

## TOWARDS PHYSICS-BASED CONTROL AND SOUND SYNTHESIS OF MULTI-AGENT SYSTEMS: APPLICATION TO SYNTHETIC HAND CLAPPING

Cumhur Erkut

Helsinki University of Technology  
Laboratory of Acoustics and Audio Signal Processing  
P.O. Box 3000, FI-02015 TKK, Finland

<http://www.acoustics.hut.fi>

[Cumhur.Erkut@tkk.fi](mailto:Cumhur.Erkut@tkk.fi)

### ABSTRACT

Physics-based sound synthesis methods excel in their efficiency, expressiveness, and intuitive control, but they usually focus on a single sound source. Extending the control strategies to multiple sound sources is not straightforward because of the self-organization and complex group behavior exhibited by a large population of objects. By regarding each object of such a population as an agent, this contribution systematically evaluates the requirements and design principles of a multi-agent system for sound synthesis and control. In a restricted domain of hand clapping synthesis, it introduces a software application that partly corresponds to these requirements and principles. A list of updates for a better match has been deduced and prioritized. The subsequent versions of the software will incorporate these updates.

### 1. INTRODUCTION

The complex interaction between synthesis and control determines the quality of a sound synthesis technique [1]. Physics-based methods excel in their efficiency, expressiveness, and intuitive control [2, 3]. Currently, with an efficient physical model and a matched control interface, most orchestral instrument sounds can be synthesized in real-time. In addition, physics-based techniques have been applied to percussive or non-musical, everyday sound sources [4, 5, 6]. Meanwhile, systematic efforts have been devoted to control in the course of realizing the full potential of the physics-based models [7, 8].

Besides the gestural control stream provided by a performer, a block-based control hierarchy have been reported in [6]. In this hierarchy, a higher-level stochastic control block governs the dynamics of a simplistic, low-level synthesis block. This approach allows the synthesis of ecologically plausible, organized macro-sound events up to tens of seconds based on physical principles. Nevertheless, the physics-based control methods have a common property in general; they govern the behavior of a single sound source.

Extending gestural or hierarchical strategies to the control of multiple sound sources is not straightforward, as animate and inanimate objects tend to exhibit various degrees of self-organization [9]. On the other hand, multiple control streams are well-defined in ecologically-based *granular synthesis* [10], where three types of stream generators are reported:

1. a single stream generator for bouncing sounds and rugged texture
2. parallel independent, phase-asynchronous stream generators for sounds like water streams
3. parallel phase-synchronous stream generators for dense sounds, such as the wind.

Hierarchical control blocks similar to the first and second type of stream generators are common in physics-based sound synthesis [4, 5, 6]. More recently, physics-based control strategies similar to the third type have been reported for synthesis of hand clapping sounds [11]. Relying on the separation of sound synthesis and event generation, these strategies can produce individual hand-claps, or the asynchronous/synchronized applause of a group of clappers.

The synthetic hand clapping can be considered a first step towards physics-based control and sound synthesis of a *multi-agent* system (MAS) that models self-organization and complex group behavior of a large population of simple objects, i.e., agents [12]. However, there are a number of unsolved problems in combining global, event-based interactions of agents with local, signal-based interactions of synthesis modules in modular physics-based sound synthesis systems [13]. The main motivation of this paper is to identify these problems, compare them to the current release of our hand clapping synthesizer, *ClaPD*<sup>1</sup>, and provide so-

<sup>1</sup>ClaPD is a Pure Data (pd) library [14] for hand clapping synthesis and control. It is one of the two software implementations discussed in [11]; the other is *ClapLab* by Perry R. Cook. While the multi-agent control strategies discussed here could be applied to ClapLab as well, this paper solely focuses on ClaPD. ClaPD is released as a free software under the GNU Public License (GPL). The current version (0.1) of ClaPD can be downloaded from <http://www.acoustics.hut.fi/go/clapd>



to update the control and audio parameters during the design phase or in run-time. The basic roles of the actors are as follows.

