An Enhanced Indoor Pedestrian Model Supporting Spatial DBMSs

Suyeong Kwak Hyunwoo Nam Chulmin Jun*

Dept. of Geoinformatics, University of Seoul, Seoul, Korea

{ksykk0, nhw612, cmjun}@uos.ac.kr (*: corresponding author)

ABSTRACT

Two-dimensional geographic information systems (GISs) are mature technology and applications such as car navigation systems are commonplace. As indoor positioning techniques are developing, indoor 3D models are attracting increasing attention. However, modeling and implementing indoor 3D models applicable to real-time, client-server environments such as 2D GIS is a challenge and no working applications have yet been reported. As part of a multi-stage project that aims to build 3D indoor applications running in real-time, we are currently developing a fire evacuation system. Although not definitely required at this stage, we used a spatial DBMS as the input data instead of CAD files; the process of building floor plans and stairs is shown here. In developing the simulation model, we improved the existing 'floor field' model such that it can accommodate the visibility factor. While the previous floor field model does not capture the visibility effect, we revised the algorithm so it can give different walking speeds to pedestrians based on the level of visibility to the exits from where the pedestrians are located. We show the process of building the proposed 3D model and test the simulation system using a campus building.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS;

I.3 [**Computer Graphics**]: Applications

General Terms

Algorithms, Experimentation

Keywords

pedestrian simulation, cellular automata, 3D model, spatial DBMS, fire evacuation

ISA'10, 2 November 2010, San Jose, CA, USA

1. INTRODUCTION

Many variations of micro-scale pedestrian models have existed for decades [2, 3, 14]. However, most focus on scientific experiments and improvements rather than on relating to real-world applications. Many studies about indoor positioning techniques have recently appeared [12, 16, 23, 25], drawing our attention to their potential application in areas such as indoor navigation and fire evacuation.

In collaboration with others, we are currently working on a government project to develop 3D indoor awareness systems. This project includes developing 3D models, indoor positioning techniques, 3D extensions for database management systems (DBMSs), and applications that use these techniques. We are focusing on developing a DBMS-based indoor fire evacuation system. This project is composed of multiple stages, from the development of a stand-alone simulator to a real-time evacuation system. It is unnecessary to point out the numerous benefits of DBMSs compared to file systems. The major anticipated advantages are similar to those of 2D geographic information systems (GISs). Current GISs are mature technology in terms of providing client-server functions, as in DBMSs. For example, navigation systems concurrently access a network database in such a way that users think of the data as being accessed by one user while other applications access the same data to retrieve polygons and related information. In order to make indoor applications real-time, server-client systems, the data must reside in a database to which different applications can send queries and from which they can retrieve different subsets of data. For example, indoor navigation uses network structure and indoor location-based services (LBSs) may need additional polygons and topological and semantic information. While positioning sensors use height information as well as floor plans, 3D visualization requires more detailed data, including data about walls and windows. The evacuation system we are working on needs to retrieve 2D floor and stairs in plan from the database.

We are in the process of developing a stand-alone simulator and could have used CAD-based files [5, 17]. However, as mentioned, our aim is to make the system real-time and integrate positioning techniques later and to be one of the applications that share the same database. Thus, we chose to use a DBMS approach and will briefly introduce the process to build the data, although it is not the main topic of the paper. We will show how we used a simplified method to build a 3D structure that can model not only indoor rooms, but also multiple floors and stairs that can effectively be stored in DBMSs and can be converted to input data for the simulator. Once the data become available in the simulator, they are used as computation for simulation as well as for visualization in 2D or 3D.

The current concern is improving the pedestrian algorithm. Instead of developing a new model, we based our algorithm on an existing model developed by Kirchner et al., which is described later. While that model uses a fixed speed, we added a 'visibility' factor, considering that people having different sight lines to the exits have a differing ability to find the exits, especially in unfamiliar environments, resulting in different walking speeds. Along with data processing using a spatial DBMS, we show how to implement the visibility factor in the simulation.

2. RELATED WORK

Pedestrian models have been studied in many areas, such as network flow, traffic assignment, and simulations [1]. These models are generally categorized into macroscopic and microscopic models [6]. Macroscopic models are typically used in traffic flow optimization or way-finding approaches outdoors and use node-link structure as the base data structure. While macroscopic models ignore the differences of individuals and see them as a homogeneous group to be assigned to nodes or links, microscopic models include individual parameters (e.g., pedestrian movement speed, reaction speed, individual tendency), the interaction of pedestrians with other pedestrians, and the physical environment (e.g., walls, obstacles, smoke). Recently, two microscopic models have attracted particular attention—the social force model and the floor field model. A frequently cited social force model was advanced by Helbing et al. [7, 8, 9]. It involves mathematically calculating the forces that act on agents to determine their movement to other destinations (e.g., exits). Helbing's model considers the effects of each agent on all other agents and on the physical environment (e.g., shoulder width, expectation speed, target spot, etc.), leading to the computation of $O(n^2)$ complexity, which is unfavorable for computer-based simulations with many agents [10, 11].

