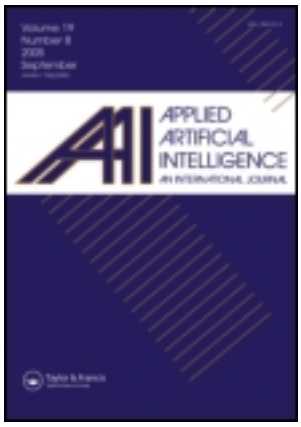


Applied Artificial Intelligence
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/uaai20

AN INTERACTIVE MODEL FOR STEERING BEHAVIORS OF GROUPS OF CHARACTERS

Rafael Araújo Rodrigues [a], Alessandro de Lima Bicho [d], Marcelo Paravisi [a], Cláudio Rosito Jung [b], Léo Pini Magalhães [c] & Soraia Raupp Musse [a]

[a] Graduate Programme in Computer Science, PUCRS, Porto Alegre/RS, Brazil

[b] Institute of Informatics, UFRGS, Porto Alegre/RS, Brazil

[c] School of Electrical and Computer Engineering, UNICAMP, Campinas/SP, Brazil

[d] Center of Computational Sciences, FURG, Rio Grande/RS, Brazil

Available online: 07 Jul 2010

To cite this article: Rafael Araújo Rodrigues, Alessandro de Lima Bicho, Marcelo Paravisi, Cláudio Rosito Jung, Léo Pini Magalhães & Soraia Raupp Musse (2010): AN INTERACTIVE MODEL FOR STEERING BEHAVIORS OF GROUPS OF CHARACTERS, Applied Artificial Intelligence, 24:6, 594-616

To link to this article: http://dx.doi.org/10.1080/08839514.2010.492167

Applied Artificial Intelligence, 24:594–616

ISSN: 0883-9514 print/1087-6545 online
DOI: 10.1080/08839514.2010.492167

Taylor & Francis
Taylor & Francis Group

AN INTERACTIVE MODEL FOR STEERING BEHAVIORS OF GROUPS OF CHARACTERS

Rafael Araújo Rodrigues[1], Alessandro de Lima Bicho[4], Marcelo Paravisi[1], Cláudio Rosito Jung[2], Léo Pini Magalhães[3], and Soraia Raupp Musse[1]

[1] *Graduate Programme in Computer Science, PUCRS, Porto Alegre/RS, Brazil*
[2] *Institute of Informatics, UFRGS, Porto Alegre/RS, Brazil*
[3] *School of Electrical and Computer Engineering, UNICAMP, Campinas/SP, Brazil*
[4] *Center of Computational Sciences, FURG, Rio Grande/RS, Brazil*

□ *This article presents an approach for generating steering behaviors of groups of characters based on the space colonization algorithm that has been used in the past for generating leaf venation patterns and tree structures. In this article, the underlying idea of the space colonization algorithm is adapted to control the motion of virtual characters, providing robust and realistic group behaviors by adjusting just a few parameters. The main contributions of this work are the robustness, flexibility, and simplicity of the proposed approach to control groups of characters in an interactive way, providing path planning and a series of group behaviors, such as group formation, alignment among others. We also introduce a possible extension of this model to provide collision avoidance among agents, mainly focused on crowd simulation. In addition, an interactive tool is provided to allow an easy manner for controlling the motion of virtual characters.*

INTRODUCTION

Intelligent virtual agents (IVAs) are virtual characters that try to mimic several characteristics of real humans, such as interaction with other agents and the environment, and that present some level of autonomy. Within the context of IVAs, the development of easy and interactive manners to control the motion of virtual objects is an important open problem that present application in different areas, such as computer graphics and robotics. In particular, steering behaviors of groups of characters is important in games, movie productions, simulation tools, among others.

This work was developed in collaboration with HP Brazil R&D and Brazilian research agencies CNPq, FINEP, and CAPES.

Address correspondence to Soraia Raupp Musse, Graduate Programme in Computer Science, PUCRS, Av. Ipiranga, 6681, Building 32, Porto Alegre/RS, Brazil. E-mail: soraia.musse@pucrs.br

Indeed, this technology can be applied in any situation in which mobile entities can be simulated.

Many models have been proposed in the past years, aiming to provide different ways to steer behaviors of groups, as discussed in Section 2. However, in spite of all existing methods, the current state-of-the-art lacks of flexibility. This article presents an approach for steering groups of virtual agents that can be used to generate different types of behaviors with simple changes in the proposed model. In fact, the main advantages of proposed model are (i) robustness, allowing the simulation of realistic behaviors in low and medium density of virtual agents; (ii) simplicity, since the just a few parameters are required; and (iii) flexibility to simulate a series of different behaviors, ranging from path planning to group formations. In addition, this article presents an interactive tool to create and control characters motion and behaviors in real time that could be used to control groups of any type of entities (fishes, birds, humans, etc.) in virtual spaces.

The proposed approach is an extension of our work published recently (Rodrigues et al. 2009), that was inspired in a biological algorithm, based on competition for space in a coherent growth of veins and branches (Sachs 1969). We adapted this idea to generate motion behavior of groups, which also compete for space to move realistically, avoiding collisions, as described in Section 4. It should be noticed that the main scope of this article is focused on individuals motion by proposing tree paths and group behaviors. The main point in Rodrigues et al. (2009) is that groups of individuals can only be recognized if density of people is not high, because group structures are not visible in highly dense crowds. This article extends the previous work in Rodrigues et al. (2009) by proposing a free-of-collision motion of crowds of people. In addition, an interactive tool to facilitate characters control is described, and new experimental results are presented.

The remainder of this article is organized as follows. In the next section we discuss some works found in literature, whereas in Section 3 the space colonization algorithm is described. In Section 4 we describe our model to provide characters motion and in Section 5 the steering behaviors of groups. Section 6 presents an extension of the original conference article (Rodrigues et al. 2009) to crowds, as well as an interactive tool to create the virtual agents and environments. Section 7 discusses some obtained results, whereas Section 8 draws final considerations.

RELATED WORK

The simulation and control of virtual groups have been studied since the early days of behavioral animation. Two seminal articles are related to models based on agents, which have some level of autonomy and

individuality. Reynolds (1987) simulated flocks of bird-like entities called *boids*, obtaining realistic animation by using only simple local rules. Tu and Terzopoulos (1994) created groups of artificial fishes endowed with synthetic vision and perception of the environment, which control their behavior.

A specific problem related to path planning is the generation of realistic motion along the path. LaValle (1998) introduced the concept of a rapidly exploring random tree (RRT) as a randomized data structure for path planning problems. An RRT is iteratively expanded by applying control inputs that drive the system slightly toward randomly-selected points. Choi et al. (2003) proposed a model based on a probabilistic path planning and hierarchical displacement mapping to generate a sequence of realistic movements of a human-like biped figure to move from a given start position to a goal with a set of prescribed motion clips. Metoyer and Hodgins (2004) proposed a method for generating reactive path following based on the user's examples of the desired behavior. Dapper et al. (2007) proposed a path planning model based on a numerical solution for boundary value problems (BVP) and field potential formalism to produce steering behaviors for virtual humans. Rodríguez et al. (2007) proposed a heuristic approach to planning in an environment with moving obstacles using dynamic global roadmap and kinodynamic local planning. Kallmann and Mataric (2004) proposed dynamic roadmaps for online motion planning in changing environments. When changes are detected in the workspace, the validity state of affected edges and nodes of a precomputed roadmap are updated accordingly.

