Research Article

Marián Hudák, Štefan Korečko*, and Branislav Sobota

LIRKIS Global Collaborative Virtual Environments: Current State and Utilization Perspective

https://doi.org/10.1515/comp-2020-0124
Received Mar 31, 2020; accepted May 07, 2020

Abstract: Recent advances in the field of web technologies, including the increasing support of virtual reality hardware, have allowed for shared virtual environments, reachable by just entering a URL in a browser. One contemporary solution that provides such a shared virtual reality is LIRKIS Global Collaborative Virtual Environments (LIRKIS G-CVE). It is a web-based software system, built on top of the A-Frame and Networked-Aframe frameworks. This paper describes LIRKIS G-CVE and introduces its two original components. The first one is the Smart-Client Interface, which turns smart devices, such as smartphones and tablets, into input devices. The advantage of this component over the standard way of user input is demonstrated by a series of experiments. The second component is the Enhanced Client Access layer, which provides access to positions and orientations of clients that share a virtual environment. The layer also stores a history of connected clients and provides limited control over the clients. The paper also outlines an ongoing experiment aimed at an evaluation of LIRKIS G-CVE in the area of virtual prototype testing.

Keywords: virtual reality, web-based, virtual environment, virtual collaboration, human-machine interaction, virtual prototype testing

1 Introduction

Contemporary virtual reality technologies provide a solid basis for the development of simulations and various activities in Virtual Environments (VE) [1]. Thanks to these technologies, we can construct immersive VE faster than a physical environment [2]. During the last few years, we observed an increased demand for shared VE, where a virtual collaboration of multiple users in real-time is possible. Therefore, Collaborative Virtual Environments (CVE) were proposed to allow groups of participants to cooperate on solving various tasks while operating over a network [3]. However, most of these CVE use a local network connection without worldwide access, which limits their possibilities.

Deployment of web-based CVE can provide global collaboration without limitation in geographical location of its users [4]. Web-based CVE possess several strengths, increasing their usability and demand for Virtual Reality (VR). Concerning cross-platform access, current VR technologies are facing hardware diversity with demands to establish software compatibility to run on various operating systems. A web cross-platform environment can simplify user access and reduce development time [5]. In contrast to the technological progress in other areas, optimization of 3D graphics and inputs for cross-platform VR remains insufficient. However, these difficulties can be resolved by an implementation of a dynamic device recognition, utilization of entity-component interfaces and automated adaptation to device performance.

The aforementioned issues sparked the development of LIRKIS Global - Collaborative Virtual Environments (LIRKIS G-CVE) at the home institution of the authors. LIRKIS G-CVE, first presented in [6], is a web-based system, built on top of the A-Frame[1] web VR framework and its extension, called Networked-Aframe[2] (NAF). Thanks to A-Frame, it is possible to run LIRKIS G-CVE on various user devices, from desktop computers and notebooks

Marián Hudák: Department of Computers and Informatics (DCI), Faculty of Electrical Engineering and Informatics (FEEI), Technical University of Košice, Slovak Republic;
Email: Marian.Hudak.2@tuke.sk
***Corresponding Author: Štefan Korečko:** DCI, FEEI, Technical University of Košice, Slovak Republic;
Email: Stefan.Korecko@tuke.sk
Branislav Sobota: DCI, FEEI, Technical University of Košice, Slovak Republic; Email: Branislav.Sobota@tuke.sk

1 https://aframe.io/
2 https://github.com/networked-aframe

through tablets and smartphones to virtual and mixed reality headsets. The integration of NAF makes the VE global and collaborative by allowing on-line multi-user interaction in real-time. LIRKIS G-CVE adds several reusable components, particularly the Smart-Client Interface (SCI) and the Enhanced Client Access (ECA) layer. SCI allows the use of tablets and smartphones as input devices. Thanks to ECA, a client may access detailed information about the avatars, representing other clients in the same shared virtual environment.

