Using Discrete-Event Cell-Based Multimodels for the Simulation of Evacuation Processes

Chapter · January 2015

DOI: 10.1007/978-3-319-15096-3_8

1 author:

G. A. Wainer
Carleton University
414 PUBLICATIONS **2,866** CITATIONS

SEE PROFILE

Using Discrete Event Cell-Based Multimodels for the Simulation of Evacuation Processes

Gabriel Wainer

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Dr. Ottawa, ON. K1S 5B6. Canada.

Abstract: Multimodeling concepts allow designers of complex systems to organize their work better and to address the different components of a given system at the right level of abstraction. Agent-based modeling and simulation allows defining the behavior of the different components in a complex application with ease, allowing one to focus on the particular behavior of the different entities in the model. These concepts, pioneered by Prof. Ören, have had an important impact in the field of modeling and simulation. Here we show how to use these ideas in combination with the DEVS formalism invented by Prof. Zeigler, in order to build complex spatial models with focus on building construction and evacuation processes.

Introduction

Prof. Ören is a pioneer in the fields of Multimodeling and Agent-Based Modeling and Simulation. In [1] he introduced the concepts of extension and generalization for multimodel formalisms, including formal M&S like DEVS (Discrete Events Systems specifications) [2]. According to [3], a multimodel can be defined as a modular mathematical entity that subsumes multiple submodels that together represent the behavior of the model. Multimodels, introduced in [2], were extended in [4] to facilitate generalization of discontinuity in piecewise continuous systems.

The work of Prof. Ören in the area of agent-based simulation is extensive; in particular, we are interested in human behavior modeling, like in [5]. In [3], Profs. Ören and Yilmaz define a detailed taxono-

my on agent-based multimodeling methodologies. The idea is that agent-based simulation allows defining entities (the agents) that can perceive and reason about their environment and provide responses in order to achieve a goal.

Our objective is to apply these ideas to the field of evacuation simulation processes. In recent years, many simulation models of real systems have been represented as multimodels with agents in the shape of cell spaces [6, 7, 8]. In particular, evacuation processes are important applications, and a necessary step in building design. Evacuation simulation is useful for various reasons, such as preventing collapse during evacuation and reducing building evacuation time. Small changes in in the building design can result in having important differences, thus, simulation can be used for studying the influence of changing the location of stairways or adding sufficient emergency exits to ensure that the building can be evacuated rapidly. In such cases, the models can be built using an agent-based approach for the behavior of the individuals (who need to find the closest exit, might exhibit panic behavior, might need to meet with friends and family or gathering their property; the behavior of the agents should be properly modeled). Using a multimodel approach is important, as different entities at different levels of abstraction exist: individuals, flocks of individuals, buildings, corridors, rooms, and stairs, elevators, obstacles, and even complete city sections. By first creating a virtual version of the building, it is possible to test many different designs to get important measurements such as evacuation time to find the best design. This way, potential problems and can be avoided and fixed before construction begins. A multimodel approach and the definition of varied behavior of the agents being evacuated can help in developing a better design.

In this chapter we will discuss how to address these issues, by defining multimodels that can be represented as cell spaces, in which agents represent the behavior of the evacuating agents, and their application in different construction scenarios. We focus on 2D and 3D visualization of the simulation results in order to make the models easier to understand and analyze.

According to the taxonomy in [3], our model can be included in the following categories:

- According to the *number of submodels* , it's a multiaspect model, as there are various submodels active at the same time (one per individual at least; we also have models specifying stairs and exits in a building)
- Based on the model's *variability*, it's a static-structure multimodel (representing the building as a cell space, and individuals moving to each cell)
- In terms of the *nature of knowledge to activate the models*, it is an adaptive multimodel, as the submodels behavior is driven by constraints (space, obstacles, building floor plan, panic, etc.)
- The *location of the knowledge* to activate the submodels is within the multimodel; therefore, we can say this is an active multimodel.

Multimodels in DEVS and Cell-DEVS

A popular multimodel method to describe agents that have spatial properties is called Cellular Automata (CA), a well-known formalism to describe cell spaces in which individual agents are spatially located in cells in a 2 or 3D spaces [9, 10]. CA, defined as infinite n-dimensional lattices of cells whose values are updated according to a local rule. Cell-DEVS [11, 12] was defined as a combination of cellular automata and DEVS (Discrete Events Systems specifications) [2]. The goal is to improve execution speed building discrete-event cell spaces, and to improve their definition by making the timing specification more expressive.

DEVS is a systems theoretical approach that allows the definition of hierarchical modular multimodels. 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. DEVS is a formalism proposed to model

discrete events systems, in which a model is built as a composite of basic (behavioral) models called **atomic** that are combined to form **coupled** models. A DEVS atomic model is defined as:

$$M = < X, S, Y, \delta_{INT}, \delta_{EXT}, \lambda, ta > \quad (1)$$

Where **X** represents a set of input events, **S** a set of states, and **Y** is the output events set. Four functions manage the model behavior: δ_{INT} the internal transitions, δ_{EXT} the external transitions, λ the outputs, and D the duration of a state. Each atomic model can be seen as having an interface consisting of input (X) and output (Y) ports to communicate with other models. Every state (S) in the model is associated with a time advance (*ta*) function, which determines the duration of the state. Once the time assigned to the state is consumed, an internal transition is triggered. At that moment, the model execution results are spread through the model's output ports by activating an output function (λ). Then, an internal transition function (δ_{INT}) is fired, producing a local state change. Input external events are collected in the input ports. An external transition function (δ_{EXT}) specifies how to react to those inputs.

Once an atomic model is defined, it can be incorporated into a coupled model is defined as:

$$CM = < X, Y, D, \{Mi\}, \{Ii\}, \{Zij\}, select > \quad (2)$$

Each coupled model consists of a set of **D** basic models **Mi**. The list of influences **Ii** of a given model is used to determine the models to which outputs (**Y**) must be sent, and to build the translation function **Zij**, in charge of converting outputs of a model into inputs (**X**) for the others. An index of influences is created for each model (**Ii**). For every j in the index, outputs of model **Mi** are connected to inputs in model **Mj**. 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 in-

puts/outputs from/to external models. The **select** function decides how to deal with simultaneous events.

Cell-DEVS extended the DEVS formalism, allowing the implementation of cellular models with timing delays. In Cell-DEVS, each cell of a cellular model is defined as an atomic DEVS. Cell-DEVS atomic models are specified as:

$$TDC = < X, Y, S, \theta, N, delay, d, \delta_{INT}, \delta_{EXT}, \tau, \lambda, D > \quad (3)$$

Each cell will use **N** inputs to compute the future state **S** using the function**τ**. The new value of the cell is transmitted to the neighbors after the consumption of the delay function. **Delay** defines the kind of delay for the cell, and **d** its duration. The outputs of a cell are transmitted after the consumption of the delay.

