Interactive Swarm Orchestra

Daniel Bisig
Institute for Computer Music and Sound Technology
Zurich University of the Arts, Switzerland
e-mail: dbisig@ifi.unizh.ch

Martin Neukom
Institute for Computer Music and Sound Technology
Zurich University of the Arts, Switzerland
e-mail: martin.neukom@zhdk.ch

John Flury
Institute for Computer Music and Sound Technology
Zurich University of the Arts, Switzerland
e-mail: flury@syntharp.com

Abstract

The project Interactive Swarm Orchestra (ISO) employs flocking algorithms to control computer sound synthesis and 3D sound positioning. Synthesis, positioning and movement of several simultaneous sound events are modeled according to swarm behavior. Camera-based tracking allows visitors to interact with this acoustic flock and thereby change its spatial distribution and synthesis properties. This paper focuses on the description of the software components that have been developed specifically for this project. These components encompass of a sound synthesis framework, functionality for 3D sound projection based on Ambisonics, a generic multi-agent simulation environment, and video tracking software for conventional video cameras and for SwissRanger 3D cameras. All software source code is publicly available.

1. Introduction

The field of computer music and its performance provides a vast territory for artistic experimentation. The qualities of synthetic sounds are not predefined by the physical properties of musical instruments nor is the creation of these sounds compulsorily associated with the manipulation of traditional interfaces. Furthermore, computer music is unbiased towards particular performance styles and freely shifts in a continuum between improvisational and compositional as well as presentational and participatory styles. Accordingly, computer music practitioners see themselves confronted with an almost unlimited number of choices for specifying and controlling the sound generation process and for defining the relations between composer, performer, machine and audience. For this reason, current research in computer music has shifted somewhat away from the development of sound synthesis techniques and becomes increasingly engaged with issues that traditionally belong to the field of human machine interaction (HMI). This comprises novel interface designs, improvements in perception and feedback (e.g. 3D sound projection or multi-modal feedback), the creation of intuitive interaction modalities (e.g. camera-based full body tracking), and novel control algorithms for sound generation that simplify real-time performance.

We believe that the application of concepts and techniques from Artificial Life (ALife) research can greatly contribute to this research topics. ALife explores artificial complex systems that exhibit life-like properties such as adaptivity, autonomy, diversity, self-organization and emergence. Complex and self-organized systems have great appeal for art, since they can continuously change, adapt and evolve [1]. By adding means for interaction to algorithms and simulations from ALife, these systems may become flexible and powerful tools that contribute to all of the previously mentioned research topics and thereby possess the potential to establish a unified foundation for interactive algorithmic forms of computer music.

Swarm simulations form an important part of ALife research and explore principles of self-organization and emergence in the appearance of group behavior [2, 3]. Such simulations are particularly interesting since they can express a large variety of different types of behaviors that range from very simple and reactive organizations up to highly complex systems that can learn and evolve. Furthermore, swarm simulations can easily be adapted to deal with any number and dimension of parameters (e.g. sound synthesis parameters). In addition, swarm simulations lend themselves very well to intuitive and natural forms of interaction. There exist several projects by researchers and artists that apply swarm simulations for the generation of music. For example, Tim Blackwell has applied swarms that act as artificial musicians during a life improvisation with human musicians [4]. This system employs a combination of swarm and stigmergic behavior [5] and has also been employed to control granular sound synthesis [6]. A collaboration of Tatsuo Unemi with one of the authors of this paper has resulted in the realisation of several interactive systems that rely on swarm behavior to generate both visual and acoustic feedback [7 - 10]. Finally Yuta Uozumi has realized a software for live composition based on a predator-prey simulation [11]. These projects illustrate the impressive creative and aesthetic potential of swarm-based computer music. Unfortunately, these efforts represent idiosyncratic combinations of very specific interaction forms, swarm behaviors and sound generation types and make no attempt at providing generic tools to explore the vast space of possibilities of swarm-based computer music. The ISO project [12] is an attempt to create such a generic tool and supports a wide variety of research and performance in swarm -based computer music.

2. Concept

The ISO project is a manifestation of our belief, that practical and conceptual ideas from Artificial Live (ALife) provide an excellent foundation towards the establishment of a coherent approach to several important aspects of computer music (sound synthesis, composition and interaction). Our approach employs a generic swarm simulation as intermediary between musician(s), sound generation and acoustic projection. It is the simulated agents' behaviors that affect the mapping of the performer's activities into musical structure and its timbral, temporal and spatial development. We intend to shift the creative focus of a musician's work towards the design of properties, behaviors and interrelationships among agents and their musical dependencies. Depending on the agents' capabilities, autonomy and reactivity, the resulting music may have unexpected and emergent properties or resembles a manually scored composition. Since the agents' characteristics can change over time, the music may progress through improvisational and pre-planned phases and can alternate between presentational or participatory performance styles. These principles can be applied to high- and low-level musical structures. For instance, the musician may choose to directly control the global development of a piece but employs generative principles for sound synthesis. Or he might link the music's long and short term temporal changes to the same agent behaviors but acting on different time scales. Finally, the musician can apply the agent simulation to create musically meaningful correlations among sound synthesis parameters. For example, these correlations can be derived from neighborhood relationships among agents. Such correlations help to reduce the dimensionality of the musical search space and thereby simplify the musician's explorative and improvised music creation during life performance.

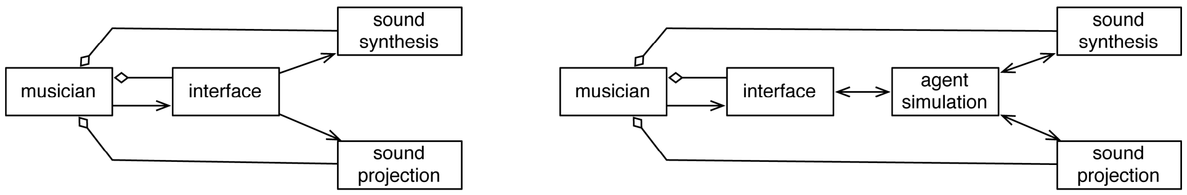

Figure 2: Highly simplified schematics of a traditional (left) and simulation-based (right) computer music performance. Normal arrows indicate control relationships. Prism shaped arrows denote perceptional feedback.

3. Implementation

An important aspect of the ISO project concerns the development of software and hardware tools that support the application of swarm simulations for the creation of computer music. To account for the huge diversity of different types of computer music and swarm-based simulations, our implementations try to be as generic and flexible as possible and minimize interdependencies among individual components. Correspondingly, we don't attempt to compete with more specialized tools with regard to user friendliness, simplicity or computational performance. With the exception of our camera-based tracking software, all software tools exist as C++ libraries. Accordingly, our system takes a programming type of approach to the creation of computer music. We admit that this approach limits our musical target audience considerably since the visual programming style of such programs as Max/MSP or PD has won over almost all practitioners in computer music. For this reason, some of our future plans concern the establishment of interfaces between our system and these highly popular music programs. All our software is based on cross platform libraries that are available under an Open Source license for Mac OSX, Linux, and Windows. By distributing the ISO tools as Open Source software and hardware, we hope that their usage will transgress the confinement of our own institute and help to create a community of programmers, musicians and researchers who contribute to their further development.

At the current project stage, we have developed individual software components for sound synthesis, sound projection, swarm simulation, and camera-based tracking.

ISO Synth

ISO Synth is a C++ library for realtime sample- and synthesis-based computer music creation, which we have developed from scratch. We resorted to this effort in order to guarantee a common implementation standard and optimal interoperability with other ISO tools. ISO Synth implements an event-based score mechanism and supports sound spatialization via two and three-dimensional ambisonic projection [13]. It supports a variety of standard signal processing and sound synthesis techniques. For signal processing, these are: sampling, pitch shifting, time stretching, conversion between time and frequency domain and filtering. For sound synthesis, these are: additive, subtractive, wavetable, granular, frequency modulation, amplitude modulation, waveshaping, phase distortion, sample-based synthesis.

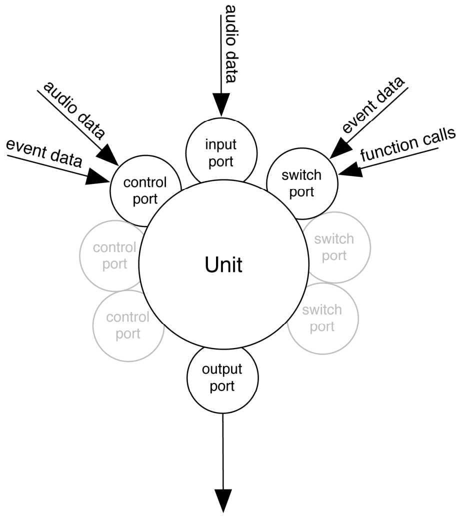

Figure 3: ISO Synth Unit Properties. Arrows indicate any number of inputs. Audio data stands for continuously changing input data. Event data represents data values that change at discrete and possibly irregular time intervals. Function calls allow the re-configuration of ports during run time.

ISO Synth implements the widespread "Music N" unit generator concept [14]. A unit forms the basic building block for the creation or processing of audio signals. Depending on the their functionality, units possess a variety of ports that allow them to exchange audio data and communicate with other units in a patch (see Figure 3).There exist four different types of ports. Input ports retrieve audio data from connected units. Output ports pass audio data to connected units. Control ports retrieve both audio data and events and thereby change some properties of the associated unit. Switch ports are intended for infrequent changes that alter a unit's behavior more substantially than control ports do (i.e. setting a unit inactive, loading a new amplitude envelope etc.). A list of currently available units is presented in table 1. ISO Synth possesses some less common properties, which contribute to its flexibility and form the basis for its interaction with other ISO tools. Every port and every unit can possess a different audio rate, channel count, and buffer size. The connections between ports automatically take care of necessary signal conversions. There exists no explicit distinction between audio and control rate. All control ports operate at audio rate unless explicitly set to a lower rate. Unit patches are not restricted to directed graphs but may contain cycles (including connections from the same unit's output port into its input or control ports). Units can be nested and communicate via internal ports. An entire patch can be serialized, saved and restored at any time during its operation. Patches can exchange arbitrary data via UDP with other ISO tools.

Unit Class Name	Function	Unit Class Name	Function
FFTUnit	Fast Fourier Transform	BWFilter	Butterworth Filter
IFFTUnit	Inverse Fast Fourier Transform	CombFilter	Comb Filter
SampleUnit	Loop, Transpose, Reverse	FSMFilter	Frequency Sampling Filter
PointEnvelope	Break Point Envelope	ResonFilter	Reson Filter
BLPulseGen	Band Limited Pulse Generator	VocalFormantFilter	Vocal Formant Filter
DCSPulseGen	Dynamically Controlled Spectrum Pulse Generator	FFTStrech	Spectral Frequency Stretching
NoiseI	Interpolating Uniform Noise	FFTThreshold	Spectral Amplitude Thresholding
NoiseS	Stochastic Noise	FFTPhaseMultiply	Spectral Phase Multiplication
WaveTableOscil	Wavetable Oscillator	FFTAmpDerivative	Spectral Amplitude Derivative
DelayUnit	Delay Line Unit	GranularUnit	Granular Synthesis
WaveTableShaper	Wave Table Shaping	Decoder	Ambisonics Decoder (2D & 3D)
AllPassFilter	All Pass Filter	Encoder	Ambisonics Encoder (2D & 3D)

Table 1: A Selection of ISO Synth Unit Types.

ISO Tracker

At the moment, ISO allows interaction via Midi or camera-based tracking. Midi-based interaction is implemented as part of the ISO Synth tool. Camera-based interaction is realized as two separate applications that rely on Intel's computer vision library [15]. Our focus on camera-based interaction (instead of other technologies such as wearable sensors, electromagnetic or ultrasonic tracking) is based on reasons of cost and flexibility and because an untethered setup simplifies participatory or casual forms of interaction. ISO Tracker detects the motion, position and orientation of an arbitrary number of persons and transmits these data via UDP to ISO Flock. Motion tracking is implemented via a pyramid approach as described in [16].

