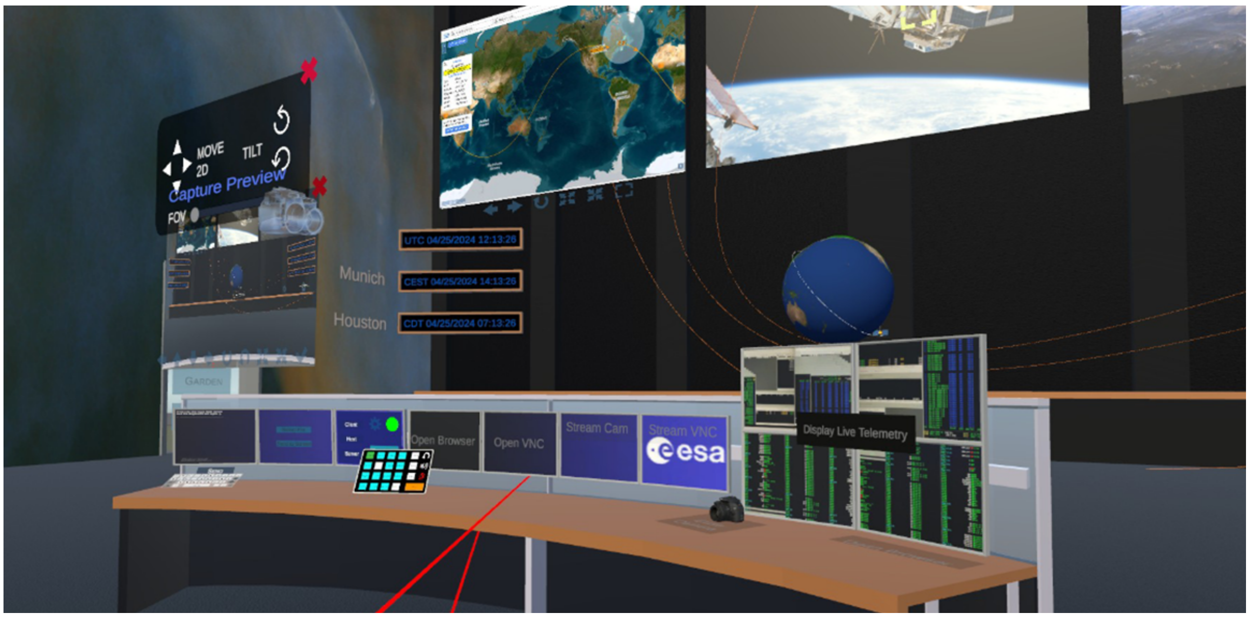


The Virtual Mission Control Room

Executive summary
early technology development

Campaign: [New ideas to make XR a reality]

Affiliation(s): Chair of Computer Science VIII – Aerospace Information Technology JMUW – Julius-Maximilians-Universität Würzburg,

Activity summary:

This activity creates a solution to free operators from the need to travel and stay in the same physical MCR but recreates the experience of collaborating within a real MCR as closely as technically feasible. The VMCR provides an adaptable collaborative virtual environment suitable for remote training sessions, small satellite operations and monitoring. It can also be used as a live demonstrator for control room redesigns. The VMCR enables universities and institutes in the small satellite community to experience working in MCR environments far beyond their monetary and office-space budget limitations.

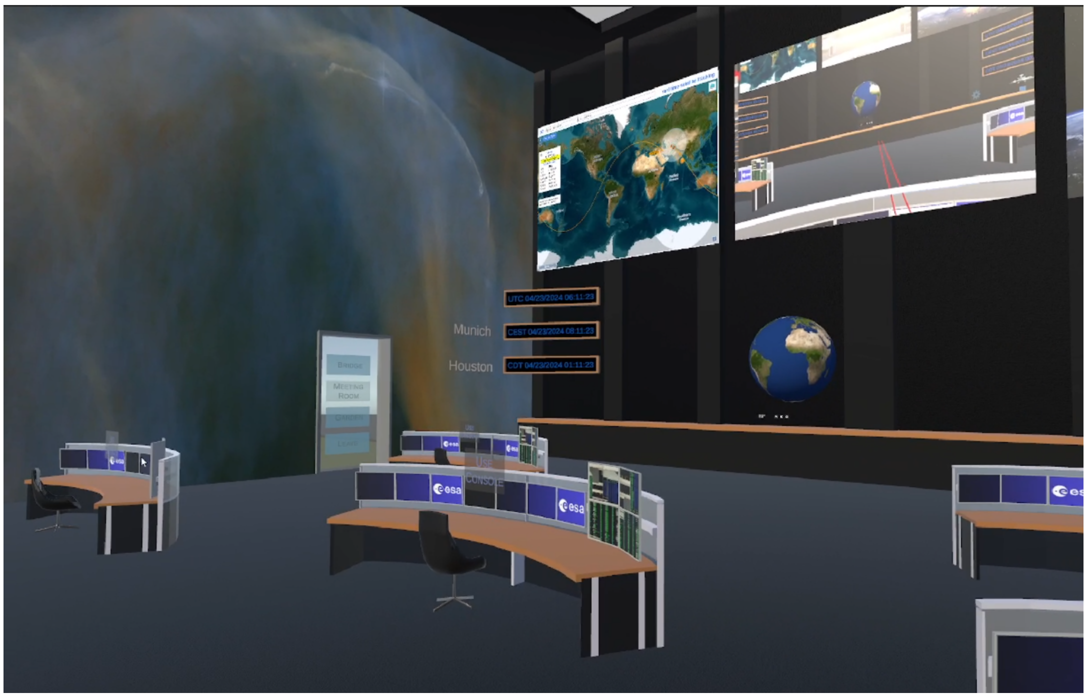
Publishing Date: 05/14/2024
Contract Number: 4000138040
Implemented as ESA Initial Support for Innovation

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für Informatik VIII
Aerospace Information Technology
Informationstechnik für Luft- und Raumfahrt

The Virtual Mission Control Room

Executive Summary



1. Introduction

For the past seventy years, mission control rooms (MCRs) have remained largely unchanged, serving as the central hubs for monitoring and commanding spacecraft operations. However, some control centers are now exploring new approaches to mission control. One such approach involves the creation of Multi-Mission Control Rooms (MMCRs), which facilitate collaboration between different mission teams. Our project, "The Virtual Mission Control Room," seeks to innovate further by transitioning MCRs into the realm of Virtual Reality (VR). The proposal for this project was developed during the COVID-19 pandemic, which highlighted the importance of alternative methods of cooperation and remote collaboration. In response to these challenges, we recognized an opportunity to leverage VR technology to create a more adaptable and resilient mission control environment.
Our project aims to improve collaboration and data visualization in mission control. Alongside standard tools like text and voice chat, and screen sharing, we're exploring innovative ways to interact with mission-critical data in a virtual environment. By transitioning operations to VR, we aim to enable effective collaboration without geographical constraints.

2. Project Objectives

The Virtual Mission Control Room (VMCR) is an attempt to free operators from the need to travel to and stay within the same physical MCR.

The main objective is to design and create a completely usable prototype, providing an adaptable collaborative virtual environment suitable for operation and monitoring of small satellites as well as operator training. The system is comprised of frontend and backend systems: The backend, designed as an easily deployable microservice architecture, takes care of server-side aspects of communication, data handling and storage, while the frontend provides the client-side communication and VR user interface within a single executable.

The VMCR features VR scenes and assets to support mission operations and teaching: a fully usable voice-loop-control panel for multi-channel voice-loop training based on Mumble/Murmur, a native VR telecommand frontend for RODOS/Corfu-based systems, screen casting, avatars, chat and tutorial scenes for new-to-VR operators. In addition, special assets, such as a floating globe depicting satellite positions, demonstrate the capabilities of presenting data in VR. Further, operators are able to adjust their active virtual environment according to their preferences. Side meetings can take place in a range of virtual meeting rooms, facilitating both the structured discussions of work group sessions the relaxed interactions akin to informal chats with colleagues during breaks.