More specifically concerning the motion of groups, Kamphuis and Overmars (2004) introduced a two-phase approach, where a path for a single agent (a backbone path) is generated by any motion planner. Next, a corridor is defined around the backbone path and all agents stay in this corridor. Rodríguez et al. (2007) proposed a model using a roadmap providing an abstract representation of global environment information to achieve different complex group behaviors that cannot be modeled with local information alone. Lien et al. (2005) proposed ways using roadmaps to simulate a type of flocking behavior called shepherding behavior in which outside agents guide or control members of a flock. Data-driven models are quite recent in comparison with other methods and aim to record motion in a preproduction stage or to use information from real life to calibrate the simulation algorithms. One example is the work proposed by Lien et al. (2005), describing a model for controlling groups motion based on automatic tracking algorithms.

Despite the existence of several algorithms for path planning and steering of groups, interfaces are usually not interactive, and sometimes even the parameters are not easy to be defined. This article proposes a

new model for steering behaviors, where group behaviors can be easily calibrated, through an interactive interface, while keeping diversity of generated results. Considerations about the methods proposed here are presented in the next sections.

BIOLOGICALLY INSPIRED ALGORITHM

The basic model for agents navigation is based on the space colonization algorithm, which has been previously used to develop leaf venation patterns (Runions et al. 2005) and tree structures (Runions et al. 2007). The venation model simulates three processes within an iterative loop: leaf blade growth, the placement of markers in the free space, and the addition of new veins. The markers correspond to sources of the plant hormone auxin, which, according to a biological hypothesis, emerge in the growing leaf regions not penetrated by veins. The markers that are approached by the advancing veins are gradually removed, because the space around them is no longer free. As the leaf grows, markers in the free space are added in the space between existing veins and markers. This process is detailed previously in Runions et al. (2005). Here, the venation model has been adapted to animate groups of characters (Rodrigues et al. 2009). Indeed, the key idea is to represent the space in an explicit way, using a set of markers (dots in the space). The markers define the "walkable space" through a discrete set of points, which are used to compute the paths of agents. These markers should be randomly distributed (according to a uniform probability density function) over the portions of the space that can be effectively occupied by the virtual agents, meaning that obstacles and other regions where agents should not move must not be filled with markers. Figure 1(a) illustrates an environment populated with markers. The markers allow the organization and facilitates the steering of group behaviors, as discussed in next sections.

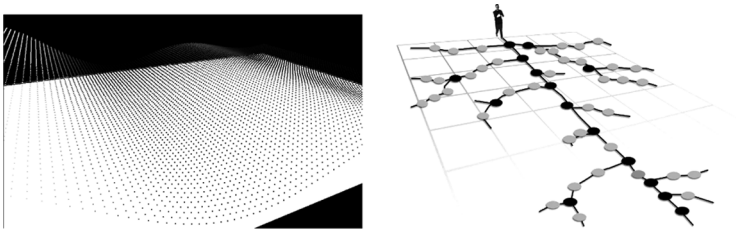


FIGURE 1 (a) Markers (dots) are discrete representation of walkable space; (b) Darker dots describe key positions, lighter dots are related with each simulated step that generates path nodes, and between two path nodes there is the path segment.

MODEL FOR CHARACTERS MOTION

This section describes our model to provide the motion of the virtual characters.[1] Our model based on space colonization algorithm can be used to provide path planning (from initial position to a fixed or mobile goal) as well as to direct motion of virtual agents without specific goals. The main difference between these two cases is that motion planning generates one or more coherent paths (called *tree paths* in this work) to reach a specific position, whereas the direct motion takes into account desired directions. So, in the last possibility, local minima are allowed, because a global path is never planned. The main application of direct motion is in situations where a specific goal is not possible, like group behavior for alignment, for instance.

Following there are the definitions of our model:

- I denote an agent in the group, having a position $\boldsymbol{p}(t)$ at each iteration t.[2]
- If an agent have objectives, these positions at each time are denoted by $\boldsymbol{g}(t)$.
- If an agent does not have specific goals, it should have a direct motion $\boldsymbol{dm}$, like in a desired group alignment.
- There is a *personal space* for each agent, modeled as a circular region (with radius R), that represents a "perception field" that limits the range of markers which can be used by each agent.
- G denote the tree path for an agent, computed by using the venation model proposed by Runions et al. (2005) and adapted to group animation.

A tree path is a set of locations (key positions) organized in a directional graph. Each step created in the path corresponds to a *path node*, whereas a path segment joins two path nodes. In our model, paths from the current position of a given agent to its goal are described through a tree, as illustrated in Fig. 1(b). In this figure, the goal is illustrated through a red flag, darker dots describe key positions where bifurcations occur, and lighter dots are related with each simulated step that generates path nodes. Between two path nodes there is a path segment, and all segments have the same length. Drawing all segments we can see the tree path generated from one agent to a specific goal.

Generating Tree Paths for Agents

The tree path G is computed through a two-stage algorithm within an iterative loop: (i) markers processing and (ii) the addition of new path nodes. The markers in Runions's (2005) model correspond to sources

of the plant hormone auxin, which emerge in the growing leaf regions not penetrated by veins, according to a biological hypothesis. For path planning markers describe the "walkable space" to define the direction of a path node. During each iteration, a path node is influenced by all the markers closer to it than any other path node. Figure 2 illustrates the process of tree paths creation and evolution from agent current position to a specific goal.

For a path node n, located at the position $\boldsymbol{n}$, the set of markers (located at positions $\boldsymbol{m}_i$) that are closer to n than any other node is denoted by

$$S(n) = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_N\}, \tag{1}$$

where N is the number of markers associated with node n. If $S(n)$ is not empty, a new path node n' can be created and attached to n through an edge representing a path segment. Each path node has a circular action region with radius A_r,[3] that limits the markers that must be evaluated (if they are not closer to any other path node) to compute the new direction of growth of the tree path. Figure 1(b) shows an example of a tree path computed from an agent to its goal. To provide a diversity of branch orientations generated for G, increasing the space occupation, the markers closer to each segment are allocated to it, and they cannot be used by other path segments that originate from the same agent I. On the other hand, the tree path G^* associated to another agent I^* can use the same markers of the tree path G. Consequently, tree paths from different agents can intercept each other. However, this fact could bring a collision situation, which is not desirable. To deal with collision-free behaviors, we used the method for minimum distance enforcement among agents, proposed by Treuille et al. (2006) and detailed in Section 4.2.

Mathematically, the algorithm for building the tree path is described as follows. Given a tree node n and a nonempty set $S(n)$, a decision must be made whether a child node n' related to n will be created or not. This decision is made in such a way that nodes closer to the goal are more likely to have children, achieving a wider variety of paths in the vicinity of the goal.

