Title:
Dynamic Simulator Integration in Virtual Reality Environment for Training Applications

Authors:
Franca Giannini, franca.giannini@ge.imati.cnr.it, IMATI, Consiglio Nazionale delle Ricerche
Katia Lupinetti, katia.lupinetti@ge.imati.cnr.it, IMATI, Consiglio Nazionale delle Ricerche
Marina Monti, marina.monti@ge.imati.cnr.it, IMATI, Consiglio Nazionale delle Ricerche
Luca Mantelli, luca.mantelli@edu.unige.it, TPG, Università degli Studi di Genova
Marco Ferrando marco.ferrando@edu.unige.it, TPG, Università degli Studi di Genova
Alberto Traverso, alberto.traverso@edu.unige.it, TPG, Università degli Studi di Genova
Sara Anastasi, s.anastasi@inail.it, DIT, Istituto Nazionale Assicurazione contro gli Infortuni sul Lavoro
Giuseppe Augugliaro, g.augugliaro@inail.it, DIT, Istituto Nazionale Assicurazione contro gli Infortuni sul Lavoro
Luigi Monica, l.monica@inail.it, DIT, Istituto Nazionale Assicurazione contro gli Infortuni sul Lavoro

Keywords:
Virtual Reality Simulator, Dynamic Model, Digital Twin, 3D Modelling

DOI: 10.14733/cadconfP.2023.318-322

Introduction:

The ability to represent and emulate real operations of different processes in real-time is becoming of utmost importance in Virtual Reality (VR) applications. At first, VR environments were exploited basically to visualize 3D digital contents in a more immersive manner where the user was just a watcher with limited interactions. Now, with the ongoing technology improvement, the usage of VR environments aims at improving the user experience, making the user an active subject in applications. For this reason, innovative interaction methods and novel simulation solutions are finding a wide range of applications from medicine to education, from industrial to cultural heritage. In the last years, there has been an increasing interest in the development and use of VR simulators for training purposes in industrial contexts, given the possibility of experiencing potentially dangerous situations in safe and controlled conditions [7] [8]. To be effective, such training systems should provide realistic and immersive environment in which learners can experience activities in an engaging way, where the naturalism of gestures makes their use easy even for beginners and where the behavior of the digital equipment reflects a real one, thus ensuring the transfer of the learned concepts to real world activities [2]. Such a realistic behavior of the equipment can be achieved combining VR environments with appropriate dynamic models capable of simulating the equipment operation according to the operator actions. Tools like Simulink or Modelica are capable of modeling cyber-physical systems whose evolution over time is determined by mathematical equations that must be solved in real-time to obtain the status of the system.

In this context, this paper suggests how to integrate the latest VR technologies with simulation tools in order to develop immersive simulators. In particular, we focus on the definition and implementation of an architecture for the communication between two models, a Matlab-Simulink model and a virtual environment model. Thanks to this connection, a change of state derived by user actions on virtual elements is passed as input to the Matlab-Simulink model which, once the data has been processed, provides information about the physical change that has taken place. These changes will consequently be displayed in the virtual environment providing users with feedback on the effect of their actions and thus improving their virtual experience.

The following section illustrates the issues faced and the methodology applied for the creation of this integrated system for the training of operators and verifiers of industrial equipment.

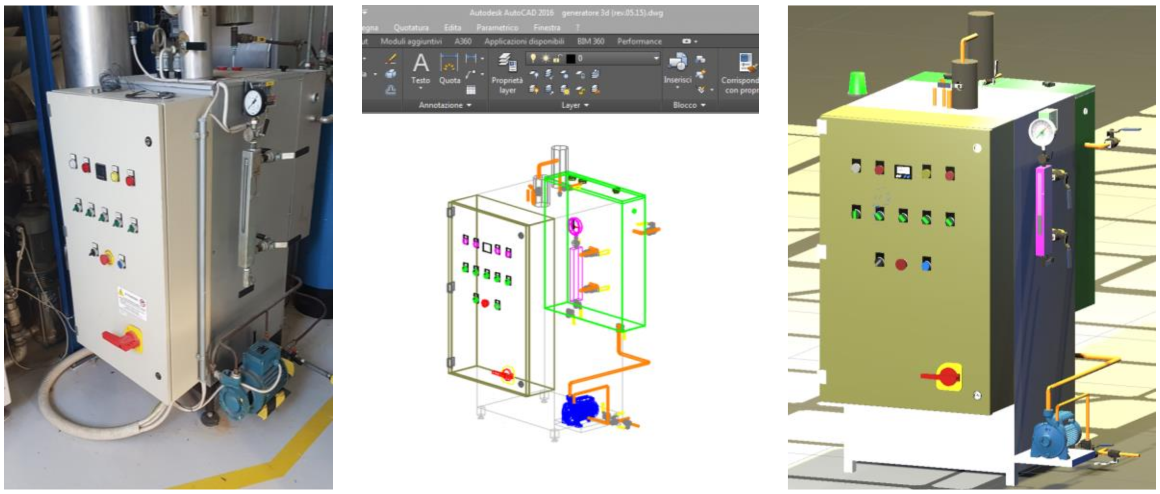
The methodology:

To facilitate the inclusion of simulation tools in VR applications, this section explains the requirements and the process of creating the two single applications (the VR application and the MatLab simulator) so that proper integration and an efficient exchange of information is possible. In particular, the focus is on the creation of the 3D digital models, the simulated physical model and the data streaming using a homogeneous structure that encodes the states of various elements present in both the applications.

Virtualization

As mentioned earlier, for some specific applications, it is important that the layout and behavior of the 3D virtual environment strongly reflect its physical counterpart in the real world. The virtualization of a real environment for immersive applications includes the reconstruction of the 3D shape of the environment and of some relevant objects, the identification of the interaction modes and the results of the user's actions. Here the focus is on the creation of 3D digital objects that users have to interact with. Modelers usually create digital contents for VR applications directly using authoring software for computer animation and 3D rendering, such as 3ds Max. Regardless of the methods used for content creation, it is essential to know which elements a user interacts with so that these components are modelled as individual elements of the 3D scene. With the aim of facilitating the creation of training applications in VR environments by limiting the intervention of game designers, we report here a workflow for the creation of 3D digital elements suitable for VR applications based on models obtained by using CAD (Computer-Aided Design) systems adopted in industrial contexts, such as SolidWorks or CATIA to name a few. In fact, pre-made CAD models of the most common equipment used for the training in industrial applications can easily be found online and, in case they are not available, can be developed more easily than other specific formats. To allow the workflow to be replicated regardless of the CAD software used, non-proprietary file formats used are for the data exchange. In a first phase, with the help of a CAD software, 3D assembly models are defined which represent real/realistic equipment. In this design phase, in addition to creating CAD models with the same dimensions as their physical counterparts, the elements that users have to interact with are modelled as individual components of the assembly. This is necessary to model the interaction behaviors of these elements once users perform certain actions. Finally, in this phase, materials and colors are added to achieve a realistic representation.

Regardless of the system used, the representation adopted in CAD software (Boundary representation) is not suitable for VR applications that require tessellated models. Therefore, it is necessary to process these models to obtain a tessellation of the boundary surfaces of the individual parts present in the assembly model. This is done by exporting all the parts involved in the CAD assembly model in an STL format while maintaining their original position in the global reference frame of the CAD system. This ensures that all the affected parts are in the correct mutual position, thus avoiding further adjustments in the VR application. The STL format is the standard format supported by all engineering CAD systems but even though it represents parts as triangles, it is not accepted by VR applications. Therefore, 3D parts in STL format must be converted to an accepted VR format, such as the OBJ format, to be included in the VR application and, if necessary, simplified in vertex count to improve rendering performance. Both of these operations can be realized by using modeling libraries, such as the CinoLib library [6], which allows processing polygonal meshes in C++ applications, or open-source systems such as MeshLab [4] which provides a set of tools for mesh editing and converting via a GUI or batch script. Fig. 1 shows an example of an industrial equipment with the representation adopted where buttons, handles, switches are individual objects in the whole assembly model.



(a) (b) (c)

Fig. 1: An example of: (a) a real industrial equipment, (b) the digital equipment reconstruction as CAD assembly model, and (c) the digital equipment in a VR environment.