The system can be accessed from normal workplaces, which will permit to offer operator training without impeding usage of the real mission control rooms. There are two main scenarios: If the normal workplace computers are VR-capable a secure connection to the backend server will suffice to use the VMCR from the normal office. In that case, only the frontend software and VR/XR headsets need to be distributed to experience the VMCR. Alternatively, the user-part of the VMCR and VPN software can be installed on a gaming-notebook and brought to users, tin combination with the headset as temporary installment for training purposes.
The VMCR is intended to be provided as free and open source solution.

3. Work Plan & Activity Schedule

In the following Gantt-chart, there are several frontend and backend tasks carried out in parallel, which is intended, as certain development and integration steps are highly dependent on each other. The backend system administration tasks are stretched over sufficiently long time slots to allow the backend-frontent-integration testing needed in complex system integration. Tasks to be handled by student collaborators are marked in a lighter shade of blue.

Unfortunately, we experienced some major personal shortages due to the COVID-19 pandemic and had to adapt the schedule accordingly. Hence, we suggested the following change to the schedule in the Milestone 01 Progress Meeting: the updated work plan, that ESA agreed on, prolonged the implementation time/ project by three months while not changing the cost.

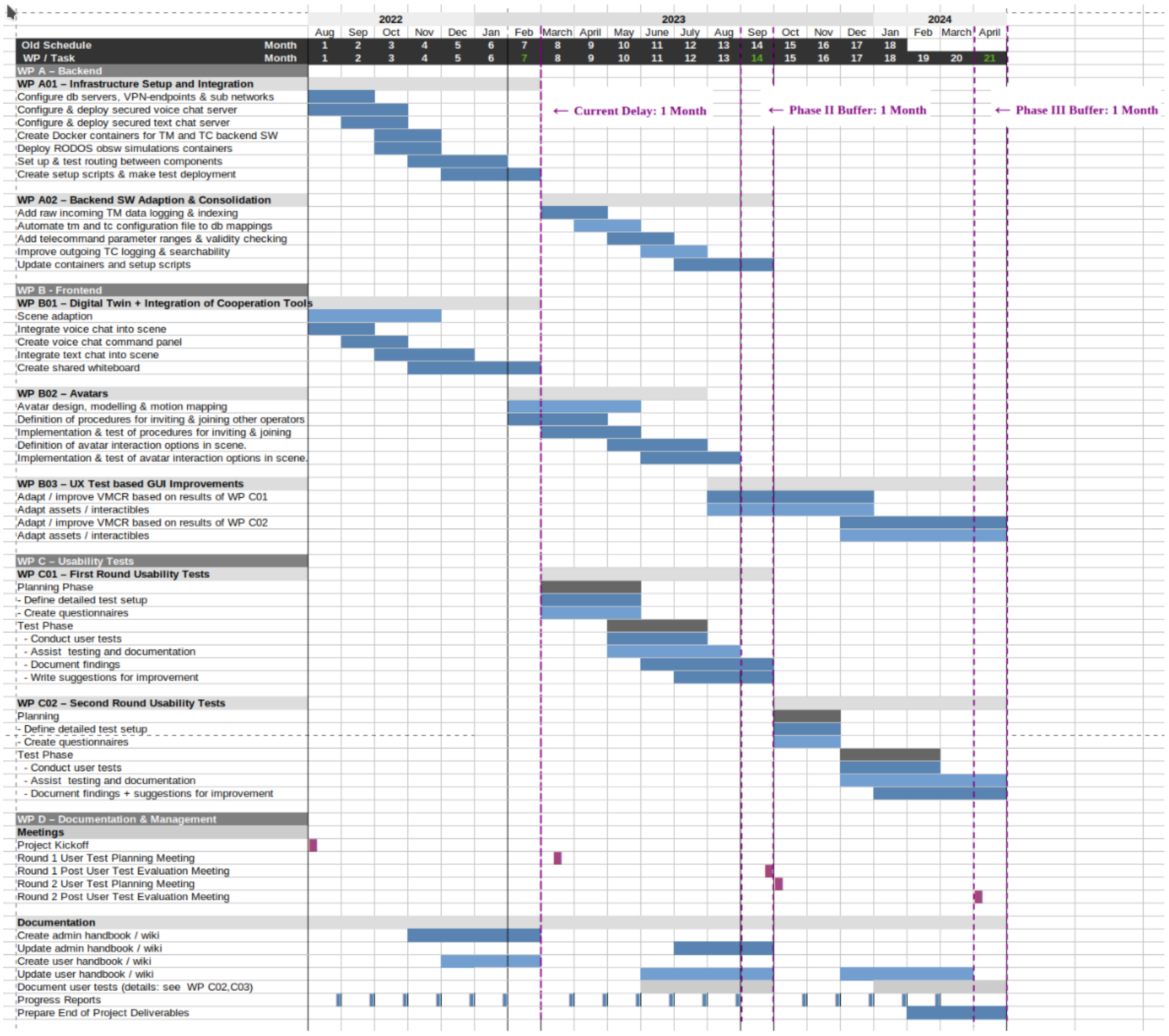


Figure 1: Updated Project Schedule

4. Achievements

4.1. WP A – Backend Infrastructure + Software

WP A01 – Infrastructure Setup and Integration & WP A02 – Backend Adaption

We redesigned the back end as microservice architecture so prospective users can switch out parts to adapt the project as needed: We choose to provide each back end service, except for the main unity-scene, within its own Docker container. These containers are linked together within in a virtual network to interact and facilitate the VMCR. While the backend is designed as a heterogeneous microservice architecture, all functionalities of the frontend software are delivered within a single executable.

Alternative Configurations:
We provide containers and configuration scripts for the following backend options:

1. A setup providing WueSpace's Telestion ground station software, including their web-based telecommand frontend plugin for Corfu-based onboard software, as seen left in Figure 02.
2. The setup depicted right in Figure 02, used to facilitate sending telecommands with our experimental auto-generated VR telecommand frontend.

In both setups, only the containers on the left side are exposed / reachable from the Unity frontend, while all other containers only interact internally.

The following containers are present in all configurations and non-exposed:

- The emulation container hosts our Corfu-based on-board software.
- The egse2influx container hosts a decoder software, that receives telemetry and stores the data in the database within the influx container.
- The influx container hosts the influx time-series database, in which decoded telemetry & telecommands are stored and made available to the grafana container. It receives decoded telemetry data from the corfu bridge.

The following containers are present in all configurations and exposed:

- The **grafana** container reads from the influxDB and serves telemetry visualizations accessible in any web browser.
- The **file browser** container enables exchanging files, from within the scene.
- The **murmur** container serves the back end for the Mumble-based multi-channel voice loop.

