Catork

CataRT skin for *LOrks

June 22, 2010

Catork Documentation

Catork is based on CataRT, the MaxMSP implementation of concatenative synthesis by Diemo Schwarz. CataRT's documentation is very helpful to understand what is going on under the hood here: <http://imtr.ircam.fr/imtr/CataRT>. For more info on concatenative synthesis, see also Schwarz, Diemo (2006) 'Concatenative sound synthesis: The early years', Journal of New Music Research, 35:1, 3-22.

Catork is a CataRT-based system for composition and performance of electronic music. It was particularly designed to be used with laptop orchestras (which apparently tend to have names ending with “Ork”, such as SLOrk, PLOrk, L2Ork, etc.)

I have been calling Catork a “CataRT skin for *LORks.” In fact, all the sound generation is done by CataRT. Catork is simply an interface that tries to make things look “nicer” for the end user; that's why I call it a skin. It is, however, more powerful than just a decorative skin: in fact, Catork provides a framework for composition and performance in which certain ways of doing things are facilitated. This includes a system of shortcuts to play the patch in real-time, as well an instant messaging system useful for coordination of laptop ensembles (one laptop being the “conductor” sending messages to the rest of the “orchestra”).

This documentation contains installation instructions, a quickstart guide, an in-depth view of Catork's compositional features, and a FAQ section.

Installation

1) You need a computer with Max Runtime installed. Max Runtime is the free version of MaxMSP that allows you to run (but not modify) patches developed by others. You can download Max Runtime from this page:

http://cycling74.com/downloads/

Advanced users can, of course, use the full version of MaxMSP in order to change, adapt or extend Catork to fit their needs. For simplicity, I will just assume refer to Max Runtime for the rest of this tutorial (best for beginners).

2) After installing Max Runtime you also need to install the FTM library:

http://ftm.ircam.fr/index.php/Download

This is a standard installer: just double click and follow the instructions. Make sure you point to the Max Runtime folder as the install location.

3) Finally, you need to copy the entire Catork folder into your Max or Max Runtime

directory (Applications > Max Runtime). Attention: you REALLY have to copy Catork inside the Max Runtime folder. If you just leave the Catork folder on your desktop, for example, the patch will not load correctly.

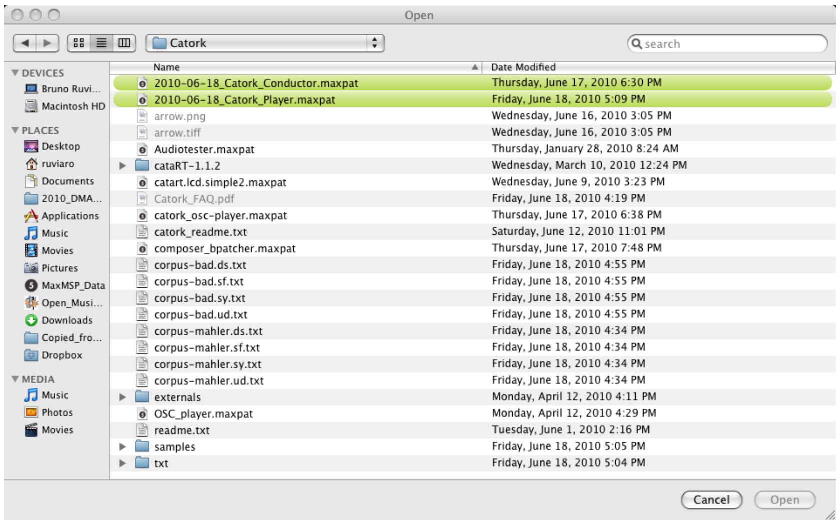
QuickStart

The goal of this section is to help you get your first sounds out of Catork, without explaining every single step in too much detail. Here's how to quickly get Catork to play something so you can try it out:

Open Max Runtime.

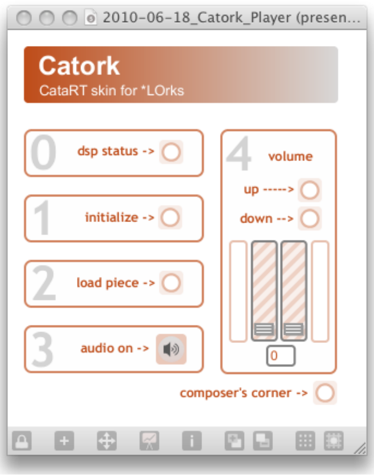
Go the menu File > Open (or shortcut command + O).

Find your way to the Catork folder. You will see two files highlighted in green, with names starting with a date in the format YYYY-MM-DD, like in the screenshot below:



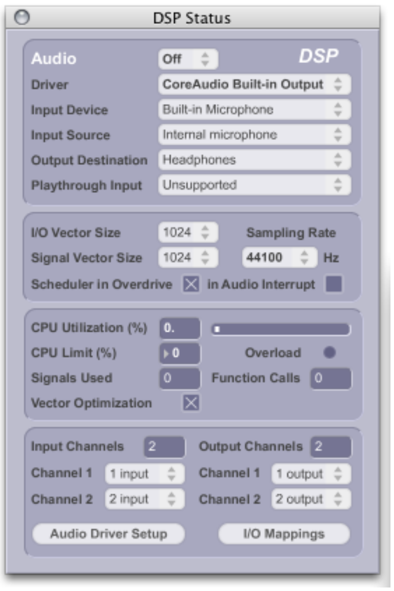
Name	Date Modified
2010-06-18_Catork_Conductor.maxpat	Thursday, June 17, 2010 6:30 PM
2010-06-18_Catork_Player.maxpat	Friday, June 18, 2010 5:09 PM
arrow.png	Wednesday, June 16, 2010 3:05 PM
arrow.tiff	Wednesday, June 16, 2010 3:05 PM
Audiotester.maxpat	Thursday, January 28, 2010 8:24 AM
cataRT-1.1.2	Wednesday, March 10, 2010 12:24 PM
catart.lcd.simple2.maxpat	Wednesday, June 9, 2010 3:23 PM
Catork_FAQ.pdf	Friday, June 18, 2010 4:19 PM
catork_osc-player.maxpat	Thursday, June 17, 2010 6:38 PM
catork_readme.txt	Saturday, June 12, 2010 11:01 PM
composer_bpatcher.maxpat	Thursday, June 17, 2010 7:48 PM
corpus-bad.ds.txt	Friday, June 18, 2010 4:55 PM
corpus-bad.sf.txt	Friday, June 18, 2010 4:55 PM
corpus-bad.sy.txt	Friday, June 18, 2010 4:55 PM
corpus-bad.ud.txt	Friday, June 18, 2010 4:55 PM
corpus-mahler.ds.txt	Friday, June 18, 2010 4:34 PM
corpus-mahler.sf.txt	Friday, June 18, 2010 4:34 PM
corpus-mahler.sy.txt	Friday, June 18, 2010 4:34 PM
corpus-mahler.ud.txt	Friday, June 18, 2010 4:34 PM
externals	Monday, April 12, 2010 4:11 PM
OSC_player.maxpat	Monday, April 12, 2010 4:29 PM
readme.txt	Tuesday, June 1, 2010 2:16 PM
samples	Friday, June 18, 2010 5:05 PM
txt	Friday, June 18, 2010 5:04 PM

Choose 2010-06-18_Catork_Player.maxpat and click “Open” (note that the date at the beginning of the file name may change, and that's fine. Just make sure you are opening Catork_Player, not Catork_Conductor). The patch may take a few seconds to load. Once the patch is open, this is the first thing you will see:



The Catork_Player window is organized in a “checklist” style. The checklist shows the steps you need to complete to make sure the patch will work properly. We will now go through the checklist.

