AVATAR
Advanced Virtual and Augmented Reality Toolkit for Learning

Development of a 3D environment for extended reality

Smart manufacturing laboratory

Eren Enver - 10947639
Saverio Rocchi - 10631378

Agenda

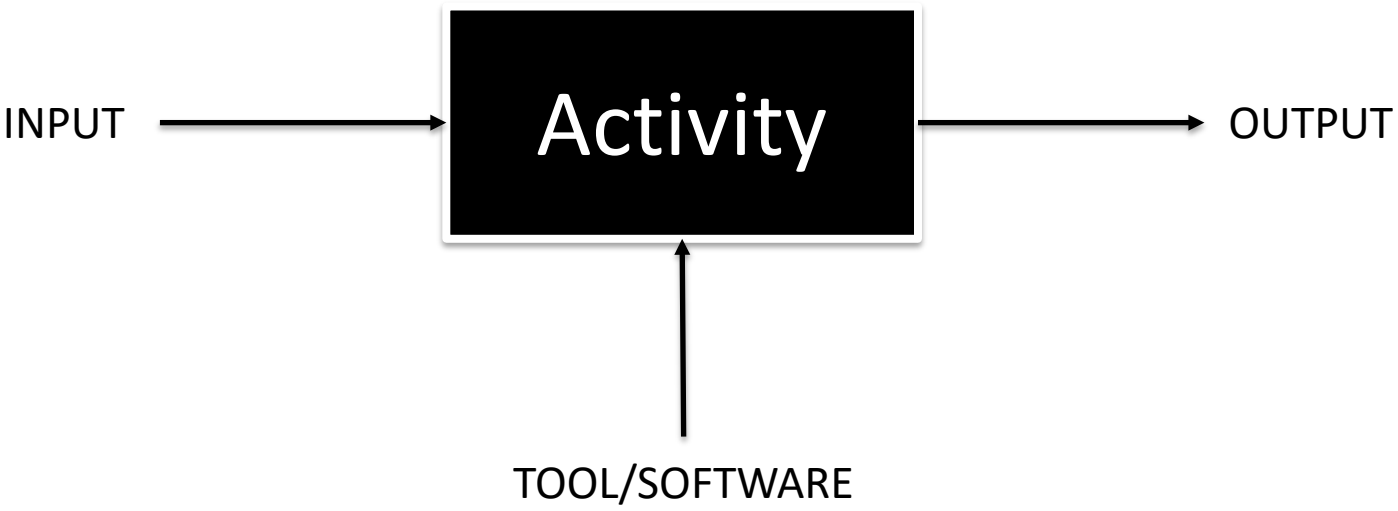
1. Introduction
2. Define the 3D environment
   - The workflow
   - Activities
3. Joint Learning Lab
   - Challenges
   - VR experience

Introduction

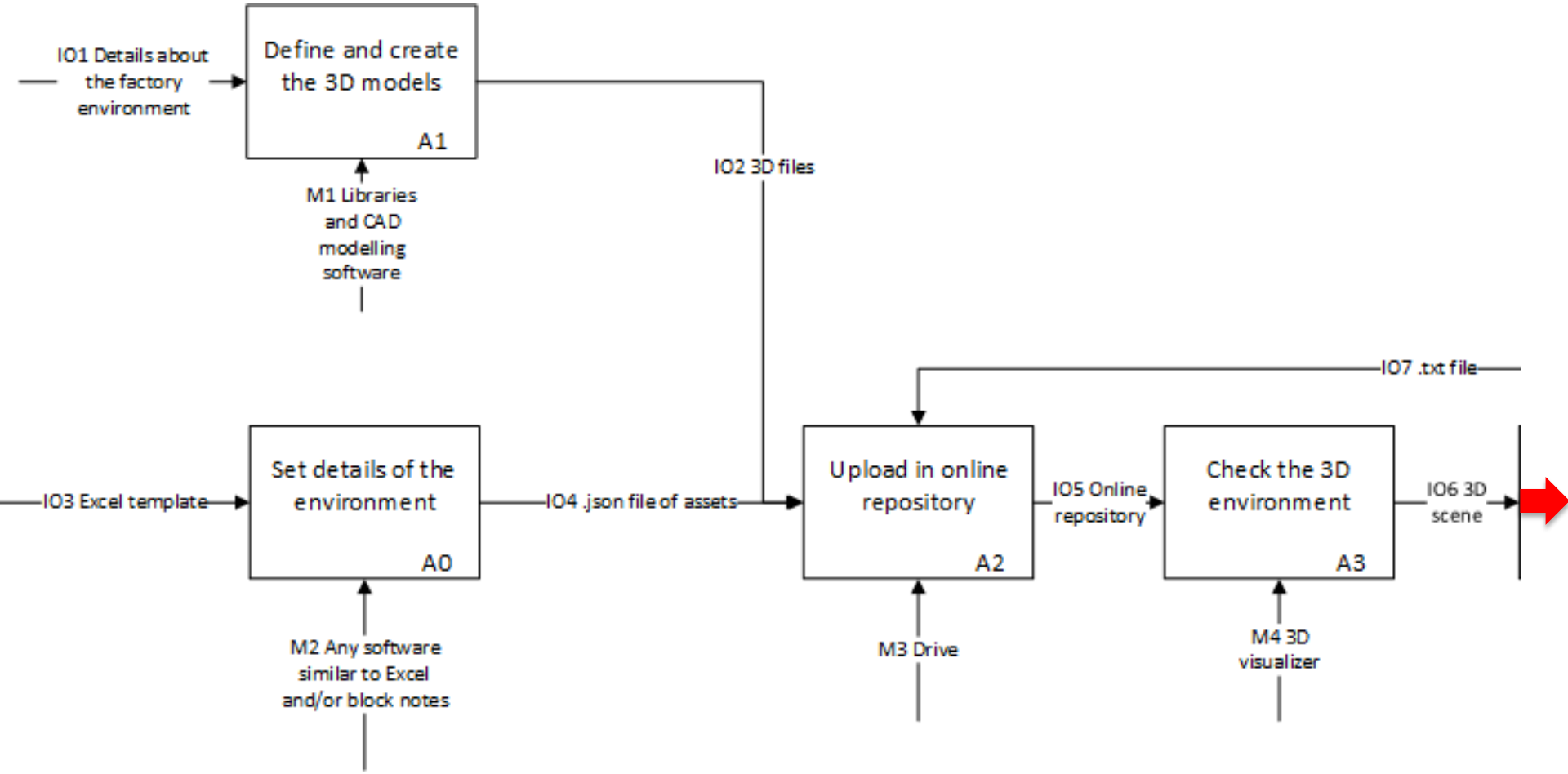
- Erasmus+ program
- VR/AR technologies in engineering
  - Virtual factory
  - Simulation
- Joint Learning Lab