In the proposed approach, the node n_g that is the closest to the goal will certainly have a child, guaranteeing that the goal will be reached by at least one trajectory. The probability $P(n)$ of any other node n having a child is given by

$$P(n) = \frac{\|\boldsymbol{n}_g - \boldsymbol{g}\|}{\|\boldsymbol{n} - \boldsymbol{g}\|}, \tag{2}$$

where $\|\cdot\|$ is the L^2 norm of a vector. To decide whether a child will be created for n, a random variable ξ with uniform distribution in the interval

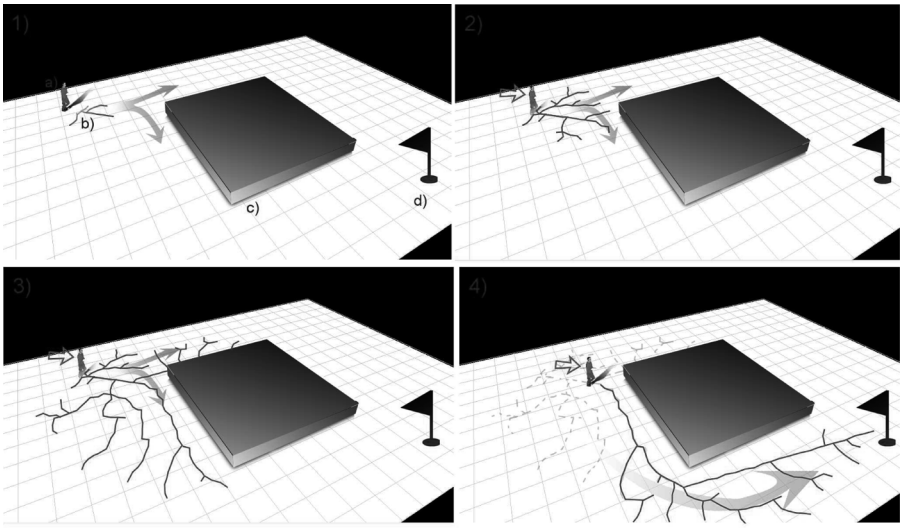


FIGURE 2 In 1) agent a) in position b) goes to its goal d) avoiding collision with obstacle c). In 2) and 3) we show the evolution of tree paths growing toward to the goal avoiding collision with obstacle. In 4) the goal is reached by the tree path while agent walks into the path. It is possible to see branches of tree path that have been deleted from the tree.

$[0, 1]$ is generated, and the the child is created if

$$P(n) \geq \xi. \tag{3}$$

It can be observed that $P(n) = 1$ when $n = n_g$, which guarantees that n_g will certainly have a child, regardless of the random value selected for ξ. Also, it is easy to verify that $P(n)$ is monotonically decreasing with respect to the distance between n and the goal g, so that child nodes that are closer to the goal tend to have more children. If a given node n is granted a child n', it will be created at a position $\boldsymbol{n}'$ through

$$\boldsymbol{n}' = \boldsymbol{n} + \alpha \frac{\boldsymbol{d}(n)}{\|\boldsymbol{d}(n)\|}, \tag{4}$$

where $\boldsymbol{d}(n)/\|\boldsymbol{d}(n)\|$ is a unit vector representing the growth direction of the branch at node n, and α is a constant step that controls the length of the path segments. Vector $\boldsymbol{d}(n)$ is obtained based on the markers in $S(n)$ and their coherence to the goal $\boldsymbol{g}$:

$$\boldsymbol{d}(n) = \sum_{k=1}^{N} w_k(\boldsymbol{m}_k - \boldsymbol{n}), \tag{5}$$

where

$$w_k = \frac{f(\boldsymbol{g}-\boldsymbol{n}, \boldsymbol{m}_k-\boldsymbol{n})}{\sum_{l=1}^{N} f(\boldsymbol{g}-\boldsymbol{n}, \boldsymbol{m}_l-\boldsymbol{n})} \tag{6}$$

are the weights of the markers computed based on a non-negative function f. This function should prioritize both markers that lead to the goal, and those that are closer to the current node n. Our choice for f that satisfies these conditions is

$$f(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \dfrac{1+\cos\theta}{1+\|\boldsymbol{y}\|} = \dfrac{1}{1+\|\boldsymbol{y}\|}\left(1+\dfrac{\langle \boldsymbol{x}, \boldsymbol{y}\rangle}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}\right), & \text{if } \|\boldsymbol{x}\|\|\boldsymbol{y}\| > 0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where θ is the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$, and $\langle \cdot, \cdot \rangle$ denotes the inner product. It can be observed that f decreases as the angle between $(\boldsymbol{g}-\boldsymbol{n})$ and $(\boldsymbol{m}_k-\boldsymbol{n})$ increases (so that markers that are aligned with the goal carry a larger weight), and also as $\|\boldsymbol{m}_k-\boldsymbol{n}\|$ increases (so that markers closer to the node carry a larger weight as well). If the number of markers is large, $\boldsymbol{d}(n)$ will point approximately toward the goal (in fact, it can be shown that $\boldsymbol{d}(n)$ points directly toward the goal if the number of markers grow to infinity). However, if the amount of markers is small, $\boldsymbol{d}(n)$ may deviate from the goal, generating a variety of paths.

The procedure described so far creates a sequence of nodes connected by path segments but no bifurcations in the tree. To create bifurcations, one father node n must be connected to at least two different children nodes n_1' and n_2'. When the first child node n_1' is created, it retrieves the markers around it according to a "restriction distance" (so that the number of markers available to n is reduced). Then, $S(n)$ is recomputed with this reduced set of markers.

To define if the node n will have another child, a new random variable ξ is generated, and the test in condition (3) is applied. If the condition is verified, a new child n_2' is granted to n, and it is obtained through Eq. (4), using the reduced set of markers (i.e., the remaining markers around n that were not restricted by n_1). This node also retrieves the markers around it, and the process for creating child nodes for n is repeated until there are no markers available to n, or condition (3) fails. It should be noticed that, at each iteration, a new random variable ξ is created and compared with the probability $P(n)$ given in Eq. (2) to decide if node n will have a child or not. Hence, some nodes may have more children than others (usually, the number of children increases near the goal, as explained before). In fact, some nodes may have just one child and do not present any bifurcation at all. Nodes that present at least two children are called "key positions," and there is a unique path between adjacent key positions (see darker dots in Fig. 1(b)).

While a tree path is being computed, the agent is able to walk along the generated paths. Although other path planning algorithms can also be used (such as the A^* search algorithm), the proposed model presents some advantages. First, for group behaviors (follow, escape, and collaborative actions such as surround behavior), one important aspect is the diversity of paths (it is desirable that a group walks to a specific goal by occupying the space with different possible paths). Another interesting aspect is that we are able to recompute our trajectories from the needed position; for example, when the target of the agent (in follow behavior) crosses the tree path, we can recompute the path from the intersection point. Further details can be found elsewhere in Rodrigues et al. (2009). The next section describes how the agents move along the tree, after the path has been computed.

