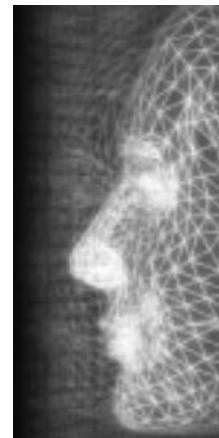


Real-time navigating crowds: scalable simulation and rendering

By Julien Pettré*, Pablo de Heras Ciechowski, Jonathan Maim, Barbara Yersin, Jean-Paul Laumond and Daniel Thalmann



This paper introduces a framework for real-time simulation and rendering of crowds navigating in a virtual environment. The solution first consists in a specific environment preprocessing technique giving rise to navigation graphs, which are then used by the navigation and simulation tasks. Second, navigation planning interactively provides various solutions to the user queries, allowing to spread a crowd by individualizing trajectories. A scalable simulation model enables the management of large crowds, while saving computation time for rendering tasks. Pedestrian graphical models are divided into three rendering fidelities ranging from billboards to dynamic meshes, allowing close-up views of detailed digital actors with a large variety of locomotion animations. Examples illustrate our method in several environments with crowds of up to 35 000 pedestrians with real-time performance. Copyright © 2006 John Wiley & Sons, Ltd.

Received: 10 April 2006; Revised: 2 May 2006; Accepted: 11 May 2006

KEY WORDS: real-time crowds; crowd simulation; crowd rendering; navigation

Introduction

We aim to provide tools to the cyberexplorer community, allowing them to populate their favorite virtual worlds. To promote ease-of-use to a large public, we need automatic and robust methods for preprocessing environments that support navigation and simulation tasks. We also need intuitive solutions to manage virtual populations using high-level directives, for designing both appearance variety and behavior in terms of navigation. We introduce scalability in simulation and rendering to preserve quality where attention is centered.

The principle of our approach is first to structure any virtual world into a graph capturing its topology. Unlike previous methods, ours is capable of automatically handling environments combining uneven and multi-layered terrains. Second, we introduce a navigation planning method adapted to produce a variety of paths,

naturally spreading the crowd. Pedestrians are steered at different levels of complexity. This allows us to scale simulations: computational resources are distributed in order to preserve quality in the observed area. Fourthly, the rendering engine reinforces believability by supporting individual variety both in terms of appearance and motion. Such a variety is due to models ranging from dynamic meshes to billboards.

Related Work

Path Planning and Simulation for Crowds

Moving several entities simultaneously using way-points can be achieved using steering or flocking methods.^{1–3} A sequence of way-points leading to a prescribed goal can be computed automatically. For this purpose, environments containing static obstacles are analyzed and preprocessed in order to identify collision-free areas.^{4,5} Collisions with dynamic obstacles, such as other pedestrians, are then avoided using local reactive methods based on a cell decomposition^{6,7} or repulsive forces.^{8,9} Other approaches solve both navigation and pedestrians inter-collision problems simultaneously,

*Correspondence to: J. Pettré, Virtual Reality Lab (VRlab), Swiss Federal Institute of Technology, CH-1015, Lausanne, Switzerland. E-mail: julien.pettre@epfl.ch

Contract/grant sponsors: Swiss National Research Foundation; Marie Curie Action; contract/grant number: FPS-2004-Mobility-5.

using prioritized motion planning techniques.^{10,11} In such approaches, navigation is planned successively for each entity which then becomes a moving obstacle for the following ones—the growing complexity limits the number of people composing the crowd. The simulation problem can be solved from the individual perception of the environment: behavioral systems range from agent models based on physical laws,¹² particle systems,¹³ or perception-action loops.¹⁴ Such models can be hierarchical in order to satisfy performance requirements.¹⁵ Most advanced crowd simulation systems, as for instance *Massive*TM, *Legion*TM, or *Simulex*TM, implement such models. However, they better fit off-line accurate simulations, for example, for security or architecture applications.

Our main goal is to simulate a large number of people navigating in a believable manner for entertainment applications. The first step is to efficiently solve the navigation problem, mainly addressed by the Robotics community.¹⁶ Among them, the probabilistic roadmaps approach (PRM)¹⁷ can be adapted to plan individual locomotion.^{18,19} Other solutions are capable of planning robot motion on rough terrains.²⁰ Recently, approaches tend to decompose environments into walkable corridors, whose width allows group navigation.²¹ None of these approaches automatically handle both uneven and multi-layered environments, which are commonly encountered in virtual worlds. PRM-based approaches suit high-dimensional problems well. However, they are not the most efficient ones for navigation planning when the problem is reduced to three dimensions. We propose a novel approach capable of decomposing such environments into sets of interconnected cylindrically navigable areas. Terrain analysis is inspired by Voronoi diagrams, and by methods using graphics hardware to compute them.²² The resulting structure captures the environment topology and can be used for solving crowd navigation queries efficiently.