The workflow – part 1

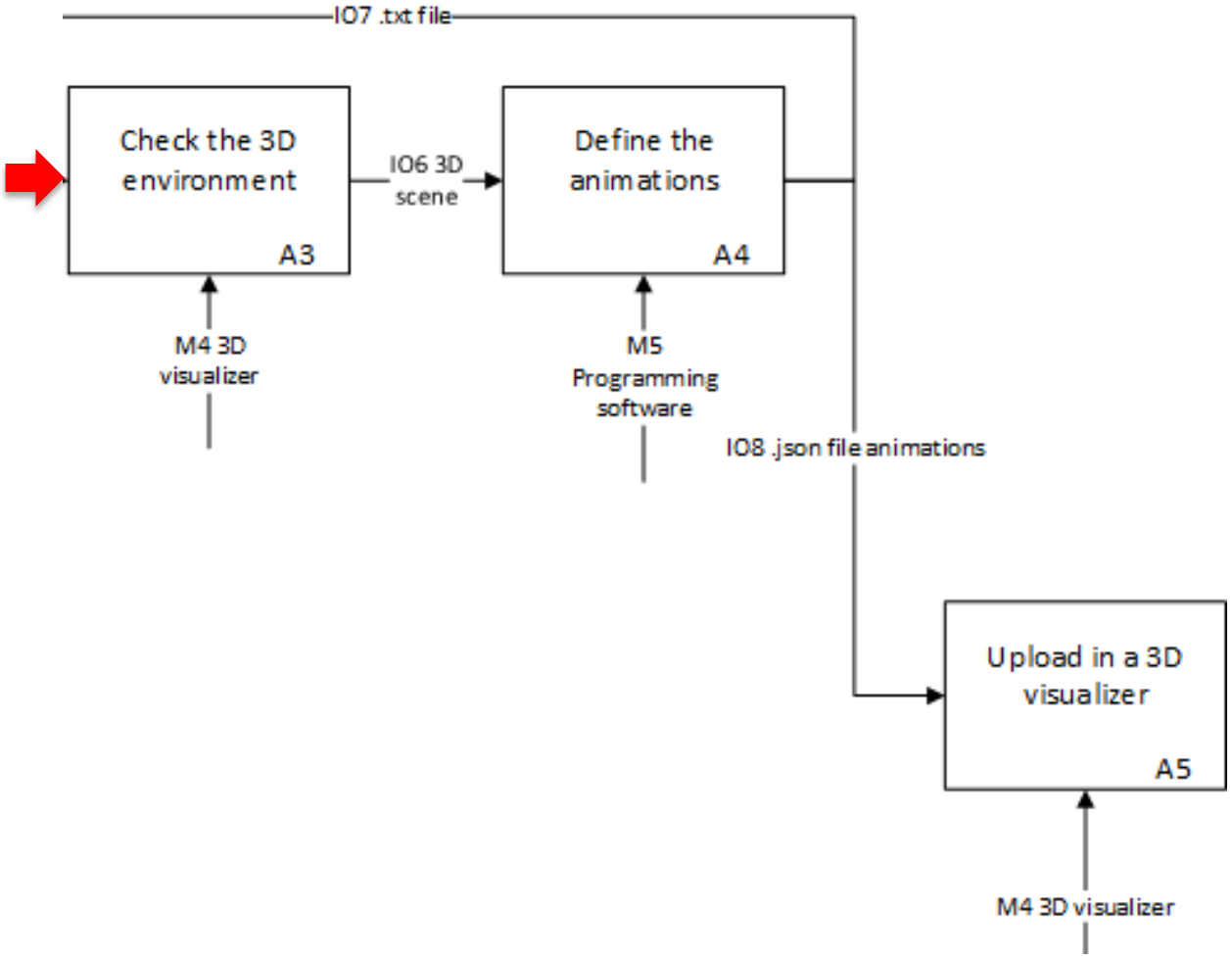
The workflow



The workflow – part 1



The workflow – part 2

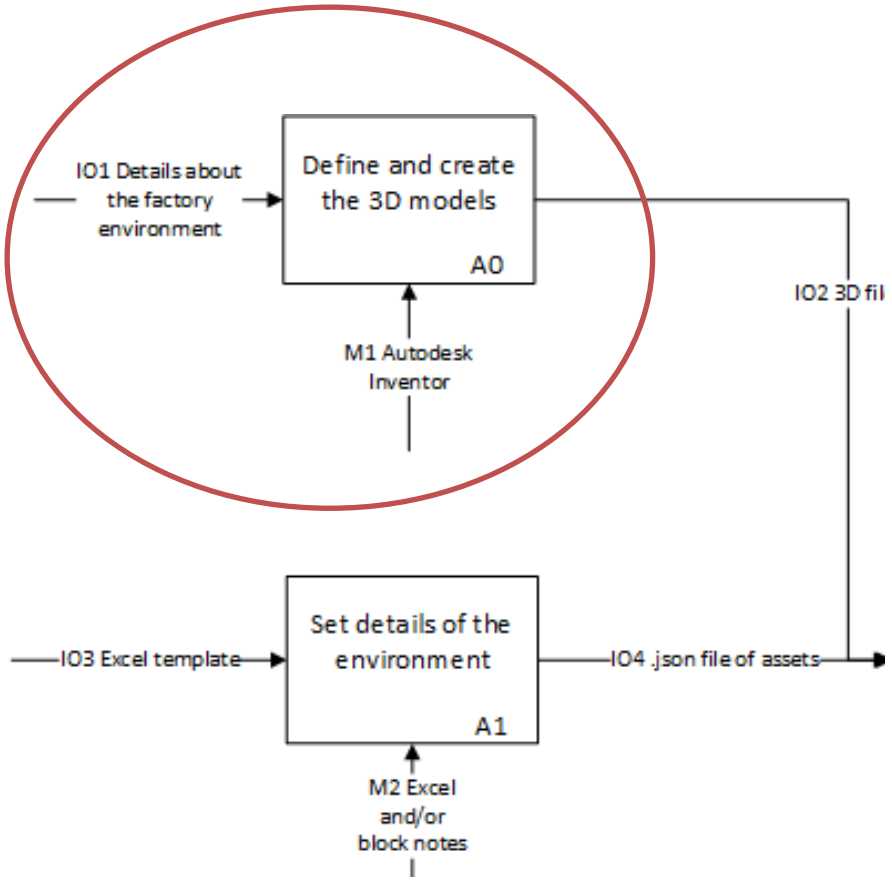


Activities

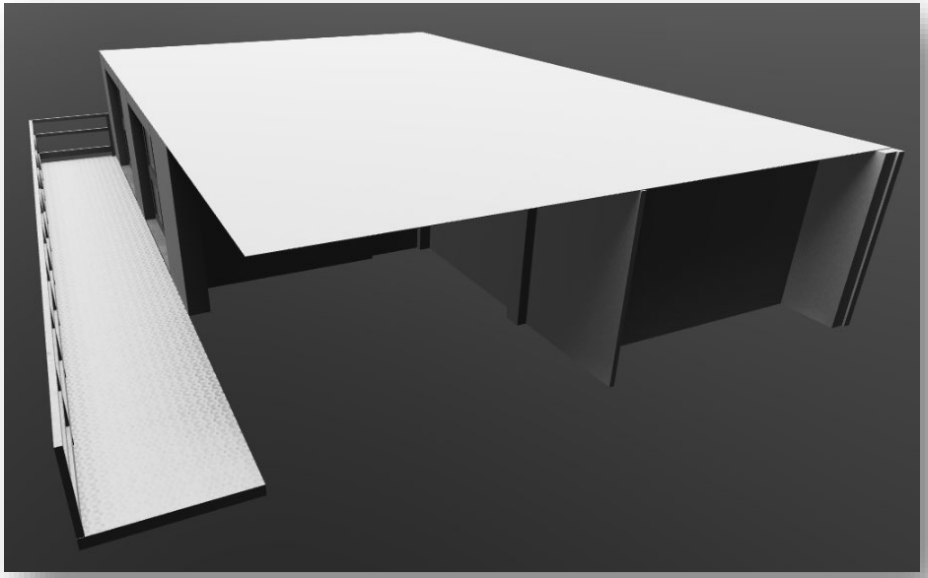
Activities – Define and create the 3D models

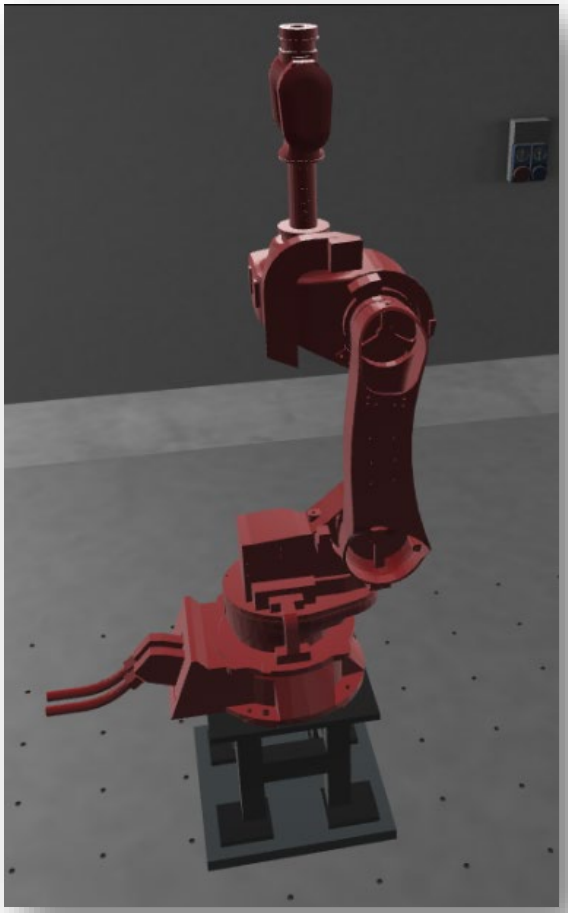
1) List of the models
2) Already existing models:
   - Laboratory structure