This paper presents the architecture of LIRKIS G-CVE (Section 3), focusing on the functionality provided by its A-Frame/NAF basis and introduces and evaluates the SCI component (Section 4). In Section 5, it describes the ECA layer and outlines an ongoing experiment aimed at a utilization of LIRKIS G-CVE in the area of virtual prototype testing which relies heavily on the ECA layer. The paper also deals with related solutions (Section 2) and concludes with a summary of achieved results and future plans in Section 6.

2 Related Work

Web technologies simplify the creation of CVE through various graphics libraries and standards supporting JavaScript, HTML and CSS. The work most related to the LIRKIS G-CVE is [7], where architecture for web-based collaborative 3D virtual spaces using DOM synchronization is presented. Both [7] and our work employ the same VR web framework, the A-Frame, for building immersive CVE. The work [7] implements the architecture for sharing 3D virtual spaces through a server aimed at the DOM and object synchronization while using a NoSQL database. In the LIRKIS G-CVE, the object synchronization is provided over client-server data channels via WebSockets. This process is more effective in asynchronous real-time connection when the number of clients and objects varies during collaboration. In the case of user interaction, our solution employs smart-devices, such as multipurpose VR controllers, while the work [7] uses the standard A-Frame inputs and focuses on visual output for head-mounted displays.

Another related architecture, called SimCEC [8], provides multi-user CVE via a Unity3D application, executed in web browsers. CVE distribution is ensured by a dedicated cloud server. Similarly to our work, the system evaluates the user's activity during the collaboration and processes results in real-time. The SimCEC focuses on Medical CVE, while our solution is able to cover an extensive range of CVE and collaborative applications. The LIRKIS G-CVE supports scripting through online IDE without installation of external tools, such as Unity3D. In order to enhance user interaction, the SimCEC includes a hand tracking to correct positioning of user's hands. Our system involves cross-platform VR interface, which is able to process smartphone sensor data to control 3D interaction.

The VirtualOulu system [9] uses a 3D virtual collaborative model of a town in web-based immersive VR. The system is intended for laptop computer utilization instead of wearable devices. On the other hand, LIRKIS G-CVE is designed to be adaptable for a variety of displays and VR equipment. Both solutions depend on the Three.js software library to render CVE in real-time.

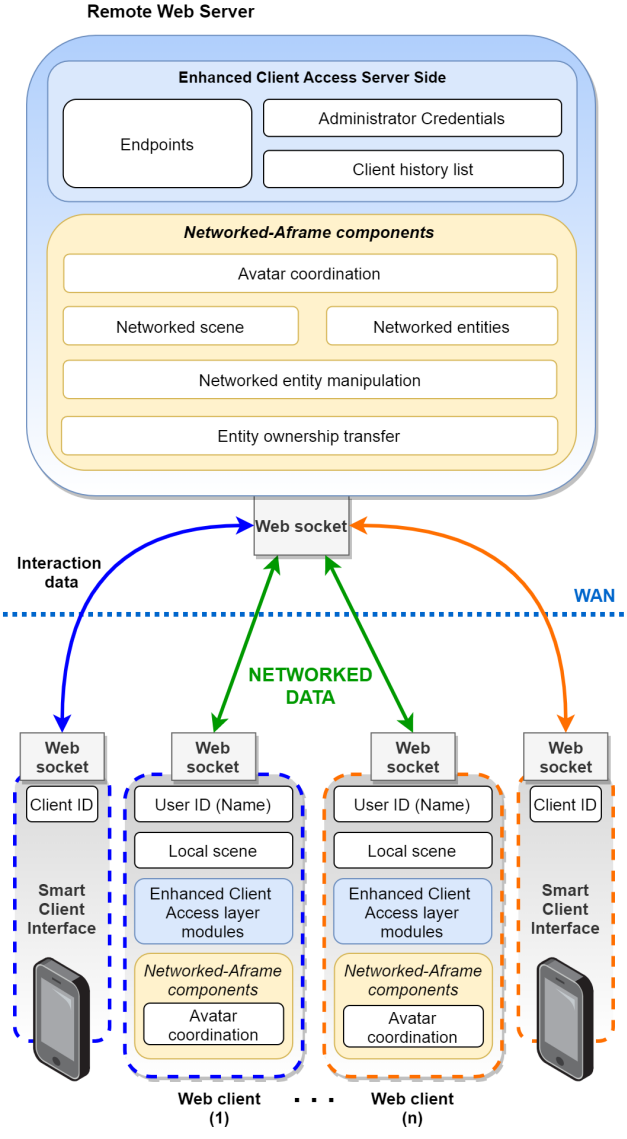


Figure 1: LIRKIS G-CVE system architecture with Smart-Client Interface and Enhanced Client Access layer.

