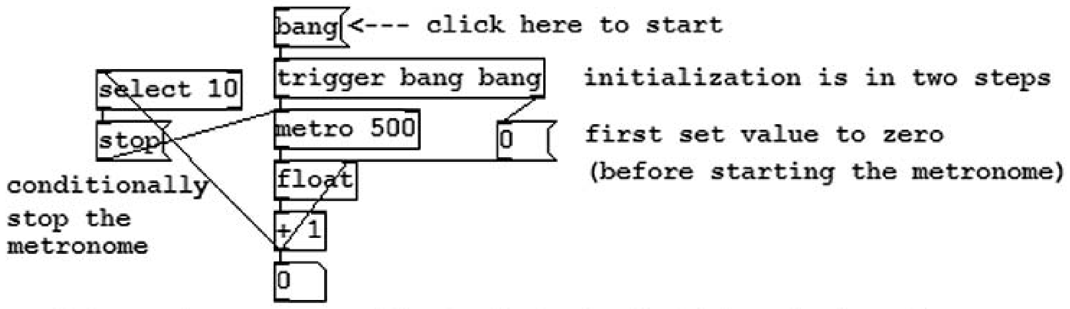
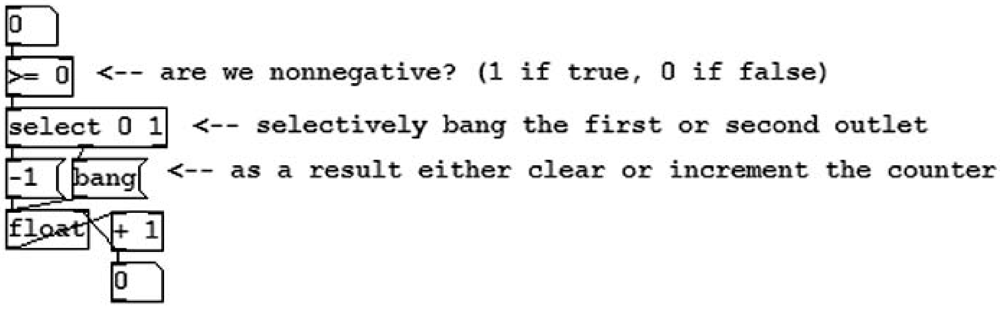
Here's a counter that counts from 1 to 10:



We're using one new object, "select, " which outputs a bang when it gets a matching value (10). This is useful for doing conditional computations, such as this one which counts while its input is 0 or positive but clears when negative:


0
>= 0
<-- are we nonnegative? (1 if true, 0 if false)
select 0 1
<-- selectively bang the first or second outlet
-1
bang
<-- as a result either clear or increment the counter
float
+ 1
0

updated for Pd version 0.34

Basic Physical Modeling Concepts

Cyrille Henry

Introduction

A computer processing unit (CPU) is a device able to perform mathematical operations with data stored as electrical voltage. It can only perform simple operations, but so many of them can be done in a second that it is possible to create a program (a defined combination of operations) to perform complex tasks. However, the processing unit of a computer is limited to processing virtual information that is electrically coded inside the computer. This information can be connected to the real world through an interface (a keyboard, a screen, etc.) but the main processing unit is not subject to physical rules like human being is. It is obvious that gravity has no effect on the way a computer performs its task; this also means that the computer has no idea what gravity is.

If you want your computer to move a fader like you do, you have to teach it how to do so. You must also teach your computer how gravity, the fader, and your muscles interact with each other in order to change the position of the fader.

First, we will compare and analyze the way a human moves a single fader (such as a mixer fader) with the simplest way a computer moves a virtual fader. Then we will explore certain physical modeling tools as a solution to teach a computer the basics of physical notions and will give a short overview of the possibilities of such tools. This article is not a complete mathematical or physical justification of particular physical modeling; rather it introduces basic ideas in order to allow anyone to use these tools.

Human / Computer Differences

Unlike computers, human beings live in a world with physical rules. Gravity, forces, and energy control their movements. The difference can be shown by studying the movement of a simple fader over time. A mixer fader can be used to change the amplitude of a sound, its frequency, or any other audio synthesis factor. We will focus only on the position of the fader and not on the effect of this movement on the synthesis.

