Visualization in 3ds Max for Cell-DEVS Models Based on Building Information Modeling

Victor Freire[1], Sixuan Wang[2], and Gabriel Wainer[2]

[1]Dept. of Systems and Computing,
Universidade Federal de Campina Grande
Rua Aprigio Veloso, 882, Campina Grande
PB 58429-900, Brazil
victor.freire@ccc.ufcg.edu.br

[2]Dept. of Systems and Computer Engineering,
Carleton University
1125 Colonel By Dr. Ottawa
ON K1S 5B6, Canada
{swang, gwainer}@sce.carleton.ca

Keywords: DEVS, Cell-DEVS, Building Information Modeling (BIM), 3ds Max, 3D Visualization

Abstract

Building Information Modeling (BIM) increasingly benefits from modeling, simulation and visualization techniques. 3D visualization can provide a better way to obtain visual simulation results in BIM authoring tools. In this paper, we focus on improving interoperability, traceability, reusability and visibility of 3D visualization. We employ the DEVS (Discrete Event Systems Specification) formalism and its cellular extension Cell-DEVS, providing a method for visualizing Cell-DEVS models based on moving entities. We used this technique to develop a 3ds Max visualization plug-in for Cell-DEVS models based on BIM. This tool can show different animation models and allows designers to filter the building for visibility. We also show two case studies applying this tool for evacuation and occupancy simulation.

1. INTRODUCTION

Building Information Modeling (BIM) plays a significant role in the whole design lifecycle in the Architecture, Engineering, and Construction (AEC) industry. BIM is a modern way to create, document, manage and exchange information about buildings. BIM uses 3D real-time object-oriented building modeling to achieve accurate AEC projects with minimized costs, improving the way architects-contractors and fabricators work (Hardin 2009). BIM authoring tools capture a rich set of digital data, including 3D geometry (instead of 2D), properties and relationships between objects.

Modeling and simulation (M&S) has been employed in BIM applications to analyze and evaluate the performance of building designs using an iterative process with several cycles through alternative simulations (Jiang et al. 2012). Each cycle can be divided into small sub-phases (e.g. pre-design, main design, detail design). In order to evaluate the performance or find the optimal solution, at the end of each sub-phase, the designers test, analyze and refine their solutions with different simulation scenarios. Related simulation tools cover different domains, such as energy consumption, CO2 emissions, occupancy management, and evacuation planning (Wang et al. 2012).

We are interested in the interoperability of BIM and generic simulation models. In particular, we are exploring the use of the Discrete Event Systems Specification (DEVS) formalism, a popular systems theoretical approach that allows the definition of hierarchical modular models, composed by behavioral (atomic) and structural (coupled) components (Zeigler et al. 2000). Also, Cell-DEVS, which extended DEVS for modeling discrete-event spatial systems (Wainer 2009), can be used with this purpose. Here, we focus on generic 3D visualization in Autodesk 3ds Max for Cell-DEVS models. The provided tool can get information from IFC file automatically, parse the simulation results into 3ds Max and provide different options in our GUI. This tool allows designers to hide different building floors and filter models for visibility, as well as different animation features of models (the realistic model for real body movement and the arrow model for pointing the moving direction).

The following sections will present the results of this effort. We first introduce the related work about visualization techniques. Then, we present the general idea for visualizing Cell-DEVS models based on moving entities. The following section describes our implementation. We employed this tool in two case studies: Building Evacuation and Occupation model of Copenhagen House. The results have shown its higher reusability and extensibility. This kind of application brings better visualization of simulation results, enabling the designers to check building

performance, compare between different solutions and find flaws for future improvement.

2. RELATED WORK

Modeling, Simulation and Visualization has been employed in BIM applications to analyze and evaluate the performance of building designs using an iterative process with several cycles through alternative simulations (Jiang et al. 2012). 3D visualization provides a more intuitive way to obtain visual simulation results in BIM authoring tools. From 3D visualization, at the end of each sub-phase, the designers can observe, analyze and refine their solutions with different simulation scenarios. Numerous related simulation and visualization tools cover different domains, such as indoor climate, energy consumption, CO2 emissions, occupancy management, and evacuation planning (Wang et al. 2012). However, most of the tools support limited 3D visualization using BIM or Cell-DEVS, in terms of interoperability with IFC, traceability of data, reusability of model and visibility options of GUI.

