Flocking Messengers

Prof. T. Unemi, BEng, MEng, DEng.
Department of Information Systems Science, Soka University, Tokyo, Japan.
e-mail: unemi@iss.soka.ac.jp

Dr. D. Bisig, BSc, MSc, PhD.
Artificial Intelligence Laboratory, University of Zurich, Switzerland.
e-mail: dbisig@ifi.unizh.ch

Abstract

Flocking Messengers is an entirely new type of real-time communication tool, which utilizes a technique from artificial life to allow two remote persons to speak to each other. Messages that travel through a computer network, cross a virtual 3D space that is inhabited by flocking agents. Attracted by the speaker's motions, agents approach and memorize his/her voice. Subsequently, they move towards the opposite space boundary and recite the speakers' message in their own voice. The rules controlling each agent are simple, but the reactions of the entire system are complex and unpredictable since they combine acoustic and visual interaction with collective behaviour. Agents possess their own emotional state, which they occasionally express through their own words. Thereby they guide the visitor's reactions and increase his/her enjoyment of the communication.

1. Introduction

When I was a shy boy at school age, I often experienced difficulty in talking to pretty girls in a face-to-face conversation. I soon noticed how different communication tools, such as a telephone or postcard, affect my capability for communication. These tools create formal protocols and cause different time delays in prompting for a response. We sometimes rely on these tools to tell something, which is difficult to say directly. Nowadays, the Internet has created a wide variety of communication tools such as e-mail, blogs, chat etc. Along with these technical improvements and an increasing popularity of computers, new styles of communication develop.

Flocking Messengers is an entirely new type of real-time communication tool, which utilizes a technique from artificial life to allow two remote persons to speak to each other. The system has been realized on two iMacs (1.83 GHz Intel Core Duo) equipped with built-in cameras and external microphones. One machine runs our application DT1 (version 1.6) in server mode. The other runs the same application in client mode. DT1 simulates the flocking behaviour of agents in a virtual 3D world. In the current version, 128 agents act as messengers. This number can be increased arbitrarily as long as the employed computer is sufficiently powerful.

The motion of users is derived from a sequence of captured video frames and causes the agents to move towards the source of the motion. By this way, users can summon agents in order to tell them their own messages or to listen to the messages of the communication partner. Both the captured video images and the audio messages are transmitted between the two connected computers. The video image obtained on one machine is displayed on the back-wall of the virtual 3D space on the other machine as shown in Figure 1. The audio message is not directly played back but rather rendered as individual agent voices. Agents possess an emotional state, which they can express through their own acoustic messages. Different agents speak at different time delays and in different pitches that create a chorus of voices. This may render the message more difficult to understand but is generally appreciated as funny and enjoyable.

The following sections describe the technical features of the system concerning flocking simulation, visual interaction between agents and human, network communication, audio processing, and visitors' reaction in an experimental exhibition.

