A Busy day at the SAT building

Gabriel Wainer Emil Poliakov James Hayes Michael Jemtrud

Abstract— **Cell-DEVS is an extension to the DEVS formalism that allows the definition of cellular models. CD++ is a modeling and simulation tool that implements DEVS and Cell-DEVS formalisms. It was successfully employed to define a variety of models for complex applications using a cell-based approach. In order to improve model validation and analysis, we introduced a 3D visualization engine, which is based on the Maya 3D visualization tool and its scripting language. The application allows virtual worlds to be developed using the Maya visualization environment, and permits interaction with DEVS models built in CD++. The result is an enhanced simulation environment, which permits improved experimentation. We discuss how to apply this environment to evacuation processes.**

***Index Terms*—evacuation, cell-devs, 3D visualization, Maya**

I. Introduction

In recent years, many simulation models of real systems have been represented as cell spaces. Cellular Automata [1] is a well-known formalism to describe these systems, defined as infinite n-dimensional lattices of cells whose values are updated according to a local rule. Cell-DEVS [2] was defined as a combination of cellular automata and DEVS (Discrete Events Systems specifications) [3]. The goal is to improve execution speed building discrete-event cell spaces, and to improve their definition by making the timing specification more expressive. Various efforts have focused on the cell-space simulation of evacuation processes ([4],[5],[6],[7],[9], [9]). Here, we discuss a recent model which represents people moving through a room or group of rooms, trying to get out through an exit door. The goal is to understand where the bottlenecks can occur, and which solutions are effective to prevent congestion. The basic idea was to simulate the behavior and movement of every single person involved in the evacuation process. We also discuss the use of these techniques in architecture and construction. To do so, we have integrated our Cell-DEVS model with a 3D visualization environment used for architectural reconstruction, using Autodesk Maya [10]. We will show how advanced evacuation can be modeled visualized using this environment. The basic idea was to simulate the behavior and movement of every single person involved in the evacuation process. A Cell-DEVS model was chosen with a minimum set of rules to characterize a person's behavior:

- A normal person goes to the closest exit.
- A person in panic goes in opposite direction to the exit.
- People move at different speeds.
- If the way is blocked, people can decide to move away and look for another way.

We also defined an advanced visualization model of evacuation. These models that could predict and present the results of human beings evacuating structures, such as buildings, ships and houses etc, during an emergency. We will show how the results of our visualization environment facilitate and ease the interpretation of the simulation results.

II. Background

DEVS is a systems theoretical approach that allows the definition of hierarchical modular models [3]. A real system modeled using DEVS can be described as a set of atomic or coupled submodels. The atomic model is the lowest level and defines dynamics, while the coupled are structural models composed of one or more atomic and/or coupled models. Coupled models are defined as a set of basic components (atomic or coupled), which are interconnected through the models' interfaces. The models' coupling defines how to convert the outputs of a model into inputs for the others, and how to handle inputs/outputs from/to external models

Cell-DEVS has extended the DEVS formalism, allowing the implementation of cellular models with timing delays [2]. Once the behavior of a cell is defined, a coupled Cell-DEVS can be created by putting together a number of cells interconnected with their neighbors. Each cell is defined as a DEVS atomic model, and it can be later integrated to a coupled model representing the cell space. Each cell uses N inputs to compute its next state. These inputs, which are received through the model's interface, activate a local computing function (**t**). A delay (**d**) can be associated with each cell.

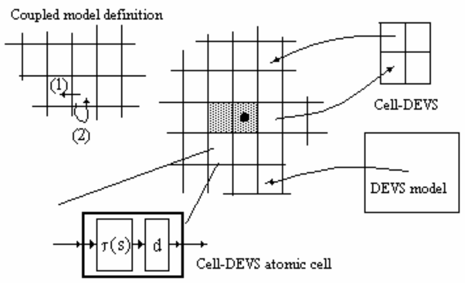


Fig. 1. Informal definition of Cell-DEVS

CD++ [11] is a modeling and simulation toolkit that implements DEVS and Cell-DEVS theory. Cell-DEVS models are defined using a built –in specification language.

