Granular Toolkit v1.0 for Cycling74's Max/MSP

Nathan Wolek
School of Music, Northwestern University
711 Elgin Rd., Evanston, IL 60208-1200, USA
Email: n-wolek@northwestern.edu

ABSTRACT

Since the generation of granular textures was first automated using a computer (Roads 1978), granular synthesis has grown to become a popular tool for creating new sounds in electro-acoustic music. Many effects can be achieved through the granulation of sampled sound including time compression (Jones and Parks 1988) and expansion (Truax 1990) independent of pitch alterations. Such effects can be created using Cycling74's Max/MSP software, allowing them to be utilized in real-time. However, the software does not include sufficient externals to meet the efficiency and flexibility needs for creating such effects. This paper details a collection of externals and abstractions for Max/MSP that the author has created with the aim of meeting these needs.

BACKGROUND

The basic concept behind granular synthesis can be traced back to Dennis Gabor, who made the assertion that "it is our most elementary experience that sound has a time pattern as well as a frequency pattern (Gabor 1947, 591)." Gabor developed a model of sound that linked these to two attributes in fundamental particles that came to be known as "grains." These short bursts of sound are only a few milliseconds in length with variable frequency content and combine with other grains to form larger sounds.

Research into the musical applications of Gabor's granular sound model was limited until computers were first used to automate the process of generating the hundreds of the short sounds per second that this model necessitates (Roads 1978). Since the first computer implementation of granular synthesis, interest has shifted toward sampling grains from a pre-recorded sound source, also known as granulation (Roads

1985; Truax 1987). These short bits of sound are then re-synthesized to create a desired effect that builds upon the original sound source. Effects such as time-contraction (Jones and Parks 1988) and time-expansion (Truax 1990) without any variation to the pitch of a source recording are two examples of granulation effects. Depending on the control parameter settings, a variety of new sounds can be created by granulating a single source sample.

Previous software for performing granular synthesis or processing has taken different forms, with each implementation having its own methods and level of control to offer the end user. Some applications have aimed to help users by generating the large amounts of data needed to realize granular sounds within some synthesis languages (Roads 1978; Helmuth 1991, 1993; Orton, Hunt and Kirk 1991). Others have developed applications that handle both the parameter and audio needs for generating granular sounds (Roads and Alexander 1995; Behles, Starke and Röbel 1998; Rolfe and Keller 1998). Because the concept of grains can be the basis for different effects, it is desirable to have software akin to a toolkit that would have both low-level operators and high-level effects for users to exploit however they see fit. This is the goal of the software described in this paper.

MAX/MSP

Cycling74's Max/MSP is "a high-level, graphical programming language (Cycling74 2001, 3)" that allows the user to easily put together signal networks and control structures for the production of audio, MIDI and multimedia projects, all of which can be controlled in real-time. The current version of the software runs on Apple PowerPC-based computers under versions 8.1 to 9.x of the Mac OS. It is an ideal environment to experiment with ideas because one can quickly connect (using virtual "patch cords") smaller blocks of code (known as "objects") together to form larger audio processing networks ("patches"). The environment offers a semi-open architecture in which anyone capable of programming in C can use the software developers' kit to develop custom objects ("externals") that can be freely distributed and used by others working with the software. In addition to the development of externals using the C programming language, Max/MSP users can encapsulate patches so that they can be re-used in other patches (most often referred to as "abstractions"). Once in this form, abstractions can be called in the same manner as objects or externals.

This level of customization makes Max/MSP an ideal environment for exploring granular concepts of sound and developing methods of employing them in processing effects. Patches that implement granular synthesis and sampling have been previously reported using an earlier incarnation of Max that ran on the IRCAM Signal Processing Workstation (Helmuth 1993; Lippe 1994; Todoroff 1995). Some externals for grain generation were also developed for this version of Max (Eckel, Rocha-Iturbide and Becker 1995). However, externals that deal with specifically with audio according to granular concepts are just beginning to appear for Cycling74's Max/MSP, including those detailed in this paper. These new granular externals should allow Max/MSP users to more easily create new and interesting patches built on the concepts of granular processing. In addition, they could be used to develop a series of new abstractions for producing basic granular effects more efficiently.

NEW GRANULAR EXTERNALS

This toolkit began with the idea that new externals were needed for the Max/MSP environment that worked at the granular level. These externals needed to manage the various control parameters, define each grain internally and ensure that parameter updates were deferred until the beginning of a new grain. The process of development was somewhat unscripted, with revelations in the development of one external leading to the concept for the next and techniques discovered in the development of another leading to revisions in externals that were previously thought to be complete. In the end, four externals were created to meet the different control needs that the various forms of granular processing require (figure 1). The name of each begins with the prefix of "grain" combined with the suffix that describes the control method each employs: 'grain.bang~', 'grain.pulse~', 'grain.stream~', and 'grain.phase~'.

