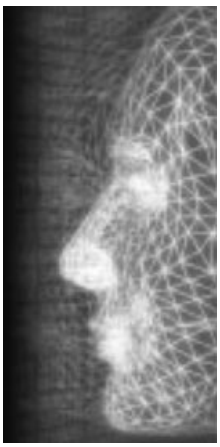
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2008; **19**: 295–305
Published online 11 August 2008 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/cav.261

Managing coherent groups



By Renato Silveira, Edson Prestes and Luciana P. Nedel*

The animation of groups of characters involves the generation of some interesting steering behaviors like following a single path, moving toward a common objective, and moving while keeping a formation. This paper presents a new approach to manage the movement of groups in dynamic environments using a simple and robust algorithm that includes a strategy to keep formations during the displacement of the group. Our method is based on a boundary value problem (BVP) involving Laplace's equation and has two layers. In the first one, a group map is built to allow for local control of each individual, while in the second one a path planning is performed for each group as a whole. Results show that our technique is robust to several situations and can be implemented on GPU, which results in real-time performance for large groups. Copyright © 2008 John Wiley & Sons, Ltd.

Received: 26 June 2008; Accepted: 27 June 2008

KEY WORDS: groups simulation; motion planning; formation-keeping; potential field

Introduction

The simulation of individuals or group of characters moving in synthetic worlds is being an important research topic since the 1970s. Its results can be used in many application fields such as entertainment (e.g., electronic games, film, collaborative virtual environments such as Second Life®, etc.), pedestrian simulation, army training systems and war simulation, evacuation planning for emergency situations, transportation research, crowds simulation, and so on.

Different areas have given different definitions for what a group is.[1] In computer science, groups are frequently defined as a collection of individuals in the same physical environment sharing a common goal.[2] Groups should move in a cohesive way, obeying certain social rules, such as following the same path, walking in the same mean velocity, keeping a regular distance from their neighbors, having a common objective, and keeping a predefined formation. Due to the huge number of elements, the simulation of groups presents yet other challenges, such as efficiency, management of dynamic interactions between characters, complexity and subtlety of behaviors.[3]

*Correspondence to: R. Silveira, Instituto de Informática, Universidade Federal do Rio Grande do Sul Porto Alegre - RS, Brasil. E-mail: rsilveira@inf.ufrgs.br

Formation control has been one of the most important topics in multi-robot system research, where multiple robots are required to move while maintaining formation in a specified geometrical shape. Many complicated tasks, such as save and rescue, transportation, and construction,[4] use formations to achieve their goals. The importance of formations in this context is to allow individual group members to concentrate their sensors across a portion of the environment, while their partners cover the rest. In computer graphics, current techniques to deal with formations are not generic enough to gracefully integrate the problem of group movement with formation-keeping, not allowing the user to interactively define the desired formation shape.

The contributions of this paper are:

- A robust algorithm to control the steering behavior of groups based on a boundary value problem (BVP) path planner. Due to the BVP path planner is based on the equation of Laplace, it is suitable for computation on massively parallel architectures such as GPUs and multi-core CPUs.
- A strategy to handle the group formation-keeping problem, enabling the user to sketch any desirable formation shape. Figure 1 illustrates some group behavior and formation-keeping. This strategy is robust regarding the number of formation shapes, group size, dynamic switching between formations and obstacle avoidance during the motion of characters when

Figure 1. Group in formation, formation-keeping while moving with the proposed method.

they are negotiating space. Furthermore, it is easily integrated with any other path planner.

The remaining of this paper is organized as follows. Next section discusses some related work in group movement and formation. Then, the path planning strategy proposed by us and the basis of the BVP planner are exposed. In a section about group control, the strategy for handling formation control in an integrated way with the BVP planner is detailed. After, we describe how the final movement of the groups is generated. Some interesting results that validate our technique with examples of emergent behaviors are presented. In a discussion section, we made important considerations about performance. Finally, some conclusions and future work summarize our work.

Related Work

The most common approach for simulating group movement for interactive computer graphics is the flocking, introduced by Reynolds.[5] In his classical work, he proposed a boids-model to describe the behavior of entities in a group using only local rules for individuals. Later, he extended his technique to include autonomous reactive behavior and achieve more convincing results.[6] Recently, he[7] implemented a high performance multi-agent simulation and animation on PlayStation 3®. This technique allowed a scalable multi-processor implementation of a large and fast crowd simulation, getting good frame rate with thousands of agents. Also working in games domain, Nieuwenhuisen *et al*.[8] focused on high quality navigation in computer games, and proposed a method based on probabilistic roadmaps that can be effectively applied to games.

Pottinger[9,10] proposed a technique that handles the formation and collision avoidance of a group to implement predefined formations in games. However, this approach produces non-natural behaviors. Berg *et al*.[11] proposed a technique for path planning and interactive navigation of agents in virtual environments in the presence of crowds and moving obstacles, making use of probabilistic roadmaps.

In robotics, researches are being conducted to control groups of robots. Some of them are especially dedicated to formation-keeping as a good way of maintaining the group coherence. Balch and Hybinette[12] proposed the use of social potentials to generate scalable formations for mobile robots. They devise a number of forces to obstacle avoidance and to steer robots, like "attachment sites" that guarantee the formation-keeping.

