Real-Time User Interaction within Socio-Physical Systems

Corina Bianca Alb
Technical University of Cluj-Napoca
Str. Memorandumului, nr.28, Cluj-Napoca
corina7bianca@gmail.com

Dorian Gorgan
Technical University of Cluj-Napoca
Str. Memorandumului, nr.28, Cluj-Napoca
dorian.gorgan@cs.utcluj.ro

ABSTRACT
In this sample paper you can find information regarding the study of human interaction with a computer application that manipulates massive amount of information in real-time. Here are discussed the matters of assuring high performance and responsiveness of the system when interacting with a human user. It is also presents a way to establish efficient communication between entities that form crowds and the user input: how to apply external forces to dynamic entities.

Author Keywords
Entity; GPU; interaction; atomic; obstacle; target; parallel computation; User input; manipulation; cluster;

ACM Classification Keywords
HCI

General Terms
Human-Computer Interaction; Socio-physics; Clustering; Entity; User input; GPU; Parallel and distributed computations; Interface responsiveness;

INTRODUCTION
Human-Computer Interaction is a wide area of study that brings together multiple disciplines, like: computer science, behavioral science, design and media studies. Those are just a few fields of interest that are also tangent to the subject of this paper.

The need for a fast means of computation in case of massive amounts of data is a matter of computer science innovation. GPU uses parallel and distributed methods in order to increase the computational performance. This paper shows examples of calculus based on those GPU methods of parallelism.

Another aspect is represented by the behavioral sciences. This is a field that requires the analysis of huge quantities of data. This is where GPU comes in handy. The analysis of information is done real-time and the responsiveness of the application assures a good user experience. Socio-physics is a branch of behavioral sciences that also uses mathematical and physical tools in order to understand the behavior of human crowds.

Socio-physical systems are representations of complex, real life interaction processes between diverse dynamic entities The interactions can be abstracted into simple mathematical operations. The individuals/entities are characterized by a series of forces and relations, out of which we differentiate in this study: (a) attraction towards a target or another individual and (b) rejection between an individual and another individual or obstacle. The study of those processes requires a means of simulation and real time visualization of scenarios; the interaction between entities must also allow to be manipulated by a user.

This paper has as purpose the development and analysis of some flexible and scalable solutions for real-time execution of socio-physical scenarios using graphical processing units (the GPU provides a good environment for real-time manipulation of massive quantities of data by using parallel computation mechanisms).

In order to achieve our purpose, we simulate generalized functional patterns applied on entities. Those are described as attraction forces, rejection forces, movement towards an end point (objective point/target) and the concept of obstacle and. Regarding crowd simulation, multiple entities can be grouped in so-called clusters. Entities in the same cluster are characterized by similar behaviors in response to external events; see the figure 1 below for a graphical representation. Each group is defined by one or more atomic actions, just as the ones presented earlier (rejection, attraction etc.). An entity can be part of multiple clusters or none, thus the increasing diversity: there are multiple possible combinations of atomic functions/actions resulting in the creation of complex behaviors.

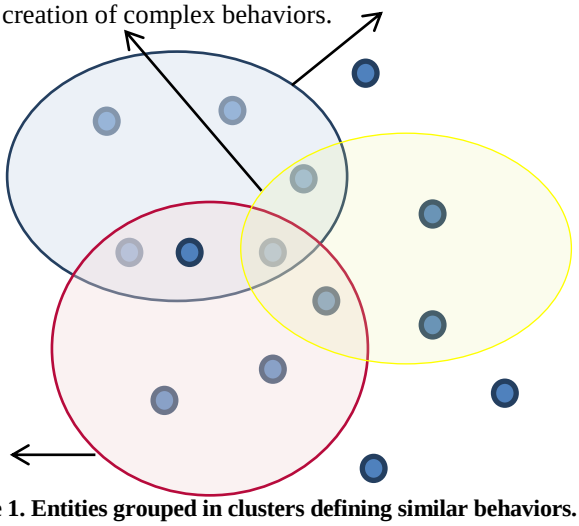

Figure 1. Entities grouped in clusters defining similar behaviors.