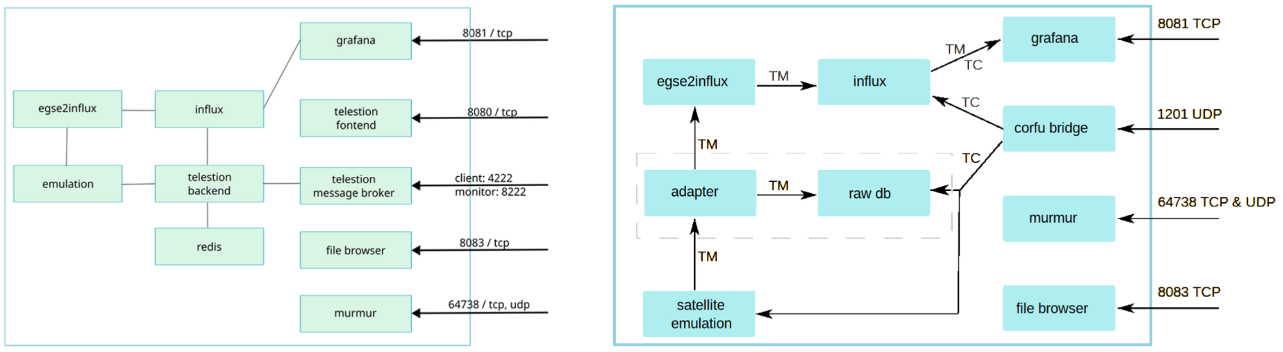


Figure 2: Network of VMCR backend containers - for Telestion (left) and VR TC GUI (right) usage

Depending, on which of the two example configurations are active, the following containers are instantiated:

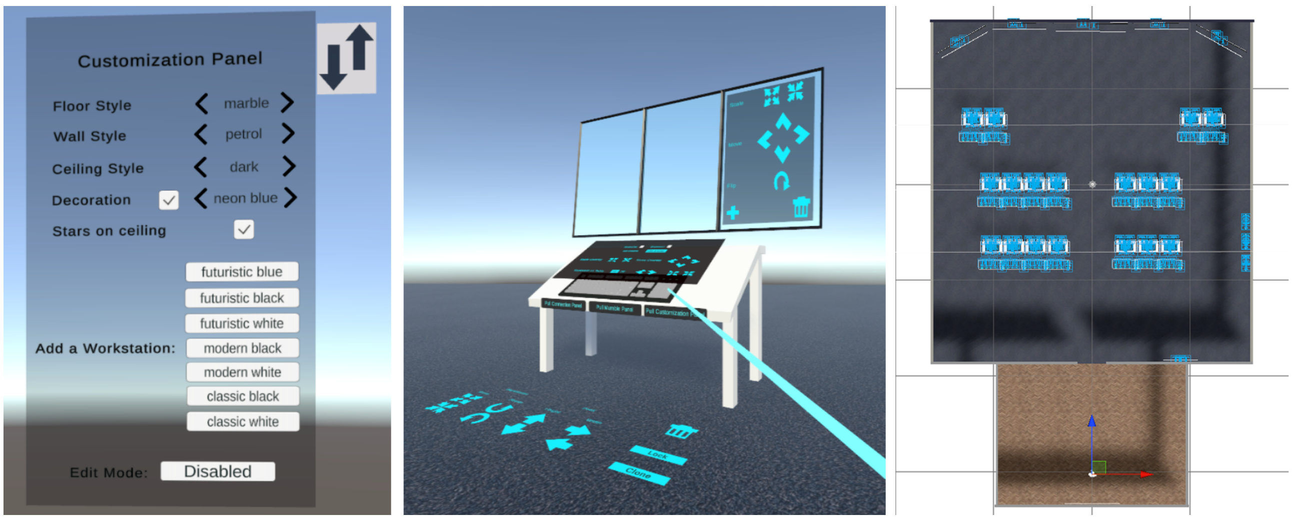
- The **telestion backend** container hosts an instance of the ground station backend. It is connected to the front end and message broker service and to two database containers, the influx instance and a redis database for long- term storage.
- The **telestion frontend** provides user authentication and sets up the message broker session between the telecommand web-app run in the operator's browser and the Telestion backend.
- **The corfu bridge** receives telecommand messages from the Unity VR telecommand GUI, creates proper Corfu-Telecommands out of them and sends these created Telecommands to the uplink. The uplink can forward the telecommand to a corfu on-board-software simulation running under Linux or a real satellite. (In the scope of this project, you'll be most likely be talking to the simulation.)
- The adapter multiplexes raw telemetry to egse2influx and the raw db.

4.2. WP B VR Scene + Frontend Integration Results

WP B01 – Scene Adaption + Integration of Cooperation Tools

Scene Adaption

This work package was implemented a student thesis. After enabling "edit mode" users could add and remove workstations, adjust the number of monitors, resize said monitors individually and rearrange them within a plane. Workstations could be scaled, moved and turned left and right. In addition, the student implemented a slot-system to add/remove front displays and assign functionalities to them. She also provided a configurable menu to assign materials to the walls, floor and ceiling of the room and switch decorative elements on/off as well as a tutorial explaining customization options to new users.



Customization Panel

Floor Style: marble
Wall Style: petrol
Ceiling Style: dark
Decoration: ✓ neon blue
Stars on ceiling: ✓
Add a Workstation: futuristic blue, futuristic black, futuristic white, modern black, modern white, classic black, classic white
Edit Mode: Disabled

Figure 3: Customization panel and overlay on futuristic workspace, right: initial digital twin floor plan

As the digital twin was created in parallel, the screenshots of the customization overlays were tested on assets from the initial futuristic scene. Some customization options had to be reimplemented as they were to closely tied to the elements of the futuristic scene provided to the student. As the code was well structured and modular, this was additional work but no big problem.

Digital Twin / Conventional MCR Scene

Like many university MCRs the MCR at our chair is a nondescript rectangular room with rows of tables and big screens at the front. Likewise was the digital twin we created early in the project and used for user tests during the implementation, as can be seen on the right side of Figure 4. In WP03, we created a virtual scene based on the MCR our team had operator training in at the DLR Columbus Control Center in Oberpfaffenhofen.

