CSOUND

Programmiersprache für Audio

Alan Wünsche

Tonseminar Wintersemester 2022/2023

Inhaltsverzeichnis

Was ist CSound – Vorteile und Nachteile

CSound ist eine Programmiersprache, um Audio zu programmieren. Mit Ihr kann man Signale mit beliebigen Wellenformen erzeugen, Rauschen generieren oder einkommendes Audio beliebig bearbeiten und ausgeben. CSound wurde 1985 am MIT von Barry Vercoe entwickelt.
Auch wenn CSound immer noch weiterentwickelt wird und es mit CSoundIDE einen neuen und spannenden Impuls bekommt, kann man sagen, dass es einen alten Ansatz verfolgt, der für Musiker heutzutage nicht so attraktiv ist wie aktuelle Alternativen.

Ein Beispiel hierfür wäre Max for Live, welches weniger Textbasiert, mehr graphisch und damit auch anwenderfreundlicher ist. Dafür ist Max for Live nicht mit allen DAWs kompatibel, während man jeden CSound Code mithilfe von Cabbage zu einem VST3 Plugin umwandeln kann. CSound läuft außerdem auf jedem Betriebssystem u.A. auch auf Android und kann in anderen Programmiersprachen wie Python, Lua, C oder Java aufgerufen und integriert werden.

Arbeitsmittel

Um mit CSound programmieren zu können benötigt man einen PC oder Laptop. Es ist hierbei egal ob man ein Windows oder ein Apple Betriebssystem benutzt. Damit der Computer den eingegebenen Text in Maschinencode und somit Audio umwandeln kann benötigt man den CSound Compiler den man auf der Website csound.com kostenlos herunterladen kann.
Um Code zu schreiben kann man jeden beliebigen Texteditor verwenden. Den geschriebenen Code muss man anschließend als .csd Datei speichern. Es empfiehlt sich einen Editor zu benutzen, der die Syntax verstehen und highlighten kann. Ein Beispiel hierfür ist der Texteditor Sublime, der bei eingegebenen Befehlen Erklärungen, erforderliche Argumente und eine Verlinkung zum Canonical CSound Reference Manual anzeigt. Sublime liefert außerdem die Möglichkeit den Code im Editor auszuführen. Benutzt man einen Editor, der diese Möglichkeit nicht bietet, muss man den Code über die CSound Konsole ausführen, was aufwändiger ist.

Canonical CSound Reference Manual

Das Canonical CSound Reference Manual ist ein hilfreiches Mittel bei der Programmierung in CSound. Es ist eine Website, die nahezu alle Befehle und vorgefertigten Klassen beinhaltet, die es in CSound gibt. Zu jedem Befehl und zu jeder Klasse liefert es Erklärungen und Beispiele. Es wird auch heutzutage immer noch mithilfe der CSound Community aktualisiert und von Andrés Cabrera gepflegt.

Grundsätzliche Struktur

Öffnet man eine bestehende CSound Datei in einem Texteditor ähnelt diese auf dem ersten Blick einem HTML oder XML-File, was daran liegt, dass die Bestandteile des Codes in sogenannte Brackets unterteilt sind.

Zwischen `<CsoundSynthesizer>` und `</CsoundSynthesizer>` befindet sich der gesamte Code, der sich auf CSound bezieht. Im Normalfall sind diese Brackets das erste und das letzte Element in einer CSound Datei.

Zwischen `<CsOptions>` und `</CsOptions>` befinden sich die Informationen, die CSound benötigt um mit angeschlossener Hardware wie Soundkarte oder MIDI-Keyboard zu interagieren.
Im Folgenden Beispiel setzt man `odac` (Output Digital Analog Converter) auf `0`. Dies befiehlt CSound die Soundkarte auszuwählen, die im Microsoft Soundmapper ausgewählt ist.
`M0` befielt das erste angeschlossene MIDI-Keyboard als Input zu wählen.
`b128` setzt die Buffersize auf 128 Samples.
Beispiel:

```
<CsOptions>
     -odac0 -M0 -b128
</CsOptions>
```

Zwischen `<CsInstruments>` und `</CsInstruments>` befinden sich die Definitionen für Instrumente. Zudem teilt man CSound mit, mit welcher Samplerate die Instrumente spielen sollen, wie viele Channels benutzt werