POLITECNICO
MILANO 1863

AVATAR

Development of a 3D environment for extended reality

Smart manufacturing laboratory

Professors of Politecnico:
Marcello Urgo
Adalberto Polenghi
CNR researcher:
Walter Terkaj

Report made by:
SAVERIO ROCCHI
Id code: 10631378

EREN ENVER
Id code: 10947639

A.Y. 2022/2023

Contents

Introduction

The AVATAR Erasmus+ project has a primary **goal** of supporting and facilitating the use of VR/AR technologies into engineering environment. These advanced digital technologies can be easily used for engineering education:

1. Entering in the three-dimensional experience allows to have a **comprehensive perspective** on engineering problems.
2. It is possible to reach a deep level of **details** so that user understand the problem in all its features.
3. The use of VR/AR from **distance** allows access to remote education and autonomous learning, this helps when there is an inability to be physical present.

In this report it is described the work of our team during the Smart Manufacturing Laboratory course in Politecnico di Milano. The laboratory typically concerns the collaborations with companies or universities. In our case we participated to the AVATAR Erasmus+ project that involved University of Belgrade and Grenoble INP and their students.

The collaboration helps creating a network all over the Europe and allow sharing knowledge and skills between students. After some introductory lecture about VR/AR held by different professors, student had the possibility to meet face to face.

The meeting with other students, this year, has been held in Politecnico di Milano and CNR-STIIMA institute. During those days they faced challenges about VR/AR to understand the potentialities of this technology.

Deadlines and goals

To have a clear view of the work done, this has been subdivided in three phases, summarized in the following table:

Phase	Task	Objective
1	Provide help in **creation** of a **guide** for phase 2	Study in advance the tools and the environment to **provide support** during the JLL (Joint Learning Lab)
2	**Complete** the **challenges** during the Joint Learning Lab in collaboration with international students	**Support** international **students** and **share knowledge** with them
3	Obtain **feedbacks** and **improve** the **documentation** provided during the phase 2	**Update the guide** for future students that want to challenge them with VR/AR

Here is a Gantt chart framing the tasks performed over the semester:

Phase	Task	Start	Finish	20-Feb	08-May	15-May	19-June
1	Creation of the guide	20/2	8/5				
2	Joint Learning Lab	8/5	15/5				
3	Updating the guide	15/5	19/6				

At the following links the output of each phase can be consulted:

1. **Guide:** https://github.com/savixy/AVATARrepository/blob/main/images/documentation.md
2. **Output of challenges:** https://github.com/savixy/AVATARrepository/tree/main/JLL

1. Creation of guide for laboratory experience

Since Politecnico was in charge of hosting the laboratory a first task we faced was the creation of part of the documentation that has been used during the laboratory to guide students in completing challenges.

The aim of the documentation is to:

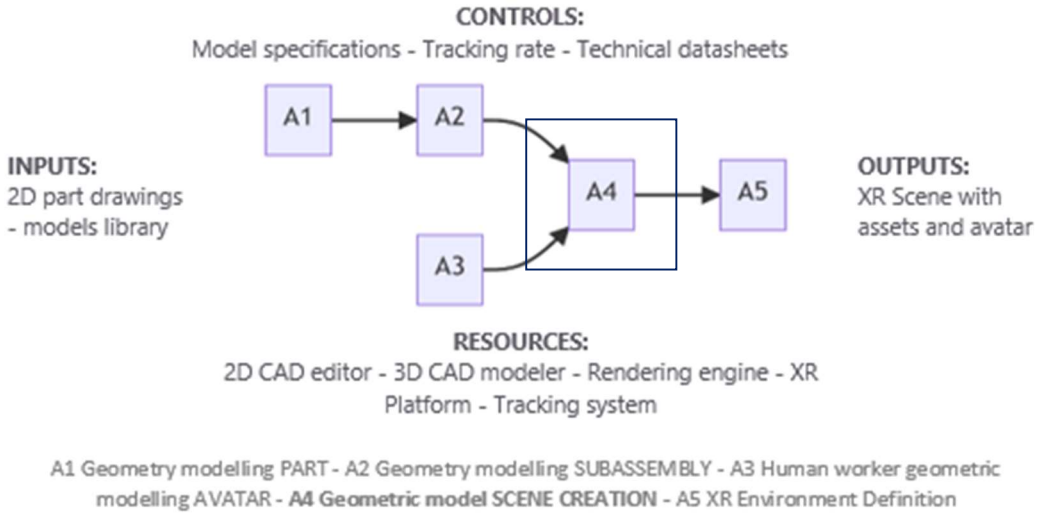
1. Generate a **workflow** that helps in the creation of a **3D environment.**
2. Describe **how to use** some specific **tools** supporting the point 1.
3. **Move a COMAU robot** following a specific path.