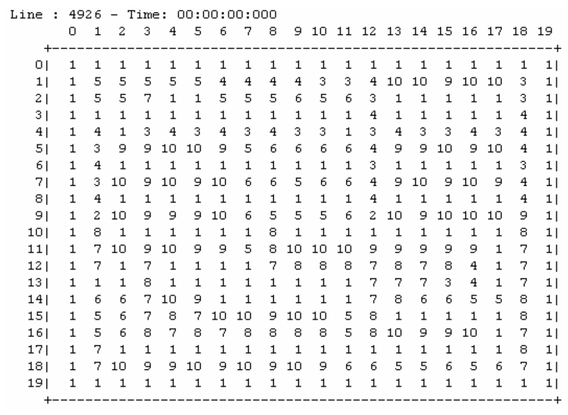
Our model represents people moving through a room or group of rooms, trying to gather their belongings or related persons and to get out through an exit door. The goal is to un-

G. Wainer and E. Poliakov are with the Dept. of Systems and Computer Engineering., Carleton University (email: gwainer@sce.carleton.ca)

E. Poliakov, J. Hayes and M. Jemtrud are with Carleton Immersive Media Studio, Carleton University.

Address: 1125 Colonel By Dr. Ottawa, ON. K1S 5B6. Canada.

derstand where the bottlenecks can occur, and which solutions are effective to prevent congestion. We defined a model to simulate the behavior and movement of every single person involved in the evacuation process. The Cell-DEVS model characterizes a person's behavior. People move at different speeds; if the way is blocked, people can decide to move away and look for another way [12], [13]. In Fig. 2 the state value "1" represents walls or obstacles, and the state value "2" represents exits. The even state values are occupied cells and the odd ones are empty cells. Each state value also represents the shortest direction to the exit.



```
Line : 4926 - Time: 00:00:00:000
      0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
   +-----------------------------------------------------------+
  0|  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1|
  1|  1  5  5  5  5  5  4  4  4  4  3  3  4 10 10  9 10 10  3  1|
  2|  1  5  5  7  1  1  5  5  5  6  5  6  3  1  1  1  1  1  3  1|
  3|  1  1  1  1  1  1  1  1  1  1  1  1  4  1  1  1  1  1  4  1|
  4|  1  4  1  3  4  3  4  3  4  3  3  1  3  4  3  3  4  3  4  1|
  5|  1  3  9  9 10 10  9  5  6  6  6  6  4  9  9 10  9 10  4  1|
  6|  1  4  1  1  1  1  1  1  1  1  1  1  3  1  1  1  1  1  3  1|
  7|  1  3 10  9 10  9 10  6  6  5  6  6  4  9 10  9 10  9  4  1|
  8|  1  4  1  1  1  1  1  1  1  1  1  1  4  1  1  1  1  1  4  1|
  9|  1  2 10  9  9  9 10  6  5  5  5  6  2 10  9 10 10 10  9  1|
 10|  1  8  1  1  1  1  1  1  8  1  1  1  1  1  1  1  1  1  8  1|
 11|  1  7 10  9 10  9  9  5  8 10 10 10  9  9  9  9  9  1  7  1|
 12|  1  7  1  7  1  1  1  1  7  8  8  8  7  8  7  8  4  1  7  1|
 13|  1  1  1  8  1  1  1  1  1  1  1  1  7  7  7  3  4  1  7  1|
 14|  1  6  6  7 10  9  1  1  1  1  1  1  7  8  6  6  5  5  8  1|
 15|  1  5  6  7  8  7 10 10  9 10 10  5  8  1  1  1  1  1  8  1|
 16|  1  5  6  8  7  8  7  8  8  8  8  5  8 10  9  9 10  1  7  1|
 17|  1  7  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  8  1|
 18|  1  7 10  9  9 10  9 10  9 10  9  6  6  5  5  6  5  6  7  1|
 19|  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1|
   +-----------------------------------------------------------+
```

Fig. 2. A Ship Evacuation Visualization

III. Visualization of Cell-DEVS Models in Maya

As seen in Fig. 2, this visualization is complex to understand. In order to improve this, we created a 3D visualization using Maya [10], a powerful application for three dimensional modeling and animation, using special effects and rendering. Maya allows one to create digital imagery, three dimensional animation and visual effects. The Maya software interface is fully customizable and it allows users to extend their functionality within Maya by providing access to the Maya Embedded Language (MEL). Using MEL, programmers can tailor the user interface to their needs and to add in-house tools. Since MEL is recognized by embedded web browsers, MEL commands can also be issued form a webpage. Maya's modeling and animation tools were used to create three-dimensional environments for Cell-DEVS and DEVS models. The current version of the CD++/Maya Simulations focuses on separating the functions by their functionality. This makes for cleaner code as well as easier future development. Aside from the main log file readers, there are three supporting functions:

1. CellPosition – responsible for reading the coordinates off the log file and translating them to the real 3D plane
2. translateTime – responsible for reading the time off the log file and translating it to the Maya timeline. This function is required to allow the simulation to follow the proper timing of events as described in the log file
3. createShader – this function is used to add colors to the scenes. Due to the wide variety of simulations and scenes that need to be created this file can be easily modified to reflect the materials currently under consideration. It can be set to draw out anything from marble to wood and grass as defined by the user.

