EG UK Theory and Practice of Computer Graphics (2010)
John Collomosse, Ian Grimstead (Editors)

Agent-based Large Scale Simulation of Pedestrians With Adaptive Realistic Navigation Vector Fields

Twin Karmakharm , Paul Richmond and Daniela M. Romano[1]

[1] University of Sheffield Department of Computer Science,211 Portobello, Sheffield, S1 4DP

Abstract

A large scale pedestrian simulation method, implemented with an agent based modelling paradigm, is presented within this paper. It allows rapid prototyping and real-time modifications, suitable for quick generation and testing of the viability of pedestrian movement in urban environments. The techniques described for pedestrian simulation make use of parallel processing through graphics card hardware allowing simulation scales to far exceed those of serial frameworks for agent based modelling. The simulation has been evaluated through benchmarking of the performances manipulating population size, navigation grid, and averaged simulation steps. The results demonstrate that this is a robust and scalable method for implementing pedestrian navigation behaviour. Furthermore an algorithm for generating smooth and realistic pedestrian navigation paths that works well in both small and large spaces is presented. An adaptive smoothing function has been utilised to optimise the path used by pedestrian agents to navigate around in a complex dynamic environment. Optimised and un-optimised vectors maps obtained by applying or not such function are compared, and the results show that the optimized path generates a more realistic flow.

Categories and Subject Descriptors (according to ACM CCS): I.2.11 [Distributed Artificial Intelligence]: Multiagent systems

1. Introduction

Pedestrian simulations have found increasing use in various fields from entertainment to disaster planning [Sof10,Sim10, TP02] where they can be used to predict and model the effect of crowd movement under various environmental conditions. As the population densities of urban areas continuous to increase, it is likely that pedestrian simulations will play a progressively more important role in ensuring safe and optimised crowd flows within urban spaces. Simulating groups of pedestrians is now almost exclusively performed by considering an individual approach, where agents interact directly to form emergent group behaviour, rather than using a top down set of equations to attempt to model group dynamics. This individual emphasis, defined as either an agent simulation or particle force simulation lends itself well to the natural description of behaviour through sets of defined rules. Likewise, heterogeneity can be introduced within the simulation as each pedestrian is a unique entity, responsible for their own actions and personal traits. In addition to simulating the interaction between agents, pedestrian simulations also require an additional technique to model the flow of pedestrians through the environment. This can be achieved using either a global navigation technique (common to all agents) or through individual path planning.

Whilst the individual approach to agent simulation has many advantages it poses a significant computation problem when considering large scale simulations. Due to the need to simulate ever increasingly large populations sizes it has become apparent that traditional serial simulation frameworks are therefore unable to meet the real time requirement needed for visual exploration with such models. Parallelisation offers a solution to this issue, however hardware architectures such as the Graphics Processing Unit (GPU) are far from easy to develop with and require specialist knowledge to gain maximum performance even when taking into account new development tools. With respect to agent navigation, previous techniques tend to diverge away from the agent based semantics meaning that navigation becomes a separate process which in most cases remains static throughout the simulation.

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG10/067-074

This paper takes an agent based approach to modelling pedestrian systems and navigational behaviour where both the interaction between agents and the agent's navigation through the environment is facilitated through a single agent framework. The advantage of this technique is that we are able to integrate our model within an agent based framework designed for efficient GPU simulation allowing us to simulate large population sizes in real time. In addition the simulations can be rendered efficiently as simulation data is already contained within the GPU device. Within this paper the navigation is based upon a set of agents arranged within a discrete grid. The use of these agents to indicate a vector field is described as is a novel process for avoiding a common issue we identify as "diagonal convergence" which is the result of generating vector fields using shortest path flows throughout the environment. An evaluation is also presented which is defined according to a real world environment and used by the simulation to test the performance and behaviour of our system.

2. Related Work

The use of agent based modelling has been widely adopted within the field of pedestrian simulation. An early example of this concept was demonstrated by Reynolds on the simulation of flocking behaviour of animals [Rey87]. Operating on only three simple rules and awareness of their environment, the agents displayed emergent complex and plausible group behaviour. A model for pedestrian behaviours was later conceived by Helbing et al [Hel91] which takes into consideration psychological factors such as social distances from other pedestrians as well as environmental obstacles. Within this particle force model, widely observed organisational behaviours of pedestrians such as lane formation emerge from the individual pedestrian behaviours The original model was later extended to include contact forces which simulate aspects of emergency situations [HFV00]. This well studied and validated model is now standard in most evacuation models and continues to form the basis for extended research into pedestrian behaviour e.g. Moussaïd et al takes in to account of the shapes small groups form in order to maximise their social communication [MPG*10].