There are a number of applications in BIM using spatial lattice to analyze the quality and performance of building designs during the AEC projects life cycle. For example, a space representation of work zones for resources management was represented with small grid units that take irregular shapes (Elbeltagi et al. 2004). A BIM-ISEE (Immersive Safety Engineering Environment) was developed, focused on the realistic virtualization in their immersive environment for integrating fire and evacuation simulation with BIM tools (Ruppel and Abolghasemzadeh 2009).

Advanced 3D visualization for simulation has attracted much attention. Bijl and Boer (2011) provided a 3D visualization tool for discrete event simulation using game technology. In (Guo et al. 2012), the authors proposed a Flight Data 3D visualization for an entire fight quality analysis. In (Zhange et al. 2010), they represented a platform of tunnel construction process in a distributed simulation environment. However, these advanced techniques are not applied in BIM authoring tools.

In recent years, different AEC projects with DEVS and Cell-DEVS have been developed. Cell-DEVS models were introduced for construction sites, to deal with the construction performance, conflict analysis and visualization of work site (Hammad and Zhang 2011). A Cell-DEVS simulation of Diffusion Limited Aggregation (DLA) was used to model the growth of mold in building walls, using Autodesk Revit for Data collection (Ahmed et al. 2010). An integrated framework was introduced for combining BIM and Cell-DEVS, in which a simulation of evacuation planning in a multi-floor building was used (Wang et al. 2012). However, the tool mentioned above usually supports well 1D or 2D visualization, with limited 3D visualization.

Besides, there are other efforts to 3D visualization focusing on Cellular Automata (CA). Capow (Rucker and Ostrov 1997) is a program for visualizing CAs. The user can control color, selecting view type and 3D view details. CASim (Freiwald et al. 2001) is an environment for simulating 3D cellular automata, allowing users to design states, transition rules, colors and icons. However, most of them only enable displaying the surface of the 3D graph, which cannot present realistic moving entities.

3. VISUALIZING CELL-DEVS MODELS IN 3D

In this section we show how to interface moving entities and Cell-DEVS models. The key idea in tracking the entities is to keep a record of the direction and position of each entity while parsing the simulation results. When a cell becomes empty, the position of the entity is set as the cell its direction was pointing to. This information can be extracted from the simulation. The visualization system extracts the *position* and *state* of a cell at a certain *time*. The following UML activity diagram (Figure 1) shows the overall architecture of the proposed solution.

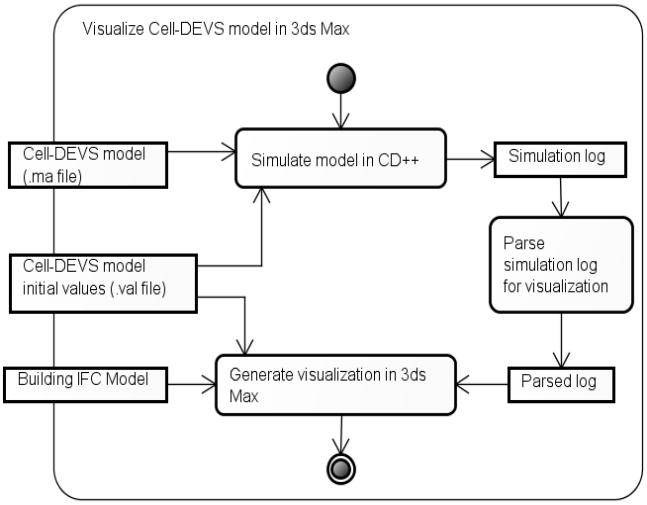

Figure 1. Activity diagram of the implementation

First, the Cell-DEVS model is simulated in CD++ and a simulation log obtained. A parsed log is provided to the 3ds Max visualization plug-in along with the file containing the

initial values of the Cell-DEVS model and the IFC model of the building.

The simulation is parsed before being passed to the visualization plug-in. This preprocessing phase facilitates the reuse of the visualization plug-in code, simplifies the plug-in code and speeds up the generation of visualization data. Reusability is accomplished by hiding details of the Cell-DEVS model from the visualization plug-in. (In the case studies to be presented in the following section, the sizes of the parsed logs were less than 5% of the original ones.) The parser extracts from the simulation log the position and state of each cell at every simulation time.

