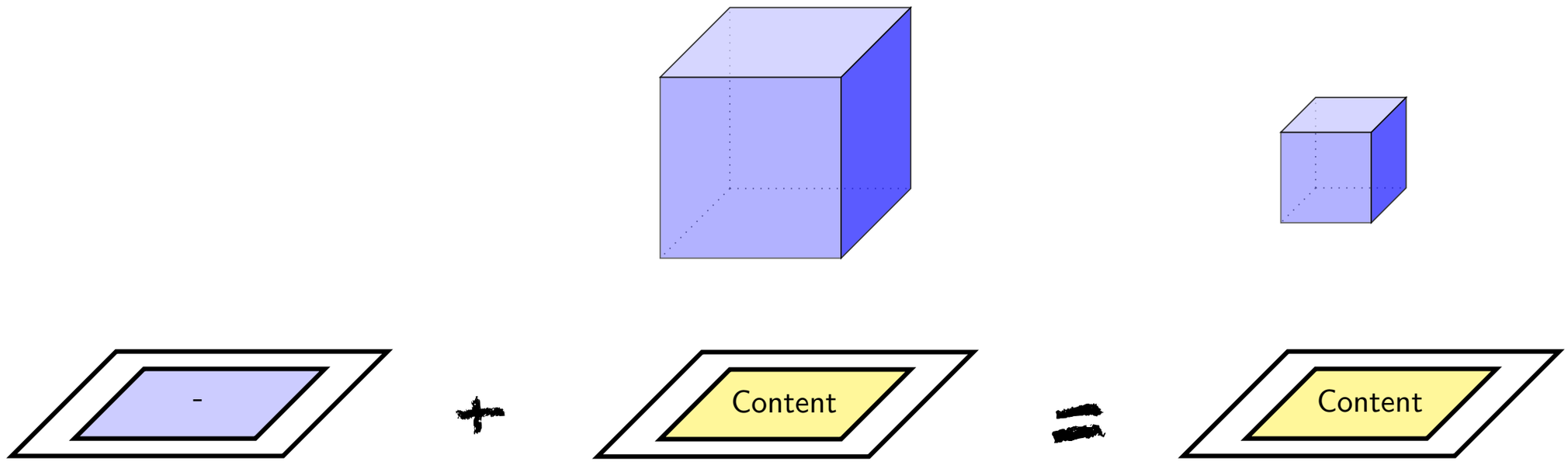
3. Observe the **virtual object grow**.

Milestone 2: Complete

Congratulations, you have successfully completed milestone 2 by importing the provided C# scripts responsible for the AR marker interaction, getting familiar with these C# scripts, and setting up your Unity scene accordingly in order to make the C# scripts work.

Milestone 3

Implementation Challenge
(AR Marker Interaction - Scale down)



3.1 Implementation Challenge

Now it is your turn to start writing your own C# code! Following the example and implementation showcased in milestone 2, your task now is to **implement** the **marker interaction between your content and minus AR marker** in order to **make your virtual object shrink** (i.e., decrease its size, scale down).

Examine the code in the **ImageTargetDistanceScaleManager.cs** C# script closely, and **implement** the necessary extensions **in this script** in order to achieve the **logic for scaling down** your virtual object accordingly.

Further Reading

If you want to learn more about C# and scripting (programming) in Unity3D, please start investigating at https://learn.unity.com/project/beginner-gameplay-scripting. If you want to learn more about the Vuforia scripting API for Unity3D, please start investigating at https://library.vuforia.com/content/vuforia-library/en/reference/unity/index.html.

3.2 Implementation Challenge: Hints

Following, some minor hints that should assist you with completing this coding challenge:

- Make sure your **ImageTarget_minus** GameObject has all required **components** attached.

