Sketch-based Interface for Crowd Animation

Masaki Oshita[1], Yusuke Ogiwara[1]

[1] Kyushu Institute of Technology
680-4 Kawazu, Iizuka, Fukuoka, 820-8502, Japan
oshita@ces.kyutech.ac.jp ogiwara@cg.ces.kyutech.ac.jp

Abstract. In this paper, we propose a novel interface for controlling crowd animation. Crowd animation is widely used in movie production and computer games. However, to make an intended crowd animation, a lot of agent model parameters have to be tuned through trial and error. Our method estimates crowd parameters based on a few example paths given by a user through a sketch-based interface. The parameters include guiding paths, moving speed, distance between agents, and adjustments of the distance (crowd regularity). Based on the computed parameters, a crowd animation is generated using an agent model. We demonstrate the effectiveness of our method through our experiments.

Keywords: Crowd Animation, Sketch-based Interface, Agent Model.

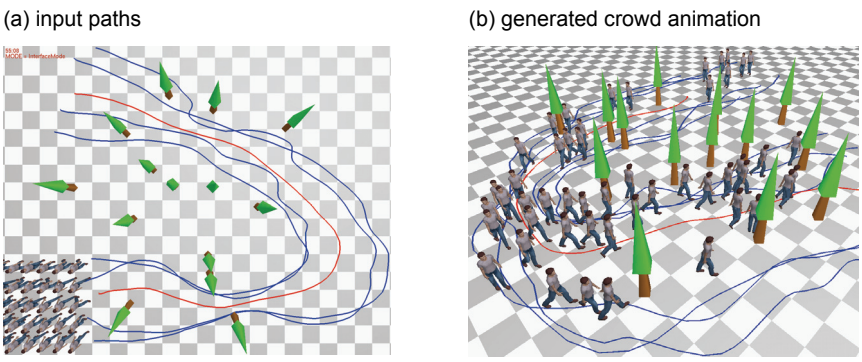


Fig. 1. Overview of our system. The parameters for crowd control are estimated from a few example paths that are given by a user (a). Based on the parameters and an agent model, a crowd animation is generated (b). The user can make a crowd animation interactively.

1 Introduction

Recently, crowd animation has been widely used for movie production and computer games. Using crowd animation techniques, a user can create an animation in which a large number of characters move around and interact with each other for scenes such as soldiers in combat, pedestrians on a street, stadium audiences, etc. Crowd

animation can be created using commercial or in-house animation systems. For the majority of systems, an agent model [1][2] is used for controlling characters (agents). An agent model controls individual agents based on application dependent rules such as following a leader, keeping a distance between other agents, avoiding obstacles, etc. As a result of the movements of individual agents, a natural crowd behavior is generated. However, to make an animation, a user has to tune various parameters for the agent model such as initial positions, paths, moving speed, distance, etc. These parameters are usually tuned through a graphical user interface which the animation system provides. Because a small parameter change can cause a drastic change in the animation, and users cannot predict the result when they change a parameter, the parameter tuning process can be very arduous work.

To solve this problem, we propose a crowd animation system in which users can specify the parameters intuitively thorough a sketch-based interface using a pen-tablet or mouse device (Figure 1). A user of our system is asked to give a few example paths for the agents to roughly follow. The system automatically estimates parameters that satisfy given paths. Based on the given paths, parameters such as guiding path, moving speed, standard distance between agents, and distance deviation are computed. By using the estimated parameters and the agent model, a crowd animation is generated. Using our system, a user can make crowd animations interactively by giving just a few example paths.

In this paper, we describe our interface design (Section 3), the parameters that our method uses and how to compute them (Section 4), and our agent model (Section 5). We also show experimental results and the effectiveness of our system (Section 6).

2 Related work

Crowd animation is an important topic in the computer graphics field. There has been a lot of research on crowd animation. As explained in the previous section, agent models have been widely used in research [1][2]. More advanced crowd models such as path planning, collision avoidance and vision-based control have also been developed.

Currently, an animator can make a crowd animation using commercial animation systems such as Maya, Max, or Softimage. However, they have to describe scripts for a crowd model and for the user interface. This is a very tedious task. Recently, an animation system specializing in crowd animation, MASSIVE [3], has been widely used for movie production. MASSIVE employs an agent model that uses sophisticated decision rule trees. A user can edit the decision rule trees using a graphical interface. However, it is difficult to construct or modify these.

There has been research on user interfaces for controlling animated crowds. Ulicny et al. [4] proposed a brush interface with which a user can specify parameters for individual agents in a crowd by painting them. However, the parameter values that are applied to these agents have to be set with a conventional interface. Moreover, the user cannot specify crowd movements such as individual paths, speed, distance between agents, regularity of crowd, etc., because those parameters not only depend on agents but also depend on positions in the scene. Sung et al. [5] proposed a crowd

animation system in which a user can specify agent behaviors on each region of a scene. However, the user cannot specify parameters and behaviors which depend on the agents themselves. Moreover, the user still has to specify rules and parameters for each region manually.

There is also research into automatic estimation of crowd parameters. Several research groups [6][7][8] proposed methods for making a crowd animation based on movies of real pedestrians shot from above. Their methods estimate the moving velocity and direction of each point in the scene. Agents are controlled based on the computed vector field. Their method can make a natural crowd animation based on real pedestrians. However, a user has to prepare a movie of pedestrians, which is difficult to shoot. Jin et al. [9] proposed a crowd animation system in which a user can specify the moving velocity and direction on any point in the scene. The system generates a vector field for crowd control. These methods can control crowd movements such as paths and widths of crowds. However, they cannot control detailed agent movements such as distance between agents and regularity of agents, both of which our method can control. In addition, our method determines such parameters based on the example paths given by the user.

3 Interface Design

Our system is designed so that the user's intention is reflected in the generated animation. In this section, we describe the interface design and explain how it works. The implementation is explained in the following sections.

A user of our system is asked to input one primary path and a few additional paths as example paths for the crowd. The primary path (red path in Figure 1 (a)) specifies the rough path of the overall crowd. The additional paths (blue paths in Figure 1 (a)) are for specifying the crowd behaviors or paths of individual agents in more detail.

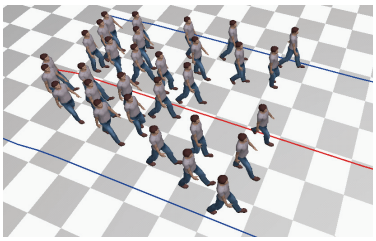
First, a user can specify the initial distribution of agents by drawing an outline of the region where the user wants to put the agents. The user then inputs one primary path and any additional paths. The system generates a new crowd animation when the user inputs new paths. The user can also undo and redo the paths. Therefore, the user can create a crowd animation in an interactive manner.

The general path and moving speed are controlled by the primary path. The crowd follows the primary path, and the moving speed of agents is based on the input speed. The agents walk slowly where the user inputs the path slowly.

The distances between agents, regularity of distances, and individual paths are controlled by the additional paths. First, the distances between agents varies based on the width of all given paths. As shown in Figure 2, when the width of the paths is wide, the agents spread out, and when the width is narrow, the agents get closer to each other. We designed this interface because we can assume that users usually intend to spread the agents out when they specify wide paths.

Second, the regularity varies based on the smoothness of the additional paths. As shown in Figure 3, where the paths are rough (zigzag), the agents move irregularly, which means that the distances between agents are not uniform. On the other hand, where the paths are smooth, the agents keep the distances between each other uniform.

(a) spread crowd



(b) close crowd

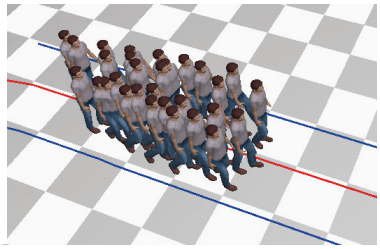
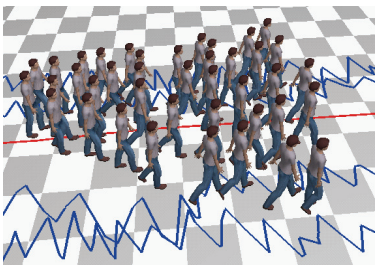


Fig. 2. Control of the distance between agents.

