Procedia Computer Science

Volume 80, 2016, Pages 2428–2432

ICCS 2016. The International Conference on Computational Science

CROWD TURBULENCE WITH ABM AND VERLET INTEGRATION ON GPU CARDS

Albert Gutierrez-Milla, Francisco Borges, Remo Suppi and Emilio Luque

Universitat Autònoma de Barcelona, Bellaterra, Barcelona, Spain
albert.gutierrez@caos.uab.cat, francisco.borges@caos.uab.cat, remo.suppi@uab.cat, emilio.luque@uab.cat

Abstract

Managing crowds is a key problem in a world with a growing population. Being able to predict and manage possible disasters directly affects the safety of crowd events. Current simulations focus mostly on navigation, but crowds have their own special characteristics and phenomena. Under specific conditions, a mass can turn into a crowd turbulence which may lead to a possible disaster. Understanding the internal phenomena is an important issue in order to model behavior. In the particular case of crowd turbulence, agents are moved by the crowd by a series of pushes, an involuntary movement that can be hard to reproduce. We propose a simple model to implement this complex problem based on intentional and involuntary interactions among the agents. The implementation is a hybrid model between the Verlet integration method and Agent Based Modeling. We implemented the proposed model using C and OpenCL and we evaluated its performance on a Nvidia GPU.

Keywords: Agent Based Modeling, Verlet Integration, Crowd Turbulence, High Density Crowds.

1 Introduction

Nowadays, there are 35 mega-cities with more than 10 million people. This number will increase in the following years due to population growth and emigration from villages to big cities. Cities are rapidly increasing in population and big events are getting more common. Managing the implicit risks of these situations is a key factor for the safety of the attendees. How to handle them and predict behavior will require models, to understand and reproduce the behavior of the crowds. Our analysis is focused on the need for a crowd model able to efficiently reproduce crowd internal phenomena.

Pedestrian dynamics are complex systems. Under low density, they have different behavior than when pressure increases. As the density increases, new phenomena appear and the crowd tends to have a behavior similar to a fluid. Forces are propagated among the agents and the mass is moved in what it is called crowd turbulence. These phenomena are not well understood at this point, and the lack of data in these conditions makes studying them even more difficult.

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2016

doi:10.1016/j.procs.2016.05.542

In this paper, we propose a model as a contribution to the understanding of this problem. Most evacuation models are based on the idea of free navigation and solving interaction problems. However, some of the models lack the representation of special events reproduced in high density and special phenomena related to the concentration of a large population. In the case of crowds, the interest lays in what is happening inside, the problematic areas and the risks involved. Crowd simulators are able to manage thousands of agents but without reproducing the interactions and events that occur in the case of high pressures and disasters.

Testimonies of crowd disasters relate how they were moved following a set of physical forces against their own will. This can be considered as particle behavior, where agents of the system are moved by their neighboring forces. Verlet integration is a numerical method based on the Newton formulation of forces that suits the problem. We propose a model based on ABM to model the pedestrian navigation and Verlet when the agent is under high density conditions. In the case of evacuations, large populations have significant computational requirements and computing times. High performance computing allows us to run simulations more efficiently, taking advantage of the capacities of modern hardware, decreasing the total computing time. Using an approach similar to real situations, agents take decisions independently considering their environment without any centralized decision. We implemented an SIMD model that was executed on a Nvidia GPU using OpenCL 1.2.

2 Related work

The evacuation problem has been tackled from a wide range of approaches. Among others, we can find: fluid dynamics, lattice gas models or even simulation with animals[1]. Nowadays, the main approaches can be divided into: continuous time and discrete time simulations. In discrete time, we find cellular automata, a computationally efficient model. It is easier to implement but can consume a quadratic space and have less realism and complexity. Even with a more limited model, there have been interesting approaches extending them. In continuous time, one of the main models is Social Force[2]. It explains self-organized phenomena such as group formation or bottlenecks. Moreover, it is a widely extended and validated model used by a considerable part of the evacuation research community. In the case of crowd turbulence, Social Force was extended by Yu and Johansson[3] to model turbulent flows. They added extra terms to the repulsive force, improving the previous Social Force model. Furthermore, Helbing's study of the Love Parade disaster in the 2012[4] describes the crowd quake phenomena, a special case of crowd turbulence. This turbulence can generate high densities leading to a critical situation where the pressures can produce chest injuries

Verlet integration has been used to integrate Newton's equations of motion. It has been used mostly in game physics engines to reproduce the collisions between world objects. It provides good stability with a relatively simple model. In the area of evacuations, free navigation and HPC, there are several approaches with GPUs. The approaches have been more related to realistic behavior of boid agents moving freely within the space that tries to provide a model for crowd evacuations. For instance, Bleiweiss[5] work on GPUs has been focused on the rendering problem and on reproducing a realistic behavior. Richmond[6] proposed the Flame GPU template, where he creates a GPU tool able to produce templates for SIMD problems and visualization. The literature on computational crowd dynamics has been improved by the game industry, aiming to find realistic behaviour of void agents moving through the space[7][8].

