Using Physical Modelling for Pure Data (pmpd) with an audio and video synthesis

Cyrille Henry
(cyrille.henry@la-kitchen.fr)

Abstract

pmpd is a collection of object for pd. These objects provide real-time simulations, specially physical behaviors. pmpd can be used to create natural dynamic systems, like a bouncing ball, string movement, Brownian movement, chaos, fluid dynamics, sand, gravitation, and more. It can also be used to create displacements thus allowing a completely dynamic approach of pd computing.

With pmpd physical dynamics can be modelled without knowing the global equation of the movement. Only the cause of the movement and the involved structure are needed for the simulation. pmpd provides the basic objects for this kind of simulation. Assembling them allows the creation of a very large variety of dynamic systems .

This article describe pmpd utilisation and explore an application of this tools for real time interaction between a musician and an audio synthesis. Virtual physical structure can act as a black box between the musician and the audio synthesis: A musician can play with the movement of a virtual structure, which produces sound.

Introduction

The aim of pmpd is to create different kinds of behavior for sound and video processing using pd. Moreover, the modelling of dynamic physical systems can easily be done with pmpd.

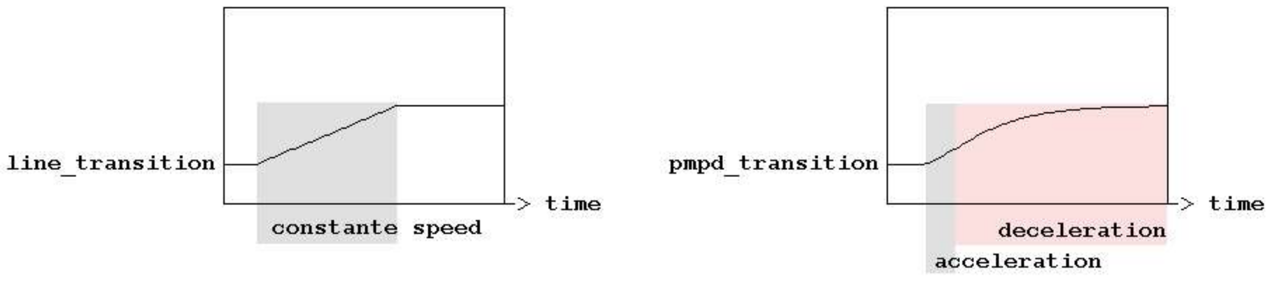
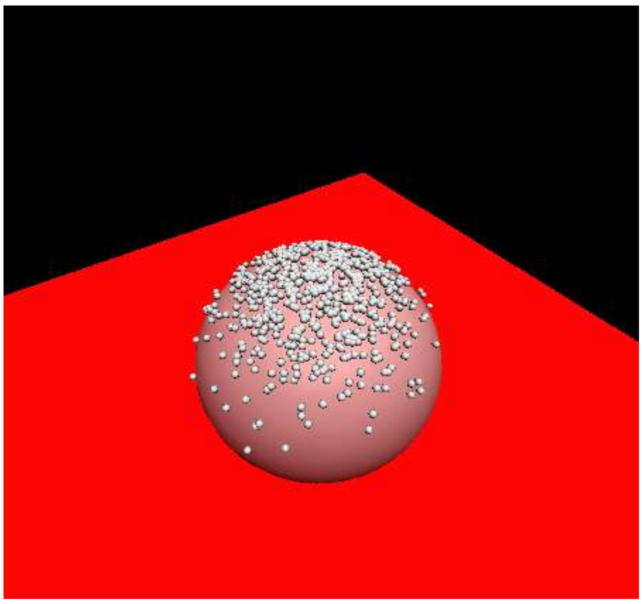


