FAME, Soft Flock Formation Control for Collective Behavior Studies and Rapid Games Development

Choon Sing Ho, Yew-Soon Ong, Xianshun Chen, and Ah-Hwee Tan

School of Computer Engineering, Nanyang Technological University,
50 Nanyang Avenue, Singapore 639798
{csho,asysong,chen0469,asahtan}@ntu.edu.sg

Abstract. We present FAME, a comprehensive C# software library package providing soft formation control for large flocks of agents. While many existing available libraries provide means to create flocks of agent equipped with simple steering behavior, none so far, to the best of our knowledge, provides an easy and hassle free approach to control the formation of the flock. Here, besides the basic flocking mechanisms, FAME provides an extensive range of advanced features that gives enhanced soft formation control over multiple flocks. These soft formation features include defining flocks in any user-defined formation, automated self-organizing agent within formation, manipulating formation shape at real-time and bending the formation shape naturally along the curvature of the path. FAME thus not only supports the research studies of collective intelligence and behaviors, it is useful for rapid development of digital games. Particularly, the development cost and time pertaining to the creation of multi-agent group formation can be significantly reduced.

Keywords: soft formation, flock, behavioral animation, collective behavior, games.

1 Introduction

Advancements in the field of digital games have evolved significantly in the recent decades and becoming ever more complex in game content. In early generation of digital games, due to the limitation of the computer systems, game contents and mechanics are kept relatively simple. As computer systems are packed with more memory and computational power, games development studios are trending towards creating new generation of digital games with substantially rich digital contents, while some bearing close resemblances to the real world. Citing the recent popular Grand Thief Auto IV (GTA) digital game, for example, which showcases an open world adventure game taking place in a fictional city with over 2000 buildings that is designed based on modern day New York. Thus, modeling of natural and social phenomena including human crowd behaviors and the flocking behaviors within groups of animals is fundamental to instill the traits of vibrancy and realism in the virtual city.

L.T. Bui et al. (Eds.): SEAL 2012, LNCS 7673, pp. 258–269, 2012.

While several libraries and source codes pertaining to flock behavioral modeling are readily available on the web, it is noted that most provide only means to achieve simple steering behavior. Although the steering mechanisms provide a good underlying building block to achieve natural and realistic motion, these libraries however, do not give high level control over the formation of the group. In the real world, we can easily observe that some animals tend to flock together in some shape or formation, for a common good – to survive. Hence, a software library equipped with comprehensive set of tools to allow easy creation of group behavioral animation with flexible controls over the formation shape of the flock would be helpful for collective behavior studies and rapid development of digital games. We envisioned that this formation constraint should be a soft one, s.t.:

1. *Soft formation constraint*: the agent should be able to temporarily deviate from their formation when encountering anomalies, such as obstacles, steep terrains, and other agents, and return to its formation when the path is clear.
2. *Self-organizing agents*: the agent should not be fixed to its assigned position in the formation, but able to self-organize to fill up the formation quickly and naturally.
3. *Real-time formation control*: the formation should be able to change or molded according to users preference during runtime.
4. *Flexible flock formation*: when the group is navigating in the environment, the formation should bend naturally along with the curvature of the path.

In this paper, we present Flocking Animation Modeling Environment denoted in short as FAME, a soft flock formation library which provides extensive controls over the flock formation. One significant advantage is that development cost and time pertaining to creation of group behavioral animations can be significantly reduced. In what follows, we will give an overview of some related work. Next we will show an overview of the system architecture deployed in FAME as well as some of the key implementation details. Subsequently, we present a successful game implemented using FAME, followed by brief concluding statements.

2 Related Works

The study on collective intelligence and cooperative behaviours among agents is an ongoing and interesting area of research [1,2]. A popular approach to simulate large crowds of animated characters is through the use of behavioral models. A behavioral model involves receiving stimuli from the environment and provides an appropriate response [3]. In the literature, three core classes of behavioral models can be identified to date to create group movement animation. They are the particle system, the flocking system, and the behavioral system.

