# [vəkəωdər] – PiF Media Technologies

Le 23 mai 2oo8. Delft, Pays-Bas.

**NOTE : Un grand merci à SINK pour toute la partie graphique. SINK a été d'une aide précieuse dans ce projet. http://www.sinkmusic.com/**

## *1. Installation*

Copier/coller le fichier PiFVocoder.dll dans votre répertoire VSTPlugins. Voilà.

## *2. Utilisation dans un séquenceur*

Insérer l'effet Vocoder en tant qu'insert dans une piste audio stéréo.
Le canal gauche de cette piste sera le signal modulant (la voix).
Le canal droit de cette piste pourra être le signal modulé (ex : le clavier, qui est modulé par la voix).
Créer une piste Midi.
Choisir Vocoder comme sortie pour la piste midi.
Choisissez votre périphérique midi comme entrée de cette piste. Il pourra contrôler le synthétiseur interne, qui fera alors office de signal modulé.

**Truc** : pour les utilisateurs de Cubase, créez une première piste audio mono, et mettez sa panoramique à fond à gauche. Créez une seconde piste audio mono et placez sa panoramique à fond à droite. Créez une piste groupe, et routez la sortie de chacune des pistes mono sur ce groupe. Placez alors Vocoder en insert dans la piste groupe. Vous aurez alors un parfait control de toutes les sources.

