Music by Interaction among Two Flocking Species and Human

Tatsuo Unemi* and Daniel Bisig**

*Department of Information Systems Science, Soka University
1-236 Tangi-machi, Hachiōji, Tokyo, 192-8577 Japan
unemi@iss.soka.ac.jp

**Artificial Intelligence Laboratory, University of Zurich
Andreasstrasse 15, CH-8050 Zürich, Switzerland
dbisig@ifi.unizh.ch

Abstract

This paper describes a new version of "Flocking Orchestra" that generates a type of music through interaction among flocking agents and human visitors. The agents move through a virtual 3D space and respond to the visitor's motions within the observation range of camera. Each agent controls a MIDI instrument whose play depends on the agent's state. The agents are divided into two groups. One group plays melodic tones and the other group plays percussive instruments. The original BOIDS algorithm was extended by adding two forces in order to support interaction. An attractive force causes the agents to move towards the front part of the virtual world when they perceive visitor motion. A repellant force pushes the agents away from the front part in the absence of any visitor's motion. By attracting or repelling the flocking agents a user can change the instrumentation, melodic and rhythmic patterns the flock generates. By this way, the user plays with the flock and can enjoy a type of generative music.

Keywords

collective behavior, flocking agents, visual interaction, generative music.

1 Introduction

This paper describes an extended version of a software entitled "Flocking Orchestra" which was developed by the authors in 2003 and 2004 [1]. The software allows realtime interaction between visitors and flocking agents. The agents move through a virtual 3D space and respond to the visitor's motions within the observation range of camera. Each agent controls a MIDI instrument whose play depends on the agent's state. By attracting or repelling the flocking agents a user can change the instrumentation, melodic and rhythmic patterns the flock generates. By this way, the user conducts the flock in a similar way as a director conducts an orchestra.

The main point of the extension is that the agents were divided into two groups. One group plays melodic tones and the other group plays percussive instruments. Each group intends to form a flock by BOIDS algorithm [2]. This extension produces more complex collective behavior by different parameter settings between groups, just as these are different species. Through an experimental exhibition, almost all of visitors enjoy playing with agents.

Interaction between virtual creatures and human is not a new feature in the field of Artificial Life. Depending on the complexity and adaptivity of the agent's behavioral response to interaction, interesting relations and impressive experiences might be produced for the visitor. A-Volve [3] utilising evolutionary computation and MIC & MUSE [4] tuned by artificial neural networks are typical art works toward this direction.

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 [5] served as basis to produce a musical flocking system of evolving agents [6]. In another project, a flock of agents moves through a three dimensional space which is segmented into different acoustical regions [7]. Whenever an agent enters a particular region, a predefined musical pattern is rendered audible. In both examples the flock behavior and the resulting acoustical output is completely autonomous. The system presented in this paper differs fundamentally from these approaches in that allows the user's behavior to influences the flock. A project by Rowe and Singer [8] employs the Boids concept for an interactive musical performance. In this system, the acoustical output of a several musicians modifies the behavior 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 [9]. 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 behavior of a virtual instrument by means of their gestures [10]. 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 behavior of a simulated flock with the behavior of 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 music serves as a means of catching attention and motivating users to engage into an interaction with the system.

The following sections describe the basic mechanisms and extension on flocking behavior and music play, introduce the technical aspects on our software implementation, summarize our experimental exhibition, and

then conclude with some possible orientations of future works.

2 Interactive Flock

BOIDS is one of well established techniques for computer graphic animation to simulate a type of collective behavior of animals, such as school of fish, flock of birds, and herd of herbivores. It has been used to make a animation film combined with cartoon and/or real action in some famous Hollywood movies. The basic idea is to draw trails of a number of animals by integrating local interaction among individuals instead of hand drawing. Each agent is controlled by a set of forces. The flocking forces cause the following behavior:

1. collision avoidance against agents and obstacles,
2. velocity alignment with neighbouring agents, and
3. flock cohesion by attraction to the neighbouring agents center.

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.

