Real-Time Visualization of the Human Evacuation Algorithm

Piotr Byszewski [*]

Department of Computer Graphics
West Pomeranian University of Technology, Szczecin
Poland

Abstract

This paper presents a real-time visualization of human behavior in the indoor environments. A simple simulation of the evacuation algorithm was implemented together with the tools that support building of the indoor environments. However, the main effort was put on real time and realistic visualization of the simulation results. In order to obtain a high frame rate, an agent based model was integrated with techniques used in computer games. The proposed model allows to reduce a number of calculations performed per every individual agent by simplified decision making system and using precalculation techniques.

Keywords: evacuation visualization, agents, real-time rendering, computer graphics

1 Introduction

In real-time applications crowd simulation is a challenging topic. Large groups of individuals must present intelligent path planning which includes environment limitations and reflects their autonomous nature. Agents should also be able to communicate in order to react to a changing environment. Crowd visualization also presents a number of challenges, including animation, visibility culling and a level of detail. Moreover, performance and memory limitations must be taken into account.
Thalmann [16] distinguished the two main areas of crowd simulations. The first one concentrates on the realism of behavioral aspects where visualization is only used to help understand simulation results. The second area is focused on high-quality visualization. In this case convincing visual results are more important than the time needed to calculate them. Nevertheless, the combination of these two trends in real time is possible.
In this paper we present a real-time visualization of human behavior during evacuation process. Our goal was to achieve visually realistic human evacuation with minimal computation requirements. Many simplifications for the simulation process were used to accomplish this task. Moreover, additional information was embedded in the environment in order to reduce the complexity of controlling each agent. That information was used to precompute paths that are used by the agents. To visualize this algorithm Hive Simulator was implemented. This application is able to visualize evacuation process in any scene created ahead. Furthermore, Hive Editor was developed in order to quickly create 3D indoor environments used by Hive Simulator.
The remainder of the paper is organized as follows. In the second section we present in general the best known crowd simulation techniques. Our Hive Editor is described in the third section. The fourth section presents Hive Simulator. It also explains our simulation algorithm. The fifth section consists of results. Finally, the last section concludes the paper and proposes future work.

2 Previous Work

Crowd behavior has been studied for few decades now. These researches are mainly focused on psychological aspects and were based on direct observation. However, models, which can successfully simulate crowd behavior on a computer, have appeared just in recent years.
Helbing [3] proposed a model in which agents are moving in the direction calculated on the basis of generalized Nevtons force equation. This technique allows to realistically simulate fundamental collective effects like the formation of lanes, jamming near bottlenecks and ignorance of available exits.
Reynolds [11, 12] demonstrated that realistic crowd behavior can be achieved by using simple rules to control each agent. There are three types of these rules. Separation rules maintain a certain separation distance between nearby agents. Cohesion rules are causing a group formation effect. Alignment rules give the agent an ability to head in the same direction as other nearby characters.
In models based on cellular automata [1, 15] the environment is discretized into a grid of cells. These cells are used to store information about obstacles, other agents and region attributes. During simulation every individual is scanning its local environment and it can move when an adjacent cell has met occupancy rules.
Hughes [4] proposed a global approach that compares pedestrian crowds behavior to gases and fluids. That work inspired Treuille [17], who noticed that pedestrian motion can be considered as a per-particle energy minimization. Simulation in that model is driven by dynamic po-