The externals are setup to operate in a consistent manner with the same control parameters for each external. All four of the externals require two arguments in order to be used. The first argument names the 'buffer~' object holding the sound file to be sampled and the second names the 'buffer~' object holding the window shape to be used. The use of a buffer as the sampling source allows these granular objects to sample in a manner similar to other objects included in the Max/MSP distribution with which most users should be familiar (such as 'groove~' or 'wave~'). The second buffer gives users the flexibility to either use the window files included in the toolkit or to design their own. Either buffer can be changed from its initial setting by using messages. To change the sampling buffer, the user would send a message containing the text 'setSound' followed by the name of a buffer. Likewise, changing the window buffer involves sending a message with the text 'setWin' and a buffer name. If these messages are used while processing is enabled, the grain externals will defer sampling from the newly specified buffer until the current grain has completely sounded. This deferment prevents any sample discontinuities that could result from switching buffers in the middle of a grain.

Interpolation can be applied to either of the buffers being sampled in the form of an all-pass interpolation filter (Dattorro 1997). By default interpolation is turned on for sampling from the sound source buffer and off for sampling from the window buffer. These decisions on the default states were made based on the improvement of audio quality compared with the added processing load that interpolation on each buffer would have. Each can be turned on or off by using either the 'sndInterp' and 'winInterp' message with an argument of 1 or 0, denoting on or off respectively.

The leftmost inlet on each of the externals is dedicated to the control input and each has only one outlet that is used to output the granular signal. The remaining inlets are capable of receiving their data in the form of either Max messages or MSP signals. In either case, changes sent to these inlets are always deferred to the beginning of the next grain generated by the object. This ensures that their implementation is consistent with the concept of grains as being fundamental units with constant parameters throughout their short durations (Xenakis 1971). The second inlet on each is labeled "sound begin" and changes sent here determine the offset in milliseconds from the beginning of the sound buffer to begin sampling for the grain specified. Values that would be beyond the boundaries of the buffer's length are simply wrapped around in a manner similar to the modulo operator. For example, a value of 1100 milliseconds for sampling from a buffer that is only 1000 milliseconds long would result in sampling actually occurring at 100 milliseconds from the beginning of the buffer; a value of 2100 milliseconds would result in the same actual offset. The last inlet is described as the "grain pitch multiplier" value and affects the sampling increment (how many samples are skipped for each sample in the audio output) for the sound buffer. It is labeled in terms of pitch because of the aural effect that changes in sampling increment have upon output. A pitch multiplier of 1.0 causes the samples to sound at their normal recorded pitch. A multiplier of 2.0 effectively causes the samples to sound an octave higher, while a multiplier of 0.5 causes the sample to sound an octave lower. A floating-point value may be specified, making more minute pitch transpositions possible. Both 'grain.bang~' and 'grain.pulse~' require an additional specification of the grain length. This is necessary due to the control methods they employ, which will be explained.

The first external is named 'grain.bang~' and, as the name suggests, is controlled with the common Max control message of 'bang'. When a 'bang' is sent into it's leftmost inlet, a single grain is output from the signal outlet. While it is in the process of outputting that grain, all bangs that might be received by the control inlet are ignored. This protects from the unwanted effect of interrupting a grain that is in the processes of sounding. Once the grain has completely sounded, the external will once again begin monitoring the inlet for bangs. However, the use of bangs as a control signal means that grain onsets can only occur at the interrupts of the Max scheduler and never in between. One consequence of this is that the interval between successive grain onsets can never be smaller than the interrupt for the scheduler. For example, if the Max scheduler is set for an interval of 20 milliseconds, this means that grains can never be closer than 20 milliseconds apart or repeat at a frequency higher than 50 Hertz. This may not seem to be a problem upon first glance since grains are typically of a length on the order of 40 to 50 milliseconds, however there is an additional consequence of the granular frequency's dependence on the scheduler.

Dependence on the scheduler means that the time between successive grain onsets will always be a multiple of the scheduler interrupt. Because of this, stable frequencies can only be produced at those frequencies that are sub-harmonically related to the scheduler’s frequency. Frequencies between these sub-harmonics will have onset intervals that bounce unpredictably between the interrupt-related values which it lies. Barry Truax made note of this limitation in his own implementation of granular synthesis, referring to timings as being "quantized (Truax 1988, 18)." The sub-harmonics can be calculated by dividing the interrupt frequency by integers greater than one. Using the example of a 20 millisecond interrupt again would mean a maximum granular frequency of 50 Hertz and lower frequencies at 25, 16.6, 12.5, 10, etc. Lowering the scheduler interval helps to alleviate the problem, since a smaller interrupt increases the maximum possible frequency, but the sub-harmonic relationship persists. An interrupt interval of one millisecond would allow for a maximum granular frequency of 1000 Hertz and sub-harmonics of 500, 333.3, 250, 200, 166.6, etc. However, lowering the interval of the scheduler can work against the computational efficiency that the interval is meant to create. It became clear during development that, in order to alleviate this problem, a control signal that operates at the sampling rate is needed. This lead to the creation of the 'grain.pulse~' external.