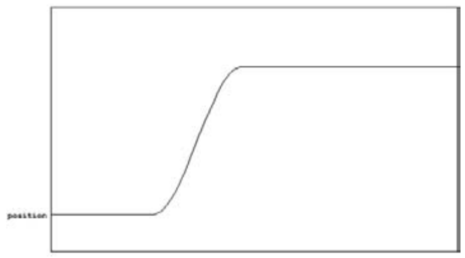


Figure 1. Human fader movement over time

Figure 1 is a typical example of a fader movement over time. The fader is moved by a human.

In this simple example, we can compute the velocity of the fader (how fast the position of the fader changes) and then its acceleration (how fast the velocity of the fader changes).

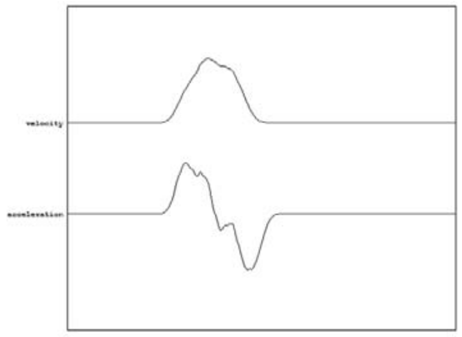


Figure 2. Human fader movement: the velocity and acceleration of the fader over time

Notice in Figure 2 that the velocity of the fader gradually increases from 0 to a maximum velocity, after which the velocity is almost stable for a short time before returning to zero. The acceleration is positive during the first part of the movement (while the velocity increases), almost null in the second part, then negative during the last part of the movement (when the velocity returns to zero). The movement is composed of these three periods.

Figure 3 represents the position, velocity, and acceleration of a virtual fader moved by a computer using pure-data.

The position of the fader changes instantaneously from one value to another. This very fast movement is highly unnatural. We will try to slow down this movement using a [line] object. This object slows to a specified target over a given time. It offers a linear variation of its output.

The movement in Figure 4 still looks different than the one made by a human.

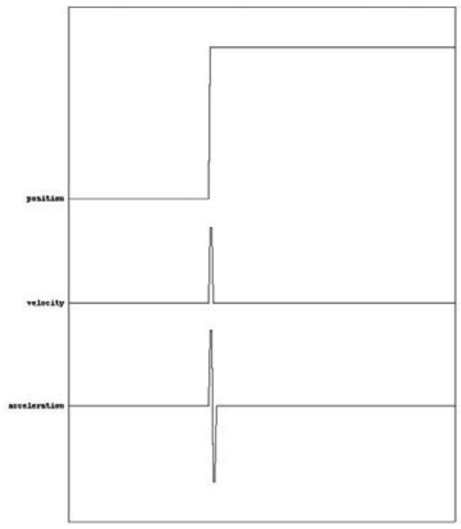


Figure 3. Switch from one value to another with a computer

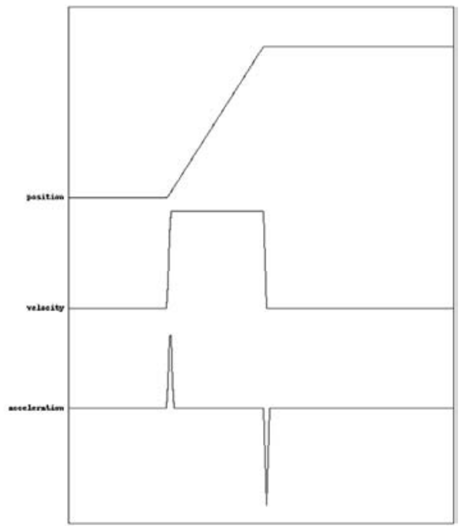


Figure 4. Computer fader movement: the velocity and acceleration of a fader over time

In the example in Figure 4, the fader has neither an acceleration nor deceleration period. The velocity changes almost instantaneously. This movement looks very "robotic" because the fader acceleration or deceleration is concentrated at the beginning and at the end of the movement. In a real movement, this acceleration and deceleration continues throughout almost the entire movement. To compensate for this very short acceleration and deceleration time, the acceleration is very great. The computer does not have any problems moving the fader in this unusual way because it only moves a virtual fader. Unlike human movement, no energy is involved in this virtual motion.