The particle system is a fast, physics based and computationally cheap approach to simulate the movement of large number of agents. Agents controlled by the particle system approach respond to a global force that defines the direction of movement while avoiding collisions. Some studies based on the particle system are listed as follows. Hughes introduces a continuum model which bears

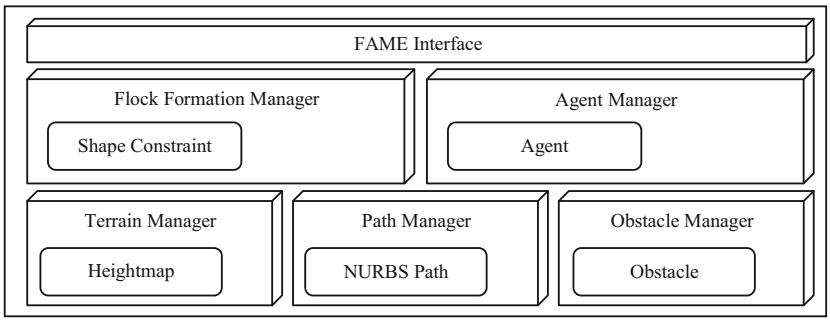


Fig. 1. FAME system architecture

close similarities to classical fluid dynamics to study on the flow of a crowd of pedestrians [4]. Chenney introduced "flow tiles" [5], a tiled based method that defines the motion of crowd for each segment that makes up the entire map.

Flocking system, on the other hand, is composed of several simple heuristics rules to simulate the natural and well-coordinated group movement behaviors such as flock of birds, a school of fish and a horde of animals in the virtual computer environment. First published by Craig Reynolds in 1987, Reynolds demonstrated a "Boids" computer simulation system [6] which simulates flock behavior using three simple rules, namely cohesion: to move towards the averaged position of all nearby agents, alignment: to steer and move in the same direction with nearby agents, and separation: to move away from nearby agent that are too close in proximity to avoid collision. This simple computer simulation has thereafter spin-off a vast array of useful real world applications including artificial life simulation, crowd simulation, military simulation, robotics, movie productions and computer games.

Behavioral system controls the agents using the rule based approach. As such, agents using the behavioral system approach are often seen as being smarter than the flocking system. An interesting research in this area is the simulation of artificial fishes by Tu et al. [7]. The research group created a virtual marine world simulation, and realistically emulated the appearance, movement and behaviour of the fishes. The behavior of the fish is driven based on the states of the fish and its intention.

Recent research interest in this field of study has been seeking for more efficient ways to achieve more realistic rendering through improved rendering techniques [8]; refining navigation algorithms to create more natural movement of autonomous agents [9,10,11,12]; increasing the number of agents that the system can handle in real time through the use of dedicated hardware, specifically the GPUs [13,14,15]; and by reducing computational cost through optimization of the data structure and efficient management of system resources [16,17].

With the maturing of work in the field, there is an increasing trend to develop toolkits to speed up development process. Recently, Shawn et al. developed a set of tools, library and test cases to ease development effort as well as to measure

steering performance of individual agents [18,19]. Similarly, Erra et al. [14] developed BehaveRT, a GPU-based library to visualize large scales of autonomous agents. However, to the best of our knowledge, none of these libraries focus on formation control in groups of autonomous agents.

3 FAME API and Architecture

In this section, the system architecture of FAME is depicted in Figure 1 and composes the following core modules: *FAMEInterface*, *FlockFormationManager*, *AgentManager*, *ObstacleManager*, *PathManager* and *TerrainManager*.

FAMEInterface: This is the point of communication between the game application and FAME. It facilitates the creation and removal of flock agents. Upon initialization, FAME computes the new positions and orientations of each the agent in the flock, for given frame rate of interest, which is a user-defined parameter.

FlockFormationManager: The flock formation manager manages the creation and removal of flock groups with the desired formations. Here in FAME, each group of agents is bounded by a polygonal shape constraint. Each shape constraint is identified by a unique index upon creation and stored within a hash. This is to facilitate fast retrieval of flock object when a query is performed.

