# Groups and crowds with behaviors specified in the environment

Isaac Rudomin, Fernando Perez, and Erik Millan

ITESM-CEM, Atizapan Mexico
rudomin@itesm.mx,
WWW home page: http://rudomin.cem.itesm.mx

**Abstract.** In this paper we describe a system where behavior of characters and groups of characters are determined by assigning attributes to groups and individuals hierarchically and by using random values as a default, combined with behavior maps that allow the user or the program to specify conditions that modify the behavior of the characters. Different path selection algorithms are used depending on the attributes of the environment and the specific characters. Some optimizations are also described that allow the system to work at interactive rates even with a relatively large number of characters.

## 1 Introduction

Groups and crowds are collections of individuals that can have a common objective (or not). Within a group or crowd, individuals retain their own attributes and different ways of reacting to the stimuli from the environment.

Previous work in this field has focused on assigning properties or attributes to virtual characters. Becheraiz et al [2] say, among many others, that the behavior of a character is based on the perception of the environment and the agents internal or emotional state, which depends on a series of attributes that are assigned to the character. In our work we follow this method by assigning some basic attributes to a character (such as personality or training), that determine a basic behavior that is modified by the internal state or the environment. This allows the generation of more complex behaviors.

Some of the scenes we are simulating involve the reaction of civilians and security personnel in emergency situations [5]. This is why, for human agents we specify a type of behavior for each individual or group (walking, resting, etc.), a triggering condition (is he within a group or isolated), a personality/training type and a reaction (that depends of the attributes of each individual, so it is statically defined). For animals,behavior is programmed basically by the type of animal (ants, mammals, birds). One important and convenient aspect of the way we assign attributes is hierarchical (groups, subgroups and individuals inherit values of attributes that are not assigned explicitly at that level), and that random values are used when a value is not explicitly assigned. The model is of course simplified to the point of being simplistic and still lacks many obvious attributes (such as mood).

Behavior maps have been used as a way of assigning environmental conditions that trigger behaviors in the characters within a scene[10] where using several layers to assign different behaviors and other related conditions that modify the behavior of the characters, such as obstacles (for avoidance), points of interest (attraction or repulsion), visibility, etc.

These behavior layers are images that let the user define the behavior of a character in a specific area of the environment by painting a specific color, so that the character, when it is within the affected area, reacts appropriately. The collision map specifies forbidden areas. In our work we do something similar, but:

- We use behavior maps generated randomly or dynamically during program execution, as well as those painted by the user
- In our case the behavior is not specified directly by the behavior map, rather, the behaviors change based on the values on the map but also based on the attributes of the specific character.

The environment is codified as layers of images. Many can be used in theory, and in our implementation one image represents the scenary as a height map and there are also images representing the damage caused by environmental factors (a volcano) or the position of food in the environment, depending on the specific application or example. The values of pixels at each of the layers, together with the attributes that have been assigned to the agents, determine the behavior of the said agents.

Another aspect that has been studied is the generation of trajectories: some follow Bezier curves[6], others treat each element of a crowd as a particle[7, 8]. Other approaches use Voronoi diagrams[3, 4] (or other path planning algorithms to generate free paths). We have also seen in the previous paragraphs the behavior map approach to path planning. In our work we have implemented different algorithms depending on the situation. Gradient maps define obstacles that are different for each character, and and a Bezier curve approach is used for areas where the gradient is too low to clearly define a path.

Another important topic is creating crowds that work in real time by using various optimization, even using complex illumination for these scenes[11]. Use of level of detail is a way of optimizing the use of available resources, and it has been tried in several ways, such as compressing textures, use of previously created images[9], impostors[1] or point based rendering[12]. These reduce the use of memory and processing time when there are large numbers of characters in the scene, but can reduce the realism of the simulation. In our work we have used the point based approach.

## 2 Behavior specification for groups and individuals: assigning attributes

Attributes are assigned in two ways: interactively or from a file. Through a file you can have more details. Interactively, many of the attributes can be assigned in a random fashion.

### 2.1 Through a file:

Every time a program is executed a file is generated where the attributes used in a particular application are stored (this allows us to load them again, later). The behavior of a crowd can be specified through a text file where we say how many groups we have as well as the attributes of each of them.