Finally, the created parts can be imported into a game engine, such as Unity3D, where the interaction behaviors are defined, i.e., how the user can interact with that particular component and how the component reacts. In order to reduce the specification of repetitive behavior on 3D instances of the same functional elements (e.g., buttons), some proposals have been presented in literature that exploit part semantics, e.g. [3], [5], and [9]. In particular [5] introduces a methodology for the automatic association of behavior to 3D shapes based on their typology. In particular, the concepts of *actuators* and *control elements* have been introduced. The actuators correspond to those components on which the operator acts to command the equipment. Control elements, on the other hand, are those components which communicate the status and important internal parameters of the equipment to the operator. Examples of actuators are switches and buttons, while alarm LEDs or gauges are types of control elements. A hierarchical classification of these types of components has been defined, which enforces the reuse of characteristics and behavior. This also includes the specification of the gestures to act on them and the modification, in terms of appearance and position, that the object must assume as a response to the user's action. Each element has a specific status associated with it, which reflects the actual condition of the element. The automatic association of 3D elements to the corresponding class simply requires specifying the correspondence class - name of the 3D shape with some salient parameters.

Concerning the development of the physical model, Matlab-Simulink has been chosen, a software widely used in academia and industry which allows complex dynamic models to be structured in a simple and intuitive way, thanks to a block visualization of the various functions used. The dynamic model should include all elements with which the operator can interact, i.e., actuators as input variables, as well as the output indicators (i.e., the control elements, such as probe displays), present in the real system. When the dynamic model is connected to the VR environment, all user actions performed on the actuators, e.g., a specific valve is opened/closed, are sent to Matlab-Simulink, while the control logic of the system is always reproduced within the dynamic model. At any time, based on these inputs, the dynamic model simulates system operations and sends the VR application all the data it needs to provide some feedback to the operator. The exchange of information between the dynamic model and the VR application is done via User Datagram Protocol (UDP) as described in the next subsection.

Data streaming

In defining the communication protocol between the two environments, account was taken of the fact that the two applications can run on different computers, therefore their communication must take place via the network. Moreover, the sharing of information between the two applications must have sufficiently low latency times to guarantee the most realistic possible interaction in the virtual environment, i.e., not to compromise the real-time simulation of the equipment.

Based on these considerations, the UDP protocol was chosen for the data exchange between the two applications. In fact, this protocol is very fast as it does not manage the reordering of packets or the retransmission of lost ones, a feature that makes it interesting for the transmission of audio-video content in real time or for online multiplayer games. To enable communication between the two systems, a Windows server running Matlab must be configured to host the Matlab-Simulink application. Two UDP blocks are integrated within the dynamic model, one for receiving and the other for sending data. The receiving data block reads two messages at two different ports, a message corresponding to the status of the operable components structured in JSON (JavaScript Object Notation) format, and the length of the message representing the number of bytes of the sent string. This last operation was necessary due to Simulink's rigidity in handling inputs of variable length, while the JSON structure was chosen to guarantee a certain flexibility in case of change or addition of information.

The input data of the message read by the Simulink UDP block is encoded in ASCII format, so to process the content of the message, it is necessary to convert the ASCII input in characters data type. Once the message is converted, it presents a set of nodes representing the operable components, where each element comprises three fields:

- Name that identifies a component of the generator, e.g., "Resistance group A".
- Status that codes the status of a component in numerical format, which can be a discrete value (0-1 for a group of resistors representing the off-on status).
- Block that represents additional information in numerical format.

For example, if a group of resistors is activated by the user but is blocked by the control logics, the value will be 0, otherwise it will be 1 (default value). These elements are then used as input to the dynamic model to perform physical simulations of the system. In the sending data block, the output set is encoded in JSON with Matlab's JSON encoding function, converted to string format and sent by a Simulink UDP send block. On the VR application side, each time the operator acts on an interactable component, the datagram corresponding to the JSON (more precisely a text string in ASCII format) is sent to the network node running the Matlab-Simulink application. The reverse process, on the other hand, occurs constantly, i.e., the VR application receives the simulation results in a separate thread by an asynchronous operation that requests and receives the information of the output components. Then, the rendering of the VR model is updated according to the frame rate in the main thread with the latest result received.