In order to generate the visualization, the plug-in needs to know the size and boundaries of the building, the number of cells in the model and the size of each cell. This information is extracted from the building model in the IFC format during the creation of the Cell-DEVS simulation. We chose to use the IFC standard to represent the building, because most BIM authoring tools support these files for collaborative development and interoperability, and it facilitates the extraction of the needed information. Figure 2 shows a part of the IFC-EXPRESS standard (buildingSMART 2012) focusing on the entity *IfcWall*, which represents the walls of the building. *IfcWall* contains the coordinate information needed to determine the size and boundaries of the building, the number of cells in the model and the size of each cell.

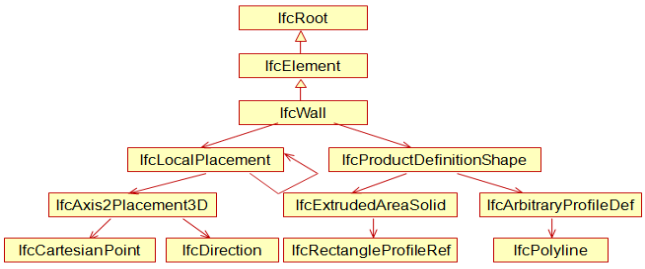

Figure 2. *IfcWall* in the IFC hierarchy

To collect this information, we used the open API from BIMServer.org for querying that the needed IFC elements are used (BIMServer.org 2012). Using this, we can obtain (*Xmin*, *Ymin)* and (*Xmax*, *Ymax)*, the points which constitute the bounding box of the building. Based on this box, the designers can define the size of each cell and the number of cells in each dimension.

4. IMPLEMENTING A VISUALIZATION PLUG-IN

3ds Max supports building plug-ins written in C++ or in MAXScript (a native programming language). The performance of MAXScript is lower than C++, however it is more productive and it enables faster prototyping since it is more abstract than C++ (Autodesk 2012a).

Figure 3 shows a UML class diagram of the overall architecture of the plug-in, which is based on the Model-View-Controller design pattern (Krasner and Pope 1988). The view is implemented by the *3DGui* class and 3ds Max objects; the controller is *CDVizController*, and the remaining classes are the model. Some of the model's logic is inside *CDVizController*, and it could be refactored in the future to improve understanding of the software.

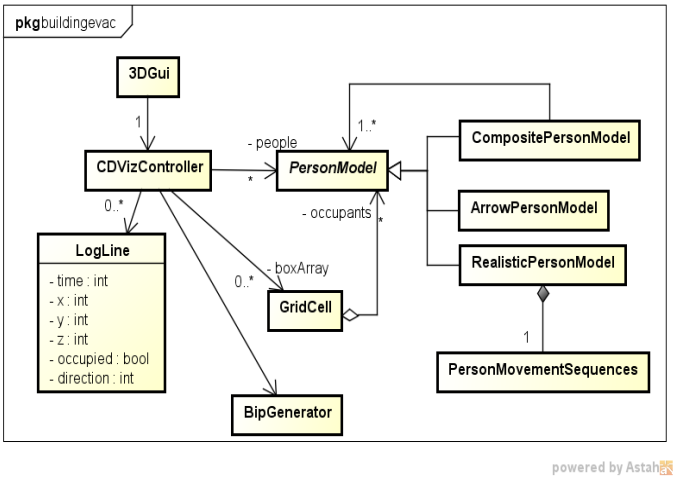

Figure 3. Class diagram of the visualization plug-in

The *LogLine* class represents a line from the parsed log and it is responsible for extracting the data from that line. *CDVizController* processes the simulation and, at each simulation timestep, sets the position of all people according to the log. The simulation grid is represented by an array of *GridCell* objects, which keep track of which person is in that cell, and each person in the simulation is represented by an instance of *PersonModel*. *GridCell* behaves like a queue, and it reports that the person at the top of the queue is the one occupying the cell. An array of *GridCell*s is used instead to prevent data being lost during the processing of the simulation (since the log is processed sequentially).

In order to allow a person to be represented graphically in one or more different ways simultaneously while abstracting this from the rest of the program, the Composite pattern was used in *PersonModel* (Gamma et al 1995). Our

implementation allows people to be represented as cones and/or 3D models that look like people.

4.1. Cone models

Cone models are implemented in the class *ArrowPersonModel*, and they allow a user to see a person's position and direction. The plug-in makes use of 3ds Max key framing ability for animations; the position and direction of the cone are set each time they change, and 3dsmax interpolates all the data producing a smooth transition between the key frames.