Once the cell atomic model is defined, they can be put together to form a coupled model. A Cell-DEVS coupled model is defined by:

$$GCC = < X_{list}, Y_{list}, X, Y, n, \{t_1,...,t_n\}, N, C, B, Z > \quad (4)$$

The cell space **C** defined by this specification is a coupled model composed by an array of atomic cells with size $\{t_1 \text{ x...x } t_n\}$. Each cell in the space is connected to the cells defined by the neighborhood **N**, and the border (**B**) can have different behavior. The **Z** function allows one to define the internal and external coupling of cells in the model. This function translates the outputs of output port m in cell C_{ij} into values for the m input port of cell C_{kl}. The input/output coupling lists (**Xlist**, **Ylist**) can be used to interchange data with other models.

The CD++ tool [12, 13] was developed following the definitions of the Cell-DEVS formalism. CD++ is a tool to simulate both DEVS and Cell-DEVS models. Cell-DEVS are described using a built-in specification language, which provides a set of primitives to define the size of the cell-space, the type of borders, a cell's interface with other DEVS models and a cell's behavior. The behavior of a cell

(the τ function of the formal specification) is defined using a set of rules of the form: *VALUE DELAY CONDITION*. When an external event is received, the rule evaluation process is triggered to calculate the new cell value. Starting with the first rule, the CONDITION is evaluated. If it is satisfied, the new cell state is obtained by evaluating the VALUE expression. The cell will transmit these changes after a DELAY. If the condition is not valid, the next rule is evaluated repeating this process until a rule is satisfied.

The specification language has a large collection of functions and operators. The most common operators are included: Boolean, comparison, and arithmetic. In addition, different types of functions are available: trigonometric, roots, power, rounding and truncation, module, logarithm, absolute value, minimum, maximum, G.C.D. and L.C.M. Other available functions allow checking if a number is integer, even, odd or prime. In addition, some common constants are defined. Figure 1 shows the definition of a very simple example of the definition of such models.

```
[life]
size:(20,20) delay:transport border:wrapped
neighbors : (-1,-1)(-1,0)(-1,1)(0,-1)(0,0)(0,1)(1,-1) (1,0) (1,1)
localtransition : life-rule

[life-rule]
Rule: 1 10 {(0,0)=1 and (truecount=3 or truecount=4) }
Rule: 1 10 { (0,0) = 0 and truecount = 3 }
Rule: 0 10 { t }
```

Fig. 1. Definition of the Life game in CD++.

The rules in this example say that a cell remains active when the number of active neighbors is 3 or 4 (*truecount* indicates the number of active neighbors) using a transport delay of 10 ms. If the cell is inactive (`(0,0)=0`) and the neighborhood has 3 active cells, the cell activated (represented by a value of 1 in the cell). In every other case, the cell remains inactive (*t* indicates that whenever the rule is evaluated, a *True* value is returned).

Agents for evacuation processes in Cell-DEVS

In recent years, models for building evacuation 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 training efficiency and creates an opportunity to understand the evacuation process better. The goal is to learn where the bottlenecks can occur, and which solutions are effective to prevent congestion during evacuation [6, 7, 8]. The basic idea of the model was to simulate the behavior and movement of every single agent (a person) involved in the evacuation process using a multimodel approach [14]. We defined a Cell-DEVS model with various rules to characterize a person's behavior:

- People try to move towards the closest exit.
- A person in panic might move in the 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 used two planes to represent this spatial model: one for the floor plan of the building and to represent the people moving, and the other for the orientation towards the exits, as we can see in Figure 2.

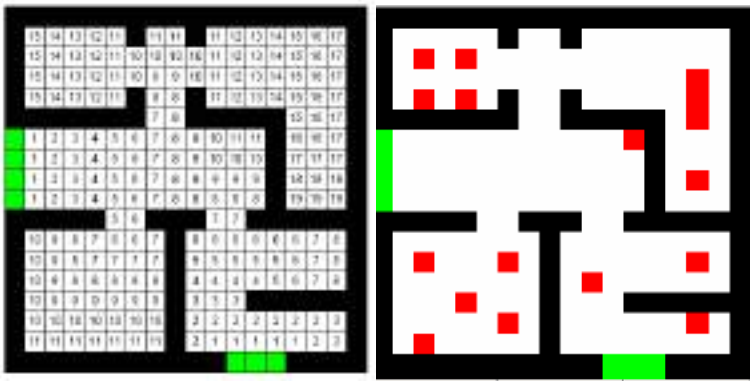


Fig. 2. (a) Orientation layer: potential value (b) Individuals.

Each cell in the grid represents 0.4 m^2 (one person per cell). The coordinates of each object are divided into two: Boundaries and people, and Objects (i.e. walls, chairs, columns, etc.). The orientation layer contains information that serves to guide persons towards the exits. We assigned a potential distance to an exit to every cell of this layer. The persons will move for the room trying to minimize the potential of the cell in which they are. That is, the people to a cell with decreasing potential. Each cell can contain different values: exits (value = -2), obstacles (value = -1), distance to the exit (positive value in the first plane), and information about the individuals in the cell. The individuals' information is represented by a six digit value, in which each digit represents a different property, as follows.

Digit	**Property**
6	Next movement direction. 1:W; 2:SW; 3:S; 4:SE; 5:E; 6:NE; 7:N; 8:NW
5	Speed (cells per second: 1 to 5)
4	Last movement direction, it can vary from 1 to 8 (as digit #6)
3	Emotional state: the higher this value is the lower the probability that a person gets in panic and changes direction.
2	Number of movements to increase the potential of a cell. If a person moves this number of times, the person, which is now in panic, can move into a different direction in which the potential is increased.
1	Panic Level, representing the number of cells that a person will move in increasing direction of potential.

Table 1. Values used to represent the agents behavior

For instance, a cell with a value of `009121` represents an individual going to the W (first digit =1), at a speed of 4.5 km/h (two cells per second, each cell is 0.4 m long), the last movement was W (the individual is keeping the current direction), a stable emotional state, the current panic level is 0 (no panic), and the person will not change the direction of potential. A person moves to decrease the movement potential by decreasing the distance to the exit. If there is no availa-

ble move that will decrease the potential, a person will try to move to a neighboring cell that has the same potential. If none is available, the person will move further away in an attempt to find another route.