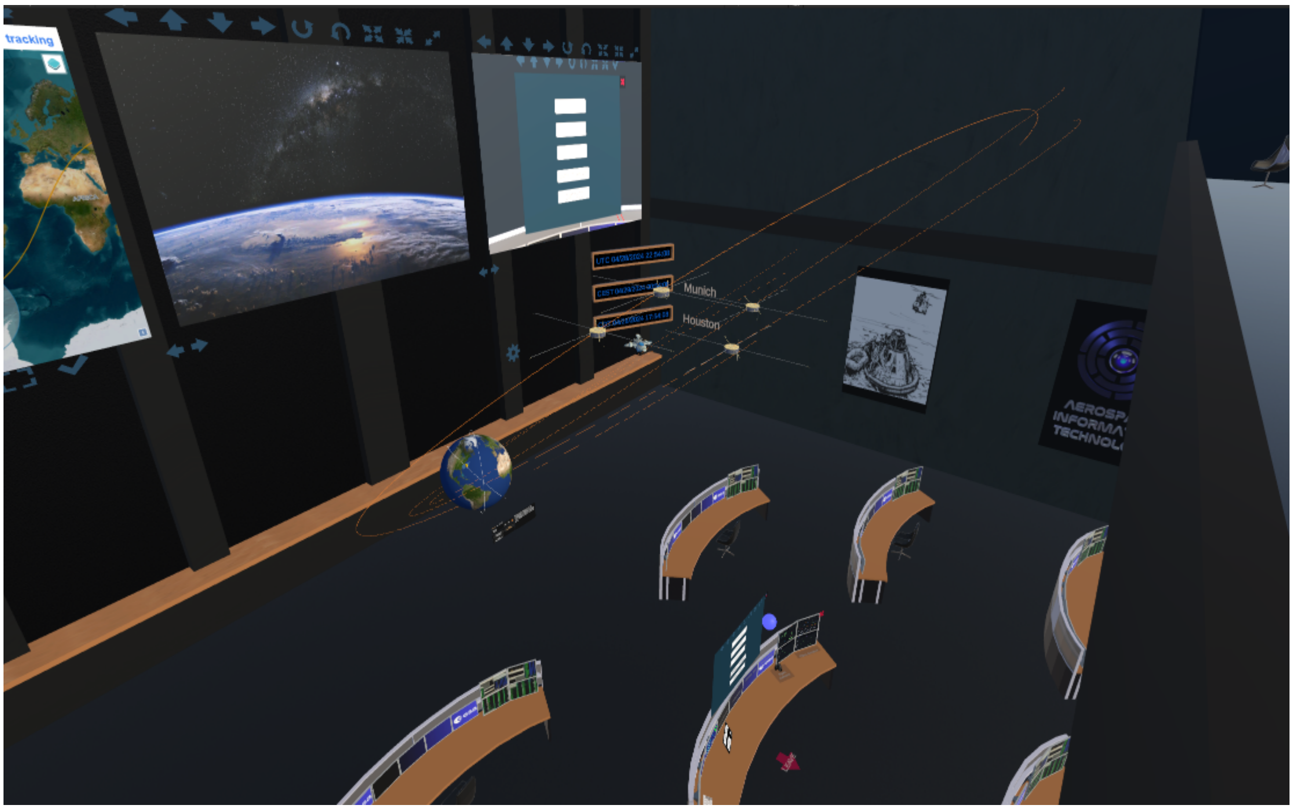


Figure 4: View from the visitors bridge into the DLR MCR inspired scene

Text Chat

The text chat has been implemented on top of UnityTransport and relies on no other connections to backend services, except for the main MCR scene. Hence it can be used as a last resort to communicate with the other operators / training instructor should voice-loop or other services fail. The GUI element is deliberately kept simple, but a more complex control overlay can be activated, e.g., to address messages to distinct groups of operators/stations. The simple text chat interface can be used as a floating asset or attached to an operator console monitor.

Voice Chat

The voice loop was implemented on top of the open source solution Mumble/ Murmur. The initial implementation was a student internship project, but has been largely redone by our staff since then.

The voice chat now features multi-channel listening, while speaking in another channel. We had to partially rewrite the used open source Unity plugin to include listening, which is a rather new Mumble feature. As suggested by ESA, we reimplemented the voice chat GUI to resemble a real MCS voice loop system. The current GUI is modeled to resemble DLR's openvocs and fully functional. The final voice loop panel design can be seen in Figure 5 on the following page.

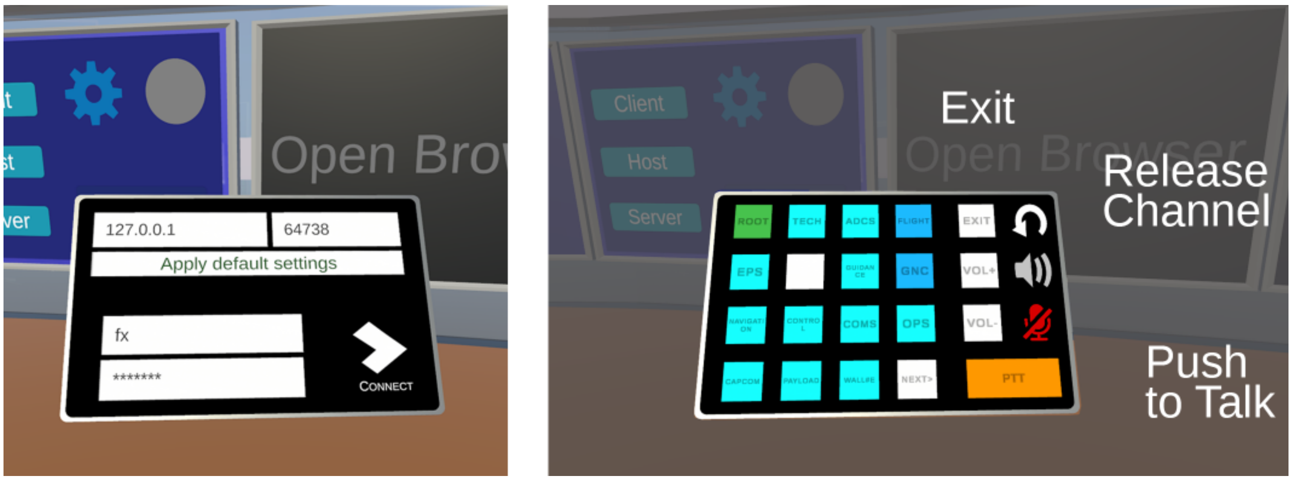
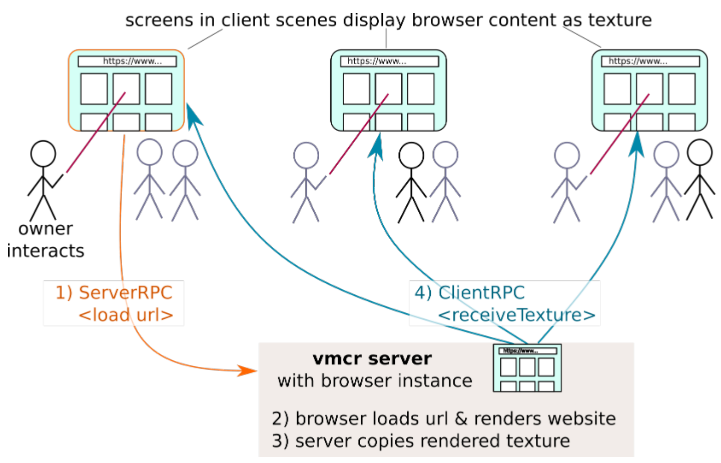


Figure 5: Voice loop control panel login screen and channel selection GUI

Shared Whiteboard/Browser

Our initial idea was to use a browser-based whiteboard solution which also allows to share website screenshots and presentations uploaded to our backend server. This was implemented as part of a student thesis in the following way:
Three of the five large front displays available within the scene at the time were configured as shareable network assets. Subsequently, a rights management system was integrated into the multi-user environment to streamline the process of claiming and releasing rights on these shared displays. The interactions of the user currently owning the shared display were transmitted back to the browser instance rendering to that display on the server via Unity ServerRPC calls.
The server browser content is then captured, serialized into bytes, and transmitted to all clients through multiple ClientRPC calls. Additionally, the sharing client's left-hand raycast could be multicast to all other clients, enabling its use as a laser pointer.


screens in client scenes display browser content as texture
https://www...
owner interacts
1) ServerRPC <load url>
4) ClientRPC <receiveTexture>
vmcr server
with browser instance
2) browser loads url & renders website
3) server copies rendered texture
Figure 6: Shared browser workflow

Unfortunately, the open source browser asset the student used in their implementation is no longer compatible with up to date Unity versions and also no longer supported. In addition, one of our ESA supervisors remarked that they had experienced bad usability with browser-based solutions like ours. Due to that, the code and GUI elements for browser display sharing were removed from the final deliverable. As part of WP B03, we developed camera streaming and VNC texture sharing as replacement assets for the browser-based shared whiteboard. Content is no longer rendered on the server, but a lot of the code was reused to build both new assets.

WP B02 – Avatars

Initially we added fully rigged avatars to the scene, but leg movement always looked unnatural and uncanny. We consulted our collaborators from the chair of Human-Computer-Interaction. They suggested compromise on floating upper-body avatars. This greately reduced the uncannyness.