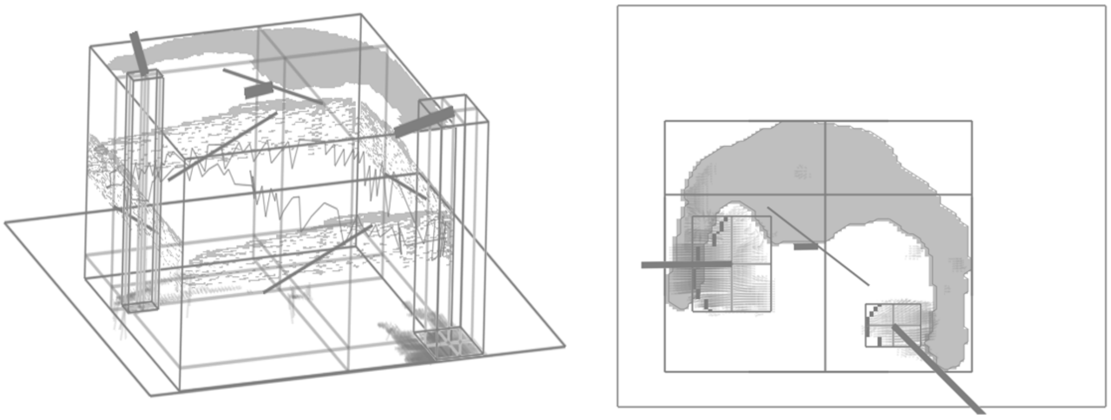

Figure 4: SwissRanger SR-2 based tracking. The screenshots depict a person's outline, bounding box, orientation (thin lines) and motion (thick lines). The person's moving hands are distinguished via Motion Segregation. In both images, the camera is mounted to the ceiling and points straight down. Left side: sidewise tracking view illustrating the combination of brightness and distance information. Right side: top-down tracking view emphasizing the detection of hand motions.

Motion segregation usually allows to distinguish between different body parts (i.e. hands, head, and feet) as long as a person is moving. The tracking software exists in two versions. One version supports standard video or web-cams that capture 2D color or grayscale information. A second version of the tracker software has been specifically written for the SwissRanger SR-2 camera [17]. This camera employs the time-of-flight principle by emitting modulated infrared light. The time of arrival of the reflected light allows the computation of a "distance image" at a resolution of 124 x 160 pixels. Thanks to the availability of distance information, this specific tracker software version exceeds the capabilities of the 2D tracker in that it tremendously simplifies occlusion problems. Furthermore, if the camera is pointing straight down from the ceiling, it allows the calculation of a person's height, vertical motion and vertical orientation. Unfortunately, the SwissRanger camera suffers from a serious drawback with regard to our application. The fixed focal length characteristics of its lens in combination with maximum tracking distance of 7 meters results in a very small surveillance area. This camera is therefore unsuitable for tracking people in a space larger than a few square meters.

ISO Flock

Most of our implementation effort has been put into the development of a generic swarm simulation library. The main non-generic aspect of this library concerns its focus on simulating large numbers of agents each of which possesses a very simple morphology. In particular, there are currently no facilities for modeling segmented body architectures that exhibit rigid or soft body dynamics. Apart from this restriction, the library allows the realization of a vast variety of swarm simulations.