Computing the Motion of Agents

Tree paths provide local goals for agents. However, an important challenge in groups motion should be treated in this model: collision avoidance. As mentioned before, tree paths G and G^* related to different agents can share markers, consequently agents walking in tree paths can collide, passing by closer (or the same) path nodes. To address this problem, we used the method for minimum distance enforcement among agents, proposed by Treuille et al. (2006). Their method proposes iterations over all pairs of agents within a threshold distance $t_{\min}$, symmetrically pushing them apart, so that the minimum distance is enforced, as describes in Eq. (8).

$$\boldsymbol{d}_{\text{enf}}(\boldsymbol{p}_i) = \frac{1}{\#A_i} \sum_{j \in A_i} \frac{\boldsymbol{p}_i - \boldsymbol{p}_j}{2}, \quad \text{with } A_i = \{j \mid d_{ij} < t_{\min}\} \tag{8}$$

where $t_{\min}$ is the minimum threshold distance allowed (that was set experimentally to 0.5 m), d_{ij} is the distance between agents i and j, and $\#A_i$ is the number of elements in the set A_i. Consequently, modifying the position of one character impacts all other characters. We implemented a translation for pairs of agents that are closer than the minimum distance. Indeed, this strategy does not present any compromise in the computational time for a small number of agents (main focus of this work). Anyway, more details about computational time are discussed in Section 7.

There are two manners to provide agents motion. The first one is used when agents have specific goal location; the second, when there are no goals. In the later case agents are affected by the markers, which can have different weights. Indeed, markers are used as discretized information

of the available space, but also they can affect differently the motion of agents, depending on associated weights. Given an agent I at the position $\boldsymbol{p}(t)$ and goal $\boldsymbol{g}(t)$, at each time iteration t and given its computed tree paths G, the next agent position is computed by

$$\boldsymbol{p}(t+1) = \boldsymbol{p}(t) + \beta \frac{\boldsymbol{d}(\boldsymbol{p}(t))}{\|\boldsymbol{d}(\boldsymbol{p}(t))\|} + \boldsymbol{d}_{\text{enf}}(\boldsymbol{p}(t)), \tag{9}$$

where β is a constant that controls the length of the agent step, $\boldsymbol{d}(\boldsymbol{p}(t))$ is an orientation vector from the agent's current position $\boldsymbol{p}(t)$ to the next path node coming from tree path, indicating its local goal (and attaining the global goal $\boldsymbol{g}(t)$). Also, we compute $\boldsymbol{d}_{\text{enf}}(\boldsymbol{p}(t))$, a result vector for minimum distance enforcement among agents, proposed by Treuille et al. (2006) to avoid collisions.

In the goal-based motion of the agent, which includes the tree path computing, three events should be iteratively managed:

1. *Agent's decision*: When an agent reaches a key position, it should take a decision to which tree branch it should follow. This decision is considered taking into account how close to the goal the tree branch brings the agent.
2. *Branch death*: There are two reasons to remove the branches in a tree: (i) when a branch was not chosen by the agent (last item), the branch and its children are removed; and (ii) when the goal changes position, branches far away from the goal (defined by the *recomputing distance* R_d) are removed and then recomputed to take into account the new goal position.
3. *Branch reaches agent's goal*: Tree path stops growing, but the agent keeps walking along the paths until it reaches the goal.

In addition to goal-based motion, there is another manner to compute the motion agents that is useful when goals are not explicit (e.g., in formation and alignment behaviors). In this case there is no goal vector but an agent motion direction $\boldsymbol{m}_d$, which is computed based on a variation of the weights of the markers within the agent's personal space. Given an agent I at the position $\boldsymbol{p}(t)$ at each iteration t, and given the set of M markers $S(\boldsymbol{p}(t)) = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_M\}$ within the personal space of the agent I (all markers are considered), we first find the set $S'(\boldsymbol{p}(t))$ of M orientation vectors from agent I to all the markers in $S(\boldsymbol{p}(t))$, to compute the agent direct motion $\boldsymbol{m}_d$:

$$S'(\boldsymbol{p}(t)) = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_M\}, \quad \text{where } \boldsymbol{u}_k = \boldsymbol{m}_k - \boldsymbol{p}(t), \quad \text{for } k = 1, \ldots, M. \tag{10}$$

The motion direction $\boldsymbol{m}_d$ of the agent is computed by

$$\boldsymbol{m}_d = \sum_{k=1}^{M} w_k \boldsymbol{u}_k, \tag{11}$$

which is very similar to the branch growing procedure given by Eq. (5). However, the weights w_k in Eq. (11) are not related to any goal but instead selected according to the desired behavior (see Sections 5.2 and 5.3 for details). The next position of the agent is given by

$$\boldsymbol{p}(t+1) = \boldsymbol{p}(t) + \beta \frac{\boldsymbol{m}_d}{\|\boldsymbol{m}_d\|} + \boldsymbol{d}_{\text{enf}}(\boldsymbol{p}(t)), \tag{12}$$

where β and $\boldsymbol{d}_{\text{enf}}(\boldsymbol{p}(t))$ are the same parameters used in Eq. (9). Next, we present some examples of behaviors that can be obtained using the two strategies for the motion of agents described in this section: based on goals and based on directions.

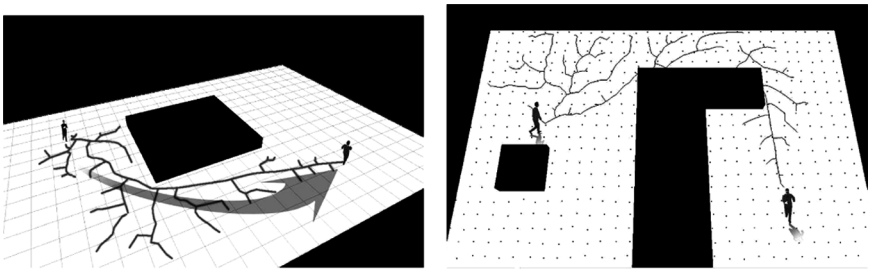
GROUPS STEERING BEHAVIORS

In addition to tree paths described in the last section, we propose group behaviors that derives from that. First, behaviors of pursuing, escaping and surrounding are described. Second, we show the possibility of representing various groups formation that can be used in entertainment as well as other applications. Then, an approach for group alignment is described, which is also important in group formation. Section 7 presents several examples of those behaviors.

Behaviors: Pursue, Escape, and Surround the Goal

The best way to describe the pursuit action takes into account a path between only two individuals (one follower and one target agent). Moreover, as a target location can change dynamically, the path between follower and target should be recomputed iteratively (as described in the last section). Figure 3(a) illustrates such behavior in two scenarios with obstacles.

In this behavior, once the target agent crosses the tree path of a follower agent, the tree path is recomputed from the intersection point. This behavior can be easily scaled with more than one follower agent. In the case of a group of agents following the same target, it is desirable to provide a surround behavior. A simple change in the rule to allocate markers can be used to obtain interesting results: Instead of sharing the markers among the tree paths of the pursuer agents, each tree path gathers markers from disjoint sets of markers. With this simple choice, the



(a) (b)

FIGURE 3 (a) Pursue behavior: one agent tries to reach another one; (b) The illustrated tree path shows the way for follower agent to achieve the target.

pursuer agents fight for space, and must find other possibilities to move. Hence, the pursue behavior arises as a consequence, as illustrated in Fig. 4.