The file is organized so that first group behavor is assigned, followed by the behavior of individuals in the group, if desired. The file contans the following:

- The scene that is being developed, and the scenary used for this scene.
- The number of groups and subgroups (subdivisions that allow us to assign attributes to the individuals in the group in a succinct fashion).
- The types of characters: this allows us to assign the personalities or training of individuals in each subgroup (for example, in a disaster simulation one would have civilians and security personnel).
- The behavior, which tells us how subgroups will conduct themselves in their environment (walking, running, resting, etc.).
- Conditions, which allow us to assign to each subgroup whether it is within a larger group or it is isolated.
- The number of clones, that is the number of characters that form a group. Here is where we start getting individual behaviors, that will be similar to those of the group but more detailed.
- Number of subclones (characters that have the exact same attributes).
- Type of character, conditions and behaviors can also be specified for an individual.

The file also has the names of each group, the size of the influence of the behavior maps, as well as the maximum steepness an individual can climb.

The assignment of all attributes can also be done randomly if left empty (the default).

### 2.2 Through the user interface:

In the user interface you can assign the following parameters: Scene, number of groups, number of clones, area of influence of the behavior map and steepness. We see an example of the user interface in figure 1. The attributes of each individual are assigned randomly in this case. Both methods of assigning attributes generate a file that can be used later.

## 3 Behavior Maps

We define a scene with a file or interactively, as well as behavior maps, that together tell us what actions the crowds will take as well as how they will react within the scene. This scene structure is linked to the scenery since the layers of behavior that are loaded depend on both.

These layers give us the stimulus/responses that will affect the crowds and depending on these the characters will have a different behavior. The layers of


**Fig. 1.** User Interface

behavior can be programmed in their range of action and they are specified by using color maps that allow us to know if in a given moment an individual is within one. If this is so, the individual will change its behavior depending on the stimulus given by the layer. In figure 2 we see two examples of height maps for different scenes (explosion and volcano).

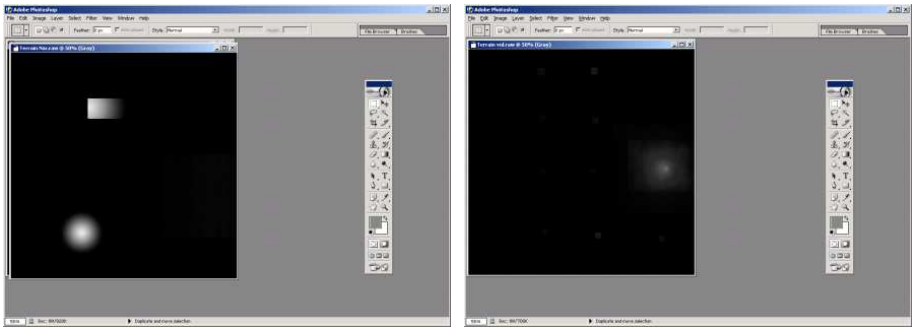

**Fig. 2.** Height maps for explosion and volcano

For example, if we have an explosion, the range of action of the layer is calculated from the point where the explosion takes place, and a color map is generated so that if an individual is within this field, it will be affected depending of the attributes that it has been programmed with.

## 4 Paths

In our application, crowds sometimes move along Bezier curves[6]. Every individual in the group moves along a different curve to make the movement less stiff and more realistic. The curve starts where the character is placed, it ends where the character must go, and random control points are added to make each character move somewhat differently.

On the other hand, sometimes it is more convenient to use gradient maps. The gradient is the maximum slope in a curve. Knowing it is important because some characters can handle more pronounced slopes than others. Given these restrictions we can determine what route a character can follow.

This maximum slope can be specified by the user interactively or in a file. The gradient map is calculated from the height field that specifies the scenery, this way, it knows the height in the terrain through the one point(x,y). The character when moving has one direction , it knows the direction and the actual position, so we can estimate the gradient in the next step. If the road becomes too steep for a character, it will look for another one by going in random directions (since it knows nothing about the area where he is).

How do we select among these algorithms for path selection? We do it depending of the relative size of the characters to the scenery where the crowds will perform; if obstacles are large, the gradient map is used (there are different thresholds for different characters), if small, the control points of the Bezier curves are used.

We do this because when obstacles are large, collisions are avoided by the use of gradients and a slope limit. Large slopes force characters to seek new routes. Where obstacles are small, however, just like for the gradient, we know the next step, and if the slope between the actual position and the next point is bigger than the maximum slope, collisions are avoided by moving the control points of the Bezier curves and modifying the trajectory, but without changing the starting and ending points.

