GPU Pedestrian Simulation

T. Karmakharm[1] and P. Richmond[1]

[1]Department of Automatic Control & Systems Engineering, University of Sheffield , United Kingdom

Abstract

A few lines of abstract text. A few lines of abstract text. A few lines of abstract text. A few lines of abstract text. A few lines of abstract text. A few lines of abstract text. A few lines of abstract text. A few lines of abstract text. A few lines of abstract text. A few lines of abstract text. A few lines of abstract text. A few lines of abstract text. A few lines of abstract text. A few lines of abstract text.

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

1. Introduction

Safety of occupants during an evacuation of large public spaces is paramount to their design. Simulation of large scale pedestrian behaviour is an essential tool in identifying potential dangers that may arise during an emergency.

Multi-agent simulation, where an individual agent is an independent operational unit with memory (e.g. a single pedestrian), has been widely adopted in representing pedestrian crowds for both research and commercial applications [Leg10, exo, Mov, PU]. Using this approach, a complex simulation that shows emergent behaviour can be created by defining few simple rules. Independent nature of agents allow these type of simulations to be parallelised by running on suitable hardware such as a computing cluster or the GPU (Graphics Processing Unit) [**?**].

In order to obtain the best design and achieve statistical significance in stochastic models, multiple simulations must be run with differing building layouts, parameters (e.g. number of pedestrians) and random seeds. The resulting metrics from these simulations must be collected and visualised to assist users in judging the effectiveness of a design.

Our prototype decision support tool for pedestrian simulation is able to achieve concurrent multi-simulation by utilising multiple GPU-enabled computers connected over a standard TCP/IP network. Our test model shows it is possible to simulate at faster than real-time speeds. It is built up on a generalised agent-based framework and behaviours in the model are generally applicable to many other scenarios.

After a review of related work, Section 3 explains the architecture and implementation of the tool. The results are then presented in Section 4 and concluding in Section 5.

2. Related Work

Reynold's work on flocking [Rey87] and steering [Rey99] behaviour shows how agent based models can be applied to the movement of animal with social characteristics. Helbing et al. applied a similar concept to pedestrians by introducing the social forces model [Hel91] and later to an evacuation scenario with introduction of contact forces [HFV00].

Clusters of GPUs have been used in many fields such as fluid dynamics for solving incompressible flow [JTS10] and shallow water equations [BSA12], in siesmic modelling [ACC*09] and have also applied to agent based simulation [APS10]. This is however, still a relatively new area to explore in crowd simulation and evacuation modelling. Furthermore, their use for an evacuation decision support tool have yet to be exploited.

The FLAME GPU framework [Ric11, RR11] is used as a basis for the simulator in this paper. It is an implementation of the Flexiable Large-scale Agent Modelling Environment (FLAME) framework [HCS06] targeted at GPU computing. Within FLAME an agent based simulation is described as a set of state based agents with an internal memory. The use of states ensures that minimum amounts of behavioural divergence in groups of agents is introduced, an issue which prevents many simulators from performing well on the GPU's vector type processing architecture.

3. Multi-simulation decision support tool

The tool is split in to four modules responsible for user interaction (Terminal), server for queuing and dispatching jobs (Simulator Manager), performing simulation runs (Simulator) and managing Simulators on local computers (Local Simulator Manager). Efficiency is increased by merging of multiple smaller simulations for running concurrently on individual GPUs (Section **??**). Metrics are collected as the simulation runs in order to be used for evaluating an effectiveness of a plan (Section **??**).

The four modules form a client-server relationship (Figure 1) and inter-application communication is performed over TCP/IP. The Simulator Manager handles all communications between the different modules. Simulation instances, containing all agent and parameter data required to start a single simulation, received from the Terminal are queued or dispatched to Simulators with available capacity. The metrics requested by Terminals also pass through the Simulator Manager. The Terminal is local to the user's machine and allows for direct user interaction. It is responsible for the creation and loading of the simulation instances, dispatching a batch of them to be simulated and receiving sets of metrics from running and completed simulations.

The Simulator contains the simulation framework and is responsible for the actual simulation of the model. An instance of a Simulator is created per GPU that exists on the machine. This makes it modular and scalable across multiple machines. The Simulator uses a modified version the CUDA based FLAME GPU framwork [Ric11, RR11] which enables simulation batching and collection of metrics. In order to save bandwidth and avoid overloading the network, all metrics data generated during a simulation run are saved locally on the machine and sent across the network when requested. As there may be multiple multiple GPUs for each local computer, the Local Simulator Manager is responsible for detecting a machine's configurations and launch instances of the Simulator accordingly.

