CLASSIFICATION OF SYSTEMS FOR SIMULATION AND VISUALIZATION OFPHYSICAL PHENOMENA

Jiří Chludil, Jiří Žára

Department of Computer Science and Engineering,
Czech Technical University in Prague, Karlovo nám. 13, Prague 2
121 35
Czech Republic
{xchludil,zara}@fel.cvut.cz http://www.cgg.cvut.cz

ABSTRACT

The paper provides an overview and a comparison of several tools for visualization of physical simulations. We concentrate on new visualization tools a nd architectures that have been developed during last years (VRToolBox, VRML + Java, etc.). Classification of these approaches is presented together with practical examples including authors' subjective opinion. This overview has been prepared with respect to visualization in three-dimensional virtual environments.

Keywords: Physical Simulation, Visualization, Virtual Environment

1 INTRODUCTION

Simulation programs and tools often offer visualization of various parts – schemes, graphs, models, etc. When simulating physical processes and physical objects, the visualization performed in three-dimensional space (3D) helps to better understand various simulated phenomena. In many cases, the simulation runs in real time, thus the real time presentation in 3D is required, too. Here we can see the tight relation between simulation and virtual reality systems.

2 TOOLS FOR VISUALIZAT ION AND SIMULATION

Simulation is a research method [Kinde80] based on replacing a dynamic system by a simulator with the behavior equ ivalent to the original system. Experiments executed on such simulator should bring new information about the examined system to users. Several specific languages for simulation have been developed. Some of them are specialized to specific purposes, while the others are for general use. Even universal programming languages like Java or C can be used for the description of a simulation process, although this usually represents a complex and difficult task.

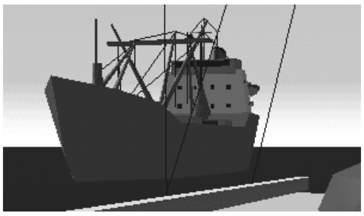
The following text contains a list of selected tools and programs. They have been chosen by commercial version availability (freeware, low prices, licences at our university).

MATLAB [MATLA] integrates mathematical computing, visualization, and a powerful language providing a flexible environment for tech nical computing. The open architecture makes it easy to use MatLab and its companion products to explore data, create algorithms and specialized custom tools.

SIMULINK [SIMUL] is an interactive tool for modeling, simulating and analyzing dynamic systems. It enables to build graphical block diagrams, evaluate system performance and refine the design. SIMULINK has been developed simultaneously with MATLAB.

The Virtual Reality Toolbox [ToolB] extends the capabilities of MATLAB and SIMULINK into the virtual reality. Utilizing standard VRML technology, it represents an open solution for rendering animated 3D scenes driven from the MATLAB/SIMULINK environment. Results of the simulation can be observed in virtual reality. The Virtual Reality Toolbox interconnec ts MATLAB and SIMULINK with arbitrary browser conformed to the ISO VRML specification [VRML].

The Virtual Reality Modeling Language (VRML) is an ISO standard [VRML] for the description of 3D interactive scenes. A platform-independent textual format allows not only define 3D objects, but also dynamically change their properties using event sending and processing. The VRML is widely accepted for presentation, visualization and simulation purposes. The VRML browsers are available on many computing platforms.

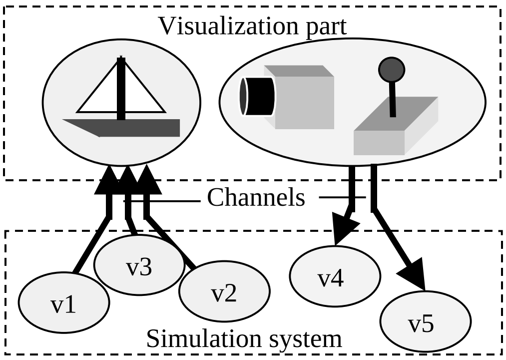


VRML scene example
Figure 1

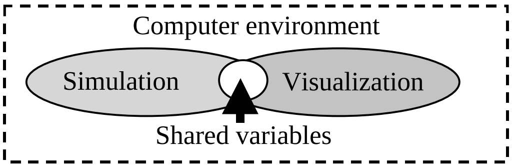
A standardized programming interface called EAI [Exter] serves for communication between a VRML scene and other programs. The virtual scene can be controlled via external programs or applets in the case of web browser hosting the VRML browser. A typical web application consists of VRML browser window and additional controls in Java applet.