We see agents using both these methods in figure 3. Compared with other algorithms, such as those mentioned in the introduction, this allows us to take advantage of the information contained in the height map, and to avoid collisions with the objects when the gradient is bigger than the maximum specified, so is not necessary to create geometric figures around the objects (such as in [8]) or to previously know the positions of the objects ( as in[4]) to calculate some predefined path.

## 5 Optimization

Rendering a large number of characters in a scene is very expensive, so various optimization tricks are required in order to render the scene at interactive rates. For level of detail we use a multi-resolution point sample rendering algorithm for keyframe animations that is slightly different to that presented by Wand et al[12].

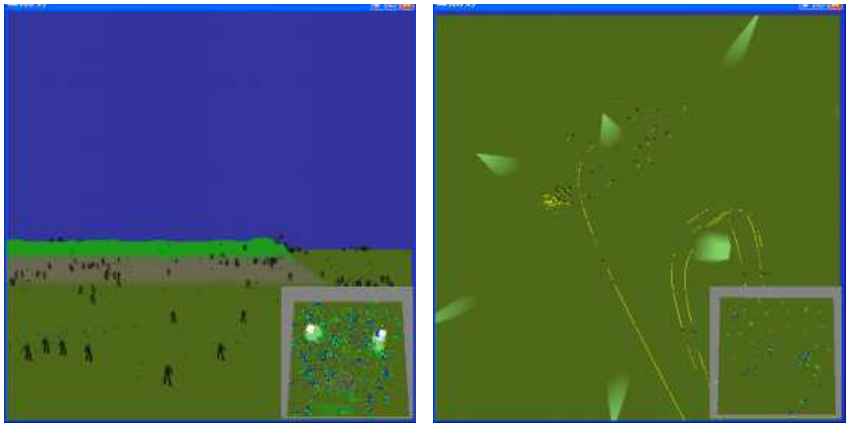

**Fig. 3.** Examples of path finding through the gradient map and through Bezier curves

There are many advantages in the point-based approach for rendering distant characters. This representation allows the smooth animation of a character frame after frame, while the number of primitives necesary is reduced according to the detail desired for the character. Furthermore, a point-based provides a good representation of the geometry from any point of view, in contrast to other image-based techniques where the model appearance is either invariant to the point of view or expensively modified when the camera moves quickly to a distant position.

Point based rendering is achieved through a hierarchical octree structure. The vertices in the model are used as the points that will be rendered in the model. These points are added one by one to the octree structure, increasing the resolution of the octree as necessary. When this octree is complete, it is traversed as shown in figure 4.

When traversing the octree , we will use (sequentially) points from each of the top level branches of the octree. If the branch has no points assigned it is skipped. If the branch contauns a single point, then it is selected. If te branch is occupied by an octree, the algorithm is recursively applied. All points thus obtained are marked as visited and stored in a list. This list will be used to select the points to render the model. The number of points to be rendered is obtained as a function of the distance for each caracter. Once this number of points is obtained, the points are selected from the beggining ot the previously generated list.

Since the models are animated, it might be thought that the octree representation should be obtained at each step of the animation, however, the sampling obtained is generally evenly sampled no matter what the pose of the model is . A different approach (that we tried in a different application) would be to create a hierarchical structure based on the skeleton of the character instead of a regular octree. It works slightly better, if you have a skeleton, but it is not the case here

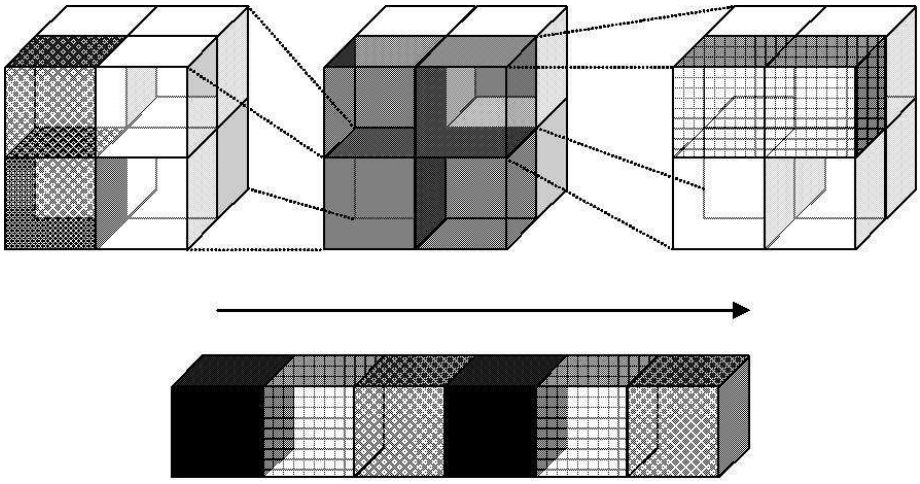

**Fig. 4.** Octree Traversal

(because the Quake characters we are using do not come with a skeleton). We use this algorithm to render animated crowd scenes where characters that are far away but still visible are drawn as points while the complete character is drawn for characters that are close.