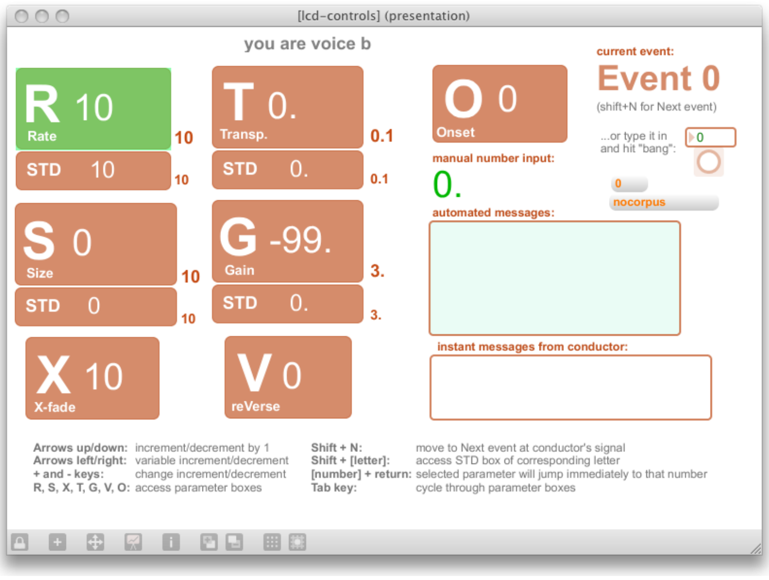
Click on checklist item #0, “**dsp status**”. The standard Max DSP Status window will appear:


DSP Status

Audio	Off
Driver	CoreAudio Built-in Output
Input Device	Built-in Microphone
Input Source	Internal microphone
Output Destination	Headphones
Playthrough Input	Unsupported
I/O Vector Size	1024
Signal Vector Size	1024
Sampling Rate	44100 Hz
Scheduler in Overdrive	☒
in Audio Interrupt	
CPU Utilization (%)	0.
CPU Limit (%)	0
Overload	●
Signals Used	0
Function Calls	0
Vector Optimization	☒
Input Channels	2
Output Channels	2
Channel 1	1 input
Channel 2	2 input
Channel 1	1 output
Channel 2	2 output

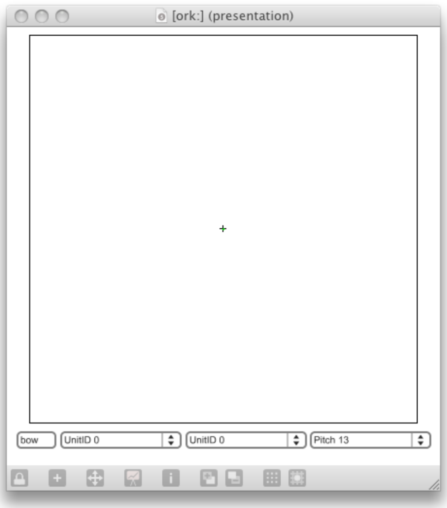
In the “Driver” drop-down menu, select either your built-in audio driver or your external audio interface, if any. Click on the small “x” (top-left corner) to close this window.

Now click on checklist item #1, “**initialize**”. Two new windows appear, one called [lcd-controls] and another called [ork:]. Spread them nicely on your screen.

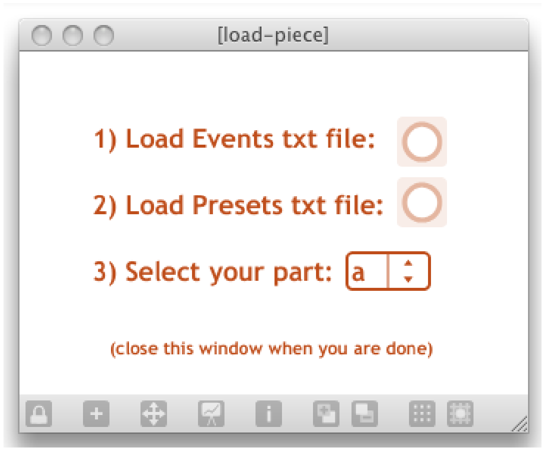

[lcd-controls] (presentation)

you are voice b

Arrows up/down:	increment/decrement by 1
Arrows left/right:	variable increment/decrement
+ and - keys:	change increment/decrement
R, S, X, T, G, V, O:	access parameter boxes
Shift + N:	move to Next event at conductor's signal
Shift + [letter]:	access STD box of corresponding letter
[number] + return:	selected parameter will jump immediately to that number
Tab key:	cycle through parameter boxes


[ork:] (presentation)

Click on checklist item #2, “**load piece**”. The following small window appears:



1) Load Events txt file:

2) Load Presets txt file:

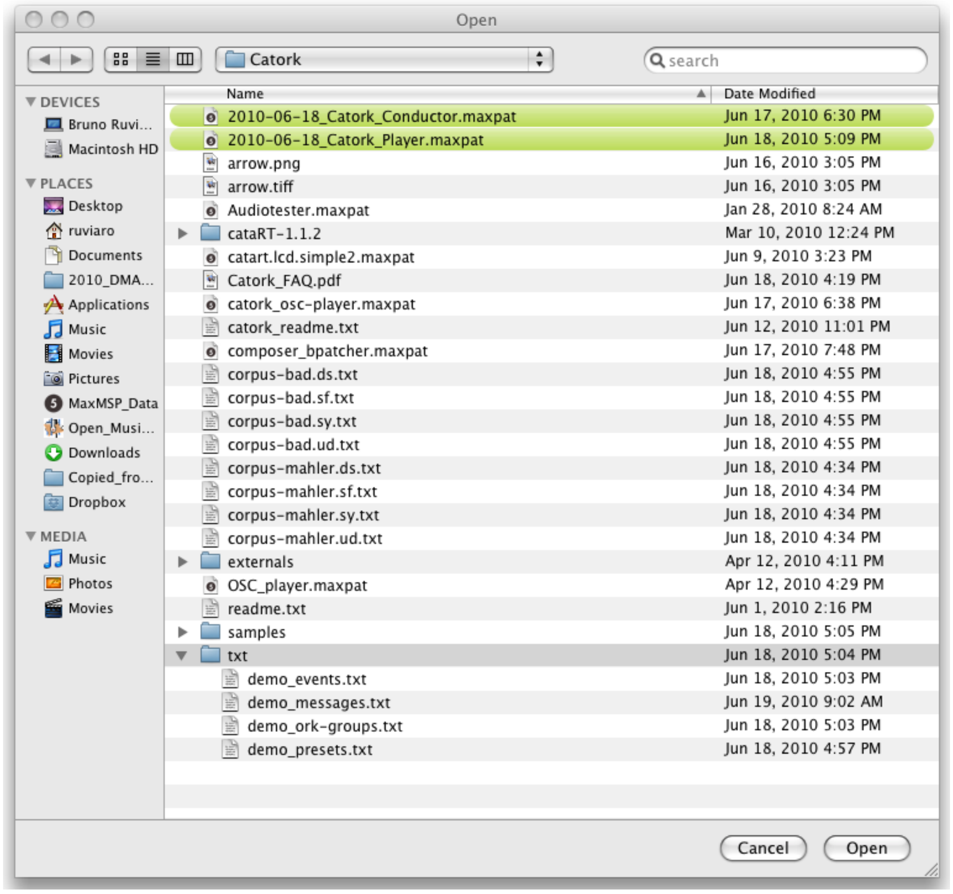
3) Select your part: a

(close this window when you are done)

Follow these three simple steps to load the demo included in the Catork folder.

Click on “1) Load Events txt file” and choose “demo_events.txt”
Click on “2) Load Presets txt file” and choose “demo_presets.txt”
Click on “3) Select your part” and choose 'a' or 'b' from the drop-down menu.

Note: you will find the Events and Presets text files in the folder 'txt', inside Catork's main folder, as you can see at the bottom of the screenshot below:



Name	Date Modified
2010-06-18_Catork_Conductor.maxpat	Jun 17, 2010 6:30 PM
2010-06-18_Catork_Player.maxpat	Jun 18, 2010 5:09 PM
arrow.png	Jun 16, 2010 3:05 PM
arrow.tiff	Jun 16, 2010 3:05 PM
Audiotester.maxpat	Jan 28, 2010 8:24 AM
cataRT-1.1.2	Mar 10, 2010 12:24 PM
catart.lcd.simple2.maxpat	Jun 9, 2010 3:23 PM
Catork_FAQ.pdf	Jun 18, 2010 4:19 PM
catork_osc-player.maxpat	Jun 17, 2010 6:38 PM
catork_readme.txt	Jun 12, 2010 11:01 PM
composer_bpatcher.maxpat	Jun 17, 2010 7:48 PM
corpus-bad.ds.txt	Jun 18, 2010 4:55 PM
corpus-bad.sf.txt	Jun 18, 2010 4:55 PM
corpus-bad.sy.txt	Jun 18, 2010 4:55 PM
corpus-bad.ud.txt	Jun 18, 2010 4:55 PM
corpus-mahler.ds.txt	Jun 18, 2010 4:34 PM
corpus-mahler.sf.txt	Jun 18, 2010 4:34 PM
corpus-mahler.sy.txt	Jun 18, 2010 4:34 PM
corpus-mahler.ud.txt	Jun 18, 2010 4:34 PM
externals	Apr 12, 2010 4:11 PM
OSC_player.maxpat	Apr 12, 2010 4:29 PM
readme.txt	Jun 1, 2010 2:16 PM
samples	Jun 18, 2010 5:05 PM
txt	Jun 18, 2010 5:04 PM
demo_events.txt	Jun 18, 2010 5:03 PM
demo_messages.txt	Jun 19, 2010 9:02 AM
demo_ork-groups.txt	Jun 18, 2010 5:03 PM
demo_presets.txt	Jun 18, 2010 4:57 PM