(a) irregular crowd



(b) regular crowd

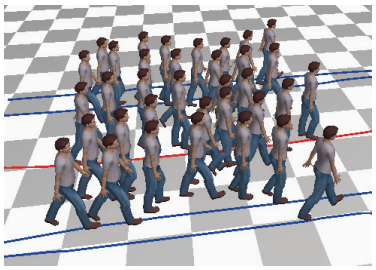
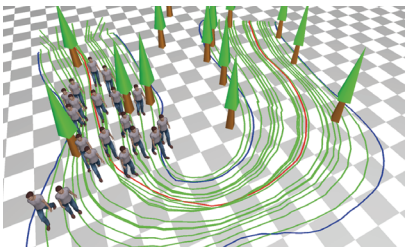


Fig. 3. Control of the regularity of agents.

(a) agent paths of two additional paths



(b) agent paths of four additional paths

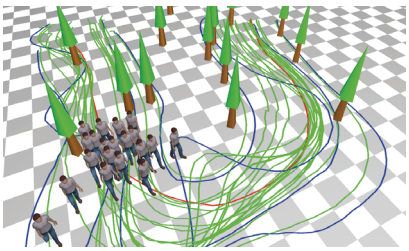


Fig. 4. Control of paths of agents.

We designed this interface because users intend to move agents irregularly when they specify non-smooth paths.

Finally, the paths of individual agents are based on the additional paths. The path of each agent is affected by the additional paths that are close to the initial position of the agent. The more additional paths the user gives, the more they can control the movements of the agents, as shown in Figure 4. The green paths in Figure 4 represent the guiding paths generated for each agent. This is explained further in the next section.

Usually a user inputs at least one or two additional paths to specify the distance and regularity. When users want to specify agents' paths in more detail, they add more additional paths.

4 Crowd Parameter Estimation

This section describes our method for computing crowd parameters. We explain the crowd parameters that are used in our method. We then explain how to compute them.

4.1 Crowd Parameter

We use an agent model [1][2] for controlling crowds. There has been research on many possible types of parameters. In our agent model, we use the crowd parameters that are shown in Table 1. All crowd parameters vary according to points along the guiding paths, which means that the crowd movements vary during the animation. We represent the varying parameters as a series of variables. The subscript $k = 1...K$ of all parameters represents the index number of points where K is the total number of points. Each agent is controlled by the parameters of the closest point. In addition, some parameters vary depending on the agents, so each agent has a different value. The subscript $j = 1...M$ of the parameters represents the index number of the agents where M is the total number of agents.

Table 1. Crowd parameters.

Name	Variables	Uniform or Individual
Guiding path	$\mathbf{P}_j = \{\mathbf{p}_{j,1}, \cdots, \mathbf{p}_{j,k}, \cdots, \mathbf{p}_{j,K}\}$	Individual
Moving speed	v_k	Uniform
Standard distance to other agents	s_k	Uniform
Adjustment of distance to other agents	$a_{j,k}$	Individual

4.2 Input Path

As explained in Section 3, the crowd parameters are computed from one primary path and a number of additional paths given by the user. The paths that the user draws on the screen are projected on the ground to compute the positions in the scene coordinates. $\mathbf{Q}_i = \{\mathbf{q}_{i,1}, \cdots, \mathbf{q}_{i,k}, \cdots, \mathbf{q}_{i,K}\}$ represents the points of the paths. The subscript $i = 1...N$ represents the index number of each path where $i = 1$ is the primary path and $i = 2...N$ are the additional paths. The subscript $k = 1...K$ represents the index number of the sampling points. Sampling points of all additional paths are normalized based on the primary path so that the points of all paths are aligned with the same number. The primary path also has $T_i = \{t_{i,1}, \cdots, t_{i,k}, \cdots, t_{i,K}\}$, times when each point is given (drawn) by the user.

For computing crowd parameters, the system computes smoothed additional paths first. $\bar{\mathbf{Q}}_i = \{\bar{\mathbf{q}}_{i,1}, \cdots, \bar{\mathbf{q}}_{i,k}, \cdots, \bar{\mathbf{q}}_{i,K}\}$ represents smoothed additional paths computed from $\mathbf{Q}_i$. The reasons that we use smoothed paths are explained later. There are a

number of methods for smoothing a path. We use a spring model in our implementation. On each iterative step, each sampling point is moved slightly to make the angle at the point smaller, and to maintain the distance to the original path. By repeating this step until the points converge, a smoothed path is computed.