Crowd Visualization

Rendering a digital actor consists in updating the animation, deforming the mesh accordingly, and finally, texturing and lighting it. Brute force execution of these tasks using a dynamic mesh, even with specialized hardware, does not scale to a large crowd. The first optimization is to decrease the geometric model complexity according to the distance from the camera, which is the key idea of a level of detail

approach.²³ In the case where such models are animated using a dynamics-based technique, simulation should be scaled.²⁴ Otherwise, to reduce the cost of animation updates as well as mesh deformations, static or baked meshes can be used^{25–27} where the vertex deformations are precomputed with respect to a set of animations. Another representation of static meshes is through surfels.^{28,29} Image-based rendering bypasses the need for animation updates and mesh deformations as it works directly on the final projected image on the screen. Bill-boarding is the de facto standard for image-based rendering of crowds and was presented in Reference 30]. It is possible to have dynamically updated billboards as presented in Reference [31], where parts of the body define a billboard hierarchy.

Our goal is first to have a large crowd, which makes a billboard approach inevitable, and second, to be able to have close interactions with characters, which requires dynamic meshes at the forefront. The most detailed pedestrians are animated using a motion-capture based technique.^{11,32,33} In our case, a walking motion database is generated from a motion blending-based locomotion engine.³⁴ This database contains walk motions at different velocities and styles; for each pedestrian, the motion having the nearest speed to the current one is chosen. We improve existing approaches^{27,30,35} by enhancing believability through individual animation variety with the use of dynamic meshes.

Structuring Environments

Environments are automatically preprocessed in order to structure them into a navigation graph. This graph supports both path planning and simulation by capturing the environment topology, and distinguishing navigable areas from impassable obstacles and terrains. Navigable areas (graph vertices) are modeled as cylinders with a variable radius. It is possible to go from an area to another one when the corresponding cylinders overlap: a vertical gate (graph edge), at the cylinders intersection, models this connection. Navigation graphs are computed automatically from the environment mesh and a small set of user-defined parameters. The method is capable of handling environments combining both uneven and multi-layered terrains. The required techniques to compute navigation graphs from previously listed inputs are described in

