# Using Simulation and Virtual Reality for Distance Education

Juan de Lara, Manuel Alfonseca
Dept. Ingeniería Informática, Universidad Autónoma de Madrid
Campus de Cantoblanco, 28049 Madrid
*E-Mail: {Juan.Lara, Manuel.Alfonseca}@ii.uam.es*

Keywords: Continuous Simulation, Web-based simulation, Virtual Reality, Distance Education.

Abstract: This paper describes the construction of virtual reality simulations for distance education trough Internet. This is accomplished by means of an object oriented continuous simulation language, called *OOCSMP*, and a Java generating compiler for this language called *C-OOL*. This compiler is also able to create VRML worlds. The behaviour of the VRML world is specified in the *OOCSMP* models. Change of simulation parameters is possible at run time by means of a Java interface, generated by the compiler. An example of the simulation of the inner Solar System is presented.

## Introduction

The growing popularity of Internet, and the increasing number of computers connected to it, makes it an ideal framework for distant education. Not only educational sciences, but also a large number of disciplines are re-thinking their traditional philosophies and techniques to adapt to the new technologies [1]. One of these disciplines is computer simulation.

Virtual Reality (VR) techniques offer immersive environments in which the user has great possibilities of interaction. The emergence of the Virtual Reality Modelling Language VRML [2] and browser plug-

ins for this language [3] has made it possible to build virtual worlds accessible through Internet. The VRML97 [4] standard defines an external API that allows us to control VRML objects from a Java applet.

Its advantages, such as new possibilities of interaction and more realistic and pleasant learning environments, are turning VR into a valuable tool in distance education [5,6].

In this paper we present some improvements to the tools that we use to automatically generate web courses based on simulation [7]. These enhancements include the possibility of incorporating VR panels to the simulation applets. An example with the simulation of the Inner Solar System is presented.

## Our Simulation Tools

The *OOCSMP* continuous simulation language was conceived in 1997 [8] as an object oriented language. The language is specially suited when the system can be modelled with similar interacting objects. A compiler (*C-OOL*) was built for this language in order to produce C++ code or Java applets with the simulation models. This approach simplifies the generation of simulation based web courses. Several web courses have been generated using this language, on gravitation, partial differential equations, ecology and basic electronics, which can be accessed from:

*http://www.ii.uam.es/~jlara/investigacion*

The language and the compiler have been designed with an educational focus, for example:

- It is possible to include several forms of output displays in the same simulation.
- Multimedia elements can be synchronized with the simulation execution.
- The user interface allows changing parameters, object attributes or even adding or deleting objects during the simulation execution.
- Alternative simulations can be designed and can be accessed from the main simulation. In this way, the teacher can identify

interesting situations that arise when changing some parameter, or when adding some object, etc.

- There are also instructions that allow us to describe the appearance of the web page where the simulation models are going to be placed.

**Extending the language to handle VR**

In *OOCSMP*, there is the possibility of associating an icon to each object taking part in the simulation. This icon can be used in all the output forms. The VR extensions allow us to assign also a *VRML* node to each object, by means of the new attribute type ***VRMLobject***.

A new instruction (***VRMLworld***), to include *OOCSMP* objects (considered as dynamic components of the Virtual World) in a virtual world, has also been added. This instruction also configures the virtual world with static elements.

For example, suppose we want to simulate the behaviour of the inner solar system [9]. To model it, we will encapsulate the behaviour of a planet in a class, and declare an object associated with each planet of the solar system. The class has an attribute to assign to the planet a VRML object. A scheme of the *OOCSMP Planet* class is shown in listing 1.

```
[1]CLASS Planet {       * Definition of Planet class
[2]  NAME        name   * Name of the planet
[3]  VRMLobject  obj    * VRML object associated with the Planet
[4]  DATA M, X0, Y0, XP0, YP0, FI
[5] INITIAL     * Compute initial data (FIR, CFI and SFI)
[6]  FIR:=FI*PI/180
[7]  CFI:=COS(FIR)
[8]  SFI:=SIN(FIR)
[9] DYNAMIC  * Distance to the Sun
[10] R2   := X*X+Y*Y
[11] R    := SQRT(R2)
[12] ...
[13]ACTION Planet P   * Mutual actions of two planets
[14] DPP2 := (P.X-X)*(P.X-X)+(P.Y-Y)*(P.Y-Y)+(P.Z-Z)*(P.Z-Z)
[15] DPP  := SQRT(DPP2)
[16] ...
[17]FINISH R<.05
[18]VRMLworld X, Y, Z}
```

**Listing 1:** Outline of an *OOCSMP* class representing a planet.

The previous *OOCSMP* class defines some blocks (***DYNAMIC*** and ***ACTION***) to simulate the behaviour of a particular planet, whose full listing can be found at [7]. It also declares a termination condition (***FINISH***), and inserts all the objects of the *Planet* class in the virtual world (***VRMLworld***), where they will be the dynamic elements. The three parameters (*X*, *Y*, *Z*) control the position of the object inside the virtual world.