Another purpose of this paper is to include user input responsiveness. That means that the user of an application such as the one presented here can interact with the entities and influence their behavior. This has great potential for further development of socio-physical oriented studies: it includes real-time and flexible manipulation of data. This enables the possibility to study more complex scenarios while maintaining a generalized pattern description for basic actions. The main idea is that the way or order in which the actions take place can be influenced by an outside force – the user.

This paper presents some simple atomic operations that form the basis of user interactivity. The approach improves the usability of the system by enabling the creation and manipulation of data into multiple scenarios. Visualization of data is also described here as the interface and the means of communication between the human user and the system: this way, the evolution of the scenario is interactively decided.

RELATED WORKS

The related works that have formed the motivation of this analysis are based on socio-physics study and crowd simulation. An important aspect is that those rely on the applicability and simulation of a group dynamics on GPU. Thus, one of the papers studies the simulation of crowd dynamics as a socio-physical model using graphical processing unit parallelism techniques [1]. The matter of performance is an important aspect when having to deal with diverse and interdependent computations on big quantities of data. This requires a specialized type of hardware architecture. The paper mentioned above is a study that analyzes the techniques used by the GPU in order to improve the performance of computations in comparison with the CPU. It focuses on high level optimization of parallel computations with application in graphical simulations.

Another work continues the analysis of performance by means of hardware improvement. Architecture for real time crowd simulation using multiple GPUs [2] relies on dividing the amount of data to be manipulated between multiple GPU units. The architectural aspect plays an important role when it comes to realistic representation of situations that require huge number of entities. Multiple GPUs on the same machine or organized as a cluster are used for creating realistic crowd simulation application. It is one of the first studies to approach this method. Even though this last presented paper describes a major performance improvement, the current study does not mainly rely on it. The major focus is the analysis of the forces that influence the behavior of entities and how it can be manipulated by the end user (the human interaction).

The next study, Social force model for pedestrian dynamics [3] has as purpose the analysis of the physical concepts that describe behaviors. It formulates a social force model using concepts on which the human behavior is based. The main focus is the study of pedestrians' motion. Just as *this* paper wants to emphasize and analyze, this publication concludes that pedestrian motion can be classified according to environment and other pedestrians' movement. A certain subject has a goal, a location on the map he/she wants to arrive to. If her/his movement is not disturbed, the motion is simple and takes the shortest path. If on the other hand, the motion of a subject is influenced by another subject, there appears what we describe in our paper, a rejection force towards an entity: their positions on the map cannot overlap. There has to be implemented a mechanism/pattern of avoidance: an alternative path has to be found. The *Social force model for pedestrian dynamics* study mentions also the concept of border. Those are represented by buildings, streets, walls, or in a general term, obstacles. Here, a *repulsive effect* is described. The pedestrian needs to acknowledge the presence of such an obstacle and avoid it; this is presented *here* as rejection force towards an obstacle. Another possible action is the *attractive effect*. Pedestrians can be attracted by other pedestrians. The correlated action in our study is the attraction force towards another entity. This related work describer mathematical formulas for each type of *effect* in order to create a generalized pattern that can be applied in similar situations for a large number of subjects/pedestrians.

Our study has a similar purpose: studying alternative methods of efficient pattern generation for entities interaction between each other and between them and other objects classified as obstacles.

TYPE OF INTERACTIONS

As presented in the introductory part, the entities in the system are influenced by a number of forces, for all of which we want to describe a functional pattern. In addition to those (the *attractive effect* and the *repulsive effect*) the user input is introduced. It can be classified as a special kind of influence because it is an outside force that can manipulate the existing patterns in order to create new specific scenarios.

We also want to take advantage of the GPU's specialized architecture for massive data representation and parallel computation capability for performance improvement purposes.

Attraction towards a target

The first and most basic type of force that influences an entity is the attraction towards an already known target when there is no obstacle or other entity to influence the trajectory. Refer to figure 2 for an example of how this is represented graphically.