Simple pmpd Description

Aim of pmpd

pmpd provides objects for the simulation of physical behaviors such as basic energetic and movement related objects. These are mass and link, allowing particle-based physical modeling. They simulate the behaviors of a single mass or link (a link is, for example, a spring between two masses) and react like ideal objects in a physical world. Just as a CPU performs complex tasks with simple instructions, assembling this small "physical" element can generate complex behaviors due to interactions. pmpd allows the user to create a network of masses and links, to simulate the behaviors of this system, and to interact with it in real time while sending forces or changing a physical parameter.

Mass

Masses are objects that react like a virtual mass. They are defined only by their position in space and their weight. The only thing a mass can do is to move according to Newtonian dynamics. This means that applying a force to this mass will change its velocity. The [mass] object has one inlet to receive forces and one outlet for its position (another outlet provides information not essential for the simulation). The pmpd mass is punctual, which means that it has no volume and cannot rotate. Although this does not exist in the physical world, it can still be a good approximation. We will see later how to simulate massive objects.

The physical notation of Newton's law is $F = ma$ where:

F = the sum of all forces applied to a specific mass
m = the weight of the mass
a = the acceleration of the mass

This physical law means that the acceleration of any mass is proportional to the force applied to it, but it also depends on the weight of the mass. This equation rules all non-relativistic physics and is the only equation for the movement of mass.

Link

A [link] object creates a connection between two masses. It allows the masses to interact with each other, i.e. to exchange energy.

A link needs the position of two masses in order to compute the two different forces to be applied to the two masses. Since the forces depend on both the position and the velocity of each mass, the [link] object has two inlets and two outlets.

A link can be elastic if forces depend only on the distance between the two masses. The force of an elastic link is proportional to the elongation of the link. The elastic link does not introduce any energy loss but can convert kinetic energy (energy related to movement) and potential energy (energy related to a deformation).

The viscosity link is related to energy loss. The forces, proportional to the damping, are opposite to the movement of the mass.

pmpd links are all visco-elastic, the sum of an elastic and a viscous link.

The equation of a link is $F = KL + DdL$ where:

F = force generated by a link
K = rigidity factor
L = elongation of the link
D = damping factor
dL = elongation variation of the link

This link is symmetric, so the forces sent to the two masses are opposing.

Metronome

Due to computer specifications, it is not possible to compute the position of a mass at all times. The position of a mass is therefore computed at a specific time, and a time reference is needed to perform a simulation. pmpd uses an "external" scheduler (a metronome). This metronome corresponds to the time discretization of the physical equations. This is another approximation of real behavior. To be accurate, this time discretization should be the smallest possible. The speed of the metronome depends on the velocity of the movement you want to simulate. The metronome should be faster than the highest frequency of the movement: the faster the metronome, the faster the simulation. Of course, the faster it is, the more positions need to be computed for each mass, resulting in more computing time.

Units

Just like Gem, the Pure Data OpenGL extension, pmpd does not use specific units; you can choose your own. If you must use units, you should try to use consistent ones, i.e. if you chose inches for the distance unit, then the unit of rigidity should be the unit of force divided by inches. Since everything is relative, a mass weighting "10" will react the same way to a force of "1" as a mass of "200" to a force of "20".

One, two, or three dimensional network

A pmpd physical model is composed of virtual objects (masses and links) that do not have any specific representations. In order to visualize the physical model, a mass can be represented by a circle or sphere and a line by a link. This is just a representation of the physical model, but another representation can be chosen for any system.