3 INTERFACE BETWEEN SIMULATION AND VISUALIZATION PARTS

The visualization part usually contains virtual models that are controlled by values provided by the simulation part. These values are sent through data channels (see Fig. 2). Data sent from simulation part represent computed values (variables v1, v2, and v3 in Fig. 2), while data received from the visualization part typically represent a feedback from sensors (variables v4 and v5 in Fig. 2).



Connection between simulation and visualization engines
Figure 2



Channel implemented using shared variables
Figure 3

The channel can be implemented in various ways, e.g. by means of shared variables (see Fig. 3), or a network (see Fig. 4). We propose the following parameters that characterize the type of interconnection between both system parts:

Number of visualization channels (VC) – represents quantity of variables that are visualized and that control the visualization process. It is supposed that one channel exists for each variable.

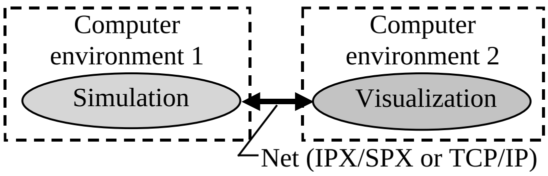
Number of sensor channels (SC) – represents quantity of variables received by simulation part. These variables are generated by sensors or control elements in the visualization part.

Number of samples per second (NS) – represents number of values that are sent/received by the simulation system to/from visualization part.

Synchronous vs. Asynchronous communication (SY/AS). Synchronous method of communication is based on sending values in regular time instances; if the channel uses network protocol, it is more efficient to cumulate the data into one packet. When working asynchronously, a simulation system sends values only when they are changed. Then the parameter **NS** is determined as the average value.

Signal delay (SD) – a time period between sending and receiving specific value.

Visualization precision (VP). If the visualization process is highly computational demanding, the whole system may not be able to present data changes in real time. Two solutions are at hand: either the entire simulation is slowed down (*Full visualization*) or the simulation still runs in real-time, but some changes are skipped (*Fragmentary visualization*).



Channel implemented using network protocol
Figure 4

4 LANGUAGES FOR SIMULATION AND VISUALIZATION

We suggest to divide programming languages into the following three categories:

Graphic languages (GRA) like OpenGL or VRML are used for data visualization very often. Such languages contain special graphic functions (NURBS, etc.). They also utilize functions for user interaction with user interface (sensors, control elements, etc.). This type of languages usually requires support from general-purpose languages.

Simulation languages (SIM) like SIMULINK, Simula, or MATLAB are used for simulation of dynamic systems. These languages include special simulation functions (process management, event processing, mathematical, memory management instructions, etc.). General-purpose languages usually implement this type of languages.

General-purpose languages (GEN) like C, C++, Java, or Pascal are used for general programming. These languages usually support simple graphics elements (line, rectangle, etc.). Language structure can support multi-process environments (Java).

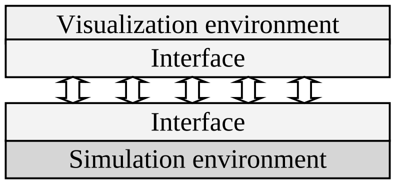
This classification is quite general and a little simplified, but adequate for our purposes; this will become clear in the next section.

5 ARCHITECTURES FOR VISUALIZATION OF SIMULATION

This section describes various visualization architectures. The first name in the following titles determines language that describes the simulation. The second name determines language used for the visualization. It should be stressed that pure simulation languages are not used for the visualization itself, thus the abbreviation SIM never appears as the name in right part of the following titles. Similarly, graphics languages are not suitable for the implementation of simulation engine (except VRML). That is why the combination GRA – GEN is also not used in the following classification.

Each of the following paragraphs contains a short description, main advantages and disadvantages and finally an example of a typical software tool that uses specified architecture.

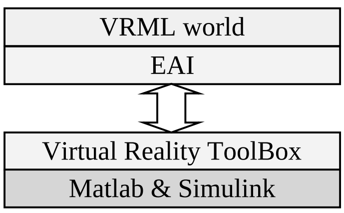
SIM – GEN: Since the simulation language has to communicate with the visualization part, a sort of interface functions should be an integral part of the simulation language (see Fig. 5). Such an interface should be general enough to support general programming languages. The data interchange can



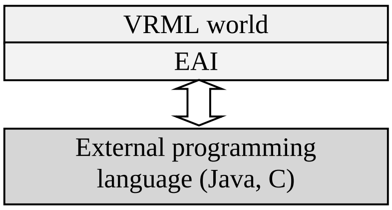
General visualization and simulation architecture
Figure 5

be based both on shared variables and network protocols like TCP/IP or IPX/IP. The main advantages are: Firstly, good support for simulation. Secondly, the simulation and visualization processes can be performed on different computers; in that case they do not influence their loads. There are also some disadvantages: An interface and data exchange between both parts is required. Another disadvantage is that when using two computers for both parts, communication delay can occur. Finally, the graphics output is mostly limited to the use of basic primitives like lines, rectangles and other simple graphic elements. Advanced 3D graphics with lighting and texturing is very rare. Typical example of this architecture is MATLAB.