Pedestrian agent modelling technique originally used Cellular Automata [BKSZ01], a discrete model where each pedestrian occupies a single cell within a grid. The order of their movements must be sequential which makes it harder to parallelize, they are also not suitable for smooth pedestrian flows in continuous environments. The continuum crowd technique, initially proposed by Huges [Hug02], defines crowds as a density field where pedestrians travel to their goals by following the gradient vectors calculated by the potential functions. This model was improved by Treuille et al. [TCP06] to incorporate empirically measured emergent pedestrian behaviours such as lane formation. Since the density field takes into account of both global goals and local obstacles including other pedestrian agents, global and local navigation is integrated and resolved at the same time. The most intensive calculation with this process is in resolving the density field with every time-step of the simulation and hence the method may not scale well for much larger environments. The solution for the scalability was addressed by Shopf et al. in the March of the Froblins demo [SBOT08] where continuum methods were instead used as a coarse density grid to drive global navigation and using agent based collision detection behaviour for local obstacles.

With respect to navigation local perceptive interaction does not provide enough information to provide long range planning through the virtual environment. As a result either a global or individual based navigation technique is required to provide long range guidance. Most global navigational techniques reduces the environment down to a set of discrete grid cells [TLCC] [ST05]. Information about obstacles occupying the cell, terrain height, and other arbitrary information can be encoded within each cell where pedestrian agents can then use to navigate within the virtual environment. Most commonly the discrete cells can be used to hold a directional force which when combined with neighbouring cells represents a vector field (also known as force fields or flow fields). This technique is very effective when combined with good local collision avoidance and computationally very efficient as each agent avoids having to perform a path planning exercise. The authoring of the vector fields can either be done manually using a software that allows a user to 'paint' the field directions, alternatively they can be automatically generated using algorithms such as wavefront propagation where the origins of the waves represent the destination(s) of the pedestrians [LaV06]. Multiple fields can be used for each group(s) of pedestrians to represent differing goals and introduce complex flows. The drawback with this method is the fact that the method often uses more memory, growing exponentially with the size of the grid and can then also place limits on the number of goals or fields that can be used. The flow tiles concept introduced by Chenney [Che04] makes the process more scalable by re-using tiles of pre-defined vector fields to save on repetitious regions of flow that are likely to occur within the environment. Work by Jin et al. [JXW*08] presents the uses of continuous vector fields constructed using radial basis functions. Anchor points are used in order to define directional flows of pedestrian agents and can be placed dynamically on the scene making crowd control interactive. Since the field is continuous, performance is dependent on the number of anchor points rather than the size of the environment and has been shown to degrade well with increasing pedestrian sizes. The author does not however, discuss how it well it would apply to more complex environments and goal based situations.

The alternative to global navigation is the use of individual path planning. Whilst simple ray query technique can be used to find paths around obstacles it would be too computationally expensive to apply it to global navigation in real-

time. To avoid this problem, the inter-connecting areas can be reduced to a set of graphs where each node represents an area and edges represent walkable paths [Sha05]. The problem then changes to a graph traversal problem which can be solved by common path-finding algorithms such as A* or Dijkstra's. Pettre et al. [PLT05] shows alternative approach to the graph based navigation where medial axis sampling was performed on the environment enabling it to be conceptualized as a series of interconnected circles. In this case the radius of each circle is its maximum clearance from static obstacles and the radical lines from intersecting circles are used to represent corridors of safe walkable paths. While graph based methods allows highly individualised goal destinations, they require more time during the simulation to calculate routes dynamically.

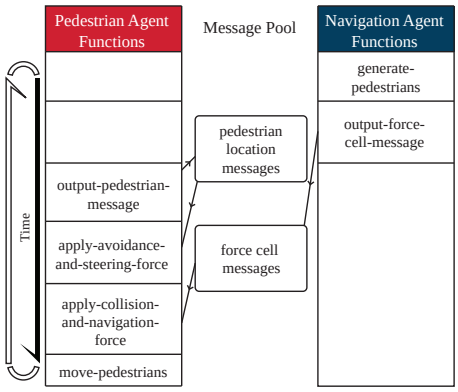

Figure 1: *Sequence diagram for function call of both agents and their utilization of the message pool.*

