HARDWARE LIBRARIES FOR ONLINE CONTROL OF INTERACTIVE SIMULATIONS

Josef Brozek[(a)], Martin Jakes[(b)]

[(a),(b)]Department of software technologies, University of Pardubice, CZ

[(a)]mail@jobro.cz, [(b)]jakesmar@gmail.com

ABSTRACT

The article presents possible ways of connecting hardware devices with computer simulation.

Today we often see interactive simulators that truly copy the modelled system. Such simulators require extensive know-how in terms of methods enabling data transfer between the running simulation core and the controlling device that is a mere electromechanical emulation of the real system.

Our libraries are largely applicable in monolithic simulations, distributed simulations, in drive-simulators design and in simulation-based decision making. Their simple application enables each reader of this article effectively employ different input-output devices for their own simulators.

An integral part of the article is an introduction of a case study aiming at consistent verification of methods used for the libraries development and for verification of the libraries as such.

Keywords: HLA, simulator, simulation, Java FX, Java, HLA-VA, Rawsberry, hardware, microcontroller, army simulator, car simulator.

1. STRUCTURE OF THE ARTICLE

The first part of the article defines the problem by introducing signals to be processed. Readers may feel that this chapter is rather abstract due to the lack of specification of input-output devices to be used. However, the article is structured in a way that introduces particular devices in chapters four and five.

Chapter four deal with the software layer. Selected programming languages are introduced, advantages and disadvantages presented in terms of particular hardware component controls and an introduction to a more complex distributed simulation concept is presented. HLA was selected for distributed simulation: this chapter gives reasons why HLA and how its properties were used in individual libraries design. Readers may find out more about the ways of use of our library in the environment of implicit and other programming languages.

In the course of development, individual solutions were designed, where the task was to connect the input-output devices with the simulator; therefore it was inevitable to form simple application (or rather application-hardware) prototypes. Chapter five describes individual prototypes with the aim to extend the basic reader´s notion of possible ways our library can be used.

Besides individual prototypes, we designed an extensive case study which is presented in Chapter six. The sixth chapter summarizes the whole solution and presents the premises for possible future development.

2. SITUATION SEARCH

In current situation exists models, which using different hardware input devices for control of simulators. But publications about it, focuses primarily on the major topic of simulation (submarine, train) and do not help community to understand, how connect hardware drivers into their simulations.

We used space between current simulation teams focus, and made very simple libraries, which should enable of simple connection hardware devices with HLA (or different) simulator.

Then, with our scope there is done relatively original work, which can be used by many developer teams for their works. Simply put, our work is professionally profiled on the border between software engineering and simulation architecture. It publishes real libraries, which after installation transfers data directly from I / O devices into the simulator (if you want to use in the HLA, you still need to adjust OMT).

3. PROBLEM DEFINITION

Recently designed interactive simulators show very good results in real visualization. Besides the standard computer simulators, drive-simulators or emulators have been often used. The difference between a high quality simulation and a high quality emulator rests in the fact that the whole simulation runs on a computer, including optional user interactions (i.e. parameterization of simulation calculation under the running simulation). The emulator carries out the

calculations through a computer; however, their input-output devices are a true imitation of real input-output devices. An automobile emulator thus has a real steering wheel, pedals and gear shift and is explicitly controlled by these devices. An automobile simulator may have a very sophisticated physics and dynamics of the drive, etc.; however, as it is only controlled by a keyboard, it is a mere simulator.

Besides the above mentioned, there is another specific simulation type called "live simulation" which is mostly dealt with in the military sector. This highly sophisticated simulator type enables VDU and vibration units (and other components) wearing users real engagement in virtual reality. At all events, our libraries have been optimized for a live simulation use.

Simulation, live simulation and emulators can be simply named as simulators. Simulators are in principle used for two main purposes: decision-making and training. The existence of input-output devices corresponding to the true image of professionally trained personnel is of the key importance for the success of simulators. You can access [Sinclair, 2013] for more information.

3.1. Types of input-output values

We can briefly characterize the types of values that a simulator is expected to work with. They are divided according to the direction, type and the processing demand.