In our specific case the 3D environment is the CNR laboratory in Milano. Moreover, the suggested tools are only some advice, any kind of software of your knowledge can suite the workflow if is able to provide the same output as the suggested software.

The main problem faced during this phase was understating how to use some of the software and tools that we have never used before. In order to study how to use them some more details were provided by researcher Terkaj (e.g. https://virtualfactory.gitbook.io/vlft/kb/fdm).

1.1 Workflow

To get a clear view of where we stand with our work within the creation of an XR environment we can watch this picture:

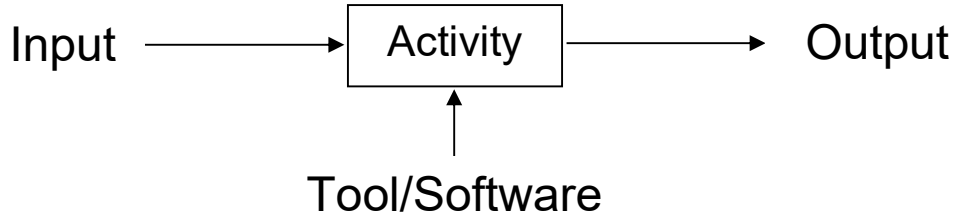


A1 Geometry modelling PART - A2 Geometry modelling SUBASSEMBLY - A3 Human worker geometric modelling AVATAR - **A4 Geometric model SCENE CREATION** - A5 XR Environment Definition

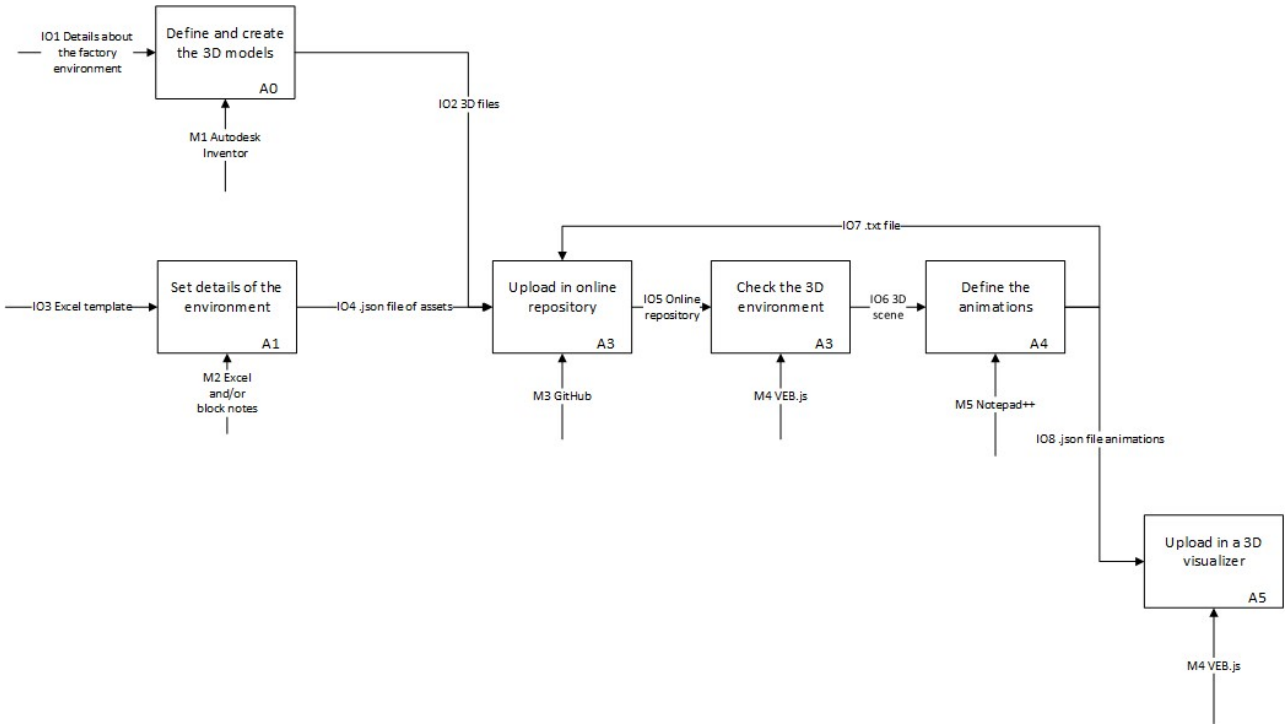
Each of the squares correspond to a phase to be completed. During this first phase of the report, we focused on the scene creation (A4) and we also worked on modelling some parts (A1).

