Wat is nieuw in WaveWizard_2013?

- Geluidssporen S1, S2 en S3 zijn **32-bits *floating point***
- Automatische lineaire interpolatie van alle buffers
- bufferwaarden met negatieve indices gedefinieerd als nul
- Spectrumanalyzer twee frequentieresoluties
- **`Chorus`**
- **`Correlatie`**
- **`Decorrelator`**
- **`DFT`** (Discrete Fourier Transform)
- **`FFT`** (Fast Fourier Transform) nu ook *complexe* signaal-*input*
- **`FIR-filter`**
- **`Flanger`**
- Foutmeldingen uitgebreid, Clip-melding
- **`Galm`** d.m.v. 16-kanaals Feedback Delay Network (FDN)
- **`int(x)`** en **`frac(x)`**
- **`LPC analyse`** en **`LPC synthese`**
- **`Open WAV`**
- **`Priemgetallen`**
- **`Resofilter`** (Helmholtz resonator)
- **`Snelle Convolutie`**
- **`Sonogram`**

Geluidssporen S1, S2 en S3 zijn 32-bits *floating point*

De geluidssporen S1, S2 en S3 zijn niet langer 16-bits *integer* buffers, maar 32-bits *floating point*. Daardoor neemt de geluidskwaliteit enorm toe. Recursieve bewerkingen, zoals resonerende filters en de *snaarplukvergelijking* kun je nu ook direct als recursieve differentievergelijking op een geluidsspoor realiseren, wat in de oude versie t.g.v. afrondingsfouten al na enkele seconden zou leiden tot een periodiek i.p.v. een dempend signaal.

Snaarpluk: **`S1[n] = 0,5*(S1[n-N] + S1[n-N+1]).`**

Resofilter: **`S2[n] = A*S1[n] + B*S2[n-1] + C*S2[n-2])`** (zie ook **`Resofilter`**).

Een ander gevolg is dat nu geen *syntax*fout meer wordt gemaakt bij overschrijding van het 16-bits getalbereik van $\pm 2^{15}$; wel blijft *clipping* optreden als je S1, S2 en S3 afspeelt buiten dat bereik, want voor de geluidskaart blijft het 16-bits bereik natuurlijk gewoon bestaan. Daarom geeft WaveWizard nu een *melding* bij Clipping.

Zie Foutmeldingen uitgebreid, Clip-melding

Automatische lineaire interpolatie van alle buffers

Stel, je hebt de volgende code:

```
S1[3] = 3000
S1[4] = 4000
Print S1[pi]
```

Als je deze code draait op de oude versie, dan krijg je als antwoord: **`S1[pi] = 3000`** want in de oude versie wordt eerst van pi = 3,14... het deel achter de komma verwijderd, zodat je krijgt pi = 3. Dan wordt gelezen: **`S1[3]`**.

Als je de code draait op de nieuwe versie, dan is het resultaat:
`S1[pi] = 3141,59265358979`

In de nieuwe versie vindt dus automatisch lineaire interpolatie plaats tussen twee opeenvolgende getallen in een buffer; zowel bij de audio-buffers S1, S2 en S3 als de double float buffers. Daardoor zijn de (discrete) buffers "quasi-continu" geworden.

Toepassingen:
(1) alle bewerkingen waarin tijdschaalveranderingen voorkomen, bijv. **`Chorus`**, **`Flanger`**;
(2) stuursignalen, zoals een envelop generator (zie H4) of laagfrequente ruis;
(3) opschaling indices grafieken.
Soms is automatische interpolatie ongewenst. Dan kun je noteren:
`S1[int(pi)]`.
Zie ook **`int(x)`** en **`frac(x)`**.

bufferwaarden met negatieve indices gedefinieerd als nul

In de oude versie kreeg je een foutmelding als je noteerde **`Print F1[-123]`**, want *negatieve* indices verwijzen naar buffergetallen die eenvoudig niet bestaan. In de nieuwe versie zijn Fk[x] en Sk[x] gelijk aan nul voor elke x < 0 of x > bufferlengte. Nu levert de instructie **`Print F1[-123]`** dus nul op.
Toepassing: het wordt nu heel eenvoudig om filters en echo-effecten direct te schrijven in de vorm van een differentievergelijking met beginconditie nul (wat meestal het geval is). Zie Sound Design H4.3).
Zie ook **`FIR-filter`**.