Previous work have shown that larger models involving tens of thousands of pedestrians are required in order to make effective use of the parallel processing capabilities of the GPU. For smaller models however, a way to increase efficiency is in merging multiple simulations together and run them simultaneously, effectively increasing the size of the simulation. The implementation spatially merges navigation vector fields of multiple simulations together effectively creating a large simulation. Pedestrians are assigned simulation identifiers in order to prevent interference in boundary cases.

In order to provide feedback, a number of metrics must be collected in order to summarise the performance of each individual simulation configuration. One such metric is the Fruin Level of Service (LOS) [**?**] which measures density in terms of pedestrian comfort level. LOS output is similar to a density map and allows intuitive identification of dangerously congested areas. Because of simulation batching,

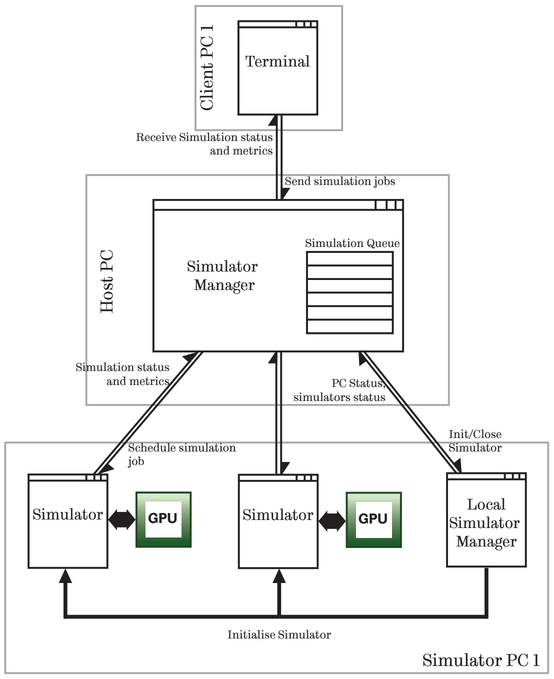

Figure 1: *The network diagram shows how the modules of the decision support tool interact. The tool is a client-server system with the Simulator Manager acting as the server. The Simulators are initialised on the local machine by the Local Simulator Manager. It accepts and run simulation jobs from the Simulator manager and sends feedback when requested. The Terminal is used to submit simulation jobs and receive feedback.*

metric results must be re-mapped in respect to each individual simulation. Parallel prefix sums and sorting algorithms are used extensively in collecting metrics.

4. Results & Discussion

Two benchmarks were conducted to measure the performance of the tool. A multi-GPU performance benchmark was carried out to show the effect of running multiple Simulator instances on the same machine. The second benchmark measures the simulation speed relative to real time when the simulation is running a normal scenario. All benchmarking were performed on a 3.4GHz Intel Core i7 machine with 8GB of RAM using the dual-core Nvidia GTX590 GPU. Visualisation on the Simulator was turned off as in reality the metrics data would be visualised on the Terminal.

Performance when running the Simulators on both available GPUs was measured. This is to evaluate the overheads in running multiple instance of the Simulator on the same machine. An environment was created and gradually filled with pedestrians. Each Simulator runs a batch of four simulations simultaneously. The simulations are then left to stabilise and an average real time step was taken over 100 simulation steps. The benchmark is run first for one instance of the Simulator (using one GPU). It is then performed again using two instances of the Simulator (two GPUs), effectively doubling the number of pedestrians simulated. The results show that there the time for each simulation step increases in the dual GPU configuration on average of 2% across the range. For 5,000, 50,000 and 100,000 pedestrian per GPU, it took an average time of 7.42, 16.99 and 39.14 ms for one time step respectively.

The model used in the second benchmark represents an evacuation of a centrally located shopping mall after a dirty bomb has exploded. The simulation starts when the first pedestrian evacuates the mall and tries to take the nearest exit from the environment (Figure 2a). After a certain time, a cordon is established to prepare pedestrians for decontamination. When this happens, pedestrians starts to gather in a pre-designated location in the environment (Figure 2b). This scenario was created after consultation with Nottingham and Avon fire services and is representative of what the tool is intended for. Local pedestrian dynamics are based on Helbing's social forces model [**?**] and global navigation and static obstacle avoidance are represented multiple force vector field maps [**?**]. Variable time steps are used to prevent numerical instabilities from exponential repulsion forces.