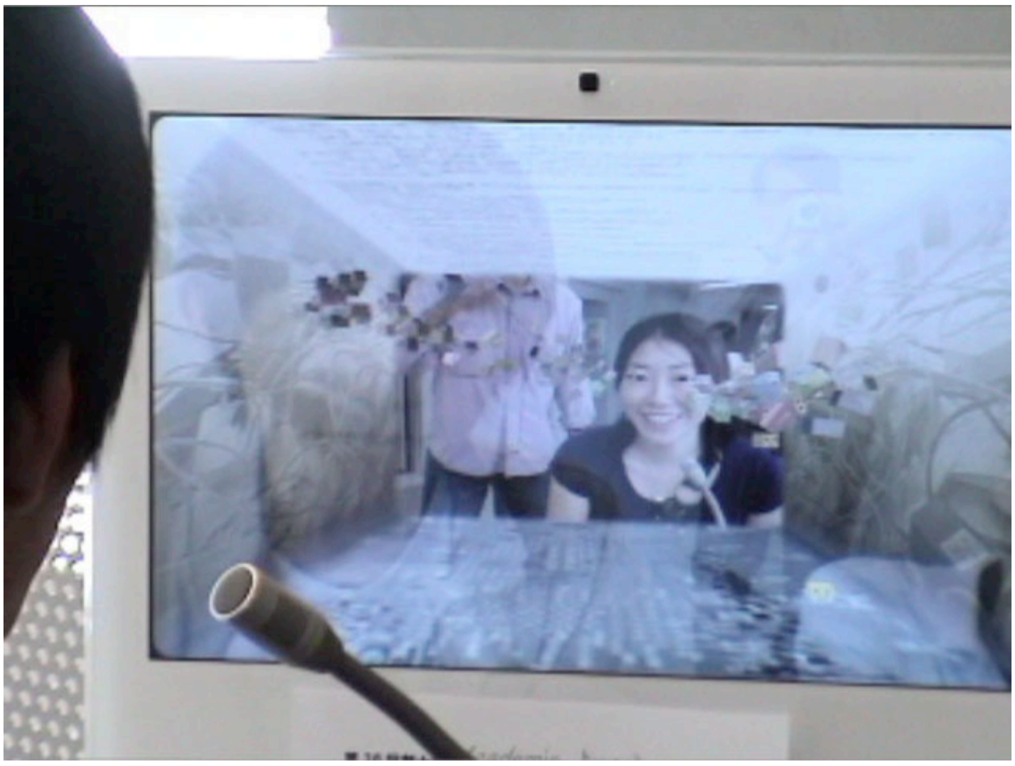

Figure 1. An example view of the screen.

2. Flocking Simulation

The implementation of the flocking simulation and visual interaction methods are largely based on our previous artwork entitled "Flocking Orchestra [1, 2]." This previous artwork is an interactive installation, which allows visitors to play music by conducting flocking agents via gestures.

Each agent is controlled by a set of forces. Some of these forces implement standard boids type of flocking rules [3] whereas other forces result from the user's visual interaction. The flocking forces cause the following behaviours:

1. collision avoidance between agents,
2. velocity alignment with neighbouring agents,
3. flock cohesion by attraction towards the centre of the neighbouring agents, and
4. collision avoidance with the boundaries of the agent world.

The repelling forces for collision avoidance are proportional to the inverse of the square of the distance between the agent and the obstacle. Based on the sum of all the forces that affect an agent, a goal angle and a goal speed are calculated. The agent tries to meet these goal values by modifying its current orientation and velocity within the allowed limitations of its steering angle and acceleration.

To calculate these forces, it is necessary to organize the set of neighbouring agents within an effective distance from each agent. If we used an exhaustive algorithm to check the distance between every possible pair of agents, the computational time complexity would be proportional to square of the number of agents. To reduce this complexity as much as possible, we introduced a method that divides the 3D space into a number of sub-areas to manage the information which agent is located in which sub-area. Since we maintain information about the locations of agents within these sub-areas, we can restrict distance calculations between agents that reside within neighbouring sub-areas.

3. Visual Interaction

The flock's behavioural response to user input is based on the calculation of interaction forces. This implementation is derived from our previous work. The interaction forces lead to the following behaviours:

- movement towards a particular target position on the front plane when user motion is detected, and movement away from the front plane in absence of user motion.

The target position is calculated in the following way:

- a difference image is calculated by summing over the absolute differences of all pixel RGB values between the current and previous capture images, and
- for each agent an individual attractor position is calculated.

This position is derived by multiplying the RGB difference values of all pixels that lie within the neighbourhood circle of an agent with their corresponding *x* and *y* position in the camera image.

The system differs from the previous implementation by the fact that interaction dependent behaviours cause agents to move either towards the front or back plane of the agent world. The attraction force for an agent is calculated by summing the vectors towards both surfaces. The distribution of repulsion forces that push agents away from the surfaces is modified as shown in Figure 2.