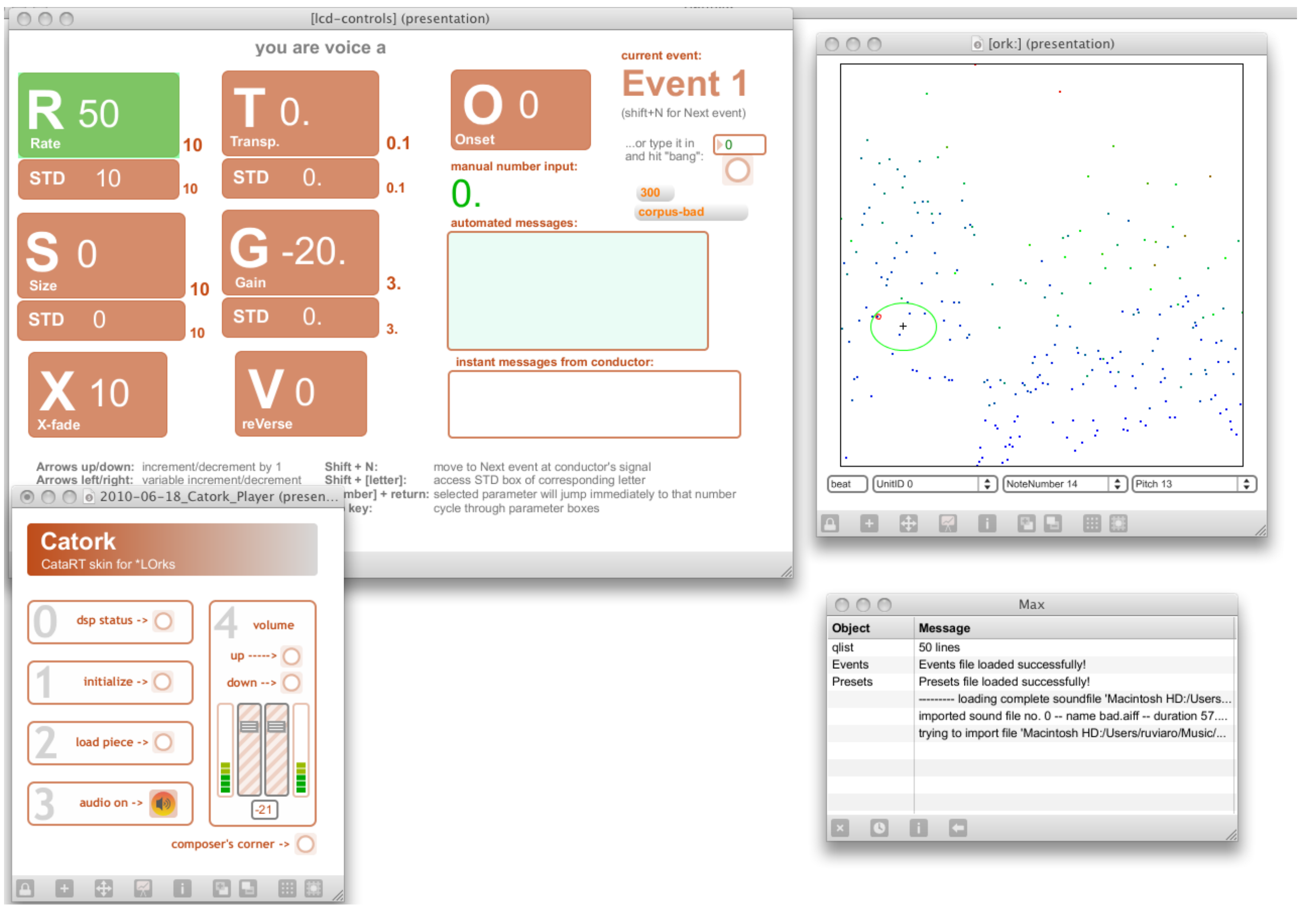
We are almost done. Close the “load-piece” window and go back to Catork's first window.

On checklist item #3, “**audio on**”, click on the button to turn audio on. It will turn red to indicate that it is on.

Finally, on checklist item #4, “**volume**”, click to turn volume up (or alternatively use the slider).

You are about to get the first sounds out of Catork. Press the shortcut Shift + N on your keyboard, and—if all goes well!—you should be now getting sound from your speakers.

After hitting Shift + N, you should be seeing something like this:



```
Arrows up/down:    increment/decrement by 1        Shift + N:          move to Next event at conductor's signal
Arrows left/right: variable increment/decrement    Shift + [letter]:   access STD box of corresponding letter
                                                   mber] + return:     selected parameter will jump immediately to that number
                                                   key:                cycle through parameter boxes
```

```
Object    Message
qlist     50 lines
Events    Events file loaded successfully!
Presets   Presets file loaded successfully!
          --------- loading complete soundfile 'Macintosh HD:/Users...
          imported sound file no. 0 -- name bad.aiff -- duration 57....
          trying to import file 'Macintosh HD:/Users/ruviaro/Music/...
```

Note that it now says “Event 1” on the top-right corner of the lcd-controls window (it was Event 0 when that window first opened). On the [ork:] window, you now see hundreds of colored dots. On the Catork_Player window, the meters indicate the sound output level.

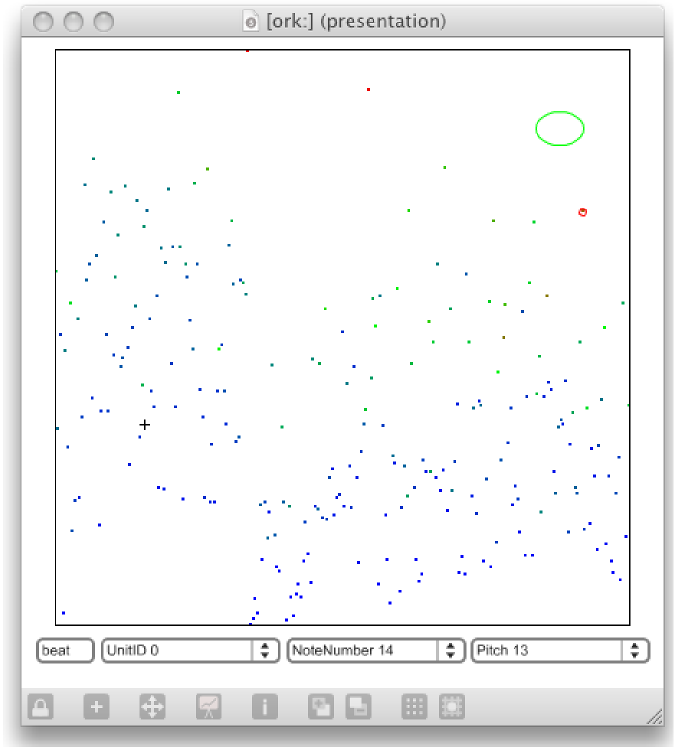
Also note: though not mandatory, you may find useful to open the default Max Window (the one you see at the bottom-right part of the screenshot below). It displays various kinds of system messages that help you see what is going on. You can make the Max Window appear with the shortcut Shift + M.

You can make transformations to the sound by changing the numbers on the [lcd-controls] window. Use the following keys:

R, S, X, T, G, V, O access any given parameter
Arrows up/down: increment/decrement by 1 (fixed)
Arrows left/right: variable increment/decrement
+ and – keys: change variable increment/decrement
Shift + N: move to next event
Shift + [letter]: access standard deviation box of corresponding parameter
[number] + return: selected parameter will jump immediately to that number
Tab: cycle through parameter boxes

The parameters Rate, Size, X-fade, Transposition, Gain, Reverse, and Onset (some of them with their respective Standard Deviation box, or STD) are CataRT's standard granulation parameters. All that Catork is doing here is to provide a playing interface based on shortcuts (thus avoiding use of the mouse almost completely).

You can also change things in the [ork:] window (the one with the colored dots):



Again, these are standard CataRT functions, simply presented in a slightly different way visually.

Move the green selection area by double-clicking anywhere on this screen. You can also resize the green selection area by clicking and holding as you drag the mouse left or right. Each dot on the screenshot above can be seen as a sample; usually, only dots that are within the green selection will be played.

The first drop-down menu (where it says “beat” in the screenshot above) allows you to change the trigger mode. “Beat” simply means that samples are played at the Rate you specify (for example, 1000 means one sample per second; 500 means one sample every 500 milliseconds, which is the same as two samples per second; and so on).

