Bending Common Music with Physical models

Anders Vinjar
Institute for Musicology
University of Oslo
Oslo, Norway
andersvi@extern.uio.no

ABSTRACT

A general CAC[1]-environment charged with physical-modelling capabilities is described. It combines **Common Music**, **ODE** and **Fluxus** in a modular way, making a powerful and flexible environment for experimenting with physical models in composition.

Composition in this respect refers to the generation and manipulation of structure typically on or above a note-, phrase or voice-level. Compared to efforts in synthesis and performance little work has gone into applying physical models to composition. Potentials in composition-applications are presumably large.

The implementation of the physically equipped CAC-environment is described in detail.

Keywords

Physical Models in composition, Common Music, Musical mapping

1. INTRODUCTION

When analyzing the amount of cpu-power and cpu-time spent on music-related computing at places like IRCAM over the years a trend stands out[1]. During early years most cpu-resources were spend on traditional composition-tasks — structuring sets of notes and rhythms and other isolated musical parameters. This is not very strange. For the early computer-composer, coding restricted algorithms to calculate a limited amount of musical parameters and high-level events, with a fair chance of achieving satisfying musical results, was more feasible then attempting to develop effective algorithms to calculate the samples of musically useful sound-waves, often ending up with dull sounding results. When knowledge and technology had developed adequately, more focus shifted towards research and development of hardware for real-time DSP, software to control it effectively in performance and analysis/synthesis techniques to gain more interesting synthesis or processing of sound. In recent years however, after realtime-processing and playing of multiple channels of hifi-sound has become obtainable in consumer-level equipment, much more resources — both human research and cpu-time — are again focused on problems concerning musical structure.

[1]Computer-Assisted Composition

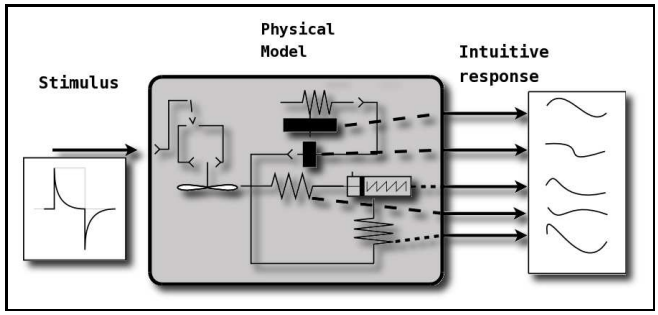

Figure 1: Intuitive response in physical model

1.1 Physical models

A physical model in this context is a computer-program made up of methods simulating nature. It simulates objects with physical properties and their behavior in time in a physical universe. Physical models have qualities which make them interesting when working on musical problems.

1. both realistic and virtual models can be made to react in ways we perceive as natural and intuitive in response to input (Figure 1).
2. because the models behave according to physical laws, and give realistic response to stimulus, we also recognize behavior in *variations* of real models as realistic.
3. physical models may be programmed so they give linear response to linear modifications. The structure of the model and the parameters of the physical forces which affect it can be modulated dynamically, and its response to stimulus changed in a predictable manner.

1.2 Using Physical Models in composition

Physical Models have proven effective in various areas of computer-music applications, and have received much attention from music-researchers and developers of music-technology. The main efforts have focused on sound-synthesis and solutions to problems concerned with musical performance. Typical musical applications are modelling acoustics or synthesizing sound, fex. room-acoustics by geometric modelling of virtual rooms[10] or virtual strings[8]. Much work has also gone into developing bio-mechanical models to generate computer-performances with humanlike expression[12], ie. *human* dynamics, ritardando etc.