   - Robot
3) From scratch:
   - Conveyor
   - Robot Base
   - Tool

AUTODESK
Inventor Professional

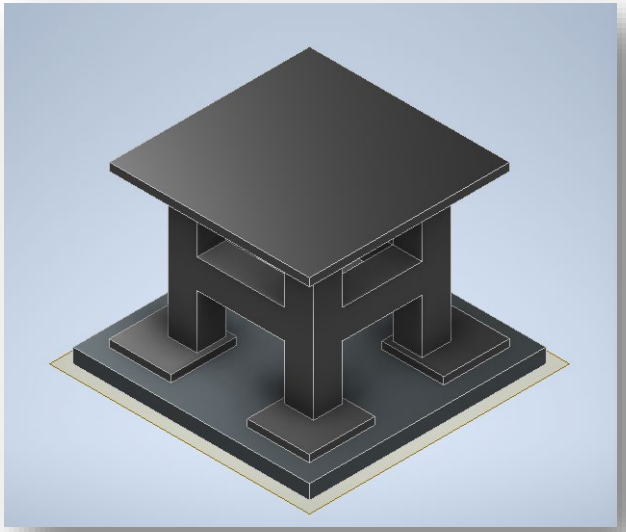


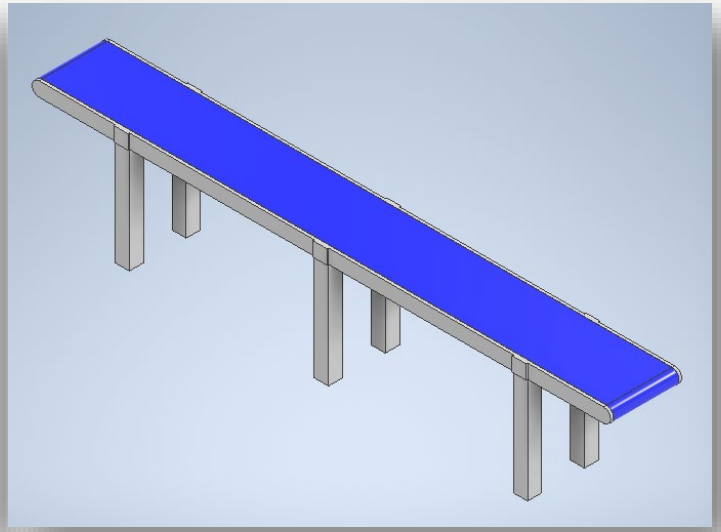
Activities – Define and create the 3D models

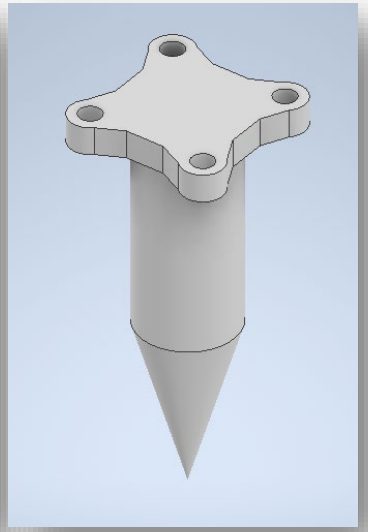




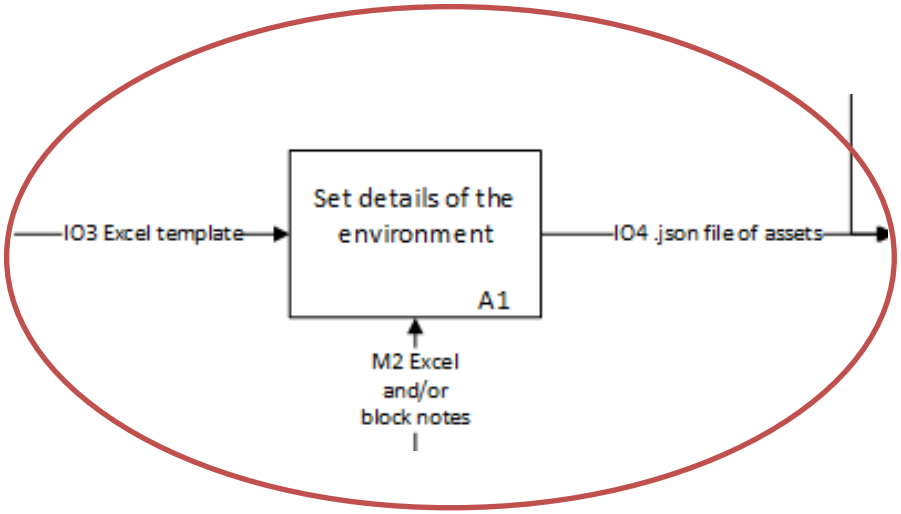
Activities – Define and create the 3D models