The 'grain.pulse~' external monitors the signal connected to its leftmost inlet for zero-to-one transitions which it reacts to by outputting a grain. This type of control signal, in which all samples within a single period have a value of zero except for a single sample that has a value of one, is typically referred to as an pulse-train. In the Max/MSP environment, this signal can be generated by the 'train~' object. Since the control signal allows a resolution of grain intervals inversely related to the sampling rate, it allows the sub-harmonics to be much closer together at the lower end of the frequency spectrum. Because granular frequencies are in this range, the audio-rate control signal provides a much better resolution. The increase in computational load that an audio-rate control signal causes within Max/MSP was found to be minimal. Just as bangs are handled by its counterpart 'grain.bang~', the 'grain.pulse~' external will ignore any pulse received while a grain is being produced. Additional parameters are set through the other inlets just as they are with the other grain externals.

With both of the previous externals, a single control event triggers the output of one grain with a predetermined length. This one-to-one correlation between control and output means that in order to create a continuous stream of grains, the control signal is required to be repetitive. In order to eliminate this requirement, 'grain.stream~' external was created. This external requires that the user only specify the granular frequency in order to create a continuous stream of grains. With this external, grains are created consecutively at the specified frequency with no delay between them. Changes in frequency are applied only at the start of a new grain and because of this, the user may experience a delay between the initiation of a frequency change and its application to the output. This may seem undesirable at first, however it is a better fit with the concept of the grain as an indivisible unit (Xenakis 1971).

A single voice of continuous grains works very much like amplitude modulation applied to the sampled sound source, but it can be combined with other voices to fill out the sound and obtain a more even re-synthesis. However, the 'grain.stream~' object provides no manner for the synchronization of phases between multiple instances of itself. The multiple streams can move in and out of phase with one another in an uncontrollable manner. As a solution to this, the 'grain.phase~' external re-introduces the need for a control signal in order to maintain more precise phase relationships. The control input takes the form of a signal modulating from zero to one which controls the phase of the window applied to the sampled source. This type of output is generated by the standard Max/MSP object called 'phasor~', multiple instances of which can maintain constant phase relationships. It is important to note that the phase signal controls only the window and not the sampling increment for the sound buffer. The sampling increment is still computed in relation to the specified pitch multiplier and remains fixed for the duration of a single grain.

ADDITIONAL EXTERNALS

Two additional externals were developed in order to more easily manage multiple instances of the 'grain.pulse~' and 'grain.phase~' objects. When using several instances of either object, it is often desirable to have their control inputs be evenly out of phase with one another, which means individually setting the initial phase for multiple instances of either the 'train~' or 'phasor~' objects. This prompted the development of externals based on these objects that have multiple outlets evenly out of phase with one another. These externals are called 'train.shift~' and 'phasor.shift~'.

They each require a single integer argument to specify the number of outlets the object should have. Once instantiated, the outlets will produce outputs similar to the objects from which they derive their names, with ‘phasor.shift~’ generating phase ramps from 0 to 1 and ‘train.shift~’ producing pulse-trains. However, the individual outlets will have initial phases increasing from left to right with values equal to the outlet number minus one divided by the total number of outlets [init_phase = (out_num – 1) / num_of_outs]. For example, if the 'phasor.shift~' external is given an argument of 2, it will have two outlets with the first having an initial value of 0.0 and the second being 0.5 (figure 2); if given an argument of 3, three outlets will be produced with initial values of 0.0, 0.33, and 0.66; and so on up to a maximum argument of 32. In the case of ‘train.shift~’, these initial values would be multiplied by the period length in order to determine when the next pulse would be produced from that outlet. These externals greatly reduce clutter in patches where multiple voices of grains are needed. Although they have been developed for use with the granular externals described here, surely others will find additional uses for them.

EFFECT ABSTRACTIONS

In addition to the new externals that have been created for this toolkit, several abstractions were developed that provide ready-made granular effects built with the externals described above. These abstractions were built with a consistent interface style in order to make incorporating them into other patches as easy as possible for users. Most of the effects have two versions: one that operates on static sound files and a second that processes audio live as it streams into the abstraction. These two parallel methods of granular sound sampling are analogous to those previously implemented by Barry Truax for the DMX-1000 (Truax 1988). All of the effects operate in real-time with very reasonable loads on the current generation of G3 and G4 PowerPC processors.