AgentManager: The agent manager handles agents at the individual level. Similar to the flock formation manager, the agent manager assigns a unique index to each agent and is stored in a hash table upon creation to facilitate fast retrieval of agent object upon query. Each agent is assigned to a flock group at one time. Within each flock group, the agent has a uniquely assigned destination location in the map, which accords to the desired formation defined.

ObstacleManager: The obstacle manager handles the list of obstacles in the scene at real time. Obstacles in FAME can be defined using a circular object or polygonal representation. Upon creation of the obstacles, the obstacle object is stored using a grid data-structure so as to reduce the real time requirements in computational effort when performing queries on them.

PathManager: The path manager handles the list of paths planned within the game environment. In FAME, a path is defined by a set of control points modeled using Non Uniform Rational B-Spline (NURBS).

TerrainManager: The terrain manager handles the terrain related properties in a game, such as the terrain size and surface elevation data. These information is used to the control movement of the agents, such as the maximum gradient that the agent is allowed to climb, and to adjust the movement speed accordingly when the agent travels uphill or downslope.

4 Soft Formation Features

In this section, we present several soft formation mechanisms provided in FAME for simulating large groups of agent flocking in a natural and believable manner, while maintaining the desired formation. These mechanisms provided includes:

1)*defining flock of irregular formation*, 2)*self-organized agent within formation*, 3)*formation shape morphing*, 4)*flexible formation path following* and 5)*seamless space partitioning mechanism*. Implementation details are as follows:

4.1 Defining Flock of Irregular Formation

FAME supports creation of flock formations that are formulated in the form of soft polygonal shape constraints. Thus agents belonging to a particular group shall stay within the defined polygon when possible. The shape constraint is populated with agents by first performing a uniform sampling, to generate the same number of sample points on the polygon. The sample points are then assigned as destination position P^{Dest} to each members of the flock. After which, the guiding steering force will attract the agents towards their individually assigned position.

Some properties of this shape constraint include a set of control points defining the shape of the flock, a list of sample points defining the agents' position in the formation, the flock centroid and orientation. This facilitates a fast and easy transformation control (translation, rotation, scale) over the flock formation in the virtual environment.

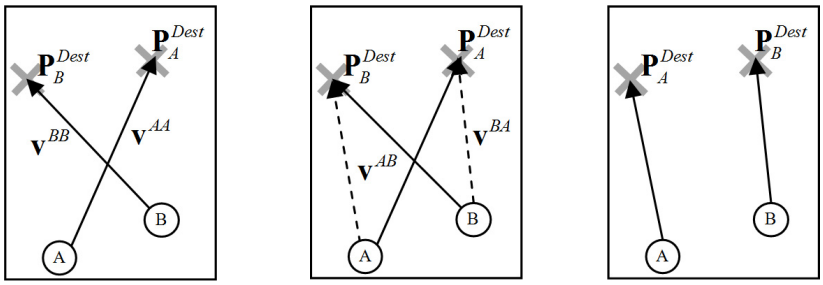
- *Translation*: The group can move around by redefining the flock centroid position.
- *Rotation*: The group can reorientate to face a particular direction by rotating the control points as well as the sample points.
- *Scale*: The group can expand or shrink the size of the formation by scaling the control points as well as the sample points.