3. Our Pedestrian Simulation System

3.1. The Agent Modelling Framework

In order to implement our pedestrian simulation as an agent model on the GPU we have decided to use the Flexible Large-scale Agent Modelling Environment framework for the GPU (FLAME GPU) [RCR09]. As well as the Compute Unified Device Architecture (CUDA) [Nvi09] enabled GPU version of the framework used within this paper there are a number of simulation backends for hardware architecture including a single core CPU and a number of common High Performance Computing (HPC) grid architectures. The framework is incredibly robust and has been used in a wide range of agent modelling areas that range from the biological science [RWCR10] through to modelling of the European economy [DVD08]. This framework itself uses a formal model of agents, which is based upon a parallel communicating state machine with internal memory more commonly known as a communicating stream X-Machine [CSH06]. Models are described using both a simple XML model definition and a corresponding set of scripted agent functions which defines agent behaviour. An XML model file briefly comprises of the following information:

1. A set of environment, or constant variables, which may be set between simulation steps - These are useful in controlling aspects of the model within a real time simulation.
2. A set of X-Machine agents - As with the formal definition of an X-Machine [Eil74] each agent definition consists of a description of the agents internal memory (the agent variables), a set of internal states (which may add a degree of diversity to an agent population) and a set of function definitions (which define the transition between any two states including any agent or message input or output). Each agent may be either discrete (in which case it is non mobile and part of a cellular automaton) or continuous in which case it may be spatially distributed or represent a more abstract non spatial entity.
3. A set of messages - Each message definition has a set of communicated information (or variables) which may be input or output from within the agent function scripts allowing indirect communication between agents. A message definition also defines any restriction on message range or the type of agent which may use it (discrete or continuous). This ensures the back end simulation uses the most efficient communication algorithm for message iteration within the agent functions.
4. A set of function layers - This describes the sequential stepwise order of execution of the agent functions within a single simulation step. A single layer may execute more than one agent function in parallel if they share no dependencies i.e. do not share any common message input or output. In the case of the GPU these are executed sequentially regardless due to the Single Program Multiple Data(SPMD) architecture.

Model descriptions are processed within FLAME GPU through a template engine which translates the model file into CUDA source code. This process abstracts the writing of GPU code from the modeller allowing fast prototyping of models without explicit knowledge of the GPUs data parallel architecture. Common agent based functionality such as birth and death allocation, communication and efficient data storage are automatically generated by the template engine and which combined with the agent function scripts can be compiled to produce GPU enabled agent simulations. The processes can be easily extended and in the case of our pedestrian simulation a visualisation template is used as a starting point to allow more advanced pedestrian simulation which includes a parallel LOD rendering technique [RR08], inclusion of detailed urban models and an interactive user interface for controlling environment variables.

In accordance with the FLAME GPU model specification technique our pedestrian simulation is expressed using two distinct type of agents, the pedestrian agents and the navi-

gation agents. Figure 1 illustrates the sequence of functions that each agent performs and how the information flow in and out of these functions are managed.

3.2. Navigational Agents

In our work, we have opted to use a variation of the vector field based method, force vector fields (FVFs) in order to assist the pedestrian agents in their global navigation. Collision and Navigation FVFs are the two distinct types that are employed in our simulation (abbreviated to CFVF and NFVF respectively). CFVFs encode repulsion forces radiating from static obstacles within the environment. The strength of the force at any location is exponentially inversed in proportion to the distance from said obstacles over a given configurable range. Only one CFVF is required for the simulation as repulsion force from obstacles is continuous and static throughout the simulation. NFVFs are then used to guide agents to their intended goals. In our case all of the goals are associated with an entrance/exit to the environment and the number of NFVFs corresponds to the number of exits. Each cell within a field is encoded with a force that points to the shortest non-obstructed direction to the goal. Details for CFVF and NFVF are further discussed in 3.2.1 and 3.2.2 respectively.

To represent these fields, our Navigation agents are made to be Cellular Automaton (CA), i.e. non-mobile, discrete and are arranged as a topologically correct grid. Each navigation agent can then be thought of representing a cell within a grid and so holds information for all CFVF and NFVFs for that particular location. In our current simulation, the Navigation agents have two functions, generate-pedestrians and output-force-cell-messages (Figure 1). When the generate-pedestrian function is called, if the navigation agent is deemed to be an exit it will randomly generate pedestrian agents at that location at a configurable (people per minute) rate. Newly generated pedestrian agents are then assigned a random exit goal according to a probabilistic function. Since all FVFs are currently generated before the start of the simulation, at run-time navigation-agents need only to broadcast information of their respective FVF cell. This is handled by the output-force-cell-message function.