Activities – Set the details of the environment

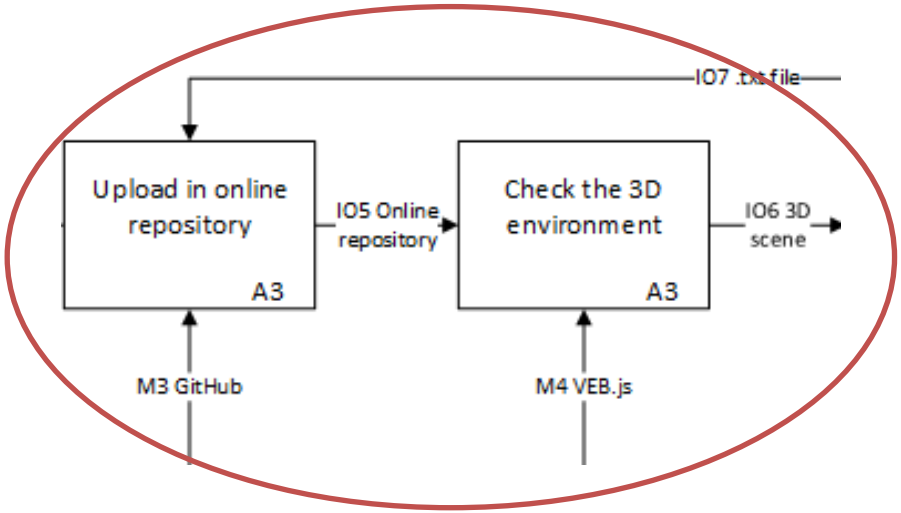


id	inScene	descr	type	model	file	unit	position			rotation			placementRelTo	parentObject

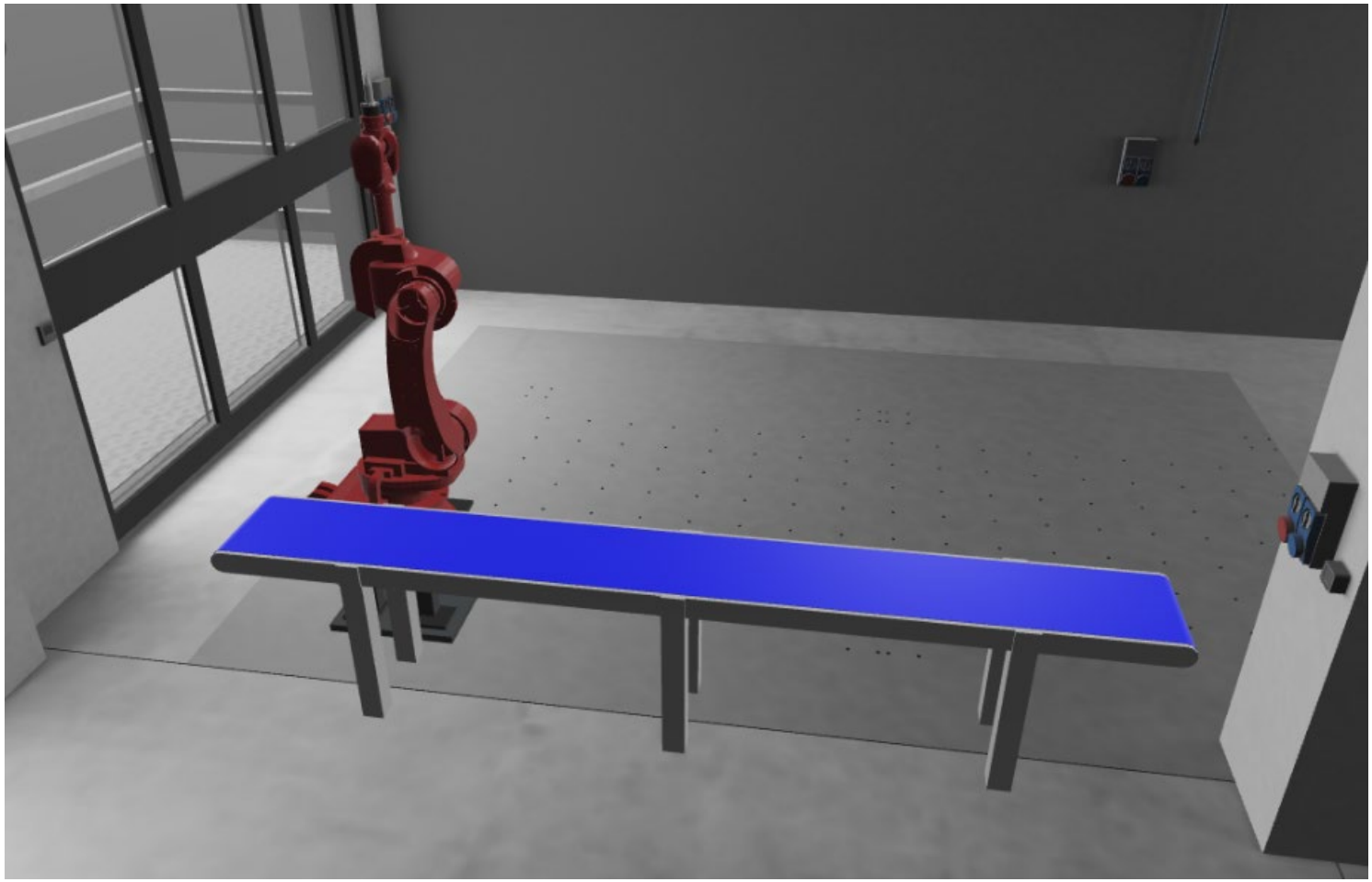
Activities – Upload and check the 3D environment