4.2 Self-organized Agents Within Formation

It is natural that while the agents are forming the formation, those who arrived first should move inwards to fill up the front spaces while those whom arrive later will fill up the rear. Here, FAME provides such feature to automatically reorganize the agents in the formation during runtime. As the agents navigate in the virtual environment, some might encounter anomalies, such as obstacles, steep terrain and other agents of different flock groups. Thus, these agents would have to make a detour and take a longer time to move back to the formation. Other agents that manages to arrive at the formation first would move inwards, allowing agents that arrive later to fill up the formation from the rear. Implementation detail is described as follows: Given an agent A and its randomly picked neighbour agent B, let $\mathbf{P_A}$ and $\mathbf{P_B}$ be the initial position of agent A and B, respectively, while $\mathbf{P_A^{Dest}}$ and $\mathbf{P_B^{Dest}}$ denote the final destinations of agent A and B, respectively. First the following direction vectors to destination are calculated,

$$\mathbf{v^{AA}} = \mathbf{P_A^{Dest}} - \mathbf{P_A} \tag{1}$$

$$\mathbf{v^{BB}} = \mathbf{P_B^{Dest}} - \mathbf{P_B} \tag{2}$$



a) Calculate V^{AA} and V^{BB} b) Calculate V^{AB} and V^{BA} c) Swap if criterion is met

Fig. 2. Criterion for two agent to swap destination

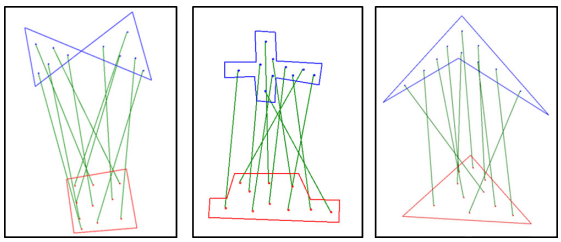


Fig. 3. Examples of agent-destination mapping from the original positions(bottom) to the destination positions in the new formation(top)

$$\mathbf{v^{AB}} = \mathbf{P_B^{Dest}} - \mathbf{P_A} \tag{3}$$

$$\mathbf{v^{BA}} = \mathbf{P_A^{Dest}} - \mathbf{P_B} \tag{4}$$

where $\mathbf{v^{AA}}$ is the directional vector from agent A's position to agent A's destination while $\mathbf{v^{BB}}$ is the directional vector from agent B's position to agent B's destination. Similarly,$\mathbf{v^{AB}}$ is the directional vector from agent A's position to agent B's destination and $\mathbf{v^{BA}}$ is the directional vector from agent B's position to agent A's destination. The agents will thus swap their destination position if the following criterion is met (see Figure 2).

$$||\mathbf{v^{AA}}|| + ||\mathbf{v^{BB}}|| > ||\mathbf{v^{AB}}|| + ||\mathbf{v^{BA}}|| \tag{5}$$

which can be computed faster by just comparing the square distance of each vector, such that

$$\mathbf{v^{AA}} \cdot \mathbf{v^{AA}} + \mathbf{v^{BB}} \cdot \mathbf{v^{BB}} > \mathbf{v^{AB}} \cdot \mathbf{v^{AB}} + \mathbf{v^{BA}} \cdot \mathbf{v^{BA}} \tag{6}$$

It is noteworthy that the above reorganization process does not need to be calculated for every time frame as the agents position is highly unlikely to change drastically. Rather it is sufficient to calculate periodically after every few seconds to reduce the computational requirements. The algorithm leverage on the

neighborhood calculation process when choosing a random neighboring agent to swap their destinations, which is already an incurred computation cost needed to compute the steering forces, and hence the additional computation cost incurred is minimal.

4.3 Formation Shape Morphing

FAME allows user to change/morph the formation shape easily during runtime. Shape morphing refers to the process where a given shape transforms into another shape. In the context of shape constraint flocking, agents housed in a shape formation constraint morph or transform into another formation shape, while filling up the space inside the new formation strategically and naturally. Here, we showcase two formation morphing scenarios: 1) morphing of formation shape from a given initial formation shape to a new user-defined formation shape of interest and 2) a shape deformation approach to changing of formation shape.

Morphing Approach. In the first scenario, we consider how the flock of agent change its current formation to take up a new formation shape. First, a new formation has to be defined. Next, a uniform sampling is performed on the new formation to generate a set of n evenly distributed points inside the shape, where n refers to the number of agents in the initial formation. These points define the new position of that the agents should move to in order to fill up the new formation shape. Finally, a one-to-one mapping process is performed to assign every agent to its new destination position in the new formation shape. The desired mapping should be such that the total Euclidean distance from each agent's current position to its new destination is small. This is to allow the agent to form the new formation within the shortest possible time. The depictions of the mappings result considered in FAME are depicted in Figure 3.