DOI: 10.1002/cav

	Formation	Coherence	Narrow passages	Performance
Pottinger[10]	Always	No	Natural	Excellent
Balch and Hybinette[12]	Always	No	Not informed	Excellent
Li and Chou[14]	Always	No	Not-natural	Excellent
Kamphuis[15]	No	Always	Not-natural	Excellent
Berg *et al.*[11]	No	No	Not-natural	Good
Our technique	Yes (configurable)	Yes (configurable)	Natural	Good

Table 1. Comparison between the techniques

Schneider and Wildermuth[4] proposed a potential field approach for motion coordination in multiple-robot formations. Each robot feels different *virtual forces* from other robots, obstacles and the shape formation. These forces are combined and used to move each robot to its desired position inside the formation. Working on local sensing, Fredslund and Mataric[13] presented a technique to obtain global behavior based on minimal communication to maintain the global goal. The idea is that each robot follows a designated friend robot through its sensor in a specific orientation.

Li and Chou[14] developed an approach that allows for the re-structuring of dynamic entities through a spherical tree hierarchy. However, their technique does not guarantee the group coherence.

Most of the works developed to this date propose methods to deal with groups and group formation-keeping based on predetermined forms. Kamphuis and Overmars[15] proposed an approach for motion planning of groups of entities where coherence is taken into account. Our method can also handle these problems with the advantage of providing a high-level and centralized group control, integrating the multi-agent path planning and the formation-keeping in real time.

Table 1 shows a qualitative comparison of our technique with the main techniques presented above considering: the possibility of using formations, the generation of natural movements when crossing through a narrow passage, and the performance. Our approach put together characteristics separately observed in the techniques considered, allowing the configuration of some features by the user without the use of "workarounds," as most of the techniques used in games.[9]

Path Planning Strategy

The path planner used by us is based on potential fields without any local minima[16] generated through the numeric solution of the BVPs. Basically, it uses Dirichlet boundary conditions and the following equation

$$\nabla^2 p(\mathbf{r}) + \epsilon \mathbf{v}.\nabla p(\mathbf{r}) = 0 \tag{1}$$

where $\mathbf{v}$ is a bias vector and ϵ is a scalar value.

This method has been recently used by Dapper *et al.*[17] in the computer graphics domain to allow synthetic actors to move negotiating space, avoiding collisions, and attaining goals while producing very individual paths. The individuality of each character can be set by changing its inner field parameters ϵ and $\mathbf{v}$.

The BVP problem can be solved with the discretization of the environment into a fixed homogeneous mesh with identical cells, like an occupancy grid that corresponds to the environment map. Each cell (i, j) is associated to a square region of the real environment and stores a potential value $p_{i,j}$. Dirichlet boundary conditions are such that, cells with high probability of having an obstacle are set to 1 (*high potential*), while cells containing the target are set to 0 (*low potential*). The high potential value prevents the character from running into obstacles whereas the low potential value generates an attraction basin that pulls the character.

Solving the BVP thus consists in interpolating potential values on the grid between obstacles and the target. Considering a grid with unitary cells, this is done using the classical Gauss–Seidel algorithm, which updates potential cells according to the equation

$$\underbrace{p_{i-1,j} + p_{i+1,j} + p_{i,j-1} + p_{i,j+1} - 4p_{i,j}}_{\nabla^2 p(\mathbf{r})} + \underbrace{\epsilon \left(\frac{(p_{i+1,j} - p_{i-1,j})}{2} v_x + \frac{(p_{i,j+1} - p_{i,j-1})}{2} v_y \right)}_{\epsilon \mathbf{v}.\nabla p(\mathbf{r})} = 0 \tag{2}$$

DOI: 10.1002/cav

that leads us directly to the update rule

$$p_{i,j} = \frac{1}{4}(p_{i-1,j} + p_{i+1,j} + p_{i,j-1} + p_{i,j+1}) + \frac{\epsilon}{8}((p_{i+1,j} - p_{i-1,j})v_x + (p_{i,j+1} - p_{i,j-1})v_y) \quad (3)$$

where $\mathbf{v} = (v_x, v_y)$ and $\epsilon \in [-2, +2]$. ϵ must be in this range, otherwise, the boundary conditions that assert the character—repelling obstacles and attracting the target—are violated. Then the method generates oscillatory and unstable behaviors that do not guarantee that the character will reach the target.

A character in the cell $\mathbf{r} = (r_x, r_y)$ uses the gradient descent of this potential defined by

$$\nabla p(\mathbf{r}) = \left(\frac{p_{r_{x+1},r_y} - p_{r_{x-1},r_y}}{2}, \frac{p_{r_x,r_{y+1}} - p_{r_x,r_{y-1}}}{2} \right) \quad (4)$$

to determine the path to follow toward the target position. As the Equation (1) is free from local minimum, the Equation (4) can never be 0. This method is formally complete, that is, if there is a path connecting the character position to the target, it will be found.

Group Control

The main purposes to have a group is to decrease the computational cost through the management of groups of characters instead of handling them independently. However, when it comes to groups, some questions arise. How to define the best group behavior when the group is passing in the middle of obstacles like corridors? How can a group keep the cohesion while crossing through small obstacles? How to guarantee that a set of characters, in formation, will move coherently? Techniques discussed in the Related Work section does not solve this problems in a unified way. Our proposal handles satisfactorily all problems related to the group coherence and group motion with and without a specified formation.

