Playing Music by Conducting BOID Agents – a Style of Interaction in the Life with A-Life

Tatsuo Unemi[1] and Daniel Bisig[2]
[1]Soka University, 1-236 Tangi-machi, Hachiōji, Tokyo, 192-8577 Japan
[2]University of Zurich, Andreasstrasse 15, CH-8050 Zürich, Switzerland
unemi@iss.soka.ac.jp

Abstract

This paper presents an interactive installation that employs flocking algorithms to produce music and visuals. The user's motions are captured by a video camera and influence the flocks behaviour. Each agent moving in a virtual 3D space controls a MIDI instrument whose playing style depends on the agent's state. In this system, the user acts as a conductor influencing the flock's musical activity. In addition to gestural interaction, the acoustic properties of the system can be modified on the fly by using an intuitive GUI. The acoustical and visual output of the system results from the combination of the flock's and user's behaviour. It therefore creates on the behavioural level a mixing of natural and artificial reality. The system has been designed to run an a variety of different computational configuration ranging from small laptops to exhibition scale installations.

Introduction

Life is beautiful, stimulating, and dramatic.

It is very natural that ALife research keeps inspiring many artists. ALife reveals both on a conceptual and practical level how complex life-like forms can be produced artificially.

In the closing speech in A-Life VI in the year 1998, Chris Langton mentioned that mixing real and artificial life might become an important trend in the future. Of course, such a development raises both technical and ethical issues. At the same time the existence of various robotic and virtual reality projects reveal, that this trend has already started. From an artist's point of view the interaction between artificial agents and humans holds tremendous potential for interactive art. Depending on the complexity and adaptivity of the agent's behavioural response to interaction, interesting relations and impressive experiences might be produced for the viewer. A-Volve (Sommerer and Mignonneau, 1996) utilising evolutionary computation and MIC & MUSE (Tosa and Nakatsu, 1996) tuned by artificial neural networks are typical art works toward this direction.

Boids (Reynolds, 1987) impressively demonstrated how a particular computer simulation can produce complex phenomena from simple mechanisms. This and similar software allows the simulation of various forms of collective behaviour such as a flock of birds, a herd of herbivores, a school of fish etc. Such software has become an important and well established technique in computer animation.

Apart from using flocking algorithms to produce visual effects, some researchers and artists have applied these algorithms to create music. For example the Breve environment (Klein, 2002) served as basis to produce a musical flocking system of evolving agents (Spector and Klein, 2002). In another project, a flock of agents moves through a three dimensional space which is segmented into different acoustical regions (Tang and Shiffman, 2003). Whenever an agent enters a particular region, a predefined musical pattern is rendered audible. In both examples the flock behaviour and the resulting acoustical output is completely autonomous. The system presented in this paper differs fundamentally from this approach in that allows the user's behaviour to influences the flock. A project by Rowe and Singer (Rowe and Singer, 1997) employs the Boids concept for an interactive musical performance. In this system, the acoustical output of a several musicians modifies the behaviour of a flock controlling the visual appearance of words on a projected screen. Contrary to our system, the flock in this project serves as a visualisation of a purely human acoustic performance but doesn't produce any sound itself. In another interesting project, a group of natural and artificial musicians collaborate in a music performance. The artificial musicians are implemented as multi-swarms, moving towards musical targets produced by the participating musicians (Blackwell and Bentley, 2002). This project differs from the one presented in this paper in that the flock acts as a musician rather than a musical instrument. The flock in our system can be regarded as a set of virtual instruments which are conducted by one or several users. It therefore possesses certain similarities to a variety of projects in which performers control the behaviour of a virtual instrument by means of their gestures (Roads, 1996). Our system differs from these approaches in that the flock as a musical instrument possesses a certain amount of autonomy and can only be influenced indirectly by the user's actions. The system therefore mixes the artificial behaviour of a simulated flock with the behaviour of