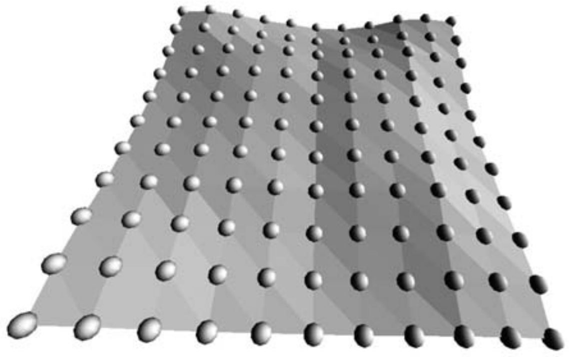


Figure 5. Visualization of a 1D membrane

Figure 5 is a representation of a physical model of an elastic membrane, like in a drum. This is a 1D system: the masses can move in only one dimension, but the representation of the system is in 3D space.
In order to model a massive "elastic" object, you can discretize it into small particles. Each particle can be modeled as an elementary pmpd mass connected to the rest of the objects by visco-elastic links.
pmpd does not allow the simulation of rigid bodies. Anyway, all objects can be seen as elastic bodies, with high rigidity.

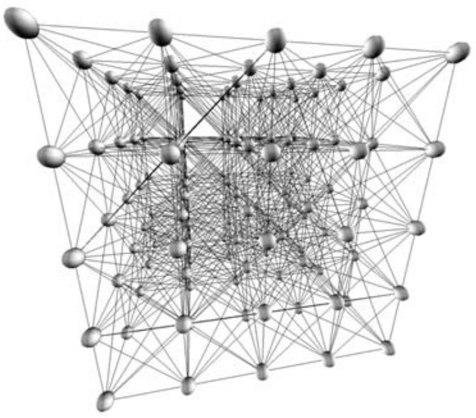


Figure 6. Example of 3D rigid body discretization

Figure 6 is a representation of a cube in 3D space. Many masses are needed in this example to model massive objects, but they also model the deformation of these objects .

Interactor

An interactor object acts as a link object but provides a patching facility. In effect, a single object can create an interaction with an entire class of masses.

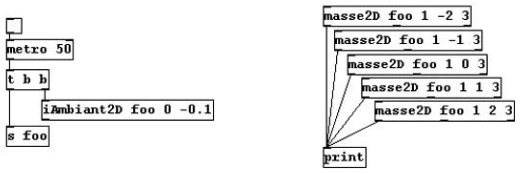


Figure 7. Sending forces to a class of masses

In Figure 7, all masses are subject to a constant ambient force. This force can be viewed as the force of gravity applied to each mass. Different interactor objects provide interactions with a point, a line and other simple primitives.

Test Objects

These objects test the position (as well as the distance, speed from a point, a line, orientation, etc.) of a mass. Thereby, Pd has access to much information regarding the state of the system. This allows interactions with the rest of the patch. Another test object gives information about the link (deformation, speed of deformation, orientation, etc.).

Limitation

The most commonly encountered problem when using this kind of physical modeling is instability. Instability comes from the approximations made with the discretization of the equations. To reduce the risk of instability, the model should be slowed down (increasing the metronome speed can be necessary to keep the desired speed of the simulation). This is usually not a problem due to the relatively low frequency of the simulation. Unlike physical modeling based audio synthesis, the structure usually doesn't need to be computed at audio rate but only at a few hundred hertz. This kind of simulation can thus be calculated for real time applications by a low-cost computer or laptop.

It is possible to choose non-physical values for pmpd parameters. For example, you can set damping to a negative value, which means the creation of energy. This is not physical and can lead to instability or saturation of the model; nevertheless, it can be useful for artistic reasons.

Use of autonomous structures for real time control over audio synthesis

Controllers are physical devices used to send information to a computer. A controller can be a joystick, a keyboard, a midi fader box or any other human interface device which sends information such as the position of a fader, a key press, or any kind of data coming from sensors. Mapping between the control parameters and the audio synthesis algorithm parameters is an important part of the "instrument".

One of the applications of physical modeling simulation is to create a mapping between a musician and the audio synthesis. A dynamic structure modified, moved, and distorted by the user can then be used to control audio synthesis. With the help of sensors, a user can create a virtual structure linked to his or her movements, to the real. The user can then play with a virtual yet "physical" instrument, allowing a natural comportment to digital audio synthesis.