[*]pbyszewski@gmail.com

tential field that unifies global navigation and local collision avoidance. Travel space is represented by regular grid cells, which are combined into a set of potential fields. These potential fields are then used to update people's positions. Computational cost of this algorithm mainly depends on the number of grid cells. The number of simulated people has minimal impact on performance. Continuum model can simulate thousands of pedestrians in real-time, but only if they are on a relatively small area. Additionally, this approach is designed to simulate large groups with common goals. It is not suited to simulate a large number of small groups.
The models presented above mainly focus on the simulation of crowd behavior ignoring the differences between the agents. These systems define the same behavior for every individual. Pelechano [5] introduced system in which Helbings social force model was extended by a set of geometrical and psychological rules. In that approach agents have different behaviors that can be triggered at certain situations. Tu and Terzopoulos [18] used artificial fish to present a behavioral model controlled by visual perception. Funge [2], on the other hand, introduced a cognitive model in which agents knowledge is used to plan actions.
Nevertheless, Massive (Multiple Agent Simulation System In Virtual Environment) [13] is the most commonly used system that allows to model and visualize different behavior for every agent in the crowd. Agents can have unique responses, emotions and different physical appearance. During simulation Massive uses artificial life approach. This technology allows agents to make a decision based on their simulated senses of sight, hearing and touch. However, the possibilities offered by Massive are computationally very expensive. Because of that, Massive is mainly used in non real time crowd simulations for the film industry.
Currently, a large number of units is also often seen in computer games [7, 8, 9, 10]. These games use a number of techniques that reduce the computing requirements. One of the most common optimization is the AI level of detail (LOD AI) where computations are performed only for the units that can be seen by a player. Another technique is a trigger system, which allows to trigger defined actions of agents only when a certain situation happens. This results in an illusion of intelligent agent reactions to the changes in environment. Scripted action sequences is a method that allows to predefine agent behavior. Agents controlled by these scripts will replay hard coded actions, which will look the same every time when they are played. One of the most important technique is discretization of walkable space. Space in which agents can move is described by graph structure. That prevents agents from walking through walls and other obstacles without providing them with visual sense simulation.

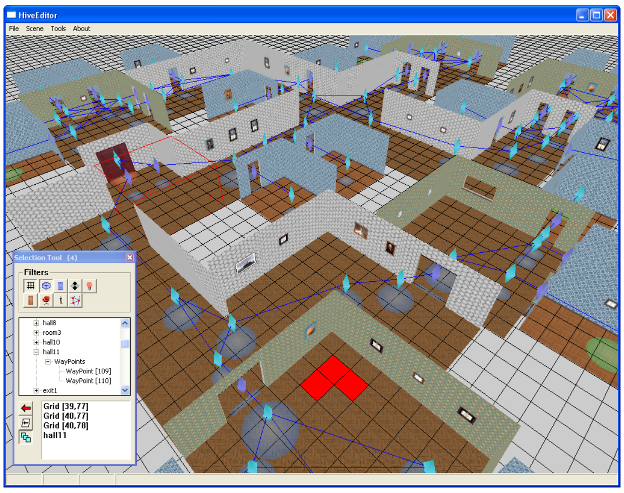


Figure 1: Hive Editor - main window.

3 Hive Editor

Hive Editor (Figure 1) is an application that can be considered as real-time level design tool, optimized for building 3D indoor environments. We automated scene creation process by providing functions that quickly perform certain standard operations. The main steps of scene creation process can be seen in Figure 2.
Once a 3D environment has been created it is saved into .hve file. That file contains all data needed to reconstruct a created scene. In this section we describe in detail how our Hive Editor can be used to create indoor environments.

3.1 Indoor environment

Almost all indoor environments can be considered as a set of simple cuboids connected with each other. This fact was used in room creation process in Hive Editor. To simplify room creation, only room position and size can be controlled. Room geometry is created automatically by room builder, which takes as an input selected grid tiles (selection process is realized through Selection Tool which can be seen in Figure 3). Before geometry is created various tests are preformed to determine if room can be constructed from given tiles in specified place. Proper vertex and quads are created only if all conditions have been met. Complex room shapes can be achieved by connecting simple cuboid-shaped rooms. Connecting rooms is also an automated process and it is realized by passage builder. Selected wall quads are taken as an input and are then analyzed along with the neighbouring geometry. This is done to determine which quads should be removed and where new corner quads should be created.
In order to differentiate rooms from each other and to make them more interesting from a perceptual point of view texturing module was implemented. Communication with this module is realized through Texture palette window (Figure 3). This tool allows to add and remove

textures from the scene. It also provides a possibility to manipulate UV coordinates and it enables basic operations such as rotation, vertical flip and horizontal flip. To accelerate texturing process we implemented the algorithm that automatically matches UV coordinates to room geometry. This algorithm assigns proper UV coordinates to selected quads depending on their size and neighborhood.