The main efforts have focused on sound-synthesis and solutions to problems concerned with musical performance. Typical musical applications are modelling acoustics or synthesizing sound, fex. room-acoustics by geometric modelling

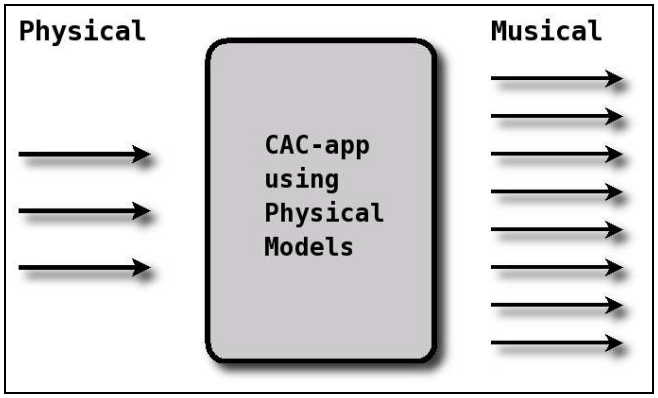

Figure 2: Black-box: "simple-to-complex"-mapping

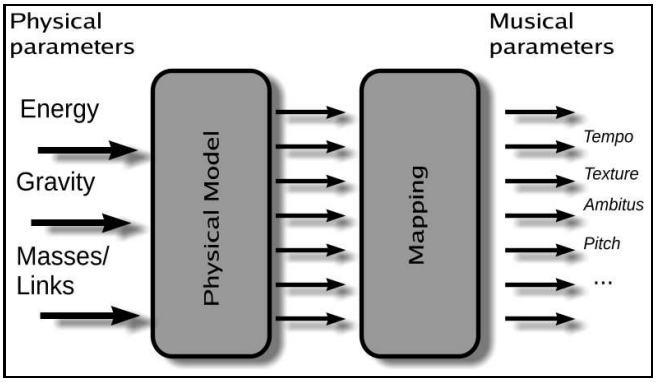

Figure 3: Simple physical control of complex musical parameter-space

of virtual rooms[10] or the Karplus-Strong algorithm[8], or developing biomechanical models to generate computer-performances with humanlike expression[12], ie. *human* dynamics, ritardando etc.

1.3 Potentials to composition-work

Relatively little research has been done on physical models in composition-work until now. Experiments has been done by the author[16] and others[3] mapping physical properties from dynamic physical models to controlling composition-parameters, suggesting that potentials in composition-applications are large. Having automatic systems controlling complex sets of musical parameters and responding intuitively to input are obviously interesting. Physical models may bring similar benefits to typical composition-tasks — structuring and disposition of material with similar or contrasting global qualities, generation of patterns with degrees of similarity, controlling evolution of interlocking sets of voices — as they have done to synthesis and performance-systems.

1.4 Black-Box architecture

A *Black Box* refers in this context to the inclusion of a construct as part of a program, where we are only interested in its input and output characteristics. Input controlling the device may be both simple and precise — and the output arbitrary complex — as long as the system reacts as we expect. This makes Black-boxes useful when controlling complex musical processes effectively.

If a black-box consists of a Physical Model — real or virtual — the response in the system will also fit with our expectations about how physical devices work. This may provide a system which is *intuitive* and easy to learn.

When composing music it is interesting to experiment with the response in the system by changing it gradually on a linear scale from obvious to unexpected. Using black-boxes also allows for mapping control-input to resulting output using any of the archetypes — *one-to-one*, *one-to-many*, *many-to-one* and *many-to-many* — mappings.

1.5 CAC-environments

Some of the powerful approaches to composition offered by modern CAC-software are rule-based and constraint-programming. Composers like them because they allow for explicitly formulating music-theoretical rules and having the computer generate music which fits, or getting a musical score automatically generated just by describing desired results. Examples of implementations of constraint-programming systems are various Constraint-libraries[9] distributed with *PatchWork* and *Open Music*, or *Common-Music*s integration with Torsten Anders' *Strasheela*[2]-environment.

In real-life situations, both constraint-based and rule-based approaches enforce strict limitations on the composer. In a constraint-based system, if a search is to find solutions at all or within acceptable time the possible set of constraints to be combined in a given search is limited. Using rule-based systems composers need to explicitly formulate detailed sets of rules and exceptions for all situations occuring in the musical processes he or she wants to prescribe.