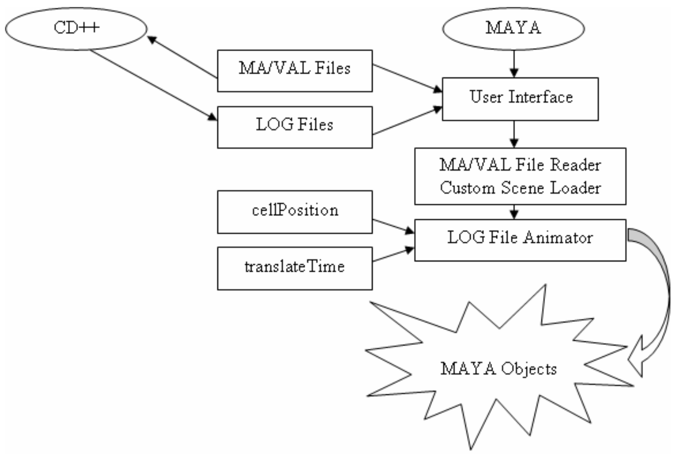


Fig. 3. Maya/CD++ Simulations Architecture

Fig. 3 above illustrates the order of events which occur for CD++ simulation visualization to be created. In the beginning, the user creates the MA (VAL Files) depending on the type of simulation to be executed. These files are then passed on to CD++ for simulation execution. During the simulation, CD++ records all events, cell values and time variables into a LOG file, which in the end is saved under a user defined name. Once the LOG file is created, we may proceed to execute the MAYA visualization engine. Depending on the simulation type, the program may need the MA file for proper grid initialization. In the case where the simulation is applied to an already created Maya scene, the MA file is ignored. The LOG file however, is an essential part of this process. It contains the cell values and time in which these values change. The entire Maya visualization engine is based on the LOG file. Once the log file is passed to the Maya scripts, they go through it and extract the required values. The translateTime function reads the time variables off the LOG file and recreates them to match the Maya movie time format.

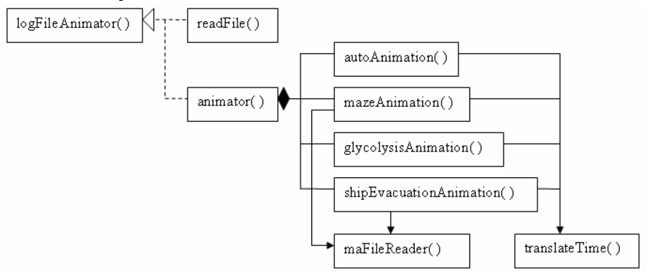


Fig. 4. Architecture of the visualization environment

Fig. 4 shows the relationships between these procedures. The *logFileAnimator* method acts as an interface requesting the user to select a particular model. The user has two choices after providing the required information; the "Print File Contents" button will instantiate *readFile* and the "Animate" button will instantiate *animator*. The *readFile* method locates and opens the file corresponding to the file name provided for the express purpose of reading it and printing the contents to the Script Editor Window in Maya. In this way, the user can analyze the detailed results found in the log files. The *animator* method instantiates the animation procedure for that particular model, associating CD++ simulation results with graphic scenes defined in Maya. Each instance of the animation procedure opens the log File, reads it and stores pertinent information, which is then used to animate the objects in the three dimensional scene opened. All the information pertaining to a particular object from the log file is used to animate that same object in the scene file. The *translateTime* method is in charge of accurately following the log File, and making the animation to match time with the time present in the simulation log.

IV. EVACUATION MODEL FOR SAT BUILDING

Sophisticated evacuation models have been developed to assist rescue and emergency response crews with proper situation analysis and prompt reaction procedures. The ability to simulate and represent such situations increases the training efficiency and creates the opportunity for better condition understanding. For the purpose of our simulation we use the SAT building's floor plan.