This method made it much easier to produce several different types of complex behavior by tuning some parameters. The original BOIDS algorithm was extended by adding two forces in order to support interaction. An attractive force causes the agents to move towards the front part of the virtual world when they perceive visitor motion. A repellant force pushes the agents away from the front part in the absence of any visitor motion. Interaction forces realize the flock's behavioral response to user input. They cause the following to behaviors;

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:

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. Figure 1 shows an illustrative explanation of attraction force.

In this new version, the agents are divided into two species as described above. The relation between individuals in the same species is not different from old version, but it is necessary to explain a bit about the interaction between agents from different species. Each agent observes other agents around it to calculate three types of forces listed above. It sees all of agents within the observation range for collision avoidance, but it ignores agents of different species to resolve the forces for cohesion and alignment. This means that the agents tends to form a flock for each species, and the flock of other species is recognized as a moving obstacle. It sometimes breaks a flock into two or more when two flocks of different species encounter each other. By this

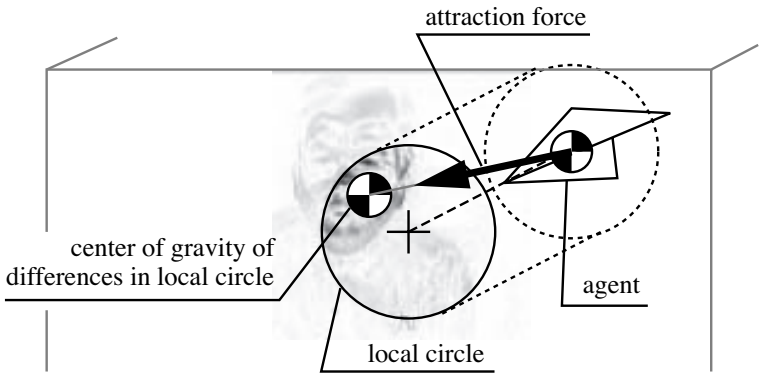

Figure 1: Attraction force toward detected motion.

effect, it produces more complex phenomena than the case of single species. Different settings between species on mobility and observation make the behavior more complex, especially on the minimum and maximum speed and the balance among forces. For example, a species of high speed and weak tendency of collision avoidance makes a steady flock that often breaks a flock of the other species.

3 Music by Flock

The agents for melodic tones control the MIDI instruments in the same way as the old version. The x, y, and z components of the agent's position map into pan, pitch, and velocity of the sounds it generates. The balance between loudness of left and right speakers is determined from the horizontal position of agent. The higher the agent flies, the higher the sound it generates becomes. To add a type of flavor to the melody, a graphical user interface was designed to restrict the playable pitches within a set of notes such as white keys of the piano, blue note, ethnic and so on. Combination of the types of instruments and scale, it generates a music with a flavor of classic, jazz, rock, traditional Japanese, avant-garde and so on. Each agent makes a sound of one note in each chance by weighted mixture of two instruments, primary and secondary. The primary instruments sounds louder when the agent is affected by stronger attraction force caused by visitor's motion. It generates a sound of only the secondary instrument when no motion is observed. A typical setting for a style of rock music is to allocate the distorted electric guitar as the primary instrument and the picked electric bass as the secondary instrument. The range of playable pitches of secondary instrument can be shifted by one or two octaves lower than the primary one. The other combinations, such as violin and cello, oboe and fagot, and alto sax and acoustic bass, are useful to produce mood of typical music genre.

