Huygens & Hordijk

Adam Gottesmann

Autumn Term, 2021

Abstract

Hordijk and Huygens are based on a couple of synthesiser modules that heavily inspired the designer.[1] They are both sonically and conceptually contrasting generative music machines, which use traditional synthesis methods in an innovative fashion. This project is therefore an attempt to implement the basic functionalities of these modules in Pure Data.[2]

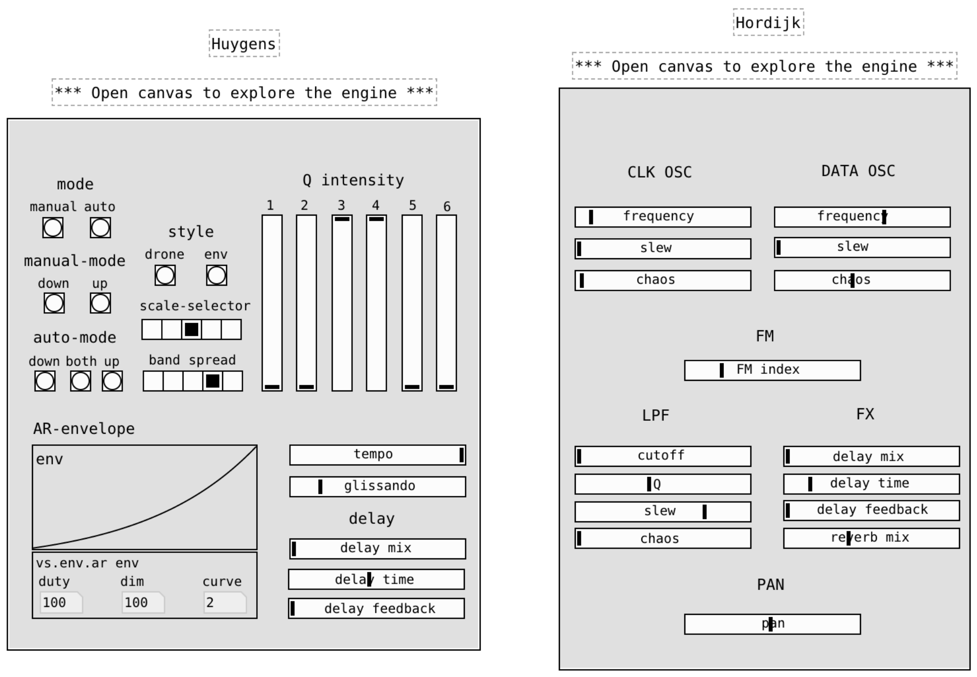

Figure 1: Hordijk's and Huygens' User Interface

[1] Rungler by Rob Hordijk and Spectral Multiband Resonator by *4ms* [1].
[2] Pure Data will henceforth be referred to as *pd*.

1 Context & Theory

To get a better understanding of Huygens and Hordijk, it is first necessary to cover the history and technical underpinnings of two synthesis techniques: Subtractive Synthesis and Frequency Modulation Synthesis.

1.1 Subtractive Synthesis

In the mid 1960s, two pioneering engineers (Bob Moog and Don Buchla) had independently designed the first commercial voltage-controlled, transistor-based musical instruments [2]. Moog's instruments were designed as ad hoc tools for the studio musician while Buchla's instruments served as mediums for experimental live performances.[3] The synthesisers designed in this technical report are heavily inspired by Buchla's experimental approach to composition, since neither a keyboard nor a linear time grid is used to control them. Nevertheless, sonic characteristics of both synthesisers are heavily influenced by the versatile subtractive synthesis techniques Moog's instruments were designed to undertake.

Subtractive synthesis aims to transform a raw sound source rich in harmonics (usually too rich for aesthetic usability) into a more controlled one with the use of filters [3]. For instance, in Huygens, white noise undergoes filtering by six resonant bandpass filters configured in parallel[4] while in Hordijk, a slightly less aggressive sound source is tamed by a resonant lowpass filter. Indeed, these are only two out of countless other applications that can be performed with this synthesis technique.

1.2 FM Synthesis

FM (Frequency Modulation) synthesis, famously discovered by John Chowning,[5] is a digital modulation technique undertaken by oscillators called operators [4]. The most basic FM configuration consists of one operator (modulator operator), modulating the frequency of another operator (carrier modulator) at a particular amplitude (index of modulation). This synthesis method is very useful because it can generate a wide variety of timbres with only a pair of operators. This computational efficiency is not only what made it such a success in the 1980s. In fact, it wouldn't be an exaggeration to claim that it completely changed the hitherto soundscape of electronic music.

