TOOLS AND ABSTRACTIONS FOR SWARM BASED MUSIC AND ART

Daniel Bisig, Philippe Kocher
Institute for Computermusic and Sound Technology
Zurich University of the Arts
daniel.bisig@zhdk.ch
philippe.kocher@zhdk.ch

ABSTRACT

A new version of our swarm simulation environment for musical and artistic applications is presented. The main improvements of this version concern the integration of an OSC based communication protocol and the addition of a graphical user interface. These extensions offer a variety of approaches for the configuration, manipulation and application of simulated swarms in real time. We hope, that these improvements open up the application of swarm simulations to a wider audience of artists and musicians.

1. INTRODUCTION

The swarm simulation tools and their application in musical and artistic creation and education have originated within the context of two research projects entitled "Interactive Swarm Orchestra" (ISO) and "Interactive Swarm Space" (ISS) that were conducted at the Institute for Computer Music and Sound Technology (ICST) of the Zurich University of the Arts (ZHdK). The projects have been motivated by the multitude of artistic and musical applications of swarm simulations such as [7, 8, 10, 11]. Due to the diversity and uniqueness of these approaches, the projects attempt to provide a systematic foundations for the integration of swarm simulations into creative processes in computer music and generative art. As part of this endeavor, we have explored a variety of strategies that emphasize the creative potential of capturing artistic ideas through highly customized swarm simulations rather than using standard swarms as black box mechanisms [2, 9]. Furthermore, we have developed software tools for the creation and control of simulated swarms [1, 3]. These tools try to strike a balance between a simple integration into existing musical and artistic software environments and a high degree of flexibility and customizability that characterizes scientific multi agent simulation software. Finally, we have realized several prototypical works that include musical compositions, interactive installations and dance performances [4, 5, 6].

Recently, we have started to integrate the results of the two research projects into our pedagogical activities. For this, we have modified and extended the software tools to help students to quickly gain an appreciation and understanding for the creative capabilities of swarm simulations and to be able to gradually expand the artistic and technical sophistication and uniqueness of their ideas in order to realize entire works.

This paper presents the current state and capabilities of the swarm simulations tools and discusses their application both in education and in our own artistic activities.

2. TECHNICAL DESCRIPTION

2.1. Simulation

The swarm simulation environment has started its existence as an open source C++ library that allows to develop a wide range of swarm behaviors and that provides a network communication protocol to control and retrieve simulation parameters in real time [3]. The simulation library is highly generic in that it does not pre-specify the physical or biological meaning of any of the agents's properties nor how these properties interrelate with each other via the agents' behaviors. Neighborhood relationships among agents are handled via their properties, which are assigned positions within corresponding property spaces and its these properties that "perceive" each other depending on their distance and visibility constraints. There is no limit with respect to the number of swarms that can coexist and interact with each other. Simulations can be saved to or restored from XML formatted data files at any point during the running simulation.

2.2. Control and Communication

The swarm simulation environment provides an OSC based communication protocol. This feature permits the usage of any OSC capable software to create arbitrary swarm simulations without having to resort to C++ programming. It also allows to retrieve swarm simulation data to control audio or video generation processes.

In order to trigger a simulation event, the OSC message format follows a syntax that employs the address part of the message to identify the type of event and the arguments part of the message to supply the necessary parameters for executing that particular event.

```
Syntax:   /Event_Type Parameter1
          Parameter2...ParameterN

Example:  /AddAgents "Boids" 5
```

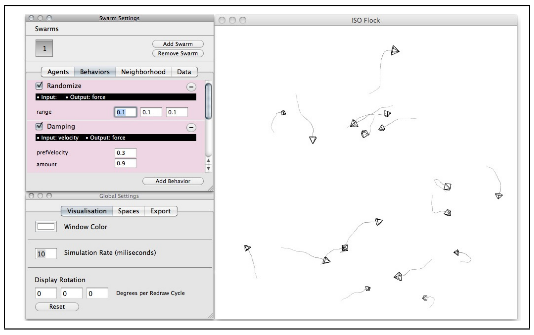
To receive OSC messages from the swarm simulation, the desired agent properties have to be registered for communication. The format of the OSC messages sent by the swarm simulation is as follows: the address part of the OSC message contains a path representing the particular agent property whereas the arguments part contains the values.

