# Using Pure Data for Real-Time Granular Synthesis Control Through Leap Motion

Damián Anache(✉)

CONICET, Consejo Nacional de Investigaciones, Científicas y Técnicas,
UNQ, Universidad Nacional de Quilmes, Roque Saenz Peña 352,
B1876BXD Bernal Buenos Aires, Argentina
damian.anache@unq.edu.ar

**Abstract.** This paper documents the development of a granular synthesis instrument programmed on PD (*Pure Data*, Miller Puckette et al. (http://www.puredata.info) for being controlled by *Leap Motion* (https://www.leapmotion.com/), a computer hardware sensor that converts hands and fingers motion into simple data, ready for end-user software inputs. The instrument named *mGIL* (my_Grainer's Instrument for Leap) uses the *my_grainer* (Developed by Pablo Di Liscia, available at: puredata.info/Members/pdiliscia/grainer.) PD's external object as its GS (granular synthesis) engine and *leapmotion* (leapmotion external developed by Chikashi Miyama. Linux version available at: http://musa.poperbu.net/index.php/tecnologia-seccions-30/-puredata-seccions-45/129-installing-leapmotion-puredata-external-on-linux. Windows version at: http://jakubvaltar.blogspot.com.ar/2013/10/leap-motion-pure-data-external-for.html) external object as its local interface. This connection between software and hardware intends to reach expressive computer music sounds, with the performer's body imprints. The present status of that interplay (software + hardware) is a work-in-progress advance of the author doctoral thesis.

**Keywords:** Real-time synthesis · Pure Data · Leap Motion · Granular synthesis · Performance

## 1 Introduction

After Isaac Beekman [1], Dennis Gabor [2], Iannis Xenakis [3], Barry Truax [4] and Curtis Roads [1] himself, GS (granular synthesis) is fully documented by Roads on his book *Microsound* [1]. Nevertheless computer technologies are constantly improving and so are new approaches to this synthesis technique as *mGIL* (*my_Grainer's Instrument for Leap*) is an example of this nowadays. This instrument, *mGIL*, was developed on *Pure Data* to handle real time GS controlled by *LeapMotion*. In this development, GS focuses the timbral organization level in order to generate individual sound objects, instead of granular clouds. The aim is to create a group of grains as a sonic entity where the individual grains are integrated in a unique sound unit. Each grain has a meaning only inside that object. This can be described as a *short time granular sound* generated between a *packed* and a *covered fill* factor as Roads describe it[1].

[1] See [1], page 105.


M. Aramaki et al. (Eds.): CMMR 2016, LNCS 10525, pp. 171–179, 2017.
DOI: 10.1007/978-3-319-67738-5_10

In the context of the author's doctoral thesis, the main motivation of using granular synthesis on this development was its capacity to allow a huge amount of parameters modifications in a short time scale, with a manifest spectral consequence. This aspect is of special interest in order to reach expressive sounds generated by synthesis.

The author's research is centered on the incidence of the interpreter/performer in computer music generated by synthesis means only. A first approach to the problem [5] suggests that granular synthesis could be an appropriate technique in order to imprint the performer bodily trace actions on every sound generated by a digital instrument. The reason of this is the huge amount of control data that it involves, which could exceed the thousand parameters in just one second[2].

According to the author's standpoint, if this feature is properly combined with the right device control, highly expressive synthesis sounds may be produced. Therefore, the lack of physical imprint[3] in computer music may be overcome. At this stage of research, the chosen device was *Leap Motion*. This device was considered an appropriate choice for the aim of this work, because it can acquire the position values for each finger of each hand (at least 36 outputs) with great accuracy at a user defined rate (among other useful data from movements and gesture analysis).

## 2 Framework

Many of the most important developments for GS can be found on Roads [1], from where the highlights are *CG – Cloud Generator* (by C. Roads), and the early real time developments by Barry Truax. On the other hand, if we focus on developments for the free and open source platform *Pure Data*, we find five synthesis units available as external objects. Some of the features of the mentioned externals is discussed in the next summary (Table 1) as well as in the following descriptions.