Try choosing “bow” instead of “beat”, for example. The “bow” trigger mode does not play anything unless you are actually moving the mouse around, as if you were 'bowing' on the dots. Other trigger modes have their own particularities. Please refer to the CataRT documentation for more information.

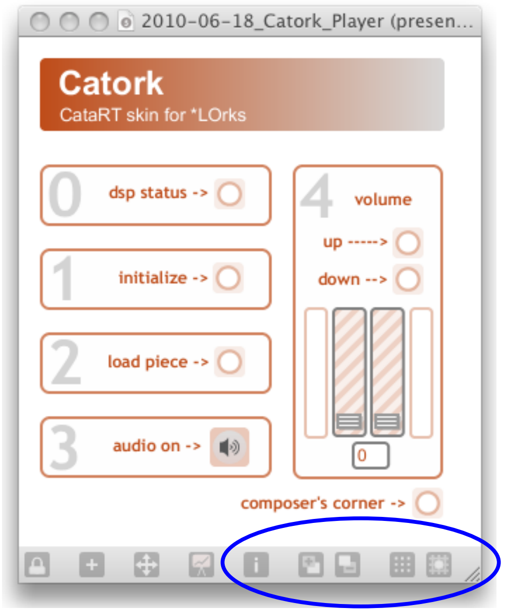
You can also change the arrangement of the colored dots (samples) in the two-dimensional space of the [ork:] window. The three bigger drop-down menus represent, respectively, the x axis, the y axis, and the color scale. Changing the way dots are organized will allow you to explore your timbre space in various different and interesting ways. Feel free to explore.

Please refer to the CataRT documentation for more information.

This is the end of the QuickStart section! Read on if you want to go into deeper waters.

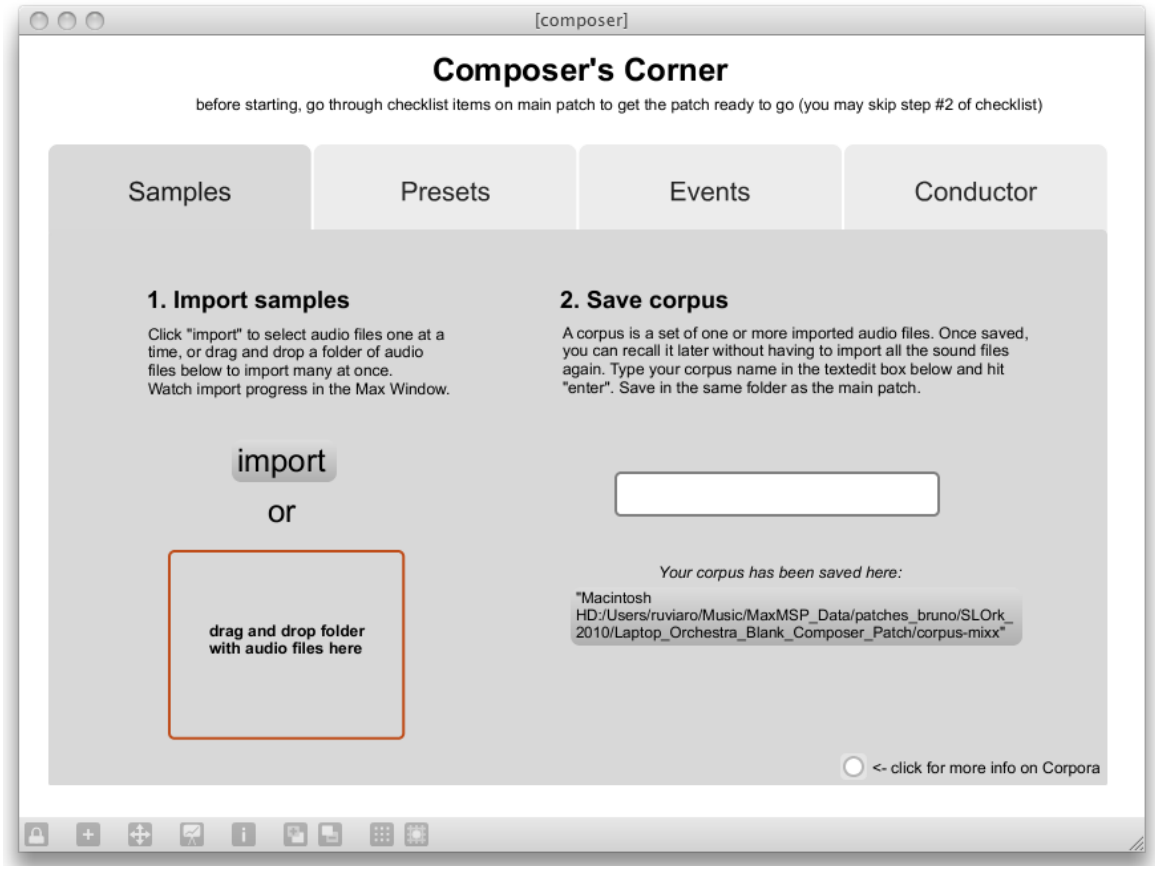
What's next?

All that you have done so far is basically to “play” Catork as a performer in a laptop ensemble would do (except there were no messages from a conductor). Now suppose you want to compose a piece choosing your own samples and defining your own Presets and Events. Let's check out the Composer's Corner, which you can access by clicking on the small button at the bottom-right corner of the Catork_Player patch.



The Composer's Corner

Open the Composer's Corner by clicking on the button indicated on the screenshot shown on the previous page. This is what you will see:


[composer]
Samples
Presets
Events
Conductor
import
or
drag and drop folder with audio files here
<- click for more info on Corpora

Composer's Corner

before starting, go through checklist items on main patch to get the patch ready to go (you may skip step #2 of checklist)

1. Import samples

Click "import" to select audio files one at a time, or drag and drop a folder of audio files below to import many at once. Watch import progress in the Max Window.

2. Save corpus

A corpus is a set of one or more imported audio files. Once saved, you can recall it later without having to import all the sound files again. Type your corpus name in the textedit box below and hit "enter". Save in the same folder as the main patch.

Your corpus has been saved here:

"Macintosh HD:/Users/ruviaro/Music/MaxMSP_Data/patches_bruno/SLOrk_2010/Laptop_Orchestra_Blank_Composer_Patch/corpus-mixx"

The Composer's Corner has four sections (tabs): Samples, Presets, Events, and Conductor. They represent the four main stages of composing with Catork. There is a lot of in-patch information there, including tips and suggestions of how to do certain things. Of course, the way you actually use the Composer's Corner will ultimately depend of the needs of your specific musical project.

Composing with Catork

A little bit of background first. Catork is one single system that treats composition and performance as two separate stages, even though overlaps may occur.[1] At performance time, one musician serves as the "conductor" (using the specific "Catork_Conductor" patch),

1 Of course, as long as you understand the mechanics of the system, you can also use it for improvisation or any other musical situation in which the boundaries of composition and performance are blurred.

responsible for sending out performance instructions (much like instant messaging) to all other musicians via the network. Performers use the main patch (“Catork_Player”) to interact in various ways with the typical parameters of concatenative synthesis, changing them via keyboard & mouse.

What (and how) performers actually play is defined beforehand by the composer(s). The composer, who will probably end up being the conductor too, will use both patches while working on the composition (Catork_Conductor and Catork_Player).

Compositional work in Catork is organized in four main steps represented by the tabs on the Composer's Corner window. Here's a quick rundown of what they entail:

– SAMPLES: choose samples you want to use. Import samples into Catork and save them as a *corpus* (or save different collections of samples as several different *corpora*; think of them as 'banks' of samples).
– PRESETS: play around with your *corpora* and save Presets that you want to use in the piece; a Preset is basically a set of parameters that you like and want to save.
– EVENTS: define the sequence of Presets that each laptop performer will play.
– MESSAGES: define the messages (cue list) that the conductor will be triggering and sending to the players.

The steps above should become more clear with practice. The composition of Presets, Events, and Messages boils down to the creation of four different text (.txt) files which, once finished, will have to be copied to the other laptops in order for them to play the piece.

Now let's take a look at each of the tabs on the Composer's Corner. Note: before you can use the Composer's Corner, you should have gone through the checklist on the Catork_Player patch in order to have everything ready to go. You can skip step #2 in this case: since your goal is to actually create a new piece, there is no need to load an existing piece into the system.

Samples

The first thing you have to do is to choose samples you want to use and load them into Catork.

