Tools for DEVS Modeling, Simulation and 3D Visualization

[Poster Abstract]

Patrick Castonguay **Tania Pendergast** **Gabriel Wainer**

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, ON K1S 5B6
{pcaston3, tpenderg}@connect.carleton.ca; gwainer@sce.carleton.ca

Descriptors
I.6.8 [**Types of Simulation**]: *Discrete event, Visual.*

Keywords
Cell-DEVS, CD++, Blender, Visualization

ABSTRACT

Cell-DEVS is an extension of the DEVS formalism which combines DEVS with Cellular Automata. It is particularly useful for defining spaces by decomposing them into individual cells. The CD++ Toolkit enables one to model and simulate a real or artificial system using either DEVS or Cell-DEVS. Although it is provided with a Modeler tool, this permits only 2D visualization of the simulation. This paper focuses upon two efforts to produce integrate DEVS simulations developed with the CD++ tool with 3D visualizations using Blender, an open-source 3D visualization software.

1. INTRODUCTION

Cell-DEVS [1,2] provides a mechanism for modeling and simulation of cell spaces that has been applied to multiple projects in different fields. The CD++ Toolkit provides DEVSView as well as the CD++ Modeler add-on [3]. These tools provide a quick representation of the model and simulation behaviour in the 2D space. Nevertheless, in order to represent the simulation results more accurately, while providing mechanism for interaction with users, 3D visualization of CD++ results has previously been done using VRML, Atlas, Maya and Blender [4].

Blender [5] is a free open source 3D content creation suite. It can be used to create 2D or 3D graphics as well as movie quality animations. Blender supports Python scripting, is free to use, open-sourced, and provides powerful 3D animation capability. Python provides a high-level dynamic object-oriented programming language. For these projects no modifications were made to Blender itself but rather its native support for Python scripting was used in order to read and animate the results from CD++. On [6] and [4], we showed how to integrate basic DEVS simulation models written in CD++ with the Blender suite, showing the integration and interoperability of both CD++ and Blender.

SIMUTOOLS '09, Rome, Italy

In this paper we will show the application of these methods and the use of the tool chain to an aircraft emergency evacuation simulation. Then we focus on implementing a minefield mapping simulation using semi-autonomous robots. These applications will show how to provide advanced visualization models using this tools, providing insight on how to create advanced 3D visualization projects based on basic simulation results.

2. MODEL DEFINITION

We first introduced a model used for aircraft emergency evacuation simulation based on the work presented in [8], which is concentrated on analyzing the effect of passenger delays at the exit doors of a very large aircraft. A dual-layered cellular grid representing the floor plan was defined, based on [6]. On the first layer, a single cell can represent a wall, a passenger, an exit or an empty space. A second layer was used to store information on distances to the nearest functioning emergency exit. Figure 1 shows a snapshot of the 2D visual representation of both layers during the execution run, as presented by the CD++Modeler toolkit. The variable under study was the hesitation at the door, thus, no other factors like panic behaviour or secondary routing seeking behavior were implemented.

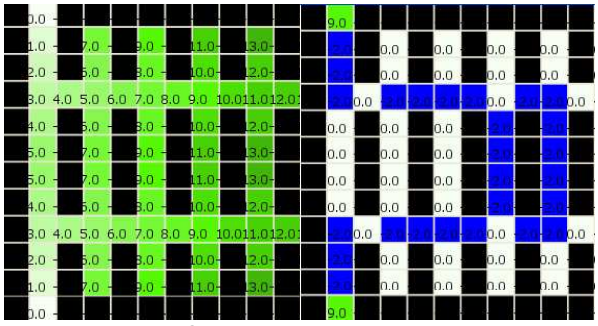

Figure 1: Aircraft Evacuation Simulation

Our second model aims to model the behaviour of one or many robots mapping a minefield. It was designed such that

robots moved about the minefield randomly and avoided occupying the same cell. The individual robots are constrained to move within the minefield and scan the ground in each cell in order to detect the presence or absence of mines. It was assumed that the robots are in contact and update a common map of the minefield. Each cell represent a realistic area of ground that a robot could be expected to scan for a mine. The minefield cell space had two layers. The first layer contained the actual disposition of the mines. The state values for this layer do not change during the simulation. The second layer contains the map of the minefield that was generated by the robots moving about the minefield. The state values for this layer changed as the robots moved about the minefield and scanned the ground for mines. Figure 2 depicts a snapshot of these two layers during the simulation execution.