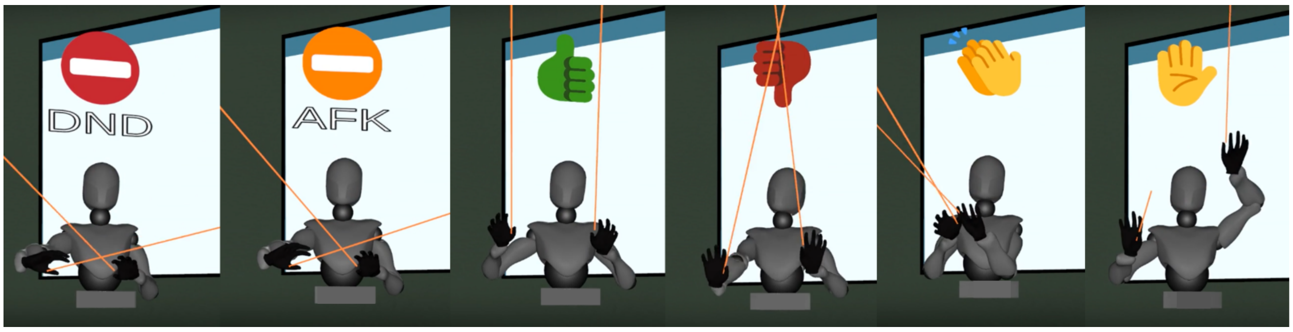
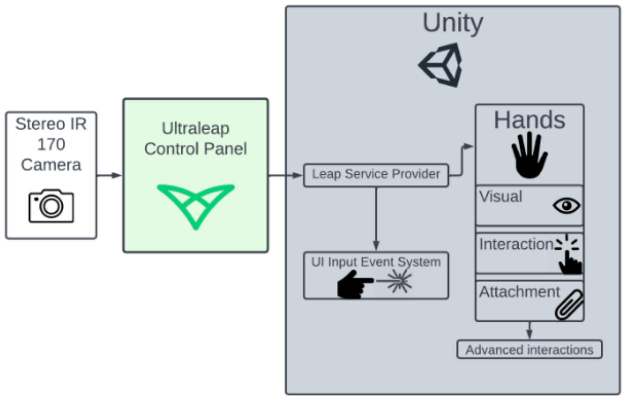


Figure 7: Avatar gestures and emotes, operator username was removed from nameplate

After adding the avatars themselves, we selected a set of gestures and emotes to represent a set of commonly used emotions, as depicted in Figure 7. Most natural gestures are enabled by hand or controller tracking, depending on the used VR headset and input controllers. Emotes: "Happy", "Sad", "Busy", "Attention!", "Thumps up/down" are shown as emoticons above avatar. Avatars can "sit down" on chairs for console interaction, when doing so, they are lowered and locked in place, but still able to turn until "standing up".

Optional Feature: UltraLeap Integration

In addition to the avatars themselves, a student integrated gesture-based interaction and motion tracking using the UltraLeap stereo camera development kit and the included licensed commercial software into the VMCR as part of their thesis. These features are not part of the main project, but optional and in a separate branch.



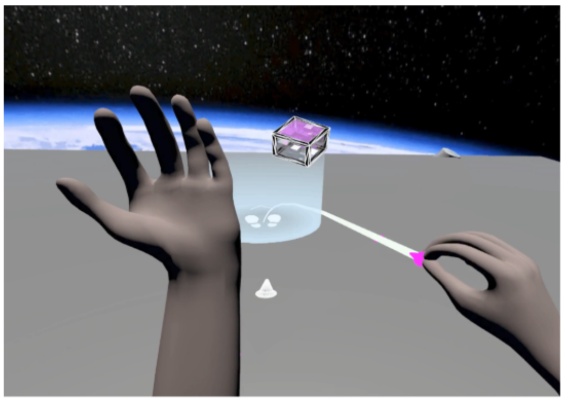


Figure 8: Ultraleap integration stack (left), gesture-based teleportation in the virtual scene (right)

The UltraLeap-based interactions are integrated into the VMCR as depicted on the left side of Figure 8. The stereo IR camera is attached to the VR headset, its Infrared LEDs illuminate objects in close range. infrared stereo camera detects the user's hands and tracks their movements. All tracking data is processed in the UltraLeap Control panel, which provides the tracking information via the Leap Service Provider asset to Unity. This data is used by the UI input event system to trigger actions and by the virtual hands to replicate user movement. In addition, complex interactions can be assigned as overlays to the virtual hands.

Apart from motion tracking to enable more realistic avatar-gesturing, this was used for a range of features, such as gesture-based teleportation, as seen on the right side of Figure 8. As it takes time to become accustomed to controlling the floating VR keyboard by ray cast, they also experimented with a more natural way to "type" in VR by integrating an open source direct touch keyboard (also provided by UltraLeap) in the scene, which felt uncanny for some test users.

4.3. WP C – Integration, Testing + Improvement:

Initial Usability Evaluation

Our partners from the chair of Human-Computer-Interaction suggested to do an initial usability assessment prior to the user testing to eliminate any obvious usability issues.

Suggested Improvements:

- Reduce movement speed and lock turning during movement.
- Re-activate teleportation to reduce simulator sickness
- Use stronger colors to highlight selected interactive elements
- Notify users when elements are spawned out of sight in edit mode
- Create a native VR telecommand GUI, as web-apps inside the browser window assets are difficult to interact with in VR.

We took to these suggestions and implemented them in WPB03 (see next chapter),

We experienced several delays in user testing due to Covid-19 rules enforced on campus (partially students were not even permitted to enter the building). Hence we could only manage to do one of the two planned big user group studies.

Study: Direct Touch versus Ray Cast Interaction

This study was conducted by a team of undergraduate student interns under supervision of the chair of Human-Computer-Interaction. The test subjects were recruited from our chairs students and colleagues previously not involved in the project. The two main objectives of the study were, first, to evaluate how deeply emerged users were in the virtual scene, and whether it caused simulation sickness and, second, to find out if interacting via direct touch or ray cast felt more intuitive to the users. The time and the number of clicks/interactions needed to complete the test scenario were recorded to also measure the objective efficiency.

The average age of the 19 test users was 30 years, all, except one of the participants were male. The main measuring method was primary appraisal, secondary appraisal (PASA), which means the mental state of the participants was captured, before, during and after completing the test scenario. The test users were informed multiple times, they should immediately leave the virtual environment upon feeling sick or uncomfortable. The users were directed to work through a (simplified) operation procedure, in which they read instructions, checked telemetry values and sent telecommands to an emulated spacecraft.

The following questionnaires were used:

- NASA Task Load Index (TXL)
- User Experience Questionnaire (UEQ)
- Virtual Reality Sickness Questionnaire (VRSQ)
- One Item Presence Questionnaire (OPQ)
- Custom questionnaire for this study

The questionnaires were filled out by the users thrice, once before and once after completing each of the test scenarios. The answers were then also evaluated by applying the classical Analysis of Variance (ANOVA).

Results:

1. Direct touch does not provide a significantly better user experience than ray casting (UEQ)
2. Ray casting offers significantly better performance than direct touch. (TLX)
3. There was no influence on the results due to differences in immersion (OPQ)
4. There was no cybersickness interference (VRSQ), neither when interacting via direct touch nor during ray cast interactions.
5. None of the test users experienced discomfort or simulator sickness.

General User Feedback & ESA Suggestions

During development, we had several people “play test” the virtual environment, and their feedback and requests also lead to changes and improvements of the VMCR.
Subsequently, we adapted the VMCR based on the suggestions our ESA supervisors offered and insights gained during DLR VOCs training we had since participated in.

Aside from smaller “look-and-feel” changes, the following major adaptions where suggested and made to the VMCR:

- Remote desktop to access other computers from within the VR scene
- Remote desktop view sharing
- General desktop/POV sharing via virtual camera streaming
- More realistic voice-loop-controls and multi-channel
- Interactive tutorials for operators new to VR

4.4. WP B03 – User Experience-based VMCR Improvements

Tutorial Scene

Several of our early test users, many of whom had never used a VR headset before, requested the addition of a tutorial on how to move and interact using the controllers.

We created a very simple tutorial scene as a first measure and subsequently tasked a student intern to create a visually more appealing interactive tutorial scene. This scene named “New Landing Strip” was included in the final delivery.

Remote Desktop Support

As accessing a mission control server via remote desktop is a common mode of operations in a real MCR, this should also be possible from within the VMCR scene.

