Parallelization of BioCrowds Model for Crowd Simulation on GPU and Inclusion of Pushing Effect

Natasha Sayuri Dias Nakashima, Léo Pini Magalhães (Advisor)
Department of Computer Engineering and Industrial Automation (DCA)
Faculty of Electrical and Computer Engineering (FEEC)
University of Campinas (Unicamp)
Av. Albert Einstein, 400 - Campinas, SP, Brazil
{sayuri,leopini}@dca.fee.unicamp.br

Abstract – BioCrowds is a method for crowd simulation proposed by Alessandro Bicho [1] based on the biologically-motivated space colonization algorithm. This algorithm was originally introduced to model leaf venation patterns and the branching architecture of trees. However, the increase on the number of individuals corresponds to a decrease on the framerate of the simulation. This work implements a solution to simulate virtual crowds using the BioCrowds and parallel computing. Additionally it approaches an extension of BioCrowds to treat pushing effect among agents. The proposed parallel algorithm was implemented using the OpenCL framework. The simulations with this algorithm resulted in a increase on the framerate, maintaining the reproduction of emergent behaviors of real crowds.

Keywords – GPU, crowd simulation, pushing effect

1. Introduction

The BioCrowds model, proposed by Bicho [2], adapts a space colonization algorithm, originally proposed for simulation of leaf venation patterns [8], to crowd modeling. It focuses on the competition for space among moving agents. In his thesis, Bicho compares qualitatively and quantitatively his model to other crowd simulation models. For all models, the increase on the number of individualus and markers in scenario corresponds to a decrease on the frame rate of the simulation. Thus, the use of parallel computing to implement algorithms for simulating crowd is an alternative to achieve better performance with BioCrowds algorithm for crowd simulation.

The objective of this paper is to present an implementation of the BioCrowds algorithm using parallel computing techniques based on GPU. Additionally it will display an extension of BioCrowds to treat simulation scenarios with physical contact among simulated agents, effect named pushing.

The remainder of the paper is organized as follows. Crowd dynamics and some works found in literature of crowd simulation based on GPU are reviewed in Section 2. The proposed algorithm for crowd simulation implemented on GPU is described in Section 3. Results of simulated experiments are presented and evaluated in Section 4. Finally, in Section 5, we present conclusions and suggest directions for future work.

2. Crowd dynamics and simulation

2.1. Crowd Dynamics

Long ago, the collective behavior is studied. In 1905, the psychologist social Gustave Le Bon published the book "The psychology of crowds" (La psychologie des foules), which made an important contribution to the study of crowds behaviours. According to Le Bon, crowd, in their ordinary sense, represents a concentration of individuals, regardless of nationality, sex or common interests. In its psychological sense, crowd is a concentration of individuals with distinct behavioral characteristics from those that would present, if they were isolated. These behavioral patterns are the result of interactions and influences with the environment and with other individuals of the crowd.

Several authors have observed and identified individual behavioral characteristics in real crowds [3][10], which are related to factors inherent to the human being as a result of their desires, beliefs and emotions, and also the patterns that emerge from the crowd when it is observed as a whole. The main individual behavior are as follows:

- goal seeking: individuals move in the environment to reaching their destinations;
- collision avoidance: individuals maintain a safe distance from other individuals and obstacles, called interpersonal distance;
- least effort strategy: individuals choose paths that minimize the distance to reach their goal.
The emergent behaviors are observed when analyzing the crowd as a whole, resulting of their self-organization. The principal emergent behaviours are as follows:

- lanes formation: in a high population density environment with groups moving in opposite directions, bidirectional flows emerge. This self-organization occurs because each individual seeks to reduce its effort, maintaining speed equals or lower than the speed of the individual immediately in front;
- prior organization: when a collision is detected, individuals anticipate their position to avoid it;
- pushing effect: this effect occurs when individuals have physical contact, e.g. environments with narrowing passageways causing a decrease on the distance among them.

2.2. Crowd simulation based on GPU

The works related to crowd simulation have achieved more realistic results in real time using a new programming paradigm based on GPU.

The graphic processing unit, known as GPU, was originally designed to image rendering on personal computers. Nowadays, because of its large capability to process data simultaneously, the GPU has been used to general purpose computing in many research areas.

Example of crowd simulation using GPU is the work introduced by researchers from AMD company [9], with the goal of increasing the number of interacting entities to be simulated. They implemented a framework that uses the GPU for planning routes for large scale crowds using a continuous approach, as proposed by Treuille in 2006. The proposed method simulates 65,000 agents in real time with a rate of 20 frames per second (fps) using a single ATI RadeonTM HD 4870.

Ondřej et al. [5] proposed an optic flow-based approach for steering pedestrians inspired by cognitive science work on human locomotion. It was implemented using OpenGL Shaders and CUDA programming, which is NVIDIA platform for general parallel processing. They used the Quadro FX 3600M GPU and obtained a rate of 25 fps to simulate 200 agents.

2.3. BioCrowds on GPU

On the original approach of BioCrowds algorithm, proposed by Bicho [2], the growth of leaf venations can be interpreted as the movement of an agent in a virtual environment and “walkable spaces” are defined through a discrete set of points. These markers are randomly arranged on the virtual scenario using the dart-throwing algorithm developed by Cook. Each agent owns the closest markers in each iteration of the simulation and as a result, its next position is calculated.

Thus, the original BioCrowds algorithm can be divided into four stages: model initialization, calculation of agent closest to each marker, calculation of instantaneous movement vector of each agent and update of its position.