Finally, another small change in the path planning algorithm can provide escape behavior. In path planning, the path node is accepted in tree path when it brings the agent closer to its goal. In a escape behavior, all agents should try to get away from a predefined position $\boldsymbol{c}$. To cope with this condition, wider variety of branches should be created far from the position $\boldsymbol{c}$, and not close to the goal as described in Eq. (2). In fact, this behavior can be achieved by replacing $P(n)$ in Eq. (2) by a new function $P_e(n)$:

$$P_e(n) = \frac{\|\boldsymbol{n} - \boldsymbol{c}\|}{\|\boldsymbol{n}_c - \boldsymbol{c}\|}, \tag{13}$$

where $\boldsymbol{n}_c$ is the node located the farthest away from $\boldsymbol{c}$.

If there is just one follower agent, $\boldsymbol{c}$ is exactly the position of the follower. If more than one follower agent is present, then $\boldsymbol{c}$ is obtained as the centroid of all the agents' position. Figure 5 illustrates such behavior.

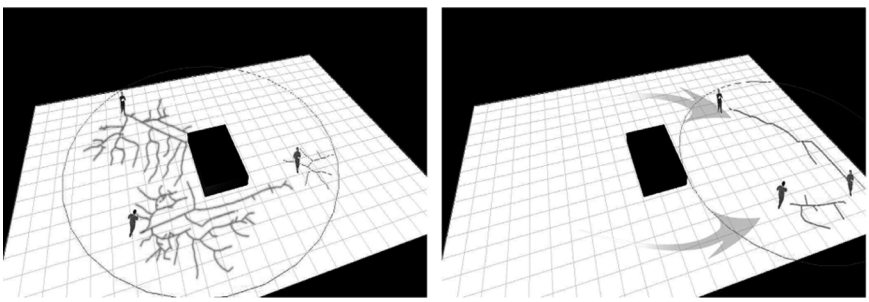


FIGURE 4 Surround behavior: two agents try to reach another one. It is possible to observe the diversity of generated tree paths causing the surround behavior.

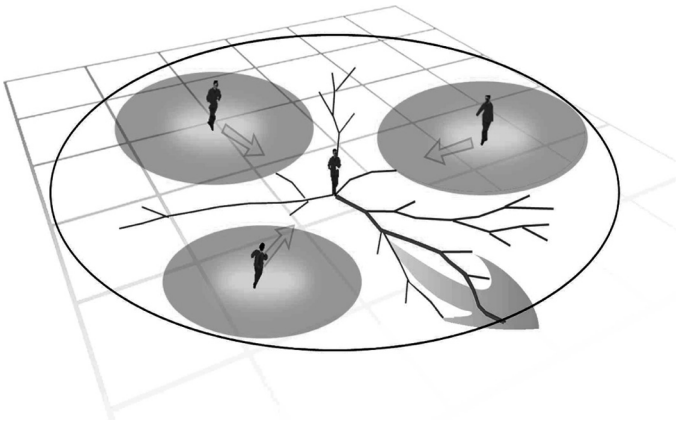


FIGURE 5 Escape behavior: one agent tries to escape from a circular region, in which three follower agents are included.

In this figure the escape region is automatically computed based on three follower agents, and consequently the tree path algorithm tries to bring agent outside such region.

Behavior: Groups Formation

This behavior aims to provide the formation of specific shapes, which is relevant in several entertainment applications. For instance, games and movies, as well as theatrical performances, can use such characteristics to provide group motion. There are at least two different ways to model such behavior. The predefined one considers the generation of specific goals into a shape, and the posterior distribution of the individuals into the group. The drawback of this approach is the low flexibility if a shape changes dynamically or if more agents want to participate in the performance, because it requires the recomputation of specific goals for each agent. The second approach describes an emergent behavior of agents to occupy the space corresponding to the desired shape. We adopted the last approach in our model by using markers in the space.

Initially, a shape region should be defined (as illustrated in Fig. 6, where the shapes are the letters V and H). It can be done by using our markers spray (more details can be found in Section 6.1). At the beginning, the environment has markers to allow the agents motion. Then, the user can spray markers to define the shape formation. Consequently, markers are painted over the environment, increasing the density of markers in formation shape. Yet, the markers into the target shape have increased weight to a defined constant $w_{\max} \geq 1$. The consequence is that markers into the target shape will have more importance in Eq. (11) than markers outside.

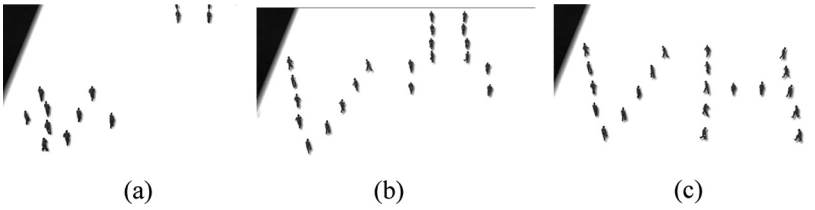


FIGURE 6 Agents form explicit shapes of letters V and H.

The algorithm is described in two steps. The first one takes into account the tree path algorithm described previously. In this case, an automatic goal into the target shape is attributed for each agent in the simulation, providing motion stimulus for each individual. The automatic goal takes into account the convex hull of the target shape, and it is selected as a random position within it. When the agent gets close to the shape region (it is identified through markers analysis, i.e., if one agent has a target marker into its personal space), the orientation vector computed in tree paths algorithm is not any more taken into account. In other words, Eq. (9) is used initially to guide the motion of agents, and then it is replaced by a modified version of Eq. (12), using the following weights w_k to obtain the motion direction $\boldsymbol{m}_d$:

$$w_k = \begin{cases} w_{\max}, & \text{if } \boldsymbol{m}_k \text{ is within the target shape} \\ 1, & \text{otherwise} \end{cases}, \tag{14}$$

where $w_{\max}$ is the increased weight within the desired formation region. Larger values for $w_{\max}$ result in a more direct motion of each agent toward its goal, whereas smaller values lead to smoother (and slower) movement. Experimentally, we observed $w_{\max} = 10$ presented good results, and used this value in all experiments.

When other agents arrive in formation shape, they fight for space, but they tend to keep inside the formation shape, because the weight of the markers is greater than outside target shape markers. Moreover, if all agents should present same specific orientation in target shape, their distribution is easily regulated after the agent entries in the shape region. The final distribution of agents for the VH shape is given in Fig. 6(c).

Behavior: Groups Alignment

This behavior, as the one described in last subsection, is useful to provide group performances in entertainment applications. Alignment of people is an interesting feature that can be used in several applications. In our model we are able to have people moving based on specific

goals (using tree path algorithm; Section 4.1) as well as without goals, by changing the weights of markers into agent personal space, and then generating the motion of agents, as in last section.