Syntax: `/SwarmName/AgentNr/Property-Name Values`

Example: `Boids/1/pos 0.1 0.4 0.3`

2.3. GUI based Control

To provide a user friendly environment for the creation and control of customized swarms, we have developed a standalone application for Mac OS X that hides the intricacies of the OSC communication syntax. This software exposes a slightly reduced set of OSC commands via a collection of graphical controls (see Figure 1). The number and types of these controls correspond to the number and types of swarm properties and behaviors. When restoring a swarm from a serialized state, the GUI modifies itself accordingly. And when manually changing, removing or adding control elements in the GUI, the corresponding properties and behaviors in the simulation are concomitantly changed, removed or added.



Randomize
• Input: • Output: force
range 0.1 0.1 0.1
Damping
• Input: velocity • Output: force
prefVelocity 0.3
amount 0.9
Window Color
10 Simulation Rate (miliseconds)
Display Rotation
0 0 0 Degrees per Redraw Cycle

Figure 1. A screenshot of the GUI application together with the visualization that is currently controlled by it.

3. LEVELS OF ABSTRACTION

The improvement and extension of the original C++ library into a software environment for remote and real time configuration and control has expanded the number of approaches with which artists and musicians can explore and integrate swarm simulations into their creative process. Each approach represents a different level of abstraction of the underlying simulation functionality and provides a different degree of flexibility and spontaneity (see table 1).

Obviously, the level of C++ offers the greatest freedom to configure a swarm simulation including the possibility to modify the core functionality of the simulation library. The main drawback of a C++ based approach lies in the required programming expertise, a necessary familiarization with the data structures and core functionalities of the simulation environment and a limited ability to spontaneously experiment with different swarm configurations.

	C++ Programming	OSC Control	GUI Control
Abstraction	low	medium	high
Flexibility	high	medium	low
Spontaneity	low	high	high
Technical Expertise	high	medium	low

Table 1. Three different approaches to configuring and controlling simulated swarms.

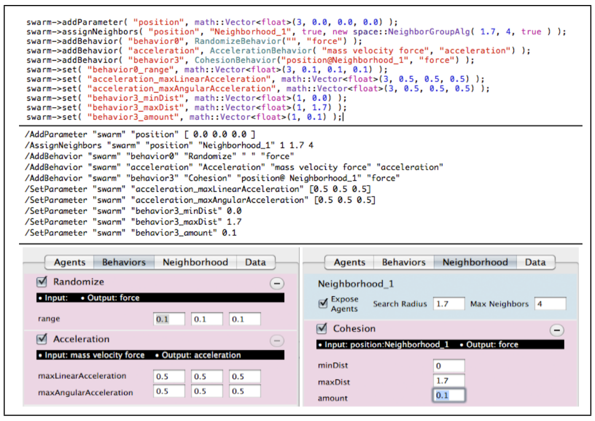
OSC commands provide almost the same degree of control as C++. The higher level of abstraction that is inherent in an OSC based approach flattens the learning curve for experimenting with swarm simulations. In addition, since OSC commands can be dispatched at any time during a running simulation, this approach allows for real time interaction and spontaneous experiments. The possibly most important benefit of using OSC lies in the fact that it enables a simple integration of the swarm simulations into whatever OSC capable software infrastructure the user is already familiar with.

The standalone GUI application does an excellent job at hiding most of the details that are required to create a swarm simulation. In particular, it automates much of the basic swarm setup procedure such as adding frequently used agent properties and behaviors and it gracefully resolves dependencies between behaviors, properties and neighborhood spaces when new behaviors are added or old ones are removed. On the other hand, this high level of abstraction limits the diversity of swarms that can be created much more than an OSC or C++ based approach.

