21M.361 Composing with Computers I (Electronic Music Composition)
Spring 2008

21M.361: Composing with Computers I (Electronic Music Composition)

Peter Whincop

Spring 2008 OCW

[All lab notes are being significantly revised for next term to reflect upgrades in software, different organization, and the incorporation of my own notes on DSP.]

Lab 4.1: Basic Max/MSP

Firstly (for the second time)... <-------- READ !!!!!!!!

Can you please clean up your folders, in other words, remove everything except files you think you might use for future assignments (e.g. your assignment 3.2 sounds might be useful for assignment 3.3). You might want to back your files up rather than delete them. The Mac's CD drive is actually a DVD-R writer. I will clear out everyone's folders in a week unless the word '-keep' is appended to their name (of the folder, not every nested folder). There is now a firewire drive called 'Enemy'; you can back things up to the folder 'student-back-up.' Only one of you has followed this instruction from last week. Next week I will be clearing the hard drive up to avoid running out of space. Thanks.

Max/MSP basics:

0. You should have already gone to http://www.cycling74.com/downloads/maxmsp for a 30-day trial full version.

1. **When installing, make sure you also install documentation**. This will include pdfs for Max and MSP separately, for reference, for tutorials, some other stuff, and folders of patches for each set of tutorials. You should already have gone over some of the Max tutorials; MSP tutorials are a little more difficult, as they introduce new things faster, but if read and played with in conjunction with the tutorial and reference pdfs (NB. there different Max and MSP pdfs) they might make sense.

2. In lab, we looked at the basics of Max/MSP and went over certain messages and objects. I shall list those below, barely explaining them; you can use manuals and help files. The Max Overview pdf might be useful to read.

3. **Command-n opens a new patch** (NB. patch=patcher). The Max window is for debugging. If it vanishes or gets buried, use command-m. To clear the Max window, focus on it and type command-x. To change the font size of the patch windows, focus on the Max window, choose the font size from the Font menu, and any new patches will use that font size. It is not applied to patches that are already open. You can change the font size of an existing patch by highlighting everything and changing the font size; if the Auto Fix Width option is checked in Options, the boxes won't have line breaks, but they might be smashed together.

4. **There are two modes of operation in Max/MSP: edit mode and run mode**. Edit mode allows you to build patches, run mode allows you (or someone else) to run patches. You can also run patches in edit mode, as will be demonstrated in a second. Everything in these labs is geared towards Mac users; Windoze users take note that option(mac)==alt(windoze) and command(mac)==control(windoze) Command-e alternates between edit and run modes. Edit mode is obvious: you have a set of icons at the top of the patch window. Run mode, no icons. You can also alternate by command-clicking on the background of a patch. That's what I do. Better still, when you are in edit

mode, hold down command and operate on an object; it won't move it about as expected in edit mode; it will do what it would do in run mode. I tend to stay in edit mode almost all the time, and use command-click-object to fake run mode.

5. **Help. Option-clicking on an object brings up a help patch**. A help patch is a patch, so if you put it in edit mode, change it, and close it, do *not* save the changes. The help patches can be fairly useless and random. Use also the reference manual, which is far better. Also, in edit mode if you **hover above inlets and outlets to objects, their function and data types and possible ranges will be displayed** in the lower left part of the patch. You will shortly read about messages. **An object can take certain messages. You can find out what these are by option-control-left-clicking on the object**.

6. **Operational things.** To check the correct soundcard is being used, go to Options>DSP Status... and make sure the driver and input and set to Digidesign. Don't worry about the lack of entry in output. Ignore the rest. In the Options menu, selecting New Object List will cause Max to bring up a window (without correct scroll bars for each column...) with all possible objects when you place a new object (actually, it's not a complete list, but it's pretty good, especially as it puts the objects into categories). I personally don't use New Object List, but it is very useful when starting out. Most people don't like segmented patch cords, but I do. Without them, you have to hold down the mouse to connect objects/messages. With segmented patch cords you can make crazy Max art, and you don't have to hold down the mouse. Also, have help in locked patches checked; don't worry what this means yet. Auto fix width is also useful to have checked. So is Extended File Preview, Max Window at Startup, Float Display Correction, and Assistance.

7. **There are three basic things in Max/MSP: objects, messages, and comments**. They are the three leftmost icons, in that order. Note their different appearances: comments are dotted boxes, messages are single line boxes, and objects have double lines on top and bottom. Comments are, well, comments; they do nothing functionally. But you have to comment your patches liberally, for your own sake—so you know what is going on and will remember in a year's time—for my sake—so I know you are not copying and pasting from some other sources—and for other users of your interactive patches. Sometimes it's just interesting to know how something works. Messages are just data—numbers, strings, lists—that are pushed around the Max flowchart. (Audio data is a little different; it's not really data.) Objects *do* things. Like add, play, filter, draw, decide, count, etc.