1) Upload files in GitHub repository
2) Position check
3) Json code updating

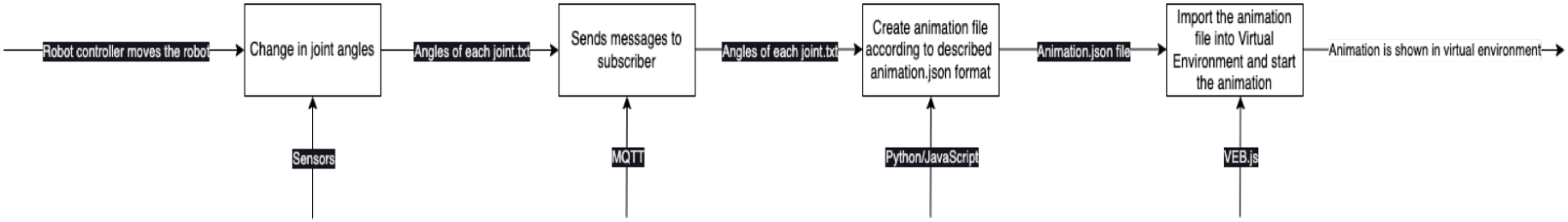
```
{
    "context": {
        "UnitOfMeasureScale": 1,
        "Zup": false,
        "RepoPath": ""
    },
    "scene": [
        "lab_building",
        "comau_ns16hand",
        "comau_ns16hand.base_link",
        "comau_ns16hand.Joint_1",
        "comau_ns16hand.Joint_2",
```



Activities – Check the 3D environment



Activities – Animations
(Workflow for conversion and crucial points)



- While converting raw data to the animation.json file, it is crucial to note that the rotation values are recorded in radians within the animation.JSON file, whereas the raw data might express rotation in another unit (e.g. in degrees, in radians).

- Additionally, it is worth noting that the choice of axis convention can vary depending on the specific environment or context in which it is being used. Different software applications or industries may adopt different conventions for representing the orientation of coordinate axes. Therefore, it is essential to consider and adapt to the axis convention employed within the relevant environment to ensure accurate interpretation and alignment of the data.

Activities – Animations (Conventions for coordinate plane)

- VEB.js convention: Yup
- URDF convention: Zup

1. Y-Up Convention:
In the Y-up convention, the Y-axis points upwards, typically representing the vertical direction in a 3D scene. The X-axis typically represents the horizontal direction, and the Z-axis represents the depth or forward-backward direction. This convention is commonly used in many computer graphics applications, including game engines, modeling software, and physics simulations.
2. Z-Up Convention:
In the Z-up convention, the Z-axis points upwards, representing the vertical direction. The X-axis typically represents the horizontal direction, and the Y-axis represents the depth or forward-backward direction. This convention is often used in geospatial applications, such as geographic information systems (GIS) and mapping software, where the elevation or altitude is important.
It's important to be aware of the convention being used in a particular context to ensure proper orientation and alignment of objects in a 3D space.

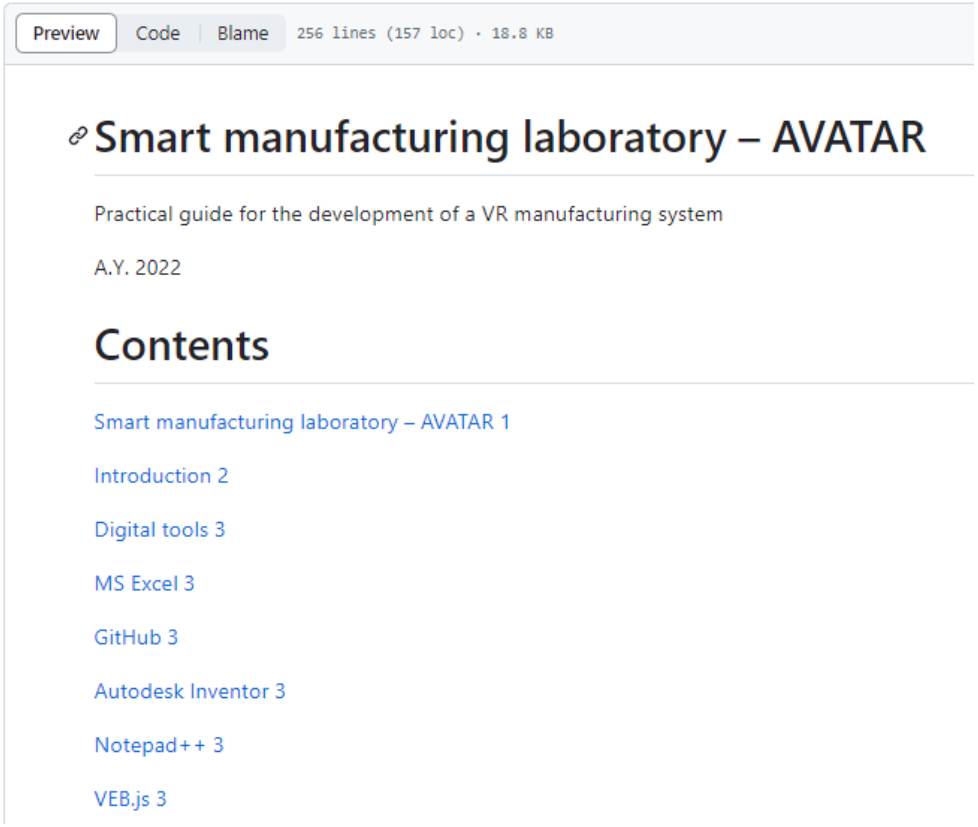
Activities – Animations
(Information about form of anim.json file)

The "context" property includes some general settings for the robotic arm, such as disabling the asset trail, setting the unit of measure scale to 1, disabling Z-up orientation, and specifying a repository animation URL.
The "nodes" property is an array of objects representing different nodes or components of the robotic arm. In this case, there are two nodes defined. Each node has an "id" and an "actions" array. The "id" represents the identifier of the node, and the "actions" array contains the actions associated with that node. In the first node, with the id "comau_ns16hand.Link_1", there is a single action defined. The action has a "trigger" and an "event". The "trigger" specifies that the action should be triggered after a certain time. The "event" type is "show", which implies that something should be shown. The "position" and "rotation" properties define the position and rotation of the object being shown, and the "placementRelTo" property specifies the reference frame relative to which the placement is defined.

```
{
    "context": {
        "assetTrail": false,
        "UnitOfMeasureScale": 1,
        "Zup": false,
        "RepoAnim": ""
    },
    "nodes": [
        {
            "id": "comau_ns16hand.Link_1",
            "actions": [
                {
                    "trigger": {
                        "type": "timestamp",
                        "data": "200"
                    },
                    "event": {
                        "type": "show",
                        "rotation": [
                            0,
                            0.0,
                            0
                        ],
                        "placementRelTo": "comau_ns16hand.Joint_1"
                    }
                },
```

The "sequences" property is an array of sequences that can be triggered by the actions. The sequence has a frames-per-second (FPS) setting and defines positions and rotations for each frame of the animation.
Overall, this code represents a configuration for controlling a robotic arm, defining actions, events, and animations associated with specific nodes of the arm.