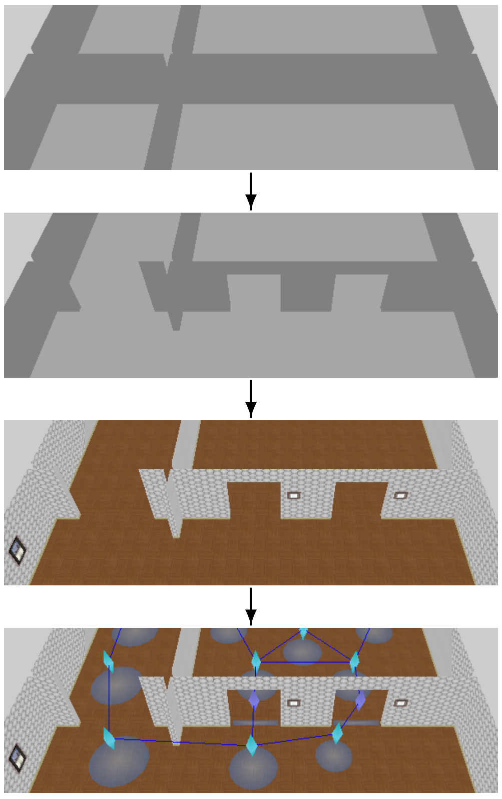
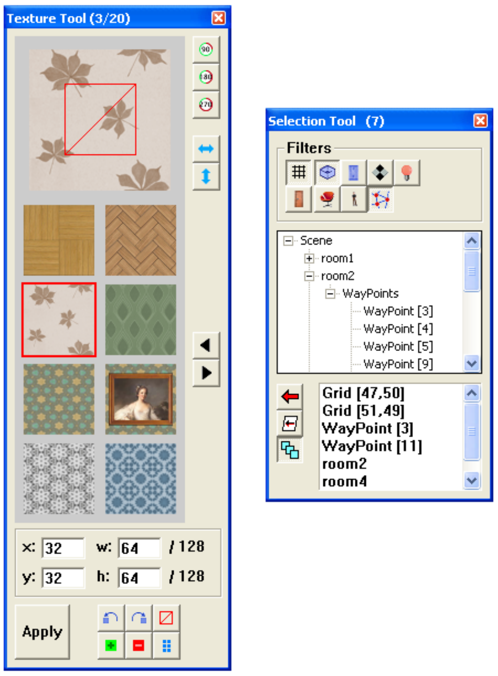


Figure 2: Scene creation process. From top to bottom: the rooms creation, connecting rooms, texture mapping, waypoint graph definition.

3.2 Waypoint graph

In big and complex environments pathfinding grows in importance. There are a number of algorithms realizing pathfinding, but the algorithm is not as important as the search space representation. Waypoint graph [8, 6] is one of the most popular representations of navigable areas. It consists of nodes that define walkable space, and of edges that determine connections between pairs of nodes.