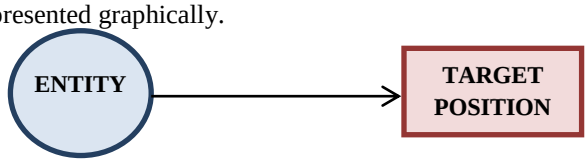

Figure 2. Attraction force of an entity towards a fixed target.

For visualization, the position of the moving entity is updated at each frame until it reaches the destination.

When a single entity is involved, the performance of the computations on the CPU is comparable with the one on the GPU. The real advantage is observed when there are a huge number of entities reaching their targets. Figure 3 represents symbolically the possibility of having multiple entities moving toward different targets.

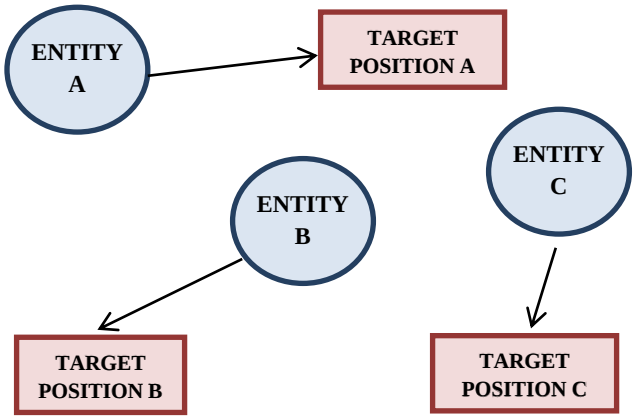

Figure 3. Scalability of the number of entities performing computations in order to reach their targets.

For a CPU implementation this kind of scenario produces glitches at visualization because of the massive amount of computations that need to be done.

The attraction towards a target gets more complicated when obstacles are involved. This is presented in further sections, as it involves more complex analysis of the environment that surrounds the entity.

Attraction towards an entity

The next type of attraction is the attraction towards another entity. This is a more complex type of force because, despite the case of a target -which is a fixed point, the subject A has to deal with a moving entity B, as suggested in figure 4. This means that at each frame the attracted entity A considers as target the current position of the entity B, which in turn is updated at every frame. The subject B could be following its own path towards a target.

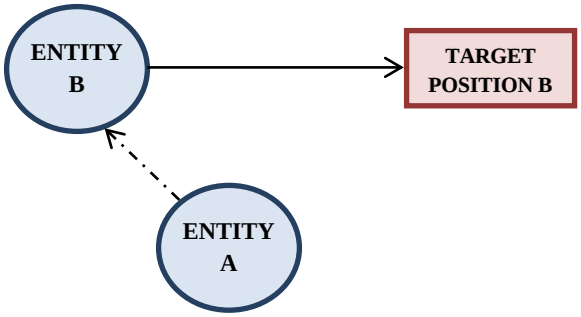

Figure 4. Entity A is attracted to the moving entity B.

To make the study more complex, let's consider the motion of B follow the model of a bouncing ball that is dropped from a distance X from the ground. There is a gravitational force that attracts it to the ground, as well as a rejection force coming from the ground and walls: this makes the ball bounce. For a more realistic visualization, there is also a speed modifier variable that makes the ball slow down as it touches the ground and walls. This effect is achieved by adjusting the velocity of the ball using the *Verlet integration* [4]; see in figure 5 the mathematical expression of the Verlet integration: the variable *t* represents the *time*, *v* is the *velocity* and *a* represents the *acceleration*; this is the physical interpretation of the formula. The method is often used in computer graphics for performance reasons, as it uses simple computations only: additions and subtractions functions of time. Programmatically, the velocity, time and acceleration have slightly different meanings: *time* is expressed as *frames* that are to be displayed; the *velocity* is actually the position of the entity at a given frame, while the *acceleration* represents the difference between the previous positions of the entity. As an example, at frame F1 (time t1), entity E has position P1, at frame F2 (time t2) it has position P2 and at frame F3 (time t3) it has position P3. Acceleration *a* is equal at frame F2 with P2-P1 and at frame F3 with P3-P2. Considering that time variance (*delta t*) represents one frame, Verlet integration uses the mean on the previous and current accelerations in order to determine the current velocity.

$$\vec{v}(t+\Delta t) = \vec{v}(t) + \frac{\vec{a}(t) + \vec{a}(t+\Delta t)}{2}\Delta t.$$

Figure 5. The mathematical expression of Verlet integration.