Before focusing on individual types, it is essential to determine the actual system to be observed. We can explain the preceding proposition on an example: a simple steering wheel can be approached from several different angles.

The basic view is a very high abstraction defining the steering wheel as a device in the state "steer angle". However, we would fail in a real system since we must take in consideration also the steering dynamics, i.e. the monitored system except the steer angle must be extended by e.g. monitoring of dynamic change, because the steering speed can be of the key importance in certain parts of the simulation.

If we extend the view, the same steering wheel can be simultaneously used as an output device – the simulator can automatically manipulate such motions as the vehicle yank in a road wash-out, etc.

3.2. Characterization by direction

The essential data characterizing feature is direction, i.e. the determination whether input or output data are in question. Many devices at real simulators are input and output elements at the same time.

The following are simple examples:

- Input device: a button for the simulated industrial operation start up
- Output device: an indicator displaying danger for train operation on railway
- Combined device: a steering wheel taking the input data from the user; however, at the same time enables an output in the form of variable resistance in turning in different terrain or according to the selected simulated vehicle.

3.3. Data types

The basic types of data are the following: continuous, digital and logic. Continuous data in driving trainers are always transferred into digital data by means of a/d converters. It is important to mention that minimum deformation occurs because even the most basic converter that transfers continuous data into a 1-byte value achieves acceptable accuracy. In our example, the continuous value is transferred to one of 256 values – a relative deviation of any step is 0.4%, which is much a higher accuracy than a person can achieve with his/her physical engagement.

Digital value processing in information technology environment is very easy and simple to use in abstraction. It is convenient (already in the problem analysis) to select a specific data type, i.e. logical (Boolean) variables that might reach only the "true – false" values. Logical value processing not only is fastest but also easy to integrate in input-output devices of the lowest level.

3.4. Processing demands

The data must be also characterized in terms of their processing demands due to close relation to quality validation and verification. In the event that an input-output device is formed on the lowest level and all data administered by a software architect, then the processing demand, as well as testing, are very high.

In addition, there exists a real risk of a blind point in the testing (i.e. situation where everything appears to be functioning until a very specific situation occurs in which the solution is non-functional / inaccurate).

A suitable way of reducing the processing demands is the use of already existing solutions and software libraries, mainly of a commercial nature. The best case of such commercial solutions is affordable price focusing on the segment highly similar to that of training devices (especially computer games segment).¨

4. OWN SOFTWARE

It is inevitable to concentrate on the next solution level, i.e. the software level. Despite the article focuses on a very low level of simulation (in the following order: simulation theory – practical solution – architecture of simulators – libraries for simulators) and the hardware aspect is dealt with as a mere overview, the principles used in software are described with more details. The reason for this is the fact that another potential user should know when the solution can be really used.

4.1. Software description

Major part of this work is software-based, then, there is need define parts of used software.

4.1.1. HLA-VA

Java FX was selected as the basic programming language because it provides a quality support for the scene animation and the code can be performed from Java programming language. This aspect is of a high importance because the HLA-VA framework in the basic scheme is implemented directly for Java and will be used in our study. Since the objective is to employ also distributed simulation, we can use the fact that pRTI provides a library for Java language.

To perform the code, a virtual machine is required (JVM); however, this is a standard approach in modern languages (most other languages use .NET virtual machine). Languages of C/C++ family are an exception: they are in fact inapplicable for high quality visualization without extensions and frameworks they use through virtual libraries. More information at Brozek, Onggo and Kavicka (2004).

For further versions and quality 3D graphic supported views can be used options of OpenGL, including its approaches (non-object programming) and specific programming languages.