An example of a single broadcast message can be found in Listing 1 which was taken from the initial configuration file created before the simulation starts. The variable height is used to encode a discrete height map of the simulated environment and is used within the visualisation to displace pedestrian agents across differing physical levels in the simulated space. The variables *collision_x* and *collision_y* encodes a CFVF cell. In this case a force in the positive x direction indicates an obstacle to the left of the navigation agent. The variables *exit*0_*x* through to *exit*1_*y* describe 2 normalised navigation force vectors which describe force direction leading agents towards one of 2 exits. Finally the variable exit is used to determine if the force cell indicates

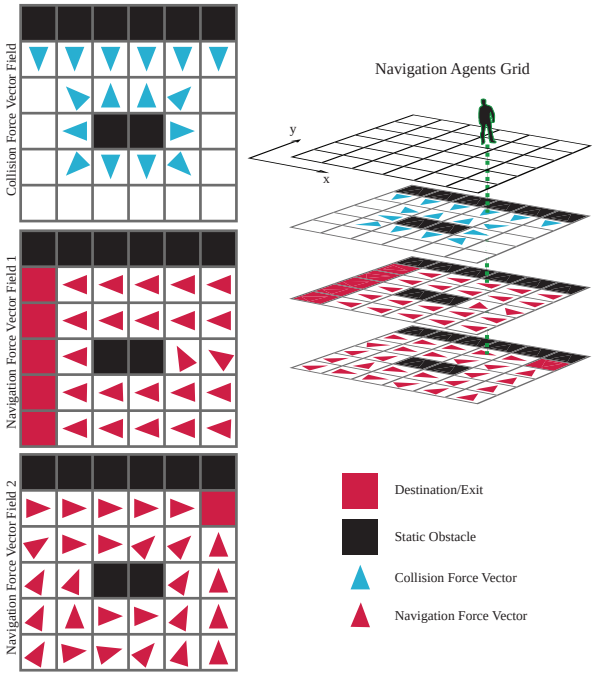

Figure 2: *An example of a simplistic 6x6 environment with 2 exits and obstacles. Navigation agents are arranged in a grid as shown (top right) and position of the agent relates to a particular cell of the force vector fields.*

the presence of an actual exit position within the simulation. Exit positions play two important roles within the simulation. The first is the previously described generation of new pedestrian agents. The second is ensuring pedestrian agents can identify (through message communication of the cell information) that they have successfully reached their destination point.

Listing 1: *XML Specification of navigation agents*

```
<xagent>
<name>force_cell</name>
<x>1</x>
<y>2</y>
<height>0.1</height>
<collision_x>1.0</collision_x>
<collision_y>0.0</collision_y>
<exit0_x>0.8</exit0_x>
<exit0_y>0.6</exit0_y>
<exit1_x>0.8</exit1_x>
<exit1_y>-0.6</exit1_y>
<exit>0</exit>
</xagent>
```

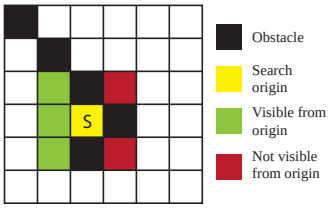

Figure 3: *Visibility rules for the wave propagation. Starting from origin (yellow), the green areas will be included while the red area is not due to it being blocked off by obstacles.*

3.2.1. Collision Force Vector Fields

After the environment has be discretized, each cell is either determined to be a walkable area or an obstacle. A wave-front propagation algorithm ([LaV06] p.378) is then used with the origin of the propagation being the list of cells that has been marked as an obstacle. The algorithm proceeds to collect all adjacent cells that are not already in the list. Having now obtained a list of valid and not already visited adjacent cells, for each of these cells the final collision vector is the normalized sum of the vectors pointing away from its neighbouring cells that was visited last iteration (so for the first iteration, it would be the vectors pointing away from obstacle cells). The normalized vector is then multiplied by the scalar force that depends on the distance (number of iterations). Force used in this simulation decreases exponentially with distance. The adjacent cells list is used as the origin for the next iteration and also added to a list of all visited cells. The process repeats until there are no more cells to visit or by reaching a fixed maximum number of iterations (the force influence distance). An example of a calculated CFVF map can be found in Figure 2 (top left) where force influence distance is set to be 1.