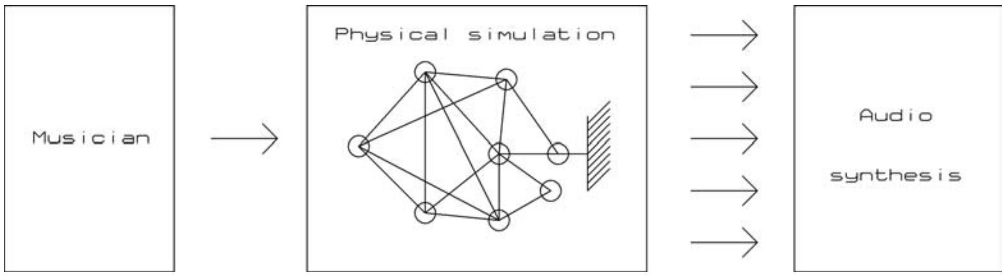


Figure 8. Using physical modeling between the musician and audio synthesis

Figure 8 shows a physical model of a structure used for the mapping of user action and audio synthesis. Such mapping has interesting specifications. For example, a few input parameters can generate many different data flows (a musician can play with only a 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. Certain control parameters can change the way the structure evolves within a time period, allowing the control of the synthesis evolution over time. Another important specification is that all data coming out of the physical model are not independent. The relation between them can be adjusted in regard to the topology of the structure.

Using data generation from a physical model shows the relative importance of audio synthesis.

For example, using simple additive synthesis as in Figure 9 can produce very different sounds depending on the shape and the physical parameters of the structure.

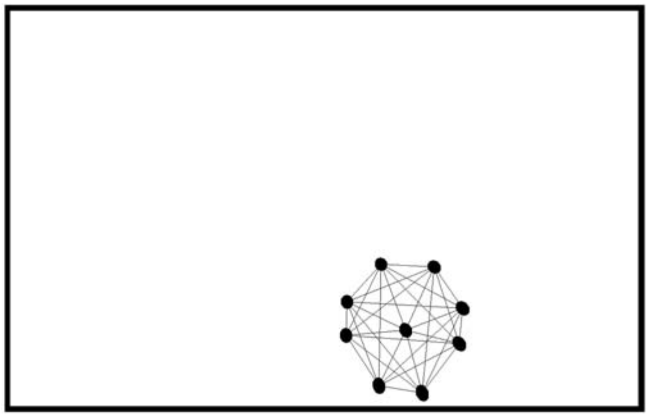


Figure 9. Bouncing ball used for additive synthesis

In Figure 9, forces applied to each mass control the amplitude of the sinusoid which performs additive synthesis. The sound produced by the structure can be controlled while moving the structure, making it bounce, etc. The evolution of the sound over time (the musical structure) can be run with a few parameters that describe the global behavior of the structure, such as rigidity or dampening of link. The physical model offers a user-friendly yet rich and complex interaction with the musical instrument.

A simple physical model can generate a non-linear interaction: the system will react in different ways to the same input solicitation depending on the state (position/velocity) before this solicitation.

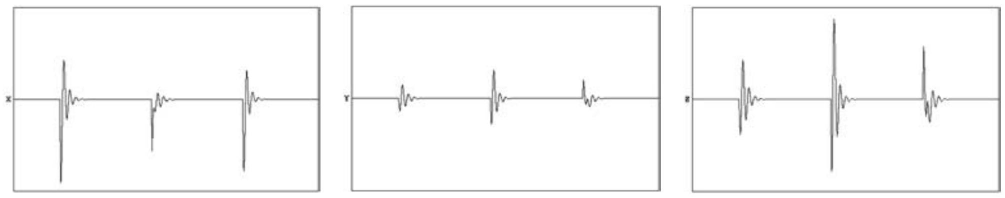


Figure 10. Non-linear system

Figure 10 represents time evolution of X, Y and Z forces of a single mass linked to a fixed point. The same force is sent to the masses in the X direction three times. The displacement depends on the position of the mass when the force is sent. In this example, a sound is produced with an additive synthesis: the amplitude of three sinusoids depends on the force over the link in X, Y, or Z. This instrument offers a rich playing technique due to its behaviors.