Listing 2 shows the remainder of the *OOCSMP* model, which uses the *Planet* class to generate a model of the inner Solar System.

```
[1] * Universal data, and Sun data
[2]  DATA G:=0.00011869, PI:=3.141592653589793, MS:=332999
[3]  INCLUDE "Planet.csm"
[4]  Planet Mercury("Mercu", “mercury.wrl”, 0.055271,      ...)
[5]  Planet Venus  ("Venus", “venus.wrl”, 0.81476,  0.7233,...)
[6]  Planet Earth  ("Earth", ”earth.wrl”, 1, 0,      1,    ...)
[7]  Planet Moon   ("Moon", “moon.wrl”, 0.01235,  0,       ...)
[8]  Planet Mars   ("Mars", “mars.wrl”, 0.10734,  1.5233,  ...)
[9]  Planet Apollo ("Apolo", “Apollo.wrl”, 1957E-14,       ...)
[10] Planet Jupiter("Jupit", “jupiter.wrl”, 317.94,       ...)
[11] Planet InnerSystem := Mercury, Venus, Earth, Moon, Mars,
[12]                       Apollo, Jupiter
[13] DYNAMIC
[14]   InnerSystem.STEP()
[15]   InnerSystem.ACTION(InnerSystem)
[16] TIMER delta:=.0005, FINTIM:=2, PRdelta:=.1, PLdelta:=.01
[17] VRMLworld “sun.wrl”
[18] METHOD ADAMS
```

**Listing 2:** The *OOCSMP* model for the inner solar system.

Line 17 adds the sun (a static element) to the VRML world. When declaring each planet, we have to specify the VRML node associated with it. In our case, each VRML node will be a sphere with an hyperlink. When the user clicks in the hyperlink, useful data of the planet will be presented in an HTML frame. For example, the following listing shows the *wrl* file associated to planet Mars.

```
#VRML V2.0 utf8

Anchor {
    url "mars.html" parameter [ "target=Details" ]
    children [
        Shape {
            appearance  Appearance {
                material  Material{
```

```
                    diffuseColor 1 0 0
                    shininess .5
                }
            }
            geometry Sphere { radius 0.25 }
        }
    ]
}
```

**Listing 3:** VRML node associated with planet Mars.

## Generating a VR simulation

When compiling the model with C-OOL, several files are generated:

- A *wrl* file, containing all the VRML objects declared in the model.
- A Java applet, with the simulation logic, the simulation controls, and the other graphical output forms, if any has been selected in the model. The Java applet allows the user to start and stop the simulation, to change the model parameters, etc., and communicates with the Virtual World by means of the External Authoring Interface (EAI) [10]. Using proprietary technologies such as Netscape's LiveConnect [11] is widely extended [2, 5], but we think our solution is better, because additional libraries are not needed, we are compatible with Internet Explorer, and the communication between the Java applet and the Virtual World is done directly from the program, rather than indirectly trough JavaScript.

Figure one shows the working scheme of all these components. There is a graphical and numerical Java library placed in the Server (for developing pourposes, it can be placed in the client), used by the generated Java programs. When the user accesses the simulation page, the VRML code, the Java applets, and the necessary Java classes are first downloaded from the server, but are then executed locally

The necessary elements to run the simulations are thus:

- A graphic browser.
- A VRML97-compliant plug-in, such as CosmoPlayer.
- A Java Virtual Machine plug-in, understanding at least Java 1.1 code.