Documentation



Preview | Code | Blame 256 lines (157 loc) · 18.8 KB

Smart manufacturing laboratory – AVATAR

Practical guide for the development of a VR manufacturing system

A.Y. 2022

Contents

https://github.com/savixy/AVATARrepository/blob/main/images/documentation.md

Joint Learning Lab – part 2

Challenges

Challenge #1 - Visualize Trajectories

1. Visualize the scene
2. Elaborate trajectories
3. Visualize trajectories
4. Check trajectories effectiveness

Challenge #2 - Receive a Trajectory via MQTT

1. Receive and check the trajectory
2. Create an MQTT client

Challenge #3 - Generate a Trajectory

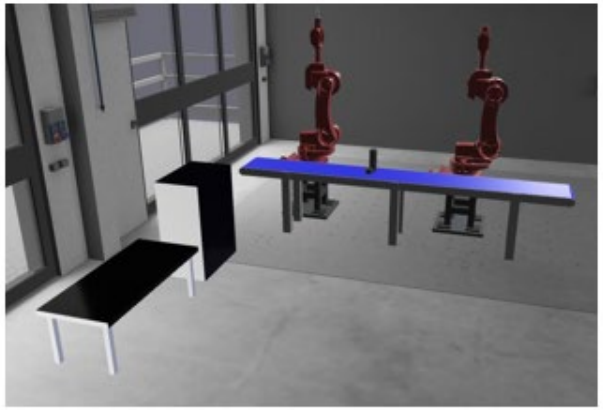
1. Given starting and target position
2. Check the generated trajectory

Challenge #1 – Scene visualization

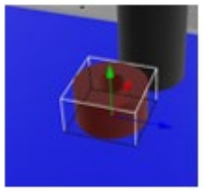
VEB.js

Tool

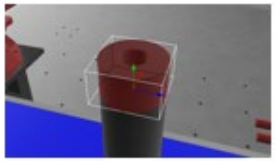
Conveyor



Scene

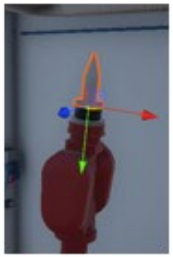


Workpiece 1

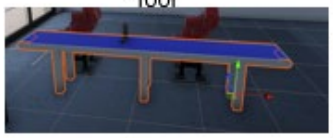


Workpiece 6

Unity

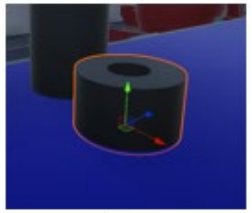


Tool

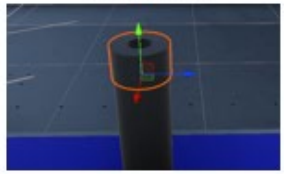


Conveyor

Scene



Workpiece 1



Workpiece 6

Challenge #1 – Elaboration of trajectories


```
[
  {
    "J1": 9.543069579201951e-05,
    "J10": 0,
    "J2": 0.00019278800445174266,
    "J3": -1.5700661261488627,
    "J4": 3.6954891742914644e-05,
    "J5": 1.5697658312829736,
    "J6": -2.8212182286010996e-05,
    "J7": 0,
    "J8": 0,
    "J9": 0
  },
  {
    "J1": 9.543069579201951e-05,
    "J10": 0,
    "J2": 0.00019233650102961179,
    "J3": -1.5700661261488627,
    "J4": 3.5966848214988896e-05,
    "J5": 1.5697664597663785,
    "J6": -2.8220163924283667e-05,
    "J7": 0,
    "J8": 0,
    "J9": 0
  },
  {
    "J1": 9.780658941017338e-05,
```


Python code
(see next slide)

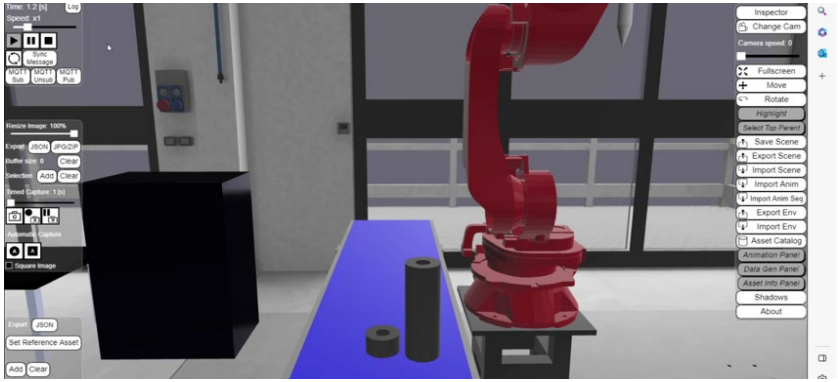

```
"context": {
    "assetTrail": false,
    "UnitOfMeasureScale": 1,
    "Zup": false,
    "RepoAnim": ""
},
"nodes": [
    {
        "id": "Robot_1.Link_1",
        "actions": [
        {
                "trigger": {
                    "type": "timestamp",
                    "data": "0"
                },
                "event": {
                    "type": "show",
                    "rotation": [
                        0,
                        9.543069579201951e-05,
                        0
                    ],
                    "placementRelTo": "Robot_1.Joint_1"
                }
            },
        {
                "trigger": {
                    "type": "timestamp",
                    "data": "100"
                },
                "event": {
                    "type": "show",
                    "rotation": [
                        0,
                        9.543069579201951e-05,
                        0
                    ],
                    "placementRelTo": "Robot_1.Joint_1"
                }
```

Challenge #1 – Elaboration of trajectories

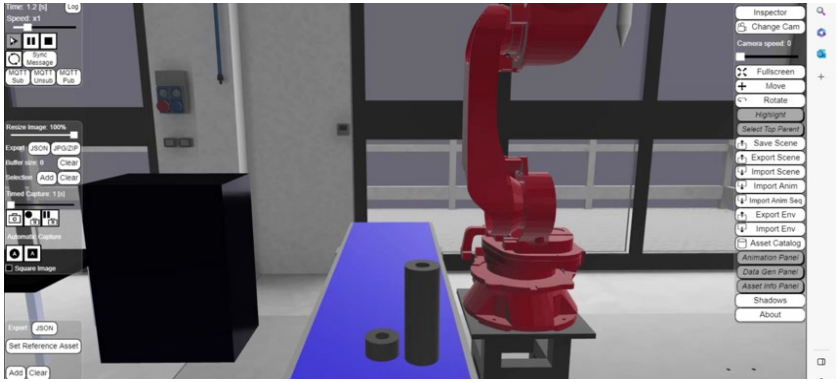
```python
import json

with open("animations/trajectory_1.json", "r") as read_file:
    mylist = json.load(read_file)

def conv(data):
    second_dict = {}
    second_dict["context"] = {"assetTrail": False,"UnitOfMeasureScale": 1,"Zup": False,"RepoAnim": ""}
    second_dict["nodes"] = []
    second_dict["sequences"] = []
    second_dict["bookmarks"] = []
    actual_time = 0
    for name, value in data[0].items():
        if int(name[1:]) <= 6:
            node = {"id": "Robot_1.Link_"  + name[1:],
                "actions": [{"trigger": {"type": "timestamp","data": str(actual_time)},
                    "event": {"type": "show","rotation": [0,value,0],"placementRelTo":  "Robot_1.Joint_"  + name[1:]}}]}
            second_dict["nodes"].append(node)
    for link in data[1:]:
        actual_time += 100
        for name, value in link.items():
            if int(name[1:]) <= 6:
                action = {"trigger": {"type": "timestamp","data": str(actual_time)},
                    "event": {"type": "show","rotation": [0,value,0],"placementRelTo": "Robot_1.Joint_"  + name[1:]}}
                second_dict["nodes"][int(name[1:])-1]["actions"].append(action)

    with open("animations/anim_traj1.json", "w") as outfile:
            json.dump(second_dict, outfile)
    outfile.close()

conv(mylist)
read_file.close()
```