**Table 1.** Pure Data's externals for GS.

| Name | Developer | Release | Output | G.Wf.[a] | G.Env.[b] |
|---|---|---|---|---|---|
| syncgrain[c] | Barknecht, F. | n.d. | Mono | Table | Fix |
| mill[d] | Keskinen, O. | n.d. | Stereo | Table | Hann |
| disis_munger1[e] | Ji-Sun, K., et al. | 2007 | Multichannel | Audio | Fix |
| granule[f] | Lyon, E. | 2012 | Stereo | Table | Table |
| my_grainer[g] | Di Liscia, O. P. | 2012 | Ambisonics | Tables | Tables |

a-G.Wf. = Grain Waveform.
b-G.Env. = Grain Envelope.
c-https://puredata.info/Members/fbar (Last access: 02/2016).
d-http://rickygraham.net/?p=130474333 (Last access: 02/2016).
e-http://l2ork.music.vt.edu/main/make-your-own-l2ork/software/ (Last access: 02/2016).
f-http://www.somasa.qub.ac.uk/∼elyon/LyonSoftware/Pd/ (Last access: 02/2016).
g-https://puredata.info/author/pdiliscia (Last access: 03/2016).

[2] See [1], page 87.

[3] This lack is marked by Anache [5] and is present on several authors of [6].

All of these objects implement the GS technique in a different way, offering advantages and disadvantages, different control parameters, possibilities and limitations. *syncgrain~* works only with synchronous GS and is a direct port of the *SndObject SyncGrain* by Victor Lazzarini. The grain waveform is obtained by reading a function table, meanwhile the grain envelope cannot be defined by the user. For *mill~* the grain waveform is also defined by user through a function table but the grain envelope is based on a *hanning* window with expanded sides possibilities. *disis_munger1~* is based on Dan Trueman's *munger~* (Computer Music Center, Columbia University). It doesn't generate grains by its own and needs an external audio input for working. Its grains envelope function is fix and the user can only change its duration. The output offers up to 64 intensity panning channels. *granule~* was developed at Department of Music and Sonic Arts Queen's University Belfast and it's included on the *LyonPotpourri* collection of externals together with other GS external object: *granulesf~*. Both, the grain waveform and envelope are defined by function tables but it only offers stereo output. Finally, *my_grainer~*'s latest version was released on 03/2016 and it can work with up to 24 different function tables at the same time for both the grain waveform and the grain envelope. Moreover it outputs an Ambisonics B-format signal (making 3D sound with full control feasible) and offers very detailed parameters in order to control the GS technique as its main reference literature explains: grain duration; gap between grains; pitch; amplitude level; spatial position; and auxiliary output level for external processing of each grain (like reverberation send.) The external also offers controls for a random deviation of each one of its parameters, different looping capacities for the audio tables read and may receive lists for specific random values choices.

## 3 Instrument Description

The instrument started as an improved version of Esteban Calcagno's patch called *Grainer_Dinamics*[4] but during the improvement process the patch achieved its own identity giving birth to *mGIL*. This new instrument keeps the original idea of being a programmable function-based controller for my_grainer~ with the addition of being triggered and further controlled through an external device. Nowadays the instrument is specially designed to operate with *LeapMotion* but it could be easily adapted to be used with other devices as well. Figure 1 below shows the instrument's main interface.

As explained before in Sect. 1, *mGIL* generates short time scale granular sounds, so its control function values are arbitrary limited in order to achieve this special kind of GS. For example, grains duration values are limited to 1–100 ms and *gap* (elapsed time between consecutive grains) values are limited to 1–500 ms. Also, as the synthesis engine allows it, *mGIL*'s parameter names are strongly related to Roads terminology [1], so it is designed for users who know the theory in advance. The user must handle carefully this interface, because some especial configurations may lead to undesired results. For example, a *gap* time smaller than the grain duration may produce overlapped grains, and therefore the audio output may be overloaded.

[4] Published on [8].

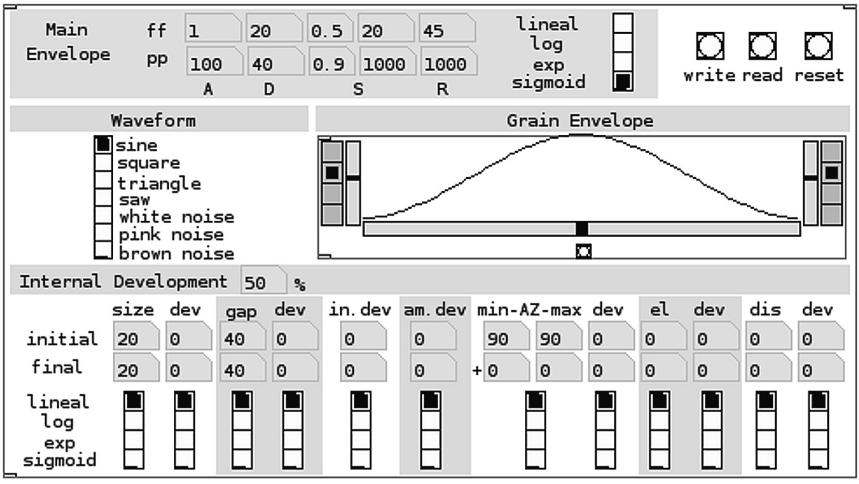


**Fig. 1.** *mGIL*'s GUI.

Each control function takes a starting value and an ending value, and a choice for the interpolation method to perform between these values. The interpolation methods must be chosen from the following options: linear, logarithmic, exponential and *S-shaped* sigmoid function.