Formation Specification

In order to move the group in a formation-keeping, initially, the user should provide the desired shape for the group formation. This shape can be loaded from a file storing a predefined formation or it can be interactively sketched by the user. The formation shape corresponds to a list $L = \{(x_i^l, y_i^l)\}$, with $|L|$ points in a bi-dimensional space $\Re^2$. As the group has n characters, if $n \leq |L|$ then only n points of this list are needed, since each character is linked to only one point in the formation. In this case, these points are chosen from the first one in the list always at a distance $\lfloor |L|/n \rfloor$ from each other. On the other hand, if $n > |L|$ then we compute a vector $\mathbf{v} = (x_{|L|}^l - x_{|L|-1}^l, y_{|L|}^l - y_{|L|-1}^l)$ using the last two points in the list. Adding up this vector to the last point in the list, we generate new points trying to keep the intention of users drawing, thus preserving the formation topology.

These points sampled from the list L are called *formation points* and comprise a new list $\mathcal{L}$, where $|\mathcal{L}| = n$. These points represent a virtual formation topology and can be thought as attraction points that will pull the characters toward them.

Group Map

Each group is associated to a specific formation and a map, called *group map*, comprised of $\gamma_x \times \gamma_y$ cells. This *group map* is then projected into the environment and its center $(\sum_i^{|\mathcal{L}|} x_i^l/|\mathcal{L}|, \sum_i^{|\mathcal{L}|} y_i^l/|\mathcal{L}|)$ is aligned with the center of the group in the environment, defined by $(\sum_i^{|\mathcal{G}|} x_i^g/|\mathcal{G}|, \sum_i^{|\mathcal{G}|} y_i^g/|\mathcal{G}|)$, where (x_i^g, y_i^g) is the position of the character i belonging to the group $\mathcal{G}$.

Each cell in the *group map* will store a particular potential value. The border of the map and each cell that corresponds to an obstacle in the environment will store a high potential value equals to 1 whereas the others (free cells) will store initially a value between 0 and 1, as commented in the section describing our path planning strategy. We use a global path planner, similarly as used by Dapper *et al*.[17] to provide a collision-free direction. This direction is used to identify cells in the border of the *group map* that will represent an intermediate goal. This goal will attract characters inside the *group map* and the resulting effect is the group motion. It is also needed to allow the characters passage through narrow passages and keep the group coherence.

Looking for obtaining the information about the proximity of a character in relation to the obstacles, we divide the group map into several regions β_i with $i \in \{1, 2, \ldots, max\}$, where each region β_i is comprised of all cells that are i cells far from nearest obstacle. Each cell in β_i region has a scalar value μ_{β_i} associated that is used to weight the distance between a character and an obstacle. We must guarantee that the scalar value associated with each region increases insofar cells are

DOI: 10.1002/cav

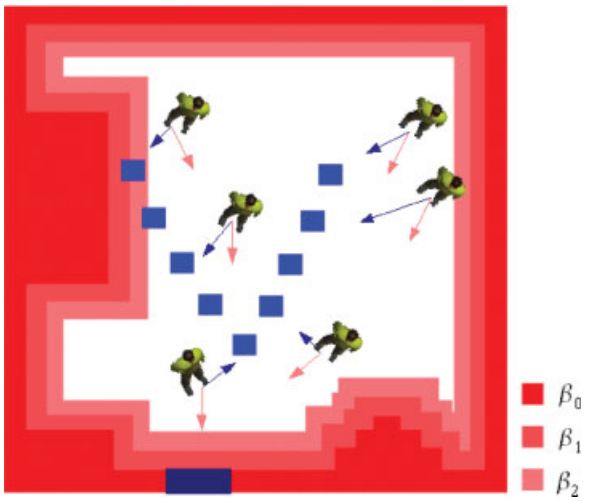


Figure 2. Group map. Characters inside this map are under the influence of two forces. The formation force (dark arrows) and the grid force (light colored arrow). β_0 is the group map border and is an obstacle zone used as the BVP boundary condition. β_1 and β_2 are the zones where the grid force has higher influence than the formation force.

approaching obstacles. In other words, $\mu_{\beta_1} = 1$, and $\mu_{\beta_1} > \mu_{\beta_2} > \cdots > \mu_{\beta_n}$. This can be done in the relaxation step, when β_i regions are created, as shown in Figure 2. When the character is in a cell associated to any of these regions ($\beta_1, \beta_2, \ldots, \beta_{\max}$), it will be influenced not only by the force exerted by the formation but also by the gradient descent computed at its position in the *group map*. The influence of the gradient associated to region β_i will be weaker insofar as $i \to max$, where *max* is the maximum distance that a character can be from the nearest obstacle, in general, $max = \lfloor \max(\gamma_x, \gamma_y)/2 \rfloor$. An illustration of the *group map* can be seen in Figure 2.

Each character is mapped to the *group map* as an obstacle in order to avoid collisions with other characters, that is, the cell associated with this character stores the potential value equals to 1. At the same time, its neighbor cells are set as a cell on β_1 region. Then, in the relaxation step, the Gauss–Seidel algorithm is employed to compute the potential of the free cells in the *group map*. With the potential value of each cell, the vector field is extracted using Equation (4) and the character motion is then influenced by two forces: formation force, and the *group map* vector field. As both forces influence in the path definition, they should be properly established, in order to avoid undesired or non-realistic behaviors.