3 LIRKIS Global Collaborative Virtual Environments

The LIRKIS Global Collaborative Virtual Environments (LIRKIS G-CVE) platform represents an immersive web-based virtual reality system with a real-time multi-user connection. The system's architecture utilizes Entity-Component-System (ECS) software architectural pattern, which offers high flexibility to create VR applications and extensions of CVE with various complexity requirements (Figure 1).

Each entity can contain multiple, fully reusable, components, which can be mixed according to the intended use. The communication between the system and users is based on client-server architecture to share data over the network. The LIRKIS G-CVE provides client-side rendering (CSR) to be able to process more complex virtual environments with a variety of visual effects, lights and shades.

3.1 Remote Web Server

The main part of the LIRKIS G-CVE is a remote web server, which is responsible for providing communication between clients and mediating networked entities in shared CVE. The server uses three JavaScript frameworks; namely, Node.js, Express.js and Networked-Aframe (NAF), all of which support full backend services. The Node.js provides real-time parallel client connection and manages the asynchronous client-server data stream. Usage of Express.js was considered as important to handle HTTP requests from clients during the connection. The NAF mediates all of the Networked-Aframe components (NAC) and synchronizes them among all clients. The NAC consist of a shared scene, its entities, and features as Entity Ownership Transfer (EOT) and Networked Entity Manipulation (NEM). The EOT creates a relation, which allows the user to take control of an object in VE and the NEM supports several object translations.

The communication between a client and the server is based on the WebSocket protocol. Each CVE is accessible under a specific URL address for clients. The server is processing all of the clients' data, including their IDs and avatar coordination. After a client is connected, the server requires its data and then broadcasts it to all of the active connections. The amount of transferred data between clients depends on properties of the CVE. These properties include interaction levels and number and complexity of shared entities. Client-server communication is bandwidth sensitive, which prevents server overloading by inactive networked entities. This feature was utilized to eliminate data transmission and reduces concurrent replication of identical data. When a client leaves a CVE, its connection is terminated by the server.

3.2 Web-client Interface

Our intention was to provide a web-client interface with asynchronous CVE rendering. In the implementation stage, the A-Frame technology was chosen with respect to its guaranteed cross-platform support and scalability through ECS. The A-Frame is built on top of the Three.js 3D JavaScript library, which is used to render 3D content on the web. By using the Asynchronous JavaScript (AJAX), it is possible to render all of the changes simultaneously and without reloading the page.

The Web-client interface provides several data elements and components for its integrity and interaction. The most important data elements are User ID and Avatar Coordination, which are sent as first after the client connects to the server. The Local scene component contains a virtual environment with all 3D entities and surrounding objects composed through NAC. To support user interaction, each Web-client utilizes an integrated Standard-VR input interface that collects all of the components responsible for user movement and object manipulation. These components are able to process data from different VR controllers and standard inputs. However, each type of input is conducted by its own component. Therefore, if a CVE requires user movement by touch and VR Gamepad, it is necessary to use two separate components. Each kind of interaction increases the number of input components, which is likely to cause a problem when users do not provide them.

4 Smart-Client Interface

As mentioned in Section 3, the LIRKIS G-CVE utilizes the ECS to provide reusable functionality mixed with different entities. However, the Standard-VR input interface contains a large number of components serving each type of input. Therefore, manipulation with a single entity by different inputs increases the number of active components concurrently. This causes their faulty integration when a user desires to utilize multiple VR input devices at the same time. It also reduces the client's performance. This may negatively affect the user interaction. An extension of the multi-purpose interface component can improve the

same functionality as the previous ones and simplify implementation of user interaction.