The grain waveform is chosen from seven presets, divided in two groups: functions and noises; functions: sine, triangle, square and saw; noises: white, pink and brown. When using the functions waveforms, *mGIL* receives pitch values for transposing the original waveform, meanwhile noises ignores that input value (pitch control is explained on Sect. 3.1). The grain envelope is defined by a Bezier-based GUI, consisting of a sequence of two variable curves with an adjustable joint point. This offers many options for regular and custom envelopes, among them: bell-shaped, triangular, *expodec*[5], *rexpodec*[6], or any other two segment type shape.

Finally, the audio generated by the GS engine is controlled by the main ADSR[7] audio envelope, also defined by keyboard input values for a more precise control. This is different than the main envelope of most digital instruments, because in this case there must be two ADSR defined envelopes. One is for the higher intensity level and the other for the lower (normalized from 0 to 1, where 0 is ***pp*** and 1 is ***ff***). So, the intermediate intensity values are generated by interpolation of the ones of the defined envelopes, according to four interpolation types: lineal, logarithmic, exponential and *S-shaped* sigmoid function. The main ADSR envelope also defines the total sound's duration. Because of this, the lengths of the transitions of all the *mGIL*'s control functions are defined proportionally to the length of the ADSR envelope (by *Internal Development* parameter on GUI).

[5] Exponentially decaying envelope, see [1], page 88.

[6] Reverse *expodec*, see [1], page 88.

[7] Acronym for *Attack, Decay, Sustain, Release*.

As shown in Fig. 2, in order to generate two outputs, *mGIL* needs two input values: pitch and intensity. The outputs are: one control-rate data package to control the GS of *my_grainer~* and an audio-rate output to control the main amplitude envelope of *my_grainer~* (the main ADSR envelope).

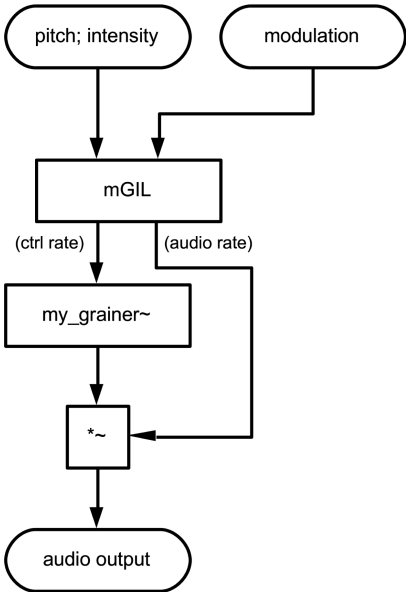


**Fig. 2.** *mGIL*'s connections scheme.

The remaining feature of *mGIL* is the modulation control. At this stage it just receives a value from 0 to 1 for scaling both the gap time and the bandwidth of the pitch variation (the later will only be taking in account if the defined GS configuration uses a random deviation of the main pitch). This will be the first feature to be improved in future *mGIL*'s versions, by adding more modulation inputs in order to achieve a more significant spectral influence.

### 3.1 Pitch and Dynamics

The design of *mGIL* offers two ways of pitch control: (A) specific frequency input (in hertz) with the option of normalized mode (from 0 to 1 for 20 Hz to 20 kHz); and (B) Bark Scale index input. For this last feature, *barkScale* was developed as a *PD*'s abstraction. It just allocates a table that receives a Bark Scale index as input and offers central frequency, bandwidth, low and high limits as outputs values. The input can also be normalized from 0 to 1 float values. On both cases, the received frequency values define the amount of transposition of the chosen grain waveform. In case that the chosen waveform is any kind of noise, the whole frequency input become meaningless and has no influence on the grain waveform.

In order to offer a balanced level audio output based on psychoacoustic data, an abstraction (*iso2262003*) was created based on the ISO 226:2003 international standard which documents the normal equal-loudness-level contours. The abstraction receives a frequency value (in hertz) and a loudness level contour (from 20 to 80 phon or normalize from 0 to 1) to compute the corresponding sound pressure level (from 0 to 120 dB SPL, or normalized from 0 to 1). It is important to notice that, as the digital domain cannot be related to specific psychoacoustic values, *mGIL*'s design assigned its internal values just to keep a relative balance of intensity across the spectrum, regardless of a strict implementation of the standard.