3.2.2. Navigation Force Vector Fields

The NFVF map for each destination is generated in a similar way to the CFVF with the difference that the origins of propagation are pre-defined exit points rather than obstacles and the vector points towards the cells of previous iterations. Propagation also follow a visibility rule (Figure 3) so that it does not go through walls or obstacles. Although this method will guarantee to find the shortest path to the destination from anywhere in the environment, a straightforward implementation of the algorithm results in a common issue which we describe as a "diagonal convergence" of vectors (Figure 4, top right). This is a result of the diagonal lines which propagate from corners representing the shortest path with surrounding flow point attempting to converge towards them. This becomes even more apparent as the grid resolution is increased around larger areas of open space and results in squashed and unnatural pedestrian flows when within a simulation (Figure 5, left).

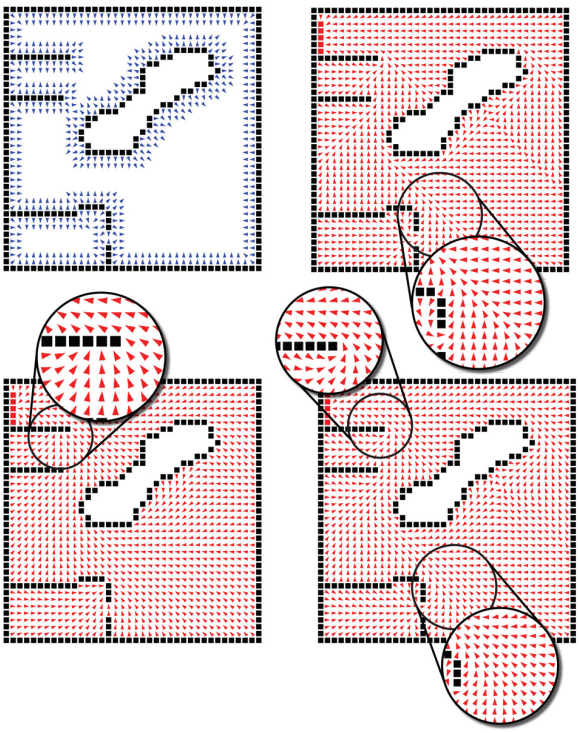

Figure 4: *Comparison of the generated force vector fields. Top left shows the CFVF with force influence distance of 2. The rest shows NFVFs with simple propagation (top right), with backflow propagation but no limit (bottom left) and the result of the full algorithm (bottom right).*

In order to tackle the problem of "diagonal convergence", it is necessary to smooth out the vector flows. We have found that a process of backflow smoothing creates a natural looking navigation flow for pedestrian agents, successfully avoiding convergence issues. Although simple neighbourhood averaging of vectors can be used, the progressive nature of the backflow smoothing results in an even smoother field especially in large open areas. It works by propagating a flow in the reverse direction for each cell of the current iteration and the navigation vector is generated by summing up the vectors pointing from the current cell to the cells in the reverse flow. This reverse propagation also follows the same visibility rules of the main wave propagation and flows around obstacles. Influence of reverse flow vectors is decreased in proportion to the increase in distance of the backflow. The final navigation vector is then normalised after the backflow reaches its maximum iteration distance. As the resolution of the grid increases it then becomes necessary to increase the number of iterations to generate smooth flow especially if there are large open areas in the scene. Using more iterations however, can cause problems in the rest

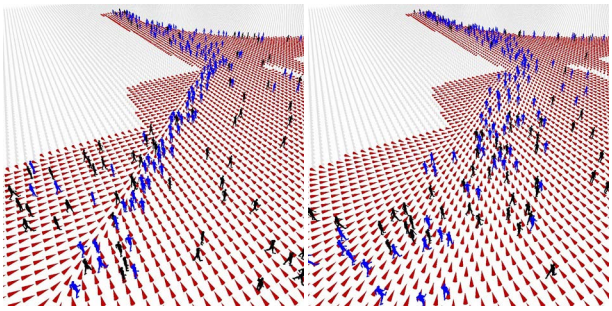

Figure 5: *Comparison of the simulation using un-optimized (left) and with backflow smoothed (right) navigation vector fields.*

of the scene especially around tight corners where vectors run into obstacles (Figure 4, bottom left). A contextual limit is placed on the backflow where the iterations stop when an obstacle is encountered. The final algorithm generates a vector field that is both smooth in open areas but is also correct around corners and narrow corridors (Figure 4, bottom right). The effect of this algorithm can be seen in Figure 5 (right) where pedestrians (in blue) can spread out to take up more space and making the simulation look more natural.