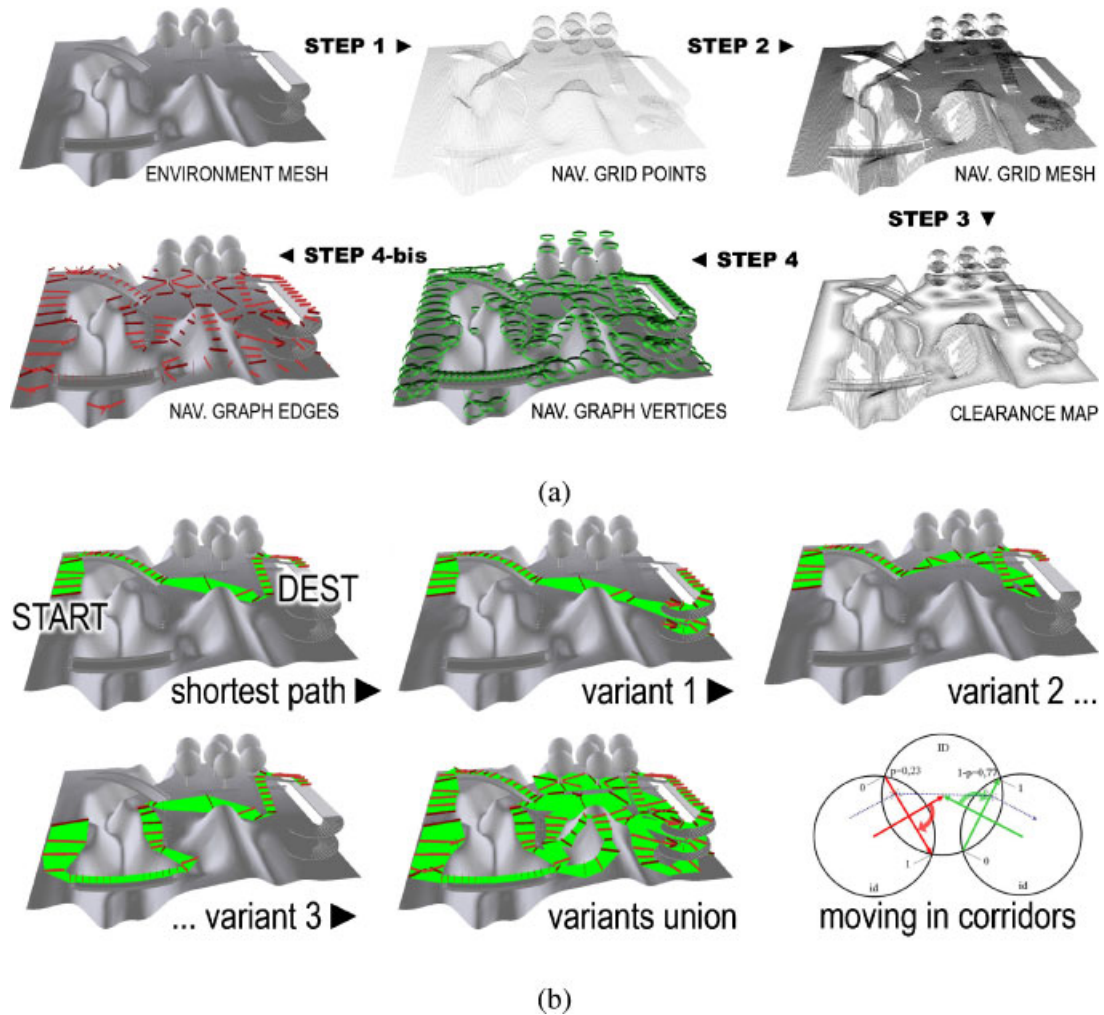


Figure 1. (a) From environment mesh to navigation graph—a 4-stage construction method. (b) Navigation planning: several solution paths are provided to a single query; the graph is pseudo oriented for individualized trajectories (bottom-right image).

Reference [36], and we summarize here the different computation steps, illustrated in Figure 1(a).

- Step 1: Navigation Grid Points.** Navigation grid points are deduced from 3D regular sampling of the environment mesh¹. Grid precision is user-defined. Grid points must correspond to collision-free locations where characters are able to stand: we filter those that are vertically superposed and separated with a too small distance.
- Step 2: Navigation Grid Mesh.** We attempt to connect each point to its neighbors: a connection

represent the ability for a character to move between the two points. This means that the in-between space must be obstacle-free and that the terrain must be flat enough. As a result, each point is potentially connected to its four neighboring points. When a point is connected to three neighbors or less, we deduce that it borders an obstacle or a too slopy area (and therefore it is named 'border point').

- Step 3: Clearance Map.** The clearance of a grid point is the distance to the nearest border point. Thus, the clearance is a lower limit to the distance to the nearest obstacle or slope.
- Step 4: Navigation Graph Construction.** Graph vertices are deduced from the grid points and the

¹Note that this not a 2.5 D elevation map: horizontal coordinates may refer to several altitudes.

clearance: according to the clearance definition, navigability is guaranteed in a cylindrical area which is centered on the considered point and whose radius equals the clearance.

In order to capture the environment topology in a compact manner, we deduce the graph vertices only from a subset of carefully (and automatically) selected points. Edges are deduced from vertices geometry.

Our implementation of the method uses graphics hardware-based operators (for step 1 and 2) and, empirically turns out to be very robust. Many classes of brute-from-design environments were tested: flat, uneven, multi-layered, large/small, inside/outside scenes etc. Due to the sampling done at the first step, computation time (see the Results section for examples) is not much sensitive to the number of triangles composing the environment mesh; the ratio between the environment size and the user-defined precision is the most influent one. The required precision depends on the size of the narrowest passage to be captured (a quarter of this size is a sufficient precision value). Oversampling the geometry is not recommended, as it can result in high-slope angles between some grid points (steps). However, this issue can be fixed by averaging angles and smoothing variations, that is, computing angles by considering n contiguous grid points instead of only two in our current implementation.

Navigation Planning and Crowd Setup

Navigation planning is done when the crowd is created. At this stage, the user can successively define pairs of starting/destination points in the environment. For each one of them, one can define the number of people to navigate in-between. Each submitted pair of points is considered as a navigation query and our goal is to solve it online. This way, the user can compose the whole population of pedestrians interactively. Since pedestrians navigate individually, our navigation planning technique must provide a variety of solutions, that is, for each pair of points, we want the pedestrians to be spread out on different paths joining them. Variety is obtained at two distinct levels. First, a specific path search technique applied on the navigation graph allows to obtain a set of redundant paths joining the two vertices submitted in a query. Each solution path is a sequence of edges, that is, a sequence

of gates to cross, which delimits a corridor between the two vertices. Second, for each path found, we exploit the resulting corridor width to spread the pedestrians again. The following sections describe these two stages.

Multiple Navigation Itineraries

Given a pair of vertices, we want a variety of paths connecting them. We first invoke Dijkstra's algorithm to find the shortest one. This shortest path is a sequence of gates of various widths to cross. We then assume that a congestion point appears along the path where the thinnest gate is present. Therefore, we edit and increase the corresponding edge cost² and allow a new Dijkstra search, resulting in a new path. This is the key idea of our specific path search technique. We iterate this process until no more edge costs can be modified (only one cost modification per edge is allowed). Figure 1(b) displays several itineraries obtained with this method, for the query of going from the left side of the environment up to the footbridge. After the shortest path is found (top-left image), many variants are discovered: their total union is also shown.

Moving in Corridors

Each pedestrian chooses one of the previously computed paths to reach his destination. The chosen path is individually transformed into a way-point sequence according to a parameter $p \in [0, 1]$. On each gate to cross, a way-point is generated. Its exact position on the gate, from left to right, is linearly dependent on p . As shown in Figure 1(b), bottom-right image, it is easier to compute way-point coordinates if gates are oriented, thus allowing to distinguish the left from the right according to the crossing direction. We introduce a pseudo-orientation of the graph based on vertex numbering. For gates crossed in the reverse direction, $1-p$ is used instead of p to compute the way-point location. In Figure 1 (bottom-right image), the vertex in the middle has a greater index than the two other ones, implying such an operation on p . The Results section demonstrates the ability of the method to plan the navigation of thousands of pedestrians in real time.