3 Model

Analyzing crowd waves and crowd turbulence videos, we realized that the behavior is similar to the particles and numerical methods reproducing waves. With high densities, crowd turbulence appears, and a movement similar to a fluid dynamic is reproduced. We propose a model with the following bases: independent navigation of agents, interaction with neighbors, high densities and agents that tend to be displaced by the mass in an involuntary movement. We consider that, when urgency increases, interactions become more common, there are sudden changes in velocity and people pushing, which turns into a propagation of forces that creates crowd turbulence. Our model is time continuous, every agent trying to move to a v_{max} with a specific acceleration, a. Agents calculate their movement without considering collision with other agents, just applying the desired velocity moving towards the exit. Then, there will be a coherence step applied to detect intersections. Because, in the case of crowd turbulence, there is an inertia of mass movement and the interaction needs to be propagated, we also implement an "inertia" phase. In this step, every agent will update its position according to the previous position. When agents push others in high densities, the push is propagated as a force problem. We will consider two ways of pushing: intentional and involuntary. Involuntary comes when the agent is moved by the crowd turbulence. The involuntary pushes are constantly applied. Intentional pushing is generated when an agent under high pressure conditions tries to get more space, and it pushes a surrounding agent to do so. The intentional pushes are applied in a peak of force and because they are more energy demanding and are not constant.

4 Simulation

We think that the usage of numerical methods can be a powerful tool in predicting the behavior agents have on crowd turbulence. The fact that agents cannot be modeled as particles forces us to mix the particle behavior with the agent behavior. To model agents as particles, we used Verlet, and, for the agent behavior, we used ABM. As previously described, we will consider agents as particles. The simulation was implemented using the model proposed in Section 3. We used numerical simulations in order to implement the formulas. Each agent decides in simulation time t what its new state at time $t+1$ will be. In order to do so, it will independently compute its new position. Every agent is considering its neighbors to calculate its new position. The system keeps the coherence, making agents remain inside wall boundaries. Lastly, the stress is applied, affecting some neighbors of the stressed agent. Agents are modeled as circles with an average radius of 0.4m, have a desired speed distributed with the mean value $v = 1.34m/s$ and the standard deviation $0.26m/s$[9]. Because up to 10 agents per square meter[10] have been observed, we allow overlapping of the agents. There is no data dependency between the agents, and this allows us to implement the model on SIMD architectures. Using the software parallelism to optimize the simulation is feasible due to the implicit parallelism found in decentralized models where the decisions are taken independently, accessing different data.

5 Results

To evaluate the proposed model, we have implemented a simulator. As a first step, we created a scenario able to hold a desired population increasing the number of agents, but maintaining the desired density. Next, we have run sequential simulations to evaluate the performance of the model and be able to evaluate the model results. Lastly, we have run simulations on a

many-core architecture with populations of 512, 1024, 2048 and 4096 agents. The simulation results showed in the model section are executed by the GPU. The area, shown in Fig. 1 (a), is a closed rectangular space with two doors where the agents are evacuated. Once the simulator was implemented, we analyzed the behavior of the agents. Fig. 1 (b) depicts the velocity over v_x and v_y of a pedestrian. The velocity evolves, going from completely stopped to movements against desired will, where the agent is moved by the mass. This shows a random pattern determined by the crowd. All simulations were executed on a hybrid system using the C programming language. The system was an Intel CPU with 2.1 GHz and 8GB RAM memory. The GPU, a Nvidia GTX 750 GPU, with a Maxwell architecture, 1,1GHz, 640 CUDA cores and 2 GB GDDR5 memory. We used OpenCL 1.2 version implemented by Nvidia, GCC compiler 4.4.7.

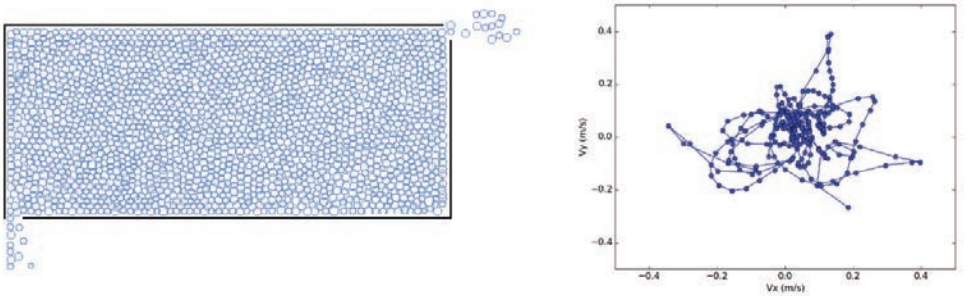

Figure 1: (a) Synthetic scenario with a crowd of 2048 agents. (b) Velocity vx and vy of an agent.

