LOUIS BEGHIN, SHYLI KONE, LUCAS SOFIANOS

SUPERCOLLIDER, FILTRES ET EFFETS

Supercollider est un logiciel audio open source qui permet aux utilisateurs de créer des sons et des sons complexes à l'aide de filtres et d'effets. Les filtres Supercollider sont des éléments clés de leur moteur de synthèse, et ils peuvent être utilisés pour modifier les caractéristiques d'un son, pour créer des effets ou pour créer des sons entièrement nouveaux.

Les principaux types de filtres Supercollider sont les filtres passe-bas, les filtres passe-haut, les filtres passe-bande et les filtres résonnants. Les filtres passe-bas sont utilisés pour enlever les fréquences les plus hautes d'un signal, tandis que les filtres passe-haut sont utilisés pour enlever les fréquences les plus basses. Les filtres passe-bande sont utilisés pour enlever des fréquences spécifiques, tandis que les filtres résonnants sont utilisés pour amplifier des fréquences spécifiques. Les filtres Supercollider peuvent également être utilisés pour créer des effets audio tels que les réverbérations, les distorsions et les delays.

LE FILTRE À PEIGNE:

Un filtre à peigne (combing filter en anglais) est un type de filtre utilisé pour créer des effets de réverbération et de résonance en utilisant des délais multiples. SuperCollider inclut plusieurs implémentations de filtres à peigne, comme CombC, CombL, CombN, qui sont tous basés sur la même idée de base mais utilisent des algorithmes de calcul différents pour obtenir des résultats légèrement différents.

Les principaux paramètres d'un filtre à peigne sont delaytime (délai) et decaytime (durée de décroissance). La delaytime détermine la durée de délai entre chaque répétition de la réverbération, tandis que la decaytime détermine combien de temps il faut pour que la réverbération s'estompe. Plus la delaytime est courte, plus la réverbération sera dense et proche de la source sonore originale, tandis qu'une delaytime plus longue donnera une réverbération plus espacée et échoique. Plus la decaytime est courte, plus la réverbération sera courte et abrupte, tandis qu'une decaytime plus longue donnera une réverbération plus longue et en douceur.

Voici un exemple de code SuperCollider pour utiliser un filtre à peigne CombC pour créer un effet de réverbération :

```
// création d'un générateur de bruit blanc
b = WhiteNoise.ar;
// création d'un filtre à peigne avec un délai de 0.2 secondes et une durée de décroissance de 1 seconde
comb = CombC.ar(b, 0.2, 1);
// envoi du signal filtré vers les sorties audio
Out.ar(0, comb);
```

Mathématiquement, le fonctionnement d'un filtre à peigne peut être décrit comme suit :

- On commence par un signal d'entrée x(n)
- On définit une réponse impulsionnelle h(n) qui décrit le délai et la réverbération souhaités
- On convole x(n) avec h(n) pour obtenir le signal de sortie y(n) :

$$y(n) = x(n) * h(n) = \Sigma\ x(k) * h(n-k)$$

La réponse impulsionnelle h(n) est généralement définie comme un ensemble de délais multiples avec des amplitudes décroissantes pour simuler la réverbération dans un espace réel. Par exemple, h(n) pourrait être défini comme un ensemble de délais équidistants avec des amplitudes décroissantes exponentiellement, ou un délai unique avec une amplitude décroissante exponentiellement.

Les paramètres de délai et de décroissance sont utilisés pour définir la réponse impulsionnelle h(n). La delaytime détermine la durée de délai entre chaque répétition de la réverbération, tandis que la decaytime détermine combien de temps il faut pour que la réverbération s'estompe. Plus la delaytime est courte, plus la réverbération sera dense et proche de la source sonore originale, tandis qu'une delaytime plus longue donnera une réverbération plus espacée et échoique. Plus la decaytime est courte, plus la réverbération sera courte et abrupte, tandis qu'une decaytime plus longue donnera une réverbération plus longue et en douceur.

Il est important de noter que la convolution est un processus très gourmand en ressources informatiques, donc les filtres à peigne sont généralement implémentés

en utilisant des techniques d'échantillonnage pour réduire la complexité computationnelle.

```
// création d'un générateur de bruit blanc
b = WhiteNoise.ar;

// création d'un filtre à peigne avec un délai de 0.1 secondes et un taux de décroissance de 0.5
comb = CombN.ar(b, 0.1, 0.5);

// création d'un filtre dent de scie avec une fréquence de coupure de 1000 Hz et un Q de 2
saw = RLPF.ar(comb, 1000, 2);

// envoi du signal filtré vers les sorties audio
Out.ar(0, saw);
```

Dans cet exemple, on utilise un générateur de bruit blanc comme source sonore. On utilise d'abord un filtreCombN pour ajouter un délai de 0.1 secondes et un taux de décroissance de 0.5 au bruit. Ensuite on utilise un filtre RLPF pour ajouter un effet de dent de scie avec une fréquence de coupure de 1000 Hz et un Q de 2. Le signal filtré est ensuite envoyé vers les sorties audio.

DECAYTIME ET FEEDBACK:

Le decaytime d'un filtre à peigne est généralement calculé à partir de la delaytime et du feedback (aussi appelé coefficient de rétroaction).

Le feedback est un paramètre qui détermine la proportion de la sortie qui est renvoyée à l'entrée. Il est généralement exprimé en pourcentage et peut varier de 0 à 1. Plus le feedback est élevé, plus la réverbération sera longue et intense.

La decaytime est généralement calculée en utilisant l'équation suivante :

Decaytime (ms) = -(Delaytime (ms) / ln(1 - Feedback))

Dans cette équation, le decaytime est calculé en prenant la durée de délai et en la divisant par ln (1-feedback). Ln est la fonction logarithme népérien, qui est utilisée pour obtenir la durée de la décroissance de la réverbération.

Il est important de noter que cette équation ne tient pas compte de tous les facteurs qui peuvent influencer le decaytime d'un filtre à peigne, tels que la forme de la réponse impulsionnelle, les paramètres de la pièce virtuelle, etc. Il peut y avoir des variations dans les calculs de decaytime en fonction de l'implémentation du filtre à peigne.

Il est important de noter aussi que cette équation est valable pour des feedbacks faibles, pour des feedbacks plus élevés il peut y avoir des divergences. Il est donc important de tester différentes valeurs pour les paramètres delaytime et feedback pour trouver les meilleures valeurs pour votre projet.

LISTE DES FILTRES SUPERCOLLIDER:

La classe de filtre de SuperCollider comprend de nombreuses sous-classes qui permettent de créer des filtres audio différents. Voici une liste non exhaustive des sous-classes de filtre les plus couramment utilisées dans SuperCollider :

- Filtres passe-bas (LowPassFilter, RLPF, RLPF)
- Filtres passe-haut (HighPassFilter, RHPF, RHPF)
- Filtres bande-passante (BandPassFilter, BRPF)
- Filtres bande-rejet (BandStopFilter, BSF)
- Filtres formant (Formlet, Formlet2)
- Filtres à résonance (Resonz, ResonZ)
- Filtres à allpass (AllPassC, AllPassL, AllPassN)
- Filtres à délai (DelayL, DelayC, DelayN)
- Filtres de comb (CombC, CombL, CombN)
- Filtres de lissage (OnePole, TwoPole, OneZero, TwoZero)
- Filtres de déviation (Hilbert, Differentiator, Integrator)
- Filtres de saturation (SoftClipper, HardClipper)
- Filtres de Distortion (Distortion, Chebyshev, Waveshaper)
- Filtres de Modulation (Lag, Slew, LagUD)
- Filtres d'EQ (EQ, EQGraphic, EQSpec)

Nous allons vous produire des définitions et exemples de codes pour quelques filtres:

FILTRE PASSE-BAS:

Le filtre passe-bas de SuperCollider est un type de filtre qui permet de passer les fréquences plus basses d'un signal tout en atténuant les fréquences plus élevées. Il est généralement utilisé pour créer des effets de "basse" dans la musique électronique ou pour enlever les bruits indésirables dans un signal audio.

Il existe plusieurs implémentations différentes de filtres passe-bas dans SuperCollider, chacune utilisant des algorithmes de calcul différents pour obtenir des résultats légèrement différents. L'un des plus couramment utilisé est le filtre RLPF (passe-bas résonant).

Mathématiquement, le filtre RLPF est basé sur un filtre passe-bas de second ordre, qui est défini par les équations suivantes :

y(n) = a0 * x(n) + a1 * x(n-1) + a2 * x(n-2) - b1 * y(n-1) - b2 * y(n-2)

où x(n) est le signal d'entrée, y(n) est le signal de sortie, a0, a1, a2 sont les coefficients d'atténuation et b1, b2 sont les coefficients de rétroaction. Ces coefficients sont calculés en fonction de la fréquence de coupure, de la fréquence de résonance et de la vitesse d'échantillonnage du signal.

Voici un exemple de code SuperCollider pour utiliser un filtre RLPF :

```
// création d'un générateur de bruit blanc
b = WhiteNoise.ar;
// création d'un filtre passe-bas résonant avec une fréquence de coupure de 400 Hz et un taux de résonance de 0.2
lpf = RLPF.ar(b, 400, 0.2);
// envoi du signal filtré vers les sorties audio
Out.ar(0, lpf);
```

Dans cet exemple, on utilise un générateur de bruit blanc comme source sonore, on utilise un filtre RLPF (passe-bas résonant) pour filtrer le bruit avec une fréquence de coupure de 400 Hz et un taux de résonance de 0.2. Enfin, il envoie le signal filtré vers la sortie audio. Il est important de noter que les valeurs de fréquence de coupure et de taux de résonance peuvent être modifiées pour obtenir des résultats différents.

FILTRE PASSE-HAUT:

Le filtre passe-haut de SuperCollider est un type de filtre qui permet de passer les fréquences plus hautes d'un signal tout en atténuant les fréquences plus basses. Il est généralement utilisé pour créer des effets de "coupure" dans la musique électronique ou pour enlever les bruits indésirables dans un signal audio.

Il existe plusieurs implémentations différentes de filtres passe-haut dans SuperCollider, chacune utilisant des algorithmes de calcul différents pour obtenir des résultats légèrement différents. L'un des plus couramment utilisé est le filtre RHPF (passe-haut résonant).

Mathématiquement, le filtre RHPF est basé sur un filtre passe-haut de second ordre, qui est défini par les équations suivantes :

$$y(n) = a0 * x(n) + a1 * x(n-1) + a2 * x(n-2) - b1 * y(n-1) - b2 * y(n-2)$$

où x(n) est le signal d'entrée, y(n) est le signal de sortie, a0, a1, a2 sont les coefficients d'atténuation et b1, b2 sont les coefficients de rétroaction. Ces coefficients sont calculés en fonction de la fréquence de coupure, de la fréquence de résonance et de la vitesse d'échantillonnage du signal.

Voici un exemple de code SuperCollider pour utiliser un filtre RHPF :

```
// création d'un générateur de bruit blanc
b = WhiteNoise.ar;
// création d'un filtre passe-haut résonant avec une fréquence de coupure de 400 Hz et un taux de résonance de 0.2
hpf = RHPF.ar
```

FILTRE PASSE-BANDE:

Le filtre passe-bande de SuperCollider est un type de filtre qui permet de passer une bande de fréquences spécifique d'un signal tout en atténuant les fréquences qui sont en dehors de cette bande. Il est généralement utilisé pour isoler ou mettre en évidence une plage de fréquences spécifique dans un signal audio.

L'un des plus couramment utilisé est le filtre BRPF (passe-bande résonant), il est basé sur un filtre passe-bande de second ordre qui est défini par les équations suivantes :

y(n) = a0 * x(n) + a1 * x(n-1) + a2 * x(n-2) - b1 * y(n-1) - b2 * y(n-2)

où x(n) est le signal d'entrée, y(n) est le signal de sortie, a0, a1, a2 sont les coefficients d'atténuation et b1, b2 sont les coefficients de rétroaction. Ces coefficients sont calculés en fonction de la fréquence de coupure, de la fréquence de résonance et de la vitesse d'échantillonnage du signal.

Voici un exemple de code SuperCollider pour utiliser un filtre passe-bande :

```
// création d'un générateur de bruit blanc
b = WhiteNoise.ar;
// création d'un filtre passe-bande résonant avec une fréquence de coupure de
400 Hz et une largeur de bande de 100 Hz et un taux de résonance de 0.2
bpf = BRPF.ar(b, 400, 100, 0.2);
// envoi du signal filtré vers les sorties audio
Out.ar(0, bpf);
```

Dans cet exemple, on utilise un générateur de bruit blanc comme source sonore, on utilise un filtre BRPF pour filtrer le bruit avec une fréquence de coupure de 400 Hz, une largeur de bande de 100 Hz et un taux de résonance de 0.2. Enfin, il envoie le signal filtré vers la sortie audio.

FILTRE PASSE-TOUT:

Le filtre passe-tout de SuperCollider est un type de filtre qui permet de passer toutes les fréquences d'un signal sans atténuation. Il est généralement utilisé pour ajouter un gain à un signal ou pour utiliser un signal comme source de modulation pour un autre filtre.

Il existe plusieurs implémentations différentes de filtres passe-tout dans SuperCollider, mais l'une des plus couramment utilisées est la classe LPF (LowPassFilter), qui est un filtre passe-bas qui passe toutes les fréquences en dessous d'une certaine fréquence de coupure spécifiée.

Mathématiquement, un filtre passe-tout peut être implémenté en utilisant une fonction de transfert qui est identique à 1 pour toutes les fréquences. Cela signifie qu'il n'y a aucune atténuation appliquée à aucune fréquence, donc toutes les fréquences passent à travers le filtre inchangées.

Voici un exemple de code SuperCollider pour utiliser un filtre passe-tout :

```
// création d'un générateur de bruit blanc
b = WhiteNoise.ar;
// création d'un filtre passe-tout avec une fréquence de coupure de 0 Hz
passt = LPF.ar(b, 0);
// envoi du signal filtré vers les sorties audio
Out.ar(0, passt);
```

Dans cet exemple, on utilise un générateur de bruit blanc comme source sonore et on utilise un filtre LPF avec une fréquence de coupure de 0 Hz pour filtrer le bruit, cela va créer un filtre passe-tout qui va laisser passer toutes les fréquences. Enfin, il envoie le signal filtré vers la sortie audio. Il est important de noter que la fréquence de coupure peut être modifiée pour obtenir des résultats différents.

FILTRE DELAY:

Le filtre delay de SuperCollider est un type de filtre qui utilise un délai pour créer des effets de répétition ou de réverbération sur un signal audio. Il est généralement utilisé pour ajouter une profondeur ou une ambiance à un son.

Il existe plusieurs implémentations différentes de filtres passe à délai dans SuperCollider, mais l'une des plus couramment utilisées est la classe DelayL (DelayLine), qui est un filtre à délai basé sur un buffer qui stocke un certain nombre d'échantillons et les renvoie à un certain délai spécifié.

Mathématiquement, un filtre passe à délai peut être implémenté en utilisant une fonction de transfert qui est basée sur un délai. Cela signifie qu'il prend une entrée, la stocke dans un buffer pour un certain temps (délai) et la renvoie à la sortie.

Voici un exemple de code SuperCollider pour utiliser un filtre passe à délai :

```
// création d'un générateur de bruit blanc
b = WhiteNoise.ar;
// création d'un filtre passe à délai avec un délai de 0.5 secondes
d = DelayL.ar(b, 0.5, 0.5);
// envoi du signal filtré vers les sorties audio
Out.ar(0, d);
```

FILTRE DE LISSAGE:

Le filtre de lissage de SuperCollider est un type de filtre qui permet de lisser les variations brusques d'un signal en utilisant une technique de moyennage. Il est généralement utilisé pour enlever les bruits parasites ou les variations de bruit dans un signal audio.

Il existe plusieurs implémentations différentes de filtres de lissage dans SuperCollider, mais l'une des plus couramment utilisées est la classe OnePole qui est un filtre de lissage à un pole.

Mathématiquement, un filtre de lissage à un pole est implémenté en utilisant une fonction de transfert qui est basée sur une combinaison de la valeur actuelle d'un signal avec les valeurs précédentes. Cela permet de créer une sorte de moyenne glissante qui suis les variations du signal d'entrée.

Voici un exemple de code SuperCollider pour utiliser un filtre de lissage

```
// création d'un générateur de bruit blanc
b = WhiteNoise.ar;
// création d'un filtre de lissage
smooth = OnePole.ar(b, 0.1);
// envoi du signal filtré vers les sorties audio
Out.ar(0, smooth);
```

Dans cet exemple, on utilise un générateur de bruit blanc comme source sonore et on utilise un filtre ONEPOLE pour lisser les variations du bruit en utilisant un coefficient de lissage de 0.1

FILTRE A DISTORSION:

Le filtre de distorsion de SuperCollider est un type de filtre qui permet de modifier la forme d'onde d'un signal en utilisant une technique de non-linéarité. Il est généralement utilisé pour ajouter de la saturation, de l'harmonisation, de l'overdrive et d'autres effets de distorsion à un signal audio.

Il existe plusieurs implémentations différentes de filtres de distorsion dans SuperCollider, mais l'une des plus couramment utilisées est la classe Distortion qui est un filtre de distorsion générique.

Mathématiquement, un filtre de distorsion est implémenté en utilisant une fonction de transfert non-linéaire qui modifie la forme d'onde d'un signal en fonction d'un certain paramètre de distorsion spécifié. Cela peut être fait en utilisant des fonctions telles que des polynômes, des exponentials ou des fonctions tanh pour changer la forme d'onde du signal.

Voici un exemple de code SuperCollider pour utiliser un filtre de distorsion :

```
// création d'un générateur de bruit blanc
b = WhiteNoise.ar;
// création d'un filtre de distorsion avec un paramètre de distorsion de 0.5
dist = Distortion.ar(b, 0.5);
// envoi du signal filtré vers les sorties audio
Out.ar(0, dist);
```

Dans cet exemple, on utilise un générateur de bruit blanc comme source sonore et on utilise un filtre Distorsion pour ajouter un effet de distorsion au bruit en utilisant un paramètre de distorsion de 0.5. Ce paramètre de distorsion peut être modifié pour obtenir des résultats différents.