²Initially, the cost of each edge is set to the distance between the two centers of the linked vertices. When increased, cost is multiplied by 10.

Real-Time Crowd Simulation

The user has previously designed a moving crowd by choosing pairs of locations in the environment and a number of pedestrians navigating in-between. The role of the simulation is to update each pedestrian's situation in real time according to the user's directives. Once two locations have been chosen, the pedestrian will continue to navigate between them indefinitely. Our goal is to preserve a maximum of computation time for crowd rendering. As a result, our simulation model is simple, mainly based on local instantaneous densities of population. In order to handle a large number of pedestrians, simulation is scalable: quality is enhanced where attention is focused.

Simulation Model

The simulation model has three roles. The first role is to spread pedestrians among the different available paths between the two locations they must attain. As pedestrians navigate endlessly between these two locations, a new path is chosen each time they reach one of them and must go back to the other one. A corresponding set of individual way-points is computed, according to the individual parameter p . The second role of the simulation is to drive pedestrians to follow the way-point sequence, and finally, its third role is to avoid collisions between them.

To reach real-time performance, the simulation model is essentially based on local instantaneous population densities which are fast to compute. We use the space partition provided by the navigation graph vertices to evaluate local densities.

Way-points are always placed within a gate. As a result, each time a pedestrian reaches his next way-point, he transits from an area covered by a vertex to another one. Thus, it is both easy and efficient to maintain the list of pedestrians currently navigating in a given vertex area. Local densities first influence the pedestrians' walk velocity. An empirical law from literature relates walk velocity and population density.³⁷ This relation is also used to compute path costs: given the population density, we evaluate the required time to cross each vertex area and then deduce the complete travel time. Each time a pedestrian has to choose a path, he follows the one which currently has the shortest traveling time. Finally, a potential field-based method allows to avoid collisions between pedestrians: way-points are attractive while pedestrians repulse each

other, as introduced in Reference [8]. Despite the simulation model simplicity, the potential size of crowds remains limited. To break limitations, scalability is required. The next section describes how levels of simulation are introduced.

Levels of Simulation

Levels of simulation (LOS) allow to distribute the available computation time spatially and temporally according to the spectator's point of view: its quality is enhanced where attention is focused and progressively decreases toward invisible zones. First, we distribute levels of simulation for each navigation graph vertex according to the point of view, as illustrated in Figure 2. An exact definition of simulation level distribution is not the point of this section. Note that the highest simulation quality is around the view point. However, it remains high in front of the user, even at far, because the human eye is sensitive to motion continuity; on the side views, quality is progressively decreased until it becomes the lowest one in invisible areas.

In invisible areas, the pedestrians' situation is updated at low frequencies, and only represented by a progression parameter. Elsewhere, different methods are used to steer pedestrians toward way-points: a linear steering is used for low-quality levels, while smoother trajectories are produced using the method proposed in Reference [3] for higher levels. Pedestrians' elevation computation is also scaled: the previously computed navigation grid is used to deduce elevations from a nearest grid point search or a bilinear interpolation. Walk velocities are related to local population densities. In highly detailed simulation levels, densities of neighboring areas are taken into account to obtain smooth accelerations. Thus, when a pedestrian is close to exit from an area, the density of the next one becomes progressively more important.

Finally, inter-collisions are checked between a limited number of pedestrians. Indeed, at a far distance, collisions are hardly detectable. As a result, we only solve them where the simulation level is the highest. Nevertheless, if a congested zone is observed, taking into account all the pedestrians contained in such high-level vertices would result in a prohibitive computation time. Thus, among them, only those that are nearer than a specified distance are selected. This distance is inversely proportional to the local population density. Note that we use the list referencing pedestrians for each vertex once again to execute all previous tasks efficiently.