4.2. Realistic models

Realistic 3D person models are implemented in the class *RealisticPersonModel*. This graphical representation is not as straightforward to implement as cones and a rigged 3D model and movement animation files are needed. The *Motion Mixer* component of 3ds Max (Autodesk 2012b) was used to mix all the animations for each person in the visualization. In the Mixer, we define when each animation occurs and the order of the animations (the interaction with the Motion Mixer API is in the *PersonMovementSequences* class).

The most important feature of the Mixer for this application is the interpolation of positions between animations. Each instance of a walking animation must make the character start moving from its current position. Without interpolation, if a walking forward animation clip A is added, followed by clip B, clip B will start the animation from the starting position of the character (and not the position at the end of A). The plug-in inserts an empty animation file in the Mixer when gaps are needed.

We create an on-the-fly version of each animation for each floor, each with the Z coordinate of the corresponding floor. The Z positions of all characters are set to the Z of the top floor and then, according to the floor where the character is in, a different version of the animation is added to the Mixer. The solution is implemented by the *BipGenerator* (the cost of generating these animations increases with the number of floors of the building).

4.3. Visibility selection

The plug-in GUI lets the user select which parts of the building are shown, so that the user can have a better view. This is implemented in the *CDVizController* class.

Hiding and revealing the building constituent objects is performed by applying the *hide* or *unhide* MAXScript functions on a range of objects that fall within a certain boundary. For instance, to hide a floor, the *hide* function is applied on all objects that lie between the floor to be hidden and the next floor above. The problem of using the *hide* and *unhide* functions for is that they do not consider 3ds Max animation time. For example, if the current animation time is X and *hide* is applied on a certain object Y, this object is hidden for all the animation (not just at or after time X). If Y is a person on the second floor at time X and on the first floor at time X + 100, and the second floor was hidden by a call to *hide* at time X, then if 3ds Max is set to show time X + 100, Y will still be hidden. The consequence is that the user has to click the *Refresh Visibility* when changing the current displayed animation time to see the objects.

4.4. Grid coordinates conversion

To convert the Cell-DEVS coordinates of a cell in the simulation grid to the corresponding coordinates in the building model, an affine transformation matrix is used. This was done by first applying a scaling transformation, and then a coordinate system transformation to the grid rectangle. The scaling transformation matrix is calculated as follows (Shirley et al. 2009):

$$\begin{pmatrix} \frac{c1-c0}{a1-a0} & 0 & 0 & 0 \\ 0 & \frac{d1-d0}{b1-b0} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Here *(a0, b0)* is the bottom-left corner of the grid rectangle, *(a1, b1)* is the top-right corner of the grid rectangle, *(c0, d0)* is the bottom-left corner of the building model and *(c1, d1)* is the top-right corner of the building model. Also, *(c0, d0)* is *(Xmin, Ymin)*; *(c1, d1)* is *(Xmax, Ymax)*, *a1 – a0* is *CellsX* and *b1 – b0* is *CellsY*. Therefore, the 4x4 scaling transformation matrix becomes:

$$\begin{pmatrix} \frac{X_{MAX}-Xmin}{CellsX} & 0 & 0 & 0 \\ 0 & \frac{Ymax-Ymin}{CellsY} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The coordinate system transformation matrix depends on the coordinate system of the building model and examples of it can be seen in the Case Study section. The final transformation matrix for converting grid coordinates to building model coordinates is given by multiplying the

scaling transformation matrix and the coordinate transformation matrix. This matrix is generated once in the *CDVizController* class before processing the parsed simulation log.

5. CASE STUDIES

5.1. Building Evacuation

The visualization system was applied to a Cell-DEVS model of the occupancy of a building model with three floors. The objective of the model is to help determine the maximum occupancy for the building that allows a reasonable emergency evacuation time (Wang et al. 2012).