Hive Editor allows to place waypoints anywhere inside rooms (Figure 4). It is also possible to manually connect selected waypoints. That will later inform navigating through waypoint graph agent so that there is no obstacle between these waypoints. Furthermore, there are two waypoint types: circle-based and rectangle-based. All their attributes can be modified in order to adjust them better to room geometry. Additionally, each waypoint can have one of three flags which allows to determine specific waypoint properties. These flags are: exit portal, spawn node and



Texture Tool (3/20)

x: 32 w: 64 / 128
y: 32 h: 64 / 128

Apply

Selection Tool (7)

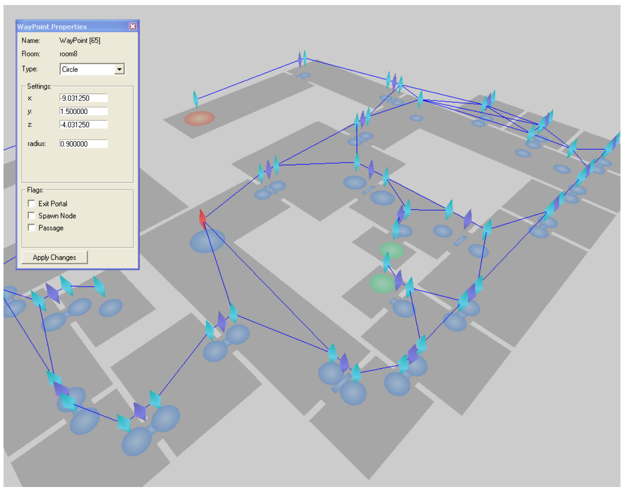
Filters

Scene
room1
room2
WayPoints
WayPoint [3]
WayPoint [4]
WayPoint [5]
WayPoint [9]

Grid [47,50]
Grid [51,49]
WayPoint [3]
WayPoint [11]
room2
room4

Figure 3: Texture tool (left) and selection tool (right).

passage. The first flag informs that a given waypoint is an exit from the scene. That waypoint will be a goal for agents during simulation process. Spawn node symbolizes an entrance to the scene. In that place new agents will appear. Scene can have more than one exit and entrance. Passage flag indicates that the given waypoint is placed between two rooms. It is the most important flag because it will inform agent that he is entering a new room.



WayPoint Properties

Name: WayPoint [65]
Room: room8
Type: Circle

Settings:
x: -9.031250
y: 1.500000
z: -4.031250
radius: 0.900000

Flags:
☐ Exit Portal
☐ Spawn Node
☐ Passage

Apply Changes

Figure 4: Waypoint graph.

4 Evacuation Simulation

Data created in Hive Editor are used by our simple simulation algorithm. Visualization of this process is realized using a program called Hive Simulator. However, there are several initializations and precomputations done before visualization starts (Figure 5).

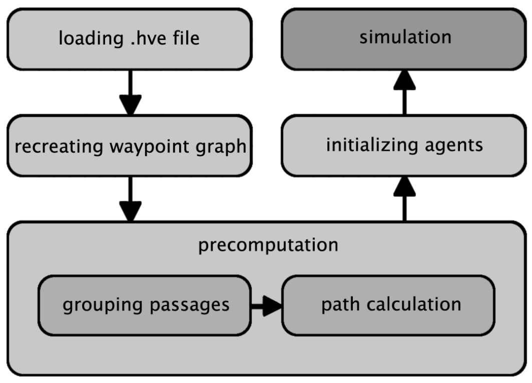


Figure 5: Schema of the initialization process.

After running Hive Simulator it is possible to specify the path to a particular .hve file. Data, stored in that file, are used to reconstruct scene geometry and a waypoint graph. In the next stage the waypoint graph is analyzed in order to create passage groups. These groups consist of passages that are directly connected with each other. This means that between every two passages within a given group there is a route that connects them without crossing the other passage. As a result passage group represents a room regardless of geometry data. Another important effect of grouping is that each passage connects exactly two groups.
The next step of the algorithm is to precompute paths between all passages in a group. Such a solution will allow to share one path between multiple agents that are moving towards the same goal. It will also significantly reduce the number of needed calculations. Additionally to provide quick access to a path, we organized them into two dimensional look-up table (LUT). The size of this LUT is equal to the number of passages in a group and their indexing corresponds to passage indexes in that group.
In the next stage we initialize agents. In current Hive Simulator version agents are positioned randomly inside all passage groups. That solution ensures that entire floor plan is roughly equally populated. We also assign to them random variables describing their movement and character. Finally, after positioning agents, simulation and visualization starts. Our simulation algorithm is described in section 4.1.