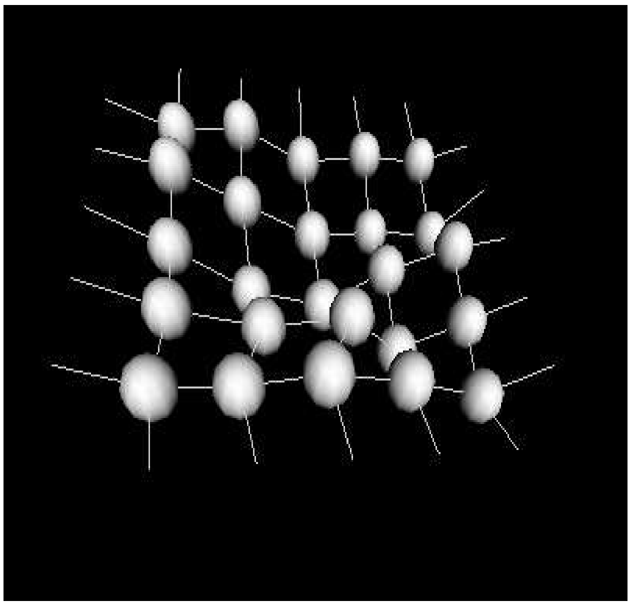
Figure 1: transition using "line" object or pmpd objects.

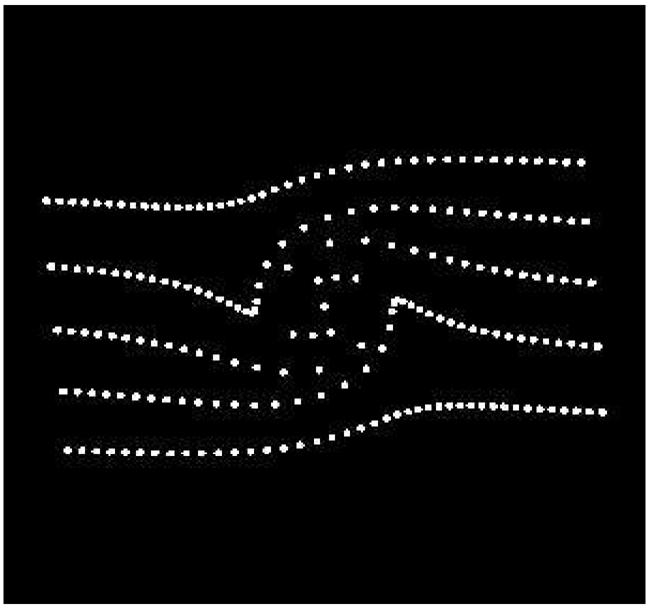
Figure 1 shows two different kind of a parameter transitions. The first transition is made with a native pd object: "line" which can go to a value in a given amount of time. The second is made using pmpd objects.
The transition made with pmpd shows a more—so to say—"natural" evolution than the first one. It shows an acceleration and a deceleration time. The shape of the transition can be changed using physical values. This idea can also be used in audio synthesis, where one may desire the sound to be modulated in a "natural" way. pmpd is just providing an easy way in pd to generate this behavior.

The pmpd objects can also be used for more complex simulation, in a 1, 2 or 3 dimensions space. For instance, simulations of gas, paste, and sand can be made in real time on common personal computers.

Examples:


a) smalls particle falling on a 3D object


b) 3D membrane


c) 2D fluid motion: vortex on a non viscus fluid

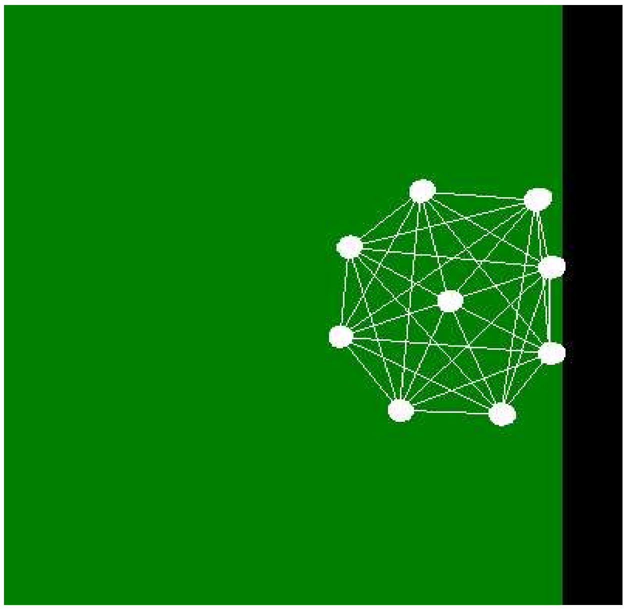

d) 2D structure interacting with a "wall"