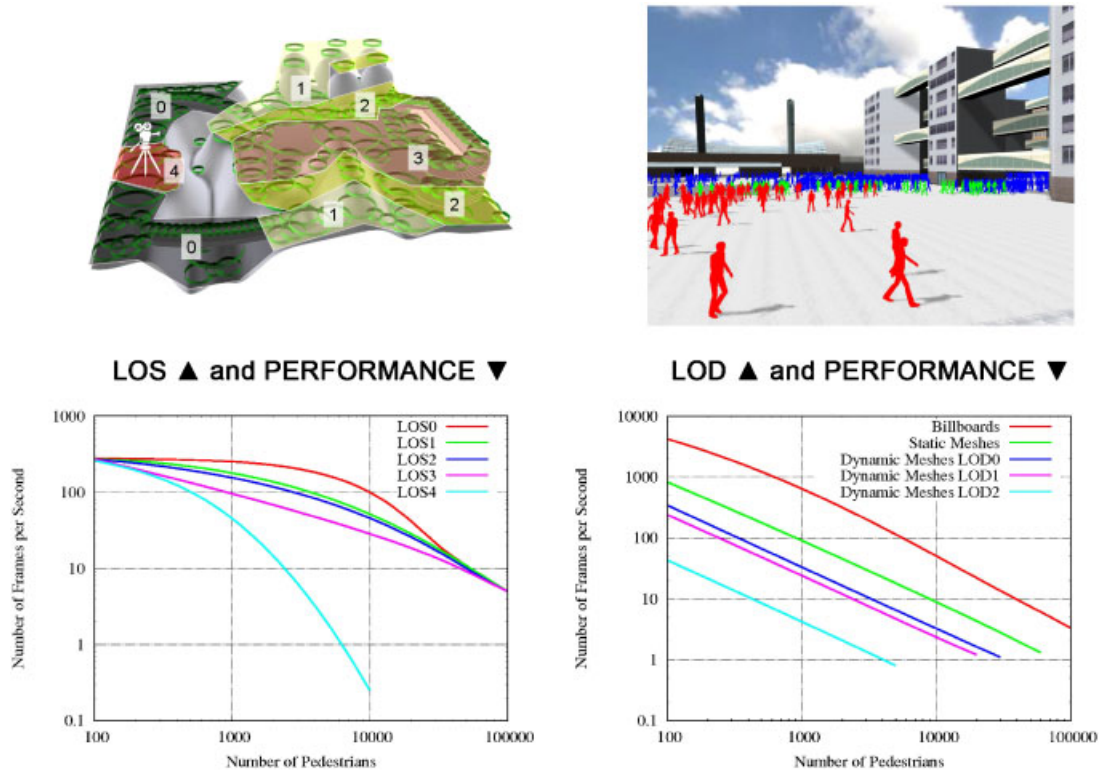


Figure 2. Distribution of the level of simulation and of the level of details (top images), and performance (bottom graphs).

Visualization

The goal of the real-time crowd visualizer is to render a large number of entities according to the current simulation state, which provides the position, orientation, and velocity for each individual. System constraints are believability, real-time updates (25 frames per second) and a number of digital actors ranging in the tens of thousands. We make these actors believable by varying their appearance (textures and colors) and animation. Their graphical representation is derived from a template as in Reference [38], which holds all the possible variations. Thus, with only a limited set of such templates, we can achieve a varied crowd, leading to considerable time savings for designers.

- a set of textures in gray scale (except for the skin) identifying color modulation areas (pants, shirt, hair etc.),
- a skeleton (kinematic structure),
- a corresponding animation database as skeletal orientations (here 1000 different walk cycles generated using a motion blending-based locomotion engine as presented in Reference [34]).

Each human in the visualization system is called an instance and is derived from a template. Individualization comes from assigning a specific gray scale texture and a color combination for each identifiable region. Instances have individualized walk velocities and are animated by blending the available walk animations.

Pipeline

The rendering pipeline advances consecutively in four steps. The first one consists in culling, that is, determining visibility, and choosing the rendering fidelity for each simulated human, as seen in Figure 3, top-right image. By re-using the information stored in the navigation graph of the simulation system, this task

Templates

A template is defined as:

- a set of three meshes with decreasing complexity (LODs),

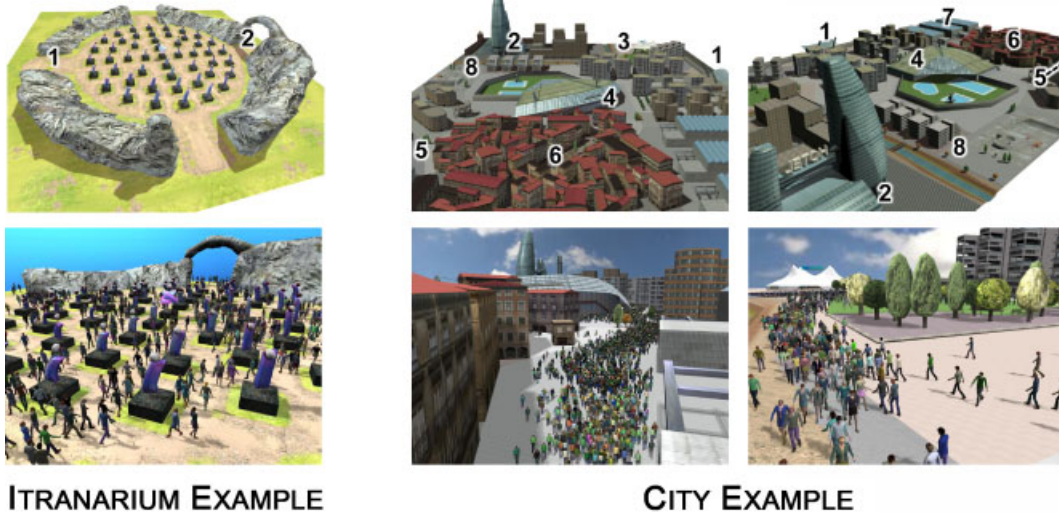


Figure 3. Examples.

is not done for each individual but at the vertex level (for which the LOS is available), thereby determining fidelities for a whole subset of characters at once. During this phase, humans are distributed in three different groups according to their fidelity level, which ensures efficient batched rendering.