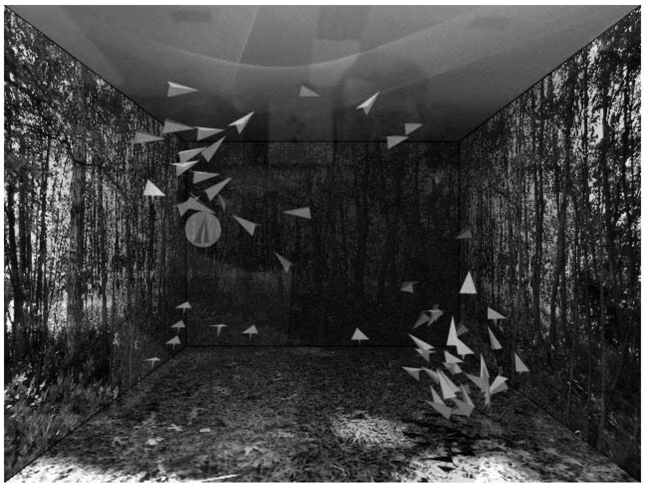


Figure 1: An example of screen-shot.

users. The relationship between the flock and a user mimics the relationship between an orchestra and a conductor. On the other hand, the flock acts completely autonomously when left on its own and the resulting musing serves as a means of catching attention and motivating users to engage into an interaction with the system.

Our system has been designed in a way that it scales very well with available computational power. Users can interact with the system running either their personal laptop or desktop computer or as an exhibition scale installation that employs high performance computer equipment and video beamers. The only particular requirement consists of a video capture device such as a web cam that captures the users gestures.

Throughout the remainder of this paper we describe the implementation of the interactive flocking system, its means of producing music, and different system set-ups for presentation. We end the paper with a short section of concluding remarks.

Interaction with Flocking Agents

Presented in Figure 1 is a screen-shot of the simulation in which depicts 64 agents flocking within a rectangular bounding box. Each agent is controlled by a set of forces. Some of these forces implement standard boids type of flocking rules whereas other forces result from the user's interaction. The flocking forces cause the following behaviour:

1. collision avoidance between agents,
2. velocity alignment with neighbouring agents,
3. flock cohesion by attraction to the neighbouring agents centre, and
4. collision avoidance with the faces of the bounding box.

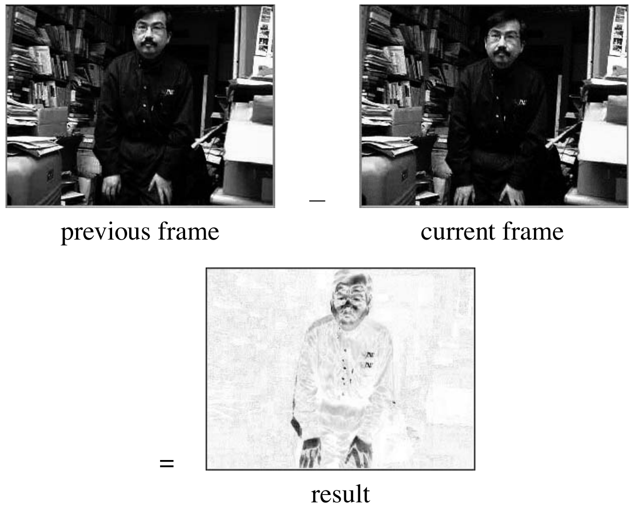


Figure 2: Example of a difference image obtained from two consecutive images.

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

Interaction forces realise the flock's behavioural respond to user input. They cause the following to behaviours;

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

The target position is calculated in the following way (see equation 1 and figure 3 and 4):

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

This position is derived by multiplying 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.

$$u_t = \frac{1}{|C|}\sum_{p\in C}(|r_p^t - r_p^{t-1}| + |g_p^t - g_p^{t-1}| + |b_p^t - b_p^{t-1}|)\, x_p, \quad (1)$$

The position of attraction for a particular agent at time t is denoted by $(u_t, v_t, 0)$. The pixel co-ordinates in the two consecutive frames are denoted by x and y. C is the set of all

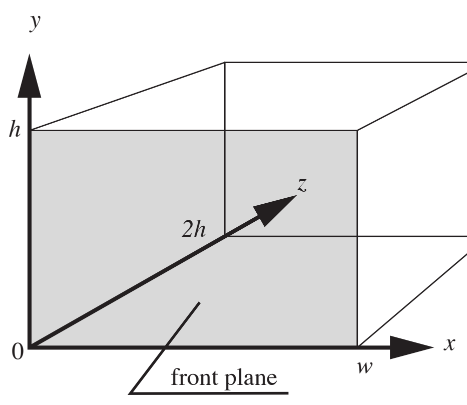


Figure 3: Co-ordinates system in flocking space.