## *3. Description des paramètres*

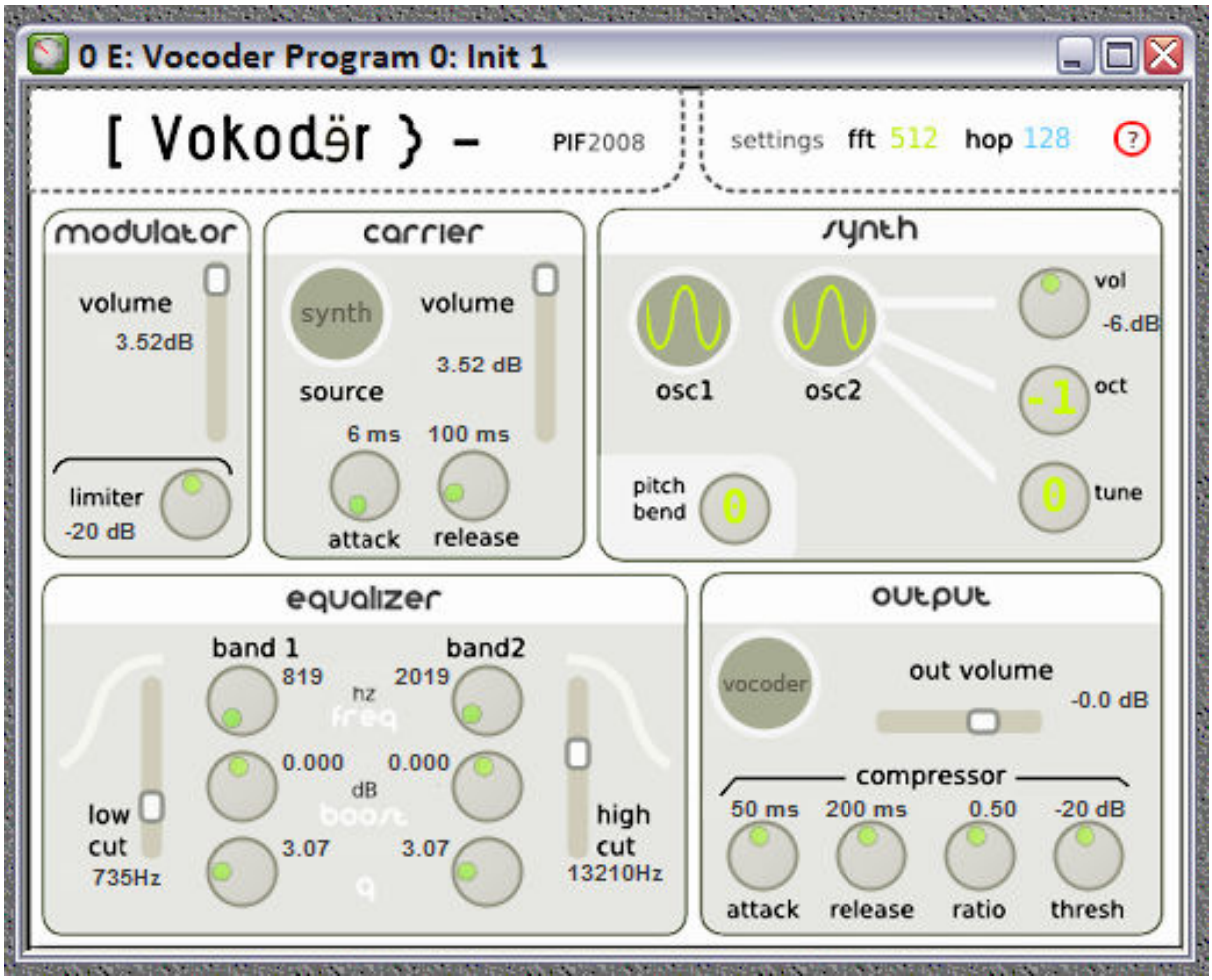


**Figure 1 - Nice GUI isn't it???**

**Module Synth :**

- Waveform 1 et 2 : vous permettent de choisir la forme d'onde des deux oscillateurs du synthétiseur de **[vəkəωdər] :** Trois formes d'ondes sont proposées ; Sinus, Dent de scie et Carré, ainsi qu'un générateur de bruit blanc.
- Wvfrm 2 Vol. : amplitude du deuxième oscillateur.
- Wvfm2 Octave : la hauteur du son est donnée par Waveform 1. Wvfm2 Octave permet de choisir à combien d'octaves supplémentaires va sonner Waveform2, de -1 à 4.
- Detune : permet d'insérer un décalage de la Waveform 2 par rapport à la note jouée, en demi-tons, de 0 à 12.
- Pitchbend : contrôle l'échelle de la molette de pitchbend, de 1 à 8 demi-tons.

Note : la molette de modulation introduira une modulation d'amplitude.

**Module Modulator :**

- Limiter : fixe le niveau du limiter de l'entrée audio du modulant (la voix)
- Volume : fixe le niveau général de l'entrée audio (il est situé après le limiter).

**Module Carrier :**

- Source : permettre de choisir entre l'entrée stéréo droite ou bien le synthétiseur comme base pour le Vocoding. Carrier = signal modulé.
- Volume : fixe le volume du carrier (signal modulé).
- Attack : temps d'attaque du compresseur sur le carrier, en millisecondes, de 1 à 200ms.
- Release : temps de décrochage du compresseur, en millisecondes, de 1 à 600ms.

**Module Eq :**

- Low Cut Freq : fréquence de coupure du filtre coupe bas, de 20 à 3000Hz.
- Peak 1 Freq : sélectionne la fréquence centrale du filtre paramétrique 1
- Peak 1 Q : sélectionne la largeur de bande du filtre paramétrique 1 autour de sa fréquence centrale.
- Peak 1 Gain: sélectionne le gain (positif ou négatif) du filtre paramétrique 1.
- Peak 2 Freq : sélectionne la fréquence centrale du filtre paramétrique 2
- Peak 2 Q : sélectionne la largeur de bande du filtre paramétrique 2 autour de sa fréquence centrale.
- Peak 2 Gain: sélectionne le gain (positif ou négatif) du filtre paramétrique 2.
- High Cut Freq : fréquence de coupure du filtre coupe haut, de 4000Hz à 20kHz.

**Module Masters & Vocoder:**

- Output : sélectionne ce qui sera la sortie du plug in; le Vocoder, le Carrier ou bien le Modulator. Cela permet de vérifier les différents signaux non traités.
- Volume : volume général de sortie
- Ratio : permet de choisir la rudesse du compresseur de sortie.
- Threshold : niveau à partir duquel le compresseur se met en marche.
- Attack : temps d'attaque du compresseur.
- Release : temps de décrochage du compresseur.