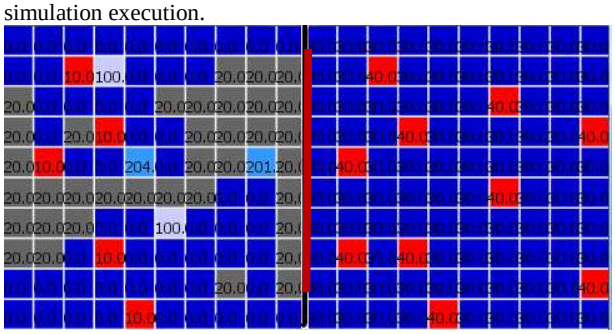

Figure 2: Minefield Mapping simulation

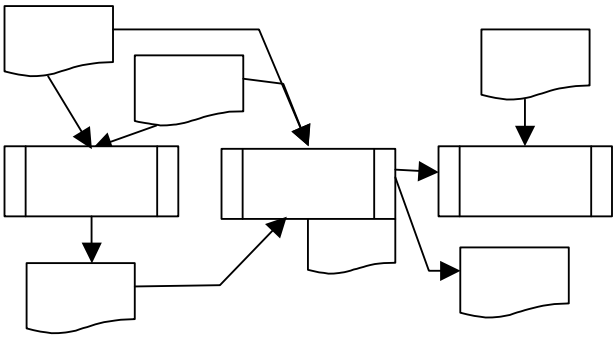

Figure 3: Visualization Architecture

3. VISUALIZATION

The CD++ Toolkit uses a model (.MA) file to define the behavior of a model. A .VAL file can be used to set the initial values of each cell of the model in the simulation. These two files are used at run-time to define the behaviour of the simulation and a .LOG file is generated which contains the state of each cell for each time-advance of the simulation. In order to increase the script's flexibility, it was extended to both recognize the existence of a .VAL file and to carry out the logging of the script's behaviour. The cell space is represented in Blender by a direct coordinate transformation of the cell position to coordinates within Blender. Once loaded, the script reads and parses the .MA file and looks for a .VAL file if needed. The .VAL file is processed before the .LOG file in order to adequately initialize the simulation behaviour. See Figure 3 for a visual representation of the CD++ Python-Blender script architecture. As the .LOG file is parsed, objects will be displayed at the appropriate coordinates within Blender. Objects will appear and disappear in sequence, thereby representing their movement.

4. CONCLUSION

We learned how one can adapt a generic script to support the visualization of results from a specific DEVS simulation. The actual 3D representations of our simulation results can be found in Figures 4a and 4b. Future work will focus on extending the script to enable automatic mapping of 3D model behaviours. As well, the automatic generation of a rendered animation will be explored, rather than having to view the raw animation within Blender. The inclusion of the Blender visual engine directly in the CD++ Toolkit is currently being investigated.

Figure 4a: Aircraft Evacuation 3D Visualization

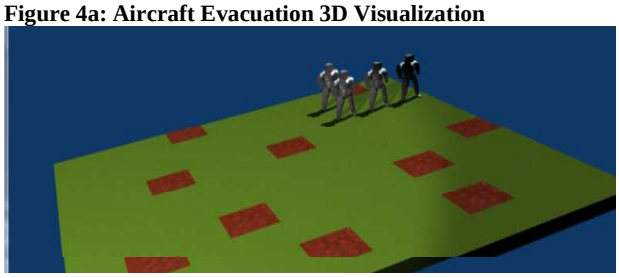

Figure 4b: Minefield Mapping 3D Visualization

5. REFERENCES

[1] Wainer, G. and Giambiasi, N. "Application of the Cell-DEVS Paradigm for Cell Spaces Modeling and Simulation". Simulation, vol. 71, No. 1, pp. 22-39, January 2001.

[2] Wainer, G. "Discrete-Event Modeling and Simulation: a Practitioner's approach". Taylor and Francis. 2009.

[3] Wainer, G. "CD++: a toolkit to define discrete-event models". Software, Practice and Experience. 32(3), 1261-1306. November 2002.

[4] Wainer, G. and Liu, Q. "Tools for Graphical Specification and Visualization of DEVS Models". Accepted for publication in *Simulation, Transactions of the SCS*. 2009.

[5] Blender Foundation, http://www.blender.org/

[6] Wainer, G., Poliakov, E., Hayes, J. and Jemtrud, M. "A Busy Day at the SAT Building". Proceedings of the International Modeling and Simulation Multiconference, Buenos Aires. 2007.

[8] Amos, M. & Woods, A. "Effect of Door Delay on Aircraft Evacuation time". http://arxiv.org/abs/cs/0509050. 2005.