8. **You place a new object by clicking on its icon on the tool bar**, and clicking it where you want it to go on your patch. **Apple-d duplicates a highlighted object**, or set of objects and their cables, **option-dragging an object will copy it**. **Right-clicking on the object icon on the toolbar will bring up a list of objects you have already used** and you can select from that lift with a left-click.

9. **You connect outlets to inlets**. When there is an outlet, hovering the mouse over it will make the tiny black rectangle larger, and when you have a cable in the making, an inlet will get bigger if it is of the right (data) kind.

10. **To delete an object, message, or comment, draw a rectangle around it and hit delete**. To delete a patch cord, highlight it, it will turn bold (or red if it is an audio cord), and hit delete. Max (data) cords are black lines, MSP (audio) cords are yellow squiggles. **You can delete multiple patch cords** by holding down option and drawing a rectangle around the patch cords you wish to annihilate. If you want to abandon a patch cord while making one, hit escape. If

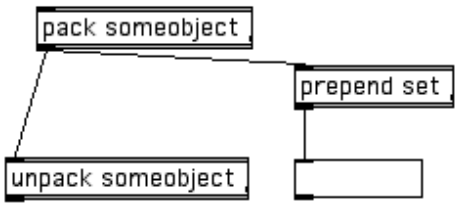
you want to abandon an object, message, etc., while it is still floating about before being placed, hit delete.

Max/MSP objects and messages:

0. In these notes, **[square brackets] mean an object, and lvertical barsl mean a message**. Don't forget numbers can act as messages, but I won't put them between vertical bars.

1. **[print]** will print to the Max window. [print *arg*] will identify what you have printed by *arg*. This is useful for debugging.

2. **Clicking on a message will send the message**. (You will learn later that "**set**"ting a message will send it, but will stop it at its next message point.) The special message **[bang] means "do it"**; it makes certain objects do things; it is the message resulting from certain objects; it is the same as clicking on a message if that is what it is connected to. **Bang is also the round button inside a tiny square that blinks yellow when clicked on or activated by some other process in Max**. **To see a message, you can print it using [print]**; or you can **divert it to the object/argument [prepend set] and then to an empty message box**. That way you can see the message as a message. If you want to use that message, don't take it from the second message box (following the prepend), rather, straight from the source.



3. **There are two number types: integers and floats (decimals)**. They can be found as the icons with > signs; the one with a dot is floats. In run mode you can scroll these; scrolling floats depends on whether you do it to the left or to the right of the decimal point. **Number boxes can initiate messages, or can show the result of operations**. There is a special window in Max called the Inspector. Some graphical objects have attributes that can be controlled by messages; for example the maximum and minimum values of a number box can be set. But this can be set from the Inspector window. One option of the number Inspector is Output Only on Mouse-Up. This means the number won't change until you unclick the mouse. In run mode, a number box can be clicked on and a number entered rather than scrolled to. When this happens, the triangle turns yellow. Something to note: a number box does not automatically load data into an object when a patch opens, and, additionally, the value of a number box is not stored when a patch is saved. (This can be done using [preset] which we cover in Lab 4.3.

4. **Arithmetic**: not so simple. **In many Max objects, all the right inlets put information in registers (memory), and the leftmost inlet will do the same, but also bang the object**, i.e., make it do its thing. This is true for arithmetic

objects, e.g. [+] [-] [*] [/] and others. **Read the manual carefully on how this works**; it will cause you at some point to get unexpected results, and correcting any error will give you the results you expected.

5. **Right-to-left**. Max works top-to-bottom by virtue of being a flowchart; an object below an other graphically might be on top of it conceptually if that is what the cables say. The left-to-right thing is trickier; read the manual carefully on this. It can be the cause of problems (even now, it throws me occasionally). Next week we will learn **[trigger]** or **[t]** for short, which forces the order of events, independent of graphical representation. From next week, that is what I will insist upon.

6. **[x] is called toggle; it produces a 0 or 1 when clicked**. **Bang alternates it, just as clicking it does; 0 sets it to 0, any non-zero number sets it; but all numbers passe through. A string will return an error** (see Max window for this when it happens). [x] is used to start and stop various objects, such as [dac~], [sfplay~], and metro. **NB. [x] isn't an object with an 'x' in it; it is just convenient shorthand for the toggle box, since they kind of look the same.**

7. **[metro] is a metronome**. The optional argument is time in ms. The right inlet overrides this, and can be changed on the fly. [x] starts or stops the metronome; the output is a bang, used to trigger other objects.