The Society for Arts and Technology (SAT) building is located on blvd. St. Laurent in downtown Montreal. This building is a center devoted to the creation, development and conservation of digital culture. We have created an advanced model to studying evacuation in the SAT building. This model uses variables such as panic level, distance from exits, etc. Taking these into consideration one can create sample situations using the rules described in Fig. 5.

```
[floor]
type : cell    dim : (49,27,2)    delay : INERTIAL
defaultDelayTime : 1 border : wrapped
localtransition : EvaRule

neighbors : (-1,-1,0) (-1,0,0) (-1,1,0) (0,-1,0)
(0,0,0) (0,1,0) (1,-1,0) (1,0,0) (1,1,0) (-1,-1,1)
(-1,0,1) (-1,1,1) (0,-1,1) (0,0,1) (0,1,1) (1,-1,1)
(1,0,1) (1,1,1)

[EvaRule]
% Rules to control the movement of each individual
rule:{#pos1+1}{1000/#pos0}{(0,0,0)>0 AND #pos0=0 ...
rule:{#pos1+3}{1000/#pos0}{(0,0,0)>0 AND #pos0=0 ...
rule:{#pos1+5}{1000/#pos0}{(0,0,0)>0 AND #pos0=0 ...
rule:{#pos1+7}{1000/#pos0}{(0,0,0)>0 AND #pos0=0 ...
rule:{#pos1+6}{1000/#pos0}{(0,0,0)>0 AND #pos0=0 ...
rule:{#pos1+8}{1000/#pos0}{((0,0,0)>0 AND #pos0=0...
```

Fig. 5. Evacuation rules as set in the CD++ Model file

The above defined rules have two separate major parts: the initialization function defines the size of the cell space, the neighborhood, the initial values of the cells and the name of the function responsible for the cell behavior. The model uses two layers. The coordinates of each object are divided in two. Layer 0 is responsible for the boundaries (walls, exits, etc) and people. Layer 1 is responsible for the objects (not people – i.e. internal walls, chairs, columns etc) and the distance numbers.

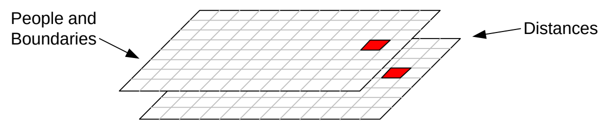


Fig. 6. Dual layer setup of the CD++ evacuation simulation.

The first set of rules in Fig. 5 serves to define what path a person should follow using the orientation plane. The basic idea is to take the direction that decrease the potential of a cell, building a path following the lower value of the neighbors. We have 8 rules to control the people's movement, one per direction. In all cases the rule analyze the 8 near neighbors to understand what direction the person should take (if all the 8 near neighbors have the same value, the movement is at random). Besides these basic rules, we have included a different set of rules governs the panic behavior: a person will take a wrong path or will not follow the orientation path. In that case, the direction will be calculated taking the path where the cell's potential is increased. In this case also we analyze the 8 near neighbors, and we also avoid people collisions. The values of the cells containing the people are set using the following number rules [6]:

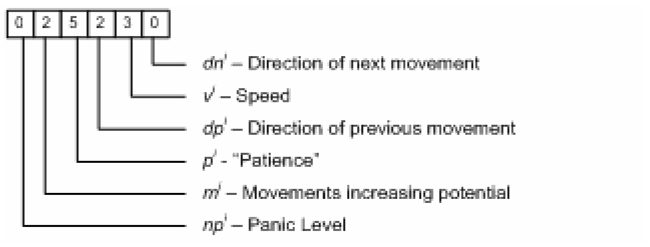

0 2 5 2 3 0
dn' – Direction of next movement
v' – Speed
dp' – Direction of previous movement
p' - "Patience"
m' – Movements increasing potential
np' – Panic Level

Fig. 7. Cell value digit description

- *dn* represents responsible for the direction of movement of the entities (1:W; 2:SW; 3:S; 4:SE; 5:E; 6:NE; 7:N; 8:NW).
- *v* is responsible for the speed of an entity. This allows us to implement different people speeds, which makes for a more realistic evacuation (expressed in cells per second: 1 to 5)
- *dp* is the Last movement direction (from 1 to 8, as in *dn*).
- *p* represents the emotional state of the person: the higher this value is the lower the probability that a person gets in panic.
- *m* represents a person's moving potential. A person moves to decrease the movement potential by decreasing the distance to the exit. If there is no available move decreasing the potential, a person will try to move to a neighboring cell that has the same potential. Otherwise, the person will move further away in an attempt to find another route, as seen in Fig. 8.
- *np* defines the panic level (represent the number of cells that a person will move increasing the cell potential). A situation where *p* is low and *np* is high will represent a high panic situation in which the entity will very likely choose a wrong move.