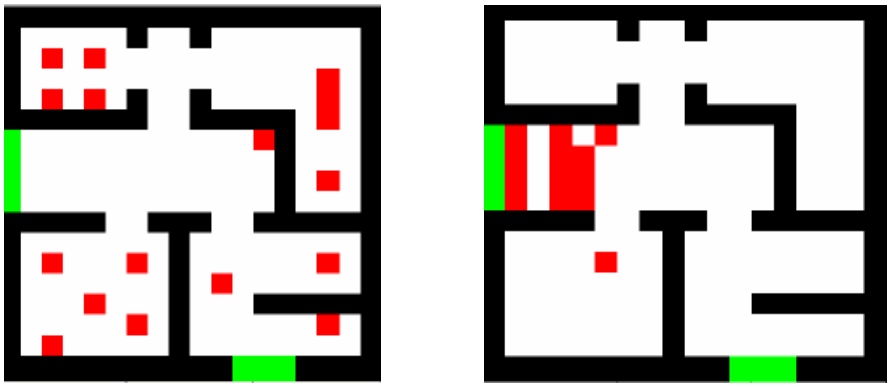
Figure 3 shows a subset of the rules used for evacuation model in CD++. We first define the Cell-DEVS multimodel (two layers, 18x18 cells each). The model uses inertial delays (which allows preemption, which is needed because we have to The first set of rules in the figure defines the path taken by a person using the orientation plane. The basic idea is to take the direction decreasing the potential of a cell, building a path following the lower value of the neighbors. We use eight different rules to control the people's movement, one for each direction. In all cases, the rules analyze the 8 near neighbors to understand what direction the person should take. We use a random direction (*randint*) when the near neighbors have the same value. The second set of rules model panic: a person in panic will take a wrong path or will not follow the orientation path. In that case, the direction is calculated by taking the path where the cell's potential is increased.

```
[evacuation]
dim : (18,18,2)        delay : inertial       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) ...

[EvaRule]
% Rules to govern people movement
rule : {trunc((0,0,0)/10)*10+1} {1000 / remainder(
         trunc((0,0,0) /10),10) } {((0,0,0)>0 AND
          remainder(trunc((0,0,0)/1),10) =0 AND remainder(trunc(
            (0,0,0)/100000),10) =0 AND ((0,-1,0)=0 OR (0,-1,0)=-2)
        AND cellPos(2)=0 )AND (((0,-1,1) <= (1,-1,1) OR (1,-1,0)>0
         OR (1,-1,0)=-1 OR (randint(5)=0)) AND ((0,-1,1)<= (1,0,1)
          OR (1,0,0)>0 OR (1,0,0)=-1 OR (randint(5)=0) ) AND
        ((0,-1,1) <= (1,1,1) OR (1,1,0)>0 OR (1,1,0)=-1 OR
          (randint(5)=0)) AND ((0,-1,1)<= (0,1,1) OR (0,1,0)>0 OR
            (0,1,0)=-1 OR (randint(5)=0)) AND ((0,-1,1)<=(-1,1,1)
              OR (-1,1,0)>0 OR (-1,1,0)=-1 OR (randint(5)=0) ) AND
               ((0,-1,1) <= (-1,0,1) OR (-1,0,0)>0 OR (-1,0,0)=-1
                 OR (randint(5)=0) ) AND ((0,-1,1) <= (-1,-1,1) OR
                  (-1,-1,0)>0 OR (-1,-1,0)=-1 OR (randint(5)=0) ))
}  ...

% Rules to control panic behavior
```

```
rule : {trunc((0,0,0)/10)*10+1}
         {1000/remainder(trunc((0,0,0)/10),10) } {((0,0,0)>0 AND
           remainder(trunc((0,0,0)/1),10)=0 AND remainder(trunc
            ((0,0,0)/100000),10)>0 AND ((0,-1,0)=0 OR (0,-1,0)=-2)
          AND cellPos(2)=0)AND (((0,-1,1)>= 1,-1,1) OR (1,-1,0)>0
           OR (1,-1,0)=-1) AND ((0,-1,1)>=(1,0,1) OR (1,0,0)>0 OR
             (1,0,0)=-1) AND ((0,-1,1)>= (1,1,1) OR (1,1,0)>0 OR
               (1,1,0)=-1) AND ((0,-1,1)>=(0,1,1) OR (0,1,0)>0 OR
                 (0,1,0)=-1) AND ((0,-1,1)>=(-1,1,1) OR (-1,1,0)>0
                   OR (-1,1,0)=-1) AND ((0,-1,1)>=(-1,0,1)
            OR (-1,0,0)>0 OR (-1,0,0)=-1) AND ((0,-1,1)>=(-1,-1,1)
                   OR (-1,-1,0)>0 OR (-1,-1,0)=-1))
}
```

Fig. 3. Specification of evacuation model.

The following figures show different visualizations for the simulation results for this model. Figure 4 shows a simple graphical representation of the simulation results. We can see the building shape (with walls in black and two exits: one to the left, and one to the bottom right), and people who want to leave the building using the exit doors. The evacuation path is the one previously presented in figure 2 (a). As we can see, there is a group of people blocking the left exit because individuals tend to move reducing the potential (and based on the original configuration the closest exit for most people is to the left).

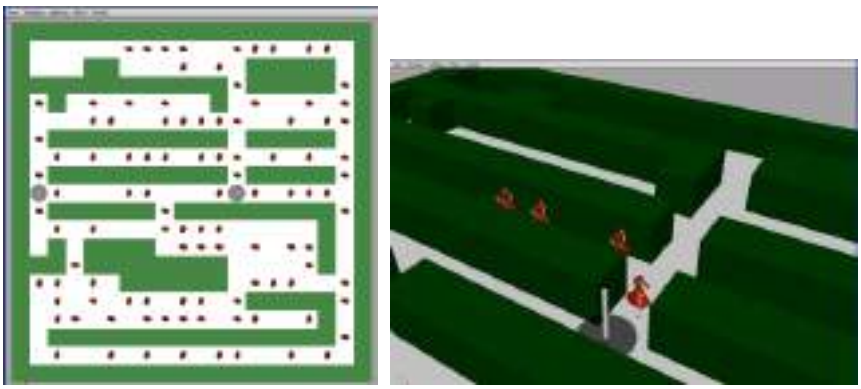


(a) **(b)**

Fig. 4. (a) People seeking an exit. (b) After 15 seconds, people found the exit.

Although CD++ provides different visual tools to as the ones used to generate the graphical results above, we need to build more sophisti-

cated 3D graphics, which improves data exploration. To overcome these drawbacks and to meet the diverse needs of different users, we have developed mechanisms to integrate the CD++ environment with a variety of both commercial and open-source visualization and rendering techniques, including Autodesk Maya, OpenGL, and Blender [15]. In this section, we will elaborate on these advanced techniques and demonstrate their capabilities with a wide range of applications. Autodesk Maya is one of the leading commercial software packages for 3D modeling, animation and visual effects. Maya software interface is fully customizable and it allows users to extend their functionality by providing access to the Maya Embedded Language (MEL). With MEL, users can tailor the GUI to fulfill their specific needs and to develop in-house tools. The MEL scripting language has been used in our research to create a high-performance 3D visualization engine [16], allowing for interoperability between a DEVS-based M&S tool and an advanced generic visualization environment like Maya. Users create a static scene in Maya, providing the necessary background for 3D animation of the simulation results. This Maya plugin allows showing different visualizations as seen in Figure 5.



(a) **(b)**

Figure 5. Evacuation Model at time (a) 00:00:00:000 (b) close up at time 00:00:05:240

As we can see, the visualization process using a 3D engine makes the results easier to observe and study. Figure 5 shows a different building configuration with two exits (one in the center and one to

the left), and people moving towards the exits and evacuating the building. A video of this visualization can be found at https://www.youtube.com/watch?v=GOOm1vFWG6Y&index=10&list=PLA7006DDBBF660D55

Evacuation example 1: the SAT Building

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 built a model based on existing floor plans to study the evacuation processes in the SAT building. This multimodel also uses various agents with different panic level, considers the distance from exits, etc. The 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.