For the agents of percussive instruments, the x and z components correspond to pan and velocity in the same way with the melodic tones, but the y component is used to select a timbre from a predefined set. The range of y value is divided into some discrete spans each of which one of timbres in the drum kit is allocated in. Some instruments, such as hi-hat cymbal, conga, triangle and so on, have plural timbres for two or three different operations, but the others, such as bass drum, chinese cymbal, wood block, and so on, have only one timbre for each. Figure 2 shows the graphical user interface to allocate timbres. The agents for percussion are divided into some groups where each group includes as same number of members as possible. A list of timbres are allocated to each group, and each agent generates a sound of a timbre in the list according to its vertical position. If the agent is flying higher, a timbre listed in an upper row is selected, and it is flying lower space, the one listed in a lower row is selected. As similarly as the case of melodic tones, two types of timbre lists are set

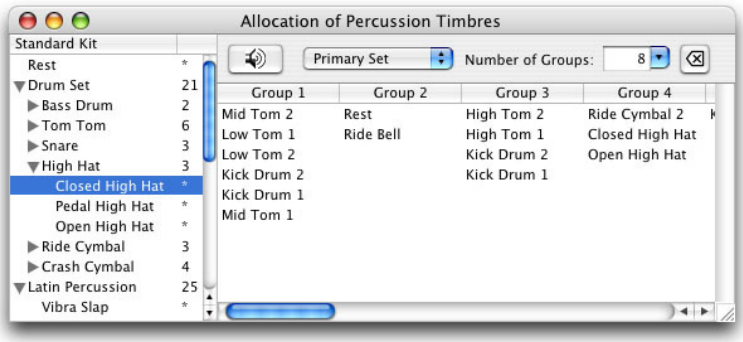

Figure 2: Graphical user interface to allocate timbres of percussive instruments.

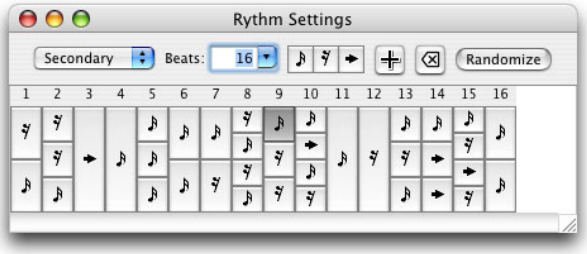

Figure 3: Graphical user interface to allocate rhythm patterns.

up for each group, primary list and secondary list. Two sounds of timbres picked up from the lists are mixed in the same way of the case of primary and secondary instruments for melodic tones. By allocating the timbres of long and strong sounds, such as crash cymbal, open hi-hat, and snare drum, for the primary list and the timbres of short and weak sounds, such as ride bell, rim shot, and closed hi-hat, for the secondary list, it becomes easier for the visitors to notice that their action is affecting the sounds.

The new feature on rhythm was also introduced in this version. The user can choose the style to determine when each agent should make a sound from the following alternatives:

1. each agent is selected in turn according to a fixed order,
2. each agent possesses the same probability of being selected,
3. the selection probability of an agent is proportional to strength of its attraction force, and
4. the selection probability of an agent is proportional to its movement speed.

The third alternative was newly added, and the functionality of the first alternative was changed. The first alternative was originally introduced to generate a music in constant tempo. However, in the previous version, the system generates the sound of one note in every step, that means there is neither a rest nor a variation of tone length. To make the music more expressive, we introduced a function to indicate a rhythm pattern. Figure 3 shows the graphical user interface to set up this information in the same style that is used in the genetic information of SBEAT [11, 12, 13]. For each beat, one mark is allocated chosen from three marks, note on, rest, and continuation, of which idea is originally from GenJam by Al Biles [14]. This new functionality makes it possible to realize atmosphere of some types of music that strongly depend on a rhythm pattern, such as latin music, eight beat rock, and house and techno music.

4 Implementation

This software runs on Apple's newer computers with G4 and G5 micro processors and MacOS X 10.3 or later as the operating system. No special hardware is needed other than a camera. This means it works on the hardware available in the commercial market. The computational power necessary for this software is depending on the resolution of screen and the number of agents. The recently available fastest machine, 2.7 GHz dual G5 processors, can drive this software in the VGA screen, 640 $\times$ 480 pixels, and 512 agents in the video frame rate, 30 fps. In the case of a slower notebook with 670 MHz G4 processor, it is necessary to reduce the resolution to 320 $\times$ 240 and the number of agents to 128 to obtain a speed faster than 15 fps which is enough for smooth run. On the capacity of memory, this software does not require more than 10 M bytes in usual settings.