Figure 2: Different simulation using pmpd with pd and GEM

Figure 2 shows four different kinds of simulation using pmpd.
a) A few hundred small particle falling on a distorted sphere (3D simulation).
b) A small 3D membrane composed by a set of spherical masses and links between the masses. This structure is linked to a non moving referential. Forces can be applied to one or more masses. The structure can move in the same way as a square elastic membrane.
c) A 2D fluid dynamics simulation: pmpd is used to display some fluid particles in motion.
d) The motion and rebounding of a "ball" is modelled in 2D with 9 masses.

pmpd description

pmpd is designed to provide low level comportment objects allowing particle-base physical modeling. Assembling these objects can generate complex behavior due to the interaction among the basic objects. A good knowledge of the global equation of the movement is not necessary to simulate very complex behaviors. The cause of the movement and the structure only are needed for the simulation. pmpd can then easily be used for the simulation of a very large variety of comportments.

All the pmpd objects work with control data (as opposed to audio signals). For instance, one cannot hear the sound of a vibrating string because it will not move fast enough; but one can use the movement of these particles along a string to perform additive synthesis.

Complex simulations are basically made from two kinds of elementary objects: “mass” and “link”.
“mass” objects send position and receive force from “link” objects. “link” objects receive the position of two masses and output forces for both of them.

pmpd does not use specific units.

Mass

“Mass” objects react like a point mass. “Mass” objects have inertia, but they have no volume (they cannot rotate). They take forces at their input, and output their positions.
For each time increment, position of a mass changes accordingly to the mass velocity, while velocity depends on its acceleration. The value of the acceleration is given by Newtonian dynamics.
When told, masses make the sum of the forces applied to them in order to compute their acceleration, and then deduce their new position.

“mass” objects have outlets for their internal state; they report their position, the total force applied to them and their speed. This parameters can be used to generate data controlling audio/video synthesis.

Link

“Link” objects take two mass positions and output two opposite forces depending on the relative position and speed of the masses. Links are visco-elastic connections between two masses. The force generated by a link is :

$$\vec{F} = K\,\vec{X} + D\,\vec{V}$$

where “K” is the rigidity , “D” is the dampening, “X” is the elongation of the link, and “V” is the relative velocity of two masses.

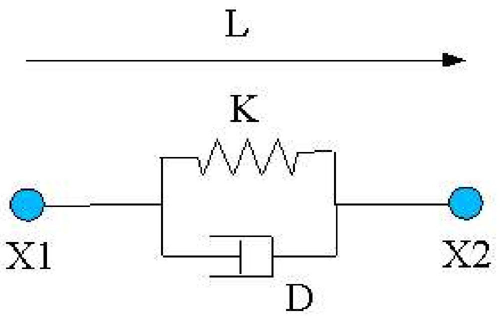


Figure 3: overview of a link

Forces and displacement

To allow real-time interaction, “mass” objects accept force messages from the user. pd can then be used to provide interaction between the user and the simulation. The extreme modularity of pd can offer a very large variety of interaction with the simulation thanks to sensors, haptic transducers, or classical computers inputs (mouse, keyboard).

Masses can also be displaced without inertia (translated), according to displacement message. This is not physical behavior, but can be useful for different reasons:

-If you know the global equation of a movement, you can use displacement message to simulate this equation (like in the provided vertex examples).

-You can use this to create non “unnatural” movements of your desire

Metronome

A time reference is needed to compute the simulation. pmpd uses an “external” scheduler. The user has to send a scheduler event to pmpd object.
This mechanism was chosen purposefully. The desired advantages were:

-You can easily change the speed of the simulation (adjust on the CPU speed).

-You can stop different parts of the simulation when you do not need them.

-You can synchronise to video rendering if desired.

-pmpd needs to compute all mass movement together and all link interaction together.

An external metronome is well adapted to this task.

The speed of the metronome should depend of the speed of the movement you want to simulate. There is no general law to know the metronome speed, but: the faster the metro, the faster the simulation can be. The metronome should be faster than the highest frequency of the movement you want to simulate.