Following a similar idea as in the previous section, the agents moving through the cells representing the space of the building use different values to represent different phenomena as follows:

```
[floor]
type : cell        dim : (49,27,2)       delay : INERTIAL
border : wrapped     initialCellsValue : eva-ej1.val
localtransition : EvaRule

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

% Neighbors in the lower level
neighbors : (-1,-1,1) (-1,0,1) (-1,1,1)
neighbors : (0,-1,1)  (0,0,1)  (0,1,1)
neighbors : (1,-1,1)  (1,0,1)  (1,1,1)

% Rules to control the movement decision of each individual
[EvaRule]
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+2} {1000/#pos0} {((0,0,0)>0 AND #pos0 =0 ...
rule : {#pos1+4} {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. 6. Evacuation rules in CD++

Once the model has been specified as above, the simulator generates a log file with the simulation results, as follows:

```
…
Message Y / 00:00:00:754 / floor(16,13,0)(893) / out / 5440
Message Y / 00:00:00:755 / floor(15,12,0)(837) / out /  0
Message Y / 00:00:01:005 / floor(16,13,0)(893) / out /  5443
Message Y / 00:00:01:006 / floor(17,13,0)(947) / out /  5340
Message Y / 00:00:01:007 / floor(16,13,0)(893) / out /  0
Message Y / 00:00:01:256 / floor(17,13,0)(947) / out /  5344
…
```

Fig. 7. Simulation log files

As we can see, the log file contains the time of the output messages generated by the agents on each cell, and the current value representing the combination of digits presented in Table 1 for each of the agents. In this case, they represent different individuals moving in different directions, some leaving a cell and others arriving into a new one. For instance, the person in cell (15, 12) abandons the cell (value = 0) and moves to cell (16, 13). The emotional state is 5 (average), it was moving in direction SW and now moves in direction S. It moves at a speed of 4 cells per second.

In the following examples we show different simulation scenarios showing different agents moving through this building. The first example, presented in Figure 8, shows a simple scenario with eight people distributed throughout the building.

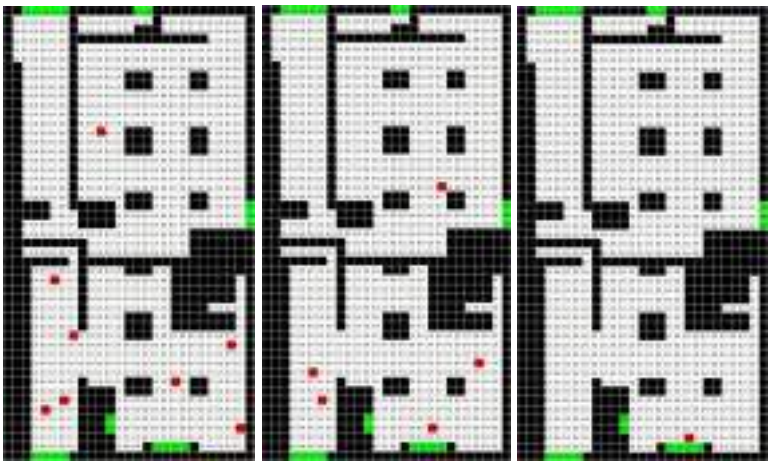


Fig. 8. SAT evacuation scenario: 8 individuals under normal circumstances.

The initial values for the cells in the figure are as follows:
(13,10,0)=5040, (36,24,0)=5010, (45,25,0)=5040, (40,18,0)=5020, (29,5,0)=5040, (35,7,0)=5030, (42,6,0)=5040, and (43,4,0)=5060.

As we can see, the first to digits on each agent have not been used (as we are not modeling panic). We can also notice different speed levels, which make for a more realistic simulation since not all people move at the same speed or pace.

We use the rules presented in Figure 6, with the agents placed at random inside the building, and following the path defined in the second layer to exit the building (no one is in panic). As the level of complexity is small, we could observe that they all followed the exit path. The building is almost empty (which is a normal condition for SAT); however, there are people in each sector. This evacuation gives us a general idea of the exit directions people will follow. In this case no one is in panic, and we did not change the movement potential, using a high level of patience. The total evacuation time for this scenario was 13:015s.

The example presented in Figure 9 also represents eight people; however they are all located in the bottom left corner of the floor plan. Although we do not include panic behavior, and the agents follow an organized evacuation pattern, we can see a bottleneck situa-