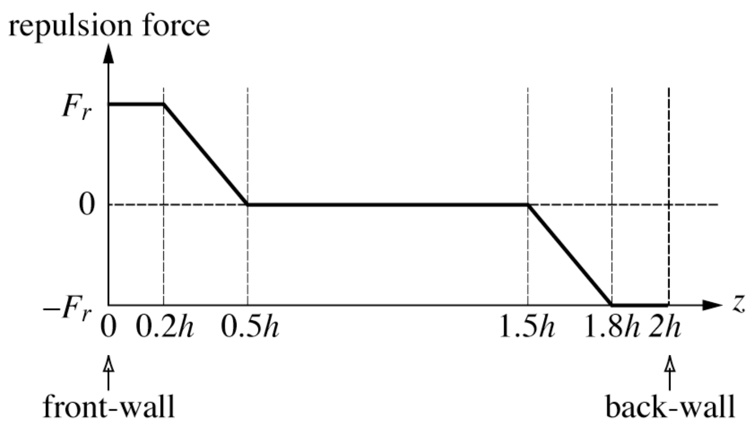

Figure 2. The distribution of repulsion forces to take agents away from the front and back surfaces.

4. Network Communication

We examined two types of methods to combine two machines for computation and communication. In either method, the images captured by the camera and the sounds captured by the microphone are transmitted from one computer to the other through the local network. The two methods differ as to which machine calculates the flocking simulation. One method is to simulate flocking behaviour independently on each machine. The other method is to simulate flocking only on one machine and to send agent position and pose information to the second machine. Prior to rendering the agents, the second machine rotates every agent by 180 degrees.

In the former method, the main part of the software is symmetrical in terms of computation and amount of data to be transmitted. To establish a connection between the two machines, one machine acts as a server and the other acts as a client for the TCP/IP connection. Otherwise, both machines exchange the same type and amount of data. This data consists of image and sound data as well as the 2D motion patterns from visual interaction. Based on the combination of two motion patterns derived by each machine, each flocking simulation calculates the corresponding forces that act on the agents. The resulting behaviours of the agents are sufficiently similar in both simulations to provide a communication between the users.

In the latter method, the flocking simulation is executed only on the server side and the result is sent to the client side. In order to perform the simulation, the server needs to acquire its own and the clients motion patterns. Thus, the client sends its motion pattern to the server. The server on the other hand sends the pose and position information of the agents to the client. This method guarantees that the same agent behaviours are displayed on both machines, as long as the network connection is sufficiently fast. Unfortunately, whenever the network communication deteriorates, the client's rendering of the agents becomes jerky. For this reason, we are currently employing the former method for exhibition setups. It would be preferable to implement a more robust communication method, which is able to deal with network fluctuations in a more robust manner.

For a future installation version, we will evaluate an alternative communication setup, which consists of one server running the simulation and several clients that each renders the visual and acoustic output. Such a setup would allow a communication between of more than two remote sites.

Each machine displays a scaled version of the live video image captured by the other machine on the back-wall of the agent world. In the current implementation, the resolution of the captured image is 640 × 480 pixels, and that of the wallpaper image is 320 × 240 pixels. Each pixel of the captured image contains three times eight bits of data to represent an RGB value. By reducing this amount to three times five bits per RGB value and by employing a loss-less compression, the memory requirements for each frame drop to 64 KB or less. The visual motion pattern possesses a resolution of 160 × 120 pixels. The amount of motion is stored as 8 bit greyscale values. Again, by employing a loss-less compression, these data are reduced to about 4 KB or less per frame. For a frame rate of at least 15 frames per second, the amount of image data that needs to be transmitted equals at most 1 MB per second.

Audio data is captured at a rate of 44.1 kHz and a sample resolution of 16 bits in LPCM (linear pulse code modulation) format. This data is sampled down to 22.05 kHz, which is of sufficient quality for the human voice. The total amount of data for video and audio material sums up 1.4 MB per second. Therefore, 10 Base-T Ethernet or IEEE 802.11b wireless connections provide insufficient speed. On the other hand, 100 Base-TX, IEEE 1394, or IEEE 802.11g wireless connections are capable of handling the required amount of data. All recent iMacs fulfil these requirements.