Challenge #1 – Visualization of trajectories

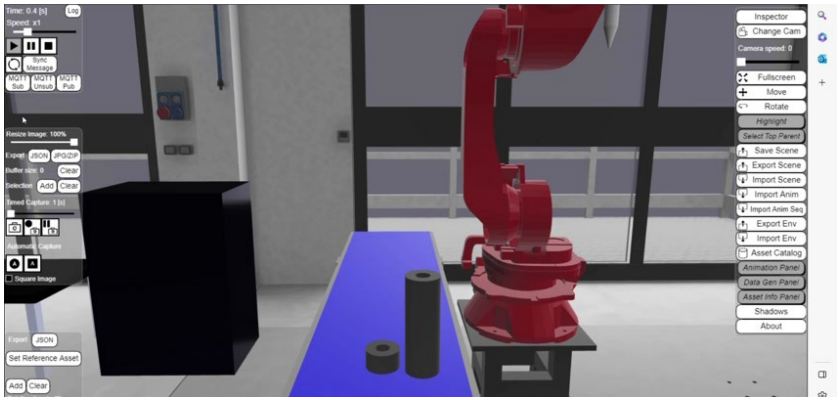
Trajectory 1



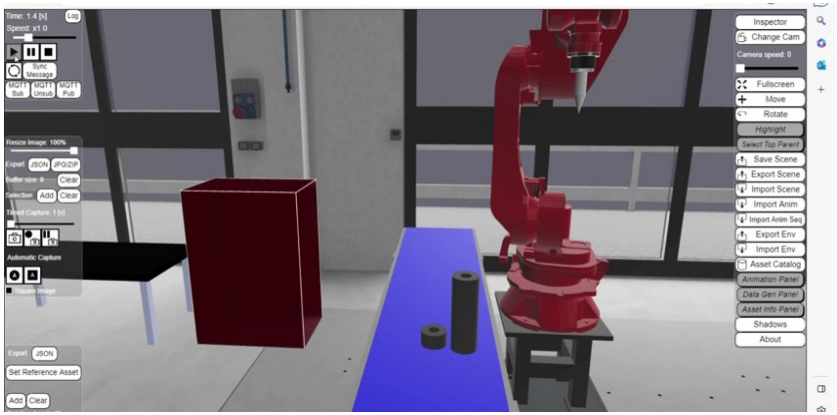
Trajectory 2



Trajectory 3



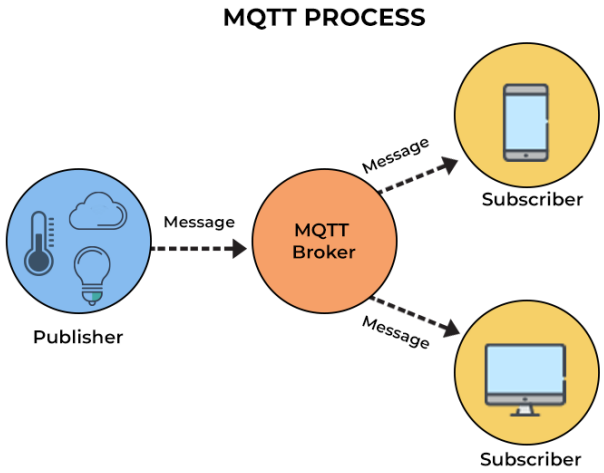
Trajectory 4



Challenge #1 – Elaboration of trajectories

	Tj.1	Tj.2	Tj.3	Tj.4
Have collision	yes	yes	no	no
Reach the goal	yes	no	yes	yes
Distance from target [m]	0,0368	0,1712	0,0368	0,0368