To summarize, there are three physical forces that act towards entity B: (1) the gravitational force, (2) the repulsive/rejection reaction of the ground and walls and (3) the bouncing slowing down force.

Coming back to entity A, it has to adjust its position taking into consideration entity B. The result is a mirrored bouncing effect: A follows the pattern that B's movement describes.

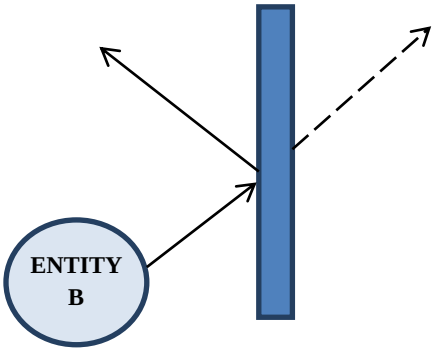

Figure 6. Entity B (simulation of a ball) is bouncing to the wall. The wall is an obstacle that reverses B's direction of movement.

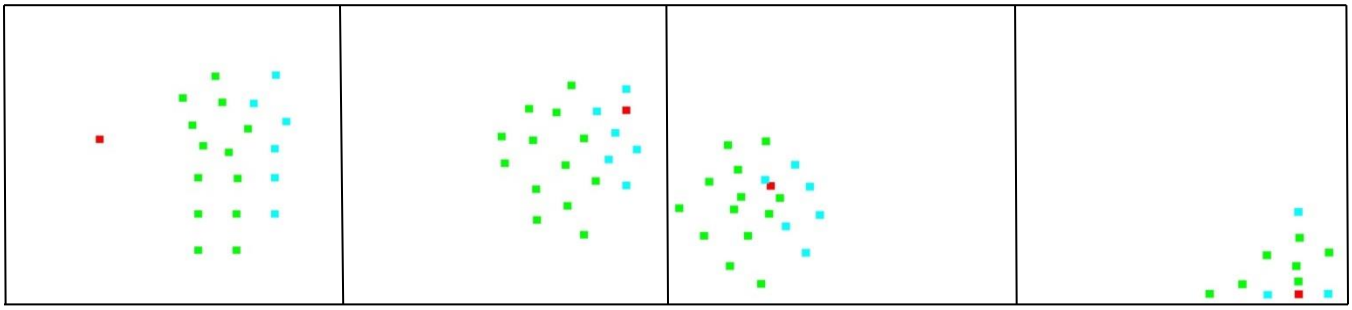

Figure 7. Rejection force between multiple entities following the same moving target.

Rejection force between an entity and an obstacle

Before going on to the obstacle concept, notice that in the previous paragraph we have talked about the bouncing effect of the entity B when it comes in contact with the walls. Figure 6 depicts this scenario. This is actually a rejection force where the walls are obstacles: they reverse B's direction of movement: the walls don't let the ball (entity B) pass through it.

Another kind of obstacle is represented by static objects that might interfere with the trajectory of moving entities. In this case, the moving subjects must find alternative ways to reach their target by avoiding the obstacles. A more complex mathematical approach is needed, thus the justification of GPU performance.