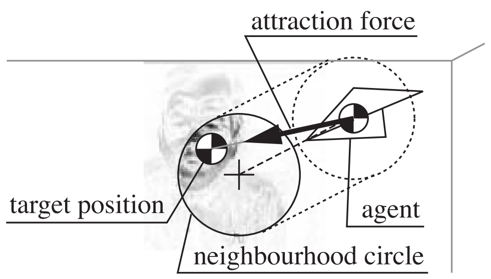


Figure 4: Illustration of the attraction force.

pixels within a fixed distance from the x, y co-ordinates of the agent, $r_p^t, r_p^{t-1}, g_p^t, g_p^{t-1}, b_p^t$, and b_p^{t-1} are the RGB colour values of pixel p of frame t and $t-1$, respectively. x_p is the image co-ordinate of pixel p. v_t is defined in the similar manner.

The second interaction force causes the agents to flock freely in a certain distance from the front plane if no user motion is detected. The strength of this force is at its maximum right at the front plane up to a distance of one fifth of the bounding box depth. At larger distances the force linearly decreases until it reaches a value of zero at half the bounding box depth. In the remainder region within the bounding this force stays zero (see figure 5).

This force causes the agents to flock in the back of the bounding box in the absence of any interaction and to quickly move towards the front when user motion is detected.

To give users an idea of what the agents are seeing and how they respond to his interactive cues the captured image is blended onto the visualisation of the flocking agents.

How Agents Play Instruments

Agents play instruments by sending MIDI (MIDI, 1995) messages representing note pitch and velocity. In order to increase the amount of feedback, each agents responds to user motion not only by moving towards the front but also

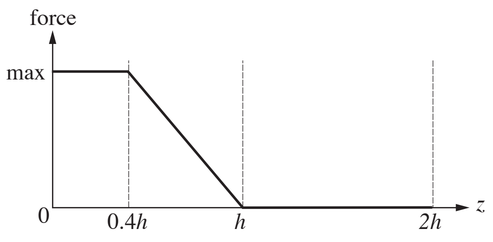
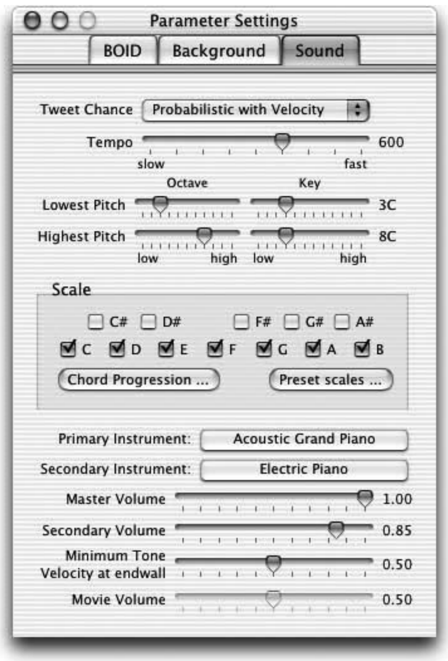


Figure 5: Distribution of the strength of the agent repulsion force. The position $z = 0$ corresponds to the front plane.



Tweet Chance: Probabilistic with Velocity

Tempo (slow – fast): 600

Lowest Pitch (Octave, Key): 3C

Highest Pitch (Octave, Key): 8C

Scale: C# [ ] D# [ ] F# [ ] G# [ ] A# [ ] C [✓] D [✓] E [✓] F [✓] G [✓] A [✓] B [✓]

Primary Instrument: Acoustic Grand Piano

Secondary Instrument: Electric Piano

Master Volume: 1.00

Secondary Volume: 0.85

Minimum Tone Velocity at endwall: 0.50

Movie Volume: 0.50

Figure 6: Graphical user interface for setting up various acoustical parameters.

by changing its MIDI instrument. For this reason, each agent controls a primary and secondary MIDI instrument. It plays its primary instrument whenever it is attracted to the user's interaction. Otherwise it plays its secondary instrument. The secondary instrument is played very quietly whereas the primary instrument produces much louder acoustic output.

Each agent possesses a certain probability of playing a note. The pitch of this note is derived from the agent's y co-ordinate with larger y values corresponding to higher pitches. The balance between left and right speaker output corresponds to the x co-ordinate of the agent. By this way, the stereo position of its acoustic output matches its visual position in flocking space. The loudness of the sound is controlled by the agent's z co-ordinate. The closer the agent is to the front plane of the bounding box the louder the sound it produces.