**Module Settings:**

- FFT : permet de choisir la taille de la fenêtre FFT. Plus grande est la taille, plus grande est la définition du vocoding. Une plus forte consommation sera induite par une grande valeur de la FFT.
- HOP : permet de choisir la résolution temporelle de la FFT (tous les combiens d'échantillons une FFT va être effectuée).

**Merci d'avoir suivi jusqu'ici en espérant avoir été utile à la compréhension du fonctionnement de ce plugin.**

**Pour ceux qui souhaitent en savoir plus sur les vocoder en général et sur [vəkəωdər] en particulier, rendez-vous page suivante !!!!**

## *4. En savoir plus sur [vəkəωdər]*

Comment un vocoder fonctionne-t-il ?
Le vocoder est à l'origine une méthode de télécommunication permettant de transmettre moins d'information sur une ligne téléphonique que si l'on transmettait directement le signal de la voix.
On s'en fout ! Ok, la zic la zic !
La version dérivée pour la musique fonctionne comme suit.
Le signal de la voix passe à travers une série de filtres passe bande, afin d'être découpé en bande fréquentielles. Pour chaque bande on mesure l'enveloppe du signal.
Le signal du synthé passe par ce même banc de filtres, et pour chaque bande, l'enveloppe du synthé est multipliée par l'enveloppe de la bande correspondante de l'audio. Voilà qui est fait, on a plus qu'à sommer les différentes bandes, et boum, ça fait des Chocapic !

Voilà seulement, la qualité du vocoder est dépendante du nombre de bandes : plus on aura de bandes, meilleure sera la mesure de l'enveloppe fréquentielle et meilleure sera l'effet de vocoding (plus transparent, plus défini etc…). Oui mais bon, plus on a de bandes, plus on a de filtres, et plus on a d'opérations de filtrage ; il arrive donc un moment ou le vocoder consomme beaucoup de ressources. (Et puis c'est chiant d'implémenter autant de filtres…). Pour infos, la majorité des vocoder des années 70 possède 32 voire 64 bandes.

N'y a-t-il pas un autre moyen de faire ceci sans passer par toutes ces opérations de filtrages ? Si, Fourier. Vous en avez déjà entendu parler, mais allons à l'essentiel : la FFT.
La FFT (Fast Fourier Transform) est un algorithme mis au point en 1974 par deux chercheurs du MIT (Cooley and Tukey) permettant d'obtenir le spectre (contenu fréquentiel) d'un signal en un nombre réduit d'opérations. Ah aaaaaah drôlement utile pour pas mal de choses, et notamment notre vocoder.
Au lieu de filtrer avec tous ces filtres, on applique une FFT aux deux signaux (audio et synthé) et on obtient directement le contenu fréquentiel. Si on fait une FFT sur 1024 échantillons de nos signaux, nous aurons 1024 « points de mesures » de leur spectre : l'équivalent de 1024 filtres avec beaucoup moins d'opérations !
On poursuit exactement le même raisonnement qu'avec les filtres : on mesure l'enveloppe spectrale de l'audio que l'on multiplie avec celle du synthé, on fait une IFFT (l'inverse de la FFT) pour retrouver nos échantillons traités, et boum ça re-fait des Chocapic, mais mieux et en moins d'opérations.

C'est cette technique qui est utilisée dans **[vəkəωdər]**, tout comme dans d'autres vocoder comme Vokator de Native Instruments. Si **[vəkəωdər]** constitue, je l'espère, un vocodeur efficace et facile à utiliser, Vokator est un outil bien plus avancé puisqu'il est doté d'un synthétiseur bien plus évolué, ainsi que bien d'autres fonctionnalités.

