Developing Cabbage Plugins For Composition and Sound Design and Sharing Them Online

Caio M. Jiacomini[1] *

Berklee College of Music
caiojmini@gmail.com

Abstract. In 2020 I discovered Csound. Using the *Csound FLOSS Manual[1]*, I started teaching myself about this amazing program. With Cabbage[2], I realized I could make my own plugins and distribute them on the internet. My goal was to create custom tools for my own composition and sound design work, and be able to share both my musical creations and the tools I designed to create them. The result of this work was *Vendaval, Granulera*, and *Cristalera*. This paper will present a detailed overview of these Cabbage plugins, and share my experience and advice about distributing them through the *itch.io* storefront.

Keywords: Csound, Cabbage, Plugins, Distribution, Granular

1 The Story Behind *Vendaval*, *Granulera*, and *Cristalera*

Vendaval a procedural synthesizer that algorithmically models the sound of wind. *Granulera* is a synthesizer that produces aleatoric granular textures. *Cristalera*[1] is a granular-based audio processing effect which produces glitchy textures.

Vendaval was originally designed to support the work I was doing as a sound designer on a video game. The idea was to create a procedural synthesis algorithm that would be rendered in real-time by Unreal Engine as a FMOD plugin compiled with Cabbage. After writing the Csound code, I realized it could be a useful tool for sound design outside of that specific project, specially if transformed into a MIDI synthesizer, which I did. [2]

Granulera was created for another project. I was hired to write the soundtrack for the game *Apotheosis*[3] and wanted to incorporate granular textures into the score. Since I didn't have access to any granular synthesizers at the time, I decided to develop my own.

* I'd like to thank my friend Pedro Sodre for introducing me to Csound, Dr. Richard Boulanger for his guidance and mentoring in writing this paper, and Rory Walsh for his crucial support in the Cabbage forum during the development of these plugins.

[1] Links to download all three plugins are included in the references section.

[2] Before learning Csound I was very new to coding. Cabbage is what inspired me to learn Csound in order to make plugins and procedural audio within game engines. I owe a great deal of gratitude to Rory Walsh for developing such an inspiring program.

[3] A free demo of the game can be downloaded here: `https://store.steampowered.com/app/1752030/Apotheosis/`

Cristalera is essentially the DSP version of *Granulera*. I was producing a song in collaboration with a friend and wanted the background vocals to have the same granular textures I achieved with *Granulera*. *Cristalera* employs the same granulation method, but using it to process live audio input.

2 About the Plugins

2.1 Vendaval

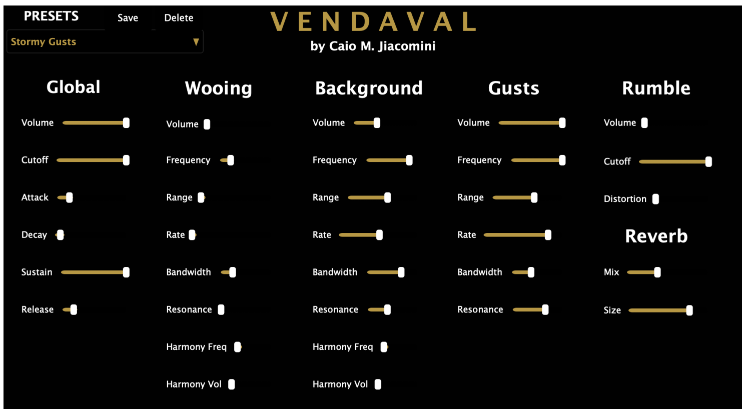

Fig. 1. *Vendaval's* user interface.

Vendaval works by generating noise, filtering it with a band-pass, and modulating the filter's center frequency randomly with the jspline opcode. There's also a low-pass filter tuned to the same frequency as the band-pass in order to allow the user to add resonance to the sound. The plugin has four modules that perform different functions employing this general chain: *Wooing*, *Background*, *Gusts*, and *Rumble*. [4]

The *Wooing* and *Background* modules work in the exact same way as described above but offer different value ranges for the user, with *Wooing* having the band-pass filter tuned to a higher frequency range with a narrower bandwidth while the *Background* module has the filter tuned to a lower frequency range with a broader bandwidth. The *Gusts* module applies a LFO to the filter frequency after the spline modulation, as a way to simulate rapid gusts of wind. The final module, *Rumble*, simply applies a low-pass filter to pink noise and distorts it, allowing the cutoff frequency to be set up to 200 Hz, in order to add a present low end for higher intensity wind soundscapes.

[4] The idea for this plugin originated after reading the book *Principles of Game Audio And Sound Design* [3], which describes how wind can be simulated procedurally with filtered noise.

The *Wooing* and *Background* modules also feature a secondary band-pass filter that harmonizes with the first one to simulate the resonance created by wind blowing through small holes. The user has control over the volume of the signal that passes through that filter as well as the frequency of the harmony, given as a multiplier of the base frequency.