Changes are made to the sound input through the leftmost inlet. For those effects working with a static buffer this inlet handles messages used to load sound files into that buffer. A bang message will bring up the appropriate dialogue window so that the user can select the file from the hard drive, while a message containing the name of a file itself will load that file into the buffer. If the user would like to use a 'buffer~' object that is within another patch, a message with the text 'setBuffer' followed by the identifying name associated with that instance of the 'buffer~' object will allow the abstraction to use it instead of the internal buffer.

For those abstractions with a streaming input buffer, the user simply needs to connect the signal that is to be processed to the leftmost input. Any signal may be used including one from a live audio input. The abstraction records the input using an internal looping buffer while the processing is being performed. These live sampling effects are able to achieve very low latency because the grain externals are programmed to read samples from the buffer in reverse. This technique was described by Barry Truax, who maintained that "at this time level there is no aural difference between forwards and reverse (Truax 1990, 105)". However, care must be taken with the current abstractions because this equivalence may be lost if the user specifies longer grains. In addition to the signal input, on the streaming abstractions the second inlet from the left allows the user to switch buffer recording on and off as needed using values of 1 or 0. A short, 10-millisecond fade is applied when switching on or off in order to prevent clicks due to signal discontinuity that this switching may cause.

All of the toolkit's abstractions have a common prefix of 'gran' to help the user identify them. Those with a static buffer are given the suffix of 'file', while those with a continuous streaming buffer are given a suffix of 'live'. These naming conventions should help the user to identify common effects, while separating the different forms of buffer sampling provided for each processing effect.

Each effect abstraction has several control parameters that can be changed in real-time to affect the processing output. By double-clicking on an instance of a specific abstraction, a window will be opened that can be used as a control interface for the effect. It uses standard Max/MSP interface items, such as number boxes and drop down menus, to provide the user with basic abilities to affect the processing output. Within the interface, effort has been made to place similar parameters next to each other and in common locations between the different abstraction interfaces. This should help the user quickly become familiar with the interface elements in a single abstraction and translate that knowledge easily to the other abstractions.

In addition to the provided interface, the abstractions offer a fairly simple method for creating custom interfaces and control mechanisms. The rightmost inlet and outlet are used to manage changes in the numerous control parameters through messages exchanged with the parent patch. All changes to the control parameters are output from the outlet in a consistent message format with the name of the parameter followed by its new value. For example, if a message were output reading "pitch_mult 1.2" would mean that the pitch multiplier had been changed to a value of 1.2. Similar messages can also be sent to the corresponding inlet in order to change the parameter setting. As with the previous example, if a message were sent reading "pitch_mult 1.0" to the rightmost inlet it would change the value for the pitch multiplier to 1.0. Using the 'route' and 'prepend' objects from the standard Max distribution, it is quite simple to create a custom interface that communicates with the abstraction (figure 3). It is important to note that changes sent to the rightmost inlet are also echoed to the rightmost outlet. Because of this, the user must protect against infinite loops when creating custom interfaces.

Any additional outlet found on a given abstraction is used to output a granular signal for either playback or further processing. Most of the abstractions produce only a single, mono signal. All abstractions accept only mono signals as input, either as single connection for the streaming buffer effects or a mono sound file for the static buffer effects. A few of the effect abstractions produce stereo output and one type produces eight individual streams at varied phases. In order to help manage processor load, all signal outlets from these abstractions have a 'pass~' object placed before their last connection. Those familiar with Max/MSP will know that this allows the abstractions to work with a standard 'mute~' object in halting their processing without producing unwanted noise in the audio output.

The names of these abstractions and a brief description of their processing effect follows. Included in the descriptions are any references on the specific type of granular processing used and the processor load of the abstraction. Processor utilization was monitored using the DSP status window in Max/MSP running on an iBook G3 clocked at 500 MHz. A change in the control parameters can affect the amount of processing the abstraction uses and so the figures stated here are only approximates using mostly default settings.

gran.ASstream.file~ & *gran.ASstream.live~* (5-6% CPU utilization) – These are perhaps the simplest of the abstractions. Using their respective form of sampling input, both produce a single stream of grains based on the concept of asynchronous granular synthesis (Roads 1991). These abstractions have very little processor load but also have minimal control parameters affecting the sound.

gran.pitch.file~ & *gran.pitch.live~* (16-17% CPU utilization) – The pitch abstractions use eight voices of granular processing with evenly spaced phases to maintain an even re-synthesis of the sampled sound. Using a single pitch multiplier, the user can transpose the pitch height of an input, which in the case of live sources could be used to produce harmonies.

gran.chord.file~ & *gran.chord.live~* (16-18% CPU utilization) – Similar construction to the above "pitch" abstractions with the added ability to specify a list of pitch multipliers. Individual grains are then transposed to selected intervals within the given list. The end result sounds like a chord based on the list of multipliers.

