A tutorial thingy about polyphony (events)

Thread Status: **Not open for further replies.**

carloskleiber
Forum Member

Messages: 537

Hello,

I promised some time ago in mmower's "New Reaktor Book" thread that I would write something about the use of polyphonic events. I was a bit reticent to start it off since and I realize I was being nervous about the need to carve it in stone in form of a .pdf file. My english is not good enough and my knowledge about the topic is limited. Also recent reactions of esteemed forum members who have largely contributed for the community saddened me, but also warned me from putting too much time and effort into such a task. This is why I'll post here, and in a relaxed fashion. Besides, writing a thread allows for questions, precisions, corrections... Sorry if you might find the title pretentious, it is just in order to help searches in the future.

I have chosen this topic because I think it is great fun and very useful knowledge, and I beleive I can write something more or less coherent about it. Along with primary and core, I would almost say that polyphony offers a third "style" of building with Reaktor. Lazyfish never used core, but made extensive use of polyphonic events in his structures. For this reason alone it is worth to have some basic understanding about it.

What can be found in the books?
...not so much, actually. The operation manual advises more than once to turn macros to mono in order to save CPU, unless it is "absolutely necessary" to keep polyphony. This seems to be common sense as we learn Reaktor aswell: since numeric readout (thus: the Event Watcher) and meter modules take monophonic messages only, and buttons, knobs, etc... send out mono anyway, we easily get the reflex to set modules and macros to mono. Understanding structures can be complicated enough like this.

Len Lasso's book gives some interesting thoughts on page 21. Indeed as we refer to polyphony it is not entirely clear what we talk about. On the one hand the number of voices of an instrument indicate how many (MIDI) notes can be hit down simultaneously. But this will be true for the most basic synth, with one oscillator and one envelope only, where clearly no exciting polyphonic activity is going on.
On the other hand, as soon as there is more than one sound source in a structure, we are in presence of polyphony in some way. It is our choice then to route them in different wires, in the same wire but different voices, or to merge them into the same voice. The choices we make will greatly influence CPU usage and the instrument's sound expression and complexity, and there are compromises to be made.

Regarding events, polyphony can be even more loosely related to the final result. Look at Newscool, with its 1024 voices, or the sqp sequencer, with 300 voices. These are cases where the polyphony is used for complex event management.

carloskleiber, Feb 7, 2010

#1

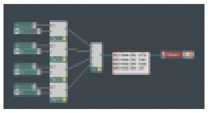
carloskleiber

Forum Member

Messages:	537

Polyphonic cuisine: the ingredients

Let us recapitulate the basics: to compile a polyphonic signal we need To Voice modules, with the relative voices indexed starting by 1, and an adder. In case of an event signal a merge will do aswell. Good to know: although a numeric readout module won't accept this signal, the wire connected to it will still display the info needed as we mouse over it.



To isolate one voice from a polyphonic signal we need a From Voice module, with the voice's number attached to the V input.

We all know the audio voice combiner module that can be found by factory setting at the output of instruments, because signals leaving an instrument must be monophonic. Its event correspondent comes in 3 flavours: event voice combiner All, Max, and Min. The Voice Info module's upper output gives the index of each voice, starting by 1. The polydisplay is much easier to program than the multidisplay, together with voice info and polyphonic signals.

carloskleiber, Feb 7, 2010

#2

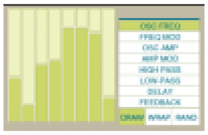
Let us start cooking!