On the other hand, Kirchner et al. proposed a floor field model that uses a cellular automata approach. Instead of considering all the effects of an agent on all other objects in the space, it only considers the local interaction of an agent with neighboring objects and computes the movement of an agent at each time step, choosing the next destination among adjacent cells. This makes computer simulation much more effective. In this study, we used a floor field model as our base model.

Kirchner's floor field model uses two types of fields, static and dynamic, which are numeric values the agent consults before moving [13]. A cell in the static field indicates the shortest distance to an exit. An agent is in a position to know the direction to the nearest exit based on the values of nearby cells. While the static field has fixed values computed using the physical distance, the dynamic field stores dynamically changing values indicating agents' virtual traces left as they move along their paths. Without having direct knowledge of where other agents are, it can follow other nearby agents by consulting the dynamic values.

An agent moves to an adjacent cell in each time step after consulting the probability assigned to entire cells, which is the normalization of the following score. Readers are advised to refer to the related studies [15, 18].

$$\text{Score(i)} = \exp(k_d D_i) * \exp(k_s S_i) * \zeta_i * \eta_i \quad (1)$$

, where Score(i) : the score at cell i
D_i : the value of the dynamic field in cell i
S_i : the value of the static field in cell i
k_d and k_s : scaling parameters governing the degree to which an agent is sensitive to dynamic or static field respectively
ζ_i : 0 for forbidden cells (e.g. walls, obstacles) and 1 otherwise
η_i : 0 if an agent is on the cell, and 1 otherwise.

The scaling factors in the formula mean the degree to which an agent is sensitive to static or dynamic fields; it is possible to simulate different pedestrian strategies by varying these values. For example, we can model herding behavior in panic situations by increasing sensitivity to the dynamic field.

In this study, we revised Kirchner's floor field model by adding a third field, called the 'visibility field'. The visibility field is a layer of numerical values that indicates the degree of visibility of a cell to the nearest exit. This is described in detail in the following section.

3. INTRODUCING VISIBILITY INTO THE FLOOR FIELD MODEL

The static field in the floor field model is a layer composed of the shortest distance to an exit, which is computed before the simulation begins and does not change during the simulation. The static field assumes that two different locations in a room having the same distance from the exit have the same distance value, indicating they have an identical attraction force to the exit. However, it is reasonable to expect that an agent blocked by an obstacle such as furniture or a wall has less visibility and will be in a less favorable position to find the exit than those who have better visibility. For example, as illustrated in Figure 1, although agent A and B are located the same distance from the exit, agent B, who is located behind an obstacle, cannot see the exit directly. Not only is it obvious that agent B needs to turn more corners to reach the exit, but it is foreseeable that the agent may spend more time to find the exit, especially if less familiar with the environment. We introduced this visibility field into the existing floor field model. In order for the visibility field to be implemented, a space first needs to be divided into sub-spaces depending on different levels of visibility. Each agent in these sub-spaces should have different walking speeds. These are described below.