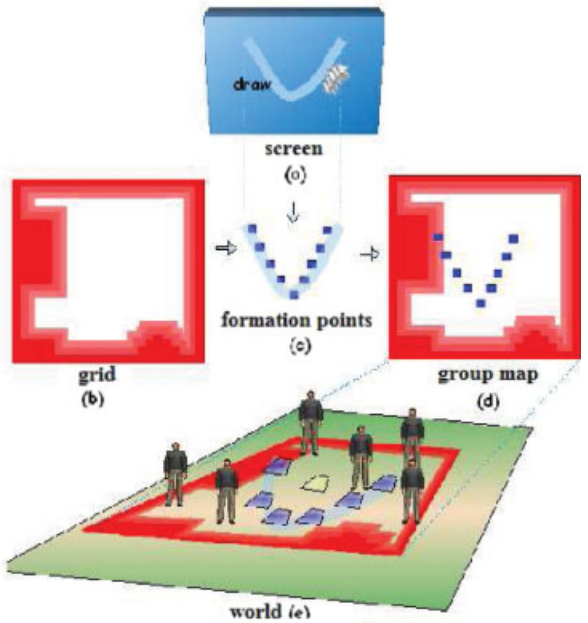


Figure 3. Motion generation: the user draws a formation in the screen (a); it is processed by generating the formation points (c); a grid is created (b); and characters move over the group map according to the resulting forces (d,e).

Motion Generation

To move the group, we use the following two steps algorithm. The first step handles the individual character movement, while the second one is responsible by moving the whole group, through the displacement of the group map. In the first step, the user draws any formation shape (Figures 3a and 3c) that is processed and combined with the *group map* (Figures 3b and 3d). All characters in the group are inside this map, and move according to the resulting forces experienced by them. The second step plans a collision-free path that will be followed by the *group map* (Figure 3e). Any path planner can be used in the second step. We have chosen to use the path planner based on sketch similar to that proposed by Dietrich *et al*.[18]

Moving Individuals

As seen in Figure 2, each character of a group is inside its respective *group map*, and sense both the force that tries to keep the character in formation and the force exerted by the vector field produced from the *group map* (gradient descent). At each step, each character combines

DOI: 10.1002/cav

these forces into a resulting force, as described below, to compute a collision-free path that tries whenever possible to keep it into its formation position.

By projecting the formation points into the *group map*, the force $\mathbf{z}$ exerted by the formation can be calculated by $\mathbf{z}_k = (x_k^l, y_k^l) - (x_k, y_k)$, where (x_k^l, y_k^l) is the position of the formation point associated to the character k.

The following equation then describe the update of the position r of character k at time t:

$$\Delta\mathbf{r} = \upsilon_{\max}\left[\underbrace{\mu_{\beta i}\mathcal{E}}_{\text{environment}} + \underbrace{(1-\mu_{\beta i})\alpha\mathcal{F}}_{\text{formation}}\right]$$

where

$$\mathcal{E} = \Psi(|\varphi^{t-1} - \zeta^t|)(\cos(\varphi^t), \sin(\varphi^t))$$

is the environment contribution and

$$\mathcal{F} = \Psi(|\omega^t - \omega^{t-1}|)(\cos(\omega^t), \sin(\omega^t))$$

is the formation contribution and

$$\alpha = \min(\sqrt{(x_k^l - x_k)^2, (y_k^l - y_k)^2}, 1)$$

φ^t is the character orientation in the instant t, defined by

$$\varphi^t = \eta\varphi^{t-1} + (1-\eta)\zeta^t$$

where $\eta \in [0, 1)$; ζ^t is the gradient descent orientation in the current character position in the instant t; ω^t is the orientation of the force $\mathbf{z}$; $\Psi : \Re \to \Re$ is

$$\Psi(x) = \begin{cases} 0, & \text{if } x > \pi/2 \\ \cos(x), & \text{otherwise} \end{cases}$$

(x_k^l, y_k^l) is the position of the formation point associated with the character; (x_k, y_k) is the character current position. $\upsilon_{\max}$ must be greater than the *group map* velocity to ensure that the character always stay inside the *group map*. The scalar value μ_{β_i} is associated to each cell in the region β_i. If we reduce the equation above to

$$\Delta\mathbf{r} = \upsilon_{\max}\mathcal{E}$$

the group will move without a defined formation.

When $\eta = 0$, the character adjust his orientation using only the environment gradient descent. If $\eta = 0.5$, the previous character orientation φ^{t-1} and the gradient descent ζ^t influence equally the computation of the new character direction. The parameter η can be viewed as an inertial factor that tends to keep constant the character direction insofar $\eta \to 1$. When $\eta \to 1$, the character reacts slowly to unexpected events, increasing its hitting probability with obstacles. However, if $|\varphi^{t-1} - \zeta^t|$ was greater than $\pi/2$ then there is a probability of collision and the function Ψ returns 0. This means that the character stops avoiding collision. Otherwise, the velocity of the character changes in proportion to collision risk, ensuring the absence of collision.