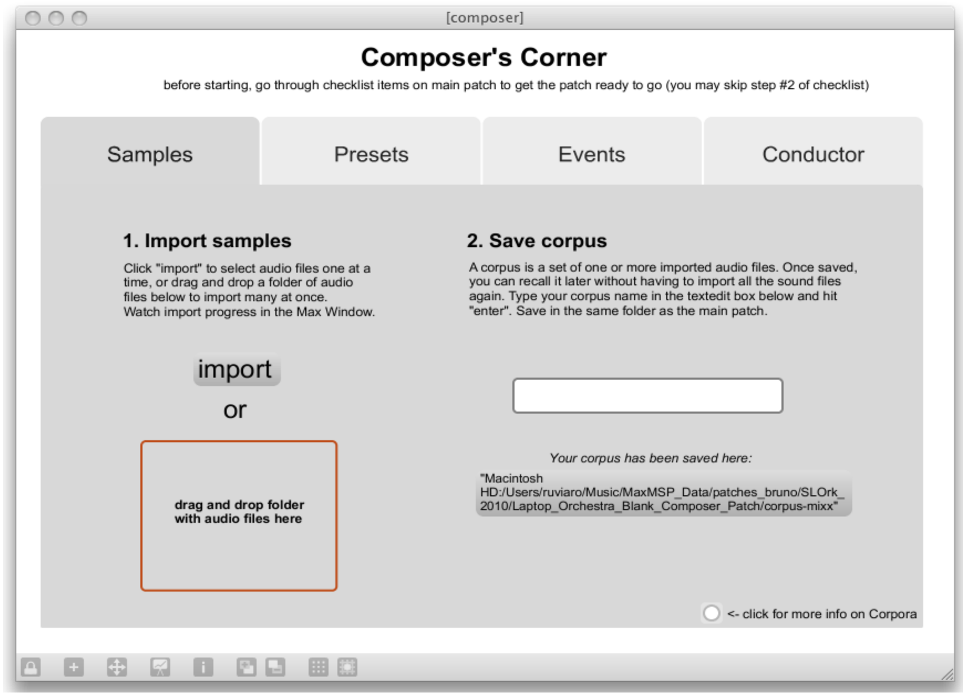
IMPORTANT: copy all sound files you want to use into the 'samples' folder inside Catork's main folder. Do not import sound files that are saved on other places on your computer.

Follow the instructions in the patch in order to import sound files. As you import them, CataRT will be already analyzing your sounds and displaying them as dots in the “colored-dots” window. If you have the Max Window open (command + M), you will see system messages indicating that the process is running.

Once you have imported them, you have to save your newly-created 'bank' of samples as a *corpus*.[2] A corpus is simply a set of imported audio files.

Once you have imported your sound files, save them as a *corpus.* You can save more than one *corpora*, if that makes sense to your musical project; for example, you might want to have one corpus made only of Beatles songs and another containing a mix of Mozart symphonies and truck sounds.

Note: if you save a corpus and want to move on to create a completely new one, you have to hit “initialize” again (step 1 of the checklist on the Catork_Player window). Otherwise, you will be just adding more sound files to the pool of previously imported ones, and all will be included in the next 'save corpus' action.


[composer]
Composer's Corner
Samples
Presets
Events
Conductor
import
or
drag and drop folder with audio files here
<- click for more info on Corpora

before starting, go through checklist items on main patch to get the patch ready to go (you may skip step #2 of checklist)

1. Import samples

Click "import" to select audio files one at a time, or drag and drop a folder of audio files below to import many at once. Watch import progress in the Max Window.

2. Save corpus

A corpus is a set of one or more imported audio files. Once saved, you can recall it later without having to import all the sound files again. Type your corpus name in the textedit box below and hit "enter". Save in the same folder as the main patch.

Your corpus has been saved here:

"Macintosh HD:/Users/ruviaro/Music/MaxMSP_Data/patches_bruno/SLOrk_2010/Laptop_Orchestra_Blank_Composer_Patch/corpus-mixx"

If you know beforehand what samples you want to use in your piece, you probably will have to do this entire “Samples” step just once. The saved corpus will then be there just waiting to be loaded the next time you open the patch. A saved corpus is nothing more than four text files with funny extensions (.ds, .sf, .sy, .ud); they tell CataRT all it needs to know about the sound files you chose.

If in the beginning of the composition process you are still unsure of what samples you want to use, feel free to try out different collections of sound files and save them as different *corpora.* Once you are done with experimenting, you can just delete the corpora you don't want anymore (simply delete the four corresponding text files).

2 Plural *corpora.*

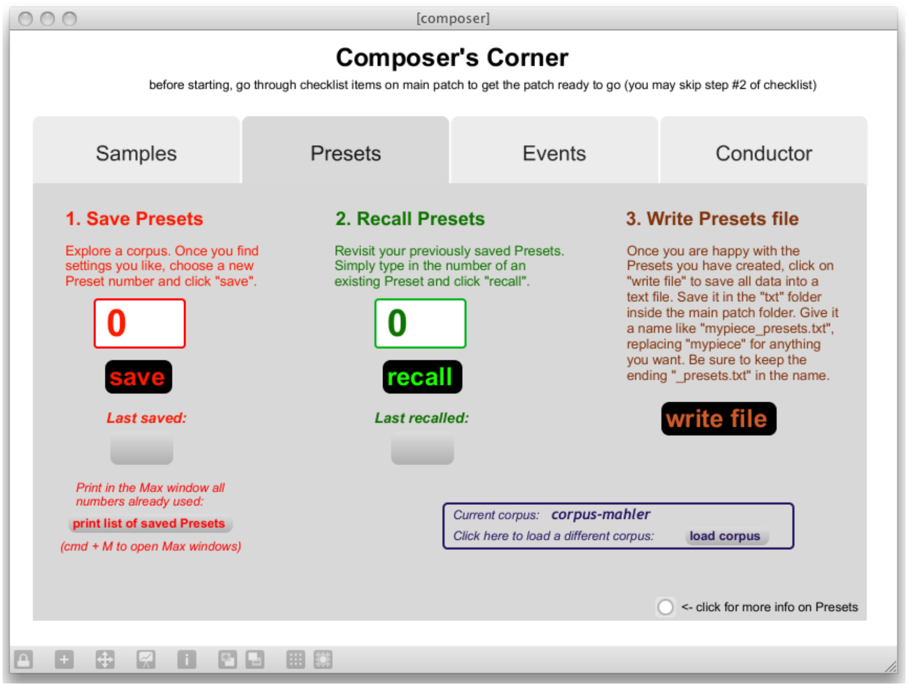
Presets

Now you have one or more set of samples (corpus) ready. The next step is to play Catork with your newly-created corpus, and look for specific combinations of parameters that you want to use in your piece. That's when you start saving your own Presets. A Preset is just this:

- a choice of corpus
- a particular set of numbers on the [lcd-controls] window
- a specific arrangement of the colored dots on the two-dimensional space (x, y axes)
- a trigger mode
- a position of the green selection on the 2D space

Catork takes care of all the memorizing. You just have to choose a number for your Preset. You will use that number later to recall it.[3]

Note that Presets are static, "frozen" combinations of values. Catork does not have the ability of memorizing changes in time (i.e., it cannot 'record' the data of you performing). Part of the work of the composer, in Catork, is precisely to define what performers will do between the triggering of two different Presets (transitions, sudden changes, repetitions etc.). These instructions will be written into performance instructions, as we will see in the tab "Messages".


[composer]

Composer's Corner

before starting, go through checklist items on main patch to get the patch ready to go (you may skip step #2 of checklist)

1. Save Presets

Explore a corpus. Once you find settings you like, choose a new Preset number and click "save".

Last saved:

Print in the Max window all numbers already used:

(cmd + M to open Max windows)

2. Recall Presets

Revisit your previously saved Presets. Simply type in the number of an existing Preset and click "recall".

Last recalled:

3. Write Presets file

Once you are happy with the Presets you have created, click on "write file" to save all data into a text file. Save it in the "txt" folder inside the main patch folder. Give it a name like "mypiece_presets.txt", replacing "mypiece" for anything you want. Be sure to keep the ending "_presets.txt" in the name.

Current corpus: corpus-mahler

Click here to load a different corpus:

<- click for more info on Presets

3 In case you are really curious, this is what a Preset really looks like when it's saved. It's just a list of stuff saved in a text file. The first number is the Preset number chosen by the composer:
100, corpus corpus-mahler trigger 2 set-descriptors Pitch UnitID Periodicity rate 250 rate_std 100 duration 600 duration_std 0 xfade 250 transposition 0. transposition_std 0. gain -6. gain_std 0. reverse 0 onset_std 0 radius 0.48 setpos 0.320802 0.498747 freeze 1;

A typical workflow in the Presets tab is: you play the patch for a while, find settings you like, save a Preset; then play the patch again, find another good moment, save it as a Preset; etc. Sometimes you may want to recall some of the saved Presets to remember some aspects of it, or just to double check that it was properly saved: that's when you use step #2, “Recall Presets”. After you have spent some time saving a bunch of Presets, click on “Write File” (step #3) to save all your work into a text file.