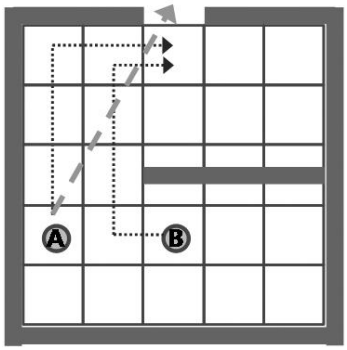


Figure 1. Two agents having different visibility to the exit

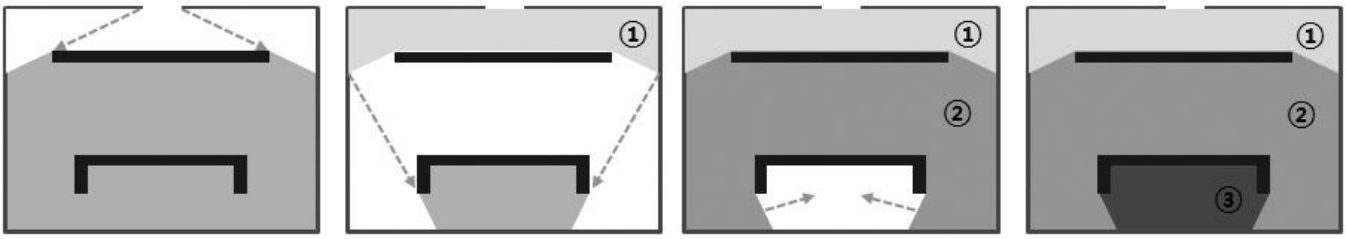


Figure 2. Space partitioning based on the levels of visibility

3.1 Space partitioning based on visibility

In partitioning a space based on visibility, we employed the space syntax theory [4]. Space syntax is a technique that has been used to derive the connectivity of urban or architectural spaces [19]. While accessibility is generally used in transportation routes to measure the relative nearness or easiness of movement using distance or travel time as the cost, space syntax only considers 'depth' as the cost and does not consider physical distance. It converts streets or corridors into a graph composed of visual paths called axial lines. When measuring the depth, it counts the number of axial lines, or turns of sight, traversed from one node to another; the greater the depth, the less structural connectivity a space has.

A traditional network model is defined using its graph *G(V, E)*, where *V* is the set of nodes defining places $\{v_i \mid i = 1, 2, \ldots, n\}$ and *E* is the set of edges or links connecting them $\{v_i, v_j \mid e_{ij}, i,j = 1, 2, \ldots, n\}$, where e_{ij} is defined as:

$$e_{ij} = \begin{cases} 1, & if\ v_i\ v_j \in E \\ 0, & otherwise \end{cases} \quad (2)$$

Here, we can define higher-order connectivity using this unit connectivity. We can count the number of paths traversed from any node *i* to any other node *j*, which is defined as S_{ij}. This index measures the depth of one node to other nodes and is in fact the basis of *the depth* in the space syntax theory. The depth S_{ij}^z where *z* is the depth of node *j* from *i* can be computed as follows:

$$Let\ S_{ij}^1 = e_{ij},\ then,$$
$$S_{ij}^z = \begin{cases} 1, if\ \sum_k S_{ik}^{z-1} e_{jk} > 0\ (k \neq i, k \neq j)\ and\ \sum_t^{z-1} S_{ij}^t = 0 \\ 0, \quad otherwise \end{cases} \quad (3)$$

Then we can define the overall depth of *j* from *i* as S_{ij} = z if $S_{ij}^z = 1$.

We applied this depth-based approach to construct the visibility field. We divided a space into sub-spaces based on the levels of visibility depths. Figure 2 illustrates how we implemented this. Let's suppose we have a room with two obstacles, as shown in Figure 2. First, the regions (①) that can be seen directly from the exit are determined. Next, those regions (②) that can be seen from region(s) ① are determined. Then, following the same method, the regions (③) are determined, which can be seen from the region(s) ②. This process continues recursively until the entire room is covered by the sub-regions. Finally, these sub-regions are assigned the depth values of 1, 2, or 3, indicating the levels of visibility to the exit. Figure 3 shows how a floor of a real building can be partitioned according to this method. It is divided into four sub-regions.

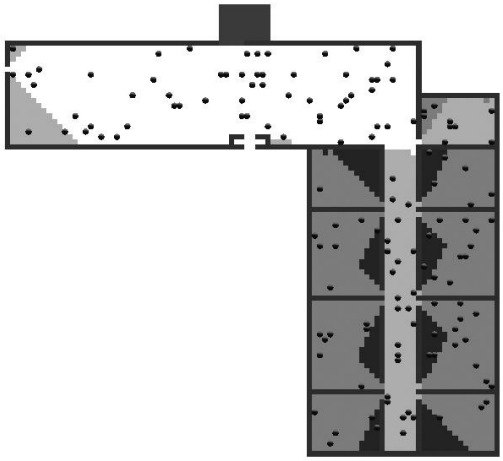


Figure 3. An example of space partitioning

3.2 Implementing the visibility field

Once the visibility field has been computed as described above, the agents consult the depth values and the static and dynamic values assigned in the cells when they move.

We implemented this by assigning different speeds according to the depths. Figure 4 shows the revised process, which includes the visibility field computation step. When implementing the walking speeds, we used time ticks, which are provided in the C# programming language. Time ticks are the waiting times until pedestrians move to the next cell. To calculate the visibility field using ticks, two kinds of parameters are required. The first is the starting value of the time tick. It is usually set to 0. The second is the increment value. The higher the value is, the longer an agent must wait before moving. Instead of moving to another cell simultaneously in every time step as in the previous model, an agent has to wait for the amount of time ticks assigned according to the visibility. Thus, those agents located in less advantageous regions in terms of visibility to the exit take longer to escape. By varying time ticks, we can simulate the differences between regions.