## *5. En savoir encore plus sur [vəkəωdər]*

### Introduction

Dans cette rubrique nous allons parler de quelques détails techniques de **[vəkəωdər],** afin de mieux comprendre comment il fonctionne, ou de satisfaire l'appétit gourmand des scientifiques avides de détails !
Premier point, la FFT. **[vəkəωdər],** fait parti de la famille des traitements temps-fréquence, ce qui signifie que ce traitement est à la fois fonction du temps et de la fréquence. Pour explication un filtrage passe-bas est un traitement uniquement fréquentiel, puisque le filtre ne varie pas dans le temps. Un effet Delay à l'inverse, est un effet uniquement temporel, puisque l'on ne modifie pas le contenu fréquentiel du son. Vous en déduisez donc qu'un traitement temps-fréquence est une sorte de « filtrage » par un filtre qui bouge dans le temps. Dans le cas d'un vocoder, le « filtre » est l'audio, et il varie dans le temps tout simplement parce que vous dite « Je suis un robot » dans le micro. Nous allons donc « filtrer » le signal du synthé (qui lui aussi varie dans le temps) par le signal de la voix.
Un traitement temps-fréquence va donc faire intervenir deux paramètres essentiels : la résolution fréquentielle (celle de notre FFT), et la résolution temporelle (tout les combien de temps va-t-on appliquer cette FFT).
Plusieurs options sont possibles pour augmenter ces deux paramètres.

### Augmenter la résolution fréquentielle

Pour augmenter la résolution fréquentielle, on peut utiliser la technique du Zero Padding. Cela consiste à dire : je veux faire une FFT sur 1024 points, mais je vais en fait en faire une sur 2048 (ou plus…). On créé donc un bloc de 2048 points dont on rempli la première moitié avec les 1024 échantillons à traiter et la deuxième

moitié avec des zéros. On obtient donc 2048 « points de mesure » de notre spectre pour seulement 1024 points. On a donc doublé notre résolution fréquentielle.

## Augmenter la résolution temporelle

Pour augmenter la résolution temporelle, on peut utiliser la technique du Overlap Add. Cela consiste à dire : je veux faire une FFT sur 1024 points, mais pas tous les 1024 échantillons, tous les 512 par exemple. Nous faisons donc se chevaucher de moitié les blocs FFT, et améliorons donc notre résolution temporelle par deux puisque nous « mesurons » le spectre deux fois plus souvent. Le nombre d'échantillons entre deux FFT est appelé Hop.
Pour des raisons théorique de traitement du signal, chaque bloc FFT doit être pondéré par une fenêtre, afin d'annuler les effets du chevauchement.
Allez, un schéma pour les visuels :

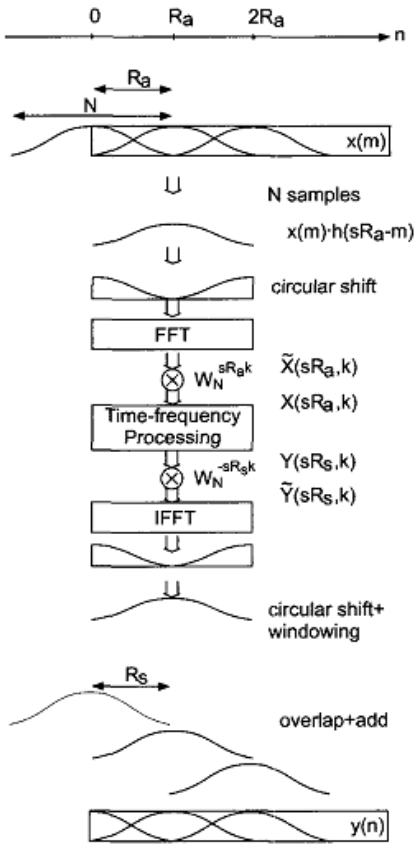


**Figure 2 - Overlap Add**

## Trouver un compromis musical

De ce que j'ai raconté dans le paragraphe précédent on pourrait déduire : « Ah oui, donc l'idéal ce serait de faire une FFT de 65536 points avec un zero padding de 65536 points, et ce à chaque échantillon ». Euh…théoriquement oui, mais en fait non (ne serait-ce que pour la quantité de calcul que devra fournir notre machine).
Contrairement à un traitement audio plus standard, les effets pour la musique ont cela de particulier que la première chose qui importe est la musicalité. Et dans le cas d'un traitement temps-fréquence, deux paramètres importants de la musicalité sont mis à mal.