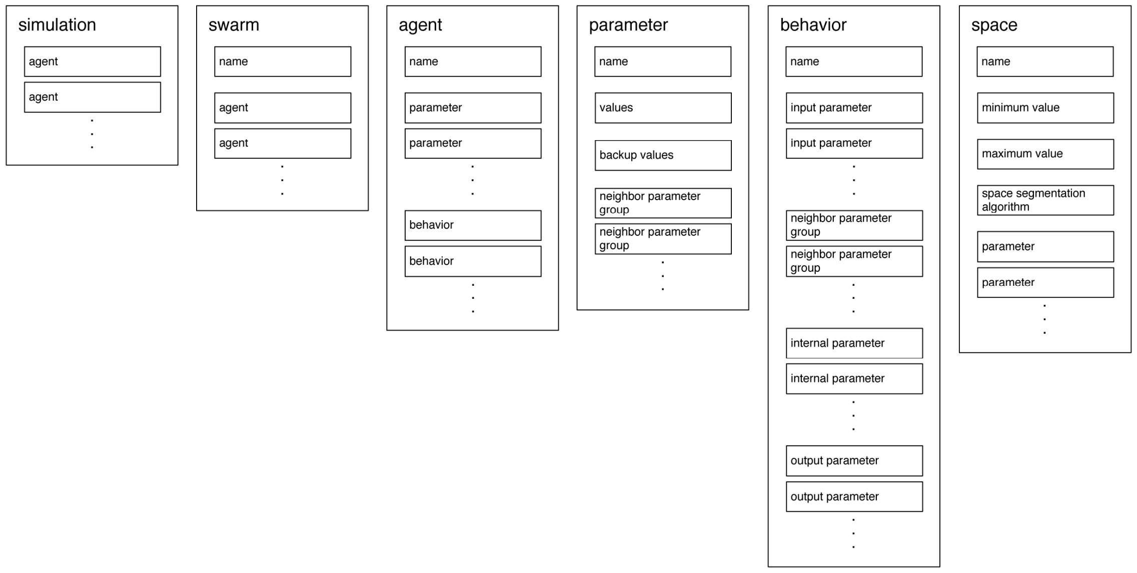

Figure 5: Main Classes of the ISO Flock Library.

The implementation of ISO Flock defines a small set of main classes (see figure 5) from which simulations can be built either by configuring these components or by creating derived classes. The simulation class manages all agents and updates all other classes at regular intervals. The swarm class acts as a labeled container for agents which are functionally equivalent. It provides functions for creating or deleting agents at runtime and manages the exchange of data via UDP with other ISO tools.

Agents are labeled containers for parameters and behaviors. Parameters represent labeled vectors of arbitrary dimension and manage relationships (euclidian distance and direction) with other parameters organized in neighbor groups. Agent behaviors define functional relationships among parameters. Behaviors distinguish between input parameters, internal parameters and output parameters. Internal behavior parameters are specific to a particular behavior and are created when the behavior is instantiated for the first time.

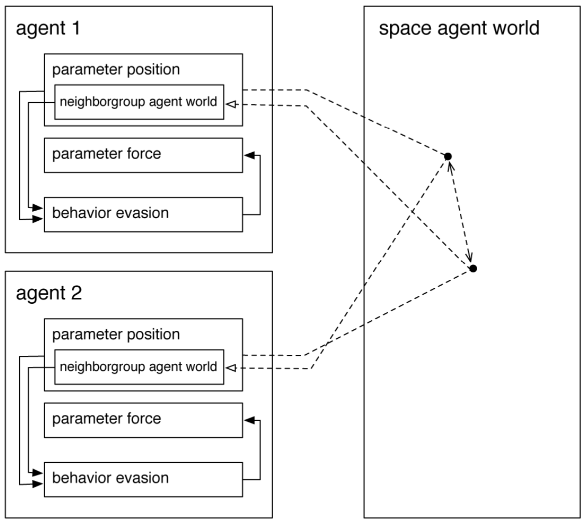

Figure 6: Evasion Behavior. For simplicity, only two agents are depicted. At the beginning of a simulation step, the location of the parameter position is updated in the space entitled "agent world". Then, the distance between the two position parameters is calculated and their neighborgroups are updated. Subsequently, the evasion behavior within each agent changes the force parameter based on the values of the position parameter and its neighbors. At the end of a simulation, all parameters update their values based on the changes that have been caused by the behaviors.

Whenever a behavior is executed, it reads from its input and internal parameters as well as neighbor groups and writes into its output parameters. These changes are buffered within the output parameters. This helps to avoid that the simulation output is affected by the particular sequence in which behaviors are executed. All parameters are normal agent parameters and their distinction into different types (internal, input, output) applies only for a particular instantiation of a behavior. Whenever a new behavior is created, the supplied parameters can have entirely different types. It is important to note that parameters cannot only be changed by behaviors but also by an event-based system. The event system provides the same functionality as in ISO Synth and permits score-like choreographing of agent parameters. Events can also be created on the fly such as for example when certain conditions are observed by the camera tracking software. The last fundamental ISO Flock class deals with spatial calculations. This space class contains spatial partition algorithms for the calculation of euclidian distances among parameters and thereby manages their neighborhood relationships. Parameters can simultaneously exist in an arbitrary number of spaces. One of the consequences of the generic nature of agents, parameters and spaces is the somewhat counterintuitive fact, that it is not the agents that exist in a particular space world but rather their parameters. It is not necessary to take this peculiarity into account when designing fairly standard types of agents (i.e. agents that possess the properties of position, velocity and acceleration and that only map their position into a space that would conceptually correspond to the classical agent world). On the other hand, our generic parameter neighborhood approach allows for unconventional interactions among agents based for example on character trait relationships. Figure 6 depicts the relationship between agents, parameters, behaviors and spaces for a simple evasion behavior.