Events

The Events tab does not have much in it, except the explanation of the concept of Events and an example. The reason is simple: the actual work of creating Events is not done within the patch, but rather in any text editor of your choice (such as Text Edit on the Mac). The goal is simply to create a text file following some simple rules.

Here's a quick explanation of the concept of Events in Catork. Suppose you have created four different Presets in the previous step (remember, Presets were just static sets of parameters that generated some sound you liked). Let's say you saved a few Presets with the numbers 101, 102, 103, 104 and 105. You have an ensemble of at least four laptops plus one conductor laptop. You would like to start the piece with each laptop playing a different Preset. Let's say your performers are labeled 'a', 'b', 'c', and 'd'. Whenever the conductor signals the performers to start the piece, they will hit Shift + N to trigger Event 1 of the piece. You want to define Event 1 to be something like this:

Event 1
Laptop 'a' plays Preset 105
Laptop 'b' plays Preset 101
Laptop 'c' plays Preset 102
Laptop 'd' plays Preset 103

As you can see, an Event is pretty much like defining the different “parts” of your polyphony; or, to use another analogy, it is like “orchestrating” the simultaneity of sounds played by different musicians. An Event just says: “at this point in time, these Presets will be played at simultaneously.”

Working in this way, you can imagine a possible sketch of Events that you want to happen in succession:

	Event 1	**Event 2**	**Event 3**	**Event 4**	**(etc...)**
Laptop 'a'	Preset 105	Preset 104	Preset 103	Preset 102	...
Laptop 'b'	Preset 101	Preset 104	Preset 101	Preset 102	...
Laptop 'c'	Preset 102	Preset 101	Preset 103	Preset 102	...
Laptop 'd'	Preset 103	Preset 105	Preset 104	Preset 102	...

Every time a performer hits Shift + N on his/her laptop, the patch would move automatically to the next Event predetermined by you in a “score” like this. In between Event triggerings, performers may be changing parameters according to whatever rules or instructions you have defined.

So how should you write the Events text file to specify something like the Events in the table above? Here's a template:

```
------------------------ 0;
0 0;
------------------------ 1;
score_call-preset a 105;
score_call-preset b 101;
score_call-preset c 102;
score_call-preset d 103;
0 1;
------------------------ 2;
score_call-preset a 104;
score_call-preset b 104;
score_call-preset c 101;
score_call-preset d 105;
0 2;
------------------------ 3;
score_call-preset a 101;
score_call-preset b 103;
score_call-preset c 101;
score_call-preset d 104;
0 3;
------------------------ 4;
score_call-preset a 102;
score_call-preset b 102;
score_call-preset c 102;
score_call-preset d 102;
0 4;
```

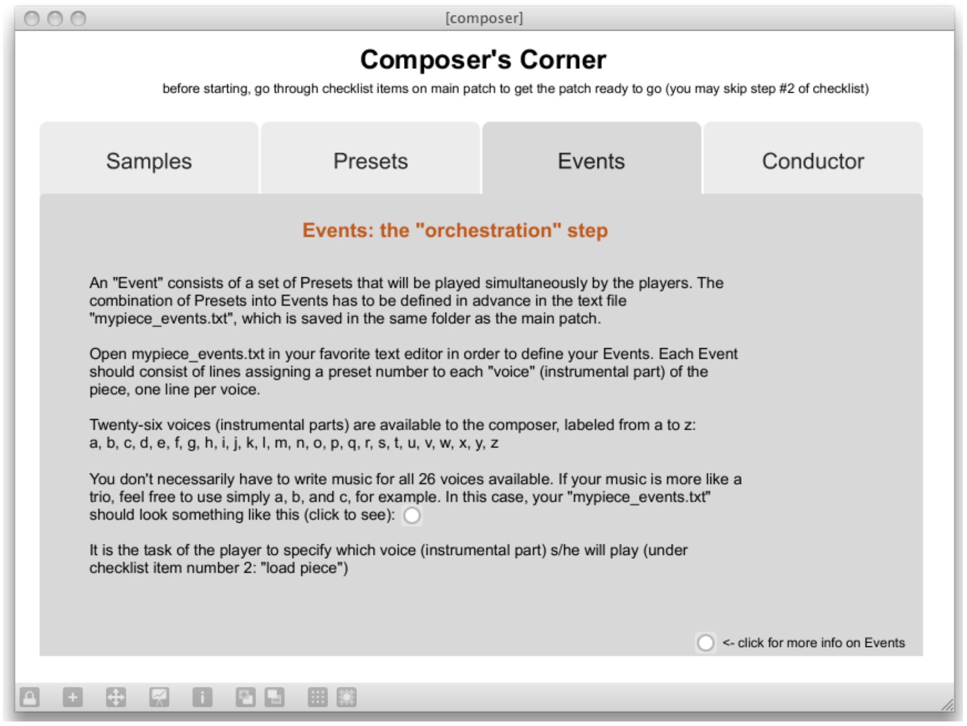
The line with dashes followed by a number indicate the beginning of an Event. The lines starting with “score_call-preset” specify the content of each Event. The last line of an event is always the number zero followed by the Event number. All lines have to end with semi-colon.

Note that the only thing you need to do is to change the Preset Values in this file. Everything else should remain as it is, including the line with dashes (the exact number of dashes is important). When you create your own Events text file, just copy and paste from the template, and change only the portions you need.

In the example above, I referred to each laptop as a letter: 'a', 'b', 'c', and 'd'. In reality, these letters do not have to represent only one single laptop each. If you have an ensemble of 8 laptops, for example, you could have two laptops playing part 'a', two playing part 'b', maybe three playing part 'c' because that's a special part, and only one playing part 'd'. Think of these letters as “instrumental parts”, or “voices” in a choir: they are representing actual lines of musical material (polyphony). How many laptops will be actually performing each line is up to you (and the size of your ensemble).

You can compose as many voices or parts as you want, up to twenty-six (from 'a' to 'z').[4]

Your Events text file should be saved in the txt folder inside the main Catork folder.


[composer]

Composer's Corner

before starting, go through checklist items on main patch to get the patch ready to go (you may skip step #2 of checklist)

Events: the "orchestration" step

An "Event" consists of a set of Presets that will be played simultaneously by the players. The combination of Presets into Events has to be defined in advance in the text file "mypiece_events.txt", which is saved in the same folder as the main patch.

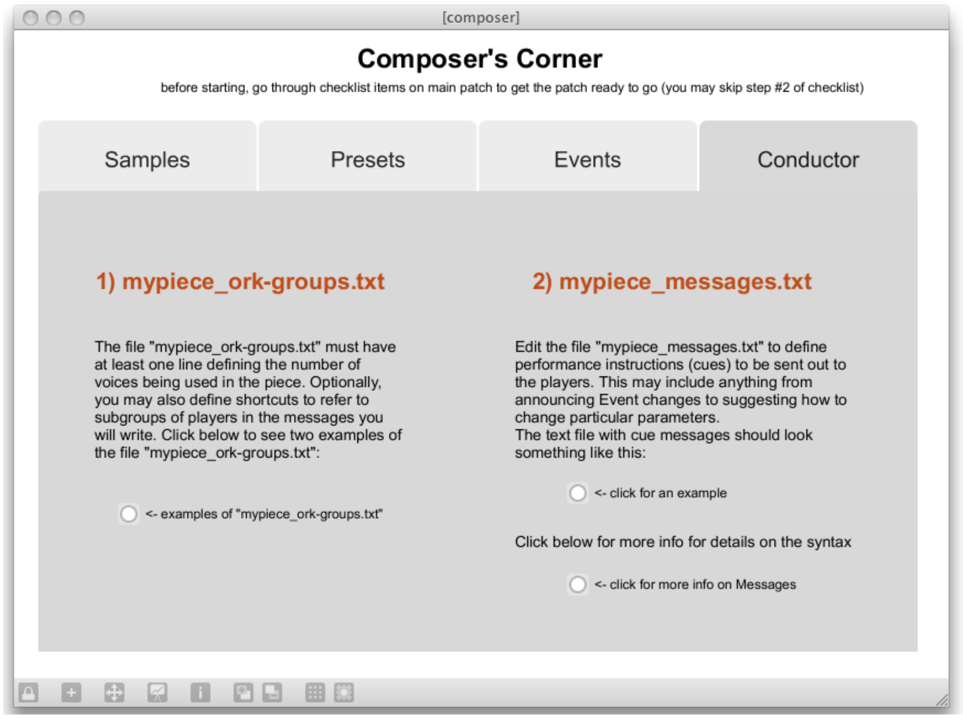
Open mypiece_events.txt in your favorite text editor in order to define your Events. Each Event should consist of lines assigning a preset number to each "voice" (instrumental part) of the piece, one line per voice.