The next step of the pipeline is the rendering of dynamic meshes, which are the most detailed fidelity capable to interpolate animations based on skeletal postures. According to the current instance state (linear and angular walk velocities and time), animations are retrieved from the database and interpolated, yielding a smooth animation, with continuous variations of velocities, and no foot-sliding. The resulting skeletal posture is sent to a hardware vertex shader and fragment shader deforming and rendering the human on the graphics card.

Then, static meshes (also called baked or pre-deformed) constitute the second rendering fidelity, which keeps a pre-transformed set of animations using the lowest resolution mesh of the deformed mesh in the previous step. Pre-computing deformations allows substantial gains in speed, but constrains the animation variety and smoothness.

The final rendering fidelity is the billboard model which, compared to previous approaches, uses a simplified scheme of sampling and lighting. World-aligned billboards³⁹ are used, with the assumption that the camera will never hover directly above the crowd. Thus, only sampled images around the waist level of the character are needed. In our case, the templates are sampled at 20 different angles, for each of the 25 key-

frames composing a walk animation. When constructing the resulting texture, the bounding box of each sampled frame is detected to pack them tightly together. When rendering billboarded pedestrians, a specificity of our technique is to apply cylindrical lighting instead of using normal maps: each vertex normal is set to point in the positive Z direction, plus a small offset on the X axis, so that it points slightly outside the frame. We then interpolate the light intensity for each pixel in the fragment shader.

Results

We have implemented the previously described architecture and experimented it on several examples. After presenting performance measures for both simulation and rendering tasks, we illustrate our results in three different environments and setups. One should refer to the accompanying video to evaluate the results.

Performance

The graphs in Figure 2 illustrate the performance of our application. The tests were measured on a desktop computer³. The system is able to simulate a large number of pedestrians with real-time performance. For 1000 pedestrians or less, the lowest quality rendering levels are not even required. Interactive frame rates

³Pentium Xeon 3.2 GHz, 1 GB of memory, and a nVidia GeForce 7800 GTX (256 MB video memory).

(10 fps) are obtained with up to 30 000 zoomed-out visible pedestrians.

The graph computation time mainly depends on the environment size and the defined precision: indeed, in the case of the city, the intermediate navigation grid is made of a large number of points and many distance computations have to be performed to deduce the clearance map. Given the complexity of Dijkstra's algorithm, path planning computation time essentially depends on the number of edges and vertices composing the graph.

Itranarium

In this first example, we illustrate the simulation model abilities to efficiently spread a crowd of people navigating between the same pair of starting/destination points. The environment is introduced in Figure 3: the Itranarium is a crowded touristic place where 1000 pedestrians walk between the indicated points 1 and 2. The environment topology allows many itineraries. The average frame rate during video capture is of 30 fps. First, we observe that people are walking everywhere, meaning that paths computed at the navigation planning step cover the whole environment. Second, we note that the simulation model allows a good distribution of people on paths (see the accelerated part of the accompanying video).

City

This second example illustrates a large urban environment, where we define eight significant destinations. A population of 35 000 pedestrians is composed of 7 groups (of 5000 pedestrians each) navigating back and forth between pairs of previously defined locations. Both simulation and rendering use all the available levels of detail. Due to path variety, even with such a reduced set of destinations, pedestrians are spread and no place remains empty. During exploration, the frame rate ranges from 10 (for large views of dense areas) to 20 fps. As expected, places defined as significant destinations are more densely populated than others. Mandatory passages such as the bridges around the hotel, are highly crowded.

Discussion and Future Work

Future work directions include quality and performance improvements with the possibility of handling larger

crowds and environments. First, the navigation graph usage is limited to static environments. However, addressing a dynamic one is feasible by splitting or disabling vertices where new obstacles appear or move. The challenging point is to reconfigure the planned navigation interactively for a potentially high number of moving pedestrians. Second, environments that are both highly complex and large may result in high-dimensional navigation graphs. Indeed, navigable areas can only be modeled as cylinders in our current solution. We aim to explore two directions to solve this limitation: the first one is to introduce new shapes for delimiting areas, the second one is to introduce hierarchical graphs with at least distinct levels for the topology of the geometry.

Behavior is limited to locomotion. To allow interactions between users and digital actors, the behavioral model of pedestrians should be improved. Also, path decisions are taken before leaving a starting point and not reconsidered until the destination is reached. This occasionally leads to unrealistic behaviors, where people wait for a specific passage to be freed while the surrounding ones are less crowded.

Integrating static objects as occluders would improve visibility culling results, and lead to a better distribution of levels of simulation. For rendering, the fidelity levels are set at static distances and should instead adapt either to the number of pedestrians currently visible on the screen, or to the local population densities. So, in the case where only a few pedestrians are visible, they would all have high-quality levels, even far from the camera.