Behavior Name	Function
AccelerationBehavior	Limit linear and angular acceleration of parameter values
AlignmentBehavior	Push parameters (typically velocity) towards similar values depending on the neighborhood of other parameters
BoundaryWrap	Wrap parameter values when they exceed certain limits
BoundaryRepulsionBehavior	Update a parameter (typically force) when other parameter values exceed certain limits
CohesionBehavior	Push neighboring parameters towards similar values
DampingBehavior	Push parameter towards a fixed value
EvasionBehavior	Push values of neighboring parameters away from each other
GridAverageBehavior	Change parameter value based on the averaged vector field values in the parameter’s neighborhood
SplineFollowBehavior	Push parameter value towards and along a spline line extending through parameter space

Table 2: A Selection of ISO Flock Behaviors.

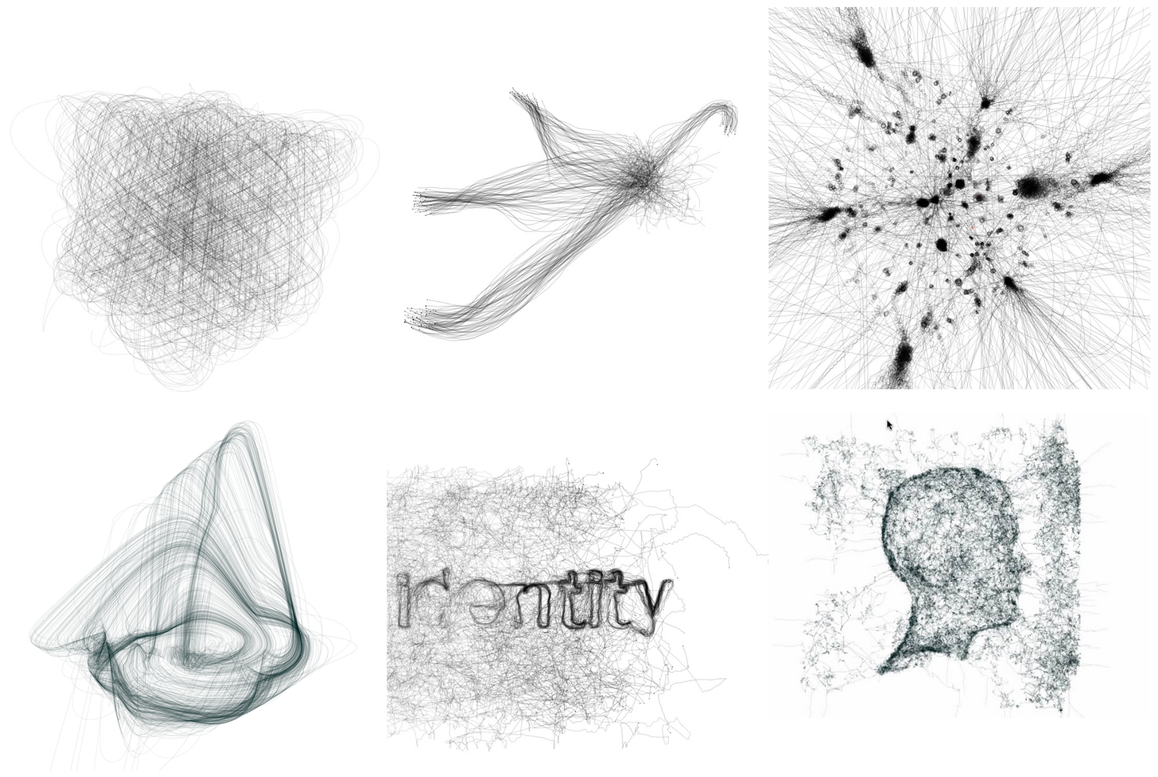

Figure 7: A Selection of ISO Flock Simulation Applications. The visual output is created by drawing the agents trajectories for the duration of 1000 simulation steps. Top row from left to right: random agent movements within repulsive boundaries, standard BOIDS type of flocking, agents moving towards point attractors. Bottom row from left to right: agents moving within a force field created from a Roessler equations, agents following splines created from letters, agents moving towards the surface of a human head mesh model.