5. Audio Processing

5.1 Capturing messages

When an agent is close to the front plane of the virtual space, usually because it has been attracted by a visitor's motion, it is ready to listen to a voice input from the microphone. Symmetrically, an agent will start to listen to the voice of a remote visitor when it appears close to the back-wall. An agent starts recording when it detects a sufficiently high sound level. It stops recording as soon as two seconds of silence have passed. If the recording is at shortest one second in duration it is memorized, otherwise it is discarded. Instead of wasting memory by recording sound data for each agent, only the time-stamp of the beginning of the recording and its duration are memorized by each agent. The recording information of the voice from the microphone is used only to control the aural and visual response of the agent, but the information for the voice from the remote machine is used additionally to render audio output.

When the agents are listening to a human voice, they visually straighten their ears and orient their faces towards the visitor even when moving into another direction (see Figure 3). They blink their eyes in random intervals, and close their eyes when the input voice is too loud. After an agent has stopped listening, it turns into the opposite direction and moves away by adding a constant force towards the remote listener and ignoring any interaction based on attraction forces towards the speaker.

For an exhibition setup, directional microphones have to be employed in order to avoid audio feedback loops. For this reason, an external cardioid type of microphone is used instead of the iMacs internal microphone. The usage of an external microphone has the additional benefit of providing a visual cue that a visitor should speak towards it. The iMacs internal microphone might be used in combination with audio output via headphones. Such a setup is suitable for private use or in order to avoid problematic sound situations at an exhibition. We will look into a software-based sound filtering methods that would allow the usage of the iMacs internal microphone and speakers. Such a solution would be beneficial since any iMac owner can employ the system without a need for additional equipment.

5.2 Modification of human voice

As described above, each agent listens to a voice message, moves to the other side, and then speaks the message in its own voice. This modification simply consists of shifting the original audio material to a higher pitch. The easiest way to lift a sound's pitch consists of increasing its playback speed. Unfortunately, the corresponding shortening in the sounds duration causes the message to become difficult to understand. To maintain the sounds original duration, our audio processing algorithm duplicates a cycle of the waveform to fill in the gaps in the higher pitched sound. The computational cost this method is too expensive to calculate individual voices for each agent. Instead, we calculate and store only three frequency shifts. These shifts correspond to a frequency ratio of 4/3, 5/3 and 2. These ratios represent a triad of a musical chord in the major scale. Accordingly, the resulting chorus of voices is not only cheap to calculate but exhibits harmonic relationships.

5.3 Speaking messages

The voices spoken by the agents are created by extracting the corresponding parts from the buffers containing the frequency shifted audio inputs and by mixing all voices together into the final sounds output from the loud speaker. Based on our preliminary experiments, we observed that the content of a voice message becomes very difficult to understand if more than five agents are simultaneously speaking at different time delays. To reduce the number of speaking agents, we implemented an agent priority queue. In the current implementation, the first five agents that arrive at the frontal plane of the agent world contribute to the audio mixing. Within this group of five agents, the amplitude of every successive agent's audio signal is multiplied with a factor of 2/3. The audio outputs of all other agents are ignored. The resulting audio output still sounds like an echo, but the message's content remains clearly understandable. Figure 4 illustrates the entire process from capturing a visitor's voice at machine A to generating the sound played by the loud speaker connected to machine B. The voices captured at machine B are processed symmetrically to generate a sound output at machine A.

When an agent is speaking, it moves its mouth. The size of mouth is adjusted to the loudness of the voice. A cartoon type of speech bubble appears above the agent as shown in figure 5. These bubbles aid the visitors to recognize the agents that are speaking. The panning of the audio output depends on the horizontal distribution of the speaking agents.