It is important to stress the fact that these three different approaches are not mutually exclusive. It is very well possible to create a swarm simulation via C++, store the simulation in an XML file and reload this XML file into the GUI application. Or it is equally imaginable to configure a basic swarm simulation via the GUI application and subsequently modify it via OSC commands. In order to promote a flexible transition between these different approaches, we have implemented a similar usage and configuration functionality and syntax for each of them. As an illustration, Figure 2 depicts an excerpt of the same basic swarm configuration steps that have to be conducted when working with C++, sending OSC commands or using the GUI application.

4. APPLICATION SCENARIOS

The following section presents three different application scenarios in which the swarm simulation tools have been extensively used. These scenarios are a teaching situation, the creation of an audiovisual composition, and the realization of an interactive installation.



```
swarm->addParameter( "position", math::Vector<float>(3, 0.0, 0.0, 0.0) );
swarm->assignNeighbors( "position", "Neighborhood_1", true, new space::NeighborGroupAlg( 1.7, 4, true ) );
swarm->addBehavior( "behavior0", RandomizeBehavior("", "force") );
swarm->addBehavior( "acceleration", AccelerationBehavior( "mass velocity force", "acceleration") );
swarm->addBehavior( "behavior3", CohesionBehavior("position@Neighborhood_1", "force") );
swarm->set( "behavior0_range", math::Vector<float>(3, 0.1, 0.1, 0.1) );
swarm->set( "acceleration_maxLinearAcceleration", math::Vector<float>(3, 0.5, 0.5, 0.5) );
swarm->set( "acceleration_maxAngularAcceleration", math::Vector<float>(3, 0.5, 0.5, 0.5) );
swarm->set( "behavior3_minDist", math::Vector<float>(1, 0.0) );
swarm->set( "behavior3_maxDist", math::Vector<float>(1, 1.7) );
swarm->set( "behavior3_amount", math::Vector<float>(1, 0.1) );
```

```
/AddParameter "swarm" "position" [ 0.0 0.0 0.0 ]
/AssignNeighbors "swarm" "position" "Neighborhood_1" 1 1.7 4
/AddBehavior "swarm" "behavior0" "Randomize" " " "force"
/AddBehavior "swarm" "acceleration" "Acceleration" "mass velocity force" "acceleration"
/AddBehavior "swarm" "behavior3" "Cohesion" "position@ Neighborhood_1" "force"
/SetParameter "swarm" "acceleration_maxLinearAcceleration" [0.5 0.5 0.5]
/SetParameter "swarm" "acceleration_maxAngularAcceleration" [0.5 0.5 0.5]
/SetParameter "swarm" "behavior3_minDist" 0.0
/SetParameter "swarm" "behavior3_maxDist" 1.7
/SetParameter "swarm" "behavior3_amount" 0.1
```

✓ Randomize
• Input: • Output: force
range 0.1 0.1 0.1
✓ Acceleration
• Input: mass velocity force • Output: acceleration
maxLinearAcceleration 0.5 0.5 0.5
maxAngularAcceleration 0.5 0.5 0.5

Neighborhood_1
✓ Expose Agents Search Radius 1.7 Max Neighbors 4
✓ Cohesion
• Input: position:Neighborhood_1 • Output: force
minDist 0
maxDist 1.7
amount 0.1

Figure 2. An excerpt of the steps necessary for configuring a simple swarm simulation. From top to bottom: C++ source code, OSC commands, GUI controls.

4.1. Teaching

In 2011, the authors have been teaching an interdisciplinary course entitled ”Interactive Swarm Space” at the ZHdK. The background of the attending students included computer and instrumental music, new media, engineering and natural sciences. The course was based on the swarm simulation tools developed thus far and aimed at familiarizing the students with sufficient conceptual background and practical skills to allow them to realize their own swarm-based artworks and compositions. The course ended with the public performance of a total of five student works.