4.1 Simulation

One of the assumptions for our simulation is that agents are moving in the unknown to them environment, which means that they do not know where an exit is. This simplification can correspond to evacuation from smoke-filled environments.

During simulation process every agent is assigned to a passage group (that represents room) and to goal passage within this group. When an individual reaches his goal he is assigned to a new passage group. Also at that moment a decision is being made which passage in a new group will be a new goal passage. The majority of agents are assigned to the most popular passage, but some agents pick the second popular passage or they make their choice randomly. That depends on their character which is currently described by two variables: courage and curiosity. Both of these parameters are constant during simulation process.
After the goal passage was chosen the agent does not change his target until he reaches that passage. If the agent enters a passage group that consists only of one passage then he automatically goes back and chooses other passage from a previous group.
Our agents have a very simplified memory model. The only passage which is remembered is the passage through which the agent came to the group and thus the group in which the agent was for the last time. That information is used during passage selection to prevent agents from circulating between two groups connected by more than one passage. Furthermore, when the agent is assigned to the group he is also checking whether the group includes an exit waypoint (which represents an exit from the scene). If there is such a waypoint it will automatically become an agent goal. To ensure a constant level of agents in the scene whenever one agent reaches an exit, a new agent is created in one of the waypoints designated as a spawn node. In a real scenario that situation can correspond to humans coming to a given floor from higher floors of the building.

4.2 Implementation techniques

Visualization process is realized by using advanced OpenGL API features. At the moment rooms which were created in Hive Editor and agents that are currently represented by diamond shapes, are rendered. Additionally, a small line near the agent is displayed to symbolize his movement direction.
To speed up a rendering process, a few optimizations were implemented. In order to avoid having to bind different textures repeatedly while rendering rooms, an atlas packer was implemented. During scene initialization all textures, used to texture rooms, are packed and vertex UV coordinates are properly recalculated. Furthermore, room geometry data is stored in high performance memory by using vertex buffer objects (VBO). Vertex positions and texture coordinates are stored in separate VBO for every room. Additionally, every room is bound with axis aligned bounding box (AABB). These bounding volumes are used to construct a quadtree, which is later used during frustum culling process.

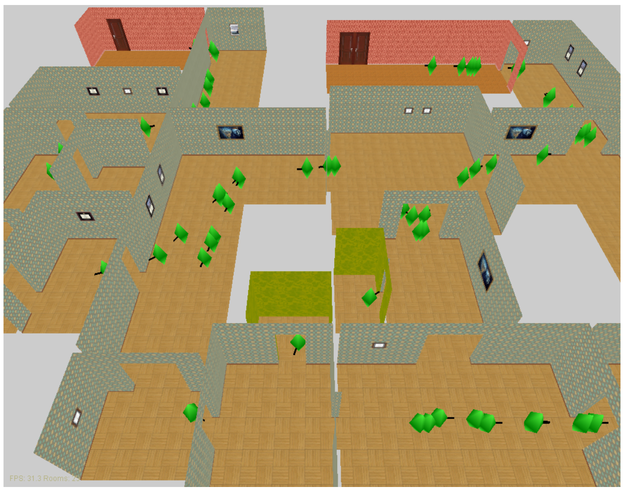


Figure 6: An example scene with 60 agents.