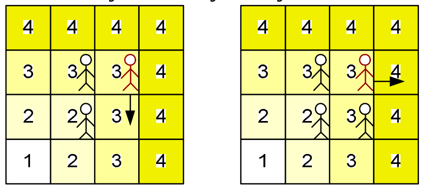
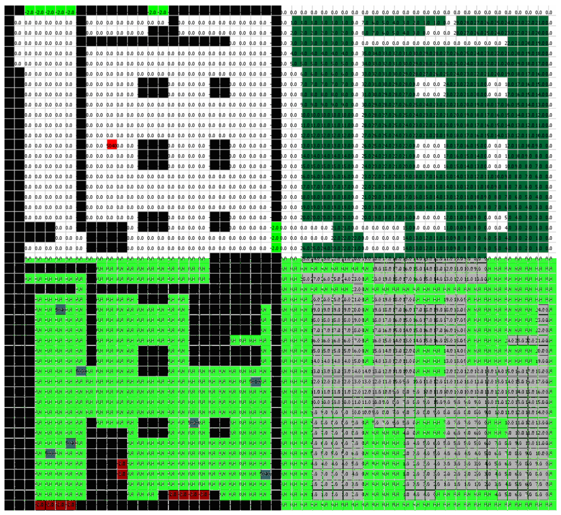


Fig. 8. Movements decreasing / increasing potential

V. EVACUATION SIMULATION RESULTS

In this section we discuss the results of different simulations we executed. They are all based on the same model (defined in Section 4), but all use different cell values which correspond to different human behaviors – speed, panic etc. Our first example considers a basic model consisting of eight people without panic behavior. They are initially placed at random inside the building, and as the level of complexity is small, we could observe that they follow the second layer to exit the building. The building is almost empty (which is a normal condition for SAT); however, there are people in each sector. This evacuation is designed to give us a general idea of the exit directions people will follow, which will help us in developing the successive simulations. Initially, there is no panic, and we did not change the movement potential, using a high level of patience. As we can see, the building is evacuated in **13:015s**.



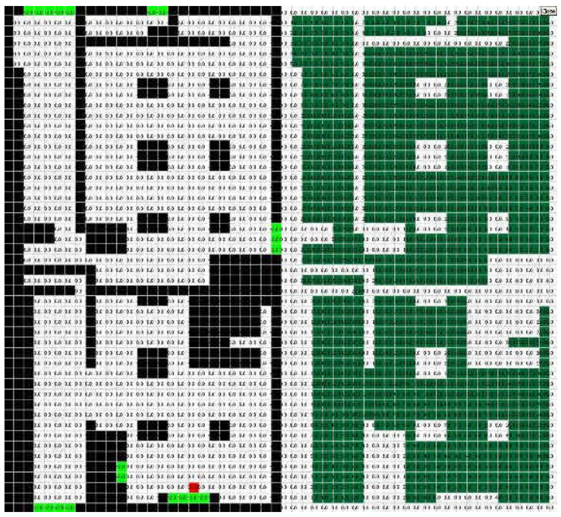


Figure 9. a) SAT at time: 00:000–Initial placement of people; b) time: 13:015–Last person to leave the building

The example presented in Figure 10 represents the placement of eight people; however they are all located in the down left hand corner of the plan. The panic level is still 0 to follow an organized simulation and show us how people would evacuate under normal conditions. In this case, we can see a bottleneck situation, and we can visualize a pile up around one specific exit. Although the total evacuation time is **04:005**, this occurs because of the proximity of the entities to the exit.

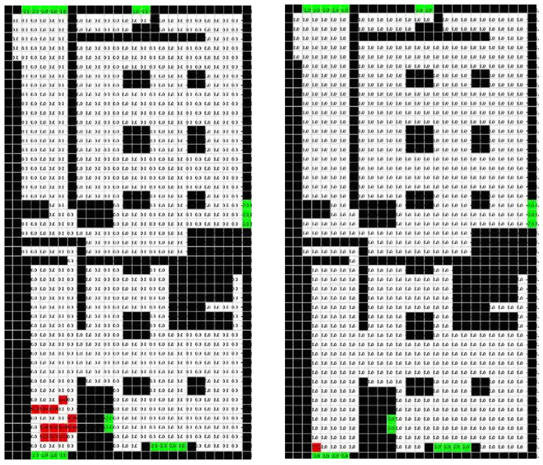


Figure 10-a) time: 00:000; b) time: 04:005

As we can see, the building is also evacuated in an orderly fashion. We then used the same model, and included panic effect into one of the people's initial value. If we analyze the execution results on Figure 11, we can notice that a person move away from the exit due to a blocked move for this person. The rest of the people leave the building normally. The total evacuation time is **05:004s.** In order to observe the effect of panic on the simulation time we used the exact same number of people and their positions as specified above. This time however we introduce a full panic level which could correspond to a very close proximity event etc.