Considering this issue, the multipurpose VR interface component called *Smart-client interface* (*SCI*) was proposed. The SCI would eliminate the number of VR input devices by using smart devices, namely smartphones and tablets. We decided to use smart devices because of the range of peripherals they provide: a variety of sensors, a touch screen, and potential haptic feedback. Smartphones also offer more functionality than dedicated VR controllers.

Generally, each Web-client (user) can access the LIRKIS G-CVE through different platforms and devices, such as desktop computers, VR headsets, holographic computers (Microsoft HoloLens), seamlessly. These devices are able to provide visual feedback and some of them support standard inputs.

Extending the LIRKIS G-CVE by the SCI can positively affect the Web-client's performance in visual output rendering. The main role of SCI is to handle the computing of user interaction. In the same way, it is responsible for 3D object manipulation similar to VR hands, tracked controllers and 3D pointers. These features allow the user to interact with 3D objects more naturally.

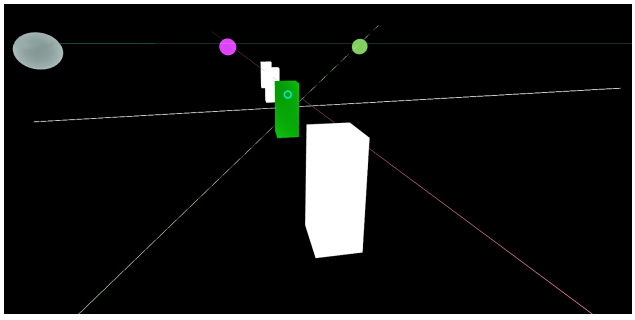
4.1 Implementation

In the first stage of implementation, we decided to extend an already existing NAF component, which was responsible for mapping device sensors through JavaScript events. The main intention was to gather all of the data that the smart-device can provide. Therefore, a dynamic device recognition was considered helpful in recognizing all the sensors and features available in the smart device. To provide 3D object manipulation, it was important to acquire data from the IMU of the smart device. The IMU stands for an Inertial Measurement Unit and usually consists of an accelerometer, a gyroscope and a magnetometer.

The accelerometer was also utilized to trigger a sensor data stream when a user shakes the device. This feature was proposed to conduct device activity, while the user is wearing a VR headset and cannot see their hands. The gyroscope and magnetometer were important to ensure 3D object orientation. To enable touch inputs for object selection and grabbing, the graphical user interface with a variety of sliders and buttons was considered necessary.

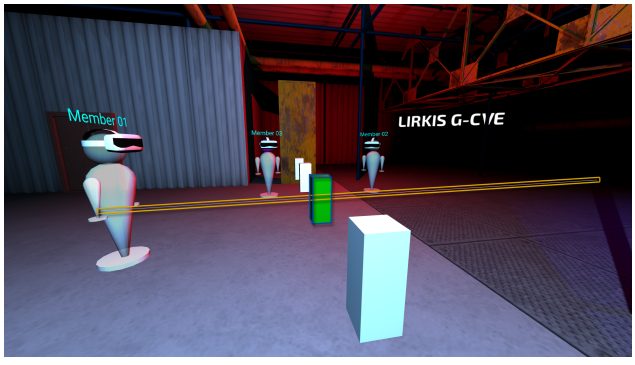
The second stage of the implementation focused on the communication between a Web-client and the SCI over the network (Figure 1). On the server side, we decided to implement pairing functions that attach the SCI to its Web-client by their IDs. The SCI shares its position, rotation and geometry with the server, which shares the data with other networked entities. In the same way, the Web-client is only observing all of the SCI activities. Considering that the SCI uses a smart device, it is able to handle all interactions autonomously and it does not overload the Web-client performance in doing so.

The third stage focused on manipulation with networked entities and interaction through the SCI. The first type of interaction was implemented using a ray-casting method (Figure 2a). The Raycaster [10] includes a 2D



(a)

(b)



(c)

Figure 2: LIRKIS G-CVE smart-client interface in practice: SCI raycasters intersection (a), hand and gaze-based interaction (b) and bounding volumes in collision detection (c).

line extended from the user towards the direction where it checks its intersection with surrounding objects (Figure 2b). Each of them can be selected and then manipulated. The second type of interaction involves 3D object collision detection between the user and the surrounding scene (Figure 2c). Each collision is processed by 3D volumes called bounding boxes (3D colliders) that are wrapped around objects [11]. A 3D collider can be formed into various shapes, such as a primitive box, a cube, a sphere or a cylinder. Utilization of 3D colliders positively affects object selection and manipulation through 3D pointers and VR hands, which can collide with surrounding objects.