Shape Deformation Approach. The second scenario showcases how agents adjust their positions within the formation when it undergoes some shape deformation. The method proposed in previous scenario works well in cases when the formation only requires to undergo one shape deformation. However it is computationally expensive when the changes to the formation shape is minor

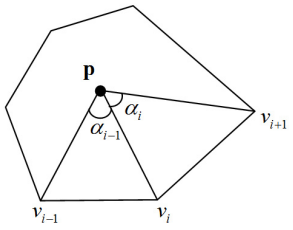


Fig. 4. Formation Shape Constraint

and continuous over time. Hence, this method is able to complement and make up for the drawbacks in the previous method.

Here, FAME uses the mean-value coordinate to calculate the Barycentric weight of each control point on the position of the agents inside a formation of any arbiturary shape. When the initial shape formation constraint is created and populated with n agents, the Barycentric coordinates is calculated for each of the initial position $P^{init} = \mathbf{P_0}, \mathbf{P_1}..., \mathbf{P_{n-1}}$ such that

$$\mathbf{p} = \sum_i B_i(\mathbf{p}) v_i, \ s.t. \sum_i B_i(\mathbf{p}) = 1 \tag{7}$$

where $\mathbf{p} \in P^{init}$ is the coordinate of the i control points defining the formation (see Figure 4). The mean value coordinate $B_i(\mathbf{p})$ is calculated as follows,

$$\sum_i B_i(\mathbf{p}) = \frac{w_i}{\sum_{j=1}^{k} w_j}, \ and \ w_i = \frac{\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2)}{||\mathbf{v_i} - \mathbf{p}||} \tag{8}$$

where $\alpha_i, 0 < \alpha_i < \pi$ is the angle formed in the triangle $[p, v_i, v_{i+1}]$.

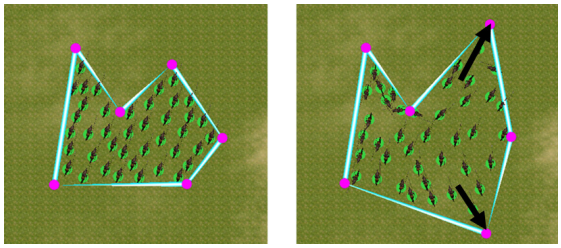


Fig. 5. Shape deformation approach to change formation from original formation (left) to the new formation (right)

With the computed Barycentric coordinate, the new position of the agent $\mathbf{p^{new}}$ can be easily determined when adjustments are made to the control points. A screen-shot of the shape morphing result using the deformation approach is depicted in Figure 5.

4.4 Flexible Formation Path following Mechanism

Path finding and navigation is one of the essential elements in digital games that allows game agent to navigate around in a complex environment in a natural and realistic manner. In addition, game environments are often non-static as such moving obstacles are commonly present in the scene and could also be introduced during runtime. Hence, it is natural for a flock of agents to be able to react and steer away from obstacles encountered in the scene autonomously, while at the same time the group formation shape should bend naturally along the path's curvature while negotiating the path. Here, FAME provides the functionality

to define a path in the form of a NURBS. Each flock group can be tasked to perform path following along the specified path and while doing so, FAME automatically bends the shape constraint along the path's curvature, resulting in a more natural and realistic movement (see Figure 6).

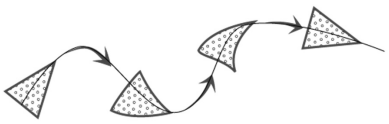


Fig. 6. Flexible formation - Ability to bend the formation along the path's curvature

Implementation details can be found in our previous work on Autonomous Multi-agents in Flexible Flock Formation [12].