The computing and storage requirements for scenes such as those used in virtual reality applications that include crowds exceed by far the capacity of graphics hardware. These scenes have a complex structure and their display requires a large number of polygons (or points if point based rendering is used). That is why we use a simple form of culling so that by surrounding the scene with a box (it is obtained from the OpenGL viewport) and drawing only the points or polygons within this box. By using culling together with point based rendering, we were able to obtain interactive rates with scenes that have up to 2,500 characters, and decent rates even with 10,000 characters, as can be seen in table 1. We used the following machines for testing:

- Machine 1 has a Xeon 2.4 GHz, 1 GB RAM, and Nvidia Quadro FX 2000
- Machine 2 has a Pentium 4 at 2.8 GHz and an ATI FireGL X1
- Machine 3 has a pentium II at 700 MHz with a GForce4 MX

Our models were Quake 2 and 3 models. We used the scene of the explosion, before the explosion happens to measure these values, that vary significantly on each run since many parameters are initialized randomly, but the values we found are typical.This table shows that even though we are within interactive rates, there is still work to be done in optimizing the applications.

| machine | num characters | num groups | num clones | fps cull | fps no cull |
|---|---|---|---|---|---|
| 1 | 100 | 10 | 10 | 38 | 26 |
| 1 | 400 | 20 | 20 | 19 | 16 |
| 1 | 2500 | 50 | 50 | 8 | 5 |
| 1 | 10000 | 100 | 100 | 2 | 1 |
| 2 | 10000 | 100 | 100 | 7 | 2 |
| 3 | 500 | 25 | 20 | 5 | 2 |

**Table 1.** Results

## 6 Scenes and Scenery: Examples

A scene consists of the actions a crowd will perform and the scenery is the environment where the scene takes place. In this paper we describe three different scenes for humans and one for animals. Scenary is created by using height maps, which is fast and simple.

In the volcano eruption disaster scene, there are two kind of characters: security personnel and civilians. Each one has loaded attributes that allow different reactions to the eruption. For example, civilians don't have training for this phenomenon, but they have natural reactions like runnig away from the eruption. Security personnel have training and will form a line protecting civilians simultaneously to their fleeing. Both civilians and security personnel can flee in group (following the lider [5]) or individually depending on its attributes.

The damage map has as its center the crater of the volcano, and there are meeting points where the characters converge to avoid damage, creating paths to arrive at these points depending on the obstacles.

See figure 5.

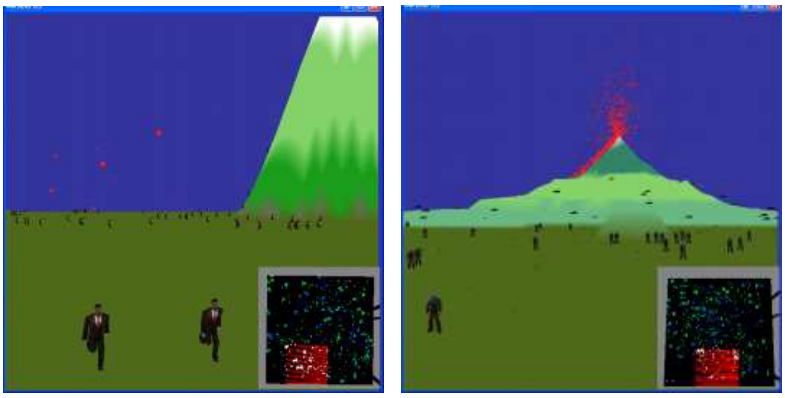

**Fig. 5.** Volcano

For the scene of the explosion, the personalities of the characters are of three kinds: security personnel, attackers and civilians. The security personnel must pursue the attackers once the explosion has occurred (maybe it is too late). The attackers flee from security personnel, and civilians respond according to their own attributes (some run, some are paralyzed by fear). See figure 6 (a).

When the explosion occurs, a damage map is created that has as its center the center of the explosion, and a size defined by the user. Individuals within the damage circle will die or be wounded.

Finally, for the animal scene, there are three kinds of animals: A herd of herbivores, a swarm of ants and a flock of birds.