Case Study

The communication system described above has been applied for the development of an immersive system for the training of conductors and verifiers of steam generators. In particular, the electric steam generator available at the Savona campus of the University of Genova has been recreated in terms of 3D layout and dynamic simulation model to realistically reproduce its behavior. In this immersive application scenario, the learners can act on the system actuators (buttons, switches, etc.) with bare hand gestures within the VR environment. Once they act on any interactable component, the state of the concerned actuator is modified accordingly and the value is communicated to the Matlab-Simulink dynamic model by changing the related input values. The dynamic model is then run in real-time, and the result of the simulation is returned to the VR application, which updates the status of the control elements and consequently their visual appearance. In this way it is possible to faithfully simulate the response of the physical system to the various actions performed by the operators, who can experience the consequences of their maneuvers as if they were acting on the real equipment. This is fundamental to understanding their mistakes in accomplishing the assigned task in a safe environment. Thus, dynamic simulation therefore allows to overcome the limits of traditional training methodologies that are barely sufficient to instruct operators for seldom-occurring perilous situations [7].To confirm the correctness of the methodology applied for the integration of the dynamic model into the virtual environment, the VR application was finally validated, defining a list of tasks that a user should perform and comparing their results with those provided by the Matlab-Simulink application performing different actions on the system. For this purpose, the developed dynamic model can also be used in standalone mode where it is possible to define the status of the input components by acting on Simulink's manual switch blocks.

Conclusions:

This paper describes a methodology applied to integrate a dynamic model that simulates the functional behavior of industrial equipment in a virtual reality application in response to user actions. The main issues and solutions adopted for the development of a training system for steam generator conductors and verifiers are described. For the future, we plan to apply and test this methodology on a more complex steam generator. To this end, a new case study has already been identified (an auxiliary steam generator of the gas turbine combined cycle power plant in Quilliano, SV, Italy). The dynamic simulation model and the virtualization of plant are currently under development. Finally, the usage of different networks can be further investigated. In fact, in the conducted tests, access to the network takes place via cable. It will be interesting to test the system with wireless connections such as a 5G or WI-FI network.

Acknowledgements:

The presented work has been partially supported by the project "PITSTOP: Piattaforma Immersiva per il Training STrutturato dell'OPeratore", funded by INAIL (Italian Workers' Compensation Authority), Bando BRIC 2019.

References:

[1] Andaluz, V. H.; Amaquina, J. L.; Quevedo, W.X.; Aguilar, J.M.; Castillo-Carion, D.; Miranda, R.J.; Perez, M.G.: Oil processes vr training, in International Symposium on Visual Computing, Springer, 2018, 712–724. https://doi.org/10.1007/978-3-030-03801-4_62

[2] Bowman, D. A.; McMahan, R. P.; Ragan, E. D.: Questioning naturalism in 3D user interfaces, Communications of the ACM, 55 (9,) 2012,78–88. https://doi.org/10.1145/2330667.2330687.

[3] Chevaillier, P.; Trinh, T.-H.; Barange, M.; De Loor, P.; Devillers, F.; Soler, J.; Querrec, R.: Semantic modeling of virtual environments using mascaret, in 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2012, 1–8. https://doi.org/10.1109/SEARIS.2012.6231174

[4] Cignoni, P.; Callieri, M.; Corsini, M.; Dellepiane, M.; Ganovelli, F.; Ranzuglia, G.: MeshLab: an Open-Source Mesh Processing Tool, Eurographics Italian Chapter Conference, V. Scarano, R. D. Chiara, and U. Erra, Eds., The Eurographics Association, 2008, https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[5] Giannini, F.; Lupinetti, K.; Monti, M.; Zhu, Y.; Mantelli, L.; Anastasi, S.; Augugliaro, G.; Monica, L.: A customizable VR system for industrial equipment operator training, Computer-Aided Design & Applications, 20(4), 2023, 716-730, https://doi.org/10.14733/cadaps.2023.716

[6] Livesu, M.: Cinolib: A generic programming header only c++ library for processing polygonal and polyhedral meshes," Transactions on Computational Science XXXIV, Lecture Notes in Computer Science, Springer, Ed., 2019, https://doi.org/10.1007/978-3-662-59958-7_4

[7] Patle, D.S.; Manca, D.; Nazir, S.; Sharma, S: Operator training simulators in virtual reality environment for process operators: A review, Virtual Reality, Augmented Reality and Commerce, 23 (3), 2019, https://doi.org/10.1007/s10055-018-0354-3

[8] Phuyal, S.; Bista, D.; Bista, R.: Challenges, opportunities and future directions of smart manufacturing: A state of art review, Sustainable Futures, 2, 2020, 100 023. https://doi.org/10.1016/j.sftr.2020.100023

[9] Walczak, K.; Flotynski, J.; Strugała, D.; et al.: Semantic modeling of virtual reality training scenarios, Virtual Reality and Augmented Reality, P. Bourdot, V. Interrante, R. Kopper, A.-H. Olivier, H. Saito, and G. Zachmann, Eds., Cham: Springer International Publishing, 2020, 128–148.