4.1.2. Use of drivers

Certain hardware types have drivers installed by the manufacturer, which can be successfully used. The programming principle is described in Figure 2 The application logic itself can lean on the driver interface, therefore in the event of hardware replacement for a device whose driver has a corresponding interface (e.g. joystick), the solution will keep functioning correctly. For more information see e.g Henninger, Cutts, Loper (2014)

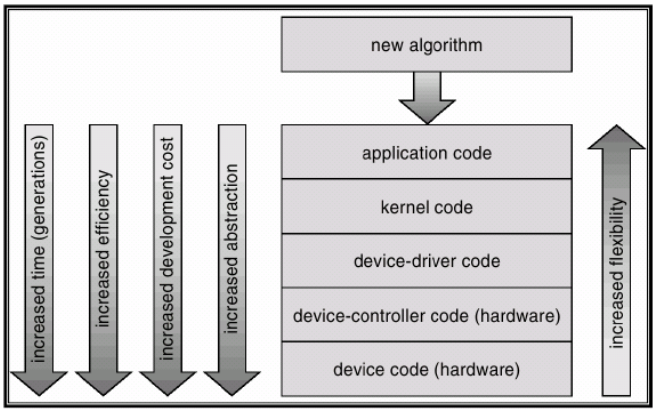

Figure 1: Programming lelvels [5]

4.1.3. Driver-free solution

Driver-free programming can be approached in several different ways, where the most effective is formation of a generic scheme that will later be capable of parametrization. All of principles are low-level and are under scope of this article. But principle which demonstrates that each individual output of the device can be set up as input - output and, at the same time, data collection can be characterized (upon request / after a change, etc.) from Huddleston (2007).

4.1.4. Interface of our solution

Interface of individual libraries is designed in an open way, but intuitively.

For the connection to actual simulator, the designer is only required to connect the hardware device, link the library and implement interface methods for the appropriate device.

The input -output interface within the library is provided independently. Should the application require a combined device (i.e. both input and output), it is essential to implement two interfaces.

Complications may occur in the event that connection of several input or output devices of the same type is needed (with the same drivers) in one application (e.g. connection of two steering wheels). This solution is currently not applicable within our libraries, because the construction does not allow for the use of parallel devices (the same solution used by most commercial software types).

4.2. Use of our solution in SW

In the event that JAVA language is used, the link of drivers coded in the same language is very simple for desktop, website and mobile applications – i.e. mobile device solutions.

Our libraries can be used in other programming languages; however, the use is slightly more complicated (although simpler than implementation from zero).

5. PROTOTYPES

The existence of input devices, their drivers and control libraries are of abstract nature and very difficult to grasp. An integral part of individual libraries formation is, among others, formation of simple software prototypes whose objective was a thorough testing of the appropriate libraries. Validation and verification of a particular library was carried out simultaneously.

5.1. Prototype 1: Automobile

At first, a simple prototype was designed. A set of steering wheels and pedals was used that was originally designed for computer games. This solution facilitated successful validation and verification of a library for PC steering wheels connected by a USB, which is very important especially in regard to the fact that it is a wide spectre of applicable devices that makes the library very useful. The library is limited to the use of basic functions (steering wheel turn, pedal operations, gears changing and six additional buttons at the most). An advantage of the library is implementation of the feedback interface, which transforms a simple input device a combined one.

The prototype was used for the needs of our case study (see Chapter 6).

5.2. Prototype 2: Heavy gun sighting system

Library for driving a joystick was tested on an example of a machine gun turret control. As in the previous example, the connection was made via USB. The same prototype was run on a tablet, which tested the libraries on another platform. The prototype itself was formed with the stress on its integration in a more extensive case study, which was performed later (see Chapter 6).

5.3. Prototype 3: Connecting online simulation, sensor network

Besides the distributed interactive simulation, our solution also aims at online simulations. We decided to carry out two case studies within the prototypes, where the first one is more extensive. Generally speaking, sensor network online reading is concerned, where actual input devices were used (buttons and levers) connected to individual pins of a Raspberry PI. The prototype (for complex connection see Sinclair (2001) and Richardson (2012)) serves for demonstration of the approach trivialities in the event that our libraries are used.

An input device can be formed only by means of a PI Raspberry, our library and almost any connective devices (even by two wires connected to the Raspberry). This prototype presents a key factor for the formation of any type of input-output devices. USB feeding (optionally from a mobile phone power bank) facilitates the creation of a super-portable computer that can be used in a live simulation since its weight (including power bank) does not exceed 400g.

