Agent simulation of the evacuation process from a building during a fire

Andrzej Kułakowski
Faculty of Elec. Eng., Autom.Ctrl and Comp.Sci.
Kielce University of Technology
Kielce, Poland
a.kulakowski@tu.kielce.pl

Bartosz Rogala

Kielce University of Technology
Kielce, Poland

***Abstract*— The paper describes a simulation application for the process of evacuating people from a building in the event of a fire. This simulation was developed as an agent system using the DEVS formalism. The following sections show the design and implementation of such a system using the Adevs library. A series of simulation tests were then performed for different simulation scenarios.**

Keywords— computer simulation; agent simulation; building evacuation; DEVS

INTRODUCTION

This Simulating the evacuation process of crowded rooms enables to improve human safety when designing new buildings. The ability to study the impact of changes significantly improves the safety of newly designed objects. Simulation of a dangerous situation, i.e. a fire, requiring the evacuation of people from the building allows anticipating its effects and introducing changes to the project at an early stage. In recent years many solutions have been developed that allow performing very advanced and meticulous simulations. In the case of behavioral simulations, the integration with the multi-agent system (M.A.S.) is a good solution. This type of simulation is gaining flexibility because each agent has its own perception, autoname, and the ability to communicate [1]. Each agent has its own state, a set of behaviours and rules that determine the status of the agent in the next step. The concept of agents represents the idea of an autonomous system that perceives the environment and acts in it. The agent has an internal state representing their knowledge and purpose, usually maximizing its usability [2,3,4]. Thanks to the independence of individuals, multi-agent systems are most commonly used in social and behavioral sciences, but also in such disciplines as epidemiology and ecology [2].

Examples of existing evacuation simulation solutions are described in [5,6,7].

SPECIFYING THE SIMULATION ENVIRONMENT

Creating each simulation requires specifying the requirements and assumptions of the project. In the case of multi-agent simulations, apart from specifying the simulation stage itself, i.e. the corresponding model, tools and parameters, consideration should also be given to creating the environment itself.

Designing an evacuation system application requires the use of map representation methods and path search algorithms. Mapping motion of an individual in a computer program requires the implementation of one of the map representation methods. The most commonly used solution is to create a navigation mesh that maps the area where one can move. To designate a path on such a map, it is necessary to create a hybrid forming a grid of triangles on the basis of its own shape [8]. To create a grid of triangles, representing the map discussed, Delaunay triangulation has been used [9].

Agents of multi-agent systems need to adapt to the changing environment they are in. In the case of human motion, this includes, eg. the observation of other agents, avoiding collision and searching destination. Representations of the map serve as a basis for finding a path while selecting an algorithm.

The main direct search strategies are: uniform search cost and heuristic search. The A* algorithm combines both approaches. This allows A* to find the optimal path while being much more efficient than other algorithms [8].

TOOLS USED

Creating any software development project requires the use of appropriate tools. For the agent simulation of the evacuation process, the Adevs library was selected as the basis for the simulation [12]. This library allows programming the behaviour and relationships between agents as well as the entire simulation. Because Adevs is written in C ++, the whole project has also been developed in this programming language. Creating a simulation environment and its visualization requires the use of tools for programming 2D graphics and graphical user interfaces. In this case, the Qt library was selected to implement all required elements.

Delaunay triangulation was used to obtain the Triangle [10] library. It allows creating several types of high quality mesh using different Delaunay and Voronoi triangulation algorithms. The library uses its own files describing the polygons and the result files with node

points generated by the grid and the triangle neighbours. An important feature of the program is the ability to handle "holes" in the polygons because they represent obstacles on the map. In addition, the increase in the number of generated triangles is transferred to the quality of the path being created [10].

The final tool needed to create a simulation environment is the simulation library itself. In this case, the Adevs library was selected. Adevs (A Discrete EVent System Simulator) is a C++ library for modelling and simulation of DEVS models [11,12,13]. The library supports the basic model DEVS, Parallel DEVS and Dynamic Structure DEVS (DSDEVS). Adevs is a free library described by [11] as the fastest among the available ones. This is very important for large multi-model projects. Developing discrete simulations in Adevs involves combining atomic models into more complex structures [12].

The purpose of the formalism was to enable the construction of discrete models controlled by events in a hierarchical, modular system. DEVS This is a formal specification for a system scheme that enables modelling and analysis of discrete systems that can be described as transient status tables [13].

PROGRAM ALGORITHM

Creating The application allows carrying out the entire simulation process, from creating the whole environment, through simulation to visualizing the results. The application design assumes that the whole process can be carried out in a few steps: creating a simulation environment, drawing navigation nets for each floor of a building: determining emergency exits and obstacles, connecting emergency exits between floors, creating an area of fire, then adding agents to the floor. The next step is setting and performing the simulation parameters, then analyzing the results of the simulation and its automatic visualization or manual visualization of each step.

Decision agent model

The agent model analyzes the environment in which it is located and communicates with other agents to make a decision to change their own status. The current agent state is determined by the decision tree. Agents have three basic parameters: knowledge of the environment, degree of courage and speed with which they move. The courage and knowledge parameters are used in taking the decision, additionally the environment and other agents are taken into account. When the decision is made, the agent performs the selected motion model.

The agent determines the motion vector to the next point on the found route or to the agent they follow. The next steps of the agent along the route are calculated by adding motion vectors to the current position. If the next step is not possible, the agent checks whether the points shifted by 45- and 90-degrees on both sides are possible to occupy, if not, they remain stationary. Otherwise, the agent occupies the desired place and updates its route.

The simulation uses the A* algorithm. The triangular grid found in this way will not be optimal and does not correspond to the actual path travelled by the person possibly moving in straight line. In addition, smoothening of the found route needs to be done, including collisions with fire and obstacles.

Simulation of real fire is one of the most complex algorithmic problems. Since the paper focuses on the evacuation process in multi-agent simulation, a simpler solution is used. Fire is represented by an ellipse that grows with time. The simulation foresees the fire expansion, blocking subsequent triangles of the grid, thereby eliminating them from locations available to agents. In addition, on the way agents check whether the next steps do not lead them towards the fire, if so, they update the route.

Simulation process

The simulation algorithm is based on the diagram model available as part of the Adevs library [12]. The simulation assumption is independent mapping of the building floors (Figure 1). Every floor has a list of agents who are currently on it. The agent can move from one floor to another using the additional Atomic teleport class. The next steps in the simulation are managed by the clock updating time, synchronizing the simulation on the floors, and sending a command to calculate the next step in the simulation.

The next simulation steps are initiated by the clock, sending the current time to all floors. The floors perform the movements of all agents, checking whether the agent does not leave the floor. In this case, the agent is sent by teleport to another floor if there is enough space on the map. On the last stage of the step, the teleport checks the status of each floor and then sends the information to the clock, continuing the loop or ending the whole process.

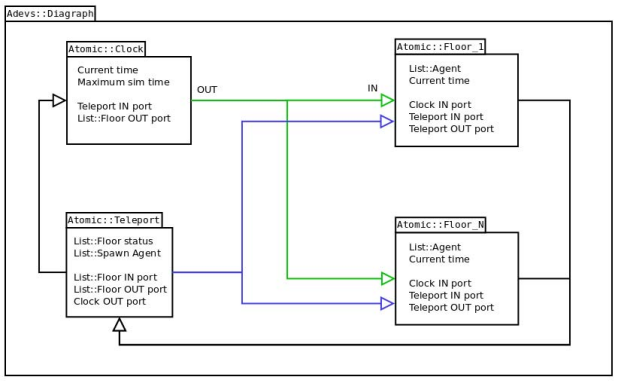

Fig. 1. Diagram of simulation model created using DEVS formalism in the Adevs library [14].

APPLICATION IMPLEMENTATION

Creating the environment project

Each floor of the project is based on a plan loaded from the graphic file. Ready-made design of the floor taking into account people, obstacles, emergency exits and fires (Figure 2).

The graphic plan serves as a guide for creating the navigation grid. It consists of three types of polygons that define the surface type: blue for floor, red for emergency exits, and gray for obstacles. Polygons can be freely joined and modified by adding or subtracting vertices. The

program will automatically create the missing polygons of the obstacle and will generate a new grid.

Then, on the ready map there are placed agents with the specified parameters and an ellipse representing the fire. Once all the floors are complete, the appropriate passages must be connected.

The last step is generating a triangle grid and checking the connections. The mesh compression can be controlled by setting the maximum area of the triangles. Triangulation is performed by Triangle external program and its results are loaded into the program.

Evacuation simulation

Simulation takes two parameters: the maximum time and the time defining the simulation step. The simulation step determines the unit of time by which the agent's position is updated. The simulation ends when the maximum time is reached or when none of the agents is able to move. The implementation in Adevs maps the model presented in Figure 1.

The simulation starts by sending an UP_TIME message containing the current time to all floors by the clock. Each of them triggers all their agents to complete the next step, defined by their status. The floor examines the number of agents moving and the number of the ones wanting to move to another floor.

The teleport receives messages from all floors, checking their type, determines whether any agents need to be forwarded. If so, the teleport sends agents to the appropriate floors and then expects a feedback. When all the floors make a move, the teleport checks their status and sends the information about the next step to the clock. If all floors have NO_MOVE responses, the simulation ends prematurely because all agents have already left the building or are no longer able to move.

Each subsequent simulation step calculates the point at which the agent should be located and writes them to the list. These values are used to recreate the positions of agents at a given time and visualize their movements. The program allows playing the simulation automatically or manually by moving the time slider.

TESTING THE SIMULATION

Each computer simulation requires a validation process for the simulation model. Testing is necessary to determine if the model meets the design assumptions. In the case of the analyzed application, testing was performed on individual agent motion systems. Three simulation scenarios are then presented to validate the appropriacy and suitability of the applications in real-life situations [14].

Motion systems

The agents' motion required the implementation of collision and analysis of the surrounding systems to represent reality most accurately. The first is to bypass the fire while mapping out the agent's route, escaping from the fire when it is close, and dying in the event of a rapidly expanding flame.

The other systems are the search for the leader (a trained person), the closest visible exit, and the collision check. The program checks collisions with walls and other obstacles to determine what the agent can see. When an agent does not know the distribution of rooms, but can see another agent possessing the knowledge, he follows him. If an agent detects a collision possibility during a movement, it bypasses the obstacle. Of course, it omits another agent as well.

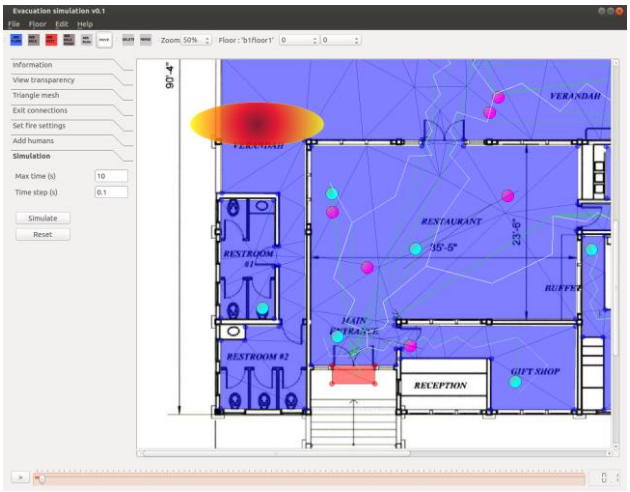

Fig. 2. Ready-made application during the evacuation simulation of a commercial building [14].

Simulation scenarios

The first scenario examines the effect of the exit system in a commercial building during the spread of a fire (Figure 2). Three simulations have been carried out with the following assumptions: a two-speed fire spreading in a restaurant building is being tested; the setting and all agent parameters are the same for each simulation.

Runs of simulations: fire spreads at a speed of 1 cm / s, the building has two exits; the fire spreads at a speed of 5 cm / s, the building has two exits; the fire spreads at 5 cm / s, the building has four exits.

The first simulation showed that when the fire develops slowly, potentially everyone could escape, but three people had to approach the fire very closely. In fact, these people would be injured or would not be able to leave the building. In the second simulation, the fire blocked the passage, resulting in the death of four people in the 11th second. Additional door in the third simulation significantly shortened the evacuation time and provided a safe route for all agents.

The second scenario analyzes the simulations of restaurant evacuation with different quantities and locations of obstacles (tables) (Figure 3). It checks the time needed for total evacuation and the average exit time of thirty-one agents. There were six simulations carried out. The fire spreading speed was set to 0.1 cm/s.

The greatest impact on the speed of evacuation of all agents was increasing the number of exits. By rearranging obstacles, you can reduce the average time an agent needed to leave the room. The solution can also be to

remove some obstacles e.g. in the vicinity of the exits, thus providing greater flow capacity in critical locations.

In public buildings like banks or offices, the clients do not always know the distribution of rooms. The last simulation checks the influence of agents' knowledge in case of changing the number of "employees" who the agent can follow towards the exit. The other simulation parameters are controlled. Office plans will be used in the analysis (Fig. 4).

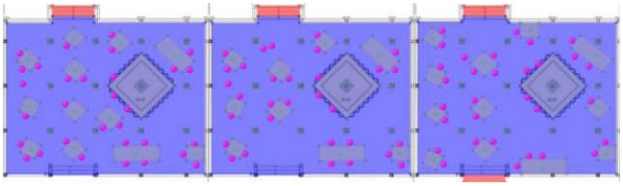

Fig. 3 Layout of the simulation room, left: initial setting, deletion of three obstacles at the exits and modification of the setting [14].

The simulation shows that the agents who are unfamiliar with the layout of the premises are not able to find the way out themselves. Some panic or wander aimlessly looking for the way. Increasing the number of agents who know the way and whom the rest can follow, affects the number of people rescued and the time of simulation. In the third simulation, it can be observed that the time of the last evacuation and average time are higher than before. This is due to the following and later loss of visual contact with the “employee”. This results in an autonomous search for the way that takes more time. With the large number of agents who know the way and their proper location, the clients follow them, leaving the building much faster.

Fig. 4 Room plan with clients only (pink) and "officials" (blue ones) [14].

The above simulations show how important building design is and how computer simulation is helpful in this process. Simulation results analysis allows you to design buildings in a faster, cheaper, and most importantly secure way. In addition, the simulation data helps in the event of an actual evacuation, allowing to predict the potential number of people injured.

SUMMARY

The paper presents selected theoretical aspects of computer simulation applied to human evacuation. An algorithm and outline of application implementation is then presented to create and simulate the evacuation process in the event of fire. The tools used allow you to create an entire simulation environment and modify it while conducting further tests. Meeting the evacuation system requirements and testing it, shows the suitability of such solutions in the real world.

Several evacuation scenarios are also presented, showing how environmental changes affect the evacuation process. It has been shown that while designing the building a large impact on security is anticipated for potential fire sources and appropriate matching number of exits and evacuation paths. Such modifications not only secured the survival of all agents, but also reduced the time needed to leave the building. Even minor changes to the obstacle setting or the addition of trained agents (fire wardens) allow for a significant improvement in the speed of evacuation.

The present form of the application does not allow for the creation of fully realistic simulations, but the presented solution of the multi-agent system allows for its further development.

REFERENCES

[1] F. Klügl, G. Klubertanz, G. Rindsfüser, "Agent-Based Pedestrian Simulation of Train Evacuation Integrating Environmental Data", KI 2009: Advances in Artificial Intelligence, Lecture Notes in Computer Science Volume 5803, 2009, pp. 631-638.

[2] A. M. Uhrmacher, D. Weyns, "Multi-Agent Systems: Simulation & Applications", CRC Press, 2009.

[3] E. Alonso, N. Karcanias, A.G. Hessami, "Multi-Agent Systems: A new paradigm for Systems of Systems", ICONS 2013 : The Eighth International Conference on Systems

[4] G. Weiss, "Multiagent Systems", A Modern Approach to Distributed Artificial Intelligence, The MIT Press, 2000.

[5] M. Van Schyndel, "Evacuation for a building with multiple levels", SYSC 5104, Carleton University 2011.

[6] J.L. Smith, "Agent-Based Simulation of Human Movements During Emergency Evacuations of Facilities" (https://www.wbdg.org/pdfs/agent_based_sim_paper.pdf).

[7] Pathfinder Thunderhead Technical Reference (http://www.thunderheadeng.com/downloads/pathfinder/tech_ref.pdf).

[8] X. Cui, H. Shi, "Direction oriented pathfinding in video games", International Journal of Artificial Intelligence & Applications (IJAIA), Vol. 2, Oct. 2011

[9] N. P. Weatherill, "Delaunay triangulation in computational fluid dynamics", Computers & Mathematics with Applications Vol. 24, 1992, pp. 129-150

[10] Triangle, 2015 (https://www.cs.cmu.edu/~quake/triangle.html).

[11] Y. Van Tendeloo, H. Vangheluwe, "The Modular Architecture of the Python(P)DEVS Simulation Kernel Work In Progress paper", G.A. Wainer, DEVS: Proceedings of the Symposium on Theory of M&S, pp. 387–392. SCS International, 2013

[12] J. Nutaro, "Adevs Manual, A Discrete EVent system Simulator", 2014, (http://web.ornl.gov/~1qn/adevs/index.html).

[13] H. Song, "Infrastructure for DEVS Modelling and Experimentation", master thesis draft, supervisor: prof. Hans Vangheluwe, Montréal, Canada: McGill'a Univesity, 2006

[14] B.Rogala, "Project and application for building evacuation agent simulation process during fire.", master thesis, supervisor: dr.eng. A.Kułakowski, Kielce, Poland: Kielce University of Technology, 2015.