The designer constructs the swarm-based sound synthesis model. She starts with the design of individual agents and observers, i.e., by defining their state and behavior. Next, she determines how to map the observed data onto the auxiliary control and synthesis blocks, and how the performer will interact with the model. The designer then focuses on the interaction of agents by enlisting their actions, and schedules these actions. The agents and schedules make up a swarm by definition, however if desired, the designer further reshapes the swarm behavior. Finally, she places the swarms into an environment. The performer interacts with the swarm and its agents and changes the auxiliary parameters in run-time within the limits the designer has set. An agent acts according to its programmed behavior and generates events, the observers probe the states of the agents and drive the auxiliary control and synthesis blocks.

## 2.2. A use-case scenario

Consider a swarm of clappers. The designer defines an agent's action as clapping and moving in a circular trajectory by a particular tempo, for instance  $k * BPM_{swarm}$ , where  $k \in \{2, 3, 4, 5, 6\}$  is a state variable of each clapper and  $BPM_{swarm}$  is the consensus tempo of the swarm; both controllable by the performer. She defines the schedule as follows: after five individual claps, an agent tries to adopt the tempo of its nearest neighbor. The swarm is constructed by arranging the agents around a circle, and by placing a set of microphones and a coordinate probe inside the circle. The designer puts the swarm in a virtual room.

The next step is the design of auxiliary blocks. The designer creates three instances of a modal resonator: a hand clap resonator, a small wooden ball, and a large metallic bowl. She designs two sonic objects: the first drives the clap resonator directly with the agents' events as impulses, the second throws the ball inside the large bowl when a particular clapper claps, and controls the trajectory of the ball with an auxiliary control block. She lets the performer choose that clapper. Focusing on the environment, the designer simulates the virtual room with an artificial reverberation unit and, by accessing the coordinate probe, she places the clappers into a three-dimensional, immersive multichannel audio environment.

The performer lets the clappers clap, throw balls, and move. She updates the swarm states  $BPM_{swarm}$  and the radius, the clapper states  $k$  and the trajectory direction, and the position of the microphone probes. She also decides which clapper should throw a ball, and morphs the small wooden ball into a metallic cube. The clappers act according to their simple behavior, but collectively as a swarm, they create complex polyrhythmic sonic events.

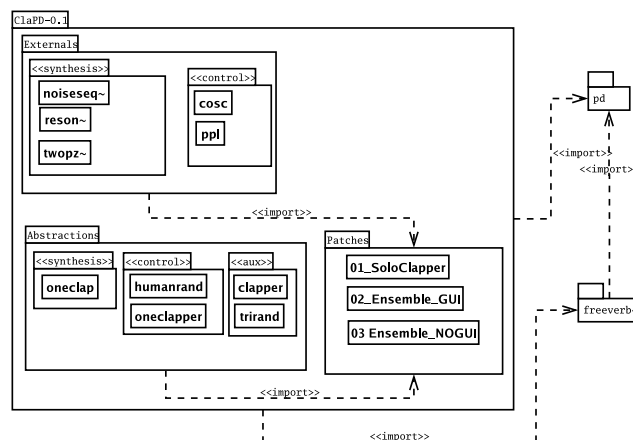


Figure 2: Package diagram of ClaPD. ClaPD consists of a set of externals, abstractions, and main patches, which are depicted as sub-packages in the diagram. The synthesis and control objects are grouped as templates. ClaPD imports pd-API and freeverb~ packages.

## 3. APPLICATION TO SYNTHETIC HAND CLAPPING: CLAPD

### 3.1. Overview of ClaPD

ClaPD is a stochastic model in realm of [5, 6]. It contains low-level synthesis and higher-level control blocks, and primitive agents for event generation, which are fine-tuned by hand-clapping statistics. ClaPD can produce expressive, human-like synthetic applause of a single clapper with adjustable hand configuration, or asynchronous or synchronous applause of a clapper population, but currently it cannot simulate a complex multi-agent scenario as the one described in Sec.2.2. The single clapper mode of ClaPD is discussed in detail in [11]. The main focus of the present paper is the second mode of ClaPD. The clapper population is called *audience*, which consists of individual *clappers*. The clappers do not interact, but listen to a *master clapper* and adjust their clapping rates accordingly.