The source code was written in Objective-C programming language embedding an efficient algorithm for each agent to find the others within the observation range to calculate the forces listed in section 2. The position of each agent is represented in three floating point real values. Without any trick, calculation of complexity order O(N^2) would be wasted to gather all of information on the affecting agents for each agent. This is an obstacle to make the interactive flock include many agents, 500 or more. To avoid this inefficiency, we divided the virtual space into a number of grid volumes to manage which agents are in which local area. By this information, the number of agents to be considered for the distance can be tremendously reduced. The information is stored in a three dimensional array consisting of double linked linear lists of pointers to the memory representing an individual agent. Each linear list holds the agents in the corresponding local volume in the virtual space. The element of the list is eliminated and is added to another appropriate list whenever the agent moved to another local area.

Parallel processing using special instruction set of AltiVec in PowerPC CPU and APIs for graphical processing unit in Quartz 2D framework are accelerating the calculation not only for agents' properties but also image processing. To detect the motion of visitors, it is necessary to calculate difference between previous and current frames for each pixel. This result is used to calculate the attraction force for each agent. Because this process could need much computational time, we employ shrinking process to reduce the captured image into one fourth in area size. The recent hi-speed web camera, such as iSight, can capture a sequence of full-color images by the VGA resolution and video frame rate. This performance is good for the audio visual chat through the network, but it includes too much data to process the images in each frame time. Fortunately in this application, VGA size is good to mix the captured image and agent animation to let the visitors recognize what the camera captures, but such a fine resolution is needless for motion detection.

Multi-threading is also facilitated for a dual processor machine. The computation for grabbing captured images by QuickTime API is already runs on a separated thread than the main code of the application, but of course it is effective to separate a part of the main code in an independent thread because the QuickTime thread is not always busy. Different threads in a same process can share the global memory, but it must be carefully coded under consideration of dependency of calculation results. In the current implementation, multi-threading is used only for calculation of agents' velocity in the next step. It is very effective when the number of agents is larger than three or four hundreds.

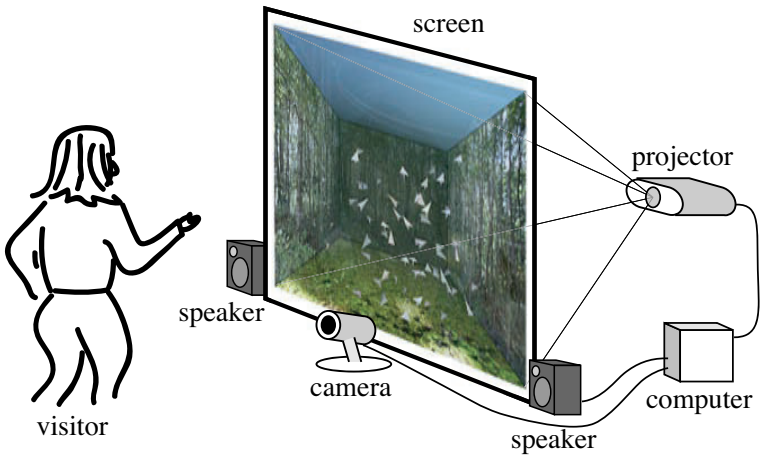

Figure 4: Configuration for experimental exhibition.

5 Experimental Exhibition

One of the authors conducted an experimental exhibition at the campus of his university in July 30 and 31 in this year, 2005. It was planned as an introduction of research activities of the department for visitors in the Open Campus. Because this event was organized to promote the university mainly for hi-school students, The more than half of visitors are hi-school students, but of course their parents, brothers, sisters, and some old boys and girls who were graduated from the university accompanied them. Totally more than two hundreds persons have visited to see our installation in these two days.

Figure 4 illustrates the configuration of hardware. To prevent projecting a shadow of visitors on the screen, we set a projector at the back side. The software ran under MacOS X 10.4 on PowerMac G5 2.5GHz dual CPU attached with iSight camera and a stereo speaker system consisting of left and right speakers and a subwoofer.