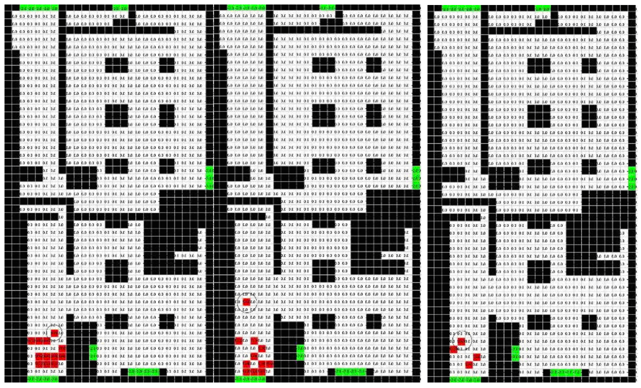


Figure 11- Evacuation with panic (one person): 05:004s.

We can notice an increase in evacuation time up to three times larger than that observed in the previous simulation (which shows us how important for the outcome the panic level is). Figure 12 below illustrates these results. The initial values are the same as specified above with the only difference being the introduction of the panic digit for every person.

We then increased the number of people but we added more people to the other two exits on the right, which offer an interesting evacuation situation: two totally separate exits in very close proximity to each other. This would allow us to follow people's behavior and proper choice of closest exit. In case of no panic the people would follow the second layer and decide where the closest exit is. However, if panic is introduced the chaotic movements result in longer traveling times.

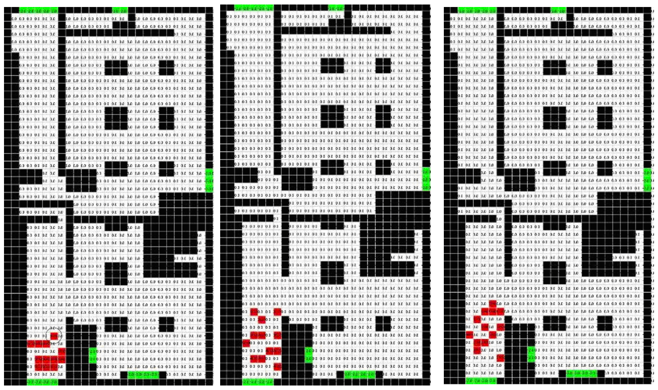


Figure 12- Evacuation with panic. 15:519s.

The simulation on Figure 13 has low panic levels introduced to all people. We introduce a larger number of entities at a low panic level. This will help us study a regular exit situation. The lack of panic will allow us to focus on the entity pile up part of the problem rather than the incorrect choice of exit and chaotic movements. The increased amount of people does not necessarily mean an increased evacuation time as we can see from Figure 13. As long as the panic level is low, the evacuation is properly controlled.

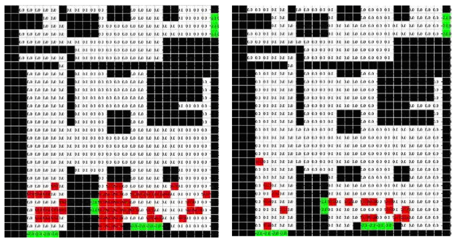


Figure 13-Low level of panic.

In the last example, we implement the same initial positions for the entities. At this point we introduce the panic factor into the cell values. Once we have figured out that a bottleneck would in fact occur in front of the two exits on the right side of the plan, we can proceed to see how panic levels will affect that. We notice chaotic movements of the entities. The total evacuation time of **25:029** also speaks of that. As can be seen in Figure 14, due to the blocked exits, people move in directions that increase their movement potential.

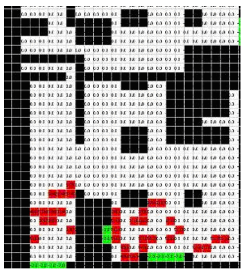


Figure 14- High level of panic

The starting positions are kept the same and the only difference between the two results is the panic level. We can notice a difference of about 10 seconds between the two -> the second evacuation being slower due to the panic which causes confusion and chaotic movements.