For our group formation, we are able to create alignment regions with predefined weight masks for markers. One possible mask used to provide horizontal and vertical alignment is illustrated in Fig. 7, on the left. In this case, the markers into the formation mask have their weights increased according to $w_k = \text{dist}(I, k)$, where $\text{dist}(I, k)$ is the Euclidean distance from the current position of agent I to the marker $\boldsymbol{m}_k$. These weights are used to obtain the motion direction $\boldsymbol{m}_d$ used in Eq. (12). If a given marker is within the personal space of more than one agent, these agents compete for the marker. In fact, the same marker may present different weights when viewed by different agents, depending on their goals and relative position w.r.t. the marker. To minimize the chance of more than one agent reaching the same marker at the same time, the weight of the marker is increased for the agent that is the closest to it. More specifically, this weight is recomputed as the sum of the weights of that specific marker as viewed by all agents having the marker within their personal spaces, so that the closest agents tends to reach the marker faster than the others.

EXTENSION TO CONTROL CROWDS

The space colonization algorithm, briefly described in Section 3, has inspired the method we introduced in the previous sections for tree paths and steering behaviors presented in this article. The key idea is to represent empty space explicitly, using a set of markers, and to treat these points as a resource for which the agents paths in the simulation compete. Consequently, tree paths are generated as a function of free space, goals, and group behavior to be applied. Such paths present discretized goals that are achieved progressively by the agents as they move along the path.

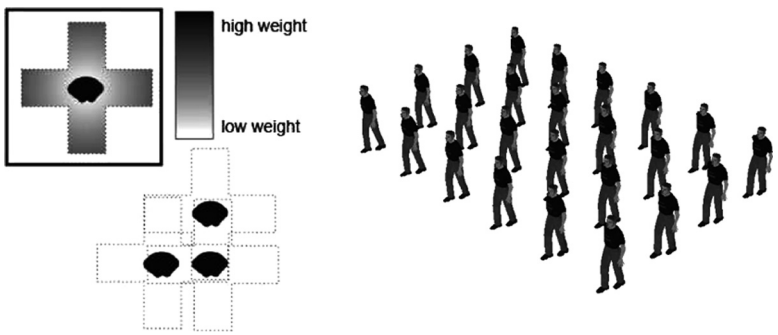


FIGURE 7 Left: Weight masks are used to define alignment behavior. Right: An example of generated behavior.

As described in Section 4.2, tree paths can share markers, so collisions among agents can happen. In the original article (Rodrigues et al. 2009), we proposed the use of an enforcement distance $\boldsymbol{d}_{\text{enf}}$, defined in Eq. 8, that is used to avoid collision with other agents. In fact, the motion for each agent along its path is given by Eq. (9). On the other hand, when agents have no specific goals (e.g., groups formation and alignment), agents move according to weights defined in the space, as explained in Section 4.2. In this case, the enforcement distance is also used to avoid collisions with other agents, as shown in Eq. (12).

Although there is no theoretical limit on the maximum number of agents that can be simulated using either of the above-mentioned approaches (with or without goals), the enforcement term $\boldsymbol{d}_{\text{enf}}$ in Eq. (8) requires the computation of all pairwise interactions of agents within a threshold distance. Hence, the cost to compute the enforcement grows quadratically as a function of the number of agents, which could be prohibitive when dealing with crowds.

To overcome this problem in the first situation (agents with goals), this extended version of the article eliminates the enforcement term $\boldsymbol{d}_{\text{enf}}$ when computing the motion of agents. In this situation, the markers that were used to generate the tree paths are also considered to guide the agents. In fact, the set of markers used to generate each node of the tree path is stored in a database when the path is created. When an agent I_i reaches a node $n_{i,k}$ in its tree path G_i, it retrieves all the corresponding markers. When it is time to move to the following node $n_{i,k+1}$, the motion is only possible if all the markers related to the destination node are not retrieved by any other agent $I_j, j \neq i$ (meaning that there is no other agent in the vicinity of its path). If that happens, agent I_i also retrieves all the markers related to $n_{i,k+1}$, and it keeps moving. If not, agent I_i stays still in its current position, until all the markers of the destination node are released.

If the radius A_r of the action region is greater than or equal to half the distance α between adjacent nodes n_1 and n_2, i.e., if $A_r \geq \alpha/2$, then the set of markers related to either n_1 or n_2 covers the whole segment between these two nodes. Hence, any other pair of adjacent nodes n_1' and n_2' that intersects the segment $\overline{n_1 n_2}$ would also share markers with either n_1 or n_2. Hence, if an agent moving along the nodes n_1 and n_2 captures the markers related to those nodes, another agent moving along n_1' and n_2' could not move, avoiding any possible collision. In all experiments, we used $A_r = \alpha/2$. Section 7 presents some experimental results in crowded scenarios.

Interactive Interface to Control Crowds

An important aspect in virtual human simulation is the behavior control. Some existing models produce interesting results; however, interfaces are not interactive and sometimes even the parameters are not

easy to be defined. In our work we took considerable effort to develop an interactive tool where groups and individual behaviors could be easily defined through a sketch interface.

First, markers are in the center of our model, because they produce high impact in generated trajectories as well as in agents' behaviors. The amount of markers and their position in the space can be easily modified using the sketch interface, by interactively spraying or erasing markers. Figure 8 shows a screenshot of our prototype system that implements such interaction.

We used dart-throwing algorithm for generating the markers (Cook 1986). This process is performed in preprocessing stage and recorded in a file, because it is not necessary to recompute it for same scenario. If new markers are sprayed/erased during the simulation, the changes can also be recorded in the file. This algorithm presents a computational complexity ($n \log n$), so we investigate the possibility of using a random distribution of markers in scenario to improve computational time. In this case we consider initialization of a grid of markers and the random process acts as a noise in markers position, providing linear complexity. Results indicate that grids and random techniques can be used without high impact in group behaviors. The evaluation included agents linear and angular velocity and trajectories smoothness. Although these analysis present similar results in both markers distribution techniques when density of markers is 60 markers/m^2, here we keep using dart-throwing algorithm as originally used by Runions et al. (2005). One reason for that is the fact that markers density cannot be fixed in interactive tool, because users can spray again in the same space.

Agents tend to follow paths with higher density of markers, so that local control can be achieved by increasing the number of markers along

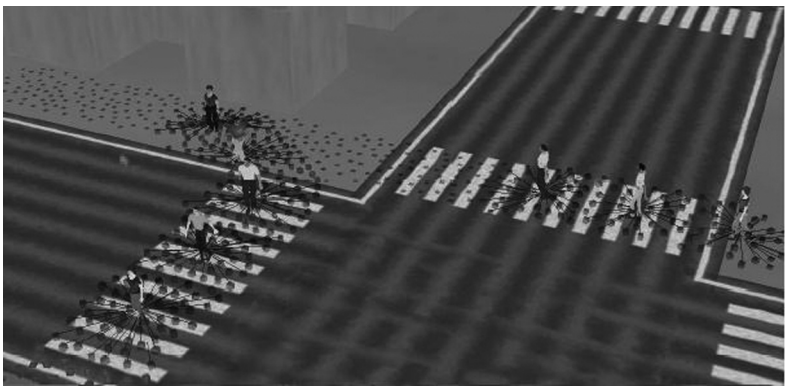


FIGURE 8 Prototype system for crowd simulation with interactive control. The user "sprays" markers (dots) on the floor. The distribution of markers directs the agents.