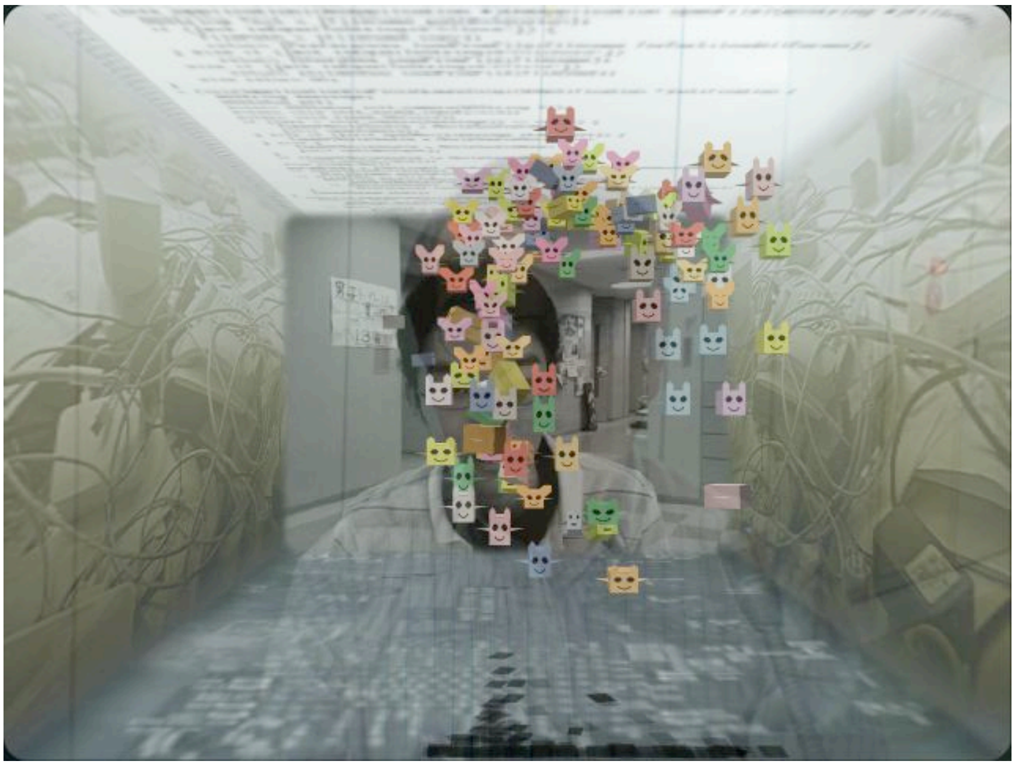

Figure 3. Agents listening to the voice. They straighten their ears and blink their eyes.

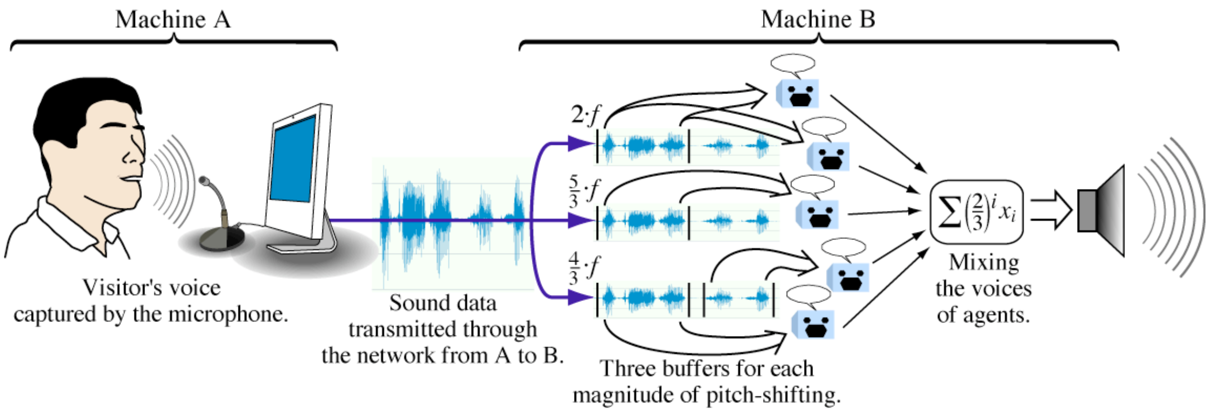
Machine A
Machine B
Visitor's voice captured by the microphone.
Sound data transmitted through the network from A to B.
$2\cdot f$
$\frac{5}{3}\cdot f$
$\frac{4}{3}\cdot f$
Three buffers for each magnitude of pitch-shifting.
$\sum\left(\frac{2}{3}\right)^i x_i$
Mixing the voices of agents.

Figure 4. The entire process for audio data.