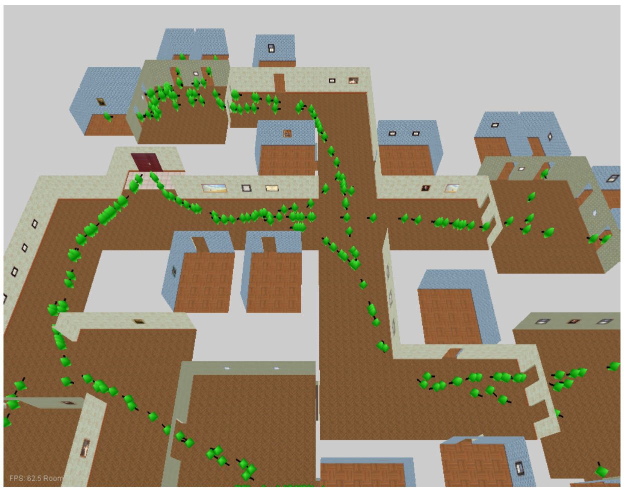


Figure 7: An example scene with 300 agents.

5 Tests and results

In order to estimate the accuracy of our simulation we conducted a series of experiments. These tests were mainly focused on speed and correctness of the simulation results. We used Hive Editor to create a number of scenes that had different number of rooms and level of complexity. Complex scenes are characterized by high levels of cyclic connections between rooms. Furthermore, each of prepared scenes was simulated with different number of agents, exit portals and spawn nodes. Two of these scenes can be seen in Figure 6 and Figure 7.

Neither the number of exit portals nor spawn nodes has an impact on the quality of the simulation. Similarly the number of rooms. However, the level of complexity of the scene has a significant impact on the validity of the simulation. If there are cycles between the groups of passages it may happen that certain agents will circulate between the same rooms.

All of the experiments were made on Intel Pentium 4 2.8 GHz with 512 MB of RAM and a GeForce FX 5700. With that hardware configuration and with 800 $\times$ 600 screen resolution our simulator can simulate 5000 agents with more then 30 frames per second. However, these tests were performed without collision detection. The lack of collisions also results in the fact that our visualization looks better when rooms are not overcrowded.

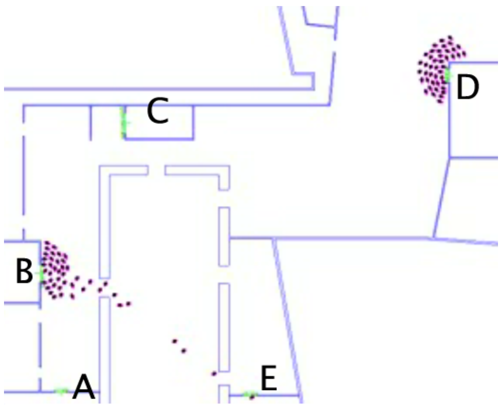
We have also compared our simulator with Simulex. Simulex from IES is advanced software that allows to define a building and its occupants, and simulate how they will evacuate during an emergency. For more information on Simulex refer to [14].

Unfortunately, we were not able to obtain license for this software but we did our comparisons basing on available recordings that presented Simulex simulation. We used our Hive Editor to define similar environments as were shown on these recordings. Next, we compared simulation realized by Hive Simulator with recorded Simulex simulation. One of the compared scenes is shown in Figure 8. The scene contains four exits, they are marked with the letters: A, B, C and D. As can be seen our agents prefer exits B, C and D while Simulex agents are concentrating only near exits B and D. In both simulations there is an exit (A) that is ignored, which is a common situation during a real evacuation. Letter E marks an entrance to the scene. Simulex agents who came through this entrance are heading to B exit. Unlike the Hive agents which are directing towards C exit. Our agent who comes through E entrance, does not calculate a path to a selected exit. It is moving towards selected passage within a current room. The first agents who came through that entrance selected passages sequentially and that led them to a given exit. Other agents were picking the same way which resulted in a lane formation towards that exit. These lanes are often not optimal and are forming only in certain conditions.