gran.groove.file~ (15-16% CPU utilization) – This abstraction is meant to mimic the behavior of the 'groove~' object which is included with Max/MSP. However, since this abstraction uses granular processing to loop the sound file, the speed and pitch can be manipulated independently. This effect does not have a streaming buffer version because its time compression and expansion capabilities require that the input sample be static. It uses the same phase-related grain streams that were previously described in order to create an even re-synthesis.

gran.play.file~ (14-15% CPU utilization) – This abstraction mimics the behavior of the 'play~' object within Max/MSP. The middle inlet of this abstraction provides the ability to update the sample offset at the signal rate and can therefore be used for a variety of looping effects. This effect also does not have a streaming buffer version for the same reasons as 'gran.groove.file~'.

gran.space2.file~ & *gran.space2.live~* (19-20% CPU utilization) – Based on the "pitch" effect, this abstraction places the individual streams at specific panning locations across the stereo spectrum. These streams are numbered one to eight from left to right and can be turned on or off in order to create different spatial effects.

gran.space8.file~ & *gran.space8.live~* (18-19% CPU utilization) – The same concept as the above "space2" effect with the difference of individual streams output from different outlets. This gives the user the ability to process the streams separately or distribute them within a multi-channel audio output system.

gran.cloud.file~ & *gran.cloud.live~* (19-20% CPU utilization) – These use the model of asynchronous granular synthesis with an element of random deviation from basic control parameters to produce "cloud" textures (Roads 1991). These abstractions provides more parameters than the others described in this paper and are far more processor intensive at higher granular frequencies.

ABSTRACTION CONTROL PARAMETERS

Many of the effect abstractions have common names for parameters that are the same or operate in a highly similar fashion. Rather than explain them in the context of the individual abstractions, the following section provides a common reference for the different parameter names and a brief explanation of what each controls. The shortened names are the labels used in the control interface displayed when the user double-clicks on a given abstraction instance. These are also the names used by the messages handled by the rightmost inlet and outlet for custom control of the parameters.

samp_offset – Sample offset. In abstractions that use a static buffer, this parameter is used to define the number of milliseconds from the beginning of the buffer to begin sampling grains. It is not available in the streaming buffer abstractions because the sampling position always follows the recording position within the buffer.

buf_size – Buffer size. Used by streaming buffer abstractions in order to define the length of the buffer used for sampling the input signal in number of milliseconds.

speed – Only available in the "gran.groove.file~" abstraction. Used to control the speed of playback for the sampled sound upon re-synthesis. For example, a value of 1.0 is equivalent to normal playback speed, 0.5 equals half the normal speed, and 2.0 twice the normal speed.

offset_bw – Offset bandwidth. Defines the number of milliseconds that the sample offset may randomly deviate from itself. The bandwidth represents the total possible deviation with both the positive and negative direction combined. The maximum deviation possible in either single direction is one half of the value specified.

freq – Frequency. For some abstractions, specifies the number of grain onsets per second in Hertz or roughly the inverse of the time interval between grain onsets expressed in seconds. Frequency is used in addition to grain length because the two are not always inversely related. The length can sometimes vary independently from the onset frequency, as is the case in asynchronous granular synthesis.

freq_bw – Frequency bandwidth. Defines as a percentage the maximum amount of random deviation allowed for the frequency parameter. The percentage is used to derive the actual deviation in Hertz using the frequency parameter value. The use of a percentage allows the bandwidth to be perceived consistently across the frequency spectrum. Again the bandwidth is the total deviation possible with both the positive and negative directions combined.

length – The length of grains to be produced in milliseconds. Lengths for granular processing are typically limited to a range of 10 to 100 milliseconds and are most often in the 40 to 50 millisecond range.

length_max – Maximum length. Expressed as a percentage of the inverse of the frequency setting, this parameter allows the grains to overlap one another producing a thicker texture. At 100 percent, no overlap should occur because the length would equal the inverse frequency.

length_min – Minimum length. Another percentage of the frequency setting's inverse that allows the grains to be shortened, creating short delays between grains. The actual length of individual grains fluctuates randomly between the maximum and minimum length.

winShape – Grain windowing function. Allows the user to choose from a selection of common windowing functions to be applied to the samples retrieved from the buffer. Different shapes have different spectral and amplitude characteristics that can change the sound of the granular output.

pitch_mult – Pitch multiplier. Used to raise or lower the perceived pitch of the grains. It is actually used to control the sampling increment of the sound buffer, but the aural result is in the form of a pitch manipulation. For most samples, a setting of 1.0 would be normal playback, with 2.0 sounding an octave higher and 0.5 an octave lower.

pitch_bw – Pitch multiplier bandwidth. Provides the bandwidth for random deviation from the specified pitch multiplier and is expressed as a percentage of that value. Again, the use of a percentage gives the bandwidth a consistent aural width across the frequency spectrum.

stream_config – Stream configuration. Used in the "space" effects, this parameter allows the user to turn the individual streams of granular sampling on or off.