In its current implementation, the ISO Flock library provides a group of specialized spaces and behaviors that inherit from the generic base classes. The “point space” class manages distance calculations among point-like spatial objects (i.e. parameters) via a Quadtree, Octree or higher dimensional space partitioning algorithm. The “shape space” class implements an R-Tree algorithm for the calculation of distances among objects in space that possess a shape (as opposed to point like objects). This allows agents to move along splines or on the surface of triangulated meshes. Such spatial objects can be employed to structure the environment within which agents exist. An example application transforms the tracked outline of people into a spline that serves as movement guide for agents. For a similar purpose another type of space manages the distribution of vectors on an n-dimensional regular grid. Such grids can serve for example as static or dynamic force fields and propel or slow down agents as they move through space. Another example application updates such a force field based on tracked visitor motion. With regard to behaviors, there exists a small selection of example implementations. Table 2 lists some of these behaviors.

We have created a series of simulation applications that are based on the functionality of the ISO Flock library. In all these applications, the agents’ positions have been drawn as trajectories in three dimensional space. Figure 7 shows the visual output of some of these applications.

4. Results and Discussion

Our implementation efforts have led to the creation of a series of ISO tools that should be sufficiently generic and flexible to create a wide variety of swarm simulation-based computer music. The stability and performance of all tools have been tested when running for extended periods of times (several days). We are therefore confident, that the tools reliability is perfectly sufficient for critical real-time performances or long-lived installation projects. All tools are extensively documented on our project site [12]. Apart from implementing the desired functionality, all ISO libraries have gone through several cycles of evaluation and feedback by musicians working at our institute which helped us to simplify the APIs. For this reason, the ISO project is on the verge of going beyond its pure implementation stage and can now be employed for research and music creation. Several composers at the institute have started working with the software. We are in the early stages of planning a dance choreography that employs ISO Flock to link the dancers' motions with a live computer music improvisation. At the current project stage, we believe it is crucial to start establishing and maintaining a community of users and programmers who contribute to the further improvement of the ISO tools. For this reason, we have started to provide workshops for interested musicians and artists and added some online community tools to the project site such as a Wiki and a Forum. A first workshop has taken place at our institute and a second one will have been conducted by the time this paper is printed at the Tama Art University in Tokyo, Japan. A further workshop is planned in Bangalore, India in the beginning of 2008. Due to the fact, that the project just recently started to focus on the promotion of its tools for application in art and science, we are not yet able to present artistic realizations or scientific experiments that help to evaluate the suitability and limitations of our approach to computer music generation. This activity will form the main focus throughout the planned extension of the ISO project.

5. Conclusion and Outlook

So far, the ISO project has successfully created a coherent and flexible software infrastructure that will help to realize swarm simulation-based computer music for research and performance. Due to the project's initial focus on implementation and documentation of these software tools, the evaluation of the general suitability of our approach to computer music generation needs to be postponed to another paper.

Feedback from our first ISO workshops indicates that many computer musicians express curiosity and interest about the combination of swarm simulations and computer music. On the other hand, most of the workshop participants possessed little more than a vague and abstract idea concerning the properties and capabilities of swarm simulations and how these properties could be linked to computer music. Hardly anybody in our workshop audience was aware of existing examples of simulation-based computer music let alone experimented with these possibilities. For these reasons, we conclude that our effort to provide an coherent and (at least for musicians with programming experience) simple to use hard- and software infrastructure constitutes a very important step towards the promotion and exploration of ALife-based computer music.

We currently hope to extend the ISO project for another two years. This extended period of time will allow us to focus on the main questions concerning swarm-based computer music and give us the opportunity to realize a variety of performances that hopefully highlight the aesthetic potential of this approach. Of particular interest are the following questions: How does swarm-based computer music improve explorative composition and improvisation styles? Does swarm-based computer music redefine the relationship, role and authorship between musician, the machine and the audience? Will swarm-based music enhance the application and control of traditional forms of sound synthesis or might it even give rise to entirely novel forms of synthesis? Is it necessary and possible to establish psycho-acoustic principles and guidelines for swarm-based computer music? Will spatial sound projection significantly contribute to and improve the perception and appreciations of swarm-based computer music? Will swarm-based computer music help to create novel forms of art collaborations for example between computer music, dance, interaction design and visual design?