Rejection force between entities

Rejection can also take place between subjects of the same type: moving entities. Let's consider the bouncing ball use-case presented earlier. This time, instead of having a single subject that is attracted to entity B, there are more. We have to take into consideration the fact that all entities that are attracted to the same point might overlap. In a 2D environment, this is not a realistic representation of motion. This is why we need to find a way to implement the rejection between entities at the same time as attraction towards the same bouncing subject B. This is done just like in the case of static obstacles, only this time each entity is an obstacle for all the others. At each frame, all entities change positions, but for each current position, all subjects adjust their coordinates in the 2D space they move in. Refer to figure 7 presented above in order to observe the evolution of such a scenario.

This use-case is fit for observing and analyzing some scalability performance. As also mentioned previously, the computations' complexity is high. A CPU implementation does introduce glitches and a poor visualization. On the other hand, the GPU memory and computation capacity are high enough to not only ensure a smooth visualization, but also scalability. Depending on the hardware specifications, the quality of the visualization remains unaltered for a certain number X of entities. When this threshold X is outrun, the system becomes slower due to the amount of time it takes to complete all the computations for all data (the amount of time to complete all the computations necessary for displaying a frame); the level of parallelism is limited by the hardware.

The system has been implemented and analyzed in order to objectively study the scalability of this scenario. The use-case relies on increasing the number of entities that are attracted to the bouncing ball B (see table 1): for a number of 10 entities, it takes approximately 400 microseconds to render a frame in GPU. The same number of entities is rendered on CPU in only 100 microseconds. For 100 entities, the amount of time on GPU slightly increases, reaching approximately 500 microseconds per frame. Operations on CPU take 400 microseconds. It takes 800 microseconds to perform all computations for 500 subjects on a GPU, while on a CPU it takes 4500 microseconds. From this point on, the advantage in performance introduced by the parallelism implemented on the graphical processing unit is evident: see table 1 for a representation of the scalability with regard to time performance (some errors might be introduces due to approximations). For a further analysis, when the number of entities is increased to 2000, the GPU takes approximately 4000 microseconds while the CPU 10 times longer: 40000 microseconds.

Entities number	GPU computation time (microseconds)	CPU computation time (microseconds)
10	400	100
100	500	400
500	800	4500
1000	2000	10000
2000	4000	40000

Table 1. Number of entities scalability and time performance.

CLUSTERING

Clustering is the technique used to group together entities that act similarly in given situations. The behavior of the group is described rather the one specific for each individual. Atomic operations, just like the ones presented in the earlier section, are used to define the reaction to a certain kind of interaction. Refer to Figure 1 for a graphical representation. The most important part here is defining the characteristics of the cluster instead of the ones that refer to an individual. Each entity can decide whether or not to be part of one or more defined groups. This way, the behavior

is deduced by the analysis of the influence that all groups bring to a specific entity.

INTRODUCTION USER INPUT

For a more flexible manipulation of the events that take place in the entities' environment, the user input has been introduced: it creates new scenarios that are not explicitly described by the program/application. This kind of interaction is valuable because it improves the usability of the application. Some default patterns can be manipulated in order to create new ones.

Here, it is considered the input from the user for a use-case when the entities cannot decide by themselves how to avoid an obstacle. The user can create the so called *temporary attraction points* that act towards entities blocked by an obstacle. He/she can also control those attraction points by activating and deactivating them at any time.

Introducing input from a user generates use-cases that are handled according to the atomic interactions presented in the previous section: an entity that encounters an obstacle in its way towards a target is blocked. The user's input helps it find an alternative way of reaching the final position. This automatically generates a scenario in which a user has impact; out of all the possibilities, somebody chose a certain one: the entity could avoid the obstacle by turning left or right (up or down), but the user specified to turn left (figure 8).

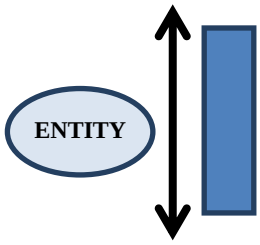

Figure 8. The entity cannot decide by itself which way to avoid the obstacle.

Going further, there are some possible use-cases: (1) The entity might encounter another entity, this generating a rejection force as the one presented earlier (rejection force between entities). Being an atomic operation, the system has a default way of handling this situation (figure 9).