The scene creation can also be subdivided in other subphases. To do that we used a IDEF0 diagram as support. It is a tool developed to describe any kind of workflows composed by different elements here described:

- The **square** is representing an **activity** that must be completed. Each square is associated with input/output, digital tools, and learning objectives.
- The **I/O** represent the necessary materials needed to complete the activity and the expected outcome after completing it.
- **Tool/software** are those necessary to complete the activity.



In the following image are shown all the activities to be completed in order to create the 3D environment. We will later go through each steps detailing the work done.



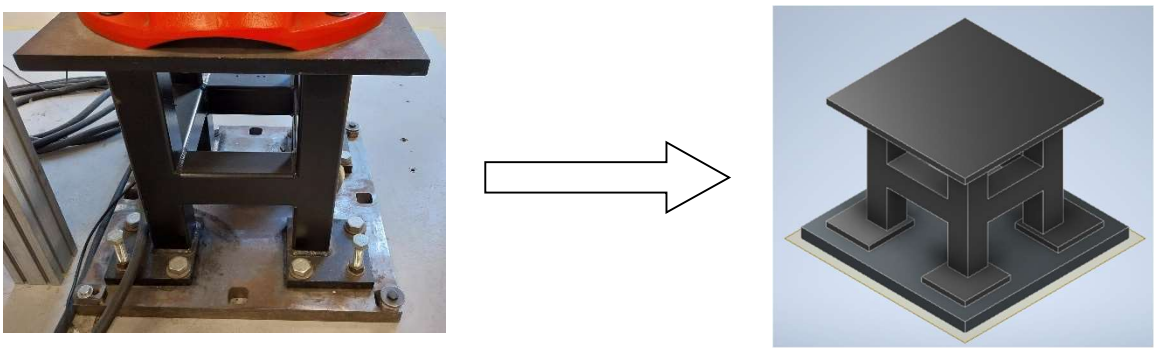
1.1.1. Define and create the 3D models

The first step to create a 3D environment is to know what objects are inside the lab environment. Depending on the level of details we want to obtain more or less objects can be shaped.

Being in contact with CNR laboratories we were able to obtain all the information about the main **objects** inside the laboratory and to visit it to clearly **measure** and register the **distance** between all the objects.

Once aware about the laboratory layout it was time to search for the 3D models. We had 2 ways to obtain them:

1. Exploit **existing models** provided by CNR or in online libraries
2. Create them **from scratch** using any CAD software. Here is shown an example of the *robot base* that we shaped using Autocad Inventor.



As said in chapter one, the third objective is to move the COMAU robot, for this purpose the level of details in the environment can be low. The laboratory structure and the robot model has been provided from researcher Terkaj. The models we had to shape from zero the following:

- 2 robot bases
- 1 conveyor
- 1 tool
- the robot controller

1.1.2. Create an online repository

Since our team was composted by two people we decided to work online so that all the files were share between all the components of the group. Moreover in the following steps this online repository will be necessary as a reference to reach the 3D models.

The online repository we used is GitHub. The way to create it was simple, after the creation of the account we were allowed to upload all the files create until now (3D files of the models).

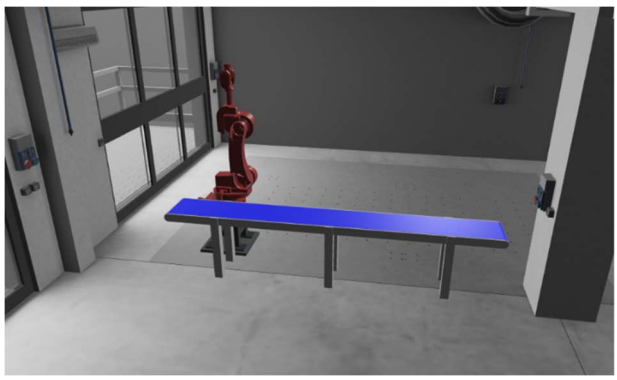
1.1.3. Set the details of the environment

Before explaining how we joined the models in the scene we have to make a premise. The suggested software we used an online software is called VEB,js. *"**VEB.js** (**V**irtual **E**nvironment based on **B**abylon.**js**) is a reconfigurable model-driven virtual environment application based on Babylon.js"* (https://virtualfactory.gitbook.io/vlft/tools/vebjs). Since this software was completely new for us and completely different from anyone we know, we faced not little problem during the creation of the 3D environment. The decision of using this software is related to the fact that is a free to use software and reconfigurable for any kind of applications.