The user can control each module's volume, filter frequency, the jspline range and rate, the bandwidth of the band-pass filter, and the low-pass resonance. The following code shows the core design of the background module:

```
aNoise noise 1, 0
aNoiseBp butterbp aNoise, kCF + kJit, kBW
aNoiseLp moogladder aNoiseBp, kCF + kBW + kJit, kReson
aNoiseBpHarm butterbp aNoise * kHarmVol, (kCF + kJit) * kHarmFreq, kBW
aNoiseLpHarm moogladder aNoiseBpHarm, (kCF + kBW + kJit) * kHarmFreq, kReson
aNoiseBalanced balance aNoiseLp + aNoiseLpHarm, aNoiseBp
aBackground = (aNoiseBalanced * kVolume) * kBackgroundVolume
```

2.2 Granulera

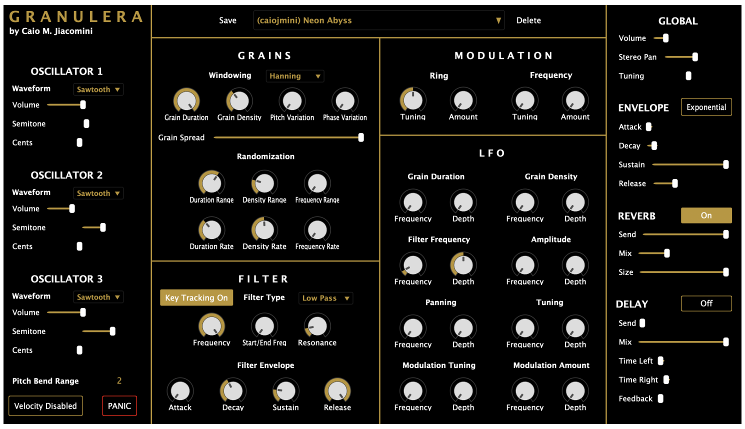
Save (caiojmini) Neon Abyss
OSCILLATOR 1 Waveform Sawtooth
OSCILLATOR 2 Waveform Sawtooth
OSCILLATOR 3 Waveform Sawtooth
Pitch Bend Range 2
Velocity Disabled
GRAINS Windowing Hanning
FILTER Key Tracking On
Filter Type Low Pass
ENVELOPE Exponential
REVERB On
DELAY Off

Fig. 2. *Granulera's* user interface.

Granulera features three oscillators capable of producing basic waveforms, a filter section with an ADSR envelope, reverb and stereo delay effects, an ADSR envelope for the global amplitude, ring and frequency modulation, and LFOs for key parameters.

The granulation of the signal was achieved by using the schedkwhen opcode to trigger each grain individually[5], allowing the stereo position of each grain

[5] The design for this plugin was based on the granulation method described in chapter 3F of the *Csound FLOSS Manual*[1].

to be randomized and passed in as a p-field. The interface allows the user to control how spread out in the stereo field the grains can be. It also offers control over the granulation windowing shape, grain duration, grain density, as well as control over the pitch and phase randomization range for each grain. On top of that, the grain duration and density values, as well as the global tuning of the instrument, are modulated randomly with the jitter opcode, with the interface allowing control over the range and rate of the modulation.

The trickiest part of developing *Granulera* was making it so the filter envelope's release behaved in a natural way if the user automated the filter frequency in the middle of the envelope. The issue was that the release stage of the envelope would always start from the frequency value that was defined during the initialization pass for the note, so if the user automated the filter frequency either through a MIDI CC or an automation lane in a digital audio workstation, the frequency would abruptly jump to whatever value it was set to during i-time for the release stage. The solution was to use the changed2 opcode to detect if the filter frequency value has been altered, using that to change the value passed to the envelope. Here's the excerpt of the code for this solution:

```
kFilterChangedTrig changed2 gkFilterFreq
kFilterChangedTrig2 init 0
if kFilterChangedTrig == 1 then
     kFilterChangedTrig2 = 1
endif
if release() == 0 then
     if kFilterChangedTrig2 == 1 then
          kFilterFreqSum = gkFilterFreq
     else
          kFilterFreqSum = kFilterEnv
     endif
else
     kFilterFreqSum = kFilterEnv
endif
```

2.3 Cristalera

Cristalera uses the same schedkwhen granulation method to process live audio input. This was achieved with four Csound instruments: one that is constantly reading the dry input signal with the inch opcode and writing it to a channel with the chnmix opcode, one instrument that triggers the grains with schedkwhen, one instrument that reads the input signal and envelops it in a window when triggered by the schedkwhen, and one instrument that mixes between the dry and the affected audio with the ntrpol opcode. Here's an excerpt of the code showcasing the schedkwhen granulation method[6]:

[6] The code was edited for the sake of size. A repository with the full source code can be found here: `https://github.com/CaioMJ/Cristalera`

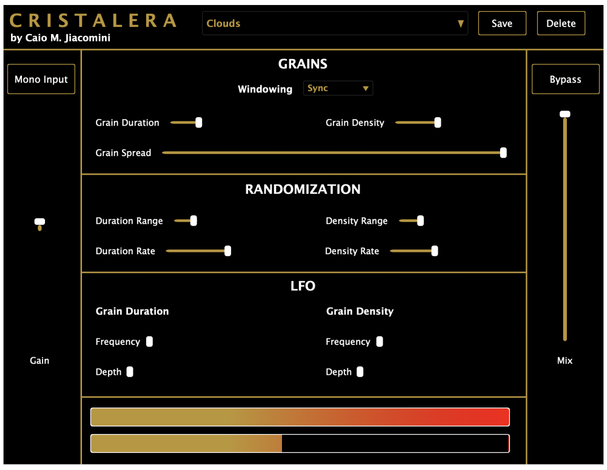

Fig. 3. *Cristalera's* user interface.

```
instr GrainTrigger
     (...)
     kDurTotal limit kDur + kDurVar + kLfoDur , 0.01, 0.9
     (...)
     kDensityTotal limit kDensity + kDensityVar + kLfoDensity, 0.1, 80
     kTrigger metro kDensityTotal
     schedkwhen kTrigger, 0, 0, "Grains", 0, kDurTotal
endin
instr Grains
     aSig chnget "DrySignalSum"
     iWfn = chnget:i("WindowingSelection")
     iBalance = 1 - p3
     kWindowIndex line 0, p3, 1
     kWindowEnv table kWindowIndex, iWfn, 1
     aWindowEnv interp kWindowEnv
     aSig *= (aWindowEnv * iBalance)
     (...)
     aGrainL, aGrainR pan2 aSig, .5 + iRandomPan
     chnmix  aGrainL, "GrainSignalL"
     chnmix  aGrainR, "GrainSignalR"
endin
```

3 Distribution: Issues and Succeses

It was always my thought that these tools might be useful to others. To share them, I decided to use the *itch.io* storefront, a platform focused on distributing independent video games but flexible enough to support the distribution of software, allowing developers to host their products for free and determine the percentage of their sales that goes back to the website. I opted to distribute my plugins in a "pay-what-you-want" model with a suggested donation amount. This allowed users to download and use them for free, with the option of financially supporting my work if they used and liked it.

While most people had no trouble setting up my plugins, I received a good number of inquiries from users who had issues running them, reporting problems that I had no idea how to solve because I don't currently own a version of the operating system they were using, so be aware you might need to spend some time doing customer support if you decide to distribute your plugins. Another issue, particularly with distribution over *itch.io*, has to do with the fact that platform sometimes charged duplicate payments. Because of this, I spent a lot time scrutinizing every transaction to refund accidental duplicated payments. Be aware that some sites that might seem easier to use initially could add an extra layer of demand due to issues like this.

However, the positive responses I got from users far overwhelms those frustrations. I've had interactions with people from all over the world writing me about how useful my plugins are, small audio blogs featuring my work with a write up, and strangers in forums and discord servers recommending them to other people. Despite being the first one I developed, *Vendaval* has been my most successful one yet, to my surprise. It's my most downloaded plugin, currently sitting at 4,180 downloads, as well as my highest grossing one. I thought the more refined and versatile *Granulera* would be my most successful one, but people ended up flocking to the simpler *Vendaval*, so don't underestimate creating simple plugins that excel at one super specific function.

4 Final Remarks and Next Steps

Csound and Cabbage have opened new possibilities for me that I would never have imagined otherwise. Designing my own plugins has been incredibly rewarding, and using them in my own work has lead to new and fulfilling aesthetic directions. Seeing others being excited about the tools I developed has also been incredible.

For next steps, I am currently planning on updating *Vendaval* to have multiple instances of each module with independent controls, allowing users to create more complex soundscapes within a single instance of the plugin. I'm also currently working on implementing those plugins in Unity with the CsoundUnity wrapper, inspiring new sets of works and tools that I am sharing at my college.[7]

[7] I currently work as a film and video game scoring tutor for my college and aim to push CsoundUnity as a tool within the department.

References

1. Heintz, J. et al.: The Csound FLOSS Manual, `https://flossmanual.csound.com`
2. Cabbage website: `https://cabbageaudio.com`
3. Sinclair, J.: Principles of Game Audio And Sound Design. Routledge (2020).
4. *Vendaval* download link: `https://caiojmini.itch.io/vendaval`
5. *Granulera* download link: `https://caiojmini.itch.io/granulera`
6. *Cristalera* download link: `https://caiojmini.itch.io/cristalera`