Attributing Chowning to the birth of FM, however, is a bit of a misnomer since technically speaking, the original FM synthesisers based on Chowning's research (e.g., Yamaha *DX7*) were using a synthesis technique called Phase Modulation, whereby the modulator operator modulates the phase rather than the frequency of the carrier operator. In fact, Hordijk makes use of Phase Modulation (PM) synthesis rather than FM synthesis although sonically speaking, the distinction is merely academic.

[3]The most famous musical examples, which show the stark contrast between the two synthesisers' outputs are Wendy Carlos' *Switched on Bach* (made with Moog) and Morton Subotnick's *Silver Apples of the Moon* (made with Buchla).

[4]This means that each band-pass filter processes the white noise independently.

[5]Well, this is only half true. Analogue FM synthesis had been used years before Chowning's discovery but Chowning indeed gets credit for implementing it algorithmically.

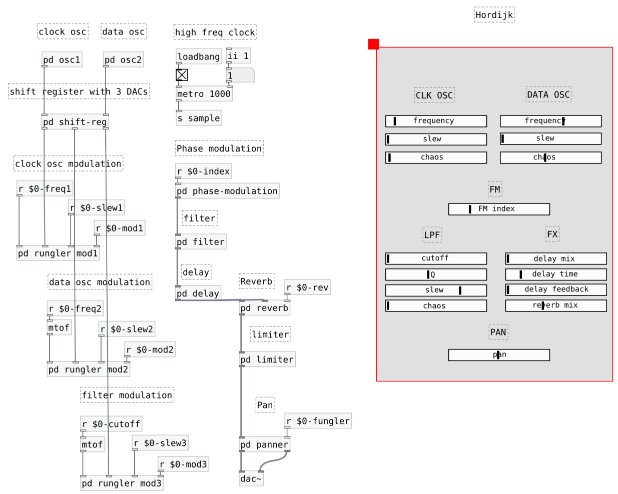

Figure 2: Hordijk's internal engine

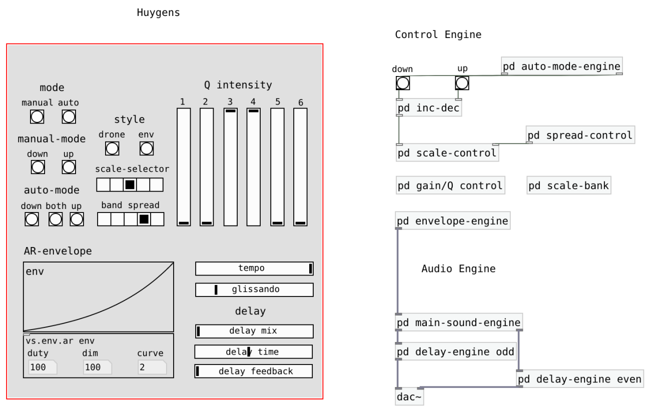

Figure 3: Huygen's internal engine

2 Design

Since we have briefly covered the two fundamental types of sound synthesis techniques both Hordijk and Huygens employ, let's take a deep dive into the technical and conceptual design of each instrument.

2.1 Hordijk

Hordijk uses both FM and subtractive synthesis techniques (Audio Engine) and is driven by a Rungler circuit (Data Engine). Let's first start analysing its Data Engine since it is the heart and soul of the synthesiser's character, and then proceed to analyse its Audio Engine. (Open the Hordijk canvas to see the entire engine.)

2.2 Hordijk's Control Engine

The Control Engine is essentially a Rungler circuit [1]. A Rungler circuit, originally conceived by Rob Hordijk, aims to mimic behaviour which is neither predictable nor completely random. The way it achieves it is by creating a stochastic pattern, realised by a pair of oscillators connected to a binary shift register.

A binary shift register is a digital circuit, which delays the data of its current stage to the next stage whenever it receives an edge-triggered clock signal (or a *bang*!). It therefore has two inputs: a data input and a clock input. In the Rungler circuit, one oscillator is connected to its clock input, while another oscillator is connected to its data input. Since the oscillators are pitch independent, chaotic patterns can emerge out of the binary shift register (open `pd shift-reg`).