Figure 5 shows a comparison between the floor field model and the one with the visibility field included. It shows snapshots taken in the same elapsed time after the start of the simulation. With only the static field taken into account, the left figure shows a bottleneck on both sides of the obstacle. This indicates that the agents behind the obstacles find the way out equally quickly

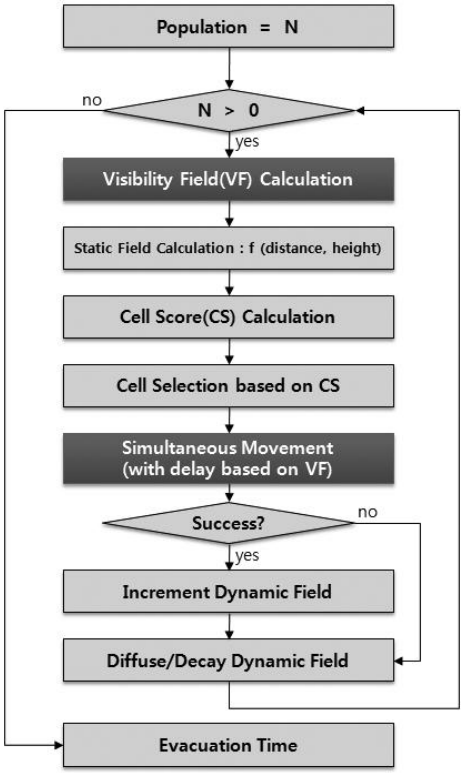


Figure 4. The simulation process including the visibility field computation

regardless of visibility once they are located the same distance from the exit. On the other hand, the figure on the right shows those agents located in higher depth areas move more slowly than those in lower depth areas.

Figure 6 shows the evacuation time of a number of agents with varying time tick increments. Increment 0 is the same as in the existing floor field model. We carried out an experiment, changing the number of agents (100, 200, 300, and 400). As shown in the table, while the existing floor field model (increment value = 0) shows little difference in evacuation time, our model with the visibility (time ticks) shows rapidly increasing time for evacuation as the number of agents increases. When a time tick increment = 3, total evacuation time (810.5 seconds) is almost four times longer than that in the existing model (205 seconds). This indicates that the people with low visibility or connectivity take more time, and the time rapidly increases with an increasing number of people. By adjusting the increment, it is possible to simulate different situations. For example, for the people who are familiar with the given space, we can use a small increment because the effect of visibility is low, and conversely, those unfamiliar with the space have a high increment value.

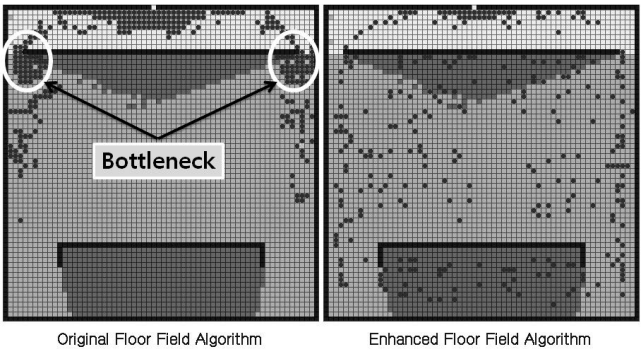
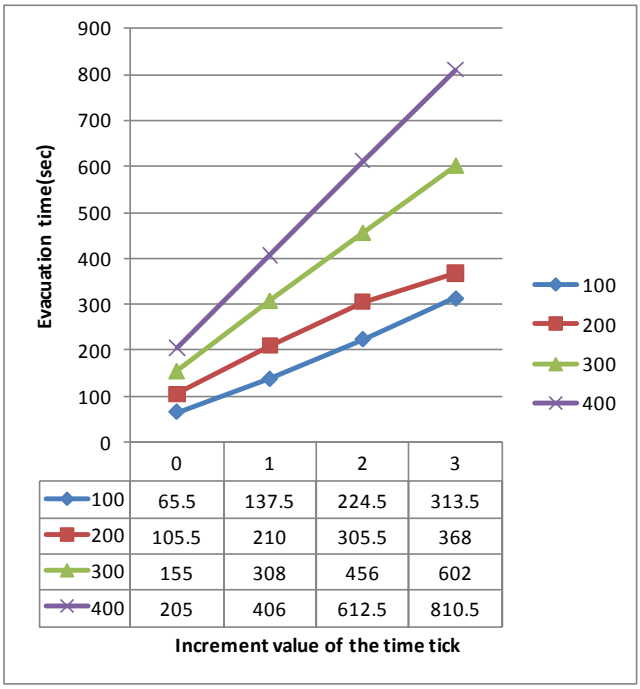


Figure 5. Comparing the floor field model and the one with the visibility field included

4. IMPLEMENTING THE SIMULATOR

The implementation process was divided into two steps. First, topologically connected building floors and stairs and attribute data were constructed and stored in a spatial database. Then, they were converted to grid cells to be used as input data for the simulation. This input data could then be used as the computation in the simulator and 3D visualization. The processes are summarized in Figure. 7.