The students were initially presented some of the concepts of algorithmic composition and generative art in general and swarm based music and art in particular. Subsequently, they were acquainted with the software tools including the standalone GUI application, example patches in Max/MSP and SuperCollider for configuring swarms and receiving swarm data, and XCode based C++ projects for creating simple swarms. Thereafter, the students started to work on their own projects. Most of the students decided to use the standalone GUI application to design their swarm simulations and a Max/MSP patch to receive agent position data via OSC for musical and visual rendering.

4.2. Composition

The two authors have collaboratively created an audiovisual composition entitled ”Trails” that combines a live performance of acoustic instruments with an electronic playback and a pre-rendered generative film (see Figure 3). Both the electronic music part of the composition and the generative film were heavily relying on a combination of standalone GUI application and OSC based configuration and control of the swarm simulations. This approach allowed for a very flexible and interactive experimentation with different swarm behaviors throughout the creation process. For the sound synthesis part, the temporal and spatial dynamics of the swarms provided interesting forms of semi-stochastic input to the audio patches. An interesting aspect of the collaboration involved the reuse of the algorithmic temporal patterns that underly the composition's score in order to trigger behavioral changes in the swarm simulation that are synchronized to the instrumental music. This approach created an overarching coherence throughout the entire piece.

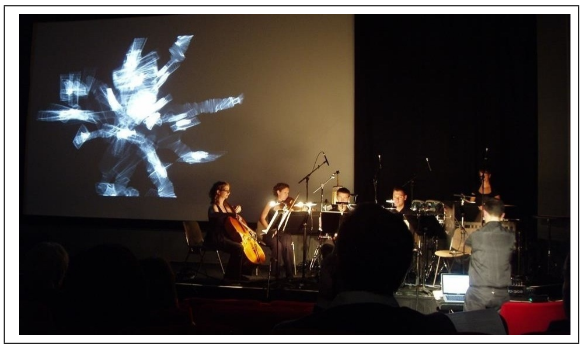


Figure 3. World premiere of Trails at the British Film Institute in London.

4.3. Installation

The installation entitled Flowspace has been realized as a collaboration between one of the authors with two members of the ICST, Jan Schacher and Martin Neukom. The installation was realized in the shape of a dodecahedron and consists of a touch sensitive surface, a 20 speakers ambisonics audio setup and three surfaces for video rear projection (see Figure 4). The installation served as a platform for the realization and presentation of interactive swarm based audiovisual compositions. So far, three different works have been realized and were shown in the context of an exhibition entitled Milieux Sonores (Kunstraum Walcheturm, Zurich, 2009 and Gray Area Foundation for the Arts, San Francisco, 2010). The installation has been extensively documented in two other publications [4, 9]. For each composition, a dedicated and highly customized swarm simulation was developed in parallel with the generative visual and acoustic programs that were controlled by the simulation. This approach helped to connect the simulation design decisions to the artistic and musical ideas that informed each of the compositions. For each of these works, the swarm simulations were programmed in C++ and subsequently diversified and serialized into different simulation states via OSC based graphical user interfaces. During the exhibition, OSC based communication provided the means to reload different simulations and states, to modify simulation properties based on the visitors' interactions, and to control the audiovisual feedback processes in a coordinated manner.

5. RESULTS AND CONCLUSIONS

The swarm simulation environment and its OSC based functionality have proven to be sufficiently robust and ver-

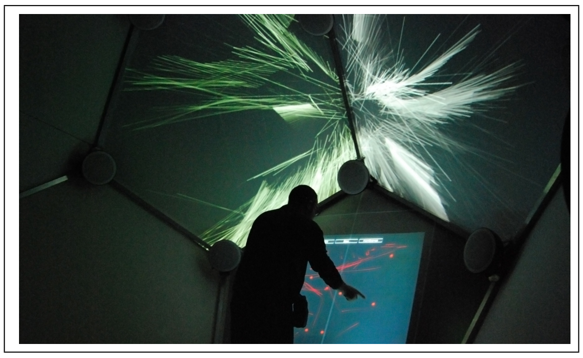


Figure 4. Interaction with the Flowspace installation. Exhibition "Milieux Sonores", Gray Area Foundation for the Arts, San Francisco, 2010.

satile to allow its usage in education and artistic realizations. The standalone GUI application complements this environment in that it not only provides a gentle introduction into the usage of swarm based simulations for users that lack programming skills, but also offers a good starting point for any artistic realizations as it allows to quickly sketch and experiment with customized swarms. Only the realization of rather exotic swarm simulations required an implementation in C++. But as these new behaviors become part of the simulation library, the limitations of a purely OSC based approach gradually decrease. The software and documentation that can be accessed via the project's website [1] keep up with these improvements.