Every time a character moves, we just restore the potential value of cells previously occupied and set the cells currently occupied by characters to obstacles. We do not need to relax the entire *group map* from the start. Few relaxation steps are enough to produce a smooth vector field. This saves time and increases system performances.

Moving Groups

The movement of an entire group is produced by handling the motion of the *group map*. For this, we consider the *group map* center point as an individual and any path planner algorithm can be used to obtain a free-collision path. As mentioned before, in this paper we are using a path planner based on sketch.

In each move step, the *group map* slides over the world, a new position for the center of the *group map* is calculated, and part of the world information is projected into this map. Obstacles and goals are mapped as proposed in our Path Planning Strategy, characters are also mapped as obstacles into the map. After, the relaxation step is performed by interpolating the potential values used to control the group. Formation points are rotated in order to align its orientation with the direction provided by the path planner, that is, formation points are oriented with the *group map* movement direction.

As the global path planner gives the next *group map* position, this position can be used to calculate the rotation of the formation points. Considering Δs the distance from the current *group map* position to the next position, and v_g the *group map* velocity, the time spent to reach the next position is

$$\Delta t = \frac{\Delta s}{v_g}$$

DOI: 10.1002/cav

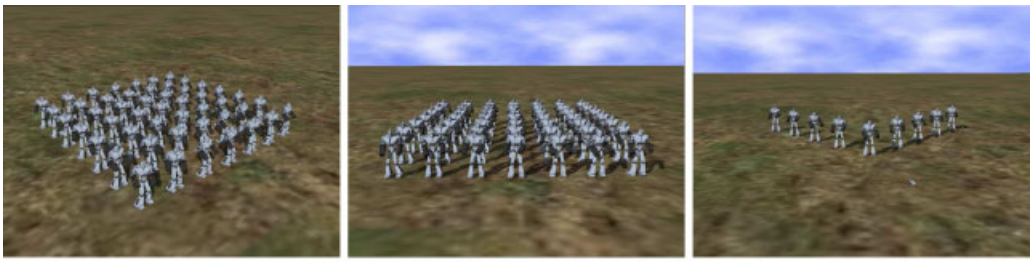


Figure 4. Predefined formations. Diamond formation (a), square formation (b) and "V" formation (c).

As we know that the final orientation obtained from the global path planner is φ^t, the angle increment must be

$$\vartheta = \frac{\varphi^t v_g}{\Delta t}$$

The formation points (x_i^l, y_i^l) can be updated during the *group map* motion at time t by

$$\begin{bmatrix} x_i^{l,t} \\ y_i^{l,t} \end{bmatrix} = \begin{bmatrix} \cos(\vartheta) & \sin(\vartheta) \\ -\sin(\vartheta) & \cos(\vartheta) \end{bmatrix} \begin{bmatrix} x_i^{l,t-1} \\ y_i^{l,t-1} \end{bmatrix} \quad (5)$$

The formation points are then re-projected into the *group map*, exerting an attractive force over the group member.

Emergent Behaviors

In this section, we present some experimental results obtained from the simulation of typical situations involving groups. This allows the reader to evaluate the quality of the animations generated by our strategy to control the movement of group.

Sketching Formations

In the Formation Specification section, we presented a way to deal with the problem of specifying the formation. The user is free to sketch any desirable formation that the members of a given group will try to keep. The formation can be dynamically modified while the group moves according to the user intentions. Figure 4 shows a group of characters in a predefined formation.

Using the same facility, we can easily resample the number of characters in a formation when, for example, one character leaves or joins the group, depending on the desire of the application designer. In the simulation of an army, as the ones in Figure 4, if few soldiers die or simply leave the group, a realistic solution could be to keep their places in the formation opened. However, if many soldiers disappear, the formation can be resampled, considering fewer positions.

Squeezing, Stretching, and Splitting Groups

Generally, techniques based on potential fields present a non-smooth behavior when the characters cross narrow passages. Our technique generates a smooth path when passing through corridors, as shown in Figure 5. Note that the formation shape is preserved whenever possible.

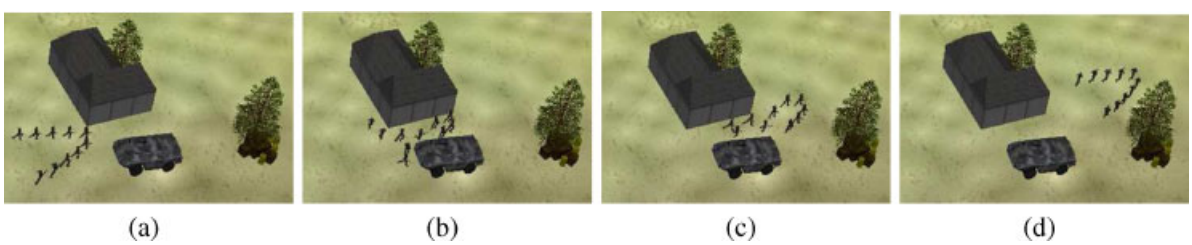


Figure 5. A group in formation (a) needs to cross a narrow passage (b); the formation is deformed to allow the group to cross it (c); and when the area is again large (d), go back to the original formation.