The colors displayed in Fig. 2 (a) describe the pressure of an area of the scenario. The intensity of the red represents the pressure of the region. From left to right and top to bottom, the frames show the time evolution of a propagation of the pressure, where the forces and velocities of the agents are modified by their neighbors. When the density is growing, strong reactions and the repulsive forces of the agents affect the rest with an oscillatory behavior.

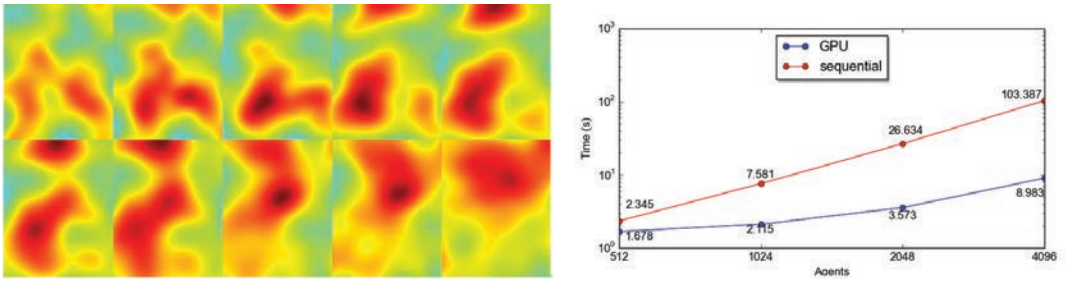

Figure 2: (a) Frames representing the evolution of pressure of a crowd turbulence. (b) Execution time of 1000 steps for the GPU and sequential versions.

The space geometry is saved in memory, and agents are created in the host and transferred to the GPU at the beginning. During the simulation, coordinates are transferred from the device to the host in order to observe or save the evolution of the crowd. In Fig. 2 (b), we

display the results of our experiment. We compare 1.000 simulation steps for different numbers of agents in the GPU version. In the GPU case, the running time includes memory transaction from the GPU to the host with the coordinates of all the agents in order to save the step. The data movements using the PCI connection penalize the GPU version, where it locally stores the data. As we increase the number of agents in logarithmically, there is no significant performance penalty. This means that it is scaling efficiently. The data parallelization and the total independence in the written operation are shown in these results. Each agent is working and manipulating only their own positions and reading everybody else's position.

6 Conclusions

In this paper, we presented a novel crowd model able to handle interactions between thousands of agents and reproduce crowd turbulence. Our method proposed the usage of the numerical method Verlet integration combined with ABM. The model was successfully implemented in a simulator for a many-core architecture. The GPU used OpenCL and it showed an outperformance over the initial sequential version. From the performance analysis, it is shown that our model is scalable, fast and able to handle thousands of agents.

Acknowledgment

This research has been supported by the MINECO (MICINN) Spain under contract TIN2014-53172-P.

References

[1] X. Zheng, T. Zhong, and M. Liu. "Modeling crowd evacuation of a building based on seven methodological approaches.", *Building and Environment.*, 2009, vol. 44.3 pp. 437-445.

[2] D. Helbing, I. Farkas, T. Vicsek, "Simulating dynamical features of escape panic" in *Nature.*, 2000, vol. 407, pp. 487-490.

[3] W. Yu, and A. Johansson. "Modeling crowd turbulence by many-particle simulations.", *Physical Review E.*, 2007, vol. 76.4: 046105.

[4] D. Helbing, and P. Mukerji. "Crowd disasters as systemic failures: analysis of the Love Parade disaster", *EPJ Data Science.*, 2012, vol. 1(1), pp. 1-40.

[5] A. Bleiweiss. "Multi Agent Navigation on the GPU", *White paper, GDC.*, 2008, vol. (9).

[6] P. Richmond, S. Coakley, and D. M. Romano. "A high performance agent based modelling framework on graphics card hardware with CUDA.", *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems.*, 2009, vol. 2.

[7] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P.Dubey. "Clearpath: highly parallel collision avoidance for multi-agent simulation." *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. ACM*, 2009, pp. 177-187.

[8] J. Snape, S. J. Guy, M. C. Lin, and D. Manocha. "Local and global planning for collision-free navigation in video games." *Planning in Games Workshop. Citeseer.* 2013.

[9] L. F. Henderson. "The statistics of crowd fluids.", *Nature.*, 1971, vol. 229, pp. 381-383.

[10] D. Helbing, A. Johansson, and H. Zein Al-Abideen. "Dynamics of crowd disasters: An empirical study." *Physical review E 75.4.*, 2007, 046109.