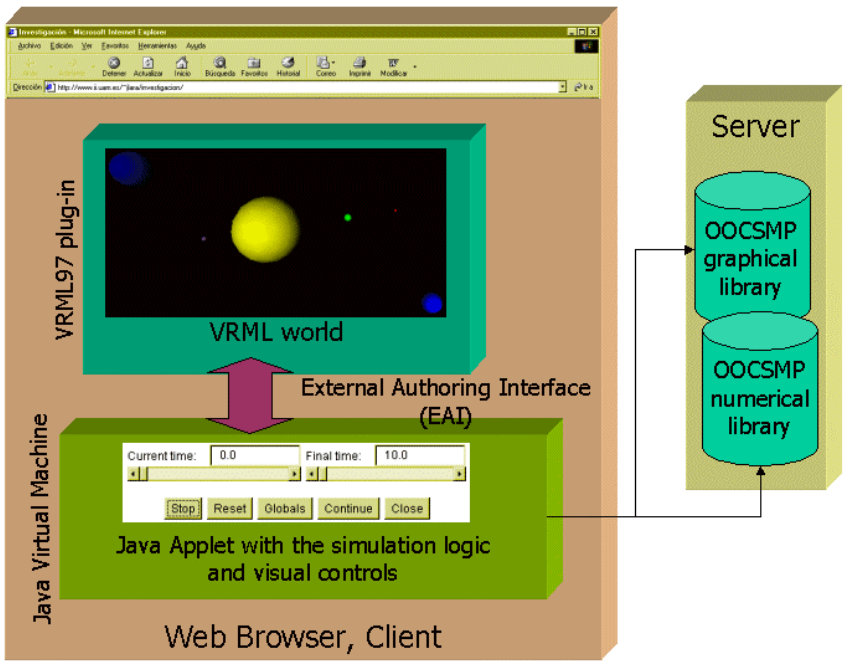


**Figure 1:** Working scheme of a VR simulation page.

The next figure shows a moment in the execution of the simulation.

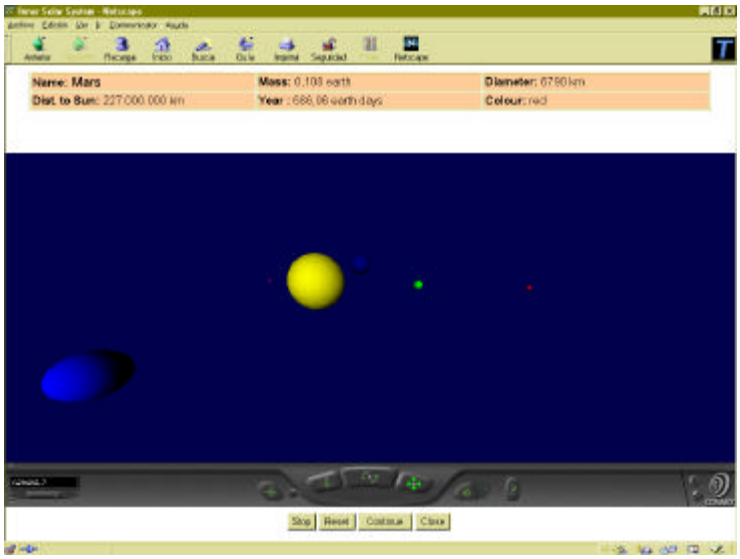


**Figure 2:** A moment in the simulation of the inner solar system.

In the previous figure, the user has clicked on the planet Mars, and the associated data are being shown in the upper frame. The sizes of the planet shapes are not proportional to their real sizes, to make them visible in the image scale. Figure 3 shows a similar simulation, where a two dimensional plot has been selected as a graphical output form. The animation of both panels is synchronized, because the simulation engine sends messages to CosmoPlayer and to the 2D Java panel every time step.