DOI: 10.1002/cav

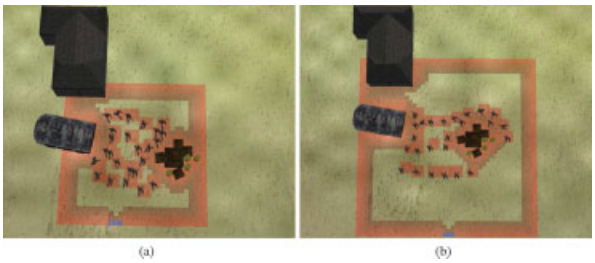


Figure 6. Using a 20×20 *group map, the group passes through an obstacle together (a); but with a larger map* (30×30)*, the same group splits (b).*

In an environment with plenty of obstacles, the group can split or not when crossing dense areas. When the group map is large, the obstacles are entirely represented inside the *group map* and are automatically avoided, splitting the group. As the splitting emergent behavior depends on the direct relation among the size of the *group map* and the size of the obstacles, we can control it adjusting adequately the map size, as we can see in Figure 6.

Other interesting example is when a group is walking in a forest. In this case, some bushes could be avoided by splitting the group, but when passing between two large rocks, the group must stay together. However, while the group is in formation-keeping, it is desired that this formation is kept when passing through obstacles. Figure 7 shows a group passing through an environment with many obstacles. We can see that group keeps the formation while avoiding collisions with other characters and obstacles.

Dealing with many groups, when collisions between groups occur, the character stops and waits until it is able to continue. This behavior is due to the environment contribution term in the Equation (5). Figure 8 shows this situation: two groups passing through each others and avoiding collision without keeping a formation.

In the next section, performance and the gain obtained with a GPU implementation is measured and discussed.

Discussion about Performance

Our technique bottleneck is the relaxation step, that is, the evaluation of Equation (3). In the second step

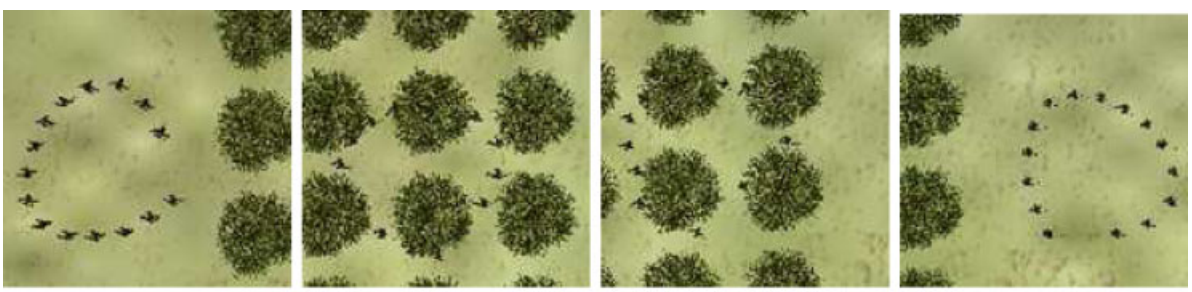


Figure 7. Group passing by many obstacles, keeping the formation whenever possible.

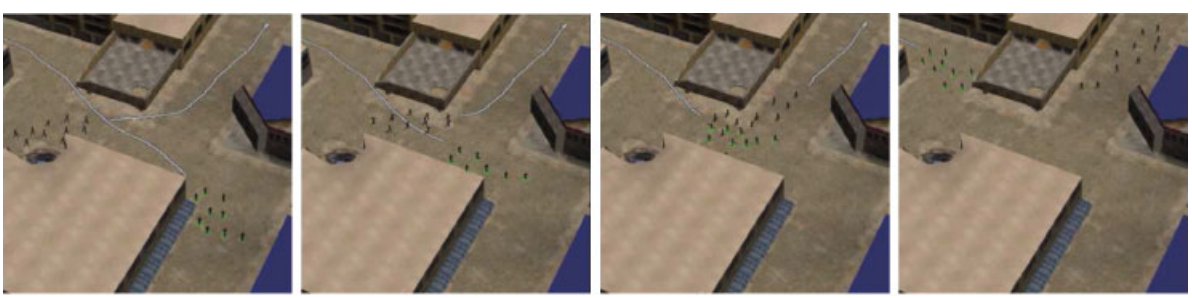


Figure 8. Two groups passing through each other without keeping the formation. The white curve indicates the path that should be followed.

DOI: 10.1002/cav

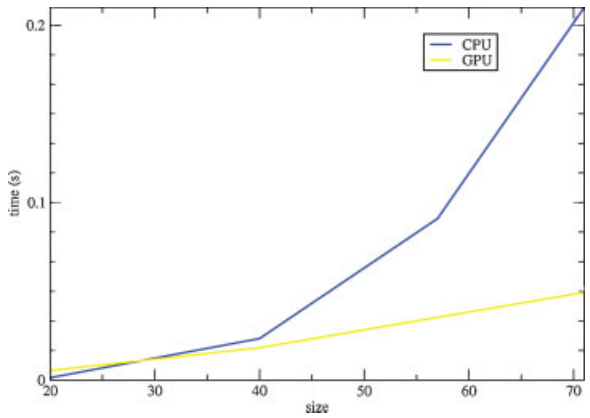


Figure 9. Comparison between CPU and GPU implementation. The light colored line shows Equation (3) linear scalability. The grid map is relaxed allowing a maximum error with magnitude equal to 10^{-3}*. The GPU time collected include the overhead to transfer data between GPU and CPU.*