BioCrowds algorithm needs to be adapted to run on processors as the GPUs, which are specially designed to process a large amount of independent data simultaneously on a SIMD strategy. For this implementation we identified two pieces of the sequential algorithm where data are independent from each other: definition of markers related to each agent on the simulation and the definition of the next position of the agent. Therefore, two kernels (program responsible for parallel processing on the GPU) were created:

1. one in which the agent closest to each marker is calculated;
2. one that creates a list of markers closer to each agent and calculates the next step of each agent.

To implement the pushing effect, we studied the method proposed by Pinheiro [7], that change the local density of markers on the virtual scenario where agents remained stopped during a certain simulation time. This method minimizes the distance among agents and they get close to each other, simulating pushing. However, regions in which this phenomenon occurs must be defined before the start of the simulation.

In our proposed algorithm, we change the density of markers during the simulation by continuously checking if the simulated agents are stopped during a certain time. If it happens, we define the smallest polygon which contains the area occupied by the agents (Convex Hull [6]) and in this area ad-
ditional markers are added. This way, agents can get closer and the pushing behaviour can be simulated.

The BioCrowds parallel algorithm can be found in [4].

2.4. Results and comments

To verify the framerate obtained with the BioCrowds parallel algorithm, we simulated 400 agents with infinitesimal representation, organized in two groups of 200 agents moving in opposite directions in a rectangular area of 40 x 12 m with density of markers equals to 60 markers/m² for two different values of interpersonal area, 0.6 and 1.2 m on a NVIDIA GTS 250 graphic card using OpenCL framework. We evaluated the agents average speed $\bar{v}$ and the average angular variation of their orientation $\bar{\gamma}$. Table 1 shows the result of these simulations.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$R$ (m)</th>
<th>$\bar{v}$ (m/s)</th>
<th>$\bar{\gamma}$ (degree)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel</td>
<td>1.2</td>
<td>1.2</td>
<td>0</td>
</tr>
<tr>
<td>Sequential</td>
<td>1.2</td>
<td>1.15</td>
<td>0.01</td>
</tr>
<tr>
<td>Parallel</td>
<td>0.6</td>
<td>1.2</td>
<td>0</td>
</tr>
<tr>
<td>Sequential</td>
<td>0.6</td>
<td>1.15</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Table 1. Comparison between the sequential BioCrowds algorithm and the parallel BioCrowds using density of marks equals to 60 marks/m². The $\sigma$ represent the standart deviation of value.

The results obtained with parallel BioCrowds have different values compared to the sequencial algorithm. It happens because the algorithm must be adapted to run on GPU, as minimizing control flows operations. However, in both algorithms we observed that an interpersonal area of 1.2 m causes an increase of the angular variation of agents orientation, therefore both algorithms have similar behaviour.

We also obtained the framerate for both algorithms and different densities for the markers on the scenario, which is shown in Table 2.

<table>
<thead>
<tr>
<th>Density (markers/m²)</th>
<th>$R$ (m)</th>
<th>Parallel (fps)</th>
<th>Sequential (fps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1.2</td>
<td>82.09</td>
<td>49.63</td>
</tr>
<tr>
<td>15</td>
<td>0.6</td>
<td>78.62</td>
<td>70.89</td>
</tr>
<tr>
<td>60</td>
<td>1.2</td>
<td>25.67</td>
<td>16.36</td>
</tr>
<tr>
<td>60</td>
<td>0.6</td>
<td>26.21</td>
<td>27.18</td>
</tr>
</tbody>
</table>

Table 2. Framerate obtained simulating crowds with BioCrowds parallel algorithm and sequential BioCrowds using NVIDIA GTS 250 graphic card and Intel® Core™2 Duo 2.13 GHz processor and 4 GB DDR2 with 800 MHz, respectively.

To evaluate the inclusion of pushing effect, we simulated 400 agents moving in the same direction on a scenario with 50 x 12 m with a narrowing on the route for three different algorithms:

Again, the framerate of the BioCrowds parallel algorithm was higher than the rate obtained with the sequential algorithm. Also, we observed that the increase of markers in the scenario influences the performance more than the increase on the number of simulated agents.
1. constant markers on the scenario[1];
2. density of makers varying during the simulation on specific areas predetermined [7];
3. density of makers varying during the simulation when needed ([4] and this work).

Table 3 shows values obtained to average speed and angular variation of agents orientation for both three algorithms.

<table>
<thead>
<tr>
<th></th>
<th>$\bar{v}$ (m/s)</th>
<th>$\bar{\gamma}$ (grau)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.54</td>
<td>13.74</td>
</tr>
<tr>
<td>2</td>
<td>0.59</td>
<td>15.49</td>
</tr>
<tr>
<td>3</td>
<td>0.99</td>
<td>22.88</td>
</tr>
</tbody>
</table>

Table 3. Average speed and angular variation of agents orientation for simulation of pushing behaviour with three different algorithms.

We observed that the increase of markers in specific areas during the simulation allows agents to vary easily their orientation causing an increase on their average speed and facilitating them to change route toward their goal. Visually, the inclusion of pushing effect reduces space among agents, as shown in Figure 2.

![Figure 2. (a) Simulation without pushing effect (Sequential BioCrowds). (b) Simulation of pushing effect with parallel BioCrowds.](image)

3. Final comments

In this paper we presented an implementation of BioCrowds [1] using GPU to increase the performance of the simulator. The results presented indicate an improvement on the performance even using a single GPU. From the point of view of accuracy of the results, the parallel algorithm, as discussed in Section 4 presents a deviation from the value obtained in the original algorithm, which did not affected the visualization of the simulation. As a sequence of this work, we are implementing routines on the parallel BioCrowds to run on multiples GPUs simultaneously. The complete work can be found on [4].

References