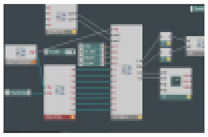
carloskleiber

Forum Member

Messages: 537

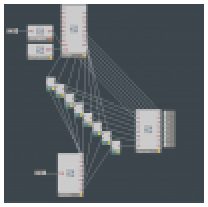
Rather that the life sequencer with its 1024 voices let us examine something much more simple and useful: the polycontrol from Skrewell. (I let you dig it out of its structure)





You may not think of Skrewell as of an example for controllability, but I will try to convince you that its polycontrol is not only a beautiful, but also very efficient piece of GUI. (It works in its full splendour with 8 voices. If you copypaste it into a new ens, there might be gaps between the columns, so raise the number of voices)
If you look into the structure you'll see that it outputs 8 polyphonic signals with values between 0 and 1. In Skrewell it controls 8 (or 6) parameters of 8 oscillators, but it can control pretty much anything you like. Besides it has a randomize function and 3 different drawing modes that might be useful or not for what you need, but definitely stuff that would be harder to implement with traditional controllers.

Allright, let us start modding the polycontrol! As you dive into the structure you find a rather intimidating spiderweb. After removing unused input ports it looks something like this:



There are five macros inside. The ChannelSelect macro contains the multipicture to the right from the polycontrol itself. If you check with the Eventwatcher you'll see that it outputs integers from 0 to 7. Since in your new setup the parameters will probably not be called "osc freq", "freq mod", etc... anymore, you can just as well delete it and replace with a list where values go from 0 to 7 just the same and you are free to give names to them.
The EditModeSelect macro contains another multipicture. This one lets you decide how to draw into the polycontrol. You might need it or not, if you think of using traditional drawing only, you can delete and connect a 0 constant instead.
The screen macro is the most complex of the five. It contains the panel elements that the polycontrol is made of: a mouse area and a polydisplay, along with quite some logic. Let us leave it for now.
The functions macro contains the randomizer. We have lost its button, so just create one and hook it up instead of the rnd input port. You can check that it works and sends random values to all 64 columns.
Finally, the snapvalue x8 macro contains... 8 snap value modules, which connect the big POLYCTRL macro to the external world. The snap values' output is also routed back to screen, to make sure the screen displays the right values as you recall a snapshot.
Between the macros there are 8 merge modules. They allow to use both methods - randomize or draw into the screen - to write data into the snap value modules AND into

the screen. Let us add a third method to enter data. Take the first merge and add a 3rd port to it. Create a knob and hook it up to the new port. Voilà! You can now control the polyctrl with a knob, which means, with an external controller if you like!
One small problem though: we now control all 8 parameters of one position in the list. To fix this, we'll add a To Voice module between our knob and the merge. To the V input we'll connect another knob, with min=1, max=8, stepsize=1. Much better

carloskleiber, Feb 8, 2010

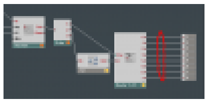
#3

carloskleiber

Forum Member

Messages: 537

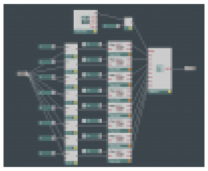
We could now play around with the randomize function. If we turn back to the original Skrewell ensemble, one useful feature could be to isolate certain parameters from being affected by randomization. Skrewell can be too harsh sometimes and this would let us smooth off the edges. Let us open the functions macro. This is how its output looks like:



All we need to do is to insert some kind of gate between the outputs of the router 1->M and the output ports of the macro. The easiest way I can think of is with separators. Connect a button to the threshold input of the separator, and one of the separator's outputs to the output port of the macro. If for example your button toggles between 0 and 1, the upper port of the separator won't let through anything on button position 1, and will let through everything on button position 0. Everything? Not exactly. It won't let through the zeros generated by the randomizer. So let us set the button to toggle between -.1 and 1 instead.

Allright, we now have 8 buttons which allow to isolate the 8 parameters (or 8 horizontal sets of values) from the global randomization. But one parameter that can cause particularly loud changes in Skrewell is the delay parameter. So I suggest we build additionnal random isolate controls for each individual voice's delay settings. Of course these extra controls only make sense if the delay randomization is enabled. So we'll insert them between our newly created gate and the output port that carries the polyphonic signal with the delay parameters.