3.3. Pedestrian Agents

Pedestrian agents are continuous spaced mobile agents, each one representing a single embodied pedestrian entity within the simulation. Within a single simulation step they first communicate their locality and velocity (through a message output, shown in Figure 1). Pedestrian agents then perform an iteration of the pedestrian locations messages as well as the navigation cell message information which can be used with any number of agent based steering techniques [Rey99, Hel91]. Within our simulation we combine a mixture of Reynolds individually perceptive steering forces and Helbing's social forces model to determine forced based steering behaviour between pedestrians. As in [HFV00], large contact forces are applied between pedestrians when they are touching each other and helps to more accurately simulate areas of higher congestion. A weighted collision and navigation force is also determined from the pedestrian's position in the FVF and is then applied to the steering velocity. The agent finally updates its position by using a numerical integration to move in the direction of the resolved force vector. This integration step is typically very small (in hundredths of a second) and is adapted depending on performance to provide real time pedestrian movement speeds. The exact agent functions are described in Figure 1.

3.4. Environment Editor

The FVF maps (both collision and navigation) are currently pre-generated through a tool created in order to facilitate rapid generation of different environments. The tool allows the user to sketch or trace over a plan of an area and marking it as an obstacle or exit. The program automatically generates a CFVF and individual NFVFs for each destination according to user-specified parameters. The parameters are force influence distance, minimum force and maximum force for CFVF generation and maximum backflow iterations for NFVFs generation (refer to Section 3.2.1 and 3.2.2). Finally an XML file is generated in the format compatible with the navigation agent definition for the FLAME GPU simulation framework.

4. Simulation & Results

4.1. Simulation Performance Benchmarking

The benchmarking was done on a 64-bit Windows 7 machine using AMD Athlon 4800+, NVIDIA GeForce 9800GX2 with one core used for visualization and one used for CUDA. The simulation's performance was measured against four square grid sizes of 64, 128, 256 and 512 widths. The variation of pedestrian population began from 1024 and increased by multiples of 4 up to 262144 pedestrians. The pedestrians were randomly generated and placed in a walkable space within the environment. Pedestrian density is always kept the same as the environment scales up with increasing pedestrian population. The graph in Figure 6 (top) shows the result of this testing. The result shows that the navigation grid size becomes less important as the number of pedestrian agent increases. This is due to the fact that pedestrian behaviour is the most time consuming part of the simulation. It proves that the use of navigation maps is very scalable and larger grid sizes can be used to represent higher environment detail or a larger area. Currently navigation maps beyond the size of 512x512 cannot be run on larger pedestrian counts (more than 262144 pedestrians) due to the lack of GPU memory as the list of pedestrian agents are double or triple buffered in order to allow reallocation for agent births and deaths. Another factor that contributed to the lack of memory is the fact that as pedestrian population gets larger, in order to keep the same density the number of space partitions (defined by the communication radius of agents [RCR09, LG08]) gets larger and hence requiring a lot more memory.

4.2. Simulation Based on a Real Urban Environment

The test environment was modelled after a busy pedestrianised area in London. It contains 7 destinations/exits with one being an access to the underground station (Figure 7). Pedestrians are generated at each exit location according to a specified emission rate (measured as people generated per minute) and each have a specific exit location to navigate to based on a pre-determined probability. All exits can be in the