### 3.2. Structure of ClaPD

ClaPD consists of a set of externals, abstractions, and main patches. Although pd hardly supports object-oriented software design paradigms in its API or graphical programming language levels, following the tradition of the pd community, the components of ClaPD are called and treated as *objects* in the sequel, and the library itself as a package. The package diagram of ClaPD is shown in Fig.2.

The low-level synthesis block in ClaPD follows the *excitation/resonator* paradigm of physical modeling [13]. The externals *noiseseq~* and *twopz~* generate and shape the ex-

citation signal, respectively, and `reson~` is the resonator. The parameters of these synthesis blocks are drawn from a triangular distribution that is implemented as an auxiliary abstraction (`trirand`). This allows a small variation between each clap. The parameters of the triangular distribution correspond to the hand configuration prototypes, and they are extracted from the analysis of recorded hand claps in an anechoic chamber [11]. The abstraction `oneclap` accommodates all the other synthesis objects and provides the main synthesis block in the ClaPD; there is only a single synthesis block regardless the number of clappers.

The auxiliary block `clapper` is the fundamental primitive agent in ClaPD; its state is the delta-time until its next clap in milliseconds, and its behavior is to update its own state. The clapping mode of the audience (asynchronous or synchronous) is user-controlled by a switch and perceived by the `clapper`.

In the asynchronous mode, the abstraction `humanrand` provides the `clapper` necessary parameters for its actions by imitating the temporal fluctuations of a human clapper. It calls the overloaded `trirand` and draws the delta-time from the triangular distribution `Tri(220,70)`. This distribution corresponds to the natural asynchronous clapping period of the human clappers [11]. Special rules apply at the beginning and the end of clapping sequence; in the first two seconds the delta-time is expanded by % 10, and in the last two seconds by % 2.

In the synchronous mode, the clappers listen to the master clapper, which is a native pd `metro` object that ticks with a fixed period around 440 ms. This imitates the period doubling of human clappers during synchronization [11]. When the master claps, a clapper calculates its offset relative to the master and determines its behavior with the help of the external object `cosc`. This control block is the most complex object in ClaPD; depending on the *entrainment* parameter [11] of the audience and relative offset between the master and a clapper, it enforces the clapper to accelerate/decelerate. Moreover, when the mode is switched back to asynchronous, `cosc` decouples the clapper from the master and accelerates its state until a natural clapping rate - drawn from `Tri(220,70)` - is achieved. The detailed dynamics of `cosc` operation can be found in [11].

The remaining objects within the library are three main patches, the external `pp1` that controls the number of clappers within the audience and the abstraction `oneclapper` that generates the control stream in single clapper mode, which is not discussed here. One of the three main patches loads the required objects and provides a run-time interface to the user. The patch `SoloClapper` is a single clapper model, whereas the other two are the models of asynchronous/synchronous applause of an audience. Both ensemble models synthesize audio streams, both only one of them (`Ensemble-GUI`) provides primitive visualization by

flashing widgets. This patch is depicted in Fig.3 and discussed in detail in the sequel, where the *user* corresponds to the performer actor of Sec.2.

### 3.3. Current use-cases of ClaPD

In the screenshot of the patch `Ensemble-GUI` in Fig.3, the `clapper` abstraction is expanded at the lower-right side of the figure. All the patches optionally include artificial reverberation to represent a listening environment via the pd external library `freeverb~`.<sup>2</sup> The parameters of `freeverb~` can be controlled by the user, and its operation can be bypassed with a switch. The output of the reverberation can optionally be written to a file via an auxiliary block.

The audio stream fed into `freeverb~` comes from the abstraction `oneclap`; the main synthesis block of the ClaPD. The user can set the hand configuration parameters by expanding this abstraction. The synthesis block receives the clapping events from individual clappers of the audience.