Challenge #2 – Receive a trajectory via MQTT and check it

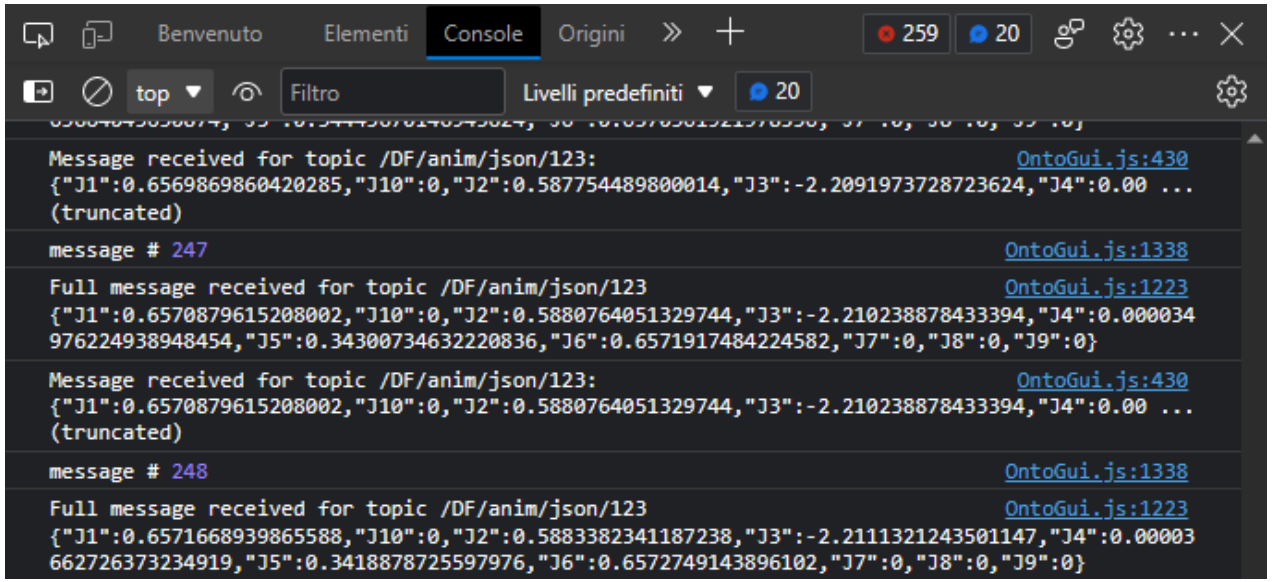


OntoGuiWeb - MQTT Synchronization

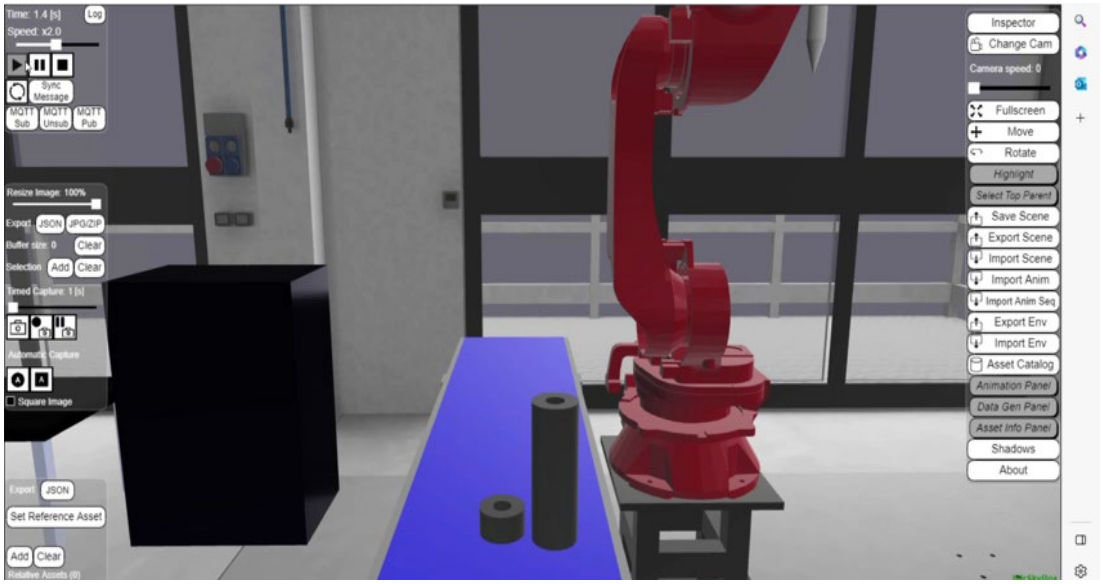
Select root graph	https://w3id.org/ontoeng/CW/RoboticCell		Update Graphs
BROKER			
Host	ws://broker.emqx.io:8083/mqtt	Connect	Disconnect
SUBSCRIBE			
Topic	/DF/anim/json/123	**QoS** 0	Subscribe
Subscribed Topics	/DF/anim/json/123		Unsubscribe
PUBLISH			
Topic	/DF/anim/json/123	**QoS** 0	
Message	fhdsjfsdfsdfds		Publish
Message Sequence	**Challenge#2 Sequence** / Start Publish	Stop Publish	**Message interval:** 0.1 [s]
	Select Sequence / Load and Start Publish		
Developed by Walter Terkaj (walter.terkaj@stiima.cnr.it) Creative Commons Attribution-NonCommercial 4.0 International Public License			License CC BY-NC 4.0

Challenge #2 – Receive a trajectory via MQTT and check it

```
 10:35:26: Message received for topic /DF/anim/json/123:
{"J1":0.6556662531785147,"J10":0,"J2":0.5834180389332906,"J3":-2.193686758680806,"J4":0.00 ... (truncated)
 10:35:26: Message received for topic /DF/anim/json/123:
{"J1":0.6558994867353635,"J10":0,"J2":0.5841238371720711,"J3":-2.196200976708661,"J4":0.00 ... (truncated)
 10:35:26: Message received for topic /DF/anim/json/123:
{"J1":0.6561057934978732,"J10":0,"J2":0.5847881355730468,"J3":-2.1985983572854773,"J4":0.0 ... (truncated)
 10:35:26: Message received for topic /DF/anim/json/123:
{"J1":0.656286361412853,"J10":0,"J2":0.5854134926030999,"J3":-2.200810597709447,"J4":0.000 ... (truncated)
```



Challenge #2 – Receive a trajectory via MQTT and check it

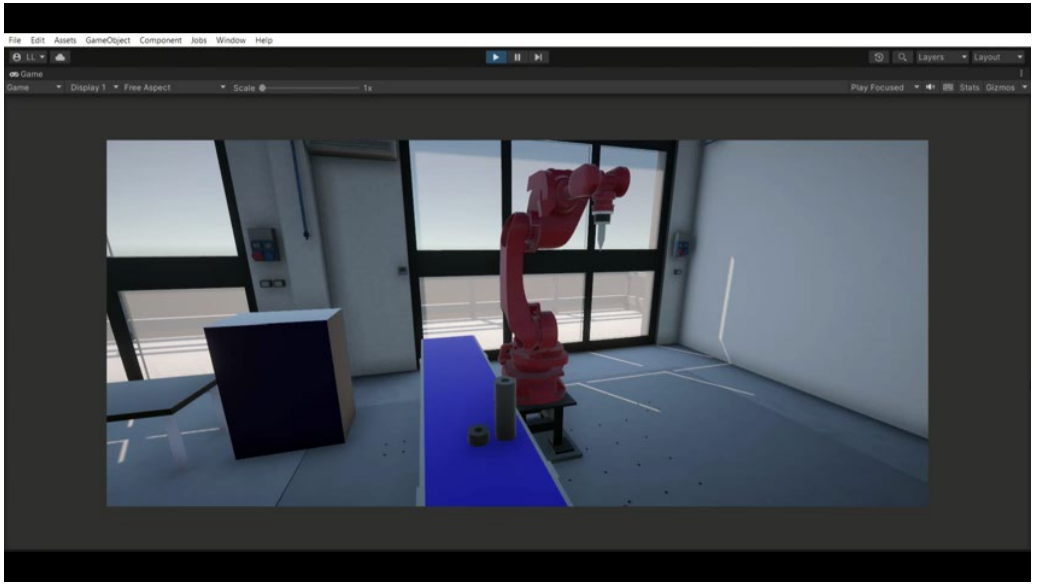


	Trajectory 5
Collision	No
Goal	No
Distance [m]	0,0799

Challenge #3 – Create a trajectory

- Take the starting position of the joints from the challenge description
- Manually in Unity using the robot model create the final positions for each joint so the robot is touching the workpiece
- Add a middle position to guide the tool around the obstacle to avoid collision
- Fill the animator and play
- Export the file with the .anim extension

	New trajectory
Goal	Yes
Collision	No
Distance	0,0455



VR for learning

Learning flow

Define the tasks

Configure the visor

Configure the scene

Thanks for your attention

POLITECNICO
MILANO 1863