Linking this patch to a force feedback joystick permits the user to feel the force it sends to this virtual mass, allowing for more precise control over the synthesis, like with a real (physical) instrument.

Another example of using a physical object to control audio synthesis is scan synthesis, where the shape of a structure is used as an audio waveform.

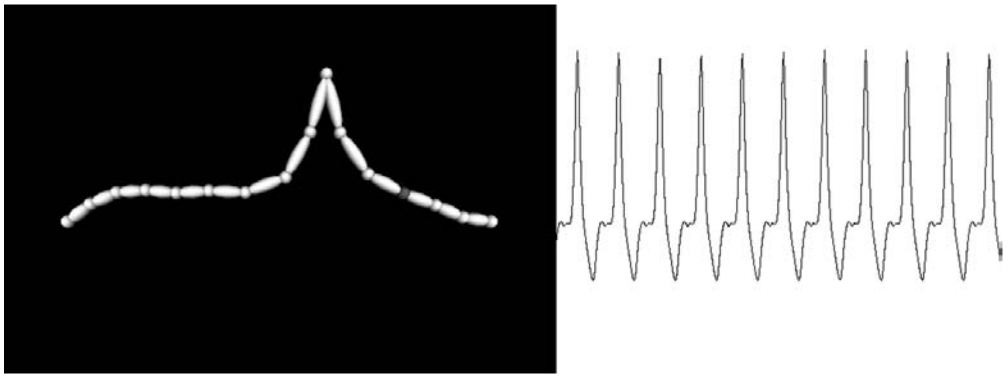


Figure 11. Example of scan synthesis

In the example in Figure 11, the user can interact with an elastic string. This string moves too slowly (only a few hertz) to generate audio sound, yet its shape is recorded in an array that can be looped at audio rate.

EXAMPLES

pmpd has been used to model many different kinds of physical phenomena such as sand, fluid mechanics, and elastic objects. It is also possible to create the behavior of non-physical objects, or objects that cannot be physically built. In fact, structures can be made depending on the kind of behavior you wish. For example, Figure 12 is a snapshot of the "Threads" performance by Ben Bogart where each "word" is a physical modeling structure.

pmpd is not the only set of physical modeling objects for Pure Data. The msd object suite allows the creation of structures inside a single object. msd objects are more optimized than pmpd, but less intuitive.

Figure 12. "Threads" by Ben Bogart

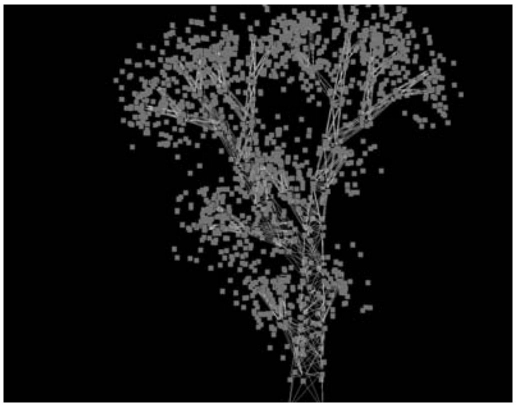


Figure 13. Simulation of the movement of a tree and its leaves

Figure 13 is an example of a simulation of a tree and its leaves. This simulation was made with a Lindenmayer system and the [msd3D] object.

Conclusion

Altogether, only a few simple equations are needed for a complex physical simulation. However, computers deal with digital values at a discrete time, unlike the physical rules that deal with analogue values in continuous time. An approximation must then be found to allow for realistic simulations.

pmpd offers objects for basic physical modeling simulations, and its integration with pure data allows for its use in a wide range of different applications. Many applications still have to be explored, for example haptic devices, which use the sense of touch to better control electronic instruments, and physical models in more than three dimensions. It is hardly possible to represent such a space, but it is possible after all to make "physical" simulations of masses and links.