Figure 5 shows four snap photos taken at the room of exhibition. Almost all of visitors were surprised and enjoying regardless to the settings of instruments and background images we examined. They usually noticed soon that the agents are following to their motion and tried to move their arms and bodies. However, it seemed difficult for about three fourths of visitors to notice that the sounds are depending on their motion without our suggestion. It might be because they easily focused their attention on visual phenomena and tended to ignore the acoustic information. The system drew a white circle at the background of each agent when it generated a sound, but it was also usually ignored. Some visitors asked us what is the sign of circles.

6 Conclusion

The new version of the software was successfully implemented to generate a richer variety of generative music, and successfully provided impressive experience for the visitors through the experimental exhibition. One point not matched with our expectation is that it was difficult to be aware of the relation between the generated sounds and the visitors' motion. It might be good idea to introduce another method to show a type of sign to indicate the agent generates the sound other than simple background circle, such as changing the shape and/or color of agent.

More extension of this software might include:

1. embedding adaptive mechanism such as learning and evolution to generate more complex behavior and relation between the agents and human,

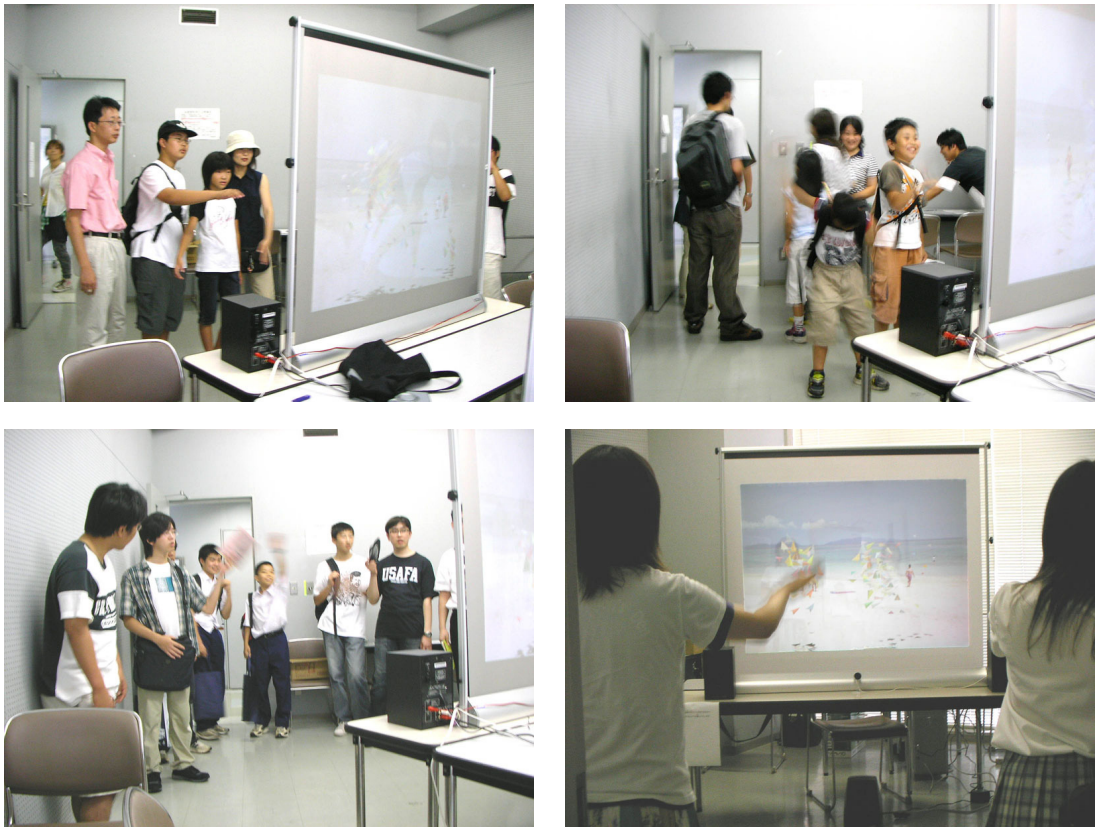

Figure 5: Photo snaps at experimental exhibition in Soka University during the Open Campus in July 30 and 31, 2005.