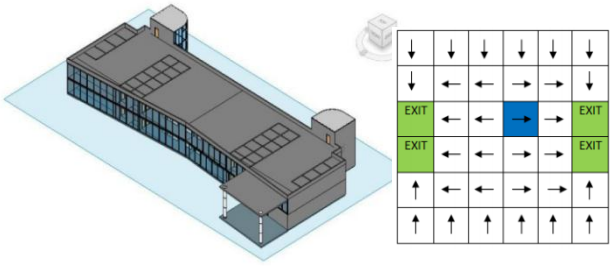

Figure 4. (a) Building for evacuation and (b) pathway

The building (shown in Figure 4a) contains three floors connected by two stairwells. There are exit doors on the first floor (in the lower right corner). In the case of emergencies, people try to evacuate along the pathway on each floor (Figure 4b), go down the stairwell, and finally get out the building through exits. This multi-level evacuation Cell-DEVS model is defined in CD++. The behavior of each cell depends on its current state whose value is determined by a set of rules after satisfying a precondition of his neighborhood. It contains different groups of rules (describes in Wang et al. 2012). For instance, to model someone entering into an unoccupied cell the rule is as follows:

rule : 4 100 { (0,0,0) = 3 and ((0,1,0) = 10 or (-1,0,0) = 4 or (0,-1,0) = 6 or (-1,0,0) = 14 or (1,0,0) = 14 or (0,1,0) = 14 or (0,-1,0) = 14)}

The parser for this model extracts the following properties from each simulation log: the *time* a change occurred, the *position* of the cell if it is now *occupied* or empty (a 3D tuple [x, y, z]) and its *direction*. Then, it builds a human-readable text line with this using the following format:

time position_x position_y position_z occupied direction

All duplicate lines are removed, and the remaining lines are sorted according to its time and outputted. The GUI allows the user to select which floors will be displayed. To obtain the height of each floor, the plug-in iterates through all the objects whose name starts with the prefix "floors" and adds their height to a list that does not allow duplicate values. The height of a floor is calculated by adding the Z coordinate of the position of the floor to the height of its bounding box. After the list is filled, it is sorted and then it contains the height of each floor in a bottom to top order.

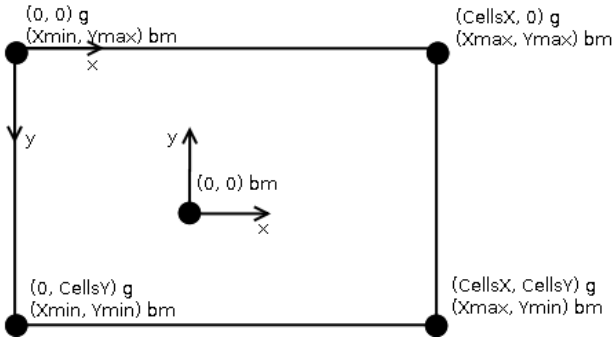

Figure 5. Diagram showing how the two coordinate systems differ.

The grid coordinate system uses the Cartesian system with an inverted Y axis in relation to the building model coordinate system. The diagram below shows how the two coordinate systems differ. The notation *(x, y) g* represents a point in the grid coordinate system and *(x, y) bm* represents a coordinate in the building model coordinate system. The rectangle represents the boundaries of the building, as seen in Figure 5.

The following matrix was used to convert from the grid coordinate system to the building model coordinate system:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Xmin & Ymax & 0 & 1 \end{pmatrix}$$

In the test machine (Intel Quad Core Q8200, 8GB DDR3, Geforce GTX 560 Ti 448), the realistic visualization took about 1 minute to load and the average visualization playback rate was 0.5 frames/s. Using cones in the same machine, the visualization takes on average 15 seconds to load and a maximum playback rate defined for the visualization of 10 frames/s is achieved.

5.2. Occupation model of Foster's Elephant House

The visualization system was applied to a Cell-DEVS model of the occupancy of Foster's Elephant House at the Copenhagen Zoo (Figure 6a). The objective of the model is to bring designers to understand different building

properties in order to facilitate the occupancy management or design suggestions for future improvement (e.g., door locations, number of stairs, rush/slash hours (Wang et al. 2013)).