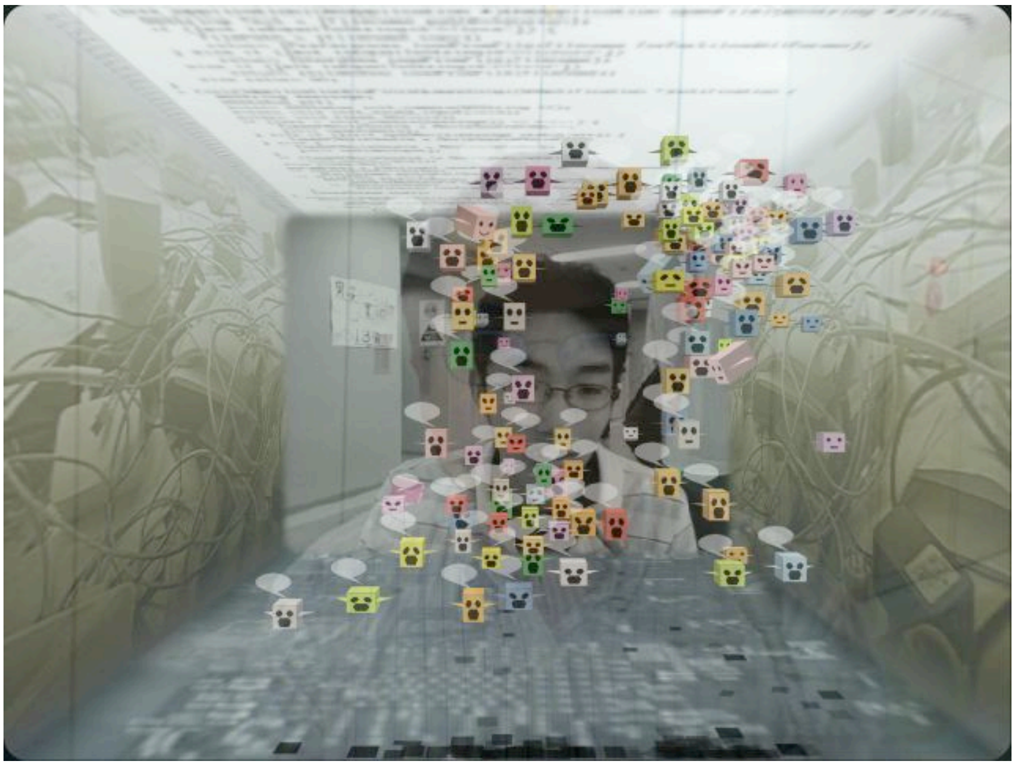

Figure 5. Agents speaking messages. They move their mouths and the bubble signs are drawn above them.

5.4 Words of agents

When we performed our first experiments with an early version of the system, we noticed that some visitors did not say anything but simply enjoyed conducting flock. We therefore concluded that it was necessary to embed some functionality to invite visitors to speak something into the microphone. We came up with a solution that causes agents to speak some words in order to motivate and prompt the visitors to say something even if they don't know how the system works. The generation of agent words is based on the following rules. If no motion is detected for longer than twenty seconds, agents become bored and say things like "boring" or "nobody there?" Once they detect a motion, they greet the visitor with statements like "hi" or "how are you?" If motion is sustained but no acoustic input is detected, agents react by saying, "say something" or "message please." As soon as agents detect a sufficiently high sound level, they start listening. The agents record any sound input that lasts for at least one second and is followed by a pause of at least two seconds. They acknowledge a successfully recorded message by saying "OK" or "I got it" and subsequently start to move towards the opposite boundary of the virtual 3D space. This opposite boundary corresponds to the front wall of the agent world on the second computer station. Once agents reach this boundary, they announce the presence of a message by saying "you got a message" or "please listen." Subsequently, they speak the message in their own voice and finally add statements such as "that's all" or "that's it."

These words stem from recordings of a human's voice. This voice has been modified by shifting its frequency by factor two. The words are stored in audio files, which form part of the application bundle. When the system renders the output sound, the words are loaded from the audio files, and are modified again by pitch shifting. This time, the shifting ratio is specific for each agent and varies uniformly from 0.8 to 1.3. The final pitch of the words is therefore higher by a factor of 1.6 to 2.6 than the original human voice.