The most complex solution made within the prototype was sensor network. Its objective was to emulate the security device for operation on railway. The total of 10 sensors with logical value can be evaluated and the data immediately transferred to the simulation. This solution can be used especially in decision-making simulations.

5.4. Prototype 4: Connection of online simulation, status indicators

The last prototype (chronologically developed as the first one) is a solution that employs low-level approaches. Microcontroller AtMega was used for data processing and input devices and mainly output devices were connected directly. Although it is the most economical solution, its implementation complexity outweighed all the others. In addition, the libraries written for the needs of this solution were eventually eliminated because their function required a specific sensor type. The competitive price cannot compensate the lack of comfort in further development. Although the prototype with a microcontroller can be a functioning solution, it is currently considered to bring more drawbacks than benefits. The article lists this solution rather for the purpose of complexity; the recommended option is clearly the use of Raspberry PI.

6. CASE STUDY

Simple prototypes cannot fully demonstrate the suitability of solutions, therefore a more extensive case study was carried out.

For more information about construction of HLA simulation models see Kuhl, Dahmann, Weatherly (2000), Manling (1999) or Rabelo and col. (2014).

6.1. Demonstration

Model for the case study was a military vehicle displayed in Figure 3. The vehicle has two important parts: the first one is driver station, the second is shooter station – the latter usually sits on the front- passenger seat and the heavy gun located on the armoured vehicle roof is controlled electronically.

Figure 2: Single car from case study

Figure 3 shows the vehicle drawing and its division into individual simulators. The first – driver simulator – contains a standard steering wheel and pedals. The second simulator is designed for the shooter and its position is emulated through joystick. Distributed interactive simulation thus demonstrates two persons in one vehicle, while each person can have his/her own point of view. One person controls the vehicle and the other manipulates the gun. The persons are independent of each other (but both depend on the vehicle). The third federate (we are unable to say that it is the simulator as such) takes care of the environment (maps) and relating calculations. The motion of the vehicle itself is not solved until the third federate. Only two computers are required for the run of the whole simulator (one for each user). Execution of the third federate in view of the simulation study is not important.

6.2. Selection of technologies

For the above indicated reasons, HLA was used for distribution of the solution. Since the remote control libraries and also HLA-VA framework supply a sufficient software solution for the formation of individual federates, the most important part of the development was visualization. Open GL was selected as visualization technology. It is essential to say that in the selection of technology, the application is not expected to have sophisticated graphics as that of PC games. Graphic parameters must comply with the requirements of distributed interactive simulation (it is not a distributed

interactive simulation in terms of the DIS standard, i.e. IEEE1278).
For more information about accessible technologies see for DIS IEEE1278, for HLA IEEE1516:2010, Fujimoto (2000) or SISO (2001).

6.3. Case study architecture

The case study is relatively simple: a simple system of permitted and enabled operations is placed on the HLA-VA framework for both of the federates. The federation diagram is relatively simple due to the use of a large number of libraries; it is demonstrated in Figure 4.

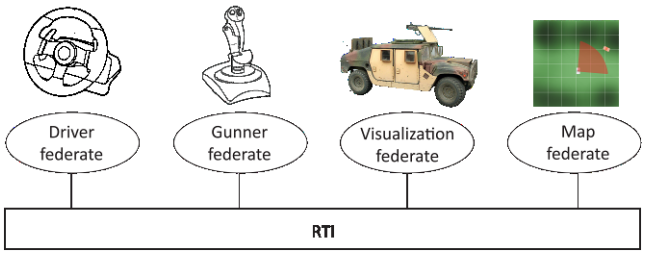

Figure 3: Real federation architecture

Few parts as Visualization federate can be run on mobile devices, see Brozek, Jakes, Gago (2014) for mor information about this principle.
There are differences in federation architecture for direct method (when hardware control devices is connected to computer), and spited method (when hardware control device is connected to simpler device without own monitor). This differences are at figure 5.