FIGURE 9 Removing markers in the environment affect the virtual humans' trajectories (the first and second agents are influenced by the new configuration of markers.) The circle represents the marker eraser, and it has been used to narrow down the region where the agents can walk.

preferred paths. When markers are removed, agents immediately adjust their paths, as shown in Fig. 9.

Besides the interactions with markers, other features are also presented, such as creation of agents, selection of one or multiple agents, definition of group behaviors to be applied, and definition of goals and paths for agents. In addition, other functionalities like illumination, texture, and camera animation are possible, as described in Musse et al. (2007).

EXPERIMENTAL RESULTS

This section presents additional results to the ones previously published (Rodrigues et al. 2009). First, we describe the main variables of the model. It is important to emphasize that these variables do not need to be recalibrated when environment or simulation scenario changes. However, they can be changed if users want to generate different results.

Parameter R is related with circular region around an agent and represents its perception field. This variable, set experimentally to 1.2 m, is used when agents are not walking in the tree path, and their motion is based on markers weight (e.g., group formation and alignment). Coherent with R, the radius of the action region A_r related to each path node, is set to the same value 1.2 m. Variables β and α are related respectively with agents step motion in the tree path and the size of path segment of tree path. Values used in all simulations are $\beta = 0.05$ m (assuming a simulation at 24 FPS) and $\alpha = 2.0$ m.

Figure 10 shows an illustration of combined behaviors for formation and alignment applied together. It is interesting to perceive that in the beginning 1) agents apply a hybrid behavior (goal-based) until they have

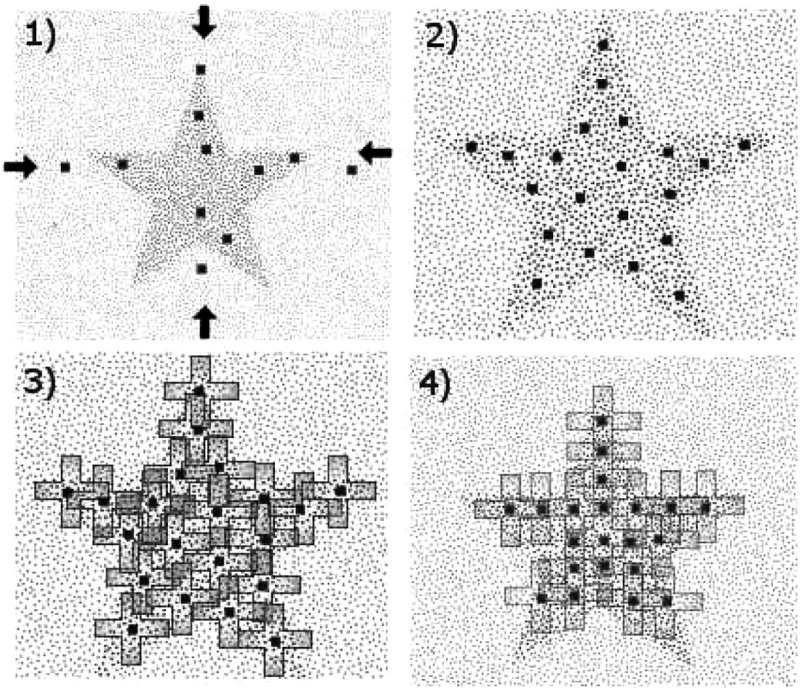


FIGURE 10 This figure shows four different time steps in the formation and alignment behaviors applied together. In 1) and 2) agents are going into a shape region. In 3) and 4) agents align their locations as a function of other agents.

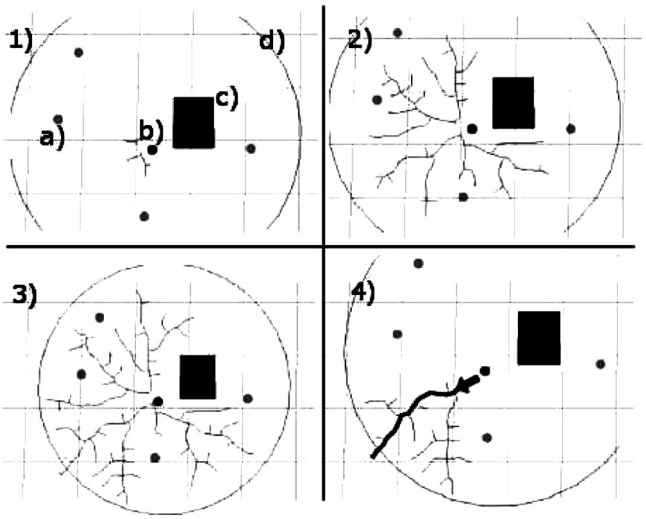


FIGURE 11 In 1), a, b, c, and d are related, respectively, with follower agent, target agent, obstacle, and a circular region, which includes all follower agents. In 2) and 3) tree path is evolving and in 4) target agent is walking and escaping from follower agents.

markers of interest region (star formation) into their perception region. Then, in 2) it is possible to see agents into formation region, then applying the alignment behavior, when weight masks are visible 3). Finally, agents occupy the formation region and in the same time they keep aligned as shown in 4). The possibility of combine different behaviors (with or without tree paths and goals) is a great potential of our method.

The escape behavior is implemented based on a small change in tree paths algorithm. Indeed, instead to be attracted by the goal, the escape behavior is attracted by the nodes which are far away from a specific location, as described in Section 5.1. Figure 11 illustrates this behavior when one target agent is followed by four other agents.

The computational performance of tree path technique is very dependent on the number of generated nodes, which also takes into account the simulated environment (obstacles, number of agents, distance to the goal). In Fig. 12 we show the time[4] consumed for 1, 5, and 10 agents, including the number of iterations performed in the simulation. It is important to note that maximum number of iterations observed in our simulations is 49, meaning that at most 49 iterations are required by the agents to reach their goals in the simulated environment. Because our model is focused on low density scenarios, we have tested a maximum of 50 agents to better visualize the groups behavior, achieving real-time performance. However, according to the worst possibility (49 iterations to reach the goal) in Fig. 12, it is theoretically possible to compute tree paths for thousands of agents.

Figure 13 shows crowd simulated using our method. In these images it is possible to see lane formation, which is an emergent behavior expected in crowds. Indeed, this effect is created because it takes less effort for

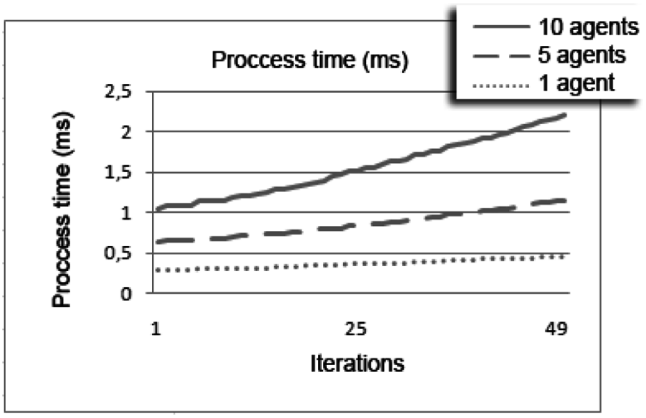


FIGURE 12 Computational time (ms) for processing 1, 5, and 10 agents, considering the evolution of tree paths.