Twenty-six voices (instrumental parts) are available to the composer, labeled from a to z:
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z

You don't necessarily have to write music for all 26 voices available. If your music is more like a trio, feel free to use simply a, b, and c, for example. In this case, your "mypiece_events.txt" should look something like this (click to see):

It is the task of the player to specify which voice (instrumental part) s/he will play (under checklist item number 2: "load piece")

<- click for more info on Events


[composer]

Composer's Corner

before starting, go through checklist items on main patch to get the patch ready to go (you may skip step #2 of checklist)

1) mypiece_ork-groups.txt

The file "mypiece_ork-groups.txt" must have at least one line defining the number of voices being used in the piece. Optionally, you may also define shortcuts to refer to subgroups of players in the messages you will write. Click below to see two examples of the file "mypiece_ork-groups.txt":

<- examples of "mypiece_ork-groups.txt"

2) mypiece_messages.txt

Edit the file "mypiece_messages.txt" to define performance instructions (cues) to be sent out to the players. This may include anything from announcing Event changes to suggesting how to change particular parameters.
The text file with cue messages should look something like this:

<- click for an example

Click below for more info for details on the syntax

<- click for more info on Messages

4 If you remember checklist step #2 on the Catork_Player window (“load piece”), that is where the choice of part happens: players have to choose the letter corresponding to the part they will be playing.

<u>Conductor</u>

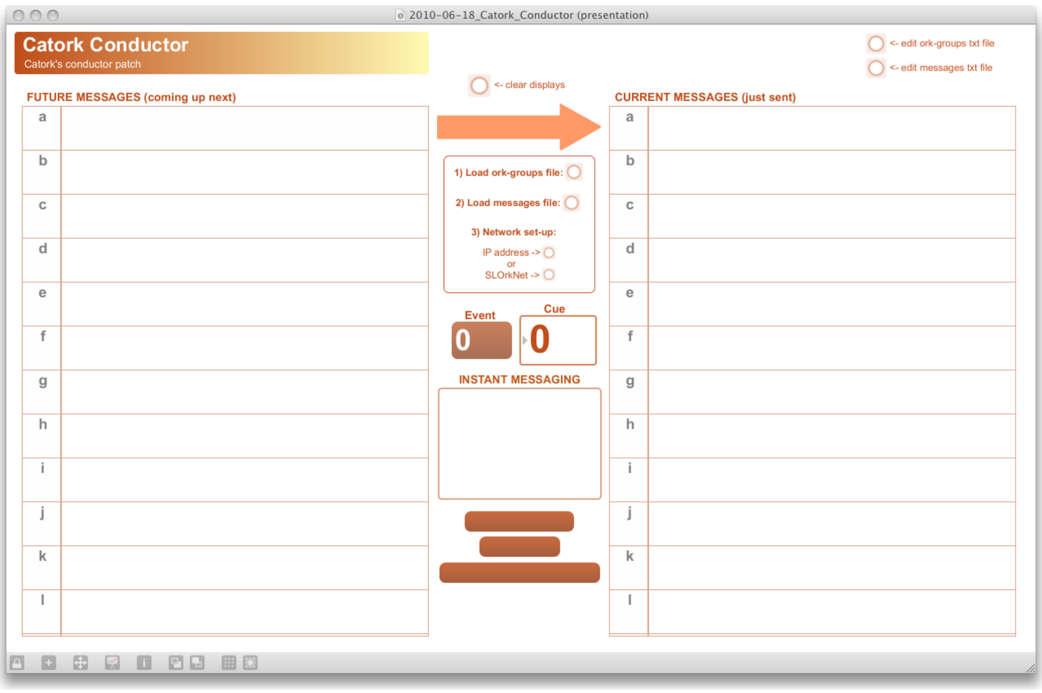
The Conductor tab, like the Events tab, does not have much in it other than some explanatory text. Again, the actual work is done through the creation of text files. In this case, two text files have to be created and saved in the same txt folder inside Catork's main folder.

The final task of the composer is to write a “cue list”: a set of messages to be sent out to the players via the network. These messages are the performance instructions. In the Events example given a few pages earlier, we described how the performer would hit Shift + N to move from Event 1 to Event 2, for example. With performance instructions, the performer may be asked to change parameters for any length of time between the triggering of Events. The composer may want to specify or suggest any number of things to the performer: a particular way to transition from one Event to another, a change of behavior, a call for improvisation with the cursor on the two-dimensional space, etc.

Messages written in the text file named “mypiece_messages.txt” will be triggered sequentially by the conductor. These are fixed and cannot be change during performance time. For more info, Please refer to the instructions and examples in the “Conductor” tab of the Composer's Corner subpatch.

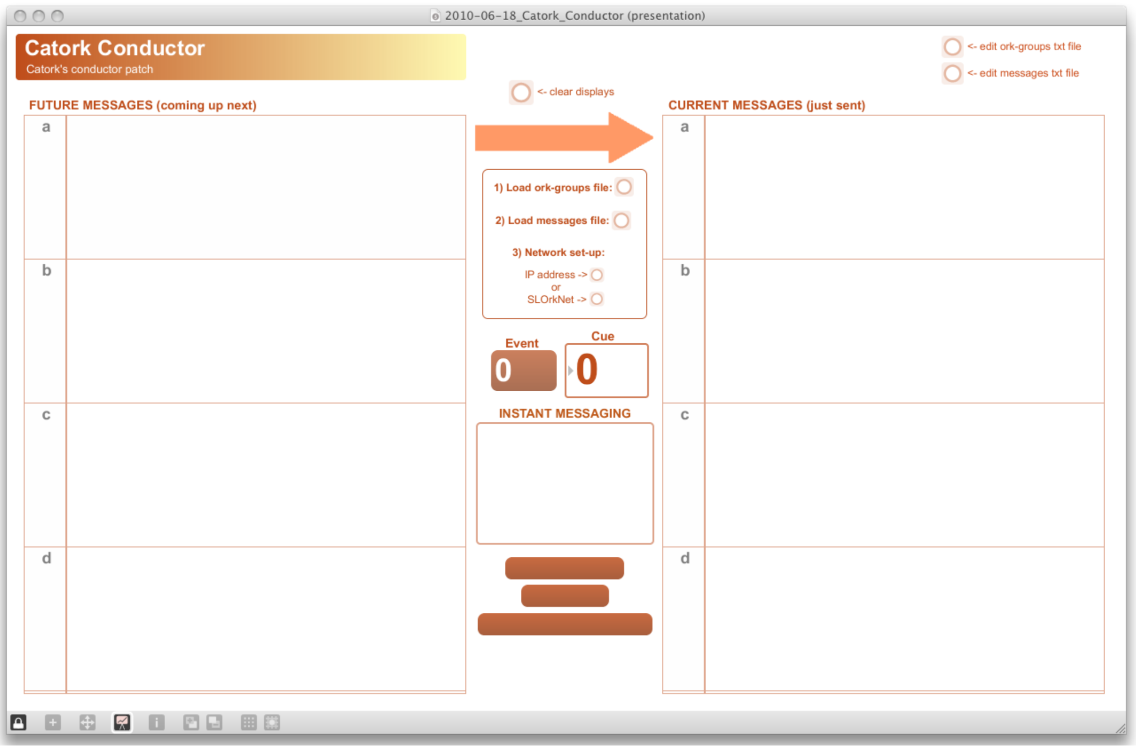
The Conductor Patch

Finally, one musician of the ensemble will take the role of the conductor. The conductor opens a different patch: 2010-06-18_Catork_Conductor.maxpat (date on file name may be more recent). Here's a screenshot of the Conductor Patch:



After the patch has been opened, there is a short checklist for the conductor to follow. The three steps are:

1) Load ork-groups file: this is one of the text files defined by the composer in the process of composition. It contains at least one line specifying the number of voices being used in the piece. Loading this file will adjust the two tables accordingly. In the screenshot above, note that the 'messages' tables had 12 rows, from 'a' to 'l'. A piece with only 4 voices (or parts) will have the conductor patch adjusted like this:



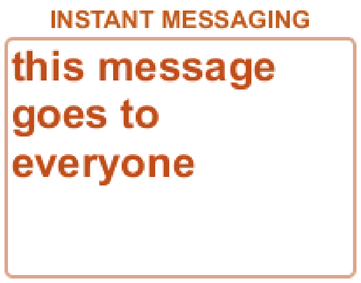
2) Load messages file: this is the text file that contains the actual messages with performance instructions.

3) Network set-up: specify the IP addresses of all machines in the ensemble. There is also a customized assignment subpatch for the Stanford Laptop Orchestra (SLOrk), since the patch was created for this ensemble.