La latence :

Premier paramètre, la latence. Inopportun lors d'un traitement « offline », un faible voir très faible temps de latence est cependant nécessaire dans un contexte temps réel, tout simplement pour jouer « live » avec un vocoder. Or un temps de latence est indissociable de l'usage d'une FFT. Pourquoi ? Pour travailler en temps réel, un logiciel audio envoie les échantillons reçus (de la carte son par exemple) au plugin par petits paquets, et nous devons attendre ces petits paquets…
Reprenons le schéma ci-dessus pour en faire un exemple temps réel. Plaçons nous à un instant quelconque : nous avons connaissance d'une certaine portion passée du signal, par exemple les 1024 derniers échantillons. Nous pouvons les traiter. Mais pour effectuer la prochaine FFT, nous devons attendre qu'un certain nombre d'échantillons nous soit fourni : ce nombre est par ailleurs égal au Hop.
Si cela ne pose pas de problème à un instant quelconque, cela pose un problème au début : eh oui, nous n'avons pas le passé du signal. Nous pouvons remplacer une partie de ce passé par (1024-Hop) zéros, de

manière à faire le premier traitement dès que nous avons reçus Hop échantillons. Mais le mal est fait ! Nous avons du attendre avant d'avoir ces Hop échantillons. Combien de temps : Hop divisé par la fréquence d'échantillonnage. Un exemple : Hop=1024, F=44100, latence = 23 ms… ce qui est beaucoup trop long pour jouer « live ».

Le comportement de l'instrument :

Lorsque l'on développe un effet audio ou un synthétiseur, c'est un instrument de musique que l'on créé : il aura ses spécificités timbrales, sa dynamique …bref…un son à lui. Et il faut en tenir compte. Utiliser les propriétés de la FFT dans un vocoder permet d'avoir une haute résolution spectrale avec peu d'opérations; parfois même une trop haute définition ! A tel point que quand la résolution est trop élevée, le Vocoder est tellement transparent que son effet n'est plus intéressant : on retrouve presque le signal original de la voix avec peu de traces du synthé. De la même manière, une trop haute résolution temporelle conduit à un comportement trop « lisse » du vocodeur : tout est tellement bien défini que les variations sont trop naturelles, pas assez « hachées » pour en faire un instrument au son dynamique !

Mes réglages pour **[vəkəωdər]** ont donc été les suivants : une FFT de 128 à 1024 points, pour un Hop de 32 à 128 échantillons.
Pour des réglages standards (FFT=512, HOP=128), le temps de latence est de 128/44100=3ms, à sommer avec la latence de votre système carte son. Une FFT de 512 points m'a paru suffisante pour avoir une définition spectrale équilibrée (1024 n'est pas mal non plus, remarque… !). J'utilise une fenêtre de Hanning comme fenêtre pour la FFT car c'est celle qui m'a semblé la plus musicale.

# Conclusion

Merci à vous qui m'avez lu jusqu'au bout ! J'espère avoir été assez clair dans mon explication. A vous de me le dire sur le forum Audiofanzine ou KVRAudio.
Si tout cela est très intéressant, il ne faut néanmoins pas oublier l'objectif premier, la zic, la zic, la zic !!!!
Pierre Fournier - *PiF*

Quelques références pour le développement de **[vəkəωdər]** et pour l'élaboration de ce document :

VST SDK – VST Software Development Kit – Steinberg. Disponible sur www.steinberg.de
DAFx – Udo Zölzer – Digital Audio Effects- Ed. John Wiley and Sons- 2002 (illustration)
FFTW – librairie utilisée pour la FFT – www.fftw.org gratuite pour des freewares, payante autrement
Un tutoriel sur la programmation de synthés polyphoniques, par ndc plugs.