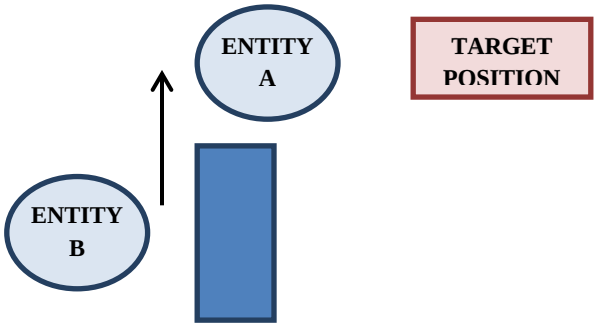

Figure 9. The system finds an alternative way for B to reach its target without overlapping entity A.

(2) Even if there is no other obstacle, the entity follows its path towards the target, which is again an atomic operation (figure 10). Notice that all operations are performed real-time. In situations when more than one outcome is possible, the user can manipulate the default behavior and experiment new scenarios. The responsiveness of the application is an important objective which this approach makes sure to accomplish.

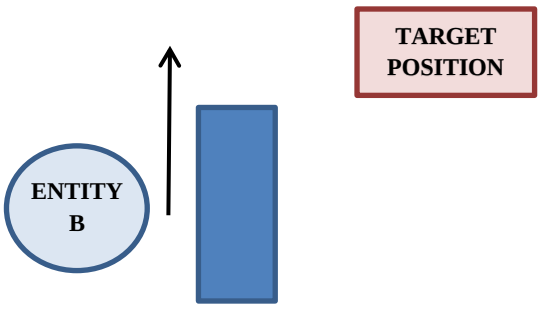

Figure 10. After avoiding the fixed obstacle, entity B follows its target normally, without any other interference.

The user also has control over the number and type of entities that create the scenario. This information is given at the beginning of the application, before the rendering of the scene. This way, he/she can study and measure the performance of the system and scalability, as well as simulating the behavior of the entities for a more or less crowded scenario.

Avoiding obstacles using user input

An example scenario that uses the user's input in order to avoid an obstacle is presented in figure 11.

The red squares denote the moving entities; those have a simple route: starting from the left part of the screen, they want to reach a target on the right part of the screen. The vertical bar is an obstacle and the points cannot move further. The two ends of the bar represent the *temporary attraction points* mentioned earlier. The entities will start moving towards that point once the user activates one (or both) of them; in the case that the user activates both attraction points at the ends of the obstacle, the subjects choose to reach the one which is closer. Temporary attraction holds until the subject reaches it; after that, the entity follows its path to the target.

APPLICABILITY

The previous section - TYPE OF INTERACTIONS presented the basic atomic operations used in the simple application referred to in image 11. The red squares have as purpose reaching a previously known target when there are obstacles on the way. This simple scenario has been implemented in order to represent the possibilities offered by the system for further developing; for example a real-life application for finding a convenient route knowing a set of constraints (dead end, the route is for trains only, obstacles like fences) and also allowing the user to choose; a scenario could be: 'here, you can go to the left where the road is

paved or you could take a shortcut through the park'. Figure 11 depicts the result of simulating such a simplified scenario (some of the red points are stuck, waiting for the user to take a decision).

CONCLUSION

The interaction between humans and computer [5] is a step forward in technological research and development. It enables people to make use of the computation power of a machine in order to solve and analyze complex problems. Human-Computer Interaction is a multidisciplinary field of study: this type of *interaction* assumes, besides computer sciences knowledge, the creation of a design or interface for the human user. This paper has as purpose the study of those types of interactions from a graphical perspective. It focuses on the improvements brought by using the GPU as principal engine of computation for data manipulation regarding the user experience. The four atomic forces presented are the basic tools that an end user can manipulate in order to create complex scenarios.