tion in one of the exits. Although the total evacuation time is short (04:005), this is occurs due of the proximity of the people to the exit.

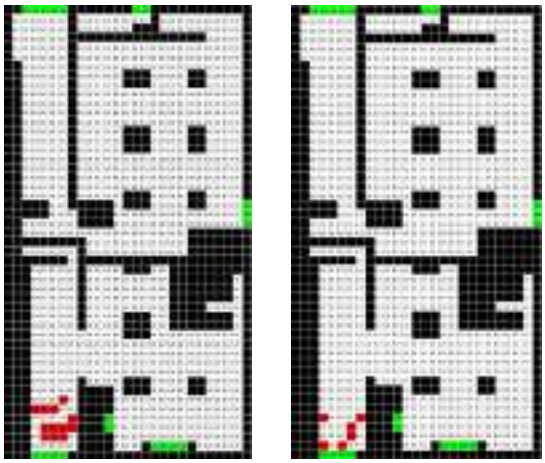


Fig. 9. a) time: 00:000; b) time: 01:005

Our following scenario includes panic behavior for one of the agents. If we analyze the execution results on Figure 10, we can notice that this person moves away from the exit, because it is blocked. The rest of the individuals leave the building normally. The total evacuation time is 05:004s (it takes longer because the person in panic finally returns to the main door after the bottleneck disappears). 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.

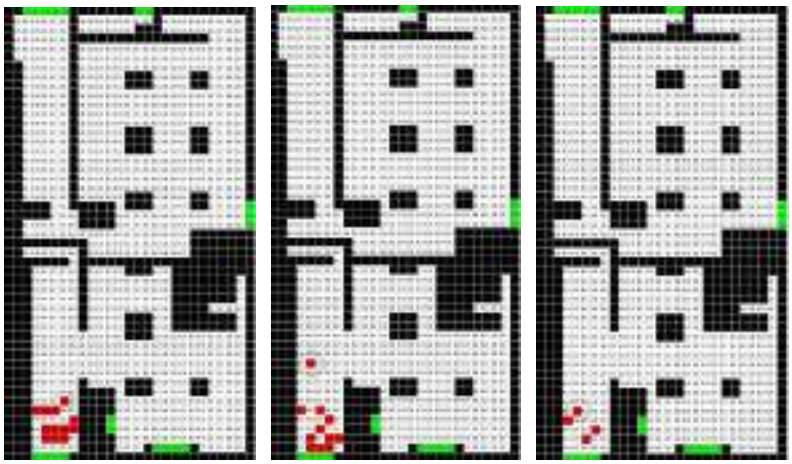


Fig. 10. Evacuation with panic (one person)

The following test uses a larger number of people in the building, but they are located closer to the exit on the right, which allows us to study the results of two separate exits close to each other. In order to observe the effect of panic on the simulation time we used the exact initial configuration on the left part of the building, and more people on the right. This time however we introduced the maximum panic level in all the individuals. We noticed an increase in the evacuation time up to three times larger than what was observed in the previous simulation (the total evacuation time was 14:774). We can see people moving away from the exits in any situation where there is a blockage, making the evacuation process much slower than in the previous cases.

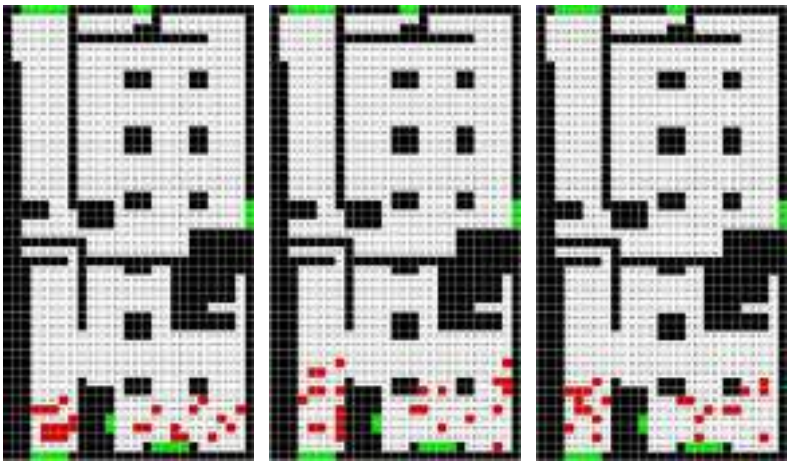


Fig. 11. Evacuation with panic (everybody)

The following scenario, presented in Figure 12, shows an initial configuration with identical people positions as in Figure 11; however the panic condition has been removed from the individuals on the right side of the building (representing, for instance, the fact that there is a fire on the left side of the building and the people on the right cannot see it). We can notice a better organized evacuation with movements focused on the exit on that side of the building. Nevertheless, the total evacuation time on the left was 15:607 because it took longer to evacuate the people in panic.

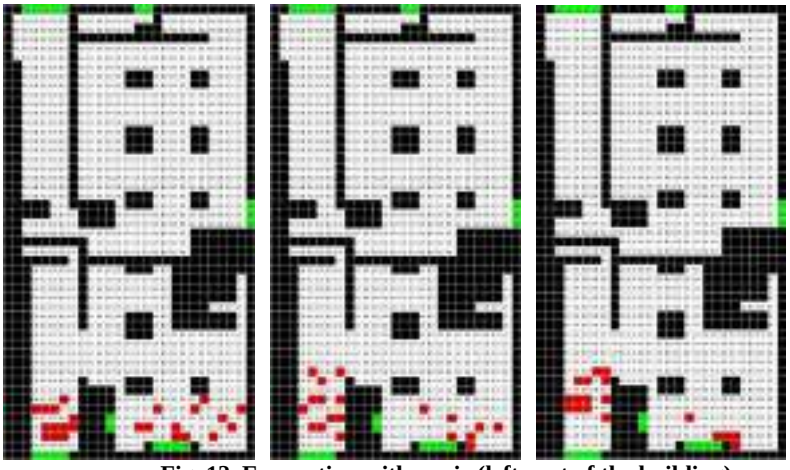


Fig. 12. Evacuation with panic (left part of the building)

As seen in Figure 13, when we increase the number of people on the right side of the building, and we change their speed (so they move slowly but with no panic), it still results on an organized evacuation on the right side while a random evacuation can be noticed on the left side due to high panic conditions (the total evacuation time is 16:611).

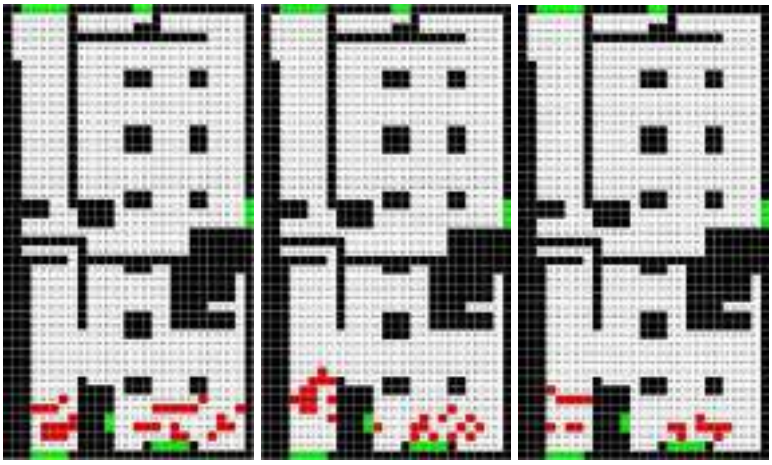


Fig. 13. Evacuation with panic (left part of the building)

As we can see in all of these examples, the multimodeling methodology allows us to define varied components for studying the evacuation process with ease, while the agent-based approach lets us focus on the individual behavior of each person, which results in a simpler mechanism to define behavior.

The same log files used to generate the figures above were used in Autodesk 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 building scene we can see better detail on the building to give us better familiarity with the setting. For the SAT building the initial scene setup looks like in Figure 14.

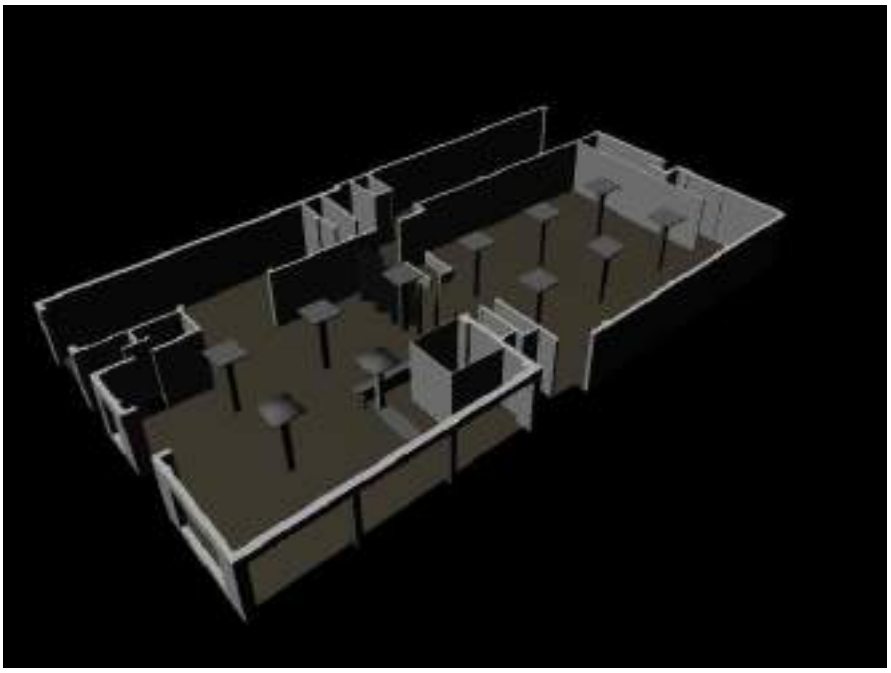


Fig. 14. Initial configuration for the SAT building in Maya