The initial positions of all agents $\mathbf{o}_j$ and the number of agents M are computed from the outline of the region that the user draws first. The system distributes agents in the region with a fixed distance.

The crowd parameters are computed from $\mathbf{o}_j$, $\mathbf{Q}_i$, T_i, and $\bar{\mathbf{Q}}_i$. In the following section, we describe how we compute each parameter.

4.3 Guiding Path

The guiding path $\mathbf{P}_j = \left\{\mathbf{p}_{j,1}, \cdots, \mathbf{p}_{j,k}, \cdots, \mathbf{p}_{j,K}\right\}$ controls the moving path of each agent. Each agent moves so that it follows the guiding path. Note that each agent may not follow the guiding path exactly, since it is also affected by other factors such as keeping a distance with other agents, and avoiding obstacles.

The guiding path for each agent is computed by blending given paths, which are the primary path and all smoothed additional paths. We use the smoothed path instead of original paths, because a user may input a zigzag path intending irregular crowd movements but not intending a zigzag movement. Smoothed guiding paths are suitable for most animations. The blending weights of all paths for each agent are computed based on the distances between the paths and the initial position of the agent. Individual agents are affected by paths close to them.

Computation of a guiding path is as follows. First, blending weights of all smoothed paths for each agent is computed by

$$w_{i,j} = \frac{1}{{l_{i,j}}^2}, \quad \bar{w}_{i,j} = \frac{w_{i,j}}{\sum_{i=1} w_{i,j}} \quad . \tag{1}$$

where $l_{i,j}$ is the distance between the *j*-th path $\bar{\mathbf{Q}}_j$ and the initial position of the *i*-th agent $\mathbf{o}_i$. We decided to use an inverse quadratic function for computing weights. The blending weights $\bar{w}_{i,j}$ are computed by normalizing the weights. Finally, each point of the guiding path $\mathbf{p}_{j,k}$ is computed by blending the positions of all paths with their matching blending weights $\bar{w}_{i,j}$. This process is repeated for all agents j.

4.4 Moving Speed

The moving speed v_k affects the crowd's maximum speed. It is simply computed from the user's drawing speed of the primary path by using the equation,

$$v_k = \frac{\left|\mathbf{q}_{1,k} - \mathbf{q}_{1,k-1}\right|}{t_{1,k} - t_{1,k-1}} \quad . \tag{2}$$

4.5 Standard Distance Between Agents

The standard distance between agents s_k is for controlling the distance between agents. This is a common parameter for all agents. The distance of each agent is altered by the adjustment of distance $a_{j,k}$ which is explained next.

The standard distance s_k is computed from the width of all paths as explained in Section 3. First, the widths of the all paths w_k are computed for each point $k = 1...K$ on the primary path $\mathbf{Q}_1$. w_k is computed by finding the farthest pair from $\left\{\overline{\mathbf{q}}_{i,k}\right\}\left(i = 1,\cdots,K\right)$ and by computing the distance between that pair. The standard distance s_k is then computed in proportion to w_k as follows,

$$s_k = s_{scale} \log\left(w_k + 1\right) + s_{min} \quad . \tag{3}$$

where s_{min}, s_{scale} are the minimum distance and distance scaling, respectively. We use a logarithmic function to avoid s_k becoming too big. s_{min} is set so that it prevents collisions between agents. $s_{min} = 2m$ is used in our experiments.

4.6 Adjustment of Distance Between Agents

The adjustment of the distance $a_{j,k}$ is an agent specific parameter to vary the distances between agents. This parameter is based on the roughness/smoothness of the additional paths as explained in Section 3. We compute the adjustment from the distance between the original path and the smoothed path. If a given path is smooth, this value should be close to zero. Otherwise, it becomes large.