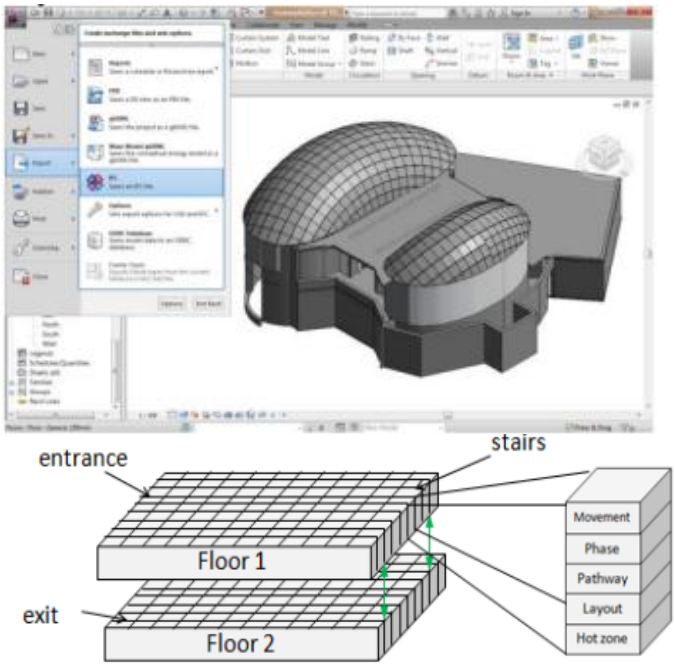

Figure 6. (a) Elephant House Building and (b) Cell states

This occupancy model simulates people's behavior for occupancy analysis. People walk in from the main *entrance* on *floor1*, go downstairs to *floor2* and then leave the house through the *exit*. People can move in different directions (which is simulated using a random selection) and they wait randomly when visiting in the two floors of this building.

This occupancy Cell-DEVS model is defined in CD++, similarly as in the model in Section 5.1. The cells of this building change their values according to a set of rules. As shown in Figure 6b, each cell has five state variables: *Movement*, *Phase*, *Pathway*, *Layout*, and *Hotzone*. The most important variable is *Phase*, which is divided into four sub-phases: *intend, grant, wait* and *move*. They occur sequentially in that order every 400ms: e.g., if an *intend* phase happens at time X, *grant* happens at time X+100, *wait* phase at time X+200, *move* phase at time X+ 300, the *intend* phase at X+400 and so on. The time is used to identify to which of these phases the event belongs to. For an occupied cell, a visitor chooses a direction randomly at the *intend* phase. If the target cell accepts it, it changes to *get grant*; otherwise, it turns to *get rejected*. If granted, the visitor would *wait* for some time randomly according to the hot zone where the visitor is standing at, and then *empty* the cell at the *move* phase. For the purpose of visualization, the only relevant information comes from the *intend* and *move* phases.

One rule in the *intend* phase is used to choose the intended direction of an occupied cell at random, and is defined as follows (Wang et al. 2013):

rule : {~movement := uniform(0,1); ~phase := 1.1}
100 { (0,0,0)~phase = 1 and (0,0,0)~movement=1 and (0,0,0)~pathway>=5}

The events of an intend-grant-wait-move sequence actually happen at the same time but are separate in simulation time for ease of modeling. The parser groups each intend-grant-wait-move sequence. The parser output will denote that events from intend phase at time X and events from move phase at time X+300 both happened at time:

$$Y = \left\lfloor \frac{X-1}{400} \right\rfloor * 100.$$

For instance, events from intend phase at 500 and move phase at 800 will be outputted as having occurred at time 100.

The parser output lines, which denote events from *intend* and *move* phases, are in the following format:

- *Intend* event:
  real_time x y z direction
- *Move* event:
  real_time x y z "m"

The lines are ordered first in increasing order by real_time and secondly by type with *intend* events coming before *move* events since the direction of the person must be set before it moves.

The GUI allows the user to hide each floor, the roof, and the side walls. The height of each floor is obtained as described in the previous case study and the position of the side walls is obtained manually and typed into the plug-in code.

The grid coordinate system is rotated by 90 degrees and transposed by $(Xmax, Ymin)$ in relation to the building model coordinate system (Figure 7):

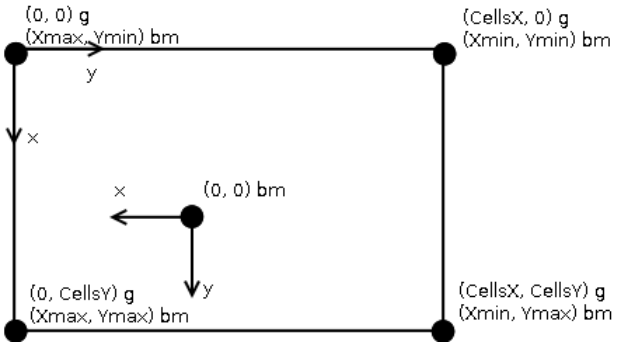

Figure 7. Diagram showing how the two coordinate systems differ.