In conclusion, we have presented a solution for simulating and rendering large navigating crowds in virtual environments. This solution is based on an automatic geometric analysis resulting in a navigation graph which captures the environment topology. The graph sustains a navigation planning technique specifically made to provide varied solutions, a scalable simulation of pedestrians, and an efficient pedestrian culling before rendering. The crowd rendering engine improves existing solutions by enhancing fidelity. Thanks to dynamic and static meshes at the forefront, we obtain high-quality animation and appearance. We have validated our solution on several examples.

ACKNOWLEDGMENTS

The authors thank Sébastien Schertenleib for technical support, Renaud Krumme-nacher and Mireille Clavien for designing the environments and Helena Grillon for proofreading the paper.

References

1. Reynolds CW. Flocks, herds, and schools: A distributed behavioral model. *ACM Computer Graphics* 1987; **21**(4): 25–34.
2. Hodgins JK, Wooten WL, Brogan DC, O'Brien JF. Animating human athletics. *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* 1995; **29**: 71–78.
3. Reynolds CW. Steering behaviors for autonomous characters. *Proceedings of Game Developers Conference*, 1999; pp. 763–782.
4. Shao W, Terzopoulos D. Autonomous pedestrians. *SCA'05: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005; pp. 19–28.
5. Lamarche F, Donikian S. Crowds of virtual humans: a new approach for real time navigation in complex and structured environments. *Eurographics '04: Computer Graphics Forum* September 2004; **23**(3): pp. 509–518.
6. Gipps PG, Marksjo B. Micro-simulation model for pedestrian flows. *Mathematics and Computers in Simulation* 1985; **27**: 95–105.
7. Klüpfel H, Meyer-König T, Wahle J, Schreckenberg M. Microscopic simulation of evacuation processes on passenger ships. *Proceedings of Fourth International Conference on Cellular Automata for Research and Industry*, 2000; pp. 63–71.
8. Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research* 1986; **5**(1): 90–98.
9. Helbing D, Molnar P. Social force model for pedestrian dynamics. *Physical Review* 1995; **51**(5): 4282–286.
10. Lau M, Kuffner J. Behavior planning for character animation. *SCA'05: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005; pp. 271–280.
11. Sung M, Kovar L, Gleicher M. Fast and accurate goal-directed motion synthesis for crowds. *SCA'05: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005; pp. 291–300.
12. Helbing D, Farkas I, Vicsek T. Simulating dynamical features of escape panic. *Nature* 2000; **407**: 487–490.
13. Bouvier Eric, Guilleateau Pascal. Crowd simulation in immersive space management. In *Proceedings of the Eurographics workshop on Virtual environments and scientific visualization '96* Springer-Verlag: London, UK, 1996; pp. 104–110.
14. Terzopoulos D. Artificial life for computer graphics. *Communications of the ACM* 1999; **42**(8): 32–42.
15. Musse S, Thalmann D. A behavioral model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics* 20017(2): 152–164.
16. Latombe J-C. *Robot Motion Planning*. Kluwer Academic Publishers: Boston, 1991.
17. Kavvaki L, Svestka P, Latombe J-C, Overmars M. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Proceedings of IEEE Transactions on Robotics and Automation*, 1996; pp. 566–580.
18. Choi MG, Lee J, Shin SY. Planning biped locomotion using motion capture data and probabilistic roadmaps. *SIGGRAPH'03: ACM Transactions on Graphics* 2003; **22**(2): 182–203.
19. Pettré J, Laumond JP, Simeon T. A 2-stages locomotion planner for digital actors. *SCA'03: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003; pp. 258–264.
20. Hait A, Siméon T, Taïx M. Algorithms for rough terrain trajectory planning. *Advanced Robotics* 2002; **16**(8): 673–699.
21. Kamphuis A, Overmars MH. Finding paths for coherent groups using clearance. *SCA'04: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004; pp. 19–28.
22. Hoff K III, Keyser J, Lin M, Manocha D, Culver T. Fast computation of generalized voronoi diagrams using graphics hardware. *SIGGRAPH'99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999; **33**: pp. 277–286.
23. Hoppe H. Progressive meshes. *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 1996; pp. 99–108.
24. Safonova A, Hodgins JK, Pollard N. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *SIGGRAPH'04: ACM Transactions on Graphics* 2004; **23**(3): 514–521.
25. Ulicny B, de Heras P, Thalmann D. Crowdbrush: interactive authoring of realtime crowd scenes. *SCA'04: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004; pp. 243–252.
26. Coic J-M, Loscos C, Meyer A. Three LOD for the realistic and real-time rendering of crowds with dynamic lighting. *Research Report LIRIS, France*, 2005.
27. Dobbyn S, Hamill J, O'Connor K, O'Sullivan C. Geopostors: a real-time geometry/impostor crowd rendering system. In *IBD'05: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2005; pp. 95–102.
28. Pfister H, Zwicker M, van Baar J, Gross M. Surfels: surface elements as rendering primitives. *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, 2000; pp. 335–342.
29. Wand M, Straßer W. Multi-resolution rendering of complex animated scenes. *Eurographics'02: Computer Graphics Forum* 2002; **21**(3): 483–491.
30. Tecchia F, Loscos C, Chrysanthou Y. Visualizing crowds in real-time. *Computer Graphics Forum* 2002; **21**(4): 753–765.
31. Aubel A, Boulic R, Thalmann D. Real-time display of virtual humans: levels of details and impostors. *IEEE Transactions on Circuits and Systems for Video Technology* 2000; **10**(2): 207–217.
32. Rose CF, Cohen MF, Bodenheimer B. Verbs and adverbs: multidimensional motion interpolation. *IEEE Computer Graphics and Applications* 1998; **18**(5): 32–40.
33. Kwon T, Shin SY. Motion modeling for on-line locomotion synthesis. *SCA'05: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005; pp. 29–38.
34. Glardon P, Boulic R, Thalmann D. Robust on-line adaptive footplant detection and enforcement for locomotion. *The Visual Computer* 2006; **22**(2): 194–209.
35. Pearce D, Ryder G, Lapeer RJ, Day AM. Exploiting partial visibility for optimised crowd scene rendering. *SCA'04: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004; pp. 34–35.
36. Pettre J, Laumond J-P, Thalmann D. A navigation graph for real-time crowd animation on multilayered and uneven