We facilitate this by including the VNC Screen asset on each operator console instance. Given that remote desktop access is a key feature, the GUI element was strategically positioned in a central spot on the console. Operators can open the VNC instance by clicking on their left middle console screen overlay, which reads “Open VNC” or “Hide VNC” depending on the current VNC asset state. After connecting the user can interact with the remote desktop using their left or right ray cast. To achieve this, we developed the VNC VR Ray Caster script. It extends the functionality of the commonly used VNC Mouse script by translating ray intersection coordinates on the VNC screen into mouse positions for the remote desktop.

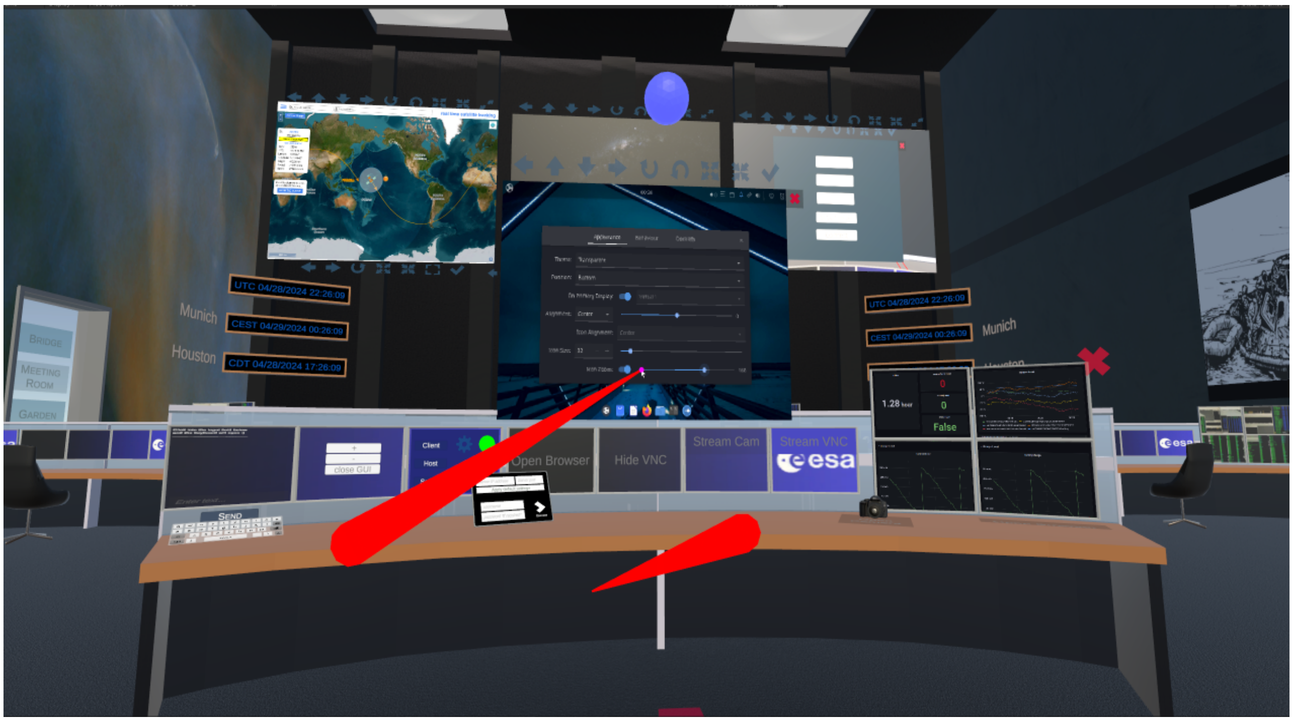


Figure 9: User interacting with a Menu on the remote Desktop

After including the UnityVNC asset into the scene, we fadded a configuration overlay in which users can enter the URL and port of the remote desktop server to connect to as well as their login credentials. Secondly, we implemented basic error handling behavior to prevent critical errors in case of connection failures. This allows users to disconnect and reconnect without needing to restart the scene We did not yet add further security measures in this proof-of-concept implementation, be aware that data is currently transmitted without encryption.

Remote Desktop Streaming

After connecting to a remote computer via VNC, operators can share the VNC screen by clicking in the “Stream VNC” Overlay in on the right frnt monitor below the VNC Screen.

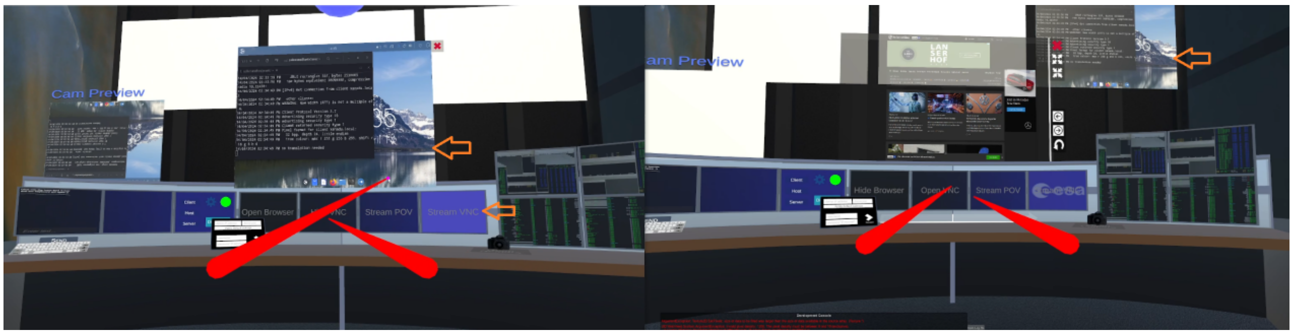


Figure 10: Streaming of a VNC session from host (left) to client (right)

Figure 10 depicts a screenshot of two instances of the VMCR run on one development computer. The left instance is remotely connected to a Linux desktop via the UnityVNC asset.
The VNC Screen contents that can be seen in front of the user on the left side are streamed to the right front panel of the user on the right side.

How it works:

When the user activates streaming, the texture is collected from the VNCScreen asset floating in front of the user, changed into bytes and streamed within multiple RPC calls using the UnityTransport protocol. As streaming consumes a lot of bandwidth, a user can only stream from one source at a time. While we undertook measures to reduce the bandwidth and compressed the data, there is still a noticeable delay between the sender and receiver video stream.

General Point of View Sharing via Virtual Camera Streaming

As users might want to share their current point of view (POV), e.g., during tutorial sessions, we also added a second virtual camera to the scene, which users can activate and reposition to record the destined part of the scene.