The metronome corresponds with the time discretization of the equations.
States of the model are computed in discreet time. i.e. displacement of a mass is not a function, but a finite serial of point: X(n).
Velocity at time = t: V(t) is define by:

$$\frac{X(t) - X(t-1)}{dT}$$

dT is the small time delta between two consecutive iterations.
Acceleration can can be described in the same way. The new mass position can be computed according to it's older positions and applied forces.

Name

For patching simplification mass and link objects have a “name”. It is the first argument for the object creation. It is used to receive information (pd messages). All masses with the same name defined a class of mass.

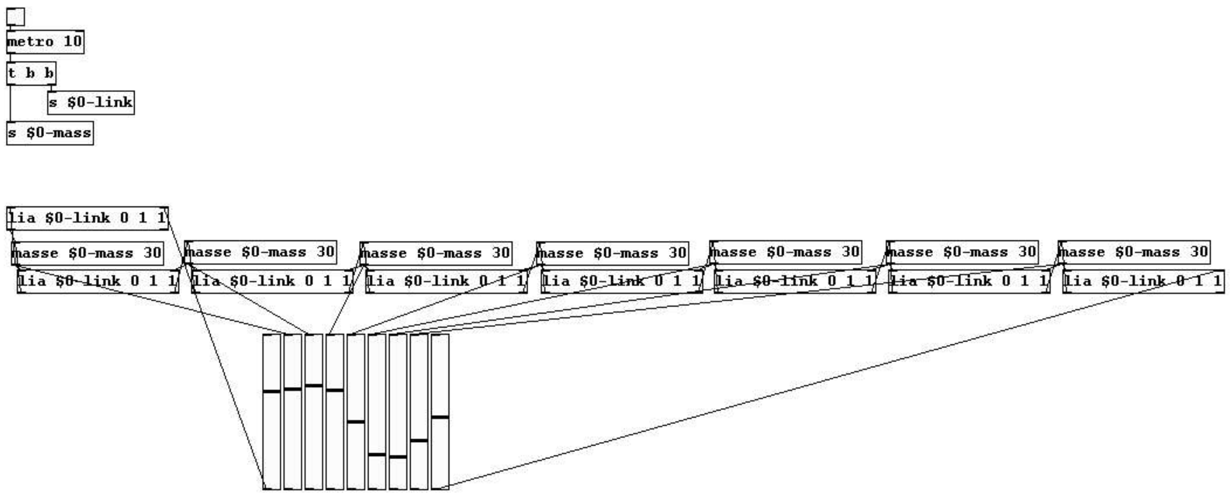


figure 4: sending bang without connection

Figure 4 shows how “bang” massages are pass to a set of masses and links without creating pd connection.
It can save patching time, and should be used for patch simplification.

Interactors objects

Interactor objects act as link object but provide patching facility.
In effect, a single object can create an interaction with a entire class of masses.
Interactor objects should be created with a name. This name is the name of the masses it is connected to.

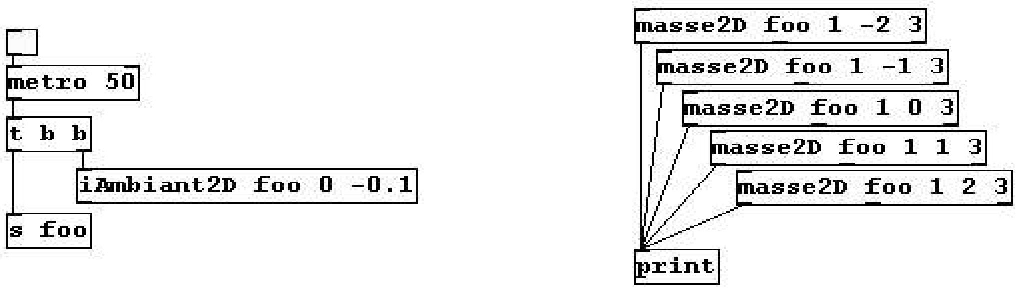


Figure 5: *sending forces to a class of mass*

In Figure 5, all masses are subject to an ambient constant force. This force can be viewed as gravity force applied to every mass named “foo”.
Masses with other name will not be sensitive to this interactor. Different interactor objects provide interactions with a point, a line and other simple primitives.