Spectrumanalyzer twee frequentieresoluties

In de oude versie had de spectrum analyzer slechts één frequentieresolutie, nl. 43 Hz. Dat is de resolutie van de FFT met 1024 samples (F_s / 2^{10} = 43,066..). In heel veel praktische gevallen heb je graag een wat hogere resolutie met name in het lage frequentiegebied. Daarom heeft de nieuwe versie een extra instelling gekregen: 0 - 5000 Hz. Daardoor wordt de FFT berekend over 4096 samples, zodat de frequentieresolutie gelijk wordt aan F_s / 2^{12} = 10,76660 Hz. Alleen het gebied van 0 tot 5000 Hz wordt afgebeeld.
Een andere verbetering is dat de spectrumanalyzer de Gemiddelden en Maxima ook afbeeldt op de logaritmische representaties.

Chorus

Het chorus-effect maakt "intern" een (instelbaar) aantal kopieën van een geluid en speelt die gelijktijdig af, elk met voortdurende, minuscule random schommelingen in de afspeelsnelheid. Daardoor ontstaat de indruk van een "koor".
Voorbeeld: Preset chorus en decorrelator.txt.
Zie ook **`Flanger`**.

Correlatie

De correlatiefunctie is zeer belangrijk in alle vormen van signaalverwerking. Een praktische toepassing bij audio is het meten van de ruimtelijkheid van een stereo-

geluid. De correlatie is genormaliseerd, d.w.z. dat de waarde ervan altijd ligt in [-1, +1]. Als de correlatie tussen het linker- en rechter spoor gelijk is aan 0, lijkt het geluid afkomstig uit een breed gebied tussen de boxen, dus een maximale stereo-ervaring. Is de correlatie tussen beide sporen gelijk aan +1, dan klinkt het geluid alsof het "door een sleutelgat" komt, precies tussen de boxen, of "midden in je hoofd" (bij koptelefoon). Is de correlatie gelijk aan -1, lijkt de geluidsbron zich achter je te bevinden.
Zie ook **Decorrelator**.

Decorrelator

Wil je een monogeluid opblazen tot stereo, dan kun je niet volstaan met het maken van een kopie van het monosignaal en die op een tweede spoor te zetten. In dat geval zou de correlatie tussen beide sporen gelijk zijn aan 1. Ook een echo toevoegen, of dat tweede spoor een andere amplitude geven, verlaagt de correlatie niet. Het gaat erom een bewerking toe te passen die de *golfvorm* (het *fase*spectrum) van de kopie zoveel mogelijk doet verschillen van die van het origineel, terwijl toch de klankkleur, het timbre, hetzelfde blijft. Dat kan door een filter dat de sterkteverhoudingen van de boventonen ongemoeid laat ("All Pass"), maar dat wel de fases ervan verandert. Zo'n filter noemen we een *decorrelator*.
Voorbeeld: Preset chorus en decorrelator.txt.
Zie ook **Correlatie**

DFT (Discrete Fourier Transform)

De Discrete Fourier Transformatie (DFT) is te beschouwen als de "computerversie" van de (continue) Fourier serie-expansie. De periode wordt uitgedrukt in aantal samples en kan elk positief geheel getal zijn. Er bestaat een buitengewoon snelle variant van de DFT, nl. de Fast Fourier Transform (FFT). Tegenover de kolossale snelheidswinst van de FFT staan enkele kleine beperkingen: (1) bij de FFT moet het aantal samples waarin de periode wordt uitgedrukt altijd een macht van 2 zijn; (2) De FFT berekent altijd het *gehele* spectrum, terwijl de DFT ook slechts een gedeelte kan uitrekenen en daardoor in bijzondere gevallen toch sneller en preciezer kan zijn dan de FFT.
Zie ook **FFT**.

FFT *ook complexe signaal-input*

In deze nieuwe versie kan ook van een *complex* signaal de FFT worden berekend (in de oude versie werd verondersteld dat het imaginaire deel van het inputsignaal altijd nul is). Bovendien is het maximale aantal samples van de periode (zie **DFT**) uitgebreid tot 2^{17} (=131072). Bij de standaard samplingfrequentie F_s = 44.100 Hz, is daarmee de maximale frequentie-resolutie gelijk aan 44.000 / 131072 = 0,3364 Hz!
Zie ook **DFT**.

