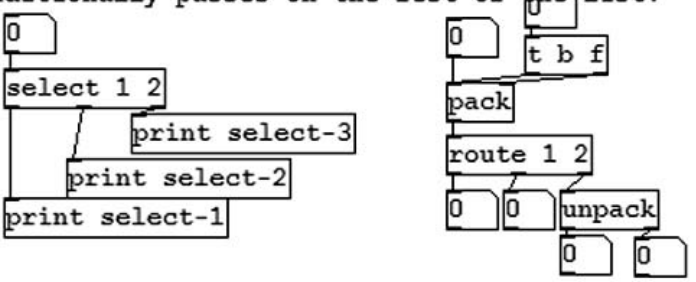
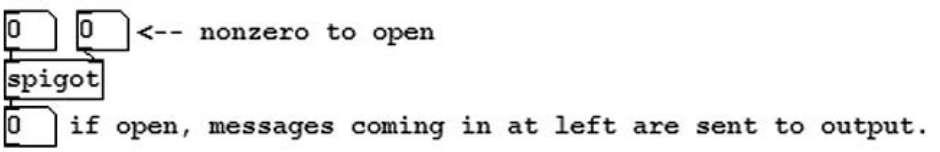
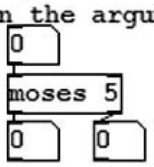
Pd provides at least four objects for doing conditioonal computations. The "select" object tests its input against its argumt(s), and outputs "bang" when they match. The "route" object works similarly but also copies data. In other wors, "route" takes a list, tests its first element, and conditionally passes on the rest of the list.



You also get "spigot" which turns a flow of messages on and off (like the Gate object in Max, but with the inputs reversed):



And finally, "moses" sends numbers to the left if they're less than the argument, right otherwise:



updated for Pd version 0.26

Two Rooms
A Short Conversation with Miller Puckette

Christian Scheib: In a discussion during the first Pd-convention in Graz you described Pd as having a kind of white canvas ideology. What was on the canvas before Pd was invented?

Miller Puckette: Your first question made me think a long while, because the question of what comes before Pd is something like, "everything I knew by the year 1996". And it's hard, even in retrospect, to know what was important and what was not. I think one central thing was a wish that computer music could be made in an even less constraining, and more open-ended, way than was made possible by Max, Pd's predecessor. I avoided re-creating the Max objects at first, and focussed instead on graphing arrays of numbers, which I needed to do anyway to make figures for a paper I was writing at the time. I thought a lot about how to visualize complicated arrangements of data that a composer might use to represent a musical idea. But although I wanted to make it possible to make complicated structures within Pd, I wanted Pd itself to be as simple as possible.

Another aspect of the blank canvas that preceded Pd, was my desire to make a unified way of handling sound recordings ("samples"), images, and control data. I started Max with the IRCAM computer music production scene in mind, but Pd's mental beginnings were more abstract. The unified approach to data storage, not making reference to specific media, was one of my strategies for making Pd as flexible as possible. The one thing I did give Pd ideas about was time and scheduling; and this is perhaps unavoidable, but time passes in the same way (from a physical point of view at least), regardless of what one is doing. But everything else about the structure of a work of art in Pd looks like undifferentiated data.

A third, and more social, aspect was the intellectual property situation. IRCAM made it clear that they didn't want me involved in the further development of Max/FTS (the branch of Max they maintained) – this was one reason I left IRCAM in 1994. There were many changes I wanted to make in Max/FTS, but in the end, I was forced to start from scratch. I made Pd an open source program so that people would want to use it, and sure enough, Pd soon had many more users than Max/FTS. In an odd twist of fate, soon after I started releasing versions of Pd, IRCAM released Max/FTS on the GNU General Public License (much to the happiness of the maintainers). If they had done this before I started Pd, I might never have started it. But by that time (1998, I think) it was too late; Max/FTS was already marginalized and Pd was coming into wide use.