In a situation as depicted in figure 1 the number of agents exceeds the number of available MIDI channels. For this reason, it is usually impossible for all agents to play their sounds simultaneously. The lowest common denominator are twelve MIDI channels which allows to use this system in conjunction with very simple general MIDI devices. In this system. the primary and secondary instruments occupy six MIDI channels each. In order to handle the MIDI channel issue efficiently, each agent possesses virtual MIDI channels. Each of these virtual MIDI channels corresponds to two real MIDI channels, ones for the left and right stereo position. One pair of real MIDI channels is shared by a maximum of four virtual MIDI channels in case these channels play notes at different pitches. Therefore, a maximum of twelve virtual MIDI channels is available to each primary and secondary instrument. In case an agent would like to play a note but all virtual channels are occupied by other agents the longest playing note is shut off and freed again. An agent gains access to a virtual MIDI channel according to one of several selection mechanisms:

1. each agent is selected in turn according to a fixed order,

2. each agent possesses the same probability of being selected, and

3. the selection probability of an agent is proportional to its movement speed.

The first selection mechanism produces repetitive phrases which are easy to recognise if relatively few agents are producing music. The second mechanisms produces similar results but both the melody and rhythm are arranged more randomly. The third mechanism is particularly good in produced feedback to the user's motion. Slow and quiet sounds are interrupted by fast and loud acoustic output as soon as user motion is detected.

The system can easily be configured to play only a restricted range of pitches and scales. These settings can be accessed via an intuitive GUI (see figure 6). Another GUI allows the user to specify particular sequence of chord progressions (see figure 7). Employing this GUI, the user arranged chord segments represented as labelled rectangles. The duration of each chord segment can be modified by dragging the right edge of its corresponding rectangle. These arrangements can be modified by a variety of standard editing commands such as copy, paste, and cut. New chord segments can be added either by importing an external chord file or by dragging chords from another table containing a number of pre-defined scales. These scales don't represent chords themselves but make it easy to create chords as long as one is aware of the fact that octave intervals produce very harmonic chords whereas other intervals such as the 9th, 11th or 13th produce highly dissonant chords. Clicking the play button automatically adapts the scale and set of pitches required to play the chord sequence. These new values serve as bounding conditions for the system to improvise its flocking music.

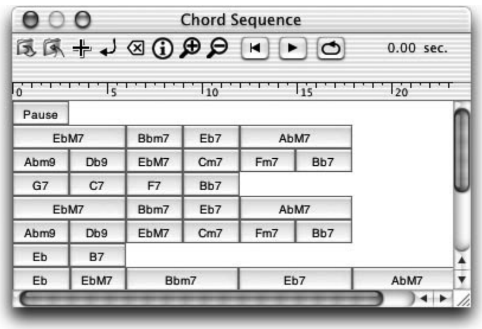
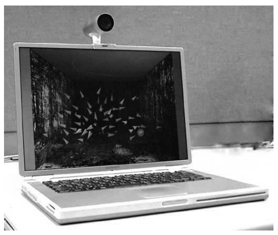


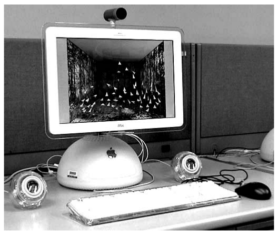
Figure 7: Graphical user interface for setting up the sequence of chord progression.

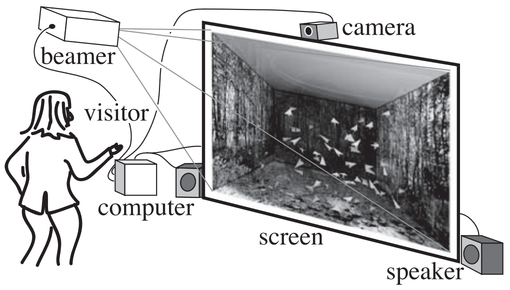
System Configuration

The system is designed for Macintosh computers (MacOS X 10.2 or higher) and equipped with a digital movie camera. The software was written in Objective-C. In order to take advantage of the AltiVec parallel processing capabilities the computer should possess either a G4 or G5 processor. Within these technical limitations the program works well on systems ranging from laptops to very fast desktop computers. For an exhibition context, the system can be expanded with a projector producing a large scale visualisation of the program. In this context two types of users would interact with the system. Visitors can influence the flocking and music production via their movements. On the other hand, operators can control the global acoustic parameters of the system through the graphical user interfaces. These changes can be done while the simulation is running as long as the GUI windows are displayed on a secondary screen. The authors are convinced that this system would be particularly interesting to explore in a dance-performance set-up.