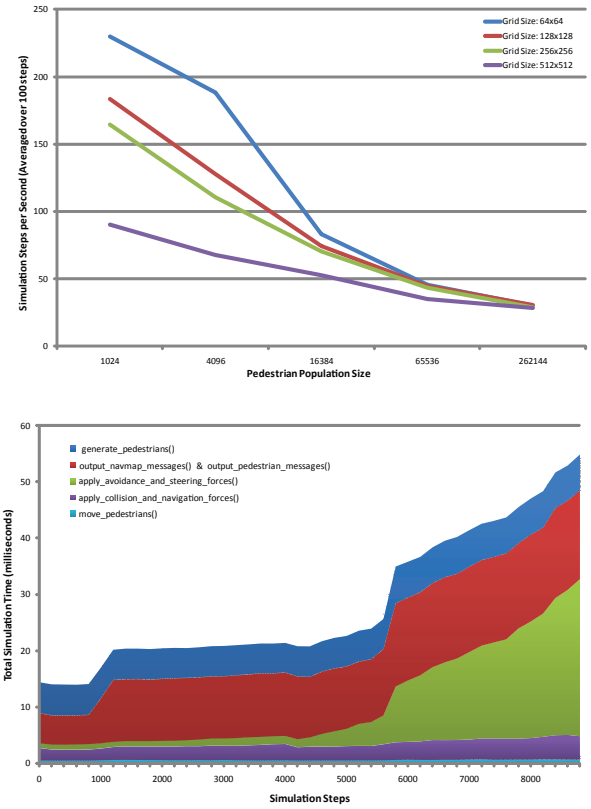

Figure 6: *Simulation benchmarks. Performance benchmarking for environments of different sizes (top). Time utilization by different agent functions with increasing pedestrian agent density (bottom).*

state of either open or closed. When a pedestrian reaches a closed exit, it will randomly choose a different exit and start navigating to the alternate destination. All FVFs are pre-generated according to Section 3.2 with the whole environment being represented as a 128x128 grid although walkable space occupies less than 70% of the total area. The purpose of the simulation is to show the effects of areas that have been flooded with pedestrian possibly as a result of an evacuation or a large public event. Result of the simulation is also expected to highlight dangerous conditions which can result from closing exits. Testing of the model will then follow the following sequence.

Firstly the simulation starts unpopulated and pedestrians are continually generated until they reach a steady state. The model will then be flooded with pedestrians while an exit is closed. Inability of pedestrians to leave the environment in a timely fashion results in heavy congestion and potentially dangerous conditions. Figure 6 (bottom) shows a breakdown of simulation time (in ms) during a real time simulation of

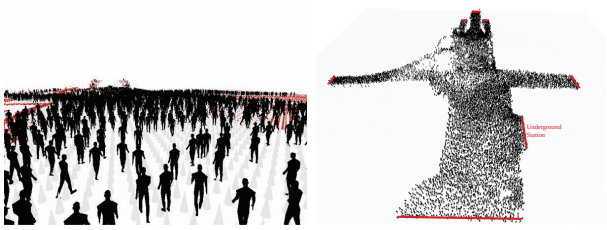

Figure 7: *Screenshots of the final pedestrian simulation. On the street level (top left). Running with 10,000 pedestrians (top right), red lines signify locations of exit.*

the London model. More specifically it shows the timings (stacked) for individual or combination of agent function calls (Figure 1). Timings have been obtained by averaging the simulation time of each function call over the period of 200 simulation steps to avoid any small fluctuations.

Over the simulation of roughly 9000 simulation steps the graph shows three distinct phases. The first occurs between step 0 and 800 and shows relative stability in simulation times. During this phase the emission rate for each exit in the model is constant and the total number of pedestrians remains at roughly 3000.The sharp rise in simulation performance after this phase is a result of increasing the emission rate of all the exits. The second phase of relative stability occurs between steps 1400 and 4400. During this phase of the simulation the large increase in emission rates has reached a stable population of roughly 11,000 pedestrian agents. The final phase of the simulation shows the effect of closing one of the 7 major exits. Any agent which reaches a closed exit is forced unable to leave the simulation at the exit point and must navigate to a new exit position. The effect of this is that first the population increases relatively slowly as a result of pedestrians failing to leave through their preferential exit (steps 4400 to 5600). After this the area around the exit becomes very congested and the number of pedestrians continues to rises very sharply to over 2000 agents. The effect of closing the exit congests the model to the extent that a stable number of pedestrians cannot be reached. It can be clearly seen that under these conditions dangerous crush conditions occur around the exit.

5. Conclusion

The paper described the details of an implementation of a high performance, large scale pedestrian simulation. Pedestrians within the simulation were shown to have the ability to navigate within complex environments, through force vector field maps which are automatically generated using a fast environment prototyping tool. Evaluation of our work shows that our implementation of the force vector field generation algorithm produces configurable smooth flows of pedestri-

ans. The use of discrete agents to host navigation information has been tested to be an efficient and scales effectively with large numbers of pedestrians. The use of real time interaction and control of the simulation model within a real urban environment was used to simulate the effect of flooding the model with pedestrians which resulted in dangerous congestion.

As we have abstracted both the agent and navigation planning behaviour to an agent based problem we are also free to interchange the navigation technique without having to change the rules of our final forced based behaviour. We are already experimenting with an agent based implementation of dynamic navigation graphs which avoid the memory overhead of using fine discrete partitions.