The creation of a .**json file** format is necessary to upload the models in the environment. To create it an Excel file has been provided to us, this latter was automatically able to create the file containing all the information about models. More details on how to compile the Excel file can be found here: https://virtualfactory.gitbook.io/vlft/kb/instantiation/assets/spreadsheet

id	inS	descr	type	model	file	unit	position			rotation			placementRelTo	parentObject
lab_building	1	in	ldingsmart.org/IFC/D		Lab.glb	1	0	0	0	0	0	0		
conveyor	1		https://w3id.org/ont		Conveyor.glb	1	2	0	**0**	0	0	0	lab_building	
comau_ns16hand	1		https://w3id.org/ont		comau_ns16hand.glb	1	4	0	**-2,5**	0	-1,57	0	lab_building	
comau_ns16hand.base_link	1		https://standards.bui		comau_ns16hand.glb#base_link	1	0	0	0	0	0	0	comau_ns16han	comau_ns16hand
comau_ns16hand.Joint_1	1		https://standards.bui		comau_ns16hand.glb#Joint_1	1	0	0,251	0	0	3,142	-3,142	comau_ns16han	comau_ns16hand.base_link

Despite the Excel file as support, we also needed to enter in the details of the .json file code to make adjustments. Sometimes the inserted models were not shown in the model or were not in the right position also due to a bug the was found thanks to the help of researcher Terkaj.
Once the json file was completed, it has been uploaded to the repository. We then open VEB.js and linked the position of the json file. In this way we were able to see the lab environment.
Here is a link to access the result on VEB.js: https://shorturl.at/cEIQU



1.1.4. Move the robot

Once the environment was completed and uploaded it was time to **move the robot**. As already said before, using and open-source software it was not so simple to complete this task. In fact, we couldn't set a starting and ending point and ask the program to automatically create the trajectory. We'll now explain all the steps we completed to show the movement of the robot.

The robot trajectory was provided as **joint position** in a **txt file**. Since the robot had 6 joints for each frame 6 joints position were provided to us for each of the frame of the movement.

The following step was to translate the txt file in a readable format for VEB.js, that means in a json file. After studying the structure of some provided animation code and the theory we managed to write json file (https://virtualfactory.gitbook.io/vlft/kb/instantiation/animations). The coding process couldn't be done manually since the number of **frames** were in the order of **thousands**. This high number of intermediate steps of the trajectory was necessary to create a smooth as possible movement to the human eye. So, we decided to use a **Python script** to automatically transform the single **joint positions in** the **text-based format** into an **animation file(.json)** that can be visualized in VEB.js.

Here is the input file, the Python script, and the output file:

Input file:


```
[
  {
    "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": 0.5791133768725185,
    "J10": 0,
    "J2": 0.6797957873695596,
    "J3": -2.0222578937757323,
    "J4": -3.073769878651504e-05,
    "J5": 0.4394223967319405,
    "J6": 0.5796292961377703,
    "J7": 0,
    "J8": 0,
    "J9": 0
  }
]
```