Since a binary shift register is a digital circuit, its data input can only receive binary values. This means that the oscillators need to pass through a logic circuit which outputs a "one" whenever the waveform is above the zero-crossing[6] and revert to zero once it goes below the zero-crossing. To make this happen in *pd*, the position of the waveform is sampled every 1ms to be able to determine the "instantaneous" value of the waveform.[7] The instantaneous value is then sent to a logic module which outputs a one whenever the waveform is above the zero-crossing and a "zero" whenever it is below it (open `pd osc2`). A similar procedure is implemented for the oscillator driving the clock input of the binary shift register. The main difference is that the clock input of the binary shift register is only activated whenever the clock oscillator outputs a *bang* (open `pd osc1`).[8]

Because a binary shift register only outputs binary values (which are of little use on their own), it is important to convert those values into an analogue-like signal. This can be achieved by modelling a primitive DAC (Digital to Analogue converter), which will convert the stream of binary values into n possible stepped values. Since the number of stages for the binary shift register and the number of DACs are open-ended, the designer has decided to model a nine-stage binary shift register, whereby every three individual outputs of the register feed into their respective three-bit DAC, resulting in a total of three three-bit DACs. The output of the first, second and third DACs are sent back to the frequency control input of the clock oscillator, the frequency control input of the data oscillator, and the cut-off frequency control input of a low-pass resonant filter respectively (the purpose of the filter will be discussed in the next paragraph). Finally, it is possible to control the depth and slew of each DAC's output for a more flexible modulation arsenal (open `pd 3bit-dac`, contained in `pd shift-reg`).

2.3 Hordijk's Audio Engine

Since the first oscillator is a sine wave and the second oscillator is a triangle wave, a convenient way of extracting a complex harmonic spectrum from them is by using a Phase Modulation configuration (open `pd phase-modulation`). Since the phase modulator's output can be too intense, a low-pass resonant filter is implemented to tame or modify its sound (open `pd filter`). The signal is then fed into a delay module which has a flexible mix (dry/wet), delay time, and

[6]The zero-crossing term is used here to refer to the zero value of the y-axis on an oscilloscope.

[7]In theory, the sampling frequency should be much higher than 1ms but it is unfortunately the maximum clock rate of the `pd metro` object. Any suggestions to either increase the sampling frequency or to make this procedure less CPU intensive are highly welcomed.

[8]The clock oscillator outputs a *bang* whenever the previous value of the sampled oscillator is of opposite polarity to its current value.

feedback control parameters (open `pd delay`).[9] Finally, one of Miller Puckette's stock reverbs is added to the signal chain as well as a cosine stereo panner [6], before being summed into *pd*'s DAC. (For safety, a look-ahead limiter is also implemented to avoid digital clipping [7].)

2.4 Huygens

Huygens is essentially a resonant six-band stereo filter bank, which uses white noise[10] as its audio source (Audio Engine). Each band can cycle either manually or algorithmically through various scales in both ascending and descending patterns (Control Engine). (Open the Huygens canvas to see the entire engine.)

2.5 Huygens' Audio Engine

Huygens, having white noise as a sound source, followed by six bandpass filters in parallel configuration, makes it in principle a subtractive synthesiser. Since the Q (resonance) of each bandpass filter can be adjusted, it is possible to generate up to six (noisy sounding) harmonics simultaneously (open `pd main-sound-engine`).[11] Instead of directly summing the six bandpass filters together, the odd and even numbered bandpass filters are routed into the left and right channels of the stereo mix respectively. Finally, each summed bandpass filter group is routed into its own delay processor (the same delay processor used in Hordijk).

2.6 Huygens' Control Engine

The most important aspect that needs to be understood about the control engine of Huygens is that all the possible frequencies for each resonant bandpass filter are pre-tabulated. By opening `pd scale-bank`, it is possible to see that an array-based technique is used to store a selection of scales that span across multiple octaves.[12] Since there are twenty values in each array, by assigning an array index to the scales list, it is possible to access any value from it (open `pd scale-control`). Since it is now clear how specific frequencies are assigned to each noise-band, it is of good momentum to explain how the "manual" and "auto" sequencing modes operate.

It is possible to either manually or algorithmically control the frequency distribution across the filter bank. The manual one is quite straightforward: the "up" command increments while the "down" command decrements the index value on the scales list for each noise-band. This means that perceptually the pitch of each noise-band either increases or decreases.[13] The "auto-mode" is slightly more complex: once it is selected, a binary random generator gets driven by a clock rate dictated by the tempo slider. Whenever the random generator outputs a "one", the index value on the list either increments, decrements, or both (open `pd auto-mode-engine`). Whether in auto or manual mode, not only can one select from multiple microtonal scale banks (via "scale-selector"), but it is also feasible to have control over the interval value of the increment/decrement functions (via "band spread").[14]

[9]The delay patch was taken from LWM Music Youtube channel [5].

[10]Technically speaking we are using pink noise, which has been heavily lowpass filtered for aesthetic purposes.

[11]These noisy harmonics are often referred to as noise-bands.

[12]Initially, the frequency values for the selected scales were partially taken from the GitHub source code of the SMR module by *4ms* [8]. Since the source code provided too much data, the final selection of both scales and frequencies had to be determined by ear.