Building floor data in CAD format were first converted to 2D vector layers such as shapefiles (Figures 7-(a) and (b)). Then, polygon and line data needed for 2D and 3D visualization and simulation were extracted and stored in a spatial DBMS. Topologically interrelated indoor compartments, including rooms and stairs, as well as attribute data were stored in the DBMS (Figure 7-(c)). The necessary parts were taken from the simulator and then converted to grid cell format for the simulation (Figure 7-(d)). Finally, the simulation was carried out using the grid cell data. The processes can be displayed in 2D and 3D while the computations are performed (Figure 7-(e)). The simulation results, which consist of the number of escaped agents by time increment, were written to a log file and stored back in the DBMS for post analysis and later integration with real-time applications.



	0	1	2	3
100	65.5	137.5	224.5	313.5
200	105.5	210	305.5	368
300	155	308	456	602
400	205	406	612.5	810.5

Figure 6. The evacuation time of varying numbers of agents with different increments

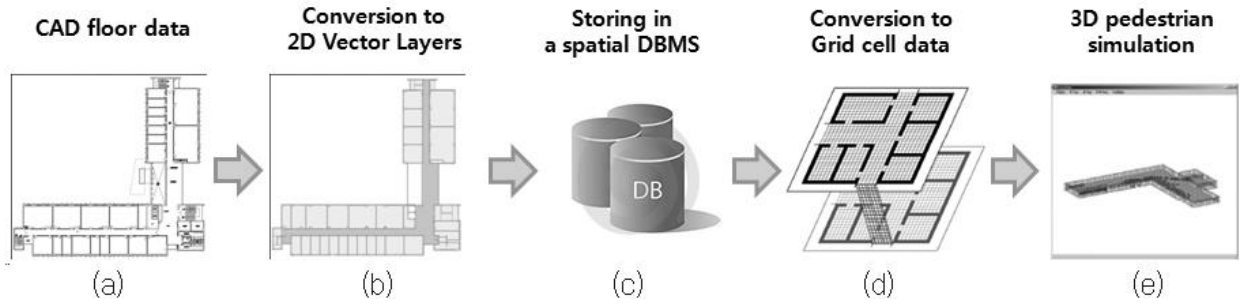


Figure 7. Data construction process

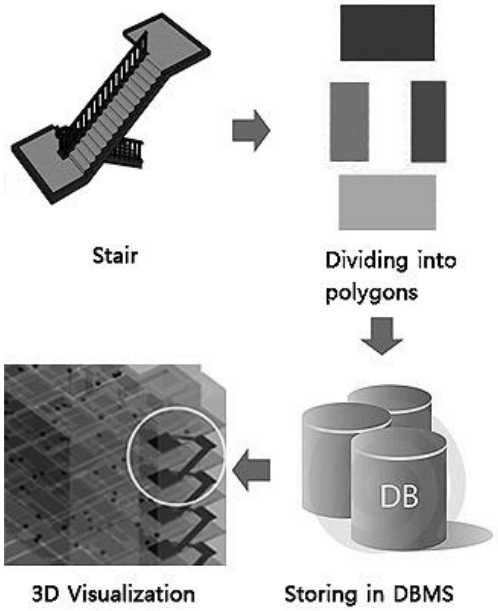


Figure 8. Data construction process for stairs

4.1 Acquisition and storage of spatial data

For the preparation of floor plan data, instead of developing a new format, we used the exiting shapefile format, which can be easily obtained using the tools included in many GIS applications. We used QuantumGIS [22] for the conversion. After the CAD files were imported to QuantumGIS, information about things such as walls, doors, rooms, corridors, and exits was extracted in the form of lines and polygons and stored into shapefiles. During the process, we also needed to assign proper coordinates for conversion to grid cell data later. Extracted 2D layers were stored in a spatial DBMS in table form. In this study, we used PostgreSQL/PostGIS for the DBMS [20]. Space elements such as corridors, rooms, walls, doors, and openings (exits) were stored in corresponding tables. Attribute data such as building floor numbers, room numbers, and room uses were also stored in the DBMS.