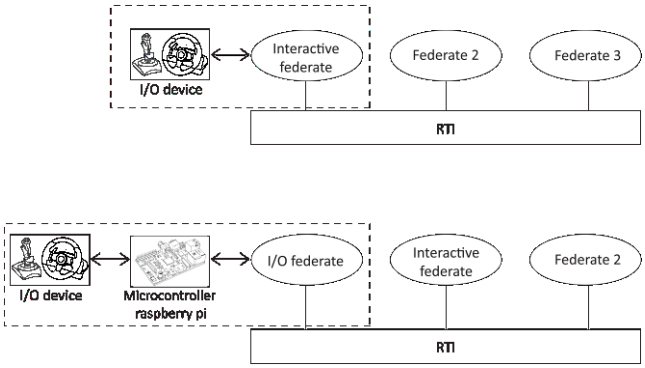

Figure 4: Two principles connection HW to HLA simulation

6.4. Testing method

Two users have been selected for testing, while each of them had the task to take up one position in the vehicle. One target was determined within the environment to be eliminated by users. All testing was carried out in relation to this target (a shooter´s target). For example, in one of the situations the driver´s task was to rotate around the target while the shooter´s task was to keep the target in the view finder of the gun (this way tested the coordination of both the federates against each other).

Since the testing used the pRTI solution by Pitch technologies, an integral part of RTI is a high quality solution for monitoring the whole running federation data. This method enables us to monitor actual synchronization as well as the values of most of the attributes. Therefore validation and verification could be performed from an independent point through the third party´s software. The running application in testing and the view of the monitoring software are displayed in Figures 6, 7 and 8.

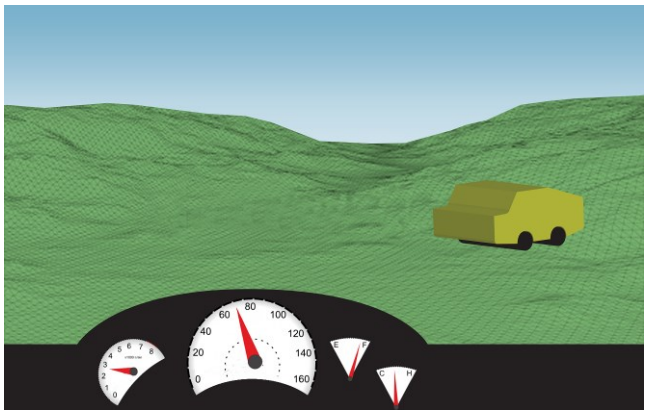

Figure 5: Drivers federate

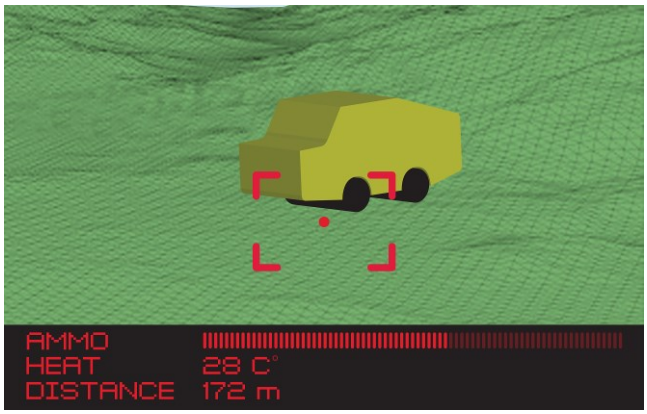

Figure 6: Gunners federate

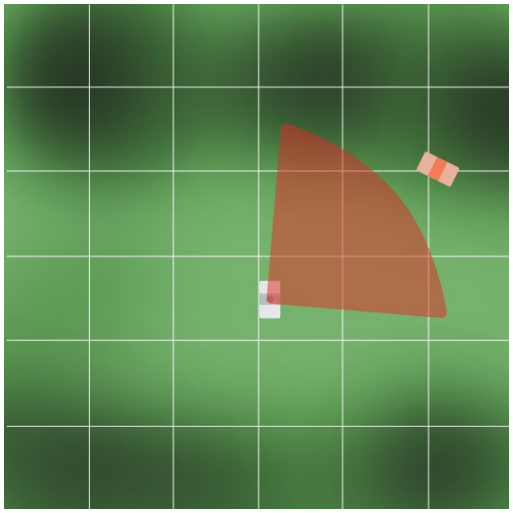

Figure 7: Visualization federate (two cars mode)

7. CONCLUSION

The article reviewed the problem of connecting input-output devices to a computer simulation and presented new solutions based on the reusability principle. Encapsulating the solution in libraries and release of this method to the simulation community results in the fact that every person within their commercial, academic or educational activity can develop simulators that - besides keyboard and mouse - can in reality use any input-output device.