The related message has a somewhat different format from those for other parameters. It must contain the parameter name followed by the stream number and a 1 or 0 to change the stream's state. For example, a message reading "stream_config 4 0" would turn stream four off, while "stream_config 4 1" would turn it on again.

gain – Provides a basic gain control over the granular output. The value is expressed in decibels and is then converted into a multiplier to scale the amplitude of the grains. For example, a value of 0.0 dB would multiply the output by 1.0 while a value of –76.0 dB would multiply the signal by 0.0.

gain_bw – Gain bandwidth. Controls the bandwidth of the random deviation from the specified gain parameter, allowing the gain to vary between individual grains. The value is expressed in decibels.

pan_min – Minimum panning position. Used only in the "cloud" abstractions, it allows individual grains to be randomly panned across the stereo spectrum. This control gives the user the ability to limit the width of the random panning. The value is expressed as a floating point number from 0.0 to 1.0, with 0.0 representing hard left and 1.0 representing hard right.

pan_max – Maximum panning position. Used in connection with "pan_min" in order to limit the area of the stereo spectrum in which the random panning can occur.

EXTRAS

The toolkit includes help files for each of the externals and abstractions so that users can reference them just as they would objects included with their standard Max/MSP installation. There are also a few extra items intended to aid the user. A number of envelopes are included in the 'winShape' menu of the granular abstractions, which should give the user ample selection in choosing an envelope for the grains. The shapes include Gaussian, quasi-Gaussian, triangle, three-stage linear, Blackman, Blackman-Harris, Hamming, Hanning, exponential decay and reversed exponential decay.

These shapes and the formulas used to generate them were drawn from several sources (Roads 1996, 2001; Oppenheim and Schafer 1999). All of the windows used in the effect abstractions are 512-sample sound files that were created within Max/MSP itself. In order to demonstrate how this was accomplished for those who may wish to create their own windows, a patch called 'gtk.winMaker' is included with the toolkit.

In addition to the patch demonstrating window construction, a patch called 'gtk.objectGuide' is included. This patch contains all of the externals and abstractions that are included with the toolkit. It is meant to serve as a quick reference to the additional tools that are available once the user has added the toolkit to his or her installation of Max/MSP.

CONCLUSION

The externals and abstractions in this granular toolkit should appeal to both the novice and expert user of granular processing. Novices will find useful effects built to be easily incorporated into their own work. Experts will find useful low-level tools that allow them to freely experiment with the possibilities of granular processing at various levels of precision. Both should be pleased with the efficiency and flexibility that the granular toolkit offers to all.

This toolkit is freely available for download from the author's web site at the following address: "http://www.nathanwolek.com" Follow the menu link to the software section in order to download a compressed archive of the files. The user must follow the included instructions on where to place the files on his or her hard drive. A working copy of Cycling74's Max/MSP must be present in order to use the toolkit.

ACKNOWLEDGEMENTS

This toolkit began during the development stages of several projects in which I participated as a programmer. They include Amnon Wolman's *Thomas and Beulah*, Paul

Hertz's *Fool's Paradise* and the Northwestern Center for Art and Technology's *Portal Project*. I would like to sincerely thank the artists involved in those projects for pushing me to develop the ideas that evolved into this toolkit. Additional support this research was provided by a grant from the Northwestern University Graduate School.

Figure 1.

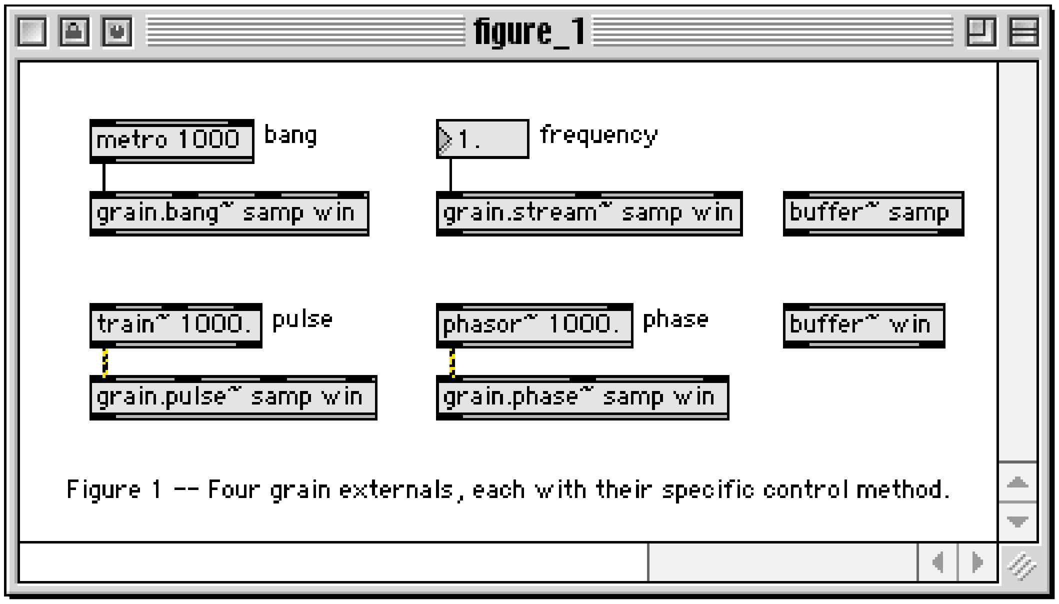

Figure 1 -- Four grain externals, each with their specific control method.