VI. 3D Visualization of the Evacuation Simulations

Once the log file has been completely generated, we use Maya to visualize the model in 3D. We start by defining the simulation type, the coordination files (in our case completed scene) and the file locations into the user interface (which can be activated through web services, allowing us to remotely execute the CD++ simulations to obtain the log files over the internet). After rendering the SAT scene we can see better detail on the building to give us better familiarity with the setting. The plan as displayed in **Error! Reference source not found.** the origin of the plan as implemented by the CD++ simulations. Once the building has been loaded, the CD++/Maya Simulations proceeds to load the initial values for the cell spaces – in our case people inside the building.

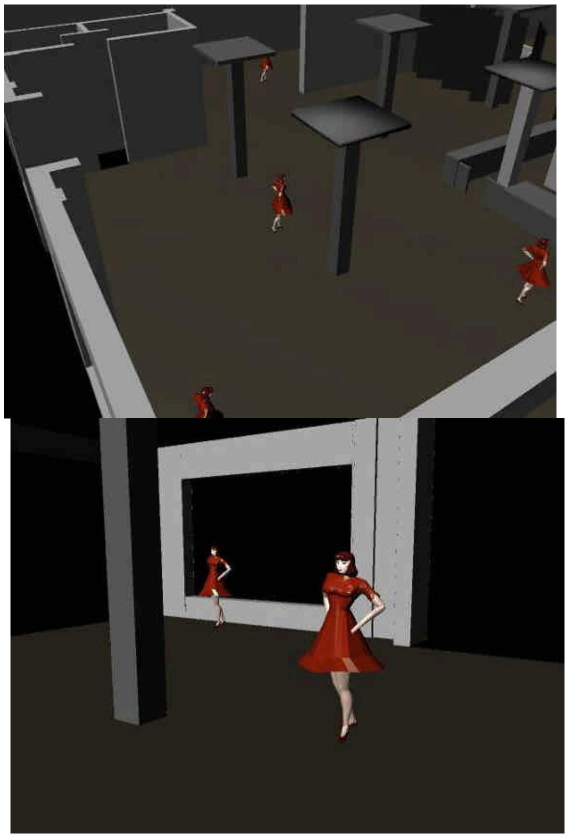


Figure 15: Different angles of visibility within Maya.

In order to visualize the scene properly, we setup different rendering cameras in Maya. This allows us to follow the entities motion throughout the building. Once the script has been initiated it reads through the log file looking for the Y messages. Every Y message carries information about the current cell values and locations. The MEL script uses these values and coordinates to relocate the human figures. This results in a frame based motion of the human figures and hence makes for an easy to see evacuation model. The following are five rendered images of five separate frames that demonstrate the progressive motion of the human figures towards the dedicated building exits as can be seen in Figure 16.

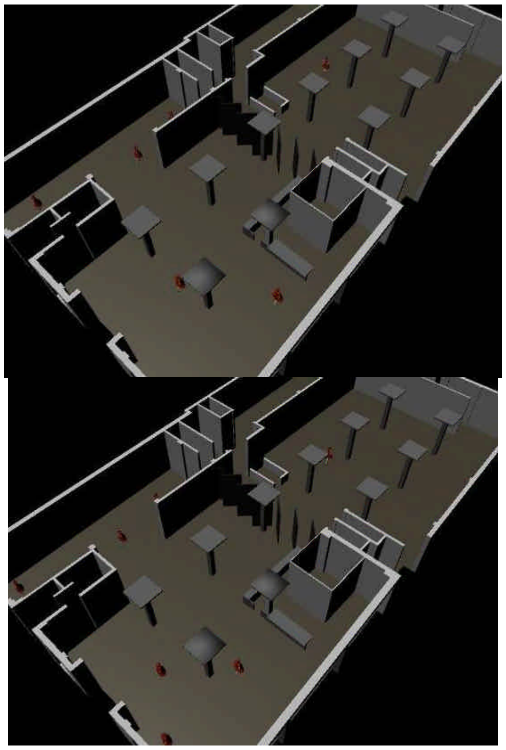


Figure 16. Rendered images of the evacuation for Figure 9

The last figure we present shows an evacuation scenario in which there is a large number of people in the building which allows us to see the potential application of this environment into crowd situations.