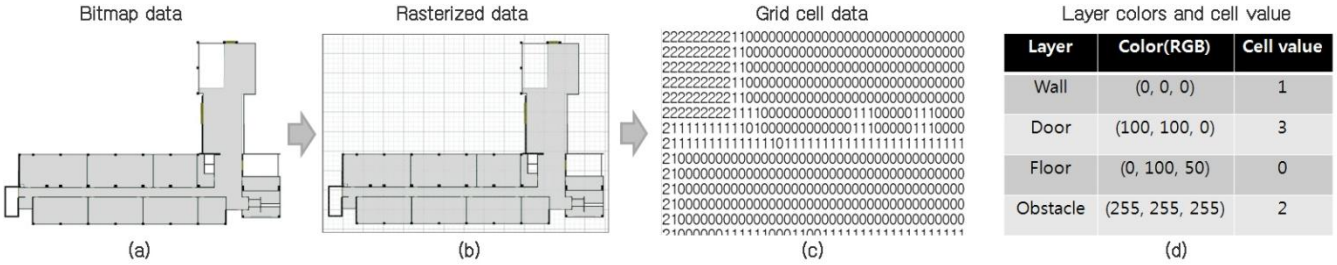
In addition, stairs were constructed and stored separately in a table because they were not in plane form (as are building floors) and cannot be converted directly from CAD files using tools. As will be described later, our evacuation simulation requires plane data as the base data structure. Figure 8 describes the process for the conversion of stairs. First, stairs were divided into a connected set of rectangles. In Figure 8, we can see a simple type of stairs is composed of four rectangles. These stair polygons were then stored in the DBMS. In this way, all the floors and stairs were prepared in the form of planes that could later be converted to 2D arrays for the simulation.

4.2 Converting to grid cells

Since we based our model on the floor field model, we needed grid cell data as the input format. In this study, we used C# programming language to implement the simulation system. We used npgsql library [21], which is compatible with PostgreSQL, for loading spatial data. Details of the data conversion process are shown in Figure 9. First, the building data and attribute data stored in the DBMS were loaded into the simulator. Then, the loaded floors and the stair data were converted to bitmap formats using SharpMap [24], shown in Figure 9-(a). The SharpMap library provides functions to visualize spatial data fetched from spatial DBMSs in 2D. The reason for converting to bitmap format is that it has a raster structure that can easily be converted to grid cells and allows us to distinguish different layers using colors. In bitmap data, the layers of doors, rooms, hallways, and walls are assigned different colors because color values are used in assigning cell values. Figure 9-(d) shows that RGB values are used to represent those layers.

Next, the coordinates of the floor and stair data were set since the converted grid data did not contain the coordinates. In this study, we used the top left corner of the first floor as the base coordinate. By using this, the coordinates of other floor spaces and stairs were relatively determined, as shown in Figure 10.

The next step was to determine the cell size and the number of cells in each spatial element. We used 40 cm x 40 cm as the cell size of the grid data, considering human shoulder width. By using this size, we could derive the number of cells in each rectangular space. Then, as described above, cell values were assigned based on the bitmap colors (Figure 9-(d)). Through this process, the floors and stairs in vector layers were converted to grid cell data. We developed the simulation system such that this conversion process can be implemented seamlessly.



Layer	Color(RGB)	Cell value
Wall	(0, 0, 0)	1
Door	(100, 100, 0)	3
Floor	(0, 100, 50)	0
Obstacle	(255, 255, 255)	2

Figure 9. Data construction process for grid cells

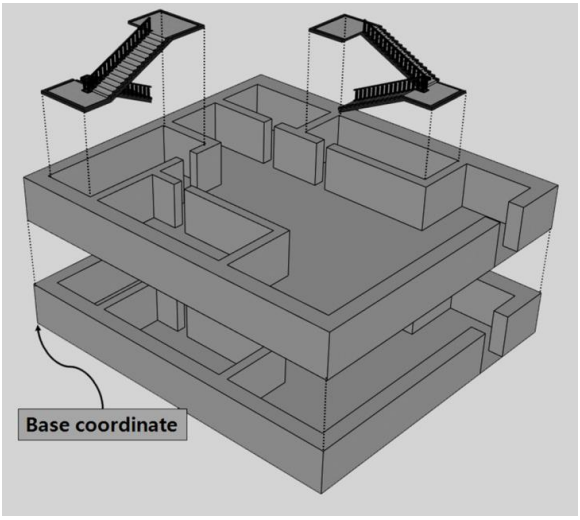


Figure 10. Determining the base coordinate and the coordinates of other spaces

4.3 Simulation and 3D visualization

We carried out evacuation simulations using the converted grid cell data. We used the revised floor field model that uses our proposed visibility field. Randomly created agents were first placed in the target building and the evacuation process was run and visualized. To visualize the building and the moving pedestrians in 3D, we used OpenGL Library. In order to construct the walls and the 3D shapes of stairs, we used the height stored as attribute data in the DBMS. Stair slopes were calculated from these coordinates and displayed in 3D.

The simulation results were stored in log files in text format. Exit locations and ID values of pedestrians according to escape time were stored. The total time it takes for all agents to escape could be obtained from this information. The system allows us to choose whether we apply the visibility field or not and to set various parameters.

5. SYSTEM TEST

We used a campus building for our system test. Following the processes described in the previous section, the building data in CAD format were converted to shapefiles and then stored in a spatial database. The system then read in the data and converted them to grid cells. Figure 11 shows the spaces partitioned based on visibility and Figure 12 shows a 3D snapshot during the system run.