The clapping events are probed by a GUI widget connected to each clapper, it flashes when the clapper sends a clap event. The maximum number of clappers are fixed at 60, but the user is allowed to set it to a lower value using the interface block. Note that the simple visualization in ClaPD is not the most efficient one: with 60 clappers it uses the half of the available CPU power on a Pentium 4 machine, whereas the audio calculations use only about %3. Without visualization, the patch `Ensemble-NOGUI` can accommodate a much larger audience.

By using the interface, the user can start and stop the applause, determine the asynchronous/synchronous mode, set the level of entrainment, and the number of people at the run-time. Note that all these operations act on the the audience, the user has no means to access to clapper states or to influence its behavior. This restriction and others are the subject of the next section.

## 4. EXTENSION OF CLAPD TOWARDS A MAS

Fig.4 depicts the use-cases of ClaPD. This diagram is a adapted version of the use-case diagram of a physics-based MAS in Fig.1, according to the ClaPD description in Sec.3. The elements of the diagram are color-coded according to their correspondence to the MA-system depicted in Fig.1; they also indicate the requirements. The difficult cases cumulate around the designer; the current version of the ClaPD implements hard-coded statistical algorithms particularly adjusted for hand-clapping synthesis. This leaves not too much space for the designer; ClaPD is mainly user-oriented. Moreover, generalization of its algorithms for the control of other

<sup>2</sup>`freeverb~` is an artificial reverberation external for pd and Max/MSP, written by Olaf Matthes and released as a free software under the GPL. The current version (1.2) of `freeverb~` can be downloaded from <http://www.akustische-kunst.org/maxmsp>

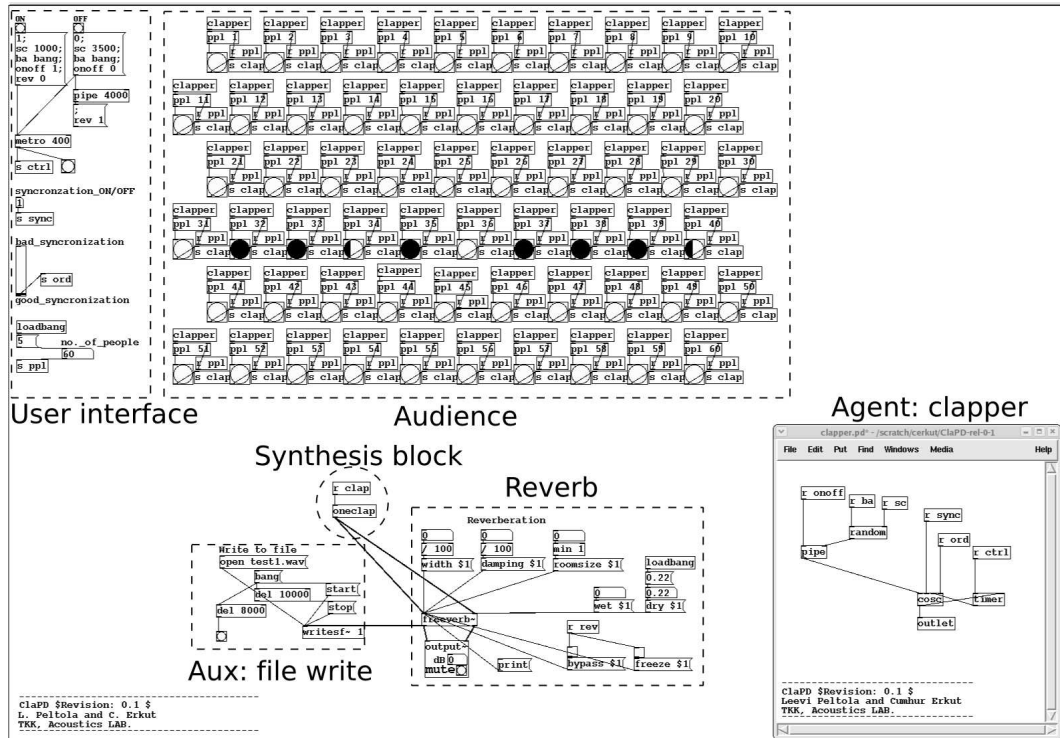


Figure 3: ClaPD in action: the main audience patch with GUI support. The widgets connected to the clappers flash when they clap. At this instant, six of the clappers are fully synchronized, and two are catching up.