Figure 2.

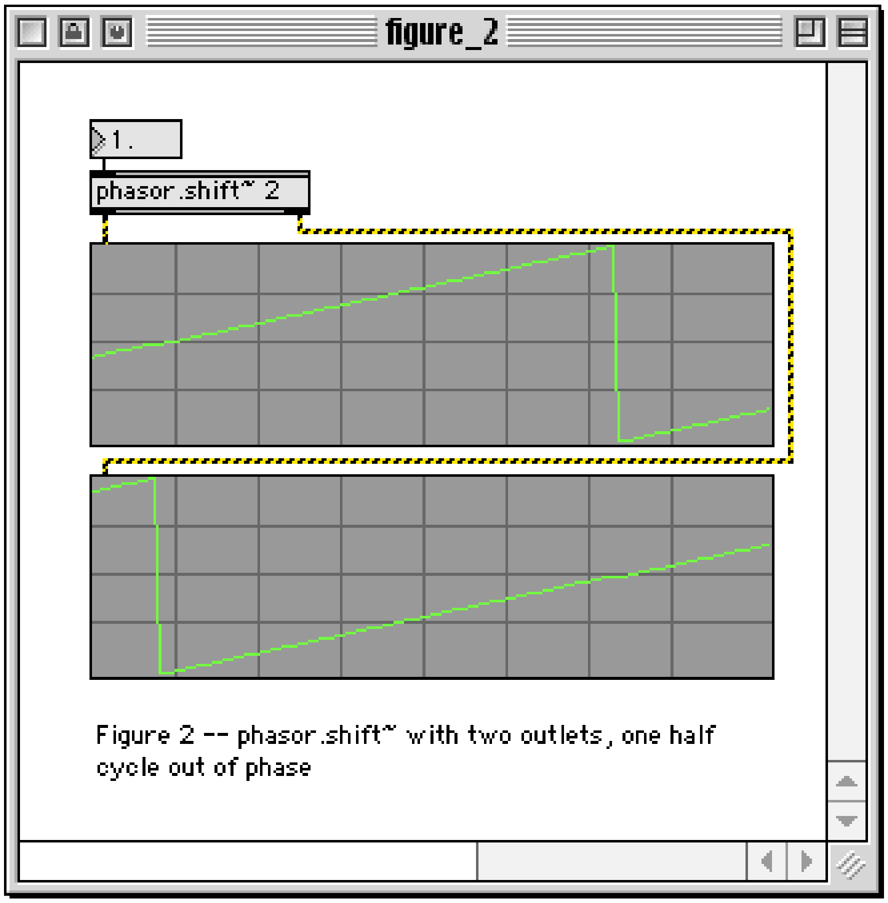

Figure 2 -- phasor.shift~ with two outlets, one half cycle out of phase

Figure 3.

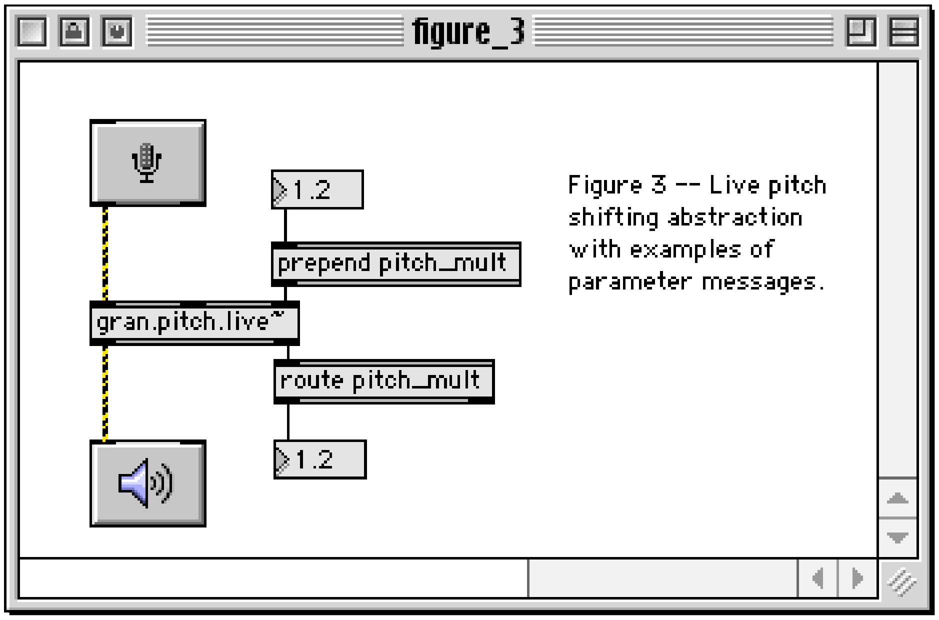

Figure 3 -- Live pitch shifting abstraction with examples of parameter messages.