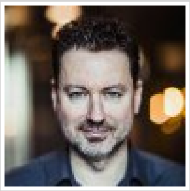
The only difference from the previous situation is that the parameters we now want to individually control travel in the same wire. First we separate them into 8 mono signal with From Voice modules. Then we do the separator trick on them with 8 brand new buttons (I know we start having a bit too many buttons). Finally we need to compile the signals back into an 8-voice signal. If we feel sick of too many modules we can use a different trick this time: we'll merge the 8 wires in 8 inputs of a selector. (It probably works with a Router M->1 too, I haven't checked). To the Pos input of the selector we'll hook up a voice info module. One problem left though: Voice info indexes starting by 1, and the selector's inputs start by 0. No big deal: we just substract 1. The voice info now outputs 0,1,2,... instead of 1,2,3,...



One important thing: we can apply algebraic operations to polyphonic signals just the same way as to monophonic signals.

carloskleiber, Feb 8, 2010

#4



mmower

NI Product Owner

Messages: 726

Carlos it's great that you've started doing this. I can't seem to PM you, can you PM me? Or email me? I'm self@ my domain (see URL in profile).

Matt

mmower, Feb 8, 2010

#5

carloskleiber

Forum Member

Messages: 537

...I think I can hear your next question: allright for algebraic operations, but what happens with a polyphonic signal going through signal path and event processing modules?
Well, I didn't know either, so I tried to check out. If you like, look at the silly little diagnostic tool attached.
...it is not such a big deal, I can just tell you in words. Basically, each value will travel through the structure as if it was a mono structure, independently from the other voices. One value will take a path, the second value another path, the third might be discarded. So the very interesting (and not straightforward) conclusion is that in many wires of a polyphonic structure there can be uncomplete signals, i.e. with missing voices.

Attached Files:

polyevent routing.ens.zip
File size: 4.3 KB
Views: 158

carloskleiber, Feb 8, 2010

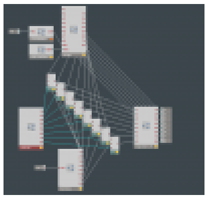
#6

carloskleiber
Forum Member

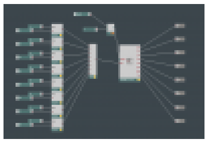
Messages: 537

...sorry for this theoretical paranthesis, I just think it was necessary to understand (for me at least) that a signal coming from a To Voice module is not a mono signal with some index on it, but a polyphonic signal with empty voices, except for the voice selected by the V input. This also explains better the second method I used to compile a polyphonic signal, the one with the Voice Info module and the selector. The pos input of the selector receives a polyphonic signal, with different values in each voice. These different values select different input ports of the selector, but the voice allocations are respected. And since this happens simultaneously, the result is ONE full polyphonic signal at the output.
...therefore one more very important conclusion: with the use of polyphony, events can happen simultaneously on primary level.

Let us go back down to earth and start building something useful! Sticking with the POLYCTRL and the idea of expanding its controllability we could build an interface for send effects. I let you choose your 8 favourite sources (like an 8-track mixer or drum machine) and 8 favourite effects (if you have less it's okay). We'll build what is in the middle.
Remember how the inside of the POLYCTRL macro looks like. In the end it should look like this:



Let us build this new macro now. We take 8 faders and use the known method to merge their outputs into a polyphonic signal. Then we place a router with a list to decide where this signal should be connected to the POLYCTRL macro. Almost too easy:



Assign MIDI controls to the faders, and check if it works. Normally you should now be able to control simultaneously all 8 columns of one view of the POLYCTRL. (Note that I created a new list for the controls, instead of using the other one: the one that is already inside the POLYCTRL, and decides about the view.)

So basically we now can remote control the send amount into one effect for 8 sources simultaneously. Additionally we want to be able to control ALL effect sends for one source simultaneously. Back to work!

carloskleiber, Feb 11, 2010

#7

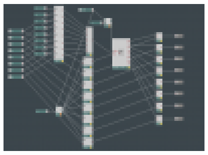
carloskleiber

Forum Member

Messages: 537

Controlling various parameters of one source in our case litterarly means using the same voice for our 8 signals. How can that be done? Using all 8 outputs of our "controls" macro! Instead of one signal with 8 voices through one output, we'll now have signals through all 8 outputs, but only one active voice in each (the same).