4.2 Performance Evaluation

After finalizing the implementation, we performed several experiments to evaluate the results of LIRKIS G-CVE with the recently deployed interface. The purpose of the experiments was to compare the previous Standard-VR input interface and the newly created Smart-Client Interface (SCI). We decided to prepare tests of both solutions with three end-user devices:

- an ASUS FX504 SERIES *notebook*, equipped with Intel Core i5-8300H CPU (2.30GHZ), 8.00GB RAM, NVIDIA GeForce GTX 1050 graphics card with 4GB video RAM (GDDR5), 5400RPM HDD and 64bit Windows 10 Home operating system,
- a *VR smartphone headset*, where the smartphone model was SAMSUNG J5 2017 SM-J530F and
- the *MS HoloLens* holographic computer (1st generation).

In the first experiment, we measured the framerate during user interaction. The second experiment evaluated the response time between interface input action and visual output rendering. Each experiment consisted of 1000 trials and used the same CVE with 300 000 polygons.

4.2.1 FPS Measuring Experiment Evaluation

The results of the first experiment, shown in Figure 3, describe the framerate in Frames Per Second (FPS) for both interfaces. Each interface was tested with a corresponding VR controller. A VR gamepad was used as the Standard-VR input interface, while the SCI was used with a smartphone. When the Standard-VR input interface was employed, measured FPS values were lower compared to the usage of the SCI. The most significant difference of FPS rates was observed while using the MS HoloLens rendering device, where the average difference was 21 FPS. The least significant difference occurred with the usage of the notebook, which can be explained by its adaptable computing performance. In this case, the difference was 13 FPS. In all cases, the SCI performed better than the Standard-VR input interface.

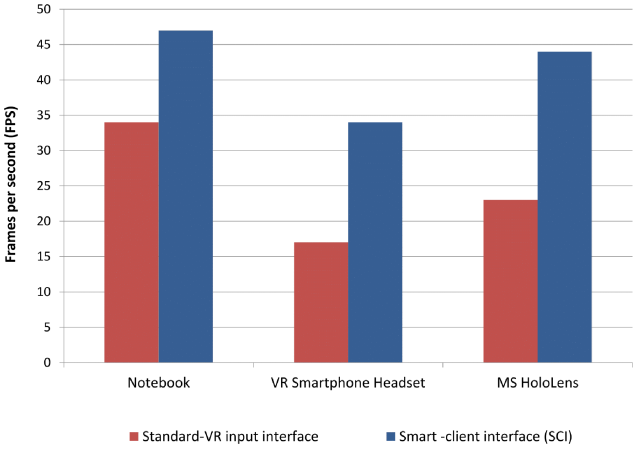


Figure 3: Rendering performance comparison.

4.2.2 Measuring Rendering Response Time

The second experiment compared the response times between interface input action and Web-client's visual output reaction. The same VR devices and interfaces were utilized as in the first experiment. The averages of response times for all 1000 trials are shown in Figure 4. For each trial, the measurement was carried out in the following way: First, the interface input action was performed and the time t_i of the action was recorded. Second, the time t_e of an event, triggered by the input action on the rendering device was recorded. Then, the response time was computed as

$$t_e - t_i.$$

The measurements confirm that the Web-client rendering response time was lower when using the SCI compared with the Standard-VR. The most significant difference was observed when the VR Smartphone headset was utilized; with the average time difference of 72 ms. Similar to the results of the first experiment, the least significant difference occurred when the notebook was used (21 ms average difference).