Python script to convert the trajectory in text-based format into animation file(.json):

```
import json
 # Import the trajectory file (in JSON format) to be converted into an animation file (also in JSON format)
with open("trajectory_1.json", "r") as read_file:
   # Assign the imported trajectory to the mylist variable
   mylist = json.load(read_file)
 def conv(data):
   # Create a dictionary (second_dict) to be used while creating the animation file
   second_dict = {}
   # Specify the variables in the context section of the animation file
   second_dict["context"] = {
      "assetTrail": False,
      "UnitOfMeasureScale": 1,
      "Zup": False,
      "RepoAnim": ""
   }

   # Create sections for nodes, sequences, and bookmarks as part of the animation file
   second_dict["nodes"] = []
   second_dict["sequences"] = []
   second_dict["bookmarks"] = []
  # Initialize the actual time to 0
   actual_time = 0
   for name, value in data[0].items():
      # Create a node for each joint and assign an ID according to the robot's hierarchy
      # There are 10 joints in the trajectory file, but only 6 of them are available in the robot and the rest are not
     used
      if int(name[1:]) <= 6:
         # Assign the initial rotation of each joint
         # Each joint can rotate only around the Z-axis according to the URDF file, which has a Z-up coordinate
         system.
         # However, we need to rotate around the Y-axis according to the animation file, which has a Y-up
         coordinate system. So assign the value to the Y-axis.
         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:]
                }
            }
        ]
    }

    # Append each node to the nodes section of the animation file initially
    second_dict["nodes"].append(node)
# For each time step in the trajectory file, create an action for each joint and assign the rotation values to each joint
for link in data[1:]:
    # Increment the actual time according to the time step of the trajectory file
    actual_time += 100
    # Iterate through the joints and create an action for each joint for each time step
    for name, value in link.items():
        if int(name[1:]) <= 6:
            # Create an action for each joint, assign the rotation and time values to each joint
            action = {
                "trigger": {
                    "type": "timestamp",
                    "data": str(actual_time)
                },
                "event": {
                    "type": "show",
                    "rotation": [0, value, 0],
                    "placementRelTo": "Robot_1.Joint_" + name[1:]
                }
            }
            # Append each action to the actions section of the related node (joint)
            second_dict["nodes"][int(name[1:])-1]["actions"].append(action)
    # We have action sequences for each joint in the nodes section of the animation file
# Create an animation file in JSON format and write the second_dict, which has the animation file format, to the file
with open("anim_traj1.json", "w") as outfile:
    json.dump(second_dict, outfile)
outfile.close()
conv(mylist)
read_file.close()
```

Output file:


```
{
  "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"
          }
        },
```


...

Here are the references to the complete files:

- Json input file: https://github.com/enverern/AVATAR_report/blob/main/trajectory_1.json
- Python script: https://github.com/enverern/AVATAR_report/blob/main/converter_json_to_anim_for%20allchallenges.py
- Json output file: https://github.com/enverern/AVATAR_report/blob/main/anim_traj1.json

2. Support the solving of challenges during JLL

Introduction

After conducting thorough research and familiarizing ourselves with the lab's configuration and conventions, we actively participated in the **Joint Learning Lab** (JLL) and engaged with various groups to tackle the challenges at hand. During the week we participated in different kind of activities. The most important, for which we developed the guide, were those done on Tuesday and Wednesday at the STIIMA-CNR building in Milan. The complete schedule is shown in this image:

	Monday May 8th 2023	Tuesday May 9th 2023	Wednesday May 10th 2023	Thursday May 11th 2023	Friday May 12th 2023
	POLIMI	STIIMA-CNR	STIIMA-CNR	POLIMI	POLIMI
09:15 – 10:15	**Welcome Introduction Lectures** Room Sala Ovale	**Visit to the laboratories** (VR lab, Robotics lab)	**Groupwork** with tutors (Meeting rooms)	**XR Laboratory Activities** Room L.04	**Groupwork** Room MEL LAB 1
10:15 – 11:15					
11:15 – 12:15	Visit to the VR Cave	**Groupwork** with tutors (Meeting rooms)			
12:15 – 13:15					
13:15 – 14:15	LUNCH	LUNCH	LUNCH	LUNCH	LUNCH
14:15 – 15:15	**Lectures** Room L.09	**Groupwork** with tutors (Meeting rooms)	**Groupwork** with tutors (Meeting rooms)	**XR Laboratory Activities** Room L.04	**Presentations of the Groupworks** Room MEL LAB 1
15:15 – 16:15			**Visit to the Robotics lab**		
16:15 – 17:15			**Group work** with tutors (Meeting rooms)		**Conclusions Farewell** Room MEL LAB 1
17:15 – 18:15					

Although we had prior knowledge about the lab, the specific details of the challenges were unknown to us. As we worked alongside our assigned groups to solve these challenges, we also shared our expertise and aided other groups based on our accumulated knowledge.

The final objective of those two days was to support out colleagues but also deepen and share the knowledge about virtual reality and robotics .

