Rauscherzeugung mit FMS

Daniel Grün

FMS (`http://sf.net/projects/fmsynth`) ist ein virtueller Synthesizer mit dem man außer periodischen Klängen auch verschiedenartiges Rauschen erzeugen kann. Das folgende Dokument soll einen Überblick über die wichtigsten Rauscherzeugungsmethoden und ihre Implementierung mit Hilfe der FMS-API geben. Die Codebeispiele können folgendermaßen in C++-Programme eingebaut werden:

```
#include ¨include/fmplayer.h¨

int main() {
FMPlayer p;
// hier Code-Beispiel einfügen
p.play();
return 0;
}
```

Beim Compilieren muss mit fmplayer.o, fmwav.o und fmval.o gelinkt werden.

1 Grundlagen

Am einfachsten wird ein Rauschen erzeugt, indem man Zufallswerte nacheinander über die Soundkarte ausgibt. Es kann dabei eine Zufallsfunktion verwendet werden, bei der alle Ergebnisse in einem vorgegebenen Bereich gleich wahrscheinlich sind, aber auch eine beliebige andere Verteilung der relativen Häufigkeiten erzielt werden. Eine Gauss-Verteilung der Zufallswerte erreicht man mit folgendem Codebeispiel (vorausgesetzt `sounds/gauss` ist der korrekte Pfad zu einer entsprechenden FMS-Sounddatei):

```
p.sound->setFilename(¨sounds/gauss¨);
p.sound->setFilemode(Rand);
```

Der Klang des ausgegebenen Rauschens ist weniger abhängig von der Verteilungskurve der Zufallswerte als von der Samplingrate. Ist diese höher, klingt das Rauschen heller. Um diesen Effekt zu nutzen, gibt es einen weiteren Rauschmodus, den Noise-Mode. Hier wird der ausgegebene Wert durch eine normale, also gleichmäßig verteilte, Zufallsfunktion ermittelt. Wie oft der gleiche Wert nacheinander ausgegeben wird, bestimmt die beliebig verteilte Zufallsfunktion.

```
p.sound->setFilename(¨sounds/gauss¨);
p.sound->setFilemode(Noise);
p.sound->setMLen(5);
```

Durch `setMLen(5)` wird die mittlere Anzahl der nacheinander ausgegebenen gleichen Werte auf 5 gesetzt. Durch die Form der verwendeten Gauss-Kurve sind somit Werte zwischen 1 und 10 möglich, am häufigsten wird ein Soundwert allerdings 4-6 mal hintereinander ausgegeben werden und nur seltener 1-3 oder 7-10 mal. Setzt man mlen kleiner, etwa auf 2, so klingt das Rauschen heller und damit zischender. Bei Werten um 30 klingt es eher wie eine startende Rakete.

2 weitere Möglichkeiten

Auch die Kombination beider Techniken, also die Verwendung von Verteilungskurven für den eigentlichen Soundwert und die Anzahl der nacheinander ausgegebenen gleichen Werte ist möglich:

```
p.sound->setFilename(¨sounds/gauss¨);
p.sound->setFilemode(RandNoise);
p.sound->setRNF();
p.sound->rnf->setFilename(¨sounds/tri¨);
```

Im Modus RandNoise ist dabei `rnf` die Verteilungskurve für die Haltedauer, bei NoiseRand für den eigentlichen Soundwert. Wird auf die Angabe von `rnf` verzichtet, so wird die soundeigene Verteilungskurve für beide Zwecke verwendet.

Die Funktion `setMLM` ermöglicht die Modulation des mlen-Wertes:

```
p.sound->setFilename(¨sounds/gauss¨);
p.sound->setFilemode(Noise);
p.sound->setMLM();
p.sound->mlm->setFilename(¨sounds/saw¨);
p.sound->mlm->setMin(1);
p.sound->mlm->setMax(5);
```

Das Beispiel ergibt einen Rauschverlauf von hell nach dumpf.

2.1 zufällige Frequenzmodulation

FMS bietet die Möglichkeit der Frequenzmodulation. Wählt man anstatt einer normalen Funktion eine der möglichen Rausch-Modes zur Frequenzmodulation, so erhält man ab einer gewissen Modulationsgeschwindigkeit den akustischen Eindruck eines Rauschens. Je nach Bandbreite der Frequenzmodulation ist auch das Rauschen schmal- oder breitbandiger. Durch Veränderung der Werte von mlen und den Minimal- bzw. Maximalfrequenzen, aber auch durch Modifikationen an der eigentlichen Soundfunktion lassen sich vielfältige Effekte erzielen.

```
p.sound->setFilename(¨sounds/saw¨);
p.sound->setFFreq();
```

```
p.sound->ffreq->setFilename(¨sounds/gauss¨);
p.sound->ffreq->setFilemode(Noise);
p.sound->ffreq->setMLen(1);
p.sound->ffreq->setMin(300);
p.sound->ffreq->setMax(600);
```

Der ausgegebene Klang erinnert an eine Biene, durch Steigerung von mlen wird sie zum Schwarm, durch Erhöhung der Maximalfrequenz klingen die Bienen aggressiver, bis sie schließlich zum nicht mehr zu identifizierenden Zischen werden. Der Übergang von wahrnehmbarer Frequenzmodulation zum Rauschen macht das folgende Beispiel deutlich:

```
p.sound->setFilename(¨sounds/sin¨);
p.sound->setFFreq();
p.sound->ffreq->setMin(0);
p.sound->ffreq->setMax(3000);
p.sound->ffreq->setFilename(¨sounds/sin¨);
p.sound->ffreq->setFilemode(Noise);
p.sound->ffreq->setMLM();
p.sound->ffreq->mlm->setFilename(¨sounds/flatgauss2¨);
p.sound->ffreq->mlm->setMin(1);
p.sound->ffreq->mlm->setMax(1000);
```

Bei einer mlen der Frequenzmodulation von 1000, also wenn die Frequenz durchschnittlich 1000 Werte lang gehalten wird, sind die Einzeltöne hörbar, am Minimalpunkt der mlen-Modulation, wenn nach jedem Wert die Frequenz neu bestimmt wird, ist nur noch ein Zischen zu hören.

2.2 zufällige Amplitudenmodulation

Auf die Gleiche Art und Weise lässt sich auch die Amplitude eines Klanges zufällig modulieren:

```
p.sound->setFilename(¨sounds/sinpart¨);
p.sound->setFreq(300);
p.sound->setHVol();
p.sound->hvol->setFilename(¨sounds/gauss¨);
p.sound->hvol->setFilemode(Noise);
p.sound->hvol->setMode(AmpMod);
p.sound->hvol->setMLen(1);
```

Das Ergebnis klingt wie ein undefiniertes, aber sehr störendes Haushaltsgerät. Die Zeile `p.sound->hvol->setMode(AmpMod)` im Gegensatz zum Standartmodus `Envelope` ist zur Unterdrückung von Knackgeräuschen bei Amplitudenmodulation mit FMS allgemein sehr empfehlenswert.

Weitere Experimente, wie die Verschachtelung der Modulationen (also z.B. die Modulation der Frequenz der Frequenzmodulation) oder Amplitudenmodulationen sowie die Modulation von mlen bleiben dem Leser überlassen.