### 3.2 External Control

*LeapMotion*'s operation is detailed documented by its developers team at its official website[8] and as anticipated on Abstract, it runs on Pure Data platform thanks to an external object developed by *Chikashi Miyama*.

The first analysis stage of this project detects active hands on each half of the interaction zone, left and right. Then, the following analysis stage is the detection of the closing/opening hands gesture inside each of these zones, done by comparison of the *sphereRadius*'[9] method output data. As shown in the following images (Figs. 3 and 4).

This gesture detection sends data for *mGIL*, where pitch and dynamics are determined by hands' elevation positions and open gesture's velocity, respectively. The hand elevation position is used to define the sound's pitch, and the velocity of the open-close gesture defines the sound level (*dynamics*). At the same time, the interaction zone is divided into two regions, one for each hand. So, the whole analysis scheme offers data outputs from each one of the two hands and is connected to two *mGIL* instances. Thanks to this design, two independent streams sound synthesis can be performed simultaneously (Figs. 5 and 6).

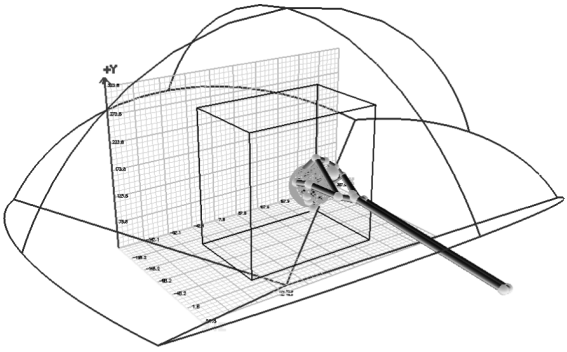

**Fig. 3.** Close hand gesture detection by *sphereradius* method.

[8] https://developer.leapmotion.com/documentation/ .

[9] See Leap Motion documentation.

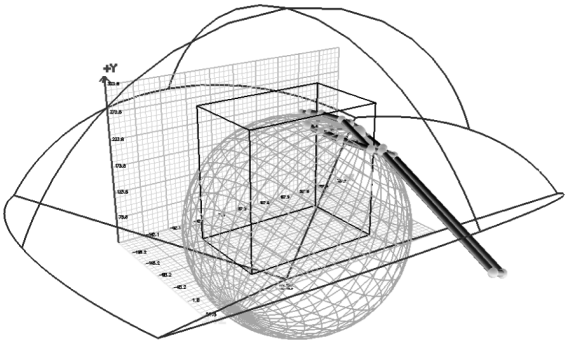

**Fig. 4.** Open hand gesture detection by *sphereradius* method.

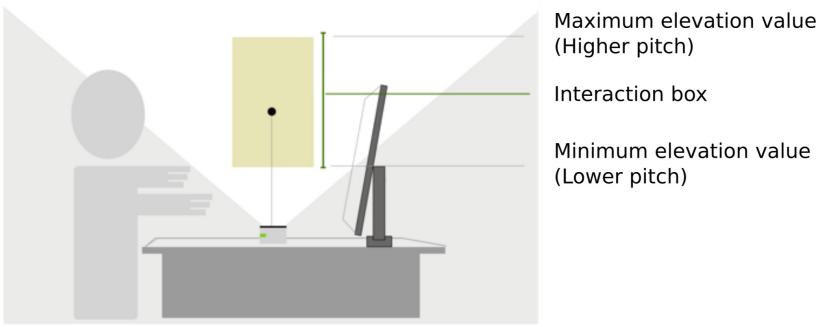


**Fig. 5.** Side view of the *interaction box,* and parameters' assignments for *mGIL.*

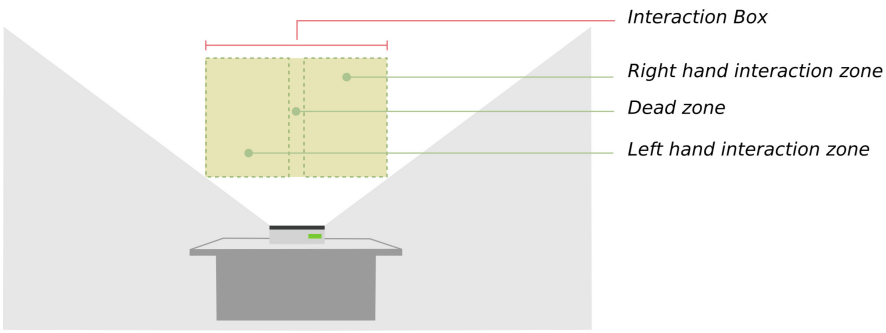


**Fig. 6.** Front view of the *interaction box,* and redefined interaction zones for *mGIL.*

### 3.3 Package Description

*mGIL* consists of a set of PD's patches and abstractions, some of them are flexible enough for being used in other instruments or even in general purpose projects. They all are listed below (Table 2).