The team

Before entering in the details of the single challenges let's present the **colleagues** of our two teams and how we decided to **assign the work**:

Team D:


Kaveh Bekhrad


Nađa Belić


Aleksandar Ćosić

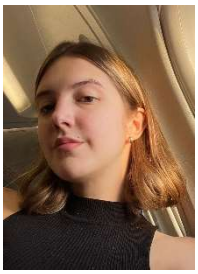

Leonardo Lomacci


Saverio Rocchi


Matteo Speranza

Team C:


Matija Žuža


Kristina Golo


Enver Eren


Kan Kouakou


Marco Varisco

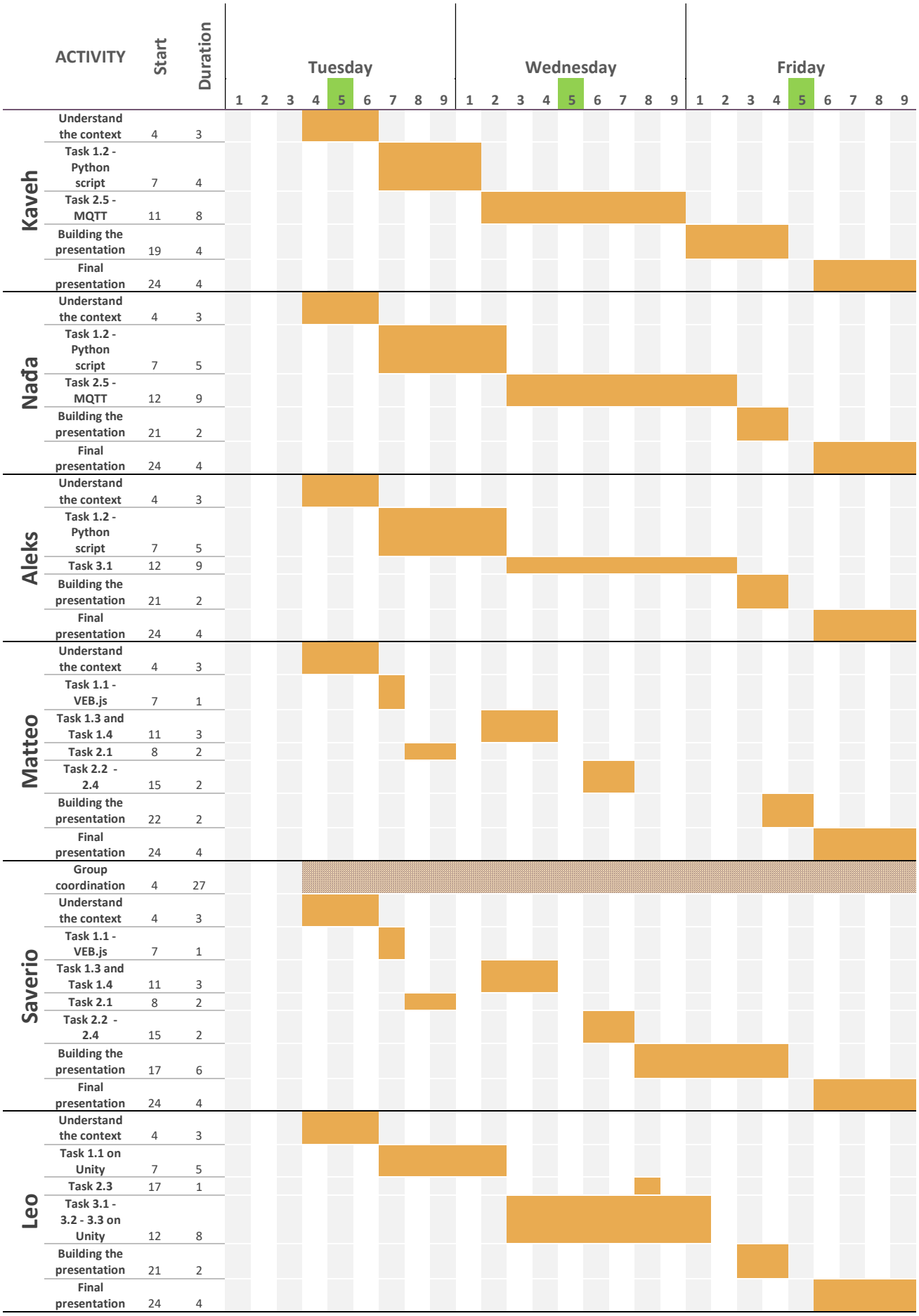

Alessandra Lupo

In team D, we decided to divide the work considering the single skills of each individual. As example as a management engineer, Saverio focused more on the creation of the PowerPoint **presentation** and coordinated the whole group, checking the finished challenges and kept track of the progresses. Kaveh, Nadia and Aleksander were more expert in **Python programming**, finally Leonardo and Matteo on **unity software**. The fact that we had different backgrounds meant that we had added value in terms of quality and goal achievement. Finally, each **problem** faced by the team members was always **shared** with all the team so that everyone was in line and could **contribute their value**.