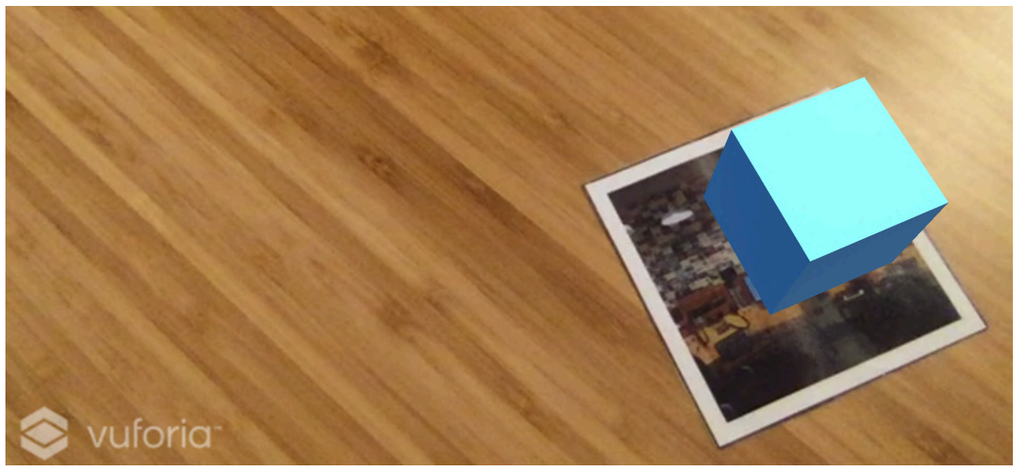
- **All the source code** necessary to implement the required extensions can be written **in the ImageTargetDistanceScaleManager.cs** C# script.

- Make sure to define and use an appropriate **minimum scale** for your virtual object (as you don’t want it to shrink “infinitively”).

- **In general**: Have a close look at the implementation (and its structure / logic) for scaling up the virtual object as illustrated during milestone 2.

3.3 AR testing time

Finally, make sure your implementation is set up and working appropriately. **Run** your application by clicking the **[Play]** button (▸) in the top center of the interface.



1. Place your physical **01_marker_content AR marker** in the field of view of your camera and make sure it gets **successfully detected**.

2. Place your physical **03_marker_minus AR marker** in the field of view of your camera and make sure it gets **successfully detected**.

3. Observe the **virtual object shrink**.

Milestone 3: Complete

Congratulations, you have successfully completed milestone 3 by writing your own C# code and implementing the AR marker interaction logic in order to make your virtual object shrink.

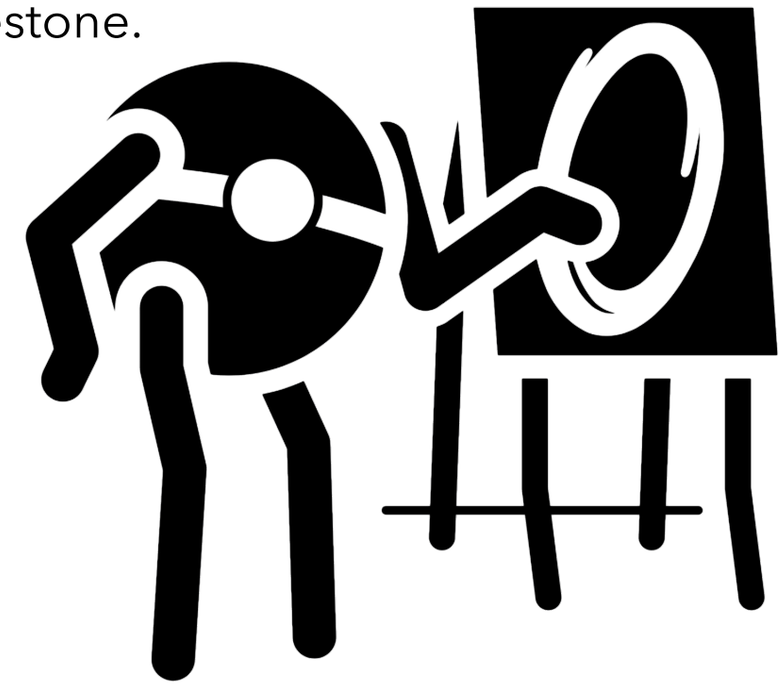
Workshop: Complete

The completion of milestone 3 also concludes this workshop (Part 2 - AR Marker Interaction).

Please read carefully the deliverable and submission instructions provided to you (see Part 2 - AR Marker Interaction: Deliverable and Part 2 - AR Marker Interaction: Deliverable Hints).

Further Reading

If you are interested in further reading and development, make sure to check out the various online resources provided to you at the beginning of each milestone.



Contact

Nico Reski

reski.nicoversity.com

@nicoversity

github.com/nicoversity

nico.reski@lnu.se

(PGP Key ID: B061D75B,
PGP Fingerprint: E826 C9FF 1701 0BAC
CA98 308C 6772 4499 B061 D75B)

Office: HUS D 2269 A

VRxAR Labs

Department of Computer Science
and Media Technology (CM)

Faculty of Technology

Linnaeus University, Växjö

Additional references

Portal icons in the presentation available via
bit.ly/portaliconpack