**Table 2.** mGIL's patches and abstractions.

| Name | Description |
|---|---|
| autopack2; autopack3 | Packs two or three floats numbers in one list regardless of the order of receiving values |
| barkScale | Offers Bark Scale's values on demand, explained on Sect. 3.1 |
| ISO2262003 | Offers ISO 226:2003 data on demand, explained on Sect. 3.1 |
| leapmotion-ctrl | Custom Leap Motion analysis for *mGIL*, needs *leapmotion* external object, explained on Sect. 3.2 |
| mGILabstraction | *mGIL*'s core, GUI shown on Fig. 1 |
| mGILmainPatch | Main patch |
| mGILmy_grainerParameters | GUI for *my_grainer*~'s parameters |
| OnebangTime | Bangs redundancy filter |

## 4 Conclusions, Final Observations and Future Development

This first version of *mGIL* only works with audio functions and noisy grain waveforms, leaving behind any other kind of audio signals like acoustic sources recordings. It was designed this way in order to completely avoid the generation of sounds that may resemble the ones produced by acoustic sources. However, conscientiously explorations with acoustic sources recordings will be tested on next instrument's versions. The control of the *my_grainer* external 3D spatialisation capacities by performing gestures is also one of the areas to be further explored. For this improvement, the aim is for the relative position of each hand to move the sound around the 360° of the Ambisonics surround sphere, as shown in the following figure. Each hand would control the location of the sound generated by its own, within the corresponding right-left zone (Fig. 7).

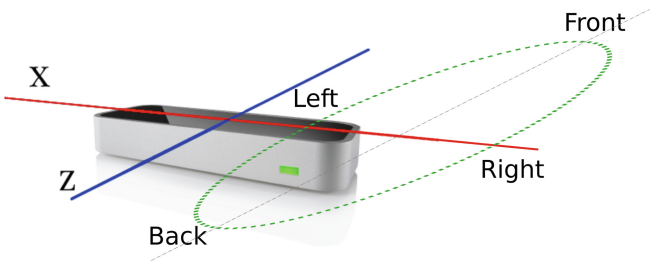


**Fig. 7.** Spatial control design for future versions.

This work also involved several suggestions of new features to Pablo Di Liscia (*my_grainer*'s developer). Some of them (for example, different looping capacities for grain waveform tables reading) are available in its last release (March, 2016) and some others maybe available on next versions.

One of the most important subject which was researched through this development is the influence of corporal gestures into the morphology of the generated sound in live performance. The actual state of this project allows the author to start his first studio compositions, performances and analysis in order to developed his doctoral thesis. At the same time, the source code, documentation and examples are freely shared on-line, so as other artists and programmers can be able to explore this research area.

## References

1. Roads, C.: Microsound. The MIT Press, England (2004)
2. Gabor, D.: Acoustical Quanta and the Theory of Hearing. Nature **159**(4044), 591–594 (1947)
3. Xenakis, I.: Formalized Music. Pendragon Press, Columbia (1992)
4. Truax, B.: Real-time granular synthesis with the DMX-1000. In: Berg, P. (ed.) Proceedings of the International Computer Music Conference, The Hague, Computer Music Association (1986)
5. Anache, D.: El Rol del Intérprete en la Música Electrónica - Estado de la Cuestión. In: Actas de la Décima Semana de la Música y la Musicología, UCA, Argentina (2013)
6. Peters, D., Eckel, G., Dorschel, A.: Bodily Expression in Electronic Music – Perspectives on Reclaiming Performativity. Routledge, Abingdon (2012)
7. Paine, G.: Gesture and morphology in laptop music. In: Dean, R.T. (ed.) The Oxford Handbook of Computer Music. Oxford University Press, USA (2009)
8. Di Liscia, O.P. (ed): Síntesis Espacial de Sonido, CMMAS Centro Mexicano para la Música y las Artes Sonoras, Mexico, ebook with aditional files (2016)