These approaches suggests either having a well-defined musical goal to end up with, or a clear idea of how to arrive there. To facilitate creative and intuitive work with yet unknown composition-problems, and still provide a deterministic and reproducable work-flow other tools may prove useful.

1.6 Using Physical Models in CAC-environments

Typical composition-processes involve experimenting with complex hierarchies of many interdependent parameters generating musical data. Some of the wanted qualities of a good CAC-environment are:

1. simplify control of complex processes with many parameters
2. flexible and effective mappings between composers input and musical data
3. intuitive or ecological relations between cause and response in parts of the system
4. function as explorative tool
5. "tune to liking" — define and save certain personal styles, approaches, presets
6. not bias the composition-work towards predefined musical styles

Incorporating physical models as black-box-modules in existing and well-functioning CAC-environments may provide such qualities. Figure 3 shows the basis of the work-flow in such a system.

In performance-control or analysis/synthesis systems the applicable physical models are often limited to define stable and linear systems. Used in synthesis or composition-work there are no real limitations as to what kind of physical models and control-strategies are legible, since everything which comes out of the system may be considered legitimate material. However, just as in sound-synthesis — and in particular when developing the topologies and parameters of models to be used for composition — starting with

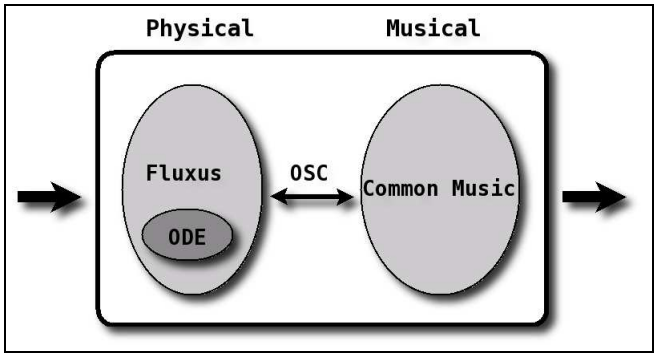

Figure 4: Components in a physically equipped CAC-environment

a model which imitates something real and subsequently modifying it may prove more effective than setting up a random topology of some virtual structure.

1.7 Related research

The questions rising when applying physical models on levels of musical structure typical of composition-tasks overlap to a large extent those studied in research on embodied music-cognition. The main efforts in this vast field are related to studies of perception and performance.

The project here is more directly related to applications to creative work. Examples of these are physical-modeling tool-kits designed for DSP-applications like Cyrille Henrys *pmpd*[7], specialized environments like ACROE's *Genesis*-project[4] or IRCAMs *Modalys*[15]-project. Some of the most relevant research is connected to using physical models together with sensor-control of virtual instruments (using the models to efficiently map sensor-data to control-data), and experimenting with bio-mechanical models in studies and simulation of human performance as in the *Samsara*[5]-project conducted by Sylvie Gibet et. al.

All work being done to understand and use physical models as black-, white- or gray-boxes in advanced control-algorithms in general[17] is of course very relevant.

2. IMPLEMENTATION

An application to aid research on physical models in composition-work is programmed by the author.

A suitable tool in this research-project shares many of the requirements of CAC-environments for working composers. The following qualities are wanted:

Effective development-environment, minimizing lag in *implement* → *test* → *response* cycle

Optimized to run efficiently on standard hardware

Flexible architecture, allowing easy modifications, updates, and even substitution of modules or components

Open Source to help sharing of development-work and experiment-results with other projects or individuals

General representation to facilitate analysis- and inter-application-work

The application is programmed as a client/server architecture consisting of two parts — *Common Music* as the client, and *Fluxus* with *ODE* built-in running as a server. The physically equipped CAC-environment is up & running, and has been used to compose musical material for new compositions.

A special OSC-protocol is developed as part of the project. Both CM and Fluxus are Scheme-based programming environments[2] and could potentially be combined in the same heap. But the way the present project is structured and the current development-state of the involved systems suggest instead to run them as separate applications and communicate through OSC.