FIR-filter

Deze functie is een directe implementatie van de algemene, niet-recursieve differentievergelijking van een feedforward systeem (zie Sound Design H4.3):

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + \ldots + a_N x_{n-N}$$

Deze berekening, die ook wel wordt aangeduid als de **convolutie(som)** van twee discrete signalen a_n en x_n, kun je natuurlijk ook zelf uitvoeren door het commando

Bewerk signaal maar **FIR-filter** is (vanwege de directe implementatie in C++) aanzienlijk sneller.
Nog weer *veel en veel* sneller is: **Snelle Convolutie**.

Flanger

Het geluid van het Inputspoor wordt intern "opnieuw opgenomen met een bandrecorder" waarbij het toerental van de motor niet constant is, maar sinusvormig toe- en afneemt. Als het geluid op het spoor bestaat uit een toon met constante frequentie, hoor je een vibrato. Dit effect treedt op als je de spoel (Eng.: *flange*) van de bandrecorder met je vinger regelmatig afremt en weer loslaat, vandaar de naam *flanging*.
Het flanger-effect wordt in de praktijk meestal gemaakt (ook door WaveWizard) door het inputsignaal gelijktijdig drie maal te "flangen" met dezelfde (instelbare) modulatiefrequentie en -diepte, maar met sinusgolven die onderling 120 graden in fase zijn verschoven. Daardoor ontstaat de kenmerkende, "glazige", meer ruimtelijke klank.
Voorbeeld: Preset Flanger.txt.
Zie ook **Chorus**.

Foutmeldingen uitgebreid, Clip-melding

Een wiskundig volkomen juiste expressie als **2sin(x)** levert in de meeste programmeertalen een "syntax error" op, vanwege het ontbreken van vermenigvuldigingsteken (*). In de oude versie van WaveWizard werd dit soort fouten niet gedecteerd, in de nieuwe versie wel.
In de nieuwe versie wordt een aantal fouten niet meer als "fataal" beschouwd. Een fatale fout is een fout waardoor WaveWizard de berekening direct afbreekt en een melding genereert. Een van de belangrijkste veranderingen is dat "Clipping" (het overschrijden van de maximale amplitude van een 16-bits signaal) niet meer als fatale fout kan worden gezien, omdat de sporen S1, S2 en S3 nu 32-bits floating point zijn. Er zijn vele situaties denkbaar waarin de signalen op de sporen tijdelijk een veel grotere amplitude hebben en pas aan het eind van een lange reeks bewerkingen worden "neergeschaald" naar 16-bits. Daarom wordt bij Clipping niet meer afgebroken. Wel verschijnt er een melding in Memo.
Zie ook **Geluidssporen S1, S2 en S3 zijn 32-bits *floating point***.

Galm

Een *state of the art* galm-model is het zg. *Feedback Delay Network* (FDN). Het levert een zeer hoogwaardige galm en is bovendien qua rekentijd erg zuinig. Je kunt het beschouwen als een "generalisatie" van het principe van de snaarplukvergelijking (Sound Design H7).
De implementatie van het FDN in de nieuwe versie van WaveWizard is 16-kanaals (hoge galmdichtheid) en geeft toegang tot de eigenlijke 114 netwerkparameters, die je instelt door middel van een preset.
Zie Preset stereo pulsrespons FDN 16X16.txt.
Zie ook **Snelle Convolutie**.
In *Sound Design Deel II* wordt een hoofdstuk gewijd aan galm en het FDN.

int(x) en frac(x)

geven van floating pointgetal ***x*** resp. het deel vóór en achter de komma:

```
int(pi)  = 3
frac(pi) = 0,141592653589793
```

Zie ook **Snelle Convolutie**.