Once the building floor plan has been loaded and rendered, the CD++/Maya plugin loads the initial values for the cell spaces – in our case people inside the building. Then, we search the log file looking for the Y messages (which, as seen in Figure 7, carries information about the current cell values and locations). The MEL script uses these values and coordinates to relocate the human figures. This organization results in a frame based motion of the human figures and hence makes an easy to see evacuation model. The following are five rendered images of separate frames that show the progressive motion of the human figures towards the dedicated building exits.

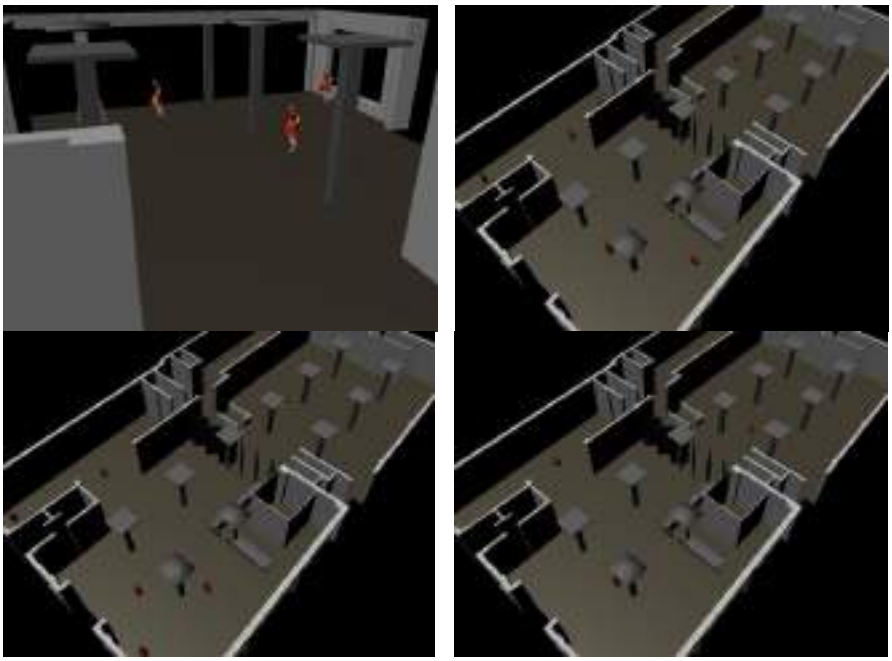


Fig. 15. SAT building evacuation simulation in Maya

Evacuation Example 2: Copenhagen Zoo's new Elephant House

In this section we present a case study focused on analyzing the occupancy levels of the Copenhagen Zoo's New Elephant House. The Copenhagen Zoo is the largest cultural institution in Denmark, attracting over 1.2 million visitors a year. The New Elephant House, which has two floors, tries to create a close visual relationship between the Zoo and the park. Visitors walk in from the main entrance, move downstairs and leave the house through the exit, moving at random and following the pathway, and spending time watching the elephants. The level of occupancy of the building is important in case of needing to evacuate it.

In this case, we have used Autodesk Revit as a tool to input the building floor plan into the simulation model, and Autodesk 3Ds

Max as the visualization tool. Figure 16 shows a view of the building using Autodesk Revit.

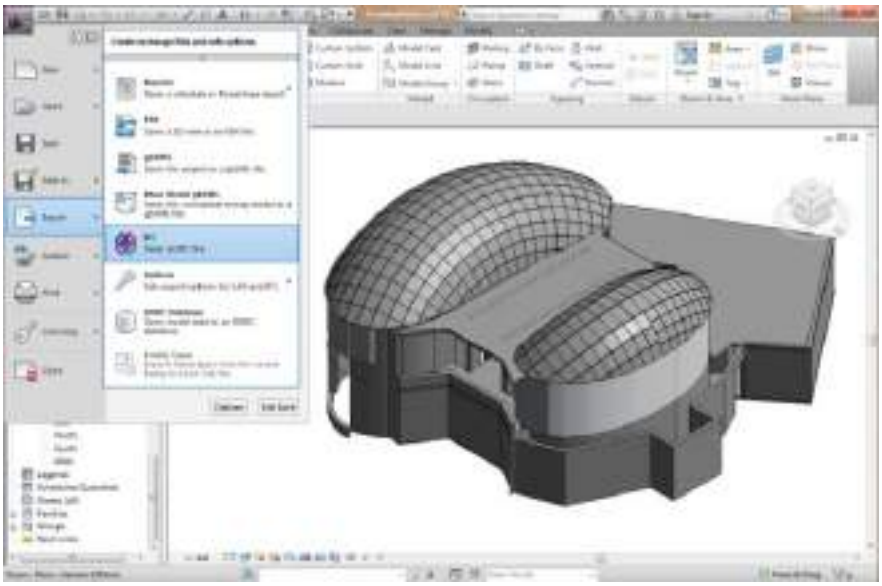


Fig. 16. Copenhagen's New Elephant House in Revit

We used Cell-DEVS to simulate the behavior of the level of occupancy of the building. Each floor uses 10x22 cells, and each cell represents a square place associated with physical horizontal coordinates. The 2 floors are connected through stairwells. Each individual on each cell uses different state variables to represent the movement:

- **Movement**: it defines the current position and the relation to the different phases, defined below
- **Phase**: each movement cycle goes through four phases (Intent, Grant, Wait, Move), to be discussed in detail later
- **Pathway**: Visitors tend to move following the pathway with certain probabilities. Normally, the pathway points to the shortest path towards the exit. In our case, we overlay a Voronoi diagram of the route to an exit or stairwell.
- **Layout**: each cell can be an empty space, a wall, an entrance, a stairwell, an exit, etc.

- **Hot Zones**: they reflect the popularity levels of certain spots influencing different potential waiting time. The higher value the hot zone is, the higher the probability that a visitor would stay.

In order to implement random movement and random waiting, the movement behavior is divided into four phases (*Intent*, *Grant*, *Wait*, and *Move*).

In general, for an occupied cell, a visitor chooses a direction at random during the *intent* phase. If the target cell is available, the visitor state changes to get *grant*; otherwise, it turns to get *rejected*. If granted, the visitor would *wait* for some time at random (according to the hot zone where the visitor is), and then *empty* the cell at the *move* phase. If rejected, the visitor needs to wait. For an *empty* cell, the logic is simpler: it chooses a surrounding cell, which is in the *grant* phase, and changes to *occupied* at the *move* phase.