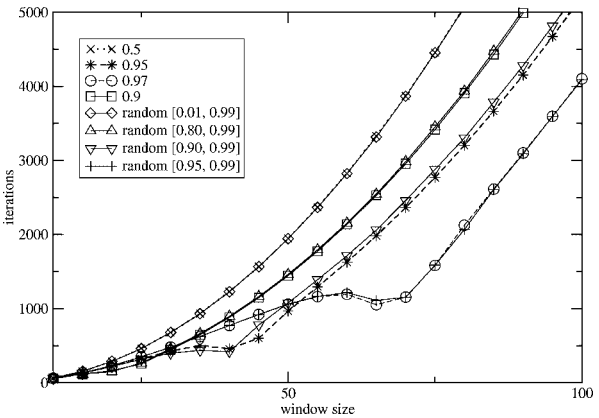


Figure 10. Comparison of the relaxation on a map with cells initialized with all potential values equals and constants, and maps initialized with random potential values.

of the movement algorithm, after each *group map* move step, we must relax the *group map*. Looking for optimization, we implemented the relaxation step both on CPU and GPU, taking advantage of its parallel nature. The relaxation step was implemented using Gauss–Seidel algorithm in the GPU and compared with the CPU solution. Figure 9a illustrates results obtained on an ATHLON 2.2 GHz with 2 GB of memory and a GeForce 8800 GTX graphics card with 768 MB of memory. Results were generated using the worst case, that is, the relaxation is done until the entire convergence.

In order to decrease the number of iterations needed for the *group map* relaxation, we analyzed some different ways to update the cells. The relaxation step consist in initializing the *group map* with a potential value equal to 1 in border and obstacle cells, potential values equal to 0 in goal cells, and potential values intermediaries between 0 and 1 in free cells.

During the relaxation step, Equation (3) is evaluated several times for each cell of the *group map*, until the convergence. After convergence, we can see that the potential values of the cells do not vary linearly with their distance to the goal. We notice that there are a lot of cells with values close to 1. This implies the choice of an initial value around 1 to the free cells is probably a good choice. Some experiments have been performed to obtain the best initial potential values that will speed up the convergence.

Figure 10 shows the number of iterations needed for the convergence of potential of the free-space cells, considering that these cells store previously constant values equal to 0.5, 0.9, 0.95, and 0.97 and random values in [0.001, 0.99], [0.8, 0.99], [0.90, 0.99], and [0.95, 0.99]. We can see that for small maps (up to 25×25 cells), to set the free cells with values close to 0.9 is a good choice, and accelerates the convergence. For large maps (greater than 50×50 cells), to set the free cells with values near to 1 accelerates the convergence in a significant way. We can also observe that the initialization of the free cells potential with random values instead of scalar values does not generate significant benefits, even when random numbers belong to a well-defined range.

Hence, the closer the potential value of each cell is from the expected ones after the potential convergence, the faster will be the relaxation process. We observed that the group map changes very slightly between successive steps. Hence, the convergence of the potential is accelerated due to the values of potential in most of cells is already close to expected values for after convergence with this new configuration. Figure 11 shows a situation where the potential of the previous *group map* configuration is kept. Initially, 116 iterations have been spent for the convergence on a map with $20\times$ 20 cells, considering an error of 10^{-3}. Keeping the *group map* potential configuration, only 86 iterations for the relaxation convergence have been spent, a reduction of 26% in this case.

DOI: 10.1002/cav

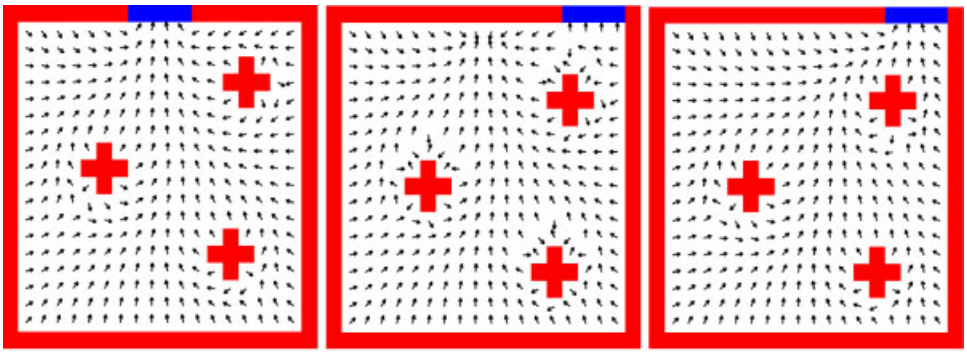


Figure 11. (a) Initial map configuration at time t. (b) At time $t + 1$, the map moves in the (0, 1) direction, producing a shift in the obstacle positions. (c) The new relaxation process needs less iterations.

Conclusions and Future Work

We presented a technique to interact with groups of characters, controlling the group movement based on the BVP problem, which is very well suited to computation on massively parallel architectures, as multi-core CPU and GPU. The proposed technique handles the group formation-keeping problem in higher-level, enabling the user to draw any desirable formation shape. Differently from most of the current techniques, we are not constrained to predefined shapes, and we can handle the problems emerged in group behavior with a unique technique.