LPC analyse en **LPC synthese**

In Sound Design H1.4 wordt een spraakvocodertechniek besproken en het filter dat daarbij wordt gebruikt (Fig 5). Voorbeeld (a) *"Maan, zaag, Fien..."* is gemaakt met de functies **LPC analyse** en **LPC synthese**.
De *analysefunctie* deelt het te analyseren signaal op in frames en berekent van elk frame een filter. De *inverse* van dat analysefilter heeft een amplitudekarakteristiek (Sound Design H5.1) die het spectrum van het analysesignaal benadert en dus overeenkomstig klinkt. Een lange reeks frames levert dus een lange reeks analysefilters op, en de inversen daarvan waarvan de klank in zeer kleine stapjes evolueert.
De *synthesefunctie* filtert een gegeven signaal met dit inverse filter. Daardoor krijgt dat gegeven signaal dezelfde functie als die de stembanden vervullen in het spraakkanaal, terwijl het inverse filter zelf de rol speelt van het spraakkanaal.

In *Sound Design Deel II* worden twee hoofdstukken gewijd aan LPC; een praktisch hoofstuk en de wiskundige afleiding van het algoritme.

Open WAV

Een geluidsbestand openen kun je op twee manieren doen: (1) direct met de muis klikken in het bekende bestand-selectievenster; (2) in de nieuwe versie ook als presetcommando **Open WAV**. Daarmee kun je een onbeperkt aantal geluidsbestanden automatisch achter elkaar te openen en op verschillende tijdstippen van verschillende sporen zetten.

Priemgetallen

Plaatst alle priemgetallen die in het opgegeven interval voorkomen in een buffer. Deze functie wordt o.a. toegepast voor het vinden van optimale delaylijn-lengten van het FDN-galmcircuit. Zie **Galm**

Resofilter (Helmholtz resonator)

De Helmholtz-resonator ("toonbol") is een akoestisch massa-veer-dempersysteem. Met de computer benader je het door een tweede-orde *recursieve differentievergelijking* (Sound Design H4) van de vorm:

$$y_n = x_n + ay_{n-1} + by_{n-2}$$

Deze vergelijking kun je beschouwen als de "basis-bouwsteen" van de (digitale) signaalverwerking; je kunt er gemakkelijk meervoudige resonanties mee bouwen, die o.a. voorkomen in spraak en alle gedempte trillingen (bijv. percussieve geluiden). Voor zekere waarden van a en b werkt deze resonator als een extreem snelle sinusgenerator. In de functie **Resofilter** worden de coëfficiënten a en b niet rechtstreeks ingevoerd maar omrekeningen daarvan, **ResoFreq** en **bandbreedte**, die wat praktischer zijn en meestal worden aangetroffen in audio-toepassingen. **Gain** is de coëfficiënt van de input-term x_n.

```
Resofilter
  Input                  puls    ! of bijv. 'S1[200]'
  Output                 S2[0]
  Gain                   1
  ResoFreq               1200
  bandbreedte            0
  duur                   20*Fs
  additief ('j' of 'n')  j
```

Een analyse van de Helmholtz-resonator volgt in *Sound Design Deel II*. Daar wordt afgeleid dat $a = 2r\cos\omega$ en $b = -r^2$, waarin $r = e^{-\pi q/F_s}$ en $\omega = 2\pi f/F_s$, met q= **bandbreedte** en f= **ResoFreq**.

Snelle Convolutie

Lees eerst: **FIR-filter**.
Deze functie maakt gebruik van de convolutie-eigenschap van de Fouriertransformatie (*"convolutie in tijddomein = vermenigvuldiging in frequentiedomein"*).
Vermenigvuldigen is veel minder bewerkelijk dan convolveren, en het noodzakelijke heen- en terugtransformeren kan dankzij de FFT (zie **DFT** een **FFT**) een spectaculaire snelheidswinst opleveren.
De orde van de differentievergelijking, *N*, kan in praktische gevallen oplopen tot in de honderdduizenden (bijv. de convolutie van een geluid met de pulsrepsons van concertzaal voor het maken van zeer realistische galm). Voor *N* >> 100, wordt **Snelle Convolutie** *tot meer dan 4000 maal zo snel als* **FIR-filter***!!* Deze snelheidswinst heeft zeer interessante, praktische toepassingsmogelijkheden, niet alleen galm, maar ook hoge-orde FIR-filters.

Sonogram

In **Sonogram** is het laatste invoer-veld:

```
Animatie ('j' of 'n')
```

gewijzigd in:

```
Raster ('j' of 'n')
```

Als je presets van WaveWizard_2010 draait op de nieuwe versie, dan wordt dit veld automatisch vervangen. Je kunt oude presets dus gewoon draaien zonder dat je een foutmelding krijgt.