References

[BKSZ01] BURSTEDDE C., KLAUCK K., SCHADSCHNEIDER A., ZITTARTZ J.: Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A 295*, 3-4 (June 2001), 507–525. 2

[Che04] CHENNEY S.: Flow tiles. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), Eurographics Association, pp. 233–242. 2

[CSH06] COAKLEY S., SMALLWOOD R., HOLCOMBE M.: Using x-machines as a formal basis for describing agents in agent-based modelling. In *Proceedings of 2006 Spring Simulation Multiconference* (April 2006), pp. 33–40. 3

[DVD08] DEISSENBERG C., VANDERHOOG S., DAWID H.: Eurace: A massively parallel agent-based model of the european economy. *Applied Mathematics and Computation 204*, 2 (October 2008), 541–552. 3

[Eil74] EILENBERG S.: *Automata, Languages, and Machines*. Academic Press, Inc., Orlando, FL, USA, 1974. 3

[Hel91] HELBING D.: A mathematical model for the behavior of pedestrians. *Behavioral Science 36*, 4 (October 1991), 298–310. 2, 6

[HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature 407*, 6803 (September 2000), 487–490. 2, 6

[Hug02] HUGHES R.: A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological 36*, 6 (July 2002), 507–535. 2

[JXW*08] JIN X., XU J., WANG C. C. L., HUANG S., ZHANG J.: Interactive control of large-crowd navigation in virtual environments using vector fields. *IEEE Computer Graphics and Applications 28*, 6 (November 2008), 37–46. 2

[LaV06] LAVALLE S. M.: *Planning Algorithms*. Cambridge University Press, May 2006. 2, 5

[LG08] LE GRAND S.: Broad-phase collision detection with cuda. *GPU Gems 3* (2008), 697–721. 6

[MPG*10] MOUSSAÏD M., PEROZO N., GARNIER S., HELBING D., THERAULAZ G.: The walking behaviour of pedestrian social groups and its impact on crowd dynamics. *PLoS ONE 5*, 4 (April 2010), e10047+. 2

[Nvi09] NVIDIA: Nvidia cuda programming guide (v 2.3), 2009. 3

[PLT05] PETTRE J., LAUMOND J., THALMANN D.: A navigation graph for real-time crowd animation on multilayered and uneven terrain. 3

[RCR09] RICHMOND P., COAKLEY S., ROMANO D.: A high performance agent based modelling framework on graphics card hardware with cuda. *Proceedings of The Eighth International Conference on Autonomous Agents and Multiagent Systems* (2009). 3, 6

[Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, July 1987), vol. 21, ACM, pp. 25–34. 2

[Rey99] REYNOLDS C.: Steering behaviors for autonomous characters. In *Game Developers Conference 1999* (1999). 6

[RR08] RICHMOND P., ROMANO D.: A high performance framework for agent based pedestrian dynamics on gpu hardware. *European Simulation and Modelling* (2008). 3

[RWCR10] RICHMOND P., WALKER D., COAKLEY S., ROMANO D.: High performance cellular level agent-based simulation with flame for the gpu. *Briefings in Bioinformatics* (2010). 3

[SBOT08] SHOPF J., BARCZAK J., OAT C., TATARCHUK N.: March of the froblins: simulation and rendering massive crowds of intelligent and detailed creatures on gpu. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes* (New York, NY, USA, 2008), ACM, pp. 52–101. 2

[Sha05] SHAO W.: Environmental modeling for autonomous virtual pedestrians, 2005. 3

[Sim10] SIMWAL K.: http://fseg.gre.ac.uk/exodus/ [last accessed jan 2010]. 1

[Sof10] SOFTWARE M.: http://www.massivesoftware.com/ [last accessed jan 2010]. 1

[ST05] SHAO W., TERZOPOULOS D.: Autonomous pedestrians. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM Press, pp. 19–28. 2

[TCP06] TREUILLE A., COOPER S., POPOVIĆ Z.: Continuum crowds. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM, pp. 1160–1168. 2

[TLCC] TECCHIA F., LOSCOS C., CONRY R., CHRYSANTHOU Y.: Agent behaviour simulator (abs): A platform for urban behaviour development. 2

[TP02] TURNER A., PENN A.: Encoding natural movement as an agent-based system: an investigation into human pedestrian behaviour in the built environment. *Environment and Planning B: Planning and Design*, 29 (2002), 473–490. 1