Separating the components makes the system more flexible. CPU-load may be shared between computers and OSC-messages may be routed and sent to/from other applications. For convenience, an OSC-sequencer is programmed in SuperCollider[13] to make storing and playback of OSC-data possible out of realtime when eligible.

2.1 Common Music

CM functions as the client. Its an object-oriented music composition environment[14] with broad support for traditional musical entities. It has a well-defined API facilitating definition of for instance new I/O-classes. CM supports OSC messaging, and a real-time-scheduler is built-in. CMs has powerful support for *processes*[3] and modular pattern-generation macros. Amongst CMs interesting features are dynamic scheduling and control of processes. This makes it possible to both control any running processes, and set-up and schedule further processes based on the current run-time situation in the environment.

In this context CM takes care of musical I/O, algorithms and intermediate representations. CM receives streams of OSC-data from the physical system, in or out of real-time, and uses this input in several ways: triggering events, dynamically controlling the construction, sprouting and evolution of CM processes based on current state in the physics-system, changing dynamic variables which are looked up by already running processes etc.

2.2 Fluxus/ODE

Fluxus[6] is a real-time, graphical live-coding environment for Scheme developed by Dave Griffiths. In this project it constitutes a Scheme-controlled physics server. Fluxus can communicate with other applications via OSC network-messages and handle input from audio, keyboard or mouse. A physics engine — ODE[11] — is built into Fluxus for time-synchronous simulations of rigid body dynamics. ODE is an open source, high performance library for simulating rigid body dynamics. It is fully featured, stable, mature and platform independent with an easy to use C/C++ API. It has advanced joint types and integrated collision detection with friction. ODE is useful for simulating objects in virtual reality environments, vehicles and virtual creatures. It is currently used in many computer games, 3D authoring tools and simulation tools.

Fluxus allows visualization and graphical interaction in real-time. Besides generating interesting graphical output[4], it provides the user with useful visual feedback on the physical system.

2.3 Virtual mechanical structures

All kinds of realistic and unrealistic virtual mechanical structures are interesting to experiment with in this context. Having access to a general toolkit for rigid-body-mechanics makes all possible shapes, structures or set of structures definable in this environment available for experimentation.

[2]CM can be built using either Scheme or Common-Lisp

[3]Algorithms with built-in functionality for handling musical time

[4]Fluxus is perhaps used most as a real-time performance tool

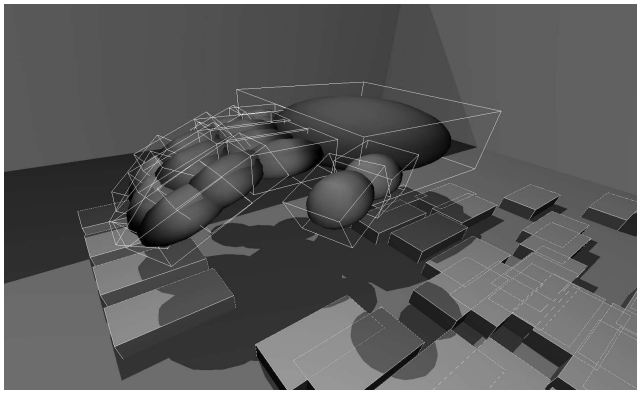

Figure 5: Virtual hand behaving in a virtual world

The interactive nature of the application suggests starting off with simple structures, observe the results, modify, and observe how the musical output changes. Systems responding realistically to input according to physical laws simplifies learning its behavior. An example of one such approach used to experiment with is a model of a hand where attributes such as length of fingers or gravity are modulatable. A certain point on the hand is subjected to externally controlled forces in three dimensions, resulting in the whole structure responding in a physically coherent way.

As part of this project virtual structures consisting of *mechanical bodies* with varying shape, mass and surface-qualities, connected by *links* of various types and qualities — eg. "balljoint", "hingejoint", "fixedjoint", "sliderjoint" — are constructed and set to interact in virtual physical worlds with arbitrary values for physical properties such as gravity or friction. All physical parameters in the system — position, speed, forces, collisions between objects, angles of joints etc. — can be read at any time, notified as OSC-messages and be mapped to musical parameters.