Christian Scheib: If I understand you correctly, something "constraining" must have been on the canvas, before you started whitening it by developing Pd. Constraining in a social way as far as copyright is concerned, constraining in a technical way as far as the difficulty with differ-

ent tools for different media is concerned and constraining in an artistic way as far as pre-produced clichés are concerned. Since there are three reasons, there are three questions.

Manual Digitalisation

Christian Scheib: Observing the outcome of Pd-usage: How has Pd worked artistically so far and is there another development you would still wish for it? Has music become less constraining and more open-ended than before Pd was available?

Miller Puckette: It's hard to describe this, but I keep hoping the computer will be able to function more like a musical instrument (less like a computer) than it has before. The usual mode of doing computer music is still very much like working in a studio (that culture lies at the root of computer music after all). I'd like to see more time-sensitive ways of responding to real-time inputs that would allow human control over the way sounds evolve. But I think this is likely to be a hard research problem.
By and large, Pd is at least as good as any other environment I know of for making computer music, and lots of really excellent work is being done in Pd. I think the limitations that now confront us are more fundamental than just a choice of software (Pd vs. Supercollider, for instance).

Christian Scheib: Hasn't the development of graphic surfaces been the "fall from grace" (in the biblical sense) of computer work anyway? Betraying understanding of what one does by pseudo-analogies that have been invented under the pretext of making everything easier or even more understandable? In other words: hasn't – on a totally different level, but still – a wish for "reacting in a more sensitive way of responding to real-time inputs with more human control" just produced the opposite in the past already? What is the difference in hope and Pd's approach rooted in? (I, too, think it is about more than just a choice of software like Supercollider, yes or no.)

Miller Puckette: I think there are two things going on. The first is that computers are tools for automation. They allow humans to work in patterns, instead of working in details. This makes it very easy to do certain kinds of things (making hundreds of sinusoids, for instance, or drawing fractal trees). But the elementary operation, the putting of a dab of paint on a canvas, for instance, is easier to do by hand than with a computer. So computer art naturally looks different from manual art. Perhaps this is paradoxical, but I hope Pd is less automated, and more "manual", than, say, a purely prescriptive programming language. It's got the right level of automation for making banks of sinusoids without too much trouble, but doesn't encourage making top-down, completely pre-planned compositions, and instead encourages moving forward through experiment.

The second thing is graphical interfaces, and the feeling of intimacy that they give with the computer's workings – a feeling that is, as you suggest, purely illusory. I think the answer is that a graphical interface can be honest or dishonest. An example of a dishonest interface is Microsoft's desktop, on which you don't actually see your files, but only those certain files that happen to be placed so that they're visible there (and aren't hidden). On the other hand, lots of things show up which aren't files at all, and might not even live on your own computer. GUI people call this a "metaphor" and hide behind that word when offering the user things which look alike, but which don't have the same functionality. The result is that you never really know what you are doing.

In contrast, I hope that people who use Pd actually "see" what they are really doing, with nothing hidden and nothing aliased to look like something different. And perhaps that allows a more intimate control of the actual making of computer music or art in Pd than is offered by more metaphor-laden systems.

Transgressing Media

Christian Scheib: To what extent has the unifying approach changed the way people use different media? In which directions are the needs, wishes, developments of the community going in this respect? Does this have aesthetic consequences?

Miller Puckette: I think that in the last couple of decades many artists and composers have become quite fluent at mixing audio and image production and passing controls and information between media. There are some pitfalls (for instance, it's easy to make things that are too predictable or pedantic), but good artists can see and avoid them with experience. I've seen some wonderful work recently, particularly at the Pd convention itself. I think it's really the artistic community which is driving this more than Pd itself, which just happens to be a good way to realize these sorts of things.

Christian Scheib: So the old antagonism between artist and tool/media/instrument is reappearing in some new form? In other words, aesthetically it has been one of the core functions of the range of instruments and/or material to define and provoke what the artist is doing or can be doing or might be trying in order to transgress. What if you were successful in providing the idea of the technologically imagined metaphor of the white canvas? What is left to transgress? Should we just abandon the concept of transgressing? But then look at or listen to the Pd-community's work in Graz: Some kind of transgression has been involved in all the convincing examples. So if the medium inevitably plays such a central role in the production of art, what would you – as an artist and as a programmer – think Pd's role is in this respect?