Figure 13 illustrates the log files containing simulation results. From this, we could obtain the ID of a certain agent and determine which exit that agent used for evacuation at a certain time. 'TIMETICK' in Figure 13 represents the time spent for the evacuation. 'AGENT ID' is the agent's id and 'ACTION' is whether or not the agent escaped. 'INIT POS' is the initial coordinates of the agent, and 'EXIT POS' is the coordinates of the exits used for escape. We could compute the time it took for all agents to evacuate. These log files were stored back in the DBMS. We could use these results to analyze the building structures, for example, to determine which parts of the building are bottlenecks that make escape difficult. If the system is integrated with localization sensors, which has not happened yet, the statistical records of people in different parts of the building captured by the sensors could then be used to refine the simulation or to help in rescuing people during real-time emergencies.

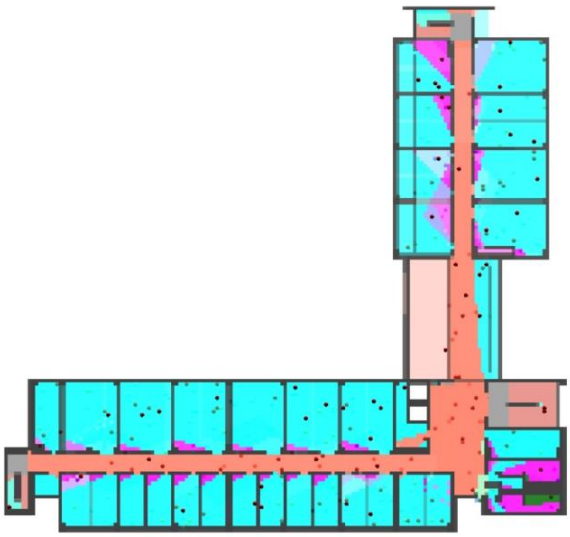


Figure 11. Simulation shown in 2D with partitioning of the spaces by a visibility field

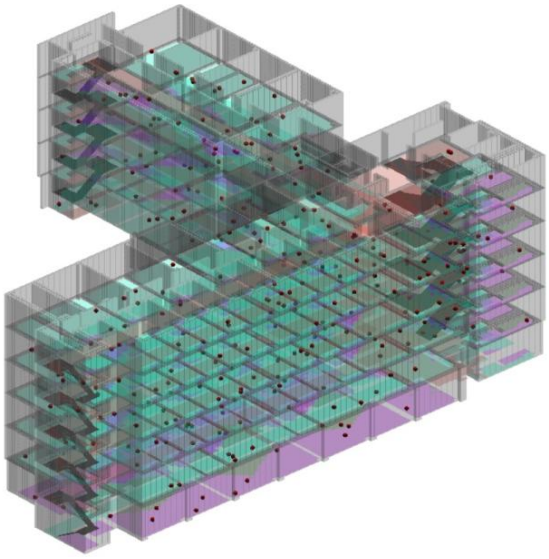


Figure 12. Simulation example displayed in 3D

# TIMETICK(SEC)	AGENT ID	ACTION	INIT POS	EXIT POS
3(1.5)	32	OUT	(3,6,0)	(0,5,0)
3(1.5)	248	OUT	(1,2,0)	(0,5,0)
4(2)	54	OUT	(44,16,0)	(47,20,0)
4(2)	223	OUT	(46,16,0)	(46,20,0)
6(3)	36	OUT	(43,14,0)	(47,20,0)
7(3.5)	202	OUT	(42,17,0)	(47,20,0)
7(3.5)	256	OUT	(42,18,0)	(46,20,0)

Figure 13. The log files containing the simulation results

6. CONCLUDING REMARKS

In this study, we developed a 3D indoor pedestrian simulator using a spatial DBMS. The enhancement we aimed for is twofold. First, we developed a process to build and store 3D indoor building spaces using a spatial DBMS. When developing the simulator, we made it communicate seamlessly with the data in the DBMS for the preparation of input grid cell data. Second, we incorporated the visibility factor into the existing floor field model. By adding visibility field, we were able to model the level of disadvantageousness of finding the exit according to the degree of visual depths which is calculated by on what degree the visual paths to the exit are allowed from the given location. The results proposed here are part of an ongoing research project that aims to develop real-time systems. The simulation results stored in a database could be used under real-time emergency conditions to judge if the current population captured by sensors is abnormal compared to the stored exit capacity and if alternative routing is necessary.

7. ACKNOWLEDGMENTS

This research was supported by a grant (08KLSGC04) from Cutting-edge Urban Development - Korean Land Spatialization Research Project funded by the Ministry of Land, Transport and Maritime Affairs and also by the Supporting Project For Education of GIS experts.

8. REFERENCES

[1] Ahuja, R. K., Magnate, T. L., Orlin, J. B. 1993. *Network Flows, Theory, Algorithms, and Applications*, Prentice Hall.

[2] Blue, V. J. and Adler, J. L., 1999. Using cellular automata microsimulation to model pedestrian movements, In A. Ceber (Ed.), *Proceedings of the 14th International Symposium on Transportation and Traffic Theory*, Jerusalem, Israel, 235-254.

[3] Burstedde, C., Klauck, K., Schadschneider, A., and Zittartz, J., 2001. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A 295*, 507-525.

[4] Cho, D. 1999. *A Study on the Tools of Analysis in the Space Syntax Theory*, Architectural Institute of Korea, 156, 71-76.

[5] Gwynne, S., Galea, E.R., Owen, M., Lawrence, P.J., Filippidis, L.L. 2005. A systematic comparison of building EXODUS predictions with experimental data from the Stapelfeldt trials and the Milburn House evacuation, *Applied Mathematical Modeling*, 29, 9(Sep.2005), 818-851.

[6] Hamacher, H. W., Tjandra, S. A. 2001. Mathematical modeling of evacuation problems- a state of art. In *M. Schreckenberg and S. Sharma, (Eds.), Pedestrian and Evacuation Dynamics*, Springer-Verlag, Berlin, 227-266.

[7] Helbing, D., Farkas, I., Molnár, P., Vicsek, T. 2001. Simulation of pedestrian crowds in normal and evacuation situations. In *M. Schreckenberg and S. Sharma, (Eds.), Pedestrian and Evacuation Dynamics*, Springer-Verlag, Berlin, 21-58.

[8] Helbing, D., Farkas, I., Vicsek, T. 2000. Simulating dynamical features of escape panic, *Nature* 407(Sep.2000), 487-490.

[9] Helbing, D., Molnár, P. 1997. Self-organization phenomena in pedestrian crowds, In *F. Schweitzer (ed.)*, Self-Organization of Complex Structures: From Individual to Collective Dynamics, Gordon & Beach, London, UK.

[10] Henein, C., White, T. 2005. Agent-based modeling of forces in crowds, In P. Davidsson, B. Logan and K. Takadama (Eds.), Multi-agent and Multi-agent-based Simulation, Lecture Notes in Computer Science, 3415, Springer, New York, 173–184.

[11] Henein, C., White, T. 2007. Macroscopic effects of microscopic forces between agents in crowd models, *Physica A*, 373, 694-712.

[12] Hightower, J., Boriello, G., and Want, R., 2000. *SpotON: An indoor 3D location sensing technology based on RF signal strength*. University of Washington CSE Report.

[13] Kirchner, A., and Schadschneider, A., 2002. Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics, *Physica A*, 312, 260-276.

[14] Klupfel, H., Konig, T., Wahle, J., and Schreckenberg, M., 2002. Microscopic simulation of evacuation processes on passenger ships, *In Proceedings of Fourth International Conference on Cellular Automata for Research and Industry*, Oct. 4-6, Karlsruhe, Germany.

[15] Kretz, T., Schrekenberg, M. 2006. Floor field- and Agent-based Simulation Tool, *International Symposium of Transport Simulation* 2006, Lausanne, Switzerland, 4 - 6.

[16] Li, B. et al., 2006. Indoor positioning techniques based on wireless LAN. *1st IEEE International Conference on Wireless Broadband and Ultra Wideband Communication*. Sydney, Australia, 13-16.

[17] Lo, S.M., Fang, Z., Lin, P., Zhi, G.S. 2004. An evacuation model: the SGEM package, *Fire Safety Journal*, 39, 3(Apr.2004), 169-190.

[18] Nishinari, K., Kirchner, A., Namazi, A., Schadschneider, A. 2005. Simulations of Evacuation by an Extended Floor Field CA Model*, Traffic and Granular Flow '03*.

[19] Penn, A., B. Hillier, D. Banister, and Xu, J. 1998. Configurational modeling of urban movement networks, *Environment and Planning B-Planning & Design 25*, 1, 59-84.

[20] PostgreSQL, http://www.postgresql.org/.

[21] PostgreSQL/npgsql, http://pgfoundry.org/projects/npgsql/.

[22] QuantumGIS, http://www.qgis.org/.

[23] Randell, C. and Muller H. 2001. Low cost indoor positioning system. *Ubicomp 2010 Conference of Ubiquitous Computing*. 42–48.

[24] SharpMap, http://www.codeplex.com/SharpMap/.

[25] Zhou, R., 2006. Wireless indoor tracking system (WITS). *In doIT Conference on Software Research*, Verlag Heidelberg, Germany, 163-177