```
rule : {~movement := 2; ~phase := 1;} 0 {uniform(0,1) <
  #VisitorRate and remainder(time, 4)=1 and (0,0,0)~phase = 0 and
     (0,0,0)~movement=0 and $layout=3}
```

At the beginning of the simulation, visitors go to the main entrances with certain probability (*VisitorRate*), in order to mimic different input flow rates with rush/slash hour during the opening time. In the current implementation, each cycle has 4s (each phase has 1s). We check to see if it is the beginning of the cycle *(remainder (time, 4) =0)*, and then generate a new individual.

During the *intent* phase, the desired direction is determined using the pathway direction and probability.

```
rule : {~movement := 10; ~phase := 2;} 1 { (0,0,0)~phase = 1 and
           (0,0,0)~movement = 1 and $layout = 5}
rule : {~movement := uniform(0,1); ~phase := 1.1;} 0 {
(0,0,0)~phase = 1 and (0,0,0)~movement=1 and (0,0,0)~pathway>=5}
...
rule : {~movement := 11; ~phase := 2;} 0 (0,0,0)~phase = 1.1 and
   (0,0,0)~pathway = 5 and (0,0,0)~movement > 0.0 and
       (0,0,0)~movement <= #Front }
...
```

```
rule : {~movement := 18; ~phase := 2;} 0 (0,0,0)~phase = 1.1 and
(0,0,0)~pathway =8 and (0,0,0)~movement > #Front + #Left-Front and
     (0,0,0)~movement <= #Front+...+#Right-Front}
```

We first check if the individual it is in the intent phase, if the cell is a stair, and if the cell below is empty (intent direction=10). If the cell is not a stair, we find the probability to move in different directions. We then generate a random number between 0 and 100, and check in which direction that the random number is located. Finally, last, the cell value changes to 10-18, whose unit value corresponds with the intent direction: D(0), E(1), NE(2), N(3), NW(4), W(5), SW(6), S(7), SE(8). E.g., for going up, it should be 13. Note here we do not care whether the target cell is available, it will be checked in the following phases.

After choosing the intended direction, we need to handle collisions (i.e., to see if more than one person want to enter into the same cell). This phase, called *grant*, is used to choose only one agent to move to a neighbor cell. To do so, it checks neighbors in the *intent* phase, and will mark one of the eight reverse directions (i.e., 41 means the current cell accepts the left neighbor to come in). The cells with *intent* direction 10-18 change to 20-28 and phase 3 (*waiting*). The rules for the Grant phase are like as follows:

```
rule : {~movement := 40; ~phase := 4;} 1 { (0,0,0)~movement = 0
            and (0,0,-1)~movement = 10 }
rule : {~movement := 41; ~phase := 4;} 1 { (0,0,0)~movement = 0
            and (0,-1,0)~movement = 11 and $layout != 2} ...
rule : {~movement := 48; ~phase := 4;} 1 { (0,0,0)~movement = 0
            and (-1,-1,0)~movement = 18 and $layout != 2}
rule:{~movement := ((0,0,0)~movement+10); ~phase := 3;} 1 {
            (0,0,0)~movement >= 10 and (0,0,0)~movement <= 18 }
```

The *wait* phase defines a random wait. If a person is *granted* to move, they wait for a random amount of time before moving, based on the hot zone where the person is. We implement this by adding different delays in the associated rules, as follows:

```
rule : {~movement := 30; ~phase := 4;} 1 { (0,0,0)~phase = 3 and
            (0,0,0)~movement = 20 and (0,0,1)~movement = 40 }
rule : {~movement := 31; ~phase := 4;} { 1 + 4*randInt($hotzone) }
            { (0,0,0)~phase = 3 and (0,0,0)~movement = 21 and
                (0,1,0)~movement = 41 }
```

```
rule : {~movement := 38; ~phase := 4;} { 1 + 4*randInt($hotzone) }
            { (0,0,0)~phase = 3 and (0,0,0)~movement = 28 and
                (1,1,0)~movement = 48 }
rule : {~movement := 39; ~phase := 4;} 1 { (0,0,0)~phase = 3 and
                (0,0,0)~movement >=20 and (0,0,0)~movement <= 28 }
```

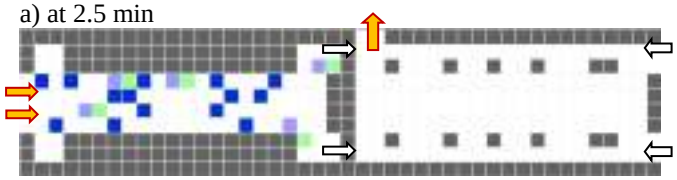
Now, every individual that intended to move, has a value of 30-38 (the movement was *granted*) or 39 (the movement was *rejected*). A granted individual can move to the target cell. To finish the moving for next cycle, we empty the intended cells (value = 0), and change the rejected ones to 1