Based on the evaluation, the SCI interface yields positive results of testing with various rendering devices. The

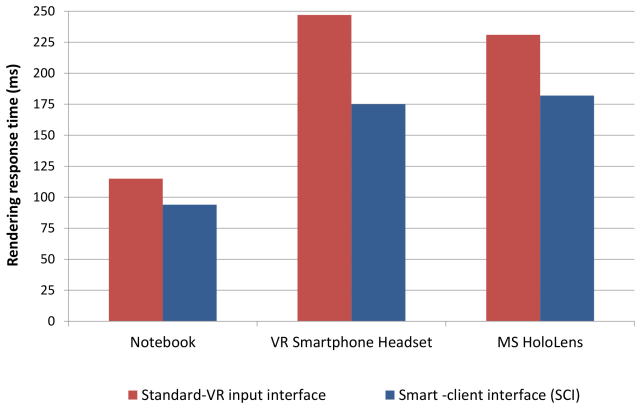


Figure 4: Measured response time comparison.

testing of the SCI proved increased rendering performance on each device. Considering the fact that the interface utilizes a smart device, it can be deployed without the need for special calibration and setup. The currently implemented multi-purpose interface improves natural interaction and simplifies user access to the LIRKIS G-CVE.

5 Enhanced Client Access

While the Networked-Aframe allows multiple clients to simultaneously access a shared virtual environment just by typing the same URL into a browser, it is not so easy for a client to get information about other clients that share the environment with them. LIRKIS G-CVE solves this problem by providing an application programming interface, called the *Enhanced Client Access* (*ECA*) layer.

The ECA layer utilizes real-time communication capabilities of NAF, implemented on the basis of the EasyRTC toolkit. The functionality the ECA provides for a LIRKIS G-CVE client can be divided into five client-side modules:

- *Client identification.* Allows a client to enter their name when connecting to a shared VE. Also provides a list of identifiers and names of the clients that share the same VE.
- *Client position and orientation.* Gives access to more detailed information about the connected clients, namely their position and rotation in all three axes.
- *Client history.* Provides a list of clients that have visited the shared VE, together with the time when they entered and left it.
- *Client administration.* Allows control over other clients in the same shared VE. In the current version, its functionality is limited to removing clients from the VE.
- *Dashboard.* Provides an administrative GUI for the previous modules, where their functionality can be accessed in a user-friendly way. The lists with client information and history are displayed in tables and the possibility of removing the clients is also available. The module is password-protected.

The modules periodically check for the clients connected to the shared VE and update the corresponding lists.

To support the modules, the ECA also includes a server component, called *Enhanced Client Access Server Side* (*ECA-SS*). The ECA-SS stores administrator credentials for accessing the dashboard module and a list of client connections for the client history module. It also includes three endpoints. The first one authenticates dashboard users against the stored credentials, the second one sends commands from the client administration to other clients in the same VE and the third one is used for the client history list maintenance.

5.1 Utilization for Prototype Testing

One of the possible LIRKIS G-CVE utilization areas that the ECA layer makes accessible is virtual prototype testing of autonomous equipment. Here, a shared VE will be used as a testing area, where one client will take on the role of the tested prototype and other clients will be testers.

The possibilities and limitations of such utilization will be explored using a VE representing a room with several places to be cleaned by an autonomous cleaning robot. The robot specification and control software will be borrowed from [12], where a similar scenario has been implemented using the jMonkeyEngine game engine and Jadex agent system.

As in [12], the robot will have a circular sensor array with eight sensors, each covering a 45° area. The robot also stores a list of places to clean. The robot, called *cbot*, will clean each place in its list while taking care of the safety of people in is vicinity. If someone gets too close, the robot will interrupt its activity (i.e. cleaning or moving).