Use of dynamic time steps means simulation performance can vary greatly according to the densities and forces experienced by pedestrians. The second benchmark measures the performance of the simulation taking this factor in to account. A maximum time step of 0.2 seconds was used, a total of 1200 pedestrian were generated and time for the cordon to be established was set to be 10 minutes in simulation time. The simulation ends at 45 minutes when the decontamination structure is established. The velocity threshold, a limit on how much pedestrians can change velocity in a single time step, was changed between each run. They take the values of 0.02, 0.05 and 0.1 as in the Figures 3a, 3b and 3c respectively. For the graphs in Figure 3, iteration count forms the x axis. The left y axis represents the real simulation speed calculated as the dynamic simulation time step (dt) in proportion of real time needed to process that iteration (rdt), values higher than 1 (represented by the red line) means that the simulation is running faster than real-time. The dt and rdt values are an average taken every 100 steps. Finally the right y axis (blue) shows the number of pedestrians currently in the environment.

Until the cordon is established, it can be seen from the graphs that real simulation speed fluctuates wildly depending on the forces that pedestrian experiences. Average speed then slightly declines as the behaviours change and pedestrians stop trying to exit the environment and start to congregate in the warm zone near the disrobing point. After all pedestrians have been generated and the simulation stabilises the speed can then be seen to increase rapidly due to the fact that pedestrians are able to spread out in to available spaces. The velocity threshold has a drastic effect on the real simulation speed. For a 45 minute simulation, the one with threshold value of 0.02 took approximately 29 minutes to complete. Simulations with threshold values of 0.05 and 0.1 completed in 13.8 and 7.2 minutes respectively. This means a conservative threshold value of 0.05 would result in the the ability to perform simulations at 3 times the speed of real-time.

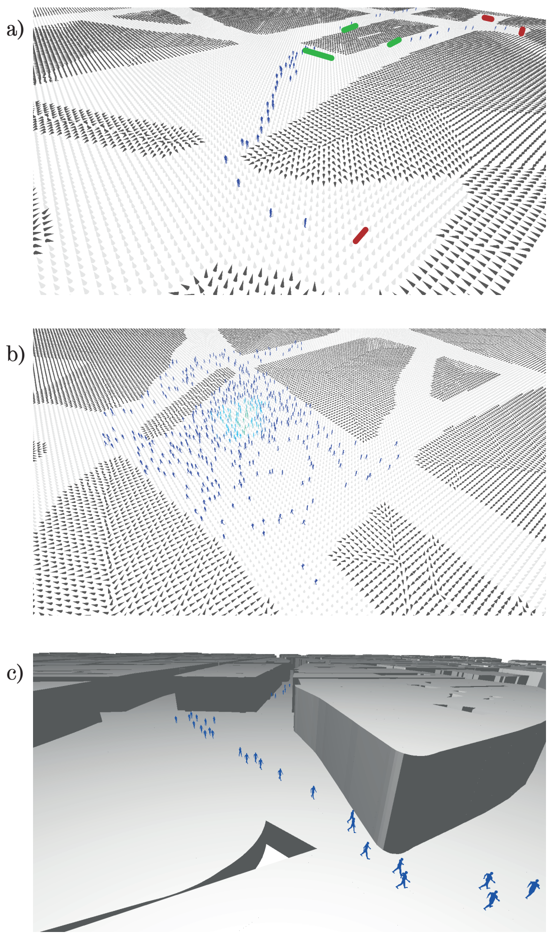

Figure 2: *Screen captures of the running simulation. Image (a) shows pedestrians being generated and taking the shortest path out of the environment. The green lines marks the emitter and red lines marks the exits. After a the cordons have been established, pedestrians congregate near the disrobing point near the centre of image (b) within the warm zone. Image (c) shows the simulation when visualised with the 3D environment.*