6. Visitors' Reaction

We organized two experimental exhibitions at Soka University. The first one was held in the Open Campus in August, and the other one was at the Campus Festival in October 2006. Both of the events gathered thousands of people and more than one hundred persons have experienced our installation during each event. In the first experiment, the software was still at an early stage and the agents did not say their own words. This feature was present in the newer software version, which was shown during the second exhibition. This time, the experiment clearly showed that the agents' words are effective to invite more visitors and promote them to speak into the microphone. During both exhibitions, a short instruction was provided that helped visitors to understand what was going on and how they could enjoy the system in their own style.

One typical visitor reaction was expressed a group of young boys ranging in age from four to ten years. They were often moving their arms to examine the effect on the agents' movements. They grasped the microphone very close to their mouths and shouted meaningless words. Older visitors often exhibited a different type of reaction. They told us that the agents are very cute and seemed to enjoy listening to the agents' voices rather than speaking themselves with the remote person. Some visitors didn't know how to end a conversation. When we are talking on the phone, it is obvious to say "goodbye" to stop the conversation. But for this new communication tool, there doesn't yet exist a common rule to end a session.

7. Concluding Remarks

So far, the Flocking Messengers project has succeeded in surprising many visitors. Some of these visitors told us that they encountered a magical experience that they had never seen before. One of the essential aspects of the system is its complexity. The behaviour of the system is unpredictable because it combines a flocking simulation with acoustic and visual interaction. The agents and visitors influence each via their movements and voices. Another important aspect is the system's real-time response to a large amount of input data from its physical environment. The default input channels of standard personal computers used to be restricted to the keyboard and mouse. Normally, when working on a PC, a user always completely knows what data he/she had input into the computer. Any unintended input could clearly be attributed to a user mistake. However, the input patterns from a camera and a microphone are too rich to be completely controlled. It is difficult to realize the same input data again, and some data are often input regardless of the operator's intention. This type of richness in input data provides the computer with a source of complexity.

The system works with MacOS X 10.3.9 or later running on a PowerPC G4, G5 or Intel CPU. The source code was written in Objective-C and C by using the Xcode programming environment. The compiled application is available on the Internet by accessing the following URL:

http://www.intlab.soka.ac.jp/~unemi/DT1/

In it's current version, network communication is based on a simple BSD socket library. This leads to the drawback that a user has to manually enter the IP address of the remote machine. A more user-friendly approach would employ a plug-and-play type of communication service such as Apple's Bonjour. We would like to distribute our software to as many people as possible. For this reason, it is very important to make Flocking Messenger very user friendly and simple to use.

A drawback of the current system consists of the fact that it doesn't distinguish between a human voice and other input sounds. Sometimes, the agents started to transmit a message when a user breathed loudly in front of the microphone. In such a situation, the agent says "OK," leaves away toward the back-wall, and makes a noisy sound at the remote side. To avoid such undesirable agent reactions, an algorithm needs to be implemented that discriminates between different types of sounds. For the next software version, we will address this problem.

In summary, each individual technique that we have implemented is very simple and rather straightforward. But the combination of these techniques is very novel and provides unique user experiences. In addition, these techniques needed to be carefully integrated in a complex real time system that works smoothly and reliably. In particular, we needed to make sure that the interactive experience is not obstructed by slow response times. As technology progresses, we are looking forward to realize further interactive installations that provide unique experiences for visitors.

References

[1] Unemi, T. and Bisig, D.: Playing Music by Conducting BOID Agents - A Style of Interaction in the Life with A-Life, Proceedings of A-Life IX, pp. 546-550, 2004.

[2] Unemi, T. and Bisig, D.: Music by Interaction among Two Flocking Species and Human, in T. Innocent ed. Proceedings of the Third International Conference on Generative Systems in Electronic Arts, Melbourne, Australia, pp. 171-179, 2005.

[3] Reynolds, C. W.: Flocks, herds, and schools: A distributed behavioural model, Computer Graphics, 21(4): 25-34, (SIGGRAPH '87 Conference Proceedings) 1987.