Test objects

“Tests” objects test the position—as well as distance, speed from a point, a line, orientation...—of a mass. Thereby, pd has access to much informations regarding the state of the system. This allows interaction with the rest of the patch. Another test object gives information about a link (deformation, speed of deformation, orientation...)

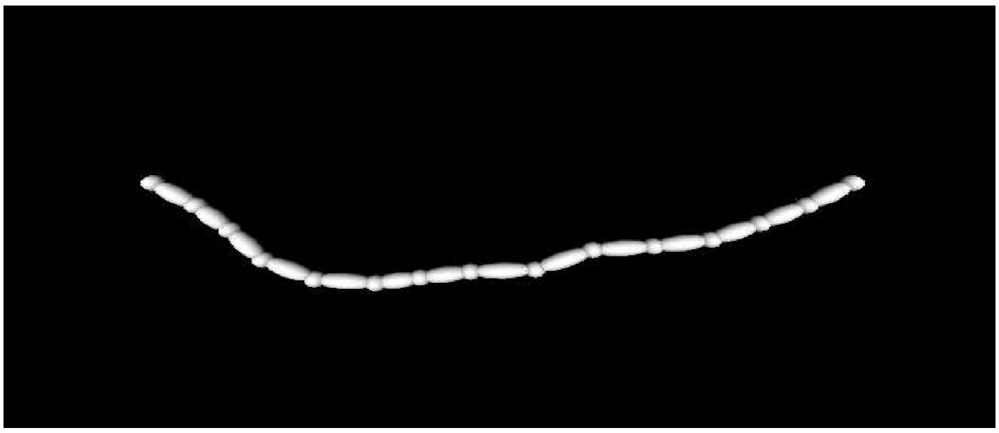


Figure 6: "tLink" example

Figure 6 shows how to use the "tLink2D" object to determine the size and orientation of a link between two masses.

connections

"Mass" and "link" objects have to be connected in the following way:

-"mass" objects send position and receive force from "link" objects

-"link" objects receive two mass positions and output forces for both masses

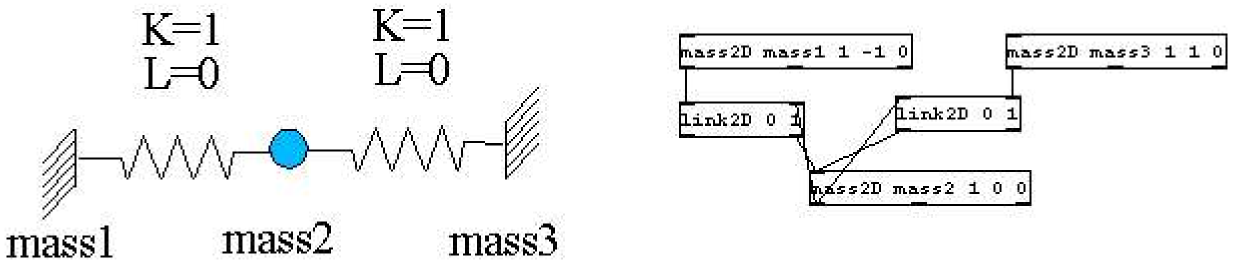


Figure 7: a small system, and it's equivalent using pmpd and pd

In figure 7, mass 1 and 3 do not receive forces (inlet are not connected), therefore they will never move; they act like a fixed point. Mass 2 is linked to 2 fixed points with springs with no damping. Hence, movement will never end (it corresponds with a system without energy loss).

Topology

A structure and its components can be modelled according to the topology : different dynamic systems can be modelled using different topologies.