[13]Since we are dealing with a list of twenty frequencies, it is evident that at the last (or first) index value, the sequence will repeat itself.

[14]In addition, the frequency distribution is designed in such a way that the adjacent noise-band always differs by an index interval defined by the band spread selector.

So far, we have explored Huygens as if it didn't have any envelopes.[15] This is fortunately not the case. A very versatile AR-envelope, designed by the Virtual Sound designer Francesco Bianchi, had been implemented to get both exponential and logarithmic control over the amplitude contour of the filter bank [9]. The envelope gets triggered (in either manual or auto modes) if the "env" style button has been pressed and that a change in the index value in the scales list has occurred. Finally, a *glissando* slider was added to control the slew rate of the transition between frequencies contained in the scales list.[16]

3 Composition

It is crucial to emphasise that both Hordijk and Huygens are self-playing synthesisers. It is therefore unnecessary to compose with them in a traditional sense. Instead, it is more practical to dial in the "right" values for each synthesiser until a final cohesive soundscape effloresces. But how does one save these parameters?

The desired values (and the duration of these values) and addresses for both synthesisers' parameters can be typed into a text file. Subsequently, by using send and receive messages, an object called `qlist` can read the data contained in the text file and output the data to the specified addresses for a defined length of time (open `pd composition`). What's more, it is worth mentioning that despite of `qlist`'s deterministic values the composition will never sound identical since Huygens' auto mode and Hordijk's "chaos" sliders enable both random and stochastic functionalities respectively.

4 Evaluation

Overall, the designer is quite content with the ergonomics and sound quality of both Huygens and Hordijk. The designer, however, was unable to overcome two main limitations imposed by *pd*'s internal engine. Firstly, since it is supposedly impossible to use the `pd metro` object with a clock rate faster than 1ms, the behaviour of the Rungler circuit in *pd* is vastly inferior to its original counterpart.[17] Secondly, the sonic artefacts created in Huygens when transitioning between a low to a high (and high to low) pitched noise-band frequency forced the designer to implement an 800ms latency, which evidently reduces the quality of the user experience in manual mode.[18] These limitations, however, were quite informative since it made it clear why one would prefer to design a synthesiser with either analogue components or with a lower-level programming paradigm.

[15]In fact, it is completely possible to assume so if one presses on the "drone" style button, which effectively bypasses Huygens' envelope generator.

[16]There is one more aspect to the Huygens design that should be mentioned to avoid any potential confusion. Due to *pd*'s DSP limitations, noise-bands cannot transition between a very high and low pitch without making an unpleasant "pingy" sound. The only way the designer could bodge this issue is by creating an 800ms latency whenever there is a change in the scales list index value. During these 800ms, enough time is given for the unpleasant artefact to settle by muting any noise-band which undergoes such a transition (open `pd bug-fix`).

[17]The original Rungler was designed with physical components hence the sampling rate of the waveform is sufficiently fast.

[18]Analogue noise-bands which undergo such transitions are not endowed with these sonic artefacts.

References

[1] K. Bjorn and C. Meyer, *Patch & Tweak - Exploring Modular Synthesis*, 1st ed. Denmark: Bjooks, 2018, pp. 169, 262.

[2] T. Holmes, *Electronic and Experimental Music: Technology, Music, and Culture*, 3rd ed. United Kingdom: Routledge, 2008, pp. 207–226.

[3] R. Bianchini and A. Cipriani, *Virtual Sound*, 1st ed. Rome, Italy: ConTempoNet, 2011, p. 75.

[4] C. Dodge and T. A. Jerse, *Computer Music: Synthesis, Composition and Performance*, 1st ed. New York: Schirmer Books, 1997, pp. 115–139.

[5] L. W. Moore. (2017, Sep.) Sunday Night Synthesis S03E09 — Making a Delay Effect in Pure Data. [Online]. Available: https://www.youtube.com/watch?v=z5nxbPenEIs&t=555s

[6] A. Farnell, *Designing Sound*, 1st ed. Cambridge, Massachusetts: The MIT Press, 2010, p. 222.

[7] D. Holzer. (2010) Normalizing and DC Offset. [Online]. Available: https://archive.flossmanuals.net/pure-data/audio-tutorials/dc-offset.html

[8] D. Green. (2020, Aug.) Spectral Multiband Resonator. [Online]. Available: http://www.github.com/4ms/SMR

[9] F. Bianchi, A. Cipriani, and M. Giri, *Pure Data: Electronic Music and Sound Design — Theory and Practice*, 1st ed. Rome, Italy: ConTempoNet, 2021, vol. 1, p. 170.