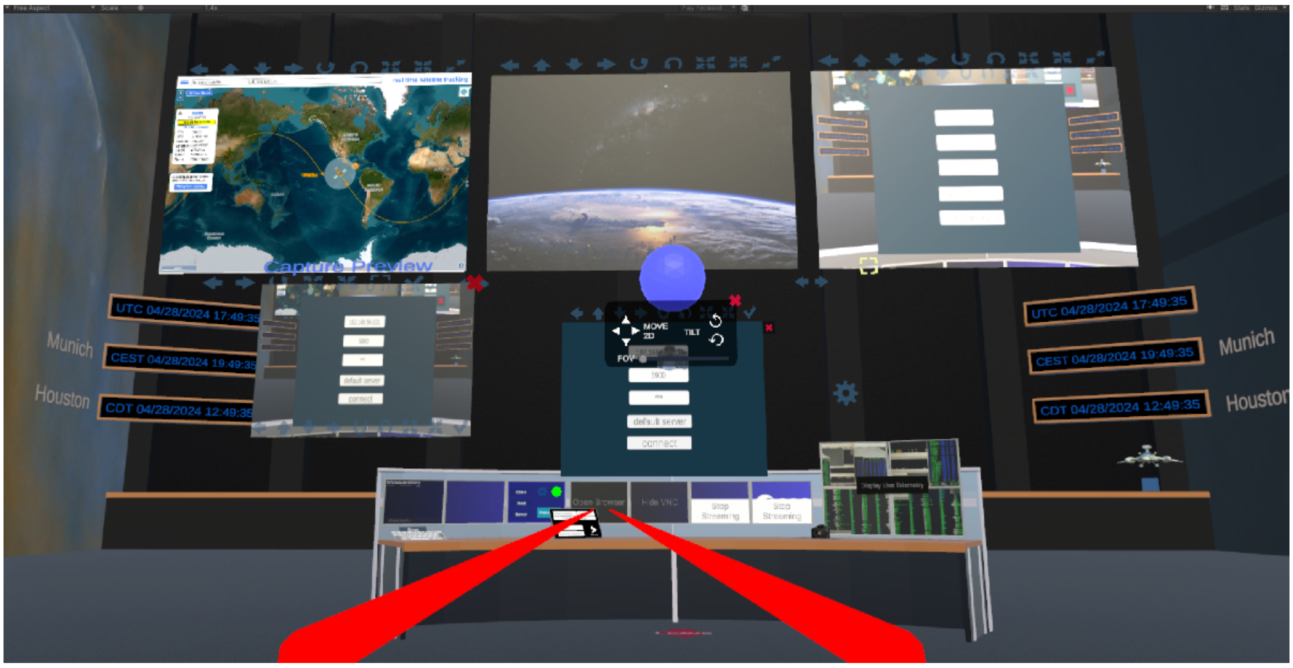


Figure 11: Movable Camera Streaming without optional Background

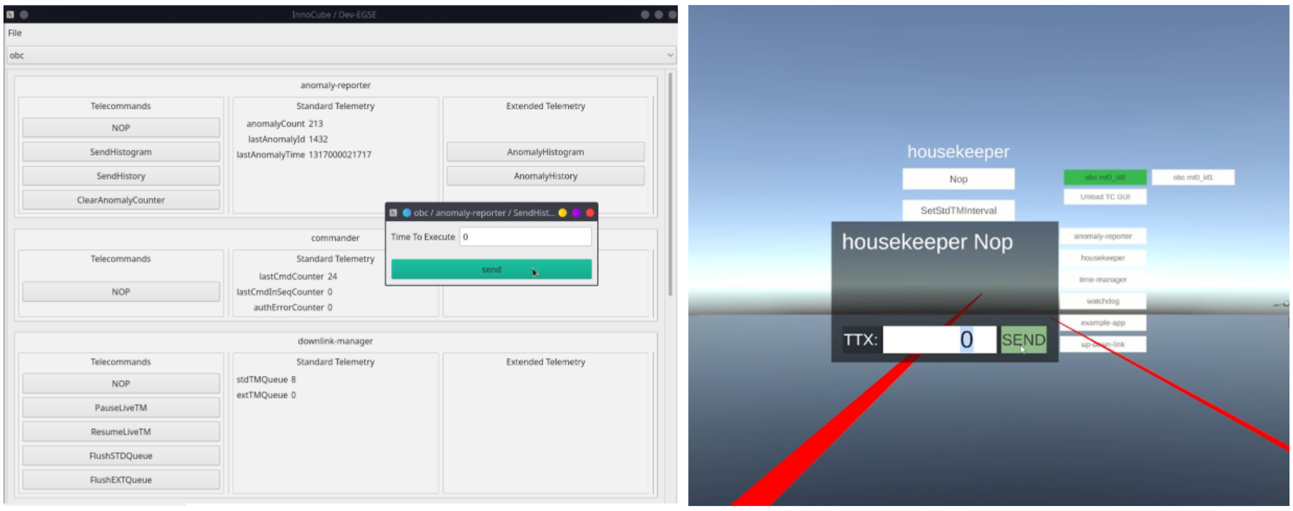
Operators can enable the streaming camera controls using the small camera model on the left side of their desktop. They can use the camera controls to move along the x- and y-axis, tilt up and down and move the camera forward and back to adjust the shot field-of-view. The captured view can be verified in the preview screen seen on the left side. Users can activate a dark background at a fixed distance from the front of the camera and subsequently adjust its size and position.

The streaming is implemented in the same way as described in the VNC streaming chapter. And likewise the movable camera / POV streaming implementation remains a proof-of-concept.
We are going to optimize the this feature after the end of the funding period, when the now experimental Unity Render Streaming plugin will become stable.

Native VR Telecommand GUI

The implementation of a VR telecommand GUI to replace the Telestion frontend accessed via browser was proposed by our partners from the chair of Human-Computer-Interaction following their initial usability survey of the existing VMCR implementation at the beginning of WP C01.

As our final operations software was still unfinished at the time, our team used a simple application named "Dev-EGSE" to send telecommands to our subsystems during development. Since our colleagues were accustomed to "commanding" their systems using this tool, we chose to begin by implementing a VR version of the Dev-EGSE for the initial usability tests. We plan to expand the VR GUI further once our real operations software is completed, which will serve as a template for development.



```
anomalyCount 213
lastAnomalyId 1432
lastAnomalyTime 1317000021717
```

Time To Execute 0

```
lastCmdCounter 24
lastCmdInSeqCounter 0
authErrorCounter 0
```

```
stdTMQueue 8
extTMQueue 0
```

TTX: 0

Figure 12: Left: minimal telecommand window on top of the Dev-EGSE GUI, right: simple VR telecommand window in front of initial proof-of-concept TC Gui elements

The Dev-EGSE is auto-generated based on our onboard software (obsw) Corfu configuration files. Likewise is the VR telecommand GUI: the operator can first choose an obsw project folder in the VR file browser. The chosen configuration is loaded and parsed, creating a hierarchical model of "nodes" (addressable subsystems), applications, and telecommands, along with their parameters. This model is then employed as a template to instantiate the main GUI, as well as telecommand GUI prefab assets for each command. An early implementation can be seen in Figure 12.

The VR telecommand GUI connects to a backend counterpart "Corfu-Bridge" to send Corfu/RODOS telecommands to the satellite simulation. The initial proof-of-concept was subsequently improved by a team of undergraduate students prior to conducting the first round user tests. The formerly floating GUI elements were fixed to a canvas and divided into multiple menu layers to declutter the scene and streamline the workflow. The formerly floating keyboard is reduced to a number pad and also fixed in place. A history of sent telecommands is permanently displayed on a panel at the right. Optionally, telecommands could also be prepared and added to the list to be sent later.

A more interesting Scene

We initially developed and tested our functional assets in simple scenes to focus on individual features. After most assets were functional we could work on creating a more complex and visually appealing VMCR scene. This is the scene depicted in most of the screenshots included in this executive summary.

The console/workstation models were created by us, are part of the final delivery, and available for unrestricted use by ESA. All menus initially created as simple free floating unity control panels, like the Connection Manager, and Voice-Loop-Control were reimplemented as monitor overlays and bound to assets such as monitors or tablets to fit into the scene.

The control room scene was completely redone to resemble the MCR we received operator training in at DLR Oberpfaffenhofen. Subsequently, we added a few futuristic elements to showcase new possibilities to present data in VR. In addition, we increased the spacing between the operator consoles make operator avatar movement easier and to give each user sufficient space for customization.