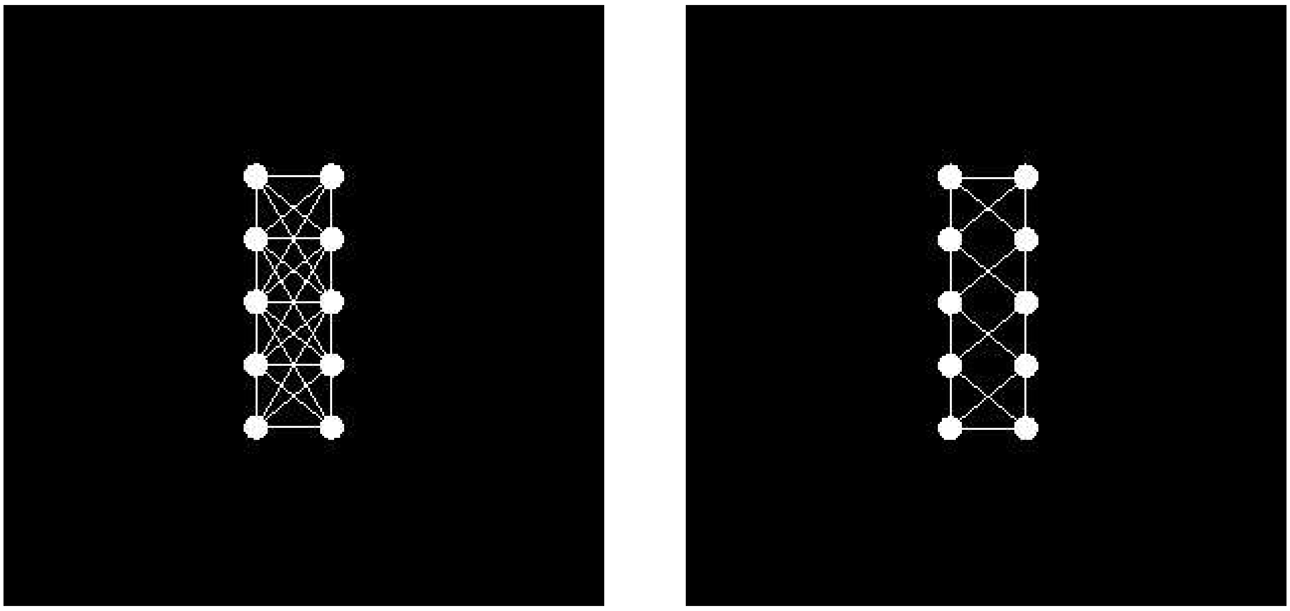


Figure 8: Two different topology for the same object.

Figure 8 is an example of a two different topologies. Masses are the spheres, link are the white lines. The same set of masses are link in two different way. The physical behavior of this structure will differ according to its topology.

Generality

Most of the objects parameters can be set at the creation of the object as argument, but they can be modified using a messages sent to the first inlet.

It is possible to choose non-physical values for pmpd parameters. For example, you can set damping to a negative value, which mean energies creation. This is not physical and can lead to instability or saturation of the model, but can be useful for artistic reason.

You should also take care while changing parameters like rigidity. This can lead to energy creation or lost, depending of the deformation of the structure.

The most important problem using physical modelling is the instability. To reduce the risk of instability you should slow down your model (increasing the metronome speed can be necessary to keep the desired speed of the simulation). Reducing force in the simulation is usually a good way to attack instability problems. Initialising masses close to a stable state can also help.

Especially with the mass class reference system, pmpd may perform many calculations at the same time. This can require much of CPU usage. Long audio buffer may help to avoid audio clicks.

The best way to understand the capabilities of pmpd is to try the set of examples. These examples provide a wild range of different applications of pmpd, and some explanation of how pmpd works. You need a recent version of pd (i.e. 0.37, not tested on earlier versions). Most of the examples use GEM for the visualisation of the virtual world. GEM 0.90 is recommended: some examples may not work on earlier version. There are no other dependencies.
GEM is then needed to use the examples. But there is no need to have gem to use pmpd:

you can use it only for audio synthesis, or with other video synthesis tool.
The aim of the provided examples is only to describe behavior of a system, not to make artful graphical effects.

Application for real time interaction with an audio synthesis

One of the applications of this kind of simulation is to create a dynamic structure that can be modified, moved, distorted by the user. This structure can then be used to “control” an audio synthesis. With the help of sensors, a user can create a virtual structure, linked to his or her own movement, to the real. The user can then play with a virtual, but “physical” instrument, allowing a natural comportment of a digital audio synthesis.