2. connecting a number of computers via network and enabling the agents to migrate among several virtual spaces on the connected machines, and

3. introducing more than two species for more instruments to enable to generate more expressive music.

The first item in the above list must be introduced with careful consideration on how humans can adapt to more complex behavior of the agents. Because it might be difficult for people to accept too much complex reaction, the process of learning and evolution should progress more slowly than humans do. This means that it takes long time for these adaptive mechanisms to be effective. It should not be implemented for an exhibition but for daily usage at home to interact in many weeks or months.

The binary executable is being distributed on the web. According to our e-mail correspondence and private communications in the Apple Users Group Meeting in Tokyo, some Mac users are enjoying this software at home. This can be seen as an example of current movement of popularization of art, or amalgamation of art and entertainment, accelerated by recent improvement of the personal computers and the internet. We may expect that it can provide not only impressive experience but also inspiration toward quite new style of computer usage as a type of technology innovation.

The up to date information of this project and the downloadable software are available from the web site of following URL:

`http://www.intlab.soka.ac.jp/~unemi/1/DT1/` .

We hope as many persons enjoy our work as possible.

Acknowledgment

The authors would like to thank students in Soka University, Tatsuya Sasaki, Ryosuke Ono, Takashi Tohyama and Ken'ichi Uemura, for their help for the experimental exhibition in the Open Campus, Mitsuo Kawakita for his cooperation to develop the newest extension, and Prof. Rolf Pfeifer for his support for this international collaboration. This work is partially supported by Grants-in-aid for Scientific Research #17500152 from Ministry of Education, Culture, Sports, Science and Technology (MEXT) in Japan.

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

[3] Sommerer, C and Mignonneau, L. : A-Volve an Evolutionary Artificial Life Environment, Proceedings of A-Life V, 167–175, 1996.

[4] Tosa, N. and Nakatsu, R. : The Esthetics of Artificial Life – Human-like Communication Character, "MIC" & Feeling Improvisation Character, "MUSE", Proceedings of A-Life V, 143–151, 1996.

[5] Klein, J. : Breve: a 3D Simulation Environment for the Simulation of Decentralized Systems and Artificial Life, Proceedings of A-Life VIII, 329–334, 2002.

[6] Spector, L. and Klein, J. : Complex Adaptive Music Systems in the BREVE Simulation Environment, Workshop Proceedings of A-Life VIII, 17–23, 2002.

[7] Tang, E. and Shiffman, D. : Musical Flocking Box, http://www.antiexperience.com/edtang/works/flockingbox.html, 2003.

[8] Rowe, R. and Singer, E. L. : Two Highly Integrated Real-Time Music and Graphics Performance Systems, Proceedings of the International Computer Music Conference, 133–140, 1997.

[9] Blackwell, T. M. and Bentley, P. J. : Improvised Music with Swarms, Proceedings of the Congress on Evolutionary Computation, 1462–1467, 2002.

[10] Roads, C. : The Computer Music Tutorial, MIT Press, 1996.

[11] Unemi, T. and Nakada, E. : A Tool for Composing Short Music Pieces by Means of Breeding, Proceedings of the IEEE Conference on Systems, Man and Cybernetics, 3458–3463, 2001.

[12] Unemi, T. and Senda, M. : A New Musical Tool for Composition and Play Based on Simulated Breeding, Proceedings of Second Iteration, 100–109, 2001.

[13] Unemi, T. : SImulated breeding – A Framework of Breeding Artifacts on the Computer, in A. Adamatzky and M. Komosinski eds. Artificial Life Models in Software, Springer, 301–322, 2005.

[14] Biles, J. A. : GenJam: A Genetic Algorithm for Generating Jazz Solos, Proceedings of the International Computer Music Conference, 131–137, 1994.