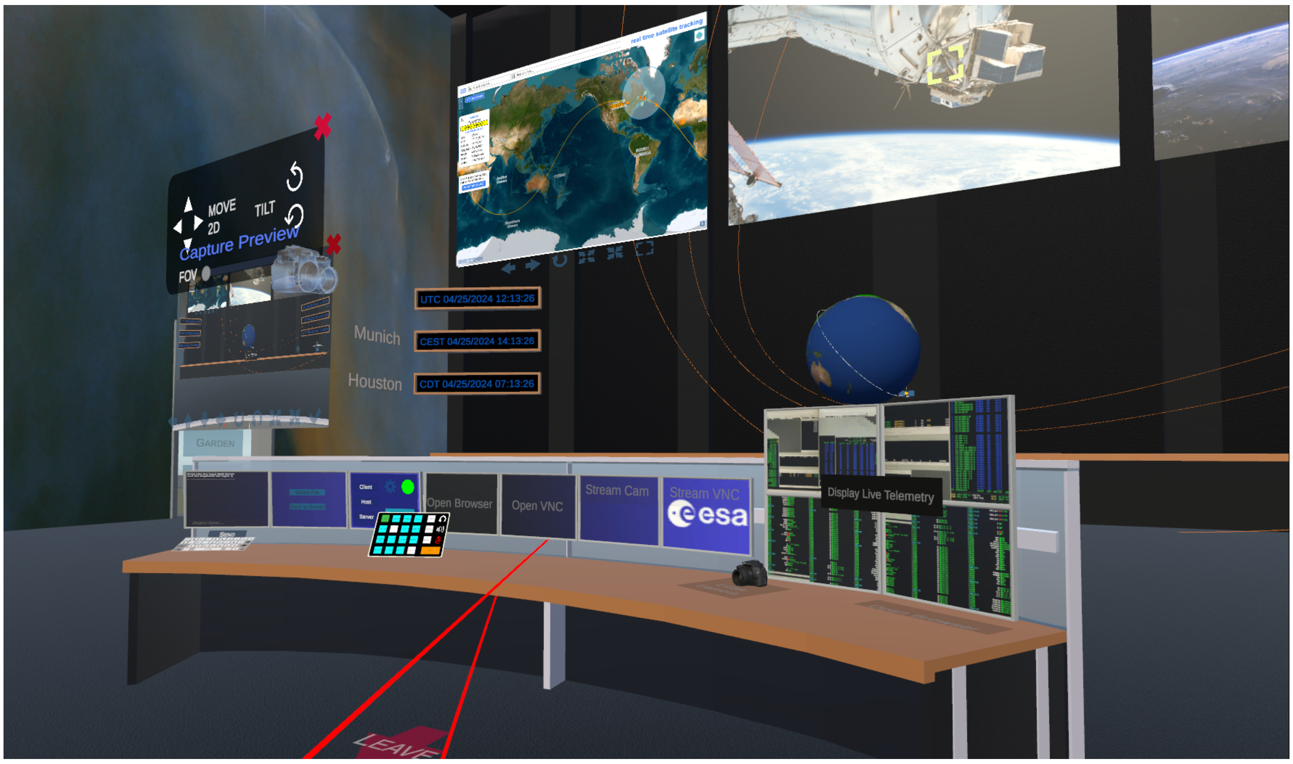


Figure 13: Operator Console with Overlay, Streaming Camera Controls & Voice Loop Control Panel

The Stellarium

The Stellarium, as can be seen in Figure 14 on the following page, is a floating globe display presenting satellite paths in a fun and intuitive way. There are two Stellarium instances in the demo: one is located centrally at the front of the mission control room itself, a second one is placed in the additional outdoors meeting area. Users can interact via the menu below the globe to select satellites / missions they would like to display, adjust the animation speed and overlay information. The Stellarium in the VMCR demo scene uses mission descriptions from ESATRACK, models from ESA Science Satellite Fleet and satellite images published by ESA.

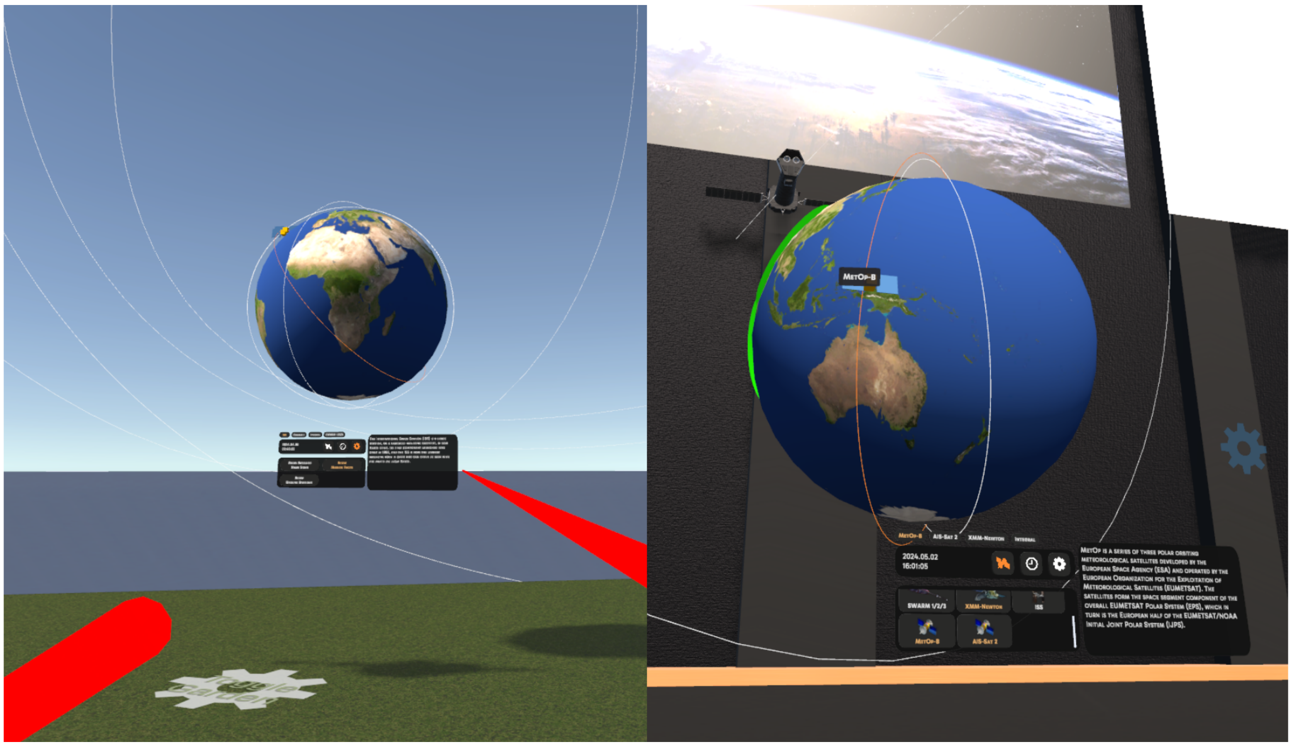


Figure 14: Stellarium Displays in the Outdoors Meeting Area and in the Front of the MCR

5. Conclusion

In this project, we developed a proof-of-concept implementation of a virtual mission control room. Covid-19 pandemic withstanding, we were able to conduct usability tests which provided us with ideas for improvement and demonstrated that the VMCR did not induce VR sickness in our test users.
At the start of the project we had anticipated making more progress towards a prototype implementation, but especially the new features added in WP03 are still in the proof-of-concept stage. Nevertheless the Virtual Mission Control Room has improved greatly throughout the funding period, in means of usability as well as visual representation. This aligns with the state of VR headset hardware development. Currently VR headsets are still lacking the comfort for prolonged use throughout the workday, but the technology is rapidly advancing and we expect this issue to be resolved within the next years. We stand by our statement that starting development on the MCR this early was imperative to ensure we will have a ready-to-use solution once the hardware reaches satisfactory levels of usability.

Development of the Virtual Mission Control Room will continue at the chair by offering feature creation as student internships and thesis topics. In addition, some of our staff are working on the open source parts of the project both at the university and in their free time.

We would like to express our sincere gratitude to the European Space Agency for funding the development of the Virtual Mission Control Room.