Miller Puckette: In any form of art-making there has to be tension. However, I would rather see the tension lie in the artistic imperatives than in a contest between the artist and the limita-

tions of his or her tools. This is an aesthetic stance, by the way: plenty of artists actually seek tension in the difficulty of the realization (Iannis Xenakis often did this). But somehow, I think that an artist who really needs difficulty of realization can always find it, and it's not my role to supply difficulties.

A favourite metaphor of mine is to compare two rooms, one tiny one and one large one. If you live in the tiny one, you might want to move to the larger one in order to have fewer boundaries. But of course the larger room simply has a larger surface, and hence more boundaries than the small one had. In the same way, the more transparent, malleable, and powerful the tool is that you use, the more ways you can hit the wall with it. So perhaps, even though Pd tries to give you the most freedom, it ends up giving you the richest possibilities for frustration.

Open Source

Christian Scheib: You mentioned that the reason for the decision for open source was "so that people would want to use it". This is a very practical way of putting something that is also heavily loaded with lots of ideology. Does a more theoretical or ideological thinking have some (hidden or intentional) influence on your work?

Miller Puckette: Well, I hate seeing big corporations rob people, so yes, to that limited extent. I see in particular the increasing use of patents as tools for keeping small players out of the game, and I think we should all resist that. But as for Pd itself, I don't see it as a political act, just as the best way I can see to navigate the situation and get tools in the hands of people who need them.

Christian Scheib: Okay, so let's stay practical instead of ideological here. Getting tools into the hands of as many people as possible may not be, but may well end up soon becoming a marketing concept. Linux for Munich or so. Are you concerned with that? (Sorry, I know this is not practical, but pseudo-ideological.)

Miller Puckette: I'm very excited by the movement toward open source in general, because I see it as breaking the strangle-hold that software corporations have over their users. It's simply unconscionable to prohibit a person from knowing how something he or she owns (a computer) actually works inside. It's also unconscionable for any democratic government to allow itself to be locked into a private vendor in order to carry out essential functions; this endangers the populace needlessly. The situation is different for artists; they aren't in charge of keeping the trains from crashing into each other. So there's nothing immoral about an artist using a proprietary piece of software. However, an artist who thinks carefully about preserving his or her work will naturally prefer the open source solution, because it's much easier to keep running than a proprietary one can ever be.

Automated Metaphors (Conclusion)

Christian Scheib: I just love your metaphor about the tiny and the large room with its widening and enlarging of boundaries at the same time. This seems to sum up pretty exactly what the potential of Pd is in the sense of possibilities as well as frustration. This leads to two closely related questions: from your observation and judging from questions and feedback you get, has the community pushed the development of Pd unambiguously into this direction of more and more "manual" openness, or are there also some tendencies to "serialize" or automate? And again from your personal observation, has art/music been developed in recent years that owes its essential quality to these characteristics of the metaphorical large room?

Miller Puckette: Hmmm....
1. I think most people attracted to computers in the first place (including most Pd users) have a tendency to like to automate things. The fact that Pd users are computer artists or musicians in the first place means the population has already been selected for that trait. So I think it natural that one sees more of a tendency to automate Pd usage than to de-automate it. But there are a couple of interesting counter-examples. In particular, the small fringe of people who are actually using the experimental graphical data editing functions of Pd seem to like it for its very explicit, even tedious, detail. So my best answer is, "both".

2. I don't know any examples myself... since it's really the artist, not the viewer, who experiences the possibility/limitation of the software, if he or she wanted to incorporate that, itself, into an artwork, he or she would have to somehow portray or offer an experience of using Pd. Running a camera while the artist works on Pd and showing the video as the artwork, for example. It might be hard to figure out how to make a convincing artwork out of that idea.