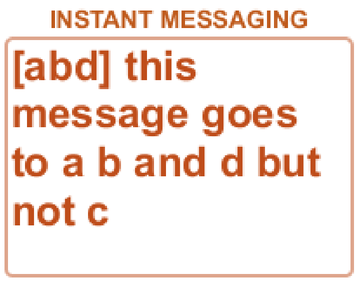
In order to trigger (send out) messages in the cue, all the conductor needs to do is to select the number box “Cue” and use the arrow up key to increase the cue number. The table on the right (“Current messages”) will display the messages that have been last sent. The table on the left (“Future messages”) will display the messages coming up with the next cue.

The conductor has also the ability of writing and sending instant messages to players by using a specific “instant message” box. These messages do not interfere with the fixed messages from the cue list (on the players' screen, they appear in a different box). This way, the conductor is able to communicate directly with the performers with real-time suggestions relevant to an ongoing performance, often complementing the instructions from the fixed messages in the cue list.

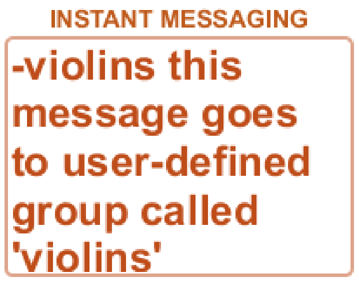
If you just write a message in the instant message box and hit enter, the message will be sent to the whole ensemble. You can, however, narrow down the addressees of the message and select specific subgroups of the ensemble to which to send the message. There is a specific syntax for this. Please refer to the the Conductor's tab on the Composer's Corner in the Player Patch for more information. The syntax for instant messages is the same as the syntax for writing the “Messages” text file.[5] Here are a few quick examples:



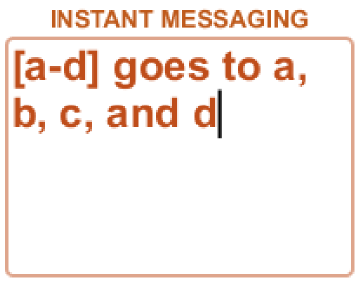
this message goes to everyone



[abd] this message goes to a b and d but not c



-violins this message goes to user-defined group called 'violins'



[a-d] goes to a, b, c, and d

5 The syntax of letters between brackets comes directly from OSC routing syntax.

Portability

After you finish composing your piece, you need to transfer the appropriate files to the other laptops of the ensemble. Assuming the other laptops in the ensemble already have MaxMSP (or Max Runtime), FTM library, and the Catork folder, here are the piece-specific files that you need to copy to all other machines:

- mypiece_events.txt [inside 'txt' folder]
- mypiece_messages.txt [inside 'txt' folder]
- mypiece_ork-groups.txt [inside 'txt' folder]
- mypiece_presets.txt [inside 'txt' folder]
- all audio files (samples) you imported into Catork [inside 'samples' folder]
- saved corpus files (four .txt files per corpus, with names such as corpus-mypiece.ds.txt, corpus-mypiece.sf.txt, corpus-mypiece.sy.txt, corpus-mypiece.ud.txt) [inside main Catork folder]

Questions, Comments, Suggestions?

Feel free to send them to me: ruviaro@ccrma.stanford.edu

I would love to hear about your (good or bad) experiences using Catork.

Most of Catork was developed for my piece Intellectual Improperty 0.6, for laptop orchestra. The piece was premiered by SLOrk (the Stanford Laptop Orchestra) in the Spring of 2010. The entire project was part of my doctoral dissertation, defended on June 2, 2010 at CCRMA / Stanford University.

Bruno Ruviaro

Catork FAQ

What is Catork?

Catork is CataRT-based system for composition and performance of electronic music. It was particularly designed to be used with laptop orchestras (which apparently tend to have names ending with "Ork", such as SLOrk, PLOrk, L2Ork, etc.). Inside Catork, CataRT (by Diemo Schwarz) does all the hard work. Catork is just an interface built on top of CataRT to facilitate compositional work.

What is CataRT?

CataRT, created by Diemo Schwarz, is the MaxMSP implementation of the technique of concatenative synthesis. For more info about CataRT and concatenative synthesis, please visit http://imtr.ircam.fr/imtr/CataRT

What do I need to run Catork?

You need a computer with MaxMSP or Max Runtime installed:
http://cycling74.com/downloads/

After installing Max you also need to install the FTM library:
http://ftm.ircam.fr/index.php/Download

Finally, you need to copy the entire Catork folder into your Max directory (Applications > Max). As of June 2010, a copy of CataRT 1.1.2 is already included within the Catork folder, so you don't have to download CataRT itself. Should you want to try newer versions of CataRT, please go to <http://imtr.ircam.fr/imtr/CataRT>.

Who can use Catork?

Anyone. Catork may be specially interesting for laptop orchestra beginners, but experienced electronic musicians may find it useful too.
If you are a beginner, you can compose a laptop ensemble piece and perform it even if you don't have a lot of programming experience. You do have to understand a few basic concepts of concatenative synthesis and the basic structure of Catork, but hopefully you would be spending more time exploring sounds and creating your music, with a minimum amount of programming.
If you are an advanced user, you can of course open up the whole thing and adapt it to your specific needs.

How does it work?

Catork is one single program (or rather, a MaxMSP patch) that treats composition and performance as two separate stages, even though overlaps may exist. Of course, as long as you understand the mechanics of the system, you can also use it for improvisation or any other musical situation in which the boundaries of composition and performance are blurred. At performance time, one laptop musician serves as the “conductor” (using the specific “Catork_Conductor” patch), responsible for sending out performance instructions to all other musicians via the network, much like instant messaging. The rest of the ensemble uses the main patch (“Catork_Player”) to interact in various ways with the typical parameters of concatenative synthesis, changing them via keyboard & mouse.
What performers play is defined beforehand by the composer(s). The composer, who will probably end up being the conductor too, uses both patches for composition (Catork_Conductor and Catork_Player). The composition stage is characterized by four main steps:

- SAMPLES: choose samples you want to use. Import samples into Catork and save them as a *corpus* (or save different collections of samples as several different *corpora*; think of them as 'banks' of samples);
- PRESETS: play around with your *corpora* and save Presets that you want to use in the piece;
- EVENTS: define the sequence of Presets that each laptop performer will play.
- MESSAGES: define the messages (cue list) that the conductor will be triggering and sending to the players.

The steps above should become more clear with practice. The composition of Presets, Events, and Messages boils down to the creation of four different text (.txt) files that, once ready, will have to be copied to the other laptops in order to play the piece.

I just downloaded Catork. How do I just get some sound going to try it out?

Assuming you have Max, FTM and the Catork folder properly installed, here's how to get your first quick sound out of Catork:

- Checklist Item #0, “dsp status”: select either your built-in audio driver or your external audio interface, if any.
- Checklist Item #1, “initialize”: click and watch two new windows appear. Spread them nicely on your screen.
- Checklist Item #2, “load piece”: a small window appears with three simple steps to load the demo included in the Catork folder. You will find the Events and Presets text files in the folder 'txt', inside Catork's main folder. In the option “Select your part”, just choose 'a' or 'b' for now.
- Checklist Item #3, “audio on”: click on the button to turn audio on.
- Checklist Item #4, “volume”: click to turn volume up (or alternatively use the slider).
- Use the shortcut Shift + N on your computer keyboard to recall Event 1 (pre-defined in the demo). You will see hundreds of colored dots appear in one of the screens. There should be sound going on by now. You can play now by clicking on the colored dots

and changing the green selection, or by changing numbers on the [lcd-controls] window. Refer to the bottom of that window for all shortcuts you need to know in order to play with all the parameters.

Once I'm done with composing, what files do I have to copy to the other laptops in order to play my piece?

Assuming the other laptops in the ensemble already have MaxMSP (or Max Runtime), FTM library, and the Catork folder, here are the piece-specific files that you need to copy to all other machines:

- mypiece_events.txt [inside 'txt' folder]
- mypiece_messages.txt [inside 'txt' folder]
- mypiece_ork-groups.txt [inside 'txt' folder]
- mypiece_presets.txt [inside 'txt' folder]
- all audio files (samples) you imported into Catork [inside 'samples' folder]
- saved corpus files (four .txt files per corpus, with names such as corpus-mypiece.ds.txt, corpus-mypiece.sf.txt, corpus-mypiece.sy.txt, corpus-mypiece.ud.txt) [inside main Catork folder]

Is there something similar for Pd?

Unfortunately there is no Catork for Pd, but there is an implementation of concatenative synthesis for Pd that resembles CataRT: it is called timbreID, by William Brent. Please take a look at http://williambrent.conflations.com/pages/research.html#timbreID