So far, ISO Synth implements a variety of popular and widespread sound synthesis algorithms. None of these algorithms is of particular interest or suitability for swarm-based computer music. It would be much more interesting to experiment with entirely novel synthesis techniques that are closely related to capabilities of swarms. For example, physical modeling techniques that rely on the interaction of non-stationary objects, changing topological relationships and dynamic alterations of object properties, such as for example chemical reactions or grainy or powdery materials, might be particularly interesting. It might also be interesting to devise biologically inspired synthesis techniques the rely on swarm behavior. For example signaling among songbirds, or mating call strategies during rutting seasons.

While developing the ISO Flock library, we relied on visual feedback for debugging and analysis of the agents' behaviors. The aesthetics of this visual feedback and its possibly important role in creating multi-modal feedback have entirely changed the priority of this initially circumstantial activity. For this reason, we plan to create an additional ISO library that will take on the role of a visual synthesizer and complements the functionality of ISO Synth.

So far, we rely entirely on camera-based interaction for controlling the swarm simulation. This type of interaction excels with regard to cost and flexibility but otherwise suffers from severe limitations. Camera-based interaction is notoriously prone to calibration and occlusion problems. Furthermore, its is very difficult to derive accurate and semantically meaningful information from camera images. Finally, camera-based interaction does in itself not provide any feedback for the performer. It will be interesting to envision novel types of interfaces that provide for example haptic cues and are specifically designed to improve the performance of swarm-based computer music.

References

[1] Sommerer, C. and Mignonneau, L. (2000). Modeling Complex Systems for Interactive Art. Applied Complexity - From Neural Nets to Managed Landscapes. Institute for Crop & Food Research, Christchurch, New Zealand, pp. 25-38.

[2] Martinoli, A. (2005). Swarm intelligence: emergence and self-organization in natural and artificial systems. Course notes, EPFL.

[3] Eberhart, R., Shi, Y. and Kennedy, J. (2001). Swarm Intelligence. Morgan Kaufmann.

[4] Blackwell, T. (2003). Swarm music: improvised music with multi-swarms. Artificial Intelligence and the Simulation of Behaviour, University of Wales.

[5] Blackwell, T. and Young, M. (2004). Self-organised Music. Organised Sound, Cambridge University Press, 9, pp. 123-136.

[6] Blackwell, T. and Young, M. (2004). Swarm Granulator. EvoWorkshops 2004, Coimbra, Portugal, pp. 399-408.

[7] Unemi T. and Bisig D. (2007). Identity SA - an interactive swarm-based animation with a deformed reflection. Proceedings of the Generative Art Conference. Milano, Italy, in print.

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

[9] Unemi, T. and Bisig, D. (2004). Playing Music by Conducting BOID Agents. Proceedings of the Ninth International Conference on Artificial Life IX, Boston, USA, pp. 546 - 550.

[10] Bisig, D. and Unemi, T. (2006). MediaFlies - A Video and Audio Remixing Multi Agent System. Proceedings of the Generative Art Conference, Milano, Italy, pp. 63-74.

[11] Uozumi, Y. (2007). GISMO2: An Application for Agent-Based Composition. Lecture Notes in Computer Science, Springer Berlin/Heidelberg

[12] ISO website: http://www.i-s-o.ch

[13] Malham, D.G. and Anthony, M. (1995). 3-D Sound Spatialization using Ambisonic Techniques, Computer Music Journal 19(4).

[14] Dodge, C. and Jerse, T.A. (1985). Computer Music, Schirmer Books, New York, USA.

[15] OpenCV website: http://www.intel.com/technology/computing/opencv/index.htm

[16] Davis, J.W. (2001). Hierarchical motion history images for recognizing human motion. Proceedings of the IEEE Workshop on Detection and Recognition of Events in Video, pp. 39-46.

[17] SwissRanger specifications:
http://www.csem.ch/detailed/pdf/p_531_SR-2_Preliminary-0355.pdf