Computation of adjustment of distance $a_{j,k}$ is as follows. First, at each *k*-th point of each *j*-th additional path, the distance $d_{j,k}$ between the original path $\mathbf{Q}_i$ and smoothed path $\overline{\mathbf{Q}}_i$ is computed. Since the distance varies from zero to the maximum distance on a zigzag path, to compute the maximum distance at each point, we find a maximum distance within a path window α. α is computed so that the path window becomes about two meters. The distance $D_{j,k}$ is computed as follows,

$$D_{j,k} = \max\left(\left|\mathbf{q}_{i,k-\alpha} - \overline{\mathbf{q}}_{i,k-\alpha}\right|, \cdots, \left|\mathbf{q}_{i,k} - \overline{\mathbf{q}}_{i,k}\right| \cdots, \left|\mathbf{q}_{i,k+\alpha} - \overline{\mathbf{q}}_{i,k+\alpha}\right|\right) \quad . \tag{4}$$

Second, the magnitude of adjustment at each *k*-th point A_k is computed by taking the average of the distance between the original and smooth paths. Finally, the

adjustment of the distance of the *j*-th agent at the *k*-th point $a_{j,k}$ is computed. To make variations in response to the magnitude of A_k, a random value from 0.5 to 1.0 is multiplied as follows where $random(a,b)$ is a random function.

$$A_k = \sum_{i=2}^{N} D_{i,k} \Big/ N \ , \quad a_{j,k} = random(0.5,1.0)\, A_k \quad . \tag{5}$$

5 Crowd Animation

This section describes how our system generates a crowd animation using an agent model. There are many variations of agent models [1][2]. We decided to introduce the following three rules as controlling rules.

- Each agent follows a specific guiding path.
- Each agent keeps a specific distance from other agents.
- Each agent avoids obstacles.

On some agent models, a rule for making agents follow a leader is introduced. However, instead of such a rule, we decided to use the first rule, because using guiding paths can result in similar movements and a user can control the agents' paths more specifically by manipulating guiding paths.

For each agent, three kinds of forces: $\mathbf{f}^{path}{}_j$, $\mathbf{f}^{distance}{}_j$, and $\mathbf{f}^{avoid}{}_j$, are computed based on the above three rules. $\mathbf{f}^{path}{}_j$ is the force for making an agent follow a guiding path. This is computed so that the agent moves toward the next sampling point of the path $\mathbf{p}_{j,n}$. $\mathbf{f}^{distance}{}_j$ is the force for making an agent keep their distance from other agents. The distance constraint between two agents is computed by taking an average of the distance parameters of the two agents (j -th and j' -th agents),

$$d_{j,j'} = s_{j,k} + \frac{e_{j,k} + e_{j',k}}{2} \quad . \tag{6}$$

The force for the *j*-th agent $\mathbf{f}^{distance}{}_j$ is computed from all nearby agents as follows.

$$\mathbf{f}^{distance}{}_j = \sum_{j'=1}^{M} \begin{cases} k_{distance}\left(\mathbf{x}_{j'} - \mathbf{x}_j\right)\left(1 - \dfrac{d_{j,j'}}{\left|\mathbf{x}_{j'} - \mathbf{x}_j\right|}\right) & if \left|\mathbf{x}_{j'} - \mathbf{x}_j\right| < \beta d_{j,j'} \\ 0 & if \left|\mathbf{x}_{j'} - \mathbf{x}_j\right| \geq \beta d_{j,j'} \end{cases} \quad . \tag{7}$$

where $k_{distance}$ is a constant magnitude of force. β defines the range of nearby agents. On our implementation, we use $\beta = 1.5$. $\mathbf{f}^{avoid}{}_j$ is the force for making an

agent avoid obstacles. This force is computed using a similar equation to equation (7). Finally, the force on each agent $\mathbf{f}_j$ is computed by summing up all the forces.

On our agent model, an acceleration $\mathbf{a}_j$ is computed from the force $\mathbf{f}_j$. The position $\mathbf{x}_j$ and velocity $\mathbf{v}_j$ is updated based on the acceleration $\mathbf{a}_j$. The velocity is restricted by the maximum moving speed v_k.

$$\mathbf{a}_j = \frac{\mathbf{f}_j}{\Delta t},\ \mathbf{v}_j{}' = \mathbf{a}_j \Delta t + \mathbf{v}_j,\ \mathbf{v}_j{}' = v_k \frac{\mathbf{v}_j{}'}{\left|\mathbf{v}_j{}'\right|}\ \ if\ \left|\mathbf{v}_j{}'\right| > v_k,\ \mathbf{x}_j{}' = \mathbf{v}_j{}' \Delta t + \mathbf{x}_j \qquad (8)$$

where Δt is the time between the current frame and the previous frame. Orientation of the agent $\mathbf{r}_j$ is computed based on the velocity $\mathbf{v}_j{}'$

To render a character on the scene, we need a posture of the character in addition to position $\mathbf{x}_j$ and orientation $\mathbf{r}_j$. In our implementation, we simply use a walking motion clip to represent the posture of each agent on each frame.