8. Bang causes certain events to happen: an example is **[random]**. The optional argument is the integer value such that [random] will output a random integer between 0 and n-1. You will have to use clever arithmetic to get, say, a random value between 0.5 and 7.5 in increments of 0.2. (Remember that +1 is sometimes need for inclusive ranges.)

9. **[delay]** (or **[del]**)receives a bang, and delays the output of that bang by the argument in ms, which can be overridden by the right inlet. Stringing delays together is better than initiating a number of them, since it means only one clock in the Mac scheduler is required. An integer to the left inlet initiates a delay of that value in ms.

10. **[gate] can be switched on or off using [x]** in the left inlet; when it is on, a message (including a number) received in the right inlet will be passed to the output. If the gate is off and a message is passed to it, then the gate is switched on, the message *won't* be passed; it doesn't store values. Gates can have many outputs; rather than using [x] you can send it a number, which refers to which output the input (from the right inlet) will go to.

11. The tutorials tell you about [slider]; prefer **[uslider]** instead, whose size is independent of its range, etc. Access range information by highlighting the [uslider] (the graphic thingy], and **typing command-i (for inspector)**. Other message or things that have inspectors include number boxes; you can set range, for instance.

12. **Now for MSP objects. These differ from Max objects because they are audio**. They are not data that are sent on demand, or according to a clock; if audio is on (by turning on [dac~] or [adc~]), audio is continually being passed around the patch at the audio rate, which is usually set to 44.1kHz. **[dac~ 1 2]** is the object that receives left and right channel audio, to be routed to 96 out 1&2. Don't use **[ezdac]**. Often Max/MSP objects need or can take special messages in their left inlets; for [dac~], [x], or 0 or 1, or start or stop, or startwindow, are required. There are no outlets to [dac~], since their output is through the soundcard (eventually). **0 or 1 (or [x] or lstopl or lstartl) stop and start all audio**. That means anything with a yellow squiggly patch cord will run, whether or not it is audio—some information, such as output by [line~], is audio rate, even though it is not (usually) sound. In other words, [line~] won't work if the

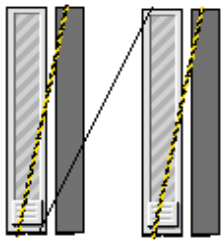
audio is off, even if you didn't need to hear anything. 0 or 1 etc. affect audio in all open windows (patches). If you open the help patch for [cycle~] and you have audio on, you will now hear sounds not coming from your patch, but from the help patch. lstartwindowl starts the audio only in that patch, but any 0 will stop all audio. There are a couple of buggy anomalies you will discover with all this.

13. **[noise~] is a white noise generator. [pink~] is a pink noise generator. [cycle~ *freq*] is a sine tone generator** (actually a cosine generator, so that at frequency=0, it is a constant 1). The left input is frequency, overriding the argument, and can be float or signal (we'll cover the difference next week, but you can experiment with it if you like, e.g. for FM synthesis, which we will also be doing next week). The right inlet controls phase.

14. **[sfplay~ 2]** is a stereo (hence the 2) soundfile player. Its left two outputs are left and right stereo channels, and the right outlet bangs when the soundfile is finished. The inlet takes several messages; **lopenl** is required to choose the soundfile—clicking on [open] will give you a dialog box; **[x] starts and stops the soundfile**, playing from the beginning; **lseekl will play part of the file**; **lloop 0l or lloop 1l** will not loop or will loop the soundfile (you can see how **$1** as an argument works buried deeply in the [sfplay~] help patch); [pause] and [resume] are also useful. Delve deeper into the help file for more tricks.

14. **[selector~ *arg*]** (or **[sel]**) takes *arg* number of audio inputs (the rightmost inlets), and the left inlet determines which inlet is sent to the outlet. 0 means nothing is passed.

15. **[meter~]** is a VU (level) meter. **[gain~] is a logarithmic signal (audio) fader**. These are well-used together. Unity gain is when the value of [gain~] (obtainable from the right outlet) is 127; up to 157 is amplification, and will distort a normal signal. Both these objects are examples of objects that turn into graphical symbols. You will also learn how to manipulate the size and shape of objects with these ones: hovering the mouse over the bottom right, but on the side, of most objects will reveal a tiny square, and the cursor will turn into arrows. You can chance the appearance of objects by dragging on the tiny square.



Courtesy of Cycling '74. Used with permission.

16. **[+~] adds audio signals**. This automatically happens at [dac~] inlets, in fact, at any audio input. **[*~] multiplies audio signals; if you multiply an audio signal by a float, it attenuates or amplifies the signal**. If you have three [cycle~]s heading for the same inlet of a [dac~], it is wise to use [*~ 0.33] to avoid clipping.