The response time of the system depends on the number of agents and the resolution of the camera and screen image. At VGA resolution, the system should run on a computer system possessing either dual G4 or single G5 processor in order the get an acceptable frame rate (above 10 frames per second). On slower computers, the resolution of the camera image should be reduced to 320×240 pixels whereas the visualisation can still be run at VGA resolution. Such a configuration works well on a single G4 CPU system with a clock rate of 667 MHz. Since our system scales well over a relatively wide range of computing power, it is both suitable for display on exhibitions as well as in personal environments such as at home or at the office. This flexibility should help to bring the experience of an interactive ALife system to a broad audience.


Portable size


Desktop size



Exhibition size

Figure 8: Typical system configurations in various sizes.

Concluding Remarks

In this paper, a musical system for real-time interaction with flocking agents was presented. The user can conduct the musical activity of the flock by means of his/her motions. The interaction takes the form of a collaboration between flock and user rather than an unidirectional exertion of complete control.

This system has been exposed only experimentally to some persons, several students and laboratory staffs, so far. Almost all of them were interested in it, and we observed that some members of a student jazz band were more strongly enjoying their own play using predefined chord progression of standard tunes, relatively than members of chorus group and symphony orchestra. It suggests that the interaction has a type of similar characteristics with improvisational play of jazz music.

In participating in the trend towards hybrid ALife systems, it is important to consider various aspects of interaction between real and artificial life. In our system, the interaction takes place on the behavioural level. This level is highly intuitive for most users. This approach will become even more interesting if the behavioural repertoire of the agents is made much more complex that in this simulation. Other forms of interaction which could be explored in combination with our approach could range from a genetic or chemical level up to the evolutionary and ecological level. It is interesting to think about how the fitness of humans and artificial system mutually influence each other in an interactive simulation. In hybrid systems of collaborate activity issues of competition and symbiosis between natural and artificial system become relevant as well.

According to the principle of ecological balance, it is important to match the behavioural complexity of agents with a complex artificial environment and a complex technical set-up for an interactive installation. Computer Grids and multi-camera systems will become very useful when trying to develop more complex behavioural simulations.

The authors hope that the system described in this paper forms part in creating an ever increasing diversity of interactive experiences with ALife systems.

Acknowledgement

The authors would like to thank Prof. Rolf Pfeifer at University of Zurich and Mr. Akio Yasumoto at Mama Top Co. for their continuing support during this collaboration.

References

Blackwell, T. M. and Bentley, P. J. (2002). Improvised music with swarms. In *Proceedings of the Congress on Evolutionary Computation*, pages 1462–1467.

Klein, J. (2002). Breve: a 3D simulation environment for the simulation of decentralized systems and artificial life. In *Proceedings of A-Life VIII*, Sydney, NSW, Australia.

MIDI (1995). *The Complete MIDI 1.0 Detailed Specification*. Midi Manufactures Association, La Habra, CA.

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

Roads, C. (1996). *The Computer Music Tutorial*. MIT Press.

Rowe, R. and Singer, E. L. (1997). Two highly integrated real-time music and graphics performance systems. In *Proceedings of the International Computer Music Conference*, pages 133–140.

Sommerer, C. and Mignonneau, L. (1996). “A-Volve” an evolutionary artificial life environment. In *Proceedings of A-Life V*, pages 167–175, Nara, Japan.

Spector, L. and Klein, J. (2002). Complex adaptive music systems in the breve simulation environment. In Bilotta, E. e. a., editor, *Workshop Proceedings of A-Life VIII*, pages 17–23, Sydney, NSW, Australia.

Tang, E. and Shiffman, D. (2003). Musical flocking box. http://www.antiexperience.com/edtang/works/flockingbox.html.

Tosa, N. and Nakatsu, R. (1996). The esthetics of artificial life – human-like communication character, “MIC” & feeling improvisation character, “MUSE”. In *Proceedings of A-Life V*, pages 143–151, Nara, Japan.