terrain. In *Proceedings The First International Workshop on Crowd Simulation (V-CROWDS'05)*, Lausanne, Switzerland, 2005. pp. 81–89.

37. Weidmann U. Transporttechnik der fussgänger. *Schriftenreihe des IVT, ETH Zürich*, 1992.
38. Yersin B, Maïm J, de Heras Ciechowski P, Schertenleib S, Thalmann D. Steering a virtual crowd based on a semantically augmented navigation graph. In *Proc. The First International Workshop on Crowd Simulation (V-CROWDS'05)*, Lausanne, Switzerland, 2005 pp. 169–178.
39. Möller TA, Haines E. *Real-Time Rendering*. 2002; AK Peters, Ltd.



Jonathan Maïm is the Ph.D. student at VRLab at the Swiss Federal Institute of Technology in Lausanne (EPFL). In April 2005, he received his Masters Degree in computer science from EPFL after achieving his Masters Project at the University of Montreal. His research efforts are currently concentrated on crowd animation and rendering.

Authors' biographies:



Julien Pettré is currently a postdoctoral fellow at the VRLab of the EPFL. He qualified as an engineer in 1998 and received his Ph.D. in Robotics from Toulouse III University in 2003, after a 4-year research at LAAS-CNRS. His research interests include digital actor animation: motion capture edition, motion planning, and crowd simulation.



Barbara Yersin has achieved her Masters project at the University of Montreal, after which she has received her Master in Computer Science in March 2005 from EPFL (Swiss Federal Institute of Technology in Lausanne). She is currently a Ph.D. student at the VRLab, EPFL. Her main interests are in computer graphics, particularly crowd simulation and animation.



Pablo de Heras Ciechowski finished his Ph.D. 'Rendering Massive Real-Time Crowds' in June 2006, under the supervision of professor Daniel Thalmann in VRLab at the Swiss federal institute of technology in Lausanne (EPFL). He finished his M.S. in computer science and engineering at Lund Institute of Technology (LTH) in February 2002 in Sweden. His Masters thesis was done in collaboration with Massive Entertainment, a game company in Malmö. His interests are in real-time rendering optimizations for large dynamic explorative worlds such as crowds of humans.



Jean-Paul Laumond is Directeur de Recherche at LAAS-CNRS in Toulouse, France. In Fall 1990, he has been invited senior scientist from Stanford University. He has been a member of the French Comité National de la Recherche Scientifique from 1991 to 1995. He is currently a member of the board of the ACI Neurosciences Intégratives et Computationnelles. He has been coordinator of two European Esprit projects PROMotion (1992–1995) and MOLOG (1999–2002), both dedicated to robot motion planning technology. In 2001 and 2002, he created and managed Kineo CAM, a spin-off company from LAAS-CNRS devoted to develop and market motion planning technology. Kineo CAM was awarded the French Research Ministry prize for innovation and enterprise in 2000. He teaches Robotics at ENSTA and Ecole Normale Supérieure in Paris. He has edited three books. He has published more than 100 papers in inter-

national journals and conferences in computer science, automatic control and robotics.



Daniel Thalmann is professor and director of The Virtual Reality Lab (VRLab) at EPFL, Switzerland. He is a pioneer in research on virtual humans. His current research interests include real-time virtual humans in

virtual reality, networked virtual environments, artificial life, and multimedia. He is coeditor-in-chief of the *Journal of Computer Animation and Virtual Worlds* and member of the editorial board of the *Visual Computer* and four other journals. Daniel Thalmann was member of numerous program committees, co-chair, and program co-chair of several conferences including IEEE VR 2000. He has also organized five courses at SIGGRAPH on human animation and crowd simulation. Daniel Thalmann has published numerous papers in graphics, animation, and virtual reality. He is coeditor of 30 books included the recent *“Handbook of Virtual Humans”*, published by John Wiley and Sons and coauthor of several books. He received his Ph.D. in Computer Science in 1977 from the University of Geneva and an Honorary Doctorate (Honoris Causa) from University Paul-Sabatier in Toulouse, France, in 2003.