We connect our faders to eight new To Voice modules, but this time we won't merge their outputs, instead we'll connect them to the 8 outputs of the macro. What we'll want to vary this time is the V input of the To Voice modules. We could create a new list with values 1 to 8 and hook it up to our new To Voice modules' V inputs, but let us not forget that we want to keep the other functionality aswell, and we cannot control both "horizontally" and "vertically" in the same time, since we only have 8 faders. It is either or. So instead of creating a new list module, let us add 8 values (9 through 16) to our existing list. To get our values 1 through 8, corresponding to the voice numbers... we just substract 8.



carloskleiber, Feb 11, 2010

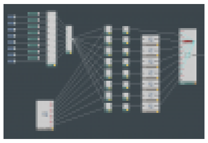
#8

carloskleiber

Forum Member

Messages: 537

The rest is simple but I post it here the same:



the nice thing is that you can take full advantage of the tidy control signals that come from the POLYCTRL by combining your 8 audio sources into one polyphonic audio signal. After that, it is just multiplying the result with the event signals corresponding to each effect (and to the dry sound). I have put audio voice combiners after the results because I assume that the effects might be too heavy on CPU if used with 8 voices. But of coure it depends on the effect. Leaving some effects polyphonic can have nice results if you want more complex routing between the effects, serial instead of parallel and so on. If you have enough CPU, go for it!

carloskleiber, Feb 11, 2010

#9

> carloskleiber said: ↑
>
> To isolate one voice from a polyphonic signal we need a From Voice module, with the voice's number attached to the V input.

> We all know the audio voice combiner module that can be found by factory setting at the output of instruments, because signals leaving an instrument must be monophonic. Its event correspondent comes in 3 flavours: event voice combiner All, Max, and Min. The Voice Info module's upper output gives the index of each voice, starting by 1. The polydisplay is much easier to program than the multidisplay, together with voice info and polyphonic signals.

CList

Moderator

Messages: 3,299

As I mentioned in another thread recently, you need to be careful with instrument start-up initialization with this sort of thing. There's no guarantee that the constant will fire it's event before the knob here (you can view the initiatialization order with the debug menu option, and delete / add modules until you get it right, but that doesn't ensure it will be right when you copy and paste the structure).

You should put an event-order after the knob, and connect output 2 of the event-order to the input of the to-voice. This will gaurantee that the constant's value for the voice number always hits the ToVoice before the knob value when the ensemble starts up!

Cheers,
Chris

CList, Feb 11, 2010

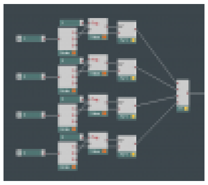
#10

carloskleiber

Forum Member

Messages: 537

Hi Chris,
You mean like this?



I guess it is the same then with the voice info, as it seems to have the same "priority level" as a knob, from what I understand from the Initialization Algorythm...

carloskleiber, Feb 11, 2010

#11

CList

Heh, no that's backwards. Sorry I can;t post a picture from this PC...
Constant -> ToVoice.V
Knob -> EventOrder.In
EventOrder.Out2 -> ToVoice.In

EventOrder.Out1 is left unconnected and there's no need for an EventValue

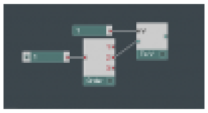
Moderator

Messages: 3,299

Cheers,
Chris

CList, Feb 11, 2010

#12



This is cool really. I didn't realize you could use order modules like this.

> Fourth Step - Process all Order-Modules
> For all active Order Modules, set the 2nd and 3rd OutPort and send Events as normal Reaktor Events.