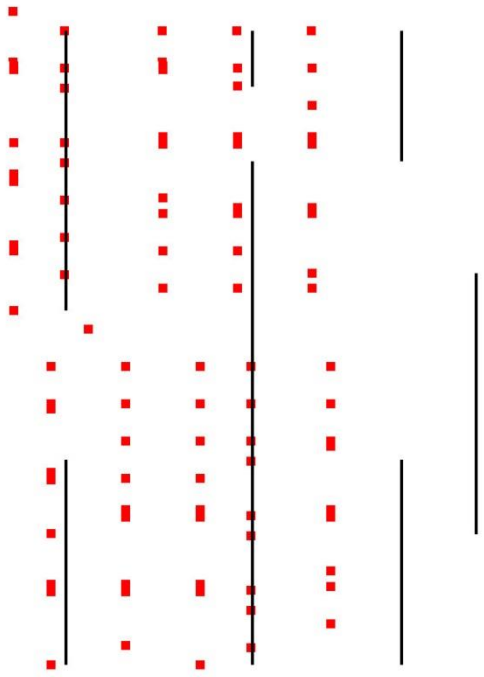

Figure 11. Moving entities (red squares) encounter a vertical obstacle.

Regarding interactivity, here can be differentiated two major categories: (1) interactions between the entities (programmed interaction that cannot be altered) and (2) user input interactions (those interactions that introduce additional data in the system, with the purpose of altering the default scenario described programmatically).

The interaction between entities makes use of the parallel and distributed computation system implemented on the GPU. The performance regarding this aspect is limited by the hardware specifications, but is far more capable than the CPU. The atomic forces: (1) attraction towards a fixed target, (2) attraction towards another entity, (3) the rejection force between an obstacle and an entity and (4) the rejection force between two entities are thus the basic tools for a real-time responsive visualization of data.

Introducing the user input is the next step in developing the interactive study. This creates responsive visualization of scenarios that alter the default(programmed) behavior of the entities. It makes use of the atomic operations presented above.

All in all, massive data manipulation human computer interactive applications can be created by making use of the parallel and distributed systems. The responsiveness is preserved by transporting the computations' overhead to the GPU. The visualization is thus smooth and offers a good user experience.

REFERENCES

1. Stefan Mocanu, Ramona Din, Daniela Saru, Cosmin Popa: Using Graphics Processing Units for Accelerated Information Retrieval. Faculty of Automatic Control and Computer Science, University Politehnica of Bucharest, 060042, Romania. Year: 2014. Article 3. Number of pages: 8.
2. Mark Joselli: Pontificia Universidade do Parana, PUC-PR. Jose Ricardo da S., Esteban Clua: Medi Lab, Universidade Federal Fluminense. An architecture for real time crowd simulation using multiple GPUs. SBC - Proceedings of the SBGames 2014 | ISSN: 2179-2259. Computing Track - Full Papers. Number of pages: 8.
3. Dirk Helbing, Peter Molnar: Social force model for pedestrian dynamics: volume 51, number 5, May 1995. Institute of Theoretical Physics, University of Stuttgart, 70550 Stuttgart, Germany. Number of pages: 5.
4. Benedikt Bitterli: A Verlet based approach for 2D game physics. Math and physics, November 20, 2009 Number of pages: 7.
5. Using research methods in human-computer interaction to design technology for resilience: Armida Guerra Lopes, Centro Algoritmi, University of Minho, Guimarães, Braga, Portugal. Instituto Politécnico of Castelo Branco, Portugal. JISTEM J.Inf.Syst. Technol. Manag. vol.13 no.3 São Paulo Sept./Dec. 2016.
6. Sociophisics: a personal testimony. Serge Galam, Laboratoire des Milieux Desordonnes et Heterogenes, Tour 13, Case 86. CNRS UMR 7603. Number of pages: 7.
7. From CUDA to OpenCL: Towards a Performance-portable Solution for Multi-platform GPU: Peng Dua, Rick Webera, Piotr Luszczeka, Stanimire Tomova, Gregory Petersona, Jack Dongarra. University of Tennessee Knoxville and University of Manchester. Number of pages: 7