Thus, the following matrix was used to convert from the grid coordinate system to the building model coordinate system:

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Xmax & Ymin & 0 & 1 \end{pmatrix}$$

The plug-in helped locate several bugs in the Cell-DEVS model, as the 3D visualization is an effective way to find errors in these types of Cell-DEVS models. These errors were detected because the plug-in works with people not cells, therefore assumptions about the behavior are made: e.g., when a cell that does not represent an exit becomes empty, it means the person who was in that cell is now in an adjacent cell. Examples of the types of errors detected were people disappearing from the simulation and people being duplicated.

These errors would manifest as 3ds Max error messages while the plug-in generated the visualization, or as visual errors with people models being displayed in incorrect places. The stack trace shown in 3ds Max error messages would pinpoint the time where the simulation was behaving wrongly and for the visual errors the name of the person model would show the time when it was created.

The performance of realistic models for this case study was very similar to the previous case study. A snapshot of the simulation visualization can be seen in Figure 8.

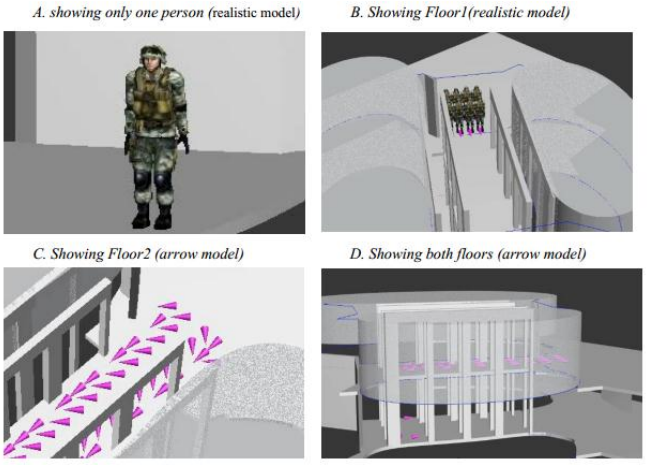

Figure 8. Screenshots of the visualization plug-in for the model of Foster's Elephant House.

6. CONCLUSION

The case studies provided evidence that the tool does solve the issues present in similar tools discussed in Section 2. The tool in both case studies: showed people moving continuously from one cell to another; was integrated into a BIM authoring tool (Autodesk 3ds Max 2012); was easily adapted to each model; and displayed specific models (cone and realistic models) that allowed the user to see easily extra information about each person, namely, where they were facing and intending to move to.

The reusability of the tool could be further improved if a uniform interface between the parser and the visualization plug-in was created. With this interface, the plug-in would not need to be changed for different underlying Cell-DEVS models. That is, only the parser would need to be changed for different applications of BIM.

The Elephant House case study suggests 3D visualization can be effective in verifying Cell-DEVS models based on moving entities. Hence, this could also be a subject for future research.

Since the performance of the realistic models was found to be unsatisfactory in the test machine, improving it in future work would be helpful to users of the tool.

Acknowledgements

We would like to thank Vinu Subashini Rajus for creating the building model used in the case study of Foster's Elephant House and the company Mixamo for making a rigged 3D person model freely available on: http://www.turbosquid.com/3d-models/free-fbx-model-soldier-military-character-rigged/516949.

References

Autodesk 2012, MAXScript Help: The SDK and MAXScript, http://docs.autodesk.com/3DSMAX/15/ENU/MAXScript-Help/index.html?url=files/GUID-C0F38270-F5DD-4356-AFFF-B037D97F0017.htm,topicNumber=d30e3671. Accessed December 14th, 2012.

Autodesk 2012, 3ds Max Reference: Motion Mixer http://download.autodesk.com/us/3dsmax/2012help/index.html?url=files/GUID-A4F4A6A6-27A9-482C-9A95-853B1ADC68C-1251.htm,topicNumber=d28e256342. Accessed December 14th, 2012.

Bijl, J. L., and Boer, C. A. 2011. Advanced 3D visualization for simulation using game technology. In Simulation Conference (WSC), Proceedings of the 2011 Winter (pp. 2810-2821). IEEE.

BIMServer.org. 2012. http://bimserver.org/buildingSMART. Accessed Nov 6. 2012.

IFC2x3. http://www.buildingsmart-tech.org/ifc/IFC2x3/ Accessed Nov 6 2012.