```
rule : {~movement := 0; ~phase := 1;} 100 { (0,0,0)~phase = 4 and
            (0,0,0)~movement = 30 and (0,0,1)~movement = 40 } ...
rule : {~movement := 1; ~phase := 1;} 100 { (0,0,0)~phase = 4 and
            (0,0,0)~movement = 48 and (-1,-1,0)~movement = 38}
```

As seen in the previous section, 3D visualization provides a more intuitive way to observe simulation results, enabling the designers to check the building performance and people behaviors under different properties. Most authoring tools support full-featured 3D visualization of buildings. Among them, Autodesk 3ds Max is a powerful tool for 3D animation and rendering. We have developed an advanced visualization tool in 3ds Max, providing options for hiding building floors for visibility, and filtering models. We include arrow models with key framing ability and humanoids to animate real body movement (using the Motion Mixer plugin).

The simulation results presented in Figure 17 show the basic behavior of the visitors under normal conditions. We can see the two floors in the building, visitors arriving in the building and moving around the floors (they arrive using the main doors on the left). Then they move to the first floor downstairs (following the white arrows) and leave the building. Each visitor goes through the four cycles discussed above.

a) at 2.5 min



b) at 7.5 min

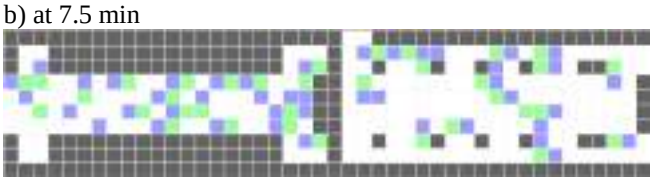


Fig. 17. Simulation results of basic properties at different simulation times.

In [17] we presented different simulation scenarios for this building, showing the impact of door location/stairs number in terms of occupancy, and simulated two modifications to the original design. It was found that arrival rate and stairs affect the occupancy level more significantly than other properties do. In order to evaluate different options, the following parameters were modified:

1) Hot zones: we decreased the probability of people waiting, representing people moving faster. The result showed that the occupancy level decreased relatively obviously, which indicates the influence of people movement speed to the occupancy.

2) Movement direction: in the simulation results showed in Figure 17, visitors have a 70% of probability to move forward. We changed this probability to 50%, giving visitors freedom to move in other directions. The differences with the original simulations were minimal. People stop to watch the exhibits longer than any influence in their moving direction, and they reach the stairs and exits as a rate similar than in the original case.

3) Arrival rate: we conducted tests with different arrival rates. When interarrival interval was longer, the simulation results showed a decrease in the occupancy of the first floor (from 38.9% to 26.7%), but only a small change in the second floor, because the flow from the first to the second floor does not change much. Nevertheless, the first floor is less congested when there are less individuals arriving.

Figure 18 shows the results of the occupancy simulation using our 3D visualization tool. As we can see, we can combine the simulation results with the original 3D floor plan in Revit, which is used to generate initial conditions for the simulation. Then, we use 3Ds Max to visualize the results of the simulation.

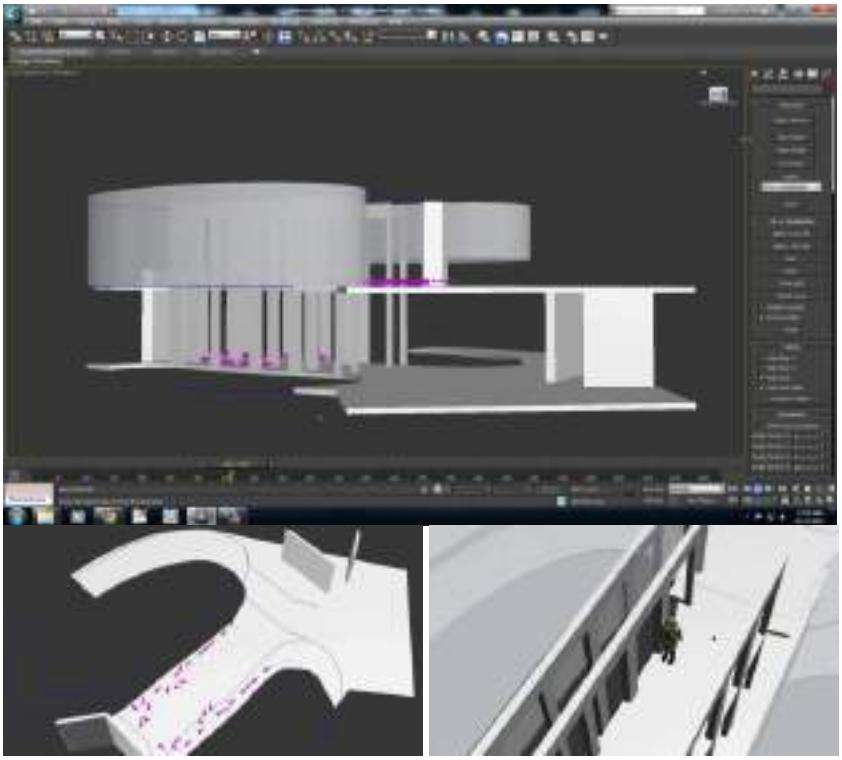


Fig. 18. Different visualization options

The figure shows some visualization results of different options. We can see the whole building and the two floors, and different individuals represented as cones in the direction of the movement of the individuals. The complete visualization of this simulation can be seen at https://www.youtube.com/watch?v=ciA5mtXdHIA.

Conclusion

Multimodeling can help builders of complex models and their simulations to organize their work better, address each of the problems at the right level of abstraction, and resolve the problems quicker and easier. Prof. Ören's invention allowed us to address these complex problems with ease. The application of multimodels into agent-based simulation provides a good combination for solving complex simu-

lation problems. Here we showed how to use these concepts combined with the DEVS formalism proposed by Prof. Zeigler, and cellular models to describe the phenomena using a spatial based notation. We showed different modeling and simulation examples focusing on evacuation and occupation of buildings. We defined a solution based on Building Information Modeling, mixing the results of buildings and simulations in Cell-DEVS. We also presented new methods to view advanced 3D visualization in 3Ds Max. We showed to different case studies: one for the SAT building in Montreal, and another one for the Copenhagen's New Elephant House.

The models are public domain and can be easily modified to be applied for other purposes. The tools can be found at http://cell-devs.sce.carleton.ca. The different models can be found at http://www.sce.carleton.ca/faculty/wainer/wbgraf.

These techniques can benefit building designers and engineers to understand better some issues related to the buildings under construction (e.g., doors location, stairs number, rush/slash hours, different movement probabilities of directions, etc.), allowing them to better manage the design and to provide suggestions for improvements.

Acknowledgements

Numerous authors participated in the research reported in this chapter (see the references for complete citation), including J. Ameghino, M. Braunstein, V. Freire, A. Khan, Q. Liu, E. Poliakov, V. Rajus, and S. Wang. This work was partially supported by NSERC, Autodesk Research, and MITACS.

References

[1] Ören T.I. "Model Update: A Model Specification Formalism with a Generalized View of Discontinuity" Proceedings of the Summer Computer Simulation Conference, Montreal, QC, Canada, 1987.

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

[3] Yilmaz, L. and Ören, T.I. “Dynamic Model Updating in Simulation with Multimodels: A Taxonomy and a Generic Agent-Based Architecture”. Proceedings of SCSC 2004 - Summer Computer Simulation Conference, San Jose, CA. 2004.

[4] Ören T.I. “Dynamic Templates and Semantic Rules for Simulation Advisors and Certifiers”. In: Knowledge-Based Simulation: Methodology and Application, P.A. Fishwick and R.B. Modjeski (Eds.). Springer-Verlag, Berlin, Heidelberg, New York, Tokyo. 1991.

[5] Ghasem-Aghaee, N. and Ören, T.I. “Towards Fuzzy Agents with Dynamic Personality for Human Behavior Simulation”. Proceedings of the 2003 Summer Computer Simulation Conference, Montreal, PQ, Canada, 2003.

[6] Jalaliana, A.; Chalupa, S.; Ostwald, M. “Architectural evaluation of simulated pedestrian spatial behavior”. Architectural Science Review. Vol. 54, Issue 2, 2011. pp. 132-140.

[7] Weifeng, F.; Lizhong, Y.; Weicheng, F. “Simulation of bi-direction pedestrian movement using a cellular automata model”. Physica A: Statistical Mechanics and its Applications, vol. 321, No. 3., pp. 633-640. 2003

[8] Vizzari, G., Bandini, S. "Studying Pedestrian and Crowd Dynamics through Integrated Analysis and Synthesis", IEEE Intelligent Systems, vol.28, no. 5, pp. 56-60, Sept.-Oct. 2013,

[9] Burks, A.W. “Von Neumann’s self-reproducing automata”. In A.W. Burks (Ed.), Essays on Cellular Automata, University of Illinois Press, Champaign, IL, pp. 3–64, 1970.

[10] S. Wolfram. “*A New Kind of Science”*. Champaign, IL: Wolfram Media. 2002.

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

[12] G. Wainer, “Discrete-Event Modeling and Simulation: a Practitioner’s approach”. CRC Press. Taylor and Francis. 2009.

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

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

[15] G. Wainer, Q. Liu. "Tools for Graphical Specification and Visualization of DEVS Models". *SIMULATION: Transactions of the Society for Modeling and Simulation International*. Vol. 85, No. 3, 131-158. 2009.

[16] “A Busy Day at the SAT Building”. E. Poliakov, G. Wainer, J. Hayes, M. Jemtrud. *Proceedings of AIS 2007, Artificial Intelligence, Simulation and Planning*. Buenos Aires, Argentina. 2007.

[17] “Occupancy Analysis using Building Information Modeling and Cell-DEVS Simulation”. S. Wang, G. Wainer, V. Rajus and R. Woodbury. *Proceedings of 2013 SCS/ACM/IEEE Symposium on Theory of Modeling and Simulation, TMS/DEVS’13*. San Diego, CA. 2013.