4.5 Seamless Space Partitioning of Agent Object

FAME provides a seamless approach to partitioning the game environment, requiring no explicit efforts from the user or game developer, to support creation of a larger number of agents running in real time. One well known pitfall of the classical flock model is that each agent's steering decision is made based on the positions of the neighboring agent that lies within its visible radius. A brute force neighborhood query process would require the distance between every pair of agents to be computed, which would incur a computation complexity of $O(n^2)$. Such an approach does not scale well with large number of agents.

The underlying mechanism utilizes a quad tree data structure to subdivide the entire map into multiple quadrants. After game agents are populated, FAME performs a recursive search to place the agent in the respective leaf node of the quad tree and updates the node whenever the agent travels across quadrants. This speeds up the neighborhood query process as agents lying in quadrants that does not intersects in the region of interest are rapidly eliminated. This divide and conquer approach brings about a reduction of computational complexity from $O(n^2)$ to $O(n \log n)$.

5 FAME in Successful Commercial Game Launch

FAME library package was used successfully in collaboration Singapore-MIT Gambit Game Lab in the development of a commercial game, Dark Dot. Dark Dot is an action shooter games released on the Apple iPad platform, where the player controls a group of minions known as Darklets in a quest to defeat the enemy (see Figure 7). Here in Dark Dot, the advanced flock formation mechanics provided by FAME is used to control the locomotion of the Darklets.

Shortly after it was released in Oct 2011, Dark Dot became the top action game in 48 countries and clinched the number one position for several months

Fig. 7. Dark Dot created by Singapore-MIT Gambit Game Lab (Screenshots taken from Dark Dot official website)

in the iTunes apps store. It received notable comments from the press, citing Dark Dot as *"not only fun and creative and lovely to look at, but it's got some stunningly original game-play mechanics."*

Interested reader can visit the the Dark Dot official website for more information about the game: `http://gambit.mit.edu/loadgame/darkdot.php`. Similarly, more information about FAME can found in the following website: `http://c2inet.sce.ntu.edu.sg/FAME/`

6 Conclusion

In this paper, we have described FAME, a comprehensive C# library designed for easy creations of soft flock formation controls. In this soft formation control, the agents are able to temporary deviate from their formation when encountering anomalies and return to its formation when the path is clear, as well as being able to self-organize within the formation to reduce the overall time required to fill up the formation. Similarly, the formation is able to change or molded according to users preference during runtime and is able to bend naturally along with the curvature while negotiating the path. FAME also provides a seamless way to partition the scene so that the overall computation requirement is reduced. Using FAME technologies, an iPad game – Dark Dot, was launched on Apple iTunes app store and became the top action game in 48 countries. As future work, we hope to leverage from the emerging field of memetic computing technologies [20,21] towards more intelligent behaviors, decision making, social interaction and cultural evolution among virtual agents in games. Also, we shall consider the use of delicated hardwares [22] for real-time performance when intensive AI technologies are deployed.

Acknowledgment. We would like to thank Singapore-MIT GAMBIT Game Lab for the funding and support provided for this project.