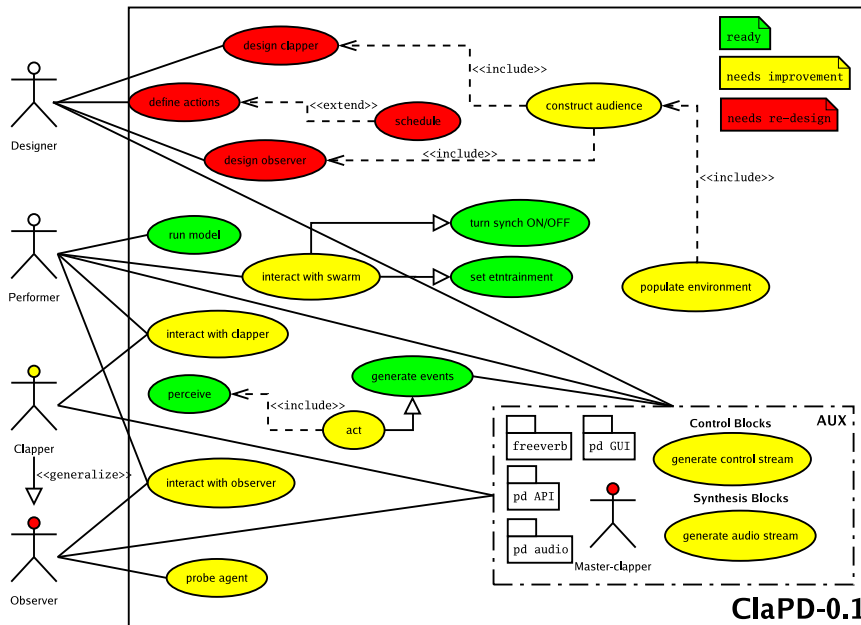


Figure 4: Use-case diagram of ClaPD. The use-cases, actors, and auxiliary elements are color-coded according to their correspondence to the MA-system depicted in Fig. 1.

sound sources is a difficult task. On the other hand, the low-level synthesis block of the ClaPD can easily be replaced by any other one.

The master clapper is an auxiliary agent that should be pruned in the future. Instead, the clappers should interact without any supervision. Currently, a master clapper rate different than 440 ms effects the dynamics of the audience in an unpredictable fashion<sup>3</sup>. The master clapper can be pruned, for instance, by implementing the algorithm reported in [16].

The construction of the audience and modeling the environment are only partly addressed in ClaPD. The clappers can be dynamically created, connected, and disconnected to a topology defined by their coordinates. The audience thus constructed can be placed on a two-dimensional grid together with the observers. This could allow also a better visualization strategy. A similar topology can be used to allow clappers to interact, and to set the level of their mutual influence. The tools needed for these extensions are readily available as externals at the pd repository<sup>4</sup>.

These requirements map to the following prioritized task list by taking the importance and the development effort into account:

- Implement a better visualization strategy
- Generalize the synthesis strategy
- Simplify the state and behavior of the clapper and allow user control
- Prune the master clapper
- Improve the modeling of the audience and the environment

These tasks are crucial yet small steps towards a fully modular, multi-agent physics-based authoring, control, and sound synthesis system that allows the composition of music interaction [17]. Tim Blackwell has done exhaustive work in this direction, albeit not fully physical at the synthesis and control levels. His work is described in detail at <http://www.timblackwell.com/>

## 5. CONCLUSIONS

This contribution has systematically evaluated the requirements and design principles of a multi-agent system for sound synthesis and control. Based on UML paradigms and Swarm - a software platform for multi-agent simulation - the domain-specific use cases are hypothesized. A software implementation for synthesis and control of hand clapping sounds called ClaPD has been introduced. A comparison of these

<sup>3</sup>The screenshot in Fig.3 is actually taken during an experiment, where the master ticks at 400 ms instead of 440 ms.