6 Experimental Results and Discussion

We have implemented the proposed method using C++ and OpenGL. The system generates crowd animation interactively. We have done preliminary experiments to show the effectiveness of our system. As well as the sketch-based interface, we also developed a conventional interface for our system so that the user could specify the crowd parameters of our methods directly with the mouse and keyboard.

To evaluate the effectiveness of our interface, we measured the times for creating a crowd animation. Four undergraduate and graduate students participated in the experiments. The subjects were asked to create a crowd animation that matched an example movie using both our sketch-based interface and the conventional interface. The subjects were asked to stop when the generated animation looked close enough to the example movie according to the judge, and the time was measured. The example crowd animations were created using our system in advance. On our experiments we used two example animations: simple and complex. They are shown in the accompanying movie.

Table 2. Average time for making a crowd animation (minutes : seconds).

Interface	Simple crowd	Complex crowd
Proposed interface	0:51	1:50
Conventional interface	10:00	30:30

Table 2 shows the results of our experiments. According to the results, our interface is more effective than the conventional interface. Moreover, the more complex the crowd movements, the more effective our interface is.

Currently our method is specialized for crowd animations where agents travel as a group. In practical crowd animation, the agents perform various actions and interact

with each other (e.g., combat, conversation, reaction of audience, etc.). These kinds of actions can be realized by user commands or behavioral models. User commands can be given through a conventional interface or possibly a gesture-based interface. To use behavioral models, complex rules and parameters have to be tuned by the user. This would be difficult through a sketch-based interface. We developed the sketch-based system explained in this paper, because we thought that a sketch-based interface would be the most suitable for controlling a traveling group. As a future work, we want to extend our system so that a user can give action commands to agents through a sketch-based interface. A sketch-based interface for describing behavioral rules is also an interesting future work.

7 Conclusion

In this paper, we have proposed an intuitive sketch-based interface for controlling crowd movements. We have implemented our method and demonstrated the effectiveness of our interface. Currently, the main target application of our method is for animation production. However, this technique can be used for various applications such as computer games, communications across metaverses, behavioral simulations and so on, because even non-professional users can use our interface very easily. Our future work includes improvement of the agent model, addition of crowd parameters and estimations, support for action commands through the sketch-based interface, and addition of group parameters for animating multiple groups.

References

1. Creg W. Reynolds: Flocks, Herds, and Schools: A Distributed Behavioral Model. In: SIGGRAPH'87, pp.25-34 (1987)
2. S.R. Musse, D. Thalmann: Hierarchical Model for Real-Time Simulation of Virtual Human Crowds. In: IEEE Trans. Visualization and Computer Graphics, vol. 7, no. 2, pp. 152-164 (2001).
3. MASSIVE, http://www.massivesoftware.com/
4. B. Ulicny, P. de Heras Ciechomski, D. Thalmann: Crowdbrush: Interactive Authoring of Real-time Crowd Scenes. In: ACM SIGGRAPH Symposium on Computer Animation 2004, pp. 243-252(2004)
5. Mankyu Sung, Michael Gleicher, Stephen Chenney: Scalable Behaviors for Crowd Simulation. In: Computer Graphics Forum, vol. 23, no 3, pp. 519-528 (2004)
6. N. Courty, T. Corpetti: Crowd Motion Capture. In: Computer Animation and Virtual Worlds, Volume 18, Issue 4-5, pp. 361-370 (2007)
7. Lee, K. H., Choi, M. G., Hong, Q., Lee, J.: Group behavior from video: a data-driven approach to crowd simulation. In: Proc. of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 109-118 (2007)
8. Alon Lerner, Yiorgos Chrysanthou, Dani Lischi: Crowds by example. In: Computer Graphics Forum, vol 26, no 3, pp. 655-664 (2007)
9. Xiaogang Jin, Jiayi Xu, Charlie C.L. Wang, Shengsheng Huang, Jun Zhang: Interactive Control of Large-Crowd Navigation in Virtual Environments Using Vector Fields. In: IEEE Computer Graphics and Applications, vol. 28, issue 6, pp.37-46, 2008.