References

1. Winfield, A., Erbas, M.: On embodied memetic evolution and the emergence of behavioural traditions in robots. Memetic Computing 3, 261–270 (2011)
2. Satizábal, H., Upegui, A., Perez-Uribe, A., Rétornaz, P., Mondada, F.: A social approach for target localization: simulation and implementation in the marxbot robot. Memetic Computing 3, 245–259 (2011)
3. Durupinar, F.: From audience to mobs: crowd simulation with psychological factors. Ph.D. dissertation, Bilkent University (2010)
4. Hughes, R.: The flow of human crowds. Annual Review of Fluid Mechanics 35, 169–182 (2003)
5. Chenney, S.: Flow tiles. In: Symposium on Computer Animation. SCA (2004)
6. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. In: Computer Graphics and Interactive Techniques. SIGGRAPH (1987)
7. Tu, X., Terzopoulos, D.: Artificial fishes: physics, locomotion, perception, behavior. In: Computer Graphics and Interactive Techniques. SIGGRAPH (1994)
8. Aubel, A., Boulic, R., Thalmann, D.: Real-time display of virtual humans: levels of details and impostors. IEEE Transactions on Circuits and Systems for Video Technology 10(2), 207–217, Comput. Graphics Lab., Swiss Fed. Inst. of Technol., Lausanne, Switzerland
9. van den Berg, J., Lin, M.C., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: IEEE International Conference on Robotics and Automation, pp. 1928–1935 (2008)
10. Sean, C., Jamie, S., Dinesh, M.: Way portals: Efficient multi-agent naviation with line-segment goals. In: Proc. ACM SIGGRAPH Symp. Interactive 3D Graphics and Games - I3D (2012)
11. Lin, M.C., Sud, A., Van den Berg, J., Gayle, R., Curtis, S., Yeh, H., Guy, S., Andersen, E., Patil, S., Sewall, J., Manocha, D.: Real-Time Path Planning and Navigation for Multi-agent and Crowd Simulations. In: Egges, A., Kamphuis, A., Overmars, M. (eds.) MIG 2008. LNCS, vol. 5277, pp. 23–32. Springer, Heidelberg (2008)
12. Ho, C.S., Nguyen, Q.H., Ong, Y.-S., Chen, X.: Autonomous Multi-agents in Flexible Flock Formation. In: Boulic, R., Chrysanthou, Y., Komura, T. (eds.) MIG 2010. LNCS, vol. 6459, pp. 375–385. Springer, Heidelberg (2010)
13. Erra, U., De Chiara, R., Scarano, V., Tatafiore, M.: Massive Simulation using GPU of a distributed behavioral model of a flock with obstacle avoidance. In: Vision, Modeling and Visualization, VMV (2004)
14. Erra, U., Frola, B., Scarano, V.: BehaveRT: A GPU-Based Library for Autonomous Characters. In: Boulic, R., Chrysanthou, Y., Komura, T. (eds.) MIG 2010. LNCS, vol. 6459, pp. 194–205. Springer, Heidelberg (2010)
15. Silva, A.R.D., Lages, W.S., Chaimowicz, L.: Boids that see: Using self-occlusion for simulating large groups on gpus. Comput. Entertain. 7, 51:1–51:20 (2010)
16. Reynolds, C.: Interaction with groups of autonomous characters. In: Game Developers Conference, pp. 449–460 (2000)
17. Passos, E.B., Joselli, M., Zamith, M., Clua, E.W.G., Montenegro, A., Conci, A., Feijo, B.: A bidimensional data structure and spatial optimization for supermassive crowd simulation on gpu. Comput. Entertain. 7, 60:1–60:15 (2010)
18. Singh, S., Naik, M., Kapadia, M., Faloutsos, P., Reinman, G.: Watch Out! A Framework for Evaluating Steering Behaviors. In: Egges, A., Kamphuis, A., Overmars, M. (eds.) MIG 2008. LNCS, vol. 5277, pp. 200–209. Springer, Heidelberg (2008)

19. Singh, S., Kapadia, M., Faloutsos, P., Reinman, G.: An Open Framework for Developing, Evaluating, and Sharing Steering Algorithms. In: Egges, A., Geraerts, R., Overmars, M. (eds.) MIG 2009. LNCS, vol. 5884, pp. 158–169. Springer, Heidelberg (2009)
20. Chen, X., Ong, Y.-S., Lim, M.-H., Tan, K.C.: A multi-facet survey on memetic computation. IEEE Trans. Evolutionary Computation, 591–607 (2011)
21. Nguyen, Q.H., Ong, Y.S., Lim, M.H., Krasnogor, N.: Adaptive Cellular Memetic Algorithms. Evolutionary Computation 17, 231–256 (2009)
22. Cao, Q., Lim, M.H., Li, J.H., Ong, Y.S., Ng, W.L.: A context switchable fuzzy inference chip. IEEE Transactions on Fuzzy Systems 14(4), 552–567 (2006)