Different structures and modes of interaction may be saved and recalled as presets, and behavior over time may be recorded and played back.

2.4 Mapping

Special mapping-layers connecting streams of data from the physical world to musical parameters, and fitting their ranges onto eligible scales, are programmed as classes and methods in the CM-environment.

The way data-streams from the physical objects are used to control evolution of musical parameters defines the resulting music. These choices are made to suit the actual compositional problem at task.

The enhancements provided by a physical black-box included before the mapping-stage are related to how parameters evolve over time and the way simple input are used to control complex sets of parameters through intuitive one-to-many mappings.

2.5 Musical parameters

The system controls parameters on various levels as illustrated on the right-hand side of figure 3. Examples of interesting compositional parameters range from low-level ones — pitch, onset, register, dynamic — to high-level attributes such as phrasing, ambitus, texture, redundancy.

3. CONCLUSION AND FUTURE WORK

As part of this project several musical works will be composed and the physically enhanced CAC-environment will be used while composing these pieces. The use and performance of special physical topologies — and physical models in general — will be studied qualitatively and with respect to efficiency, and compared with alternative approaches to solving the same musical problems.

To ensure exchange of relevant results, the research will be done in conjunction with relevant research groups at the University of Oslo and elsewhere.

Results will be presented on various scenes, exhibitions, concerts, performances, workshops, seminars and conferences.

4. REFERENCES

[1] C. Agon, G. Assayag, and J. Bresson, editors. *The OM Composer's Book*, volume 1. Delatour France/Ircam-Centre Pompidou, 2006.

[2] T. Anders, C. Anagnostopoulou, and M. Alcorn. *Multiparadigm Programming in Mozart/OZ*, volume Volume 3389, chapter Strasheela: Design and Usage of a Music Composition Environment Based on the Oz Programming Model. Springer Berlin/Heidelberg, 2005.

[3] C. Cadoz. The Physical Model as Metaphor for Musical Creation. pico.. TERA, a Piece Entirely Generated by a Physical Model. *Proceedings of the 2002 International Computer Music Conference*, 2002.

[4] N. Castagne and C. Cadoz. Creating music by means of 'physical thinking': The musician oriented genesis environment. In *Proc. of the 5th. Int. Conference on Digital Audio Effects*. DAFX-02, 2002.

[5] S. Gibet, N. Courty, and J.-F. Kamp, editors. *Gesture in Human-Computer Interaction and Simulation, 6th International Gesture Workshop, GW 2005, Revised Selected Papers*, volume 3881 of *LNAI*. Springer, Berder Island, France, May 2006.

[6] D. Griffiths. *Fluxus, http://www.pawfal.org/fluxus*.

[7] C. Henry. pmpd: Physical modelling for pure data, 2004.

[8] K. Karplus and A. Strong. Digital synthesis of plucked string and drum timbres. *Computer Music Journal*, 7(2):43–55, 1983.

[9] M. Laurson. Pwconstraints. *X Colloquio di Informatica Musicale*, X:332–335, 1993.

[10] F. L. Lezcano. dlocsig, http://ccrma.stanford.edu/ nando/clm/dlocsig/.

[11] *Open Dynamics Engine, http://www.ode.org*.

[12] R. Parncutt. Modeling piano performance: Physics and cognition of a virtual pianist. *ICMC Proceedings*, pages 15–18, 1997.

[13] *SuperCollider, http://supercollider.sourceforge.net/*.

[14] R. Taube. *Common Music*. http://commonmusic.sourceforge.net.

[15] H. Vinet. Recent research and development at ircam. *Computer Music Journal*, 23(3):9–17, 1993.

[16] A. Vinjar. Oppspent line. MIC-recording, 1994. Musical composition.

[17] M. J. Willis and M. T. Tham. Advanced process control, April 1994. Web-document.