Libraries as such open the way to building live simulators or distributed simulation models connected into an online simulation.

7.1. Potential for future development

The research was finalized and the designed libraries were tested. In view of this fact, the development has been completed. On the other hand, results published in this article present only one of partial tasks to be carried out to develop a high quality simulator for railway staff training. Despite the development of libraries was finished, further steps within our workplace consist in their application.

ACKNOWLEDGMENTS

Expenses related with publication of article were funded from SGS announced by University of Pardubice at 2015, in section Application section software technology.

REFERENCES

Brozek J., Jakes M., Gago L. Using tablets in distributed simulation. Pardubice, 2014. ISBN 978-889799932-4. Conference Paper. EMSS 2014 Proceedings, University of Bordeaux.

Brozek J., Onggo B.S., Kavicka A. High level architecture virtual assistant framework. Pardubice, 2014. ISBN 978-889799932-4. Conference Paper. EMSS 2014 Proceedings, University of Bordeaux.

Fujimoto R.M., c2000, Parallel and distributed simulation systems. New York; John Wiley & Sons. ISBN 04-711-8383-0.

Henninger A.E., Cutts D., Loper M. Live Virtual Constructive Architecture Roadmap (LVCAR) Final Report, Institute for Defense Analysis, Sept, 2014, Available from: http://www.msco.mil/files/MSCO%20Online%20Library/LVCAR%20-%201%20of%205%20-%20Final%20Report%20-%2020090814.pdf

Kuhl F., Dahmann J., Weatherly R., Creating Computer Simulation Systems: An Introduction to the High Level Architecture, c2000, Upper Saddle River, NJ; Prentice Hall PTR. ISBN 01-302-2511-8.

Manlig, F., 1999. Computer simulation of discrete events. Available from: http://www2.humusoft.cz/www/archived/pub/witness/9910/manlig.htm [accessed 15 July 2014].

Rabelo, L., Sala-Diakanda S., Pastrana J., Marin M., Bhide S., Joledo O., Bardina J., 2013. Simulation Modeling of Space Missions Using the High Level Architecture. Available from: http://www.hindawi.com/journals/mse/2013/967483/ [accessed 15 July 2014].

Richardson M., Wallace S.P. Getting started with Raspberry Pi. 1st ed. Sebastopol, CA: O'Reilly Media, 2012, xiii, 161 p. Make: projects. ISBN 14-493-4421-6.

Sinclair I. Sensors and transducers. 3rd ed. Oxford: Newnes, 2001, xiv, 306 s. ISBN 07-506-4932-1.

The institute of electrical and electronics engineers, Inc, 2010, IEEE1516:2010: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Framework and Rules. New York; IEEE. ISBN 978-0-7381-6251-5.

The institute of electrical and electronics engineers, Inc, 2010, IEEE1516:2010: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Object Model Template (OMT) Specifications. New York; IEEE. 2010.ISBN 978-0-7381-6249-2.

The institute of electrical and electronics engineers, Inc, 2010, IEEE1516:2010: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Federate Interface Specification. New York;IEEE. ISBN 978-0-7381-6247-8.

The institute of electrical and electronics engineers, Inc, 2010, IEEE standard for distributed interactive simulation application protocols. New York: Institute of Electrical and Electronics Engineers, 2002. ISBN 07-381-0992-4.

The simulation interoperability standards organization, Independent Throughput and Latency Benchmarking for the Evaluation of RTI Implementations, 2001, The Simulation Interoperability Standards Organization. Fall. DOI: SISO-01F-SIW-033.