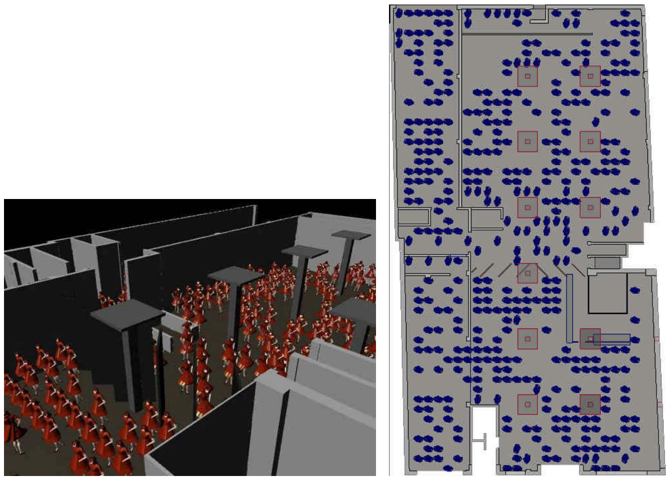


Figure 17. A busy day at the SAT building.

VII. CONCLUSION

Simulation is becoming increasingly important in the analysis and design of complex systems. CD++ is a tool for the simulation of complex physical systems that can be used to simulate a variety of models. To facilitate the users to use the CD++ simulator, we extended its design to provide a number of services. The 3D visualization GUI enables sophisticated visualization of DEVS and Cell-DEVS models. To better understand the results, the user can select shapes to represent a node in the 3D space, select different colors, shapes, edit scenes, etc. The current facilities have highly improved the use of the previously existing tools, thus enhancing the analysis experience of the modelers using the toolkit. The high level language of CD++ reduces the algorithmic complexity for the modeler while allowing complex cellular timing behaviors. DEVS allows independence of the simulator, the models developed, the experiment conducted and the visual engine, while maintaining unity in the model specification and tool interoperation.

The 3D visualization GUI enables sophisticated visualization to better understand the results. The current facilities have highly improved the use of the previously existing tools, thus enhancing the analysis experience of the modelers using the toolkit. The visual models have visual state transition systems, which define how the simulation models are graphically represented during visualization. The visual models also have event animation rules to create animations for certain events.

REFERENCES

[1] S. Wolfram. “Theory and applications of cellular automata”. Vol. 1, Advances Series on Complex Systems. World Scientific, Singapore, 1986.

[2] G. Wainer; N. Giambiasi,. "Timed Cell-DEVS: modelling and simulation of cell spaces". In Discrete Event Modeling & Simulation: Enabling Future Technologies. 2000. Springer-Verlag.

[3] B. Zeigler, H. Praehofer, T. Kim. Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. 2000. Academic Press.

[4] Treiber, M.; Hennecke, A.; Helbing, D. “Congested Traffic States in Empirical Observations and Microscopic Simulations”. Physical Review E 62, 2000, 1805-1824.

[5] Klüpfel, H.; Meyer-König, T.; Wahle J.; Schreckenberg, M. “Microscopic Simulation of Evacuation Process on Passenger Ships”. In "Theoretical and Practical Issues on Cellular Automata”, Springer-verlag 2001.

[6] Meyer-König, T.; Klüpfel, H.; Schreckenberg, M. “A microscopic model for simulating mustering and evacuation processes onboard passenger ships”. In Proceedings of TIEMS 2001, Oslo, Norway. 2001.

[7] Hongtae Kim, Dongkon Lee, Jin-Hyoung Park, Jong-Gap Lee, Beom-Jim Park and Seung-Hyun Lee. “Establishing the Methodologies for Human Evacuation Simulation in Marine Accidents”. Proceedings of 29th Conference on Computers and Industrial Engineering. Montréal, QC. Canada. 2001.

[8] J. R. Weimar. “Cellular automata model for ship evacuation”, http://www.jweimar.de/jcasim/schiff1.html), [June 2006].

[9] C&CA. Proceedings of the 1st Workshop on Crowds and Cellular Automata. ACRI 2006. Perpignan, France. 2006.

[10] Autodesk Corp. "Maya 6 Features in Detail,” http://www.alias.com/eng/products-services/maya/file /maya6_features _in_detail.pdf. 2004,

[11] Wainer, G. "CD++: a toolkit to define discrete-event models". Software, Practice and Experience. Wiley. Vol. 32, No.3. pp. 1261-1306. November 2002.

[12] Braunstein, M; Ameghino, J; Wainer, G.. “Modeling evacuation processes using Cell-DEVS”. Internal report. Computer Science Department. Universidad de Buenos Aires. 2003.

[13] J. Ameghino; G. Wainer: “Application of the Cell-DEVS formalism for modeling cell spaces” In Proceedings of AIS’2004, Jeju Island, Korea, Lecture Notes in Computer Science, 2004.

[14] G. Wainer and W. Chen. "A framework for remote execution and visualization of Cell-DEVS models". *Simulation*. Vol. 79, pp. 626-647. November 2003.