Elbeltagi, E., Hegazy, T., and Eldosouky, A. 2004. Dynamic Layout of Construction Temporary Facilities Considering Safety. Journal of Construction Engineering and Management. 130(4):534-541

Freiwald, U., Weimar, J. 2001. JCASim - a Java system for Simulating Cellular Automata, Theoretical & Practical Issues on Cellular Automata (ACRI 2000), S. Bandini and T.Worsch (eds.), Springer Verlag, London.

Friendly, M., and Denis, D. J.. 2012. Milestones in the history of thematic cartography, statistical graphics, and data visualization. Accessed Nov 6.

Gamma, E., Helm, R., Johnson, R., AND Vlissides, J. M. 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. ISBN 0-201-63361-2. pp. 395

Guo, H., Pang, J., Han, L., and Shan, Z. 2012. Flight Data Visualization for Simulation & Evaluation: A General Framework. InComputational Intelligence and Design (ISCID), 2012 Fifth International Symposium on (Vol. 1, pp. 497-502). IEEE.

Hammad, A., and Zhang., C. 2011. Towards Real-time Simulation of Construction Activities Considering Spatio-temporal Resolution Requirements for Improving Safety and Productivity. In Proceedings of the 2011 WSC, AZ. 3533-3544.

Hardin, B. 2009. BIM and Construction Management: Proven Tools, Methods, and Workflows. Wiley.

Jiang, Y., Ming, J., Wu, D., Yen, J., Mitra, P., Messner, J. I., and Leicht, R. 2012. BIM Server Requirements to Support the Energy Efficient Building Lifecycle. In Proc. 2012 ASCE international conference on computing in civil engineering.

Khan, A., Wainer, G. A. 2005 Advanced Visualization of DEVS and Cell-DEVS Models in CD++/Maya. Proceedings of SISO Fall Interoperability Workshop, San Diego, CA. U.S.A - 2005

Krasner, G. E., Pope, S. T. 1988, A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System, http://www.itu.dk/courses/VOP/E2005/VOP2005E/8_mvc_krasner_and_pope.pdf. Accessed Nov 6, 2012.

Malinowsky, B., and Kastner, W. 2010. Integrating Process Communication in Building Information Models with IFC and LON. 8th IEEE International Workshop on Factory Communication Systems.

Rucker, R., Ostrov, D. 1997. Continuous-Valued Cellular Automata for Non-Linear Wave Equations, Complex Systems 10 (1196) 91-119.

Ruppel, U., and Abolghasemzadeh, P. 2009. BIM-based Immersive Evacuation Simulations. In 18th In-ternational Conference on the Application of Computer Science and Mathematics in Architecture and Civil Engineering. Weimar, Germany.

Shirley, P., Marschner, S., Ashikhmin, M., Gleicher, M., Hoffman, N., Johnson, G., Munzner, T., Reinhard, E., Sung, K., Thompson, W. B., Willemsen, P., Wyvill, B. 2009. Fundamentals of Computer Graphics. AK Peters 3rd ed ISBN 978-1-56881-469-8. pp.132

Wainer, G., 2009. Discrete-event Modeling and Simulation: a Practitioner's Approach. CRC/Taylor & Francis.

Wainer, G., Poliakov, E., Hayes, J., and Jemtrud, M. 2007. A Busy day at the SAT building. In Proceedings of the International Modeling and Simulation Multiconference Buenos Aires, Argentina.

Wang, S., Schyndel, M.V., Wainer, G., Subashini, V., and Woodbury, R. 2012. DEVS-based Building Information Modeling AND Simulation for Emergency Evacuation.bIn Proceedings of the 2012 Winter Simulation Conference. Berlin, Germany. IEEE.

Wang, S., Wainer, G., Rajus, V.S., and Woodbury, R. 2013 (accepted). Occupancy Analysis Using Building Information Modeling and Cell-DEVS Simulation. Symposium on Theory of Modeling and Simulation. TMS'13. San Diego, USA

Venhola, W., and Wainer, G. 2006. DEVSView: A tool for visualizing CD++ simulation models. SIMULATION SERIES, 38(1), 133.

Zeigler, B.P., H. Praehofer, and T.G. Kim. 2000. Theory of Modeling and Simulation. Academic Press

Zhang, Y., Moghani, E., AbouRizk, S. M. and Fernando, S. 2010. 3D CAD modeling and visualization of the tunnel construction process in a distributed simulation environment. In Simulation Conference (WSC), Proceedings of the 2010 Winter (pp. 3189-3200). IEEE.