(quote from Initialization Algorythm)

It rings a bell now

carloskleiber, Feb 11, 2010

#13

carloskleiber

Forum Member

Messages: 537

herw

NI Product Owner

Messages: 5,868

I would like to discuss here but i am not able to follow to your attachment; is it possible to upload here the whole test ensemble or is it part of a big ensemble?

ciao herw

herw, Feb 16, 2010

#14

CList

Moderator

The issue I'm talking about can be found in the Vierring factory ensemble. It does not initialize it's values for the filter properly when it starts up, and requires a change in snapshot or a press of panel switch before it will start working correctly.

This is because the 4 filters are each run on different polyphonic voices, and the constants attached to the ToVoice modules fire their events before the knobs attached to them, so the knobs values don't get properly routed to the correct voice when the ensemble first

Messages: 3,299

start up.

Once the ensemble is running it works fine because the constants have been initialized.

Cheers,
Chris

CList, Feb 16, 2010

#15

carloskleiber
Forum Member

Messages: 537

@herw, do you mean the ensemble used for this thread? I haven't uploaded anything, I thought in the tutorial perspective it is best to do it with illustrations only. The central piece, the polycontrol macro, is to be found in the Skrewell factory ensemble. I was going to upload the final result as an .ens after a last part I wanted to write in the next couple of days, but if you want I can upload something sooner..

@CList actually I haven't experienced this issue in stuff I have been working with (including the derivates of what is being built here) but I totally trust you on this one. I have always thought that the outputs of the order module had a relative order only, and now after rereading the initialization algorythm I am a bit confused about how to use correctly this module, and when can I just hook up port 2 instead of port 1 to delay an event and when not.
...but not so sure if this is directly connected to this thread. I have a similar problem in another thread in the building section. If you could have a look it would be really great

I'll finish this tutorial (or whatever I meant by doing) as soon as possible...

carloskleiber, Feb 16, 2010

#16

CList
Moderator

Messages: 3,299

The only reason you'd use an EventOrder with output 1 being disconnected and only using output 2 (or output 2 and 3) is when you want to make sure that the event gets processed AFTER all of the constants get processed.

This is only an issue when the instrument first starts up

From that point onwards, the thing the constants are attached have all received the constant values and we don't have to worry about the ordering. If at start up a constant has not fired it's value then the port the constant's connected to has a value of "0".

In the case I've talking about here, you MIGHT run into the situation...but you might not (it depends on the order the modules were added to the structure and is somewhat arbitrary!

Cheers,
Chris

CList, Feb 16, 2010

#17

herw

NI Product Owner

Messages: 5,868

> carloskleiber said: ↑
>
> @herw, do you mean the ensemble used for this thread? I haven't uploaded anything, I thought in the tutorial perspective it is best to do it with illustrations only. The central piece, the polycontrol macro, is to be found in the Skrewell factory ensemble. I was going to upload the final result as an .ens after a last part I wanted to write in the next couple of days, but if you want I can upload something sooner..
>
> [...]

yes i would like to see it because your algorithm is a from voice - to voice combination. If it is without changing the voice maybe there would be another solution.

ciao herw

herw, Feb 16, 2010

#18

carloskleiber

Forum Member

Messages: 537

Here is the structure as we have come to so far. There is one change only. I have added one extra port with a dummy output to the router 1->M that you see on the picture in post #8. Indeed routers seem to act buggy sometimes: if their pos input receives a value higher than their number of outputs, they will route the incoming signal through the bottom output (the one with the highest number), even if not in wrap mode.

There are no input sources and the effects are empty macros for now. You have to complete the structure to make it functional.

Attached Files:

8-voice control.ens.zip	
File size:	17.7 KB
Views:	139

carloskleiber, Feb 17, 2010

#19

herw

NI Product Owner

Messages: 5,868

thanks - i will have a look at it.

ciao herw

Last edited: Feb 17, 2010

herw, Feb 17, 2010

#20

1 2 Next >

(You must log in or sign up to reply here.)

Thread Status: **Not open for further replies.**

Forums ... **REAKTOR**

Legal Information | Privacy Policy | Contact

English (US)