Our teaching experience proved that the standalone GUI application is very helpful in conveying a practical understanding of the principles and capabilities of swarm simulations as it enables a hands-on approach where the students can immediately experience the effects upon changing parameters. Throughout the course, most of the students kept working solely with the standalone GUI application and did not consider to modify the simulations more thoroughly on a lower level of abstraction. They rather focused on the design of their audiovisual Max/MSP patches and would only return to experimentation with the swarm simulation itself when their envisioned result could not be achieved by modifications to those patches alone. It remains to be seen whether a prolonged use of the simulation tools will lead the students to integrate swarm simulation into their works on a more fundamental level.

As for the authors themselves, the swarm simulation tools have proven to be extremely inspiring and useful both for the realization of musical and artistic works. The tools' flexibility has allowed us to transfer a wide variety of artistic and musical ideas into swarm based approaches. Furthermore, their OSC based real time configuration and control capabilities has allowed us to creatively exploit the swarms' high level of responsiveness both in the creation process and for the final performance and exhibition situations. In their current state, the simulation tools extensively support the manual design and refinement of swarm simulations and their communication with musical and visual processes. However they do not provide functionality for automated forms of configuration and modification via mechanisms of adaptation such as evolution or learning. Also, the tools fall somewhat short in the creation of simulations that gradually change over extended periods of time and thereby provide the opportunity to experiment with emergent macro scale structures in musical compositions and visual designs. Accordingly, we plan to include adaptive mechanisms in future versions of the simulation tools.

6. REFERENCES

[1] http://swarms.cc.

[2] D. Bisig and M. Neukom, "Swarm based computer music - towards a repertory of strategies," in *Proceedings of the Generative Art Conference*, Milano, Italy, 2008.

[3] D. Bisig, M. Neukom, and J. Flury, "Interactive swarm orchestra - a generic programming environment for swarm based computer music," in *Proceedings of the International Computer Music Conference*, Belfast, Ireland, 2008.

[4] D. Bisig, J. Schacher, and M. Neukom, "Flowspace – a hybrid ecosystem," in *Proceedings of the New Interfaces for Musical Expression Conference*, Oslo, Norway, 2011.

[5] D. Bisig and T. Unemi, "Swarms on stage - swarm simulations for dance performance," in *Proceedings of the Generative Art Conference*, Milano, Italy, 2009.

[6] ——, "Cycles - blending natural and artificial properties in a generative artwork," in *Proceedings of the Generative Art Conference*, Milano, Italy, 2010.

[7] T. Blackwell and P. Bentley, "Improvised music with swarms," in *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002.

[8] J. E. Boyd, G. Hushlak, and C. J. Jacob, "Swarmart: interactive art from swarm intelligence," in *Proceedings of the 12th annual ACM international conference on Multimedia*, 2004.

[9] J. Schacher, D. Bisig, and M. Neukom, "Composing with swarm algorithms - creating interactive audio-visual pieces using flocking behavior," in *Proceedings of the International Computer Music Conference*, Huddersfield, England, 2011.

[10] D. Shiffman, "Swarm," SIGGRAPH emerging technologies exhibition, 2004.

[11] R. Vitorino, "Self-organizing the abstract: Canvas as a swarm habitat for collective memory, perception and cooperative distributed creativity," *CoRR*, 2004. [Online]. Available: http://alfa.ist.utl.pt/~cvrm/staff/vramos/Artsbot.html