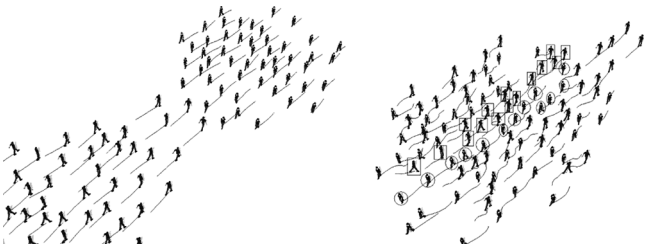


FIGURE 13 Left: Two groups are approaching to another. Right: Lanes can be visualized.

people to follow immediately behind someone who is already moving in their direction than it does to push their own way through a crowd. Emerged as a function of least effort hypothesis, the formation of lanes arises, because people change trajectory whenever they encounter an entity moving in the opposite direction. This action forms chains of entities walking in line, as we would expect.

FINAL REMARKS

This article presented an algorithm to provide motion of groups of agents. It is based on a biologically inspired technique used in the past for generating leaf venation patterns and tree structures, simulating the competition for space between growing veins or branches. Here, we presented some behaviors to simulate groups of agents, such as group path planning (we called tree paths), pursue behavior, surround behavior, escape behavior, groups formation, and groups alignment. The key innovation is the simple way in which the paths are created, by "observing" and "occupying" free space, which is represented using a set of marker points, which leads to a simple yet computationally effective implementation of the competition for space. Global tree path is modeled by biasing the influence of the captured marker points according to their agreement with each agent's direction to its goal, which can be assigned to individual agents or groups. In addition, agents motion can be goal based or influenced by variation of weights attributed to markers, originating alignment and formation behaviors.

Comparing with other models, although RRT (LaValle 1998) explores uniformly the "walkable space," defining positions randomly in this space that will guide the expansion of the search tree, in the proposed model the space is previously discretized using a uniform distribution. Once the space is known, the proposed model considers weighted positions so that the growth of branches is directed to the goal. This difference allows optimizing the construction of the tree and the search for the path in this structure. Compared with roadmaps, the proposed model allows

the generation of a connected tree using a small amount of edges. In a roadmap (Lien et al. 2005), the performance of the connected graph is impaired due to the number of edges necessary to explore the entire space. The proposed model also requires single nearest-neighbor queries, whereas roadmaps require more-expensive *k*-nearest neighbor queries.

This article also presents an extension to the method providing collision avoidance among agents into a crowd. This extension is naturally integrated with markers and their association in the space, which path nodes. In this idea, as described in Section 6, the markers associated to specific path nodes can only be used by agents in such nodes. Consequently, agents avoid collision with others. As future work, we intend to provide other group behaviors, focused on individualities, such as agents skills.

REFERENCES

Bicho, A. L. 2009. From plants to crowd dynamics: A bio-inspired model (in portuguese). PhD thesis, School of Electrical and Computer Engineering, University of Campinas.

Choi, M. G., J. Lee, and S. Y. Shin. 2003. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Trans. Graph.* 22(2):182–203.

Cook, R. L. 1986. Stochastic sampling in computer graphics. *ACM Trans. Graph.* 5(1):51–72.

Dapper, F., E. Prestes, and L. P. Nedel. 2007. Generating steering behaviors for virtual humanoids using bvp control. In: *Proc. of XXV Computer Graphics International*, 105–114. RJ, Brazil, May 2007.

Kallmann, M., and M. Mataric. 2004. Motion planning using dynamic roadmaps. *Proceedings of IEEE International Conference on Robotics and Automation* (ICRA), New Orleans, LA, April 26–May 1, 4399–4404, 2004.

Kamphuis, A., and M. H. Overmars. 2004. Finding paths for coherent groups using clearance. In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 19–28. Aire-la-Ville, Switzerland: Eurographics Association.

LaValle, S. 1998. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR98-11, Dep. of Computer Science, Iowa State University.

Lien, J. M., S. Rodríguez, J. P. Malric, and N. M. Amato. 2005. Shepherding behaviors with multiple shepherds. In: *Proceedings of the IEEE Inter. Conf. on Robotics and Automation*, 3402–3407.

Metoyer, R. A., and J. K. Hodgins. 2004. Reactive pedestrian path following from examples. *The Visual Computer* 20(10):635–649.

Musse, S. R., C. R. Jung, Jr., J. C. S. Jacques, and A. Braun. 2007. Using computer vision to simulate the motion of virtual agents. *Computer Animation and Virtual Worlds* 18(2):83–93.

Musse, S. R., R. Rodrigues, M. Paravisi, Jr., J. C. S. Jacques, and C. R. Jung. 2007. Using synthetic ground truth data to evaluate computer vision techniques. In: *IEEE Workshop on Performance Evaluation of Tracking Systems (in conjunction with ICCV 07)*, 25–32. Rio de Janeiro, Brazil.

Reynolds, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. In: *Computer Graphics*, 21(4) *(SIGGRAPH '87 Conference Proceedings)*, 25–34.

Rodrigues, R., A. Bicho, M. Paravisi, M., C. R. Jung, L. P. Magalhàes, and S. R. Musse. 2009. A model for generating and animating groups of virtual agents. *Lecture Notes in Computer Science* 5773:358–371.

Rodríguez, S., J. M. Lien, and N. M. Amato. 2007. A framework for planning motion in environments with moving obstacles. In: *IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems*, 3309–3314. USA: IEEE Press.

Rodríguez, S., R. Salazar, T. McMahon, and N. M. Amato. 2007. Roadmap-based group behaviors: Generation and evaluation. Technical Report TR07-004, Dep. of Computer Science, Texas A&M University.

Runions, A., M. Fuhrer, B. Lane, P. Federl, A. G. Rolland-Lagan, and P. Prusinkiewicz. 2005. Modeling and visualization of leaf venation patterns. *ACM Trans. Graph.* 24(3):702–711.

Runions, A., B. Lane, and P. Prusinkiewicz, 2007. Modeling trees with a space colonization algorithm. In: *Proc. of the Euro. Workshop on Natural Phenomena*, 63–70. Prague, Czech Republic.

Sachs, T. 1969. Polarity and the induction of organized vascular tissues. *Annals of Botany* 33(2):263–275.

Treuille, A., S. Cooper, and Z. Popović. 2006. Continuum crowds. *ACM Trans. Graph.* 25(3):1160–1168.

Tu, X., and D. Terzopoulos. 1994. Artificial fishes: Physics, locomotion, perception, behavior. In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94)*, 43–50, New York: ACM Press.

NOTES

1. Details are described in PhD thesis authored by Bicho (2009).
2. For the sake of clarity, the time index t will be removed from this point on and used only when necessary.
3. Its size can be calibrated, but normally we use the same radius R defined for the agent personal space.
4. Average of 20 simulated experiments. These results were obtained using monothread implementation without characters' rendering on Intel® Core™ 2 Duo 2.2 GHz, 3 GB DDR2 at 667 MHz and NVIDIA® GeForce® 8400M GS 128 MB.