REFERENCES

Behles, G., S. Starke and A. Röbel. 1998. "Quasi-Synchronous and Pitch-Synchronous Granular Sound Processing with Stampede II." *Computer Music Journal* 22 (2): 44-51.

Cycling74. 2001. *Max 4: Getting Started*. Online documentation. Internet: http://www.synthesisters.com/download/Max4GettingStarted.pdf, 1 March 2002.

Dattorro, Jon. 1997. "Effect Design, Part 2: Delay-Line Modulation and Chorus." *Journal of the Audio Engineering Society* 45 (10): 764-788.

Eckel, G., M. Rocha-Iturbide and B. Becker. 1995. "The development of GiST, a Granular Synthesis Toolkit Based on an Extension of the FOF Generator." In R. Bidlack, ed. *Proceedings of the 1995 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 296-302.

Gabor, Dennis. 1947. "Acoustical Quanta and the Theory of Hearing." *Nature* 159 (4044): 591-594.

Helmuth, Mara. 1991. "Patchmix and StochGran: Two Graphical Interfaces." In B. Alphonce and B. Pennycook, eds. *Proceedings of the 1991 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 563-566.

Helmuth, Mara. 1993. "Granular Synthesis with Cmix and MAX." In S. Ohteru, ed. *Proceedings of the 1993 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 459-452.

Jones, Douglas L. and Parks, Thomas W. 1988. "Generation and Combination of Grains of Sound." *Computer Music Journal* 12 (2): 27-33.

Lippe, Cort. 1994. "Real-Time Granular Sampling Using the IRCAM Signal Processing Workstation." *Contemporary Music Review* 10 (2): 149-155.

Oppenheim, Alan V. and Schafer, Ronald W. 1999. *Discrete-Time Signal Processing.* 2nd edition. Upper Saddle River, New Jersey: Prentice Hall.

Orton, R., A. Hunt and R. Kirk. 1991. "Graphical Control of Granular Synthesis using Cellular Automata and the Freehand Program." In B. Alphonce and B. Pennycook, eds. *Proceedings of the 1991 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 416-418.

Roads, Curtis. 1978. "Automated Granular Synthesis of Sound." *Computer Music Journal* 2 (2): 61-62.

Roads, Curtis. 1985. "Granular Synthesis of Sound." In C. Roads and J. Strawn, eds. 1985. *Foundations of Computer Music.* Cambridge: MIT Press, pp. 146-159.

Roads, Curtis. 1991. "Asynchronous granular synthesis." In G. DePoli, A, Piccialli, and C. Roads, eds. *Representations of Musical Signals*. Cambridge: MIT Press, pp. 143-186.

Roads, Curtis and Alexander, John. 1995. *Cloud Generator.* Computer software. Internet: ftp://ftp.create.ucsb.edu/pub/CloudGenerator/cg.nofpu.hqx, 1 September 2002.

Roads, Curtis. 1996. *The Computer Music Tutorial.* Cambridge: MIT Press.

Roads, Curtis. 2001. *Microsound.* Cambridge: MIT Press.

Rolfe, Chris and Keller, Damian. 1998. *MacPod.* Computer software. Internet: http://www.thirdmonk.com/Download/MacPod13.hqx, 28 August 2002.

Todoroff, Todor. 1995. “Real-Time Granular Morphing and Spatialisation of Sounds with Gestual Control within Max/FTS.” In R. Bidlack, ed. *Proceedings of the 1995 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 315-318.

Truax, Barry. 1987. “Real-Time Granulation of Sampled Sound with the DMX-1000.” In S. Tipei and J. Beauchamp, eds. *Proceedings of the 1987 International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 138-145.

Truax, Barry. 1988. "Real-Time Granular Synthesis with a Digital Signal Processing Computer." *Computer Music Journal* 12 (2): 14-26.

Truax, Barry. 1990. "Time-Shifting of Sampled Sound with a Real-Time Granulation Technique." In *Proceedings of the 1990 International Computer Music Conference.* San Francisco: International Computer Music Association, pp. 104-107.

Xenakis, Iannis. 1971. *Formalized Music*. Bloomington: Indiana University Press.