<sup>4</sup>It resides at <http://pure-data.cvs.sourceforge.net/pure-data/externals/> Relative links to the particular libraries of interest are `grill/dyn` and `unauthorized/grid`

systems reveals that, in its application domain, the structure and function of ClaPD corresponds to a restricted multi-agent system. The requirements for relaxing these restrictions are deduced, and they are prioritized as a reference for future releases of ClaPD. The current, as well as the future releases of ClaPD can be downloaded from <http://www.acoustics.hut.fi/go/clapd>

## 6. ACKNOWLEDGMENTS

This research has been funded by the Academy of Finland (Projects 104934 and 105651). Thanks to Leevi Peltola for developing ClaPD in a very short time-frame, GPL'ing it for extensions and maintenance, and for fruitful discussions enabling this work. Vesa Välimäki is highly acknowledged for his general support and for presenting this work at NoMuTe06. Finally, thanks to Perry R. Cook for a spatially remote, but otherwise *in-phase* collaboration during the predecessor of this work.

## 7. REFERENCES

- [1] D. A. Jaffe, "Ten criteria for evaluating synthesis techniques," *Computer Music J.*, vol. 19, no. 1, pp. 76–87, 1995.
- [2] J. O. Smith, "Virtual acoustic musical instruments: Review and update," *J. New Music Research*, vol. 33, pp. 283–304, Sept. 2004.
- [3] V. Välimäki, J. Pakarinen, C. Erkut, and M. Karjalainen, "Discrete-time modelling of musical instruments," *Rep. Prog. Phys.*, vol. 69, pp. 1–78, Jan. 2006.
- [4] P. R. Cook, "Physically informed sonic modeling (PhISM): Synthesis of percussive sounds," *Computer Music J.*, vol. 21, no. 3, pp. 38–49, 1997.
- [5] P. R. Cook, *Real Sound Synthesis for Interactive Applications*. A. K. Peters, Ltd., 2002. ISBN 1-56881-168-3.
- [6] D. Rocchesso, "Physically-based sounding objects, as we develop them today," *J. New Music Research*, vol. 33, no. 3, pp. 305–313, 2004.
- [7] P. R. Cook, "Remutualizing the musical instrument: Co-design of synthesis algorithms and controllers," *J. New Music Research*, vol. 33, pp. 315–320, Sept. 2004.
- [8] M. M. Wanderley and P. Depalle, "Gestural control of sound synthesis," *Proc. IEEE*, vol. 92, pp. 632–644, April 2004.

- [9] S. Strogatz, *Sync*. London, UK: Allen Lane, The Penguin Press, 2003.
- [10] D. Keller and B. Truax, "Ecologically-based granular synthesis," *Proc. International Computer Music Conf.*, pp. 117–120, 1998.
- [11] L. Peltola, C. Erkut, P. R. Cook, and V. Välimäki, "Synthesis of hand clapping sounds," *IEEE Trans. Audio, Speech and Language Processing*, Mar. 2007. Accepted for publication.
- [12] N. Minar, R. Burkhart, C. Langton, and M. Askenazi, "The Swarm simulation system: a toolkit for building multi-agent simulations," working paper 96-06-042, Santa Fe Institute, Santa Fe, CA, USA, 1996.
- [13] C. Erkut, "Modular interactions and hybrid models: A conceptual map for model-based sound synthesis," in *Proc. European Sig. Proc. Conf.*, (Antalya, Turkey), Sept. 2005.
- [14] M. Puckette, "Pure data," in *Proc. Int. Computer Music Conf.*, (Thessaloniki, Greece), pp. 224–227, Sept. 1997.
- [15] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall PTR, 2004.
- [16] Z. Nedá, A. Nikitin, and T. Vicsek, "Synchronization of two-mode stochastic oscillators: a new model for rhythmic applause and much more," *Physica A*, vol. 321, pp. 238–247, 2003.
- [17] M. Hamman, "From symbol to semiotic: Representation, signification, and the composition of music interaction," *J. New Music Research*, vol. 28, no. 2, pp. 90–104, 1999.