The method has proven itself to be robust in the number of formation shapes, group size, dynamic switching between formations, and in obstacles avoidance while characters move negotiating space. With this technique, groups composed of hundreds of characters can be used in real-time applications. Besides the BVP planner, our method can also be used with any path planner.

As future work, we intend to test the use of efficient data structures in order to speed up the method and its implementation in GPU. We are also evaluating the method extension to plan 3D motion in an efficient way. This could be useful to plan fish trajectories while swimming or birds flying.

ACKNOWLEDGEMENTS

We thank the Computer Graphics group of UFRGS for their valuable help in this work. We are also grateful to Brazilian Council for Research and Development (CNPq), Brazilian Education Ministry (MEC-CAPES), and the State of Rio Grande do Sul Research Foundation (FAPERGS) for the financial support. Microsoft Brazil also provided additional support.

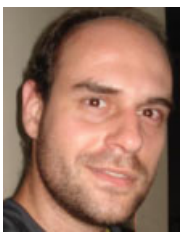
References

1. Musse SR, Ulicny B, Aubel A, Thalmann D. Groups and crowd simulation. In *ACM SIGGRAPH 2005 Courses*, ACM Press, New York, NY, USA, 2005; 2.
2. Musse SR, Thalmann D. Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics* 2001; **7**(2): 152–164.
3. Treuille A, Cooper S, Popović Z. Continuum crowds. In *SIGGRAPH'06: ACM SIGGRAPH 2006 Papers*, ACM, New York, NY, USA, 2006; 1160–1168.
4. Song P, Kumar V. A potential field based approach to multi-robot manipulation. In *IEEE International Conference on Robotics and Automation*, 2002; 1217–1222.
5. Reynolds C. Flocks, herds and schools: a distributed behavioral model. In *SIGGRAPH'87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, 1987; 25–34.
6. Reynolds C. Steering behaviors for autonomous characters. In *Game Developers Conference*, Miller Freeman Game Group, San Francisco, CA, USA, 1999; 763–782.
7. Reynolds C. Big fast crowds on ps3. In *Sandbox'06: Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames*, ACM, New York, NY, USA, 2006; 113–121.
8. Nieuwenhuisen D, Kamphuis A, Overmars MH. High quality navigation in computer games. *Science of Computer Programming* 2007; **67**(1): 91–104.
9. Pottinger D. Implementing coordinated movement. *Game Developer*, 1999; 48–58.
10. Pottinger D. Coordinated unit movement. *Game Developer*, 1999; 42–51.
11. Berg J, Patil S, Sewall J, Manocha D, Lin M. Interactive navigation of multiple agents in crowded environments. In *SI3D'08: Proceedings of the Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, 2008; 139–147.

DOI: 10.1002/cav

12. Balch T, Hybinette M. Social potentials for scalable multirobot formations. In *IEEE International Conference on Robotics and Automation (ICRA-2000)*, San Francisco, 2000.
13. Fredslund J, Mataric M. A general algorithm for robot formations using local sensing and minimal communications. *IEEE Transactions on Robotics and Automation* 2002; **18**(5): 837–846.
14. Li T, Chou H. Motion planning for a crowd of robots. In *International Conference on Robotics and Automation (ICRA)*, IEEE Press, San Diego, CA, 2003.
15. Kamphuis A, Overmars MH. Finding paths for coherent groups using clearance. In *SCA'04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2004; 19–28.
16. Trevisan M, Idiart MAP, Prestes E, Engel PM. Exploratory navigation based on dynamic boundary value problems. *Journal of Intelligent and Robotic Systems* 2006; **45**: 101–114.
17. Dapper F, Prestes E, Nedel LP. Generating steering behaviors for virtual humanoids using BDP control. In *Proceedings of Computer Graphics International (CGI)*, 2007; 105–114.
18. Dietrich CA, Nedel LP, Comba JLC. A sketch-based interface to real-time strategy games based on a cellular automaton. In *Game Programming Gems*, Charles River Media, 2008; 59–67.

Authors' biographies:



Renato Silveira received his BSc degree in Computer Science from the Federal University of Lavras (UFLA), Brazil, in 2005 and MSc from the Federal University of Rio Grande do Sul (UFRGS), Brazil, in 2008. Currently, he is a first year PhD student from Federal University of Rio Grande do Sul (UFRGS). His main interests include: computer animation, path-planning, motion planning and games.

Edson Prestes received his BSc degree in Computer Science from the Federal University of Pará (UFPa), Brazil, in 1996 and MSc and PhD in CS from the Federal University of Rio Grande do Sul (UFRGS), Brazil, in 1999 and 2003, respectively. Nowdays, he is a lecturer at UFRGS since 2005. His main field is artificial intelligence applied to mobile robots and perceptual user interfaces.

Luciana P. Nedel works at Federal University of Rio Grande do Sul (UFRGS), where she has an assistant professor position since 2002 and makes research in the Computer Graphics and Virtual Reality Group, at Informatics Institute. She received the Dr. degree in Computer Science from Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland under the supervision of Prof. Daniel Thalmann in 1998. Her main interests include: interactive visualization, virtual humans, computer animation, path-planning, tiled displays, and non-conventional interaction techniques and devices. In these fields, she published more than 60 scientific articles in journals, conference Proceedings and edited books.

DOI: 10.1002/cav