SIM – GRA: This combination seems to be optimal. Both simulation and visualization part are implemented using languages that are specific to the target subsystem. We see two main advantages: The first is a good support for simulation and visualization. Secondly, the simulation and visualization processes can be performed on different computers. We can also find some disadvantages: Interface and data exchange between both parts (both languages have to implement a communication interface). Finally, possible communication delay when using two computers running in parallel. Typical example of this architecture is a combination of MATLAB, VR ToolBox and VRML (see Fig. 6). The simulation language here is SIMULINK, the layer between simulation and visualization is implemented using VR ToolBox and EAI for VRML. Similar approach is used in applications like WorldToolKit.



Java with EAI and MATLAB with VRToolBox
Figure 6



Java with EAI and external programming language
Figure 7

GEN – GEN: The use of general languages for both parts is a compromise between simulation and visualization requirements. This architecture has one main advantage: since both parts are implemented using the same programming language, the communication and data exchange is straightforward without any additional overhead. There are also some disadvantages: poor support for simulation (the mathematical possibilities and process management are very limited). General languages require experienced programmers, because implementation usually utilizes special methods and data structures. We can find such approach in special simulation applications such as [Linds].

GEN – GRA : This architecture is suitable for the higher requirements on visualization. Advanced 3D graphics with complex geometrical shapes, lighting and texturing is available. Sound and even other media can enrich the visual presentation. This architecture has three main advantages. Firstly, good support for visualization. Secondly, if both parts are implemented using the same programming language, the communication and data exchange is straightforward without any additional overhead (C and OpenGL). And finally, the simulation and visualization processes can be performed on different computers. The disadvantages are: poor support for simulation (the mathematical possibilities and process management are limited); if both parts are implemented with different programming languages, the same disadvantages apply as in SIM – GRA paragraph. Typical example is our project Nautilus [Chlud01]. It is an experimental system for teaching and testing yacht captains. The simulation and virtual environment has been built using VRML and Java. The implemented training system utilizes web environment where the virtual sea and various kinds of ships are presented in VRML window, while the movement of ships and their behavior are controlled by a Java applet.

GRA – GRA: The implementation of simulation using graphic languages is very unusual but it is possible in simple cases. It is true mainly for languages describing virtual reality like VRML. This architecture has two main advantages. Firstly, good support for visualization. Secondly, VRML has event system with time-stamps supporting simulation. Main disadvantages are: it is suitable mainly for simple problems; support for mathematical functions is poor. Typical examples of this architecture are various VRML worlds [Paral].

6 CONCLUSION

Parameters characterizing the type of interconnection between visualization and simulation system parts have been described in this paper. These parameters are useful for comparing various system architectures, their capabilities, extensions etc.

Selected visualization architectures have been presented. Comparing advantages and disadvantages of the particular architectures one can choose the architecture that is suitable for the specific purpose, i.e. that gives the best results and provides optimum performance.

7 ACKNOWLEDGMENT

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under the grant FRVS/1891/2001 (Education in Virtual Environments).

REFERENCES

[MATLA] MATLAB, *The MathWorks* http://www.mathworks.com/products/matlab/index.shtml

[SIMUL] SIMULINK, *The MathWorks* http://www.mathworks.com/products/simulink/index.shtml

[ToolB] Virtual Reality ToolBox, *Humusoft* http://www.humusoft.cz/vr/index.htm

[VRML] The Virtual Reality Modeling Language. International Standard ISO/IEC 14772-1:1997. http://www.web3d.org/technicalinfo/specifications/vrml97/

[Exter] The VRML External Authoring Interface. *Draft International Standard* ISO/IEC 14772-2, http://www.vrml.org/WorkingGroups/vrml-eai/Specification/

[Kindle80] Evžen Kindler: Simulační programovací jazyky (in Czech), *SNTL*, 1980.

[Chlud01] Chludil, Žára: Yacht Captain Training System in VRML, CATE - IDET, Brno, Czech Republic, 2001.

[Linds] C. S. Lindsey: Physics simulation with Java, http://www.particle.kth.se/~fmi/kurs/PhysicsSimulation

[Paral] Parallel Graphics, VRML worlds selection, http://www.parallelgraphics.com/products/cortona/best