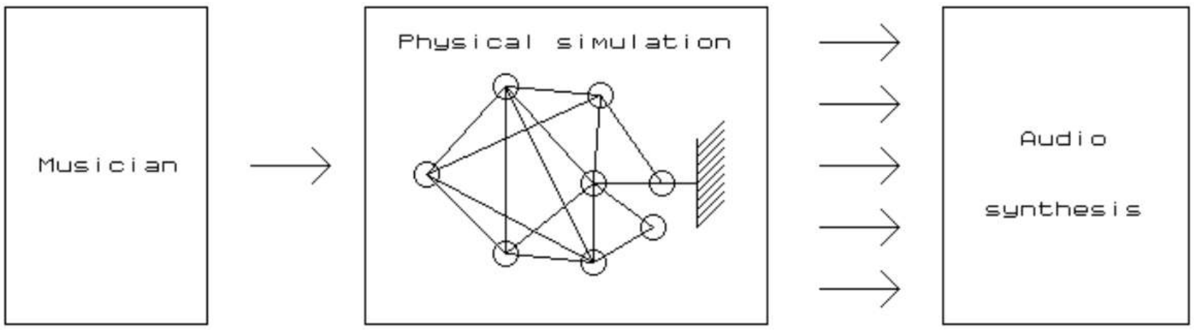


Figure 2. using physical modeling between musician and audio synthesis

The figure 2 shows a virtual structure used as a black box between user action and audio synthesis. This black box has interesting specifications. Only few input parameters can generate lots of different data flows (a musician can play with only few control parameters on the whole structure, and then generate lots of data to control any audio synthesis). Moreover, the control parameters are intuitive because they correspond to physical values. Playing with such a system can be very intuitive for the performer as the system reacts in a instinctive way. Some control parameters can change the way the structure evolves within a time period. Another important specification is that all data coming out of the physical model are not independent. The relation between them can be adjusted regarding the topology of the structure.

The most commonly encountered problem while using physical modeling is the instability. To reduce the risk of instability, one’s model should be slowed down (increasing the metronome speed can be necessary to keep the desired speed of the simulation). This is not a problem due to the relatively low frequency needed for the simulation. In this case, the structure needn’t be computed at audio rate unlike physical modeling based audio synthesis, but at only a few hundreds hertz. So, this simulation can be calculated by low-cost computer or personal laptop for live application.

Conclusion

pmpd provides basic objects for flexible behavior simulation of dynamic systems. Its basic components are specially suited for particular physical modelling.
It can be used for the simulation of many different systems without requiring knowledge of the equations describing the movement. Gravitation, structures, fluid dynamics, and Brownian movement are some examples of such systems. It provides a flexible data generating tool for controlling audio synthesis modulation, rhythmic generation, non real

time audio generation or any other musical process, but can be also used simply for data processing. Naturally, it can be use for video animations where physical behavior is desired, as well as to create smooth transitions and natural evolution of these behaviors.

pmpd is released as free software under the GPL, and has been compiled on most common platforms (Linux, Windows, and Mac osX). Binaries, sources, examples and documentation can be downloaded from : http://drpichon.free.fr/pmpd, sources are in the pd CVS.

Links

http://drpichon.free.fr/pmpd/

http://www.crca.ucsd.edu/~msp/software.html

http://web.ukonline.co.uk/taosynth/

http://www-2.cs.cmu.edu/~fp/courses/02-graphics/pdf-2up/12-physical.pdf

http://www.sodaplay.com/

http://www.pure-data.org/downloads/

http://ccrma-www.stanford.edu/~cc/misc-papers/ica04.pdf

http://perso.wanadoo.fr/gmem/evenements/jim2002/articles/L09_Castagne.pdf

http://acroe.imag.fr/ACROE/recherche/cordis-anima/cordis-anima.html

http://profs.sci.univr.it/~fontana/paper/21.pdf

http://www.music.columbia.edu/PeRColate/

http://echo.gaps.ssr.upm.es/costg6/bibliography/proceedings/djoharian.pdf

http://gem.iem.at/