The arrangement of LIRKIS G-CVE for this purpose can be seen in Figure 5. The clients $\mathsf{Visitor}_1$ to $\mathsf{Visitor}_n$ represent people in the room the robot is cleaning. They are ordinary LIRKIS G-CVE clients. The robot is represented by the `cbot` client, which utilizes the ECA layer, its client position and orientation module in particular. While the visitor clients will be operated by humans, `cbot` will be actualized automatically using the following procedure, executed inside the simulation loop of the VE:

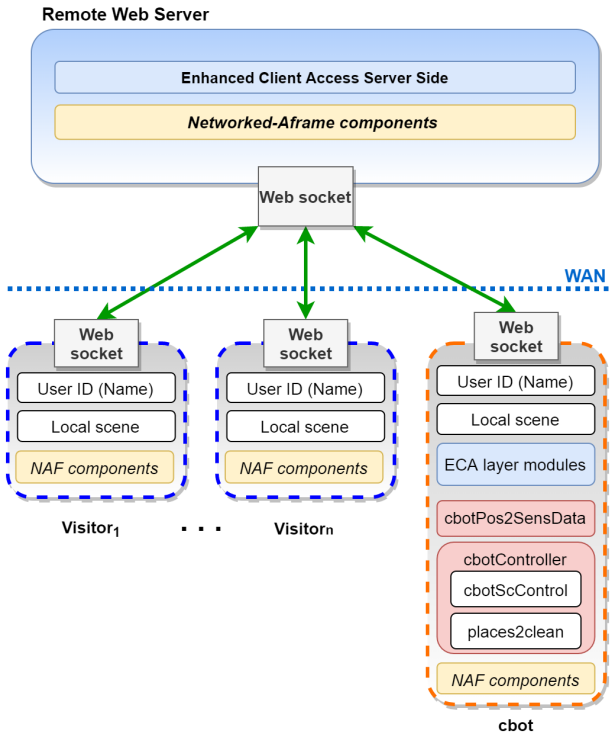


Figure 5: LIRKIS G-CVE system configuration for the cleaning robot prototype testing.

1. Acquire positions of other clients using the ECA layer and call the `cbotPos2SensData` function to transform them to the form of robot sensor data.
2. Check the list `places2clean` of the places to be cleaned to determine the next activity of the robot.
3. Check the robot sensor data to determine whether it is safe to perform the next activity.
4. Manipulate the *cbot* according to the activity chosen.

Steps 2 and 3 are performed by the control program `cbotController` of the `cbot` client and its `cbotScControl` part is responsible for Step 3.

The shared VE for cbot is already available and its appearance can be seen in Figure 6. The screenshot is taken from the position of a third visitor, who is looking at the cbot (in the middle) and at the first and second visitors. However, the cleaning functionality is not yet implemented and the robot just moves between the three positions, represented by the purple, yellow and blue blocks. The visitor clients are available at http://intersim.glitch.me/ and the cbot client at http://intersim.glitch.me/cbot.html. To set up the environment properly, it is necessary to run several instances of the visitor client and one instance of the cbot client.

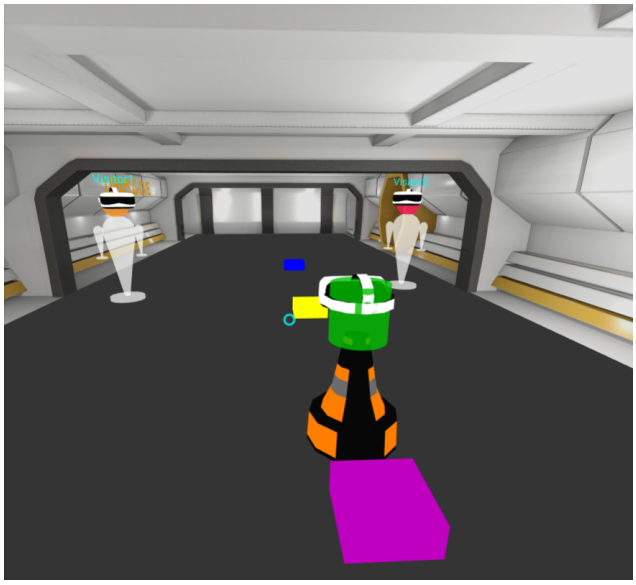


Figure 6: Shared virtual environment for the cleaning robot prototype testing.

6 Conclusion

This paper introduced two key components of the LIRKIS G-CVE online collaborative VR system, built on the basis of the A-Frame web VR framework and its Networked-Aframe extension.