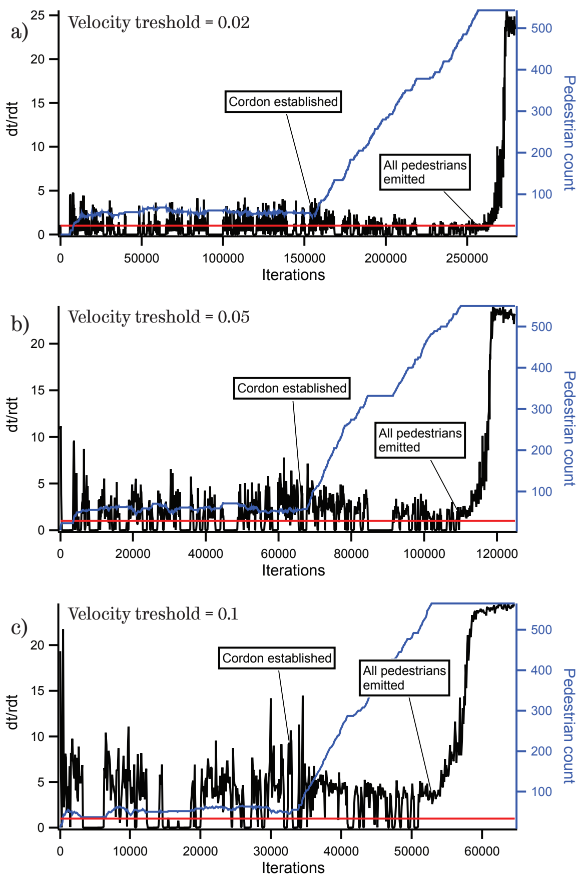

Figure 3: *The three graphs shows how the real simulation speed (dt/rdt) changes throughout a simulation in normal conditions. The value dt stands for the dynamic time step and the rdt stands for the amount of real time taken to process that step of the simulation. Both these values are averaged over 100 steps. Real simulation speed values over 1 (above the red line) means it is running faster than real time. The line in blue along with the right axis shows the number of pedestrians currently in the simulation.*

5. Conclusion

This paper described the implementation of a prototype decision support tool to help in pedestrian evacuation planning. The tool utilises multiple simulators connected through standard TCP/IP network in order to facilitate multiple concurrent simulations. Smaller simulations are also batched in order to maximise efficiency. Feedback is provided through collected metrics.

From preliminary results, the performance of simulation batching and running multiple simulator for each machine is promising. Testing on more machine equipped with larger numbers of GPUs will provide better indication of performance. A method to determine and rank effectiveness of a plan will be obtained by further investigation with our domain experts.

References

[ACC*09] Abdelkhalek R., Calandra H., Coulaud O., Roman J., Latu G.: Fast seismic modeling and Reverse Time Migration on a GPU cluster. In *High Performance Computing & Simulation, 2009. HPCS '09. International Conference on* (June 2009), IEEE, pp. 36–43. 1

[APS10] Aaby B. G., Perumalla K. S., Seal S. K.: Efficient simulation of agent-based models on multi-GPU and multi-core clusters. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques* (ICST, Brussels, Belgium, Belgium, 2010), SIMUTools '10, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 1–10. 1

[BSA12] Brodtkorb A. R., Sætra M. L., Altinakar M.: Efficient shallow water simulations on GPUs: Implementation, visualization, verification, and validation. *Computers & Fluids 55* (Feb. 2012), 1–12. 1

[exo] Exodus Website [Last accessed Jan 2012]. 1

[HCS06] Holcombe M., Coakley S., Smallwood R.: A General Framework for agent-based modelling of complex systems. In *Proceedings of the 2006 European Conference on Complex Systems* (2006). 1

[Hel91] Helbing D.: A mathematical model for the behavior of pedestrians. *Behavioral Science 36*, 4 (Oct. 1991), 298–310. 1

[HFV00] Helbing D., Farkas I., Vicsek T.: Simulating dynamical features of escape panic. *Nature 407*, 6803 (Sept. 2000), 487–490. 1

[JTS10] Jacobsen D., Thibault J. C., Senocak I.: An MPI-CUDA Implementation for Massively Parallel Incompressible Flow Computations on Multi-GPU Clusters. In *American Institute of Aeronautics and Astronautics (AIAA) 48th Aerospace Science Meeting Proceedings* (2010). 1

[Leg10] Legion: Legion website [Last accessed Jan 2012]. 1

[Mov] Move S.: http://www.smart-solutions-network.com [Last accessed Jan 2012]. 1

[PU] PedestrianSimulation UB F.: http://www.pedestrian-simulation.com/ [Last accessed Jan 2012]. 1

[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. 1

[Rey99] Reynolds C.: Steering Behaviors for Autonomous Characters. In *Game Developers Conference 1999* (1999). 1

[Ric11] Richmond P.: *FLAME GPU Technical Report and User Guide (CS-11-03)*. Tech. rep., Department of Computer Science, University of Sheffield, 2011. 1, 2

[RR11] Richmond P., Romano D.: Template-Driven Agent-Based Modeling and Simulation with CUDA. In *GPU Computing Gems Emerald Edition*, Hwu W.-m. W., (Ed.), 1 ed., Applications of GPU Computing Series. Morgan Kaufmann, Feb. 2011, ch. 21, pp. 313–324. 1, 2