Ants and herbivores look for food. In this case behavior maps are created around the points where food is present. An animal within these areas will react according to its kind (strangely enough, herbivores and ants eat the same food in this simulation). Flocks of birds will be looking for places where to rest. See figure 6 (b).

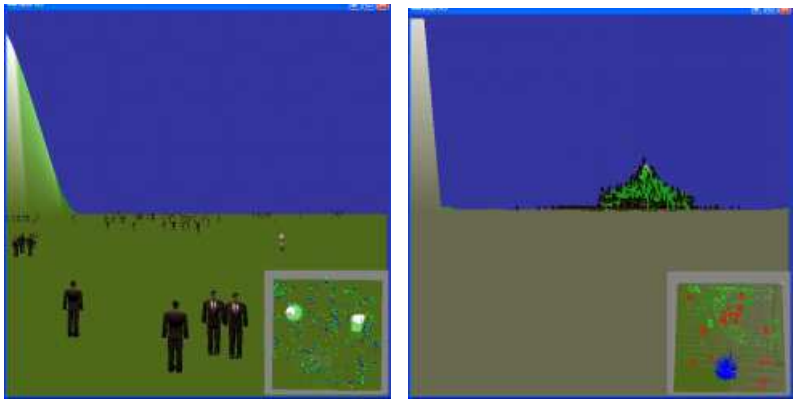

**Fig. 6.** Explosion and animals examples

## 7 Conclusions and Future Work

The results obtained were satisfactory: by this we mean:

- We obtained complex behaviors by specifying group and individual attributes hierarchically and by using behavior layers.
- We have a way of easily creating complex scenes.
- Several different algorithms are used for path selection, depending on the attributes of the environment and its relation to the specific characters.
- Culling and point based rendering made the system work at interactive rates with a relatively large number of characters.

One aspect that is grounds for future work is that the programming of the characters and their reactions to the environment are hard-wired and static. Implementing scripting will make the system much more flexible by allowing us to assign specific movements or behaviors to each group and specific tasks at runtime. There is still work to be done on optimizing the application and improving image quality.

## References


1. Aubel, A., Boulic, R. and Thalmann, D.: Real-time Display of Virtual Humans: Levels of Detail and impostors. IEEE Transactions on Circuits and Systems for Video Technology (2000)
2. Becheiraz, P. and Thalmann, D.: A Behavioral Animation System for Autonomous Actors personified by Emotions, Proc. First Workshop on Embodied Conversational Characters (WECC 98) Lake Tahoe, USA (1998)
3. Hoff, K., Culver, T., Keyser, J., Lin, M. and Manocha D.: Interactive motion planning using hardware accelerated computation of generalized voronoi diagrams. IEEE Conference on Robotics and Automation (2000) 2931–2937
4. Hoff, K., Culver, T., Keyser, J., Lin, M. and Manocha D.: Fast computation of generalized voronoi diagrams using graphics hardware. Proceedings of ACM SIGGRAPH 1999 (1999) 277–286
5. Murakami, Y., Minami, K., Kawasoe, T., and Ishida, T.: Multi-Agent Simulation for Crisis Management IEEE Workshop on Knowledge Media Networking (KMN'02) Kyoto, JAPAN July 10 - 12, 2002 135
6. Raupp, S., Thalmann D.: Hierarchicall Model For Real Time Simulation of Virtual Human Crowds. IEEE Transactions on Visualization and Computer Graphics, Vol. 7, No.2 (April–June 2001) 152–164
7. Reynolds, C.: Flocks, Herds, and Schools: A Distributed Behavioral Model, Proceedings of ACM SIGGRAPH'87 (1987) 25–34
8. Reynolds, C., Steering Behaviors For Autonomous Characters, Proceedings of Game Developers Conference (1999)
9. Tecchia, F., Loscos, C. and Chrysanthou, Y.: Image-based Crowd Rendering. IEEE Computer Graphics and Applications, Vol. 22 No. 2 (March–April 2002)
10. Tecchia, F., Loscos, C., Conroy R.and Chrysanthou, Y.: Agent Behaviour Simulator (ABS):A Platform for Urban Behaviour Development GTEC'2001, Hong Kong, (2001)
11. Tecchia, F.: Real-Time Rendering of Densely Populated Urban Environments. (2000)
12. Wand, M., Strasser, W.: Multi-Resolution Rendering of Complex Animated Scenes. In: Computer Graphics Forum, Vol. 21 No. 3 (2002) 483–491