The first component, the Smart-Client Interface allows the use of smart mobile devices as VR controllers, which makes VR more accessible to ordinary people. Based on the experimental evaluation results presented in the paper, we may consider the development and deployment of the SCI into the LIRKIS G-CVE a success and a significant improvement. The interface is currently used in several VE developed for LIRKIS G-CVE, including a virtual hand application to be used for motor function rehabilitation.

The second component, the Enhanced Client Access layer, is essential in providing additional information about, and control of, the clients sharing a virtual environment. Among other benefits, it allows to utilize automated clients that are aware of their surroundings in VE. This can be used for a virtual testing of autonomous equipment, as in the case of the cleaning robot environment introduced in the paper. The environment will be used to evaluate the concept of virtual testing in LIRKIS G-CVE, including a utilization of VR headsets.

Acknowledgement: This work has been supported by the APVV grant no. APVV-16-0202 “Enhancing cognition and motor rehabilitation using mixed reality”.

References

[1] Sharma, S., Devreaux, P., Scribner, D., Grynovicki, J., Grazaitis, P., Megacity: A collaborative virtual reality environment for emergency response, training, and decision making, Electronic Imaging, 2017,(1),70–77. DOI: https://doi.org/10.2352/ISSN.2470-1173.2017.1.VDA-390

[2] Correia, A., Fonseca, B., Paredes, H., Martins, P., Morgado, L., Computer-simulated 3d virtual environments in collaborative learning and training: meta-review, refinement, and roadmap, Handbook on 3D3C Platforms, Springer, 2016, 403–440

[3] Poppe, E., Brown, R. A., Recker, J. C., Johnson, D. M., Improving remote collaborative process modelling using embodiment in 3d virtual environments, Proceedings of the Ninth Asia-Pacific Conference on Conceptual Modelling, Australian Computer Society Inc., 2013, 51—60

[4] Seo, D., Yoo, B., Ko, H., Webizing collaborative interaction space for cross reality with various human interface devices, Proceedings of the 23rd International ACM Conference on 3D Web Technology, ACM, 2018, DOI: https://doi.org/10.1145/3208806.3208808

[5] El-Kassas, W. S., Abdullah, B. A., Yousef, A. H., Wahba, A. M., Taxonomy of cross-platform mobile applications development approaches, Ain Shams Engineering Journal, 2017, 8(2), 163–190.

[6] Hudák, M., Sivý, M., Web-based collaborative virtual environments to support cross-platform access. Proceedings of Poster 2019 International student scientific conference, 2019, 178–182.

[7] Gadea, C., Hong, D., Ionescu, D., Ionescu, B., An architecture for web-based collaborative 3d virtual spaces using synchronization, Proceedings of 2016 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), IEEE, 2016, DOI: https://doi.org/10.1109/CIVEMSA.2016.7524313

[8] Paiva, P. V., Machado, L. S., Valença, A. M. G., Batista, T. V., Moraes, R. M., Simcec: A collaborative vr-based simulator for surgical teamwork education, Computers in Entertainment, 2018, 16(2), 1–26, DOI: https://doi.org/10.1145/3177747

[9] Alatalo, T., Koskela, T., Pouke, M., Alavesa, P., Ojala, T., Virtualoulu: collaborative, immersive and extensible 3d city model on the web, Proceedings of the 21st International Conference on Web3D Technology, 2016, 95–103, DOI: https://doi.org/10.1145/2945292.2945305

[10] Nukarinen, T., Kangas, J., Rantala, J., Koskinen, O., Raisamo, R., Evaluating ray casting and two gaze-based pointing techniques for object selection in virtual reality, Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology, ACM, 2018, DOI: https://doi.org/10.1145/3281505.3283382

[11] Allard, J., Faure, F., Courtecuisse, H., Falipou, F., Duriez, C., Kry, P. G., Volume contact constraints at arbitrary resolution., ACM SIGGRAPH 2010 papers, ACM, 2010, DOI: https://doi.org/10.1145/1833349.1778819

[12] Korečko, Š., Sobota, B., Zemianek, P., Jadex/jbdiemo emotional agents in games with purpose: a feasibility demonstration, SNE, 2016, 26(4), 195–204, DOI: https://doi.org/10.11128/sne.26.tn.10351