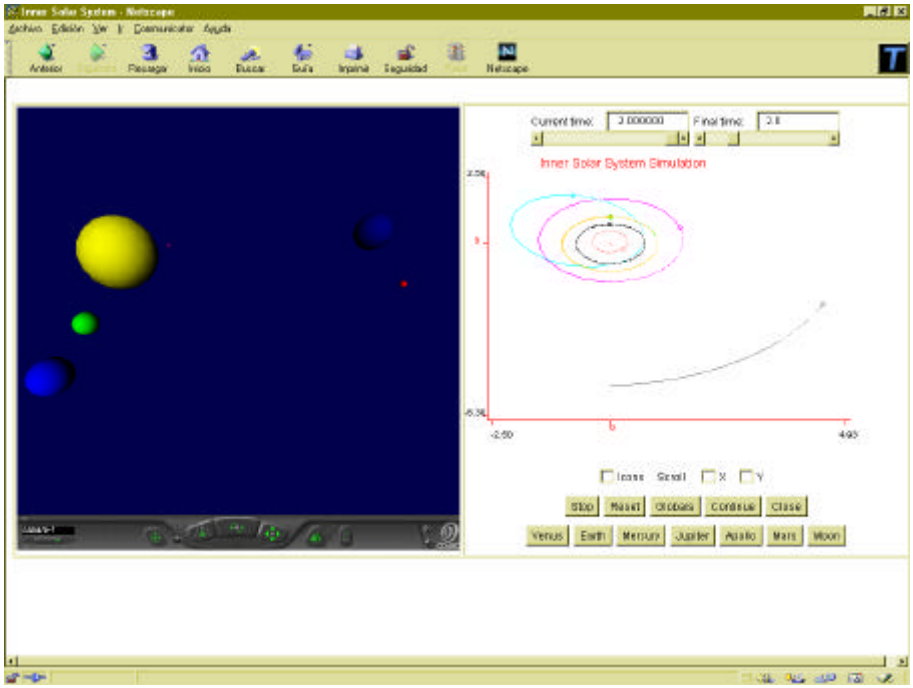


**Figure 3:** Simulation using Virtual Reality and 2D plots.

There is a problem in the integration of VRML panels in the user interface generated by our compiler, because these panels are not placed inside the Java applet, but in the HTML page in an *<EMBED>* HTML tag. This complicates the construction of the user interface, that we have solved using tables. In figure 3, the VRML world and the applet have both been placed inside an HTML table.

## Conclusions and future work

In the present article, we have presented some extensions to facilitate the construction of Virtual Reality simulations trough Internet. The simulation behaviour is modelled by means of the *OOCSMP* language, that allows us to assign VRML nodes to the *OOCSMP* objects. The compiler takes charge of generating the VR world and the necessary Java files. The programmer only has to model the system behaviour, in a high-level simulation language, and specify the appearance of each object by means of simple VRML nodes. The inclusion of these simulations in educational courses is thus straightforward. In the generated Java code, the inclusion of a VRML panel only represents a few lines of code: some of them in the inizialization, to get a handle to the VRML browser, the others in the main simulation loop (in the function that updates all the graphical output forms), to actualize the position of the corresponding VRML nodes.

VRML also offers other possibilities of interaction, that we have exploited in the example, such as setting hyperlinks in the simulation objects, which can be used to explain the role of the object in the simulation or, as in our case, to show additional data.

The example can be accessed from:

*http://www.ii.uam.es/~jlara/investigacion/ecomm/solar3.html*

The primitives that add Virtual Reality capabilities to our system only deal with the displacement of objects. Other primitives could also be added to handle other object properties, such as rotation, size, colour, etc.

We are currently working in an authoring tool for the construction of simulated-based web documents. These documents can be educational web courses, but also electronic articles - with simulation and interactive elements - or interactive presentations. We are planning to incorporate the possibility of designing VRML worlds to this tool, or at least allow an easy interaction with some VRML development tool.

**Acknowledgment**

This paper has been sponsored by the Spanish Interdepartmental Commission of Science and Technology (CICYT), project number TEL1999-0181.

**References**

[1] Page E.H. Buss, A., Fishwick, P.A., Healy, K., Nance, R.E., Paul, R.J. 2000. *Web-Based Simulation: Revolution or Evolution?*, to appear in ACM Transactions on Modeling and Computer Simulation.

[2] Hartman, J., Wernecke, J. 1996. *The VRML 2.0 Handbook. Building Moving Worlds on the Web* . Addison-Wesley.

[3] Cosmo Player, at: http://www.cosmosoftware.com/products/player

[4] VRML97 Specification: www.web3d.or/Specifications/VRML97

[5] Schmid, Ch. 1999. *A Remote Laboratory Using Virtual Reality on the Web* . Simulation. Special Issue: Web-Based Simulation. Vol.73:1, 13-21.

[6] Virtual Reality for Education Resources on the Net: http://www.hitl.washington.edu/projects/knowledge_base/education.html

[7] Alfonseca, M., de Lara, J., Pulido, E., 1999. "*Semiautomatic Generation of Web Courses by Means of an Object-Oriented Simulation Language* , Simulation. Special Issue: Web-Based Simulation, Vol 73:1, 5-12.

[8] Alfonseca, M., Pulido, E., Orosco, R., de Lara, J. 1997. *"OOCSMP: an object-oriented simulation language"*. ESS'97, Passau, pp. 44-48.

[9] Alfonseca,M., de Lara, J., Pulido, E. 1998. *"Semiautomatic Generation of Educational Courses in the Internet by Means of an Object-Oriented Continuous Simulation Language"*. ESM'98, Manchester, pp. 547-551.

[10] Introduction to the VRML and Java EAI: http://www.dcs.gls.uk/people/personal/snowdonp/vrml/intr.html

[11] LiveConnect, at Netscape ftp site: ftp.netscape.com in pub/sdk/plugin/windows/oct_21_97