A similar thing has been done in team C but in this case, each group component focused on a specific challenge:

- Enver and Kan challenge 1
- Matija and Kristina challenge 2
- Marco and Alessandra challenge 3

In the next page you can see a Gantt Chart about the division of the workload.


ACTIVITY
Start
Duration
Tuesday
Wednesday
Friday
1 2 3 4 5 6 7 8 9
Kaveh
Understand the context 4 3
Task 1.2 - Python script 7 4
Task 2.5 - MQTT 11 8
Building the presentation 19 4
Final presentation 24 4
Nađa
Understand the context 4 3
Task 1.2 - Python script 7 5
Task 2.5 - MQTT 12 9
Building the presentation 21 2
Final presentation 24 4
Aleks
Understand the context 4 3
Task 1.2 - Python script 7 5
Task 3.1 12 9
Building the presentation 21 2
Final presentation 24 4
Matteo
Understand the context 4 3
Task 1.1 - VEB.js 7 1
Task 1.3 and Task 1.4 11 3
Task 2.1 8 2
Task 2.2 - 2.4 15 2
Building the presentation 22 2
Final presentation 24 4
Saverio
Group coordination 4 27
Understand the context 4 3
Task 1.1 - VEB.js 7 1
Task 1.3 and Task 1.4 11 3
Task 2.1 8 2
Task 2.2 - 2.4 15 2
Building the presentation 17 6
Final presentation 24 4
Leo
Understand the context 4 3
Task 1.1 on Unity 7 5
Task 2.3 17 1
Task 3.1 - 3.2 - 3.3 on Unity 12 8
Building the presentation 21 2
Final presentation 24 4

Challenges

Let's now deepen the completed activities during the JLL.

Initially, the material provided to the groups to start working was the following:

1. The **documentation**, in part comprehending the guide we developed, explaining the context and the theory necessary to solve the challenges: https://github.com/difactory/DF/blob/main/docs/AVATAR-JLL/JLL_doc.md
2. The **challenges**, to be completed as many as possible to take the most points: https://github.com/difactory/DF/blob/main/docs/AVATAR-JLL/JLL_challenge.md

by using those two links each of the group was in charge of completing the challenges within 2 days with our help and also the support of 2 more expert tutors.

Below, we outline supporting processes and a small description of the challenges:

- Challenge 1 was about getting in contact with the 3D **environment visualization** on VEB.js and the visualization of some **trajectories**. To complete this challenge was necessary a medium knowledge of Python programming:
    - We **helped** group members in resolving the points where they got stuck in their **code**, without interfering in their understanding process of the formats of trajectory.json and animation.json files.
    - We answered the participants' dobts regarding VEB.js functionalities such as toolbar, animation panel, data generation panel, and asset info panel.

- During challenge 2, we:
    - Provided guidance on using OntoGuiWeb, a graphical interface that, in our case, supported us in the sending and receiving of MQTT messages. (https://virtualfactory.gitbook.io/vlft/tools/ontoguiweb)
    - Helped on how to save the Challenge#2 sequence of messages, due to browser layout a part of the messages was not shown.
    - Helped resolve errors in the codes written to convert the saved messages into a correct and readable text-based format.
    - Provided guidance and assistance on what needs to be done for the bonus part.

- Challenge 3:
    - We assisted participants in identifying tools that can be used to create trajectories.
    - We provided help in resolving any issues or errors they encountered while using these tools.

Through challenge solving, we successfully tackled complex tasks that required **different skills** and **knowledge** thanks to the **collaboration** of students from all over the world.

In fact, knowledge sharing was critical to the achievement of our goals. We were able to overcome language and cultural barriers, creating a **collaborative environment** where each team member had the opportunity to contribute their unique skills. In addition, we were also able to create **inclusive** environment in which all team members felt valued and encouraged to express their ideas.

The **support** provided to the international students who worked with us enabled us to **achieve** our goals more **efficiently** and **effectively**.

This experience allowed us to understand the importance of having people with **different skills within a team**. This approach allowed to take on even more ambitious challenges and achieve extraordinary results in any field considered.

3. Receive feedback and update the guide

Receiving Feedbacks

Challenge Format:

- Challenge #1: Visualize Trajectories
  - T.1.1: Access the virtual environment and locate objects
  - T.1.2: Convert raw trajectory data into animation.json format
  - T.1.3: Upload animation.json to VEB.js for visualization
  - T.1.4: Measure distance between tool tip and target for evaluation
- Challenge #2: Receive a Trajectory via MQTT
  - T.2.1: Access OntoGuiWeb platform and subscribe to MQTT topic
  - T.2.2: Extract trajectory from console logs
  - T.2.3: Convert trajectory to animation.json and import it to VEB.js
  - T.2.4: Measure distance between tool tip and target for evaluation
- Challenge #3: Generate a Trajectory
  - T.3.1: Generate a trajectory based on initial and target joint angles

The main target of these challenges is to **develop skills** and demonstrate proficiency in various aspects related to **3D environments** and **trajectories visualization/generation**.

- Challenge #1 focuses on visualizing trajectories in a virtual environment.
- Challenge #2 involves receiving trajectories via MQTT (a messaging protocol).
- Challenge #3 focusing on generating trajectories based on specific requirements.

By completing these challenges, participants can enhance their understanding and ability to work with trajectory data, utilize tools for visualization, process data from MQTT messages, and generate trajectories programmatically.

After the lab was completed, we engaged in discussions within our groups to gather feedback on the challenges and identify any difficulties encountered. Additionally, through our supporting processes in the Joint Learning Lab (JLL), we gained valuable insights into the challenging aspects of the tasks.

Creation of Solutions File

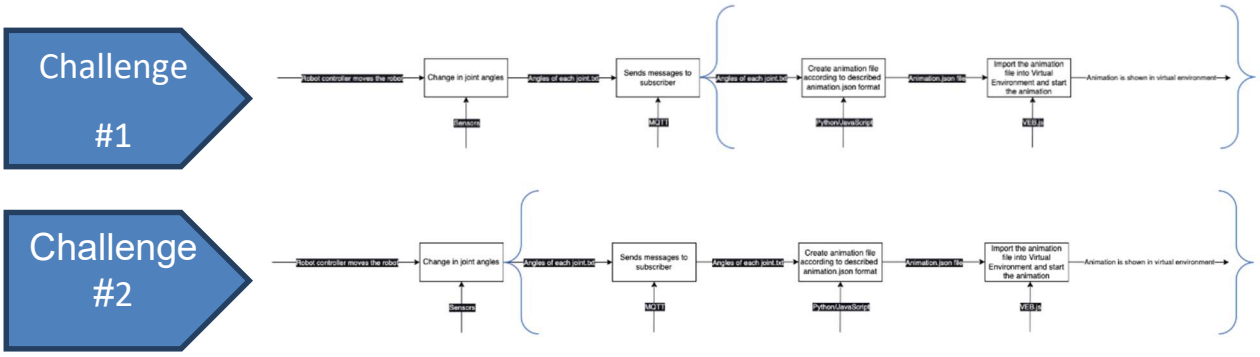
While most participants had a good understanding of the lab's structure and conventions, many faced challenges in the programming aspects. To address this, we prepared a solutions file that provides example codes, workflows, step-by-step guidance, and useful links. This document aims to assist participants in overcoming the programming difficulties they encountered.

https://github.com/enverern/SMLab_Avatar_Project/blob/main/solutions_folder%20(3rd%20file)/solutions.md

Another reason for creating solution file that includes example solutions is that group members are often assigned to different challenges. While all members may have information about the general structure of the lab and the tasks involved, they may not be aware of the specific solution for each task during the JLL event. In such cases, having an explanatory solution file at the end becomes valuable as it allows them to easily understand the complete solution process for each challenge.

Updating Guide and Challenges Documents

During the preparation of the solutions file, we some guide workflows to address specific issues. These flows are not only useful for the solutions files, but also serving as insightful hints and path guides within the challenges file. Consequently, we have attached multiple workflows in the challenges documentation.



Also, to promote clarity we have incorporated additional explanations in areas where potential ambiguities were identified, ensuring a comprehensive grasp of all aspects involved. Furthermore, we have fixed broken links, corrected typos, and implemented other necessary updates within the guide and challenge papers, enhancing the overall coherence and smoothness of the documentation.