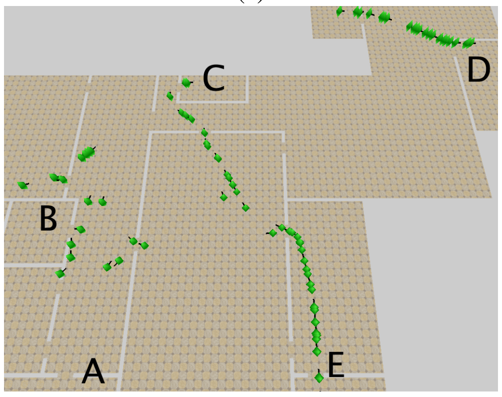
6 Conclusions and Future Work

This paper has described a simple model to visualize the evacuation process in real time. The presented simulation algorithm is based on agent architecture and it simulates human evacuation from unknown to them environment in restricted visibility conditions. In order to visualize that process we have created Hive Simulator. This program provides the possibility to perform simulation in any indoor environment for a specified number of agents. The scene in which the simulation takes place is created by our Hive Editor tool, which in addition to the geometric data, allows to create data necessary for visualization.

Our evacuation algorithm is not yet finished. In order to increase the accuracy of the simulation we will implement collision detection between agents located within the same passage group. This issue is related to the effect of jamming near bottlenecks, which will also be added. The agents decision making system will be expanded. We are planning to add stress level to agents, which will be impor-



(a)



(b)

Figure 8: Simulation process: (a) Simulex [14], (b) HiveSimulator.

tant during a goal passage selection. A variable representing stares level will change during simulation depending on the agent surroundings and his previous decision.
In the future we also expect to improve the simulation appearance. The current representation of the agent will be replaced with a human-like model, which is animated by using skeletal based animation system. Furthermore to increase overall scene appearance, a possibility to load and position static meshes inside rooms will be added to Hive Editor. We are also planning to add lightmaps which significantly improve the quality of a rendered scene.

References

[1] C. Burstedde, K. Klauck, A. Schadschneider, and J. Zittartz. Simulation of pedestrian dynamics using a 2-dimensional cellular automaton, 2001.

[2] John Funge, Xiaoyuan Tu, and Demetri Terzopoulos. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. pages 29–38, 1999.

[3] D. Helbing, I. Farkas, P. Molnar, and T. Vicsek. Simulation of pedestrian crowds in normal and evacuation situations. In M. Schreckenberg and S.D. Sharma, editors, *Pedestrian and Evacuation Dynamics*, pages 21–58, Berlin, 2002. Springer.

[4] R.L. Hughes. The flow of large crowds of pedestrians. *Mathematics and Computers in Simulation*, 53:367–370, Oct 2000.

[5] N. Pelechano, J. M. Allbeck, and N. I. Badler. Controlling individual agents in high-density crowd simulation. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 99–108, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[6] J. Pettre, J. Laumond, and D. Thalmann. A navigation graph for real-time crowd animation on multilayered and uneven terrain. *First International Workshop on Crowd Simulation*, pages 81–90, 2005.

[7] Steve Rabin, editor. *AI Game programming wisdom*. Charles River Media, 2002.

[8] Steve Rabin, editor. *AI Game programming wisdom 2*. Charles River Media, 2004.

[9] Steve Rabin, editor. *AI Game programming wisdom 3*. Charles River Media, 2006.

[10] Steve Rabin, editor. *AI Game programming wisdom 4*. Charles River Media, 2008.

[11] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics*, pages 25–34, 1987.

[12] Craig W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference 1999*, 1999.

[13] Massive Software. Massive. *http://www.massivesoftware.com/*, visited: 30.01.2009.

[14] Integrated Environmental Solutions. Simulex. *http://www.iesve.com*, visited: 30.01.2009.

[15] Franco Tecchia, Cline Loscos, Ruth Conroy, and Yiorgos Chrysanthou. Agent behaviour simulator (abs): a platform for urban behaviour development. In *In GTEC2001*, pages 17–21, 2001.

[16] Daniel Thalmann and Soraia Raupp Musse. *Crowd Simulation*. Springer, 2007.

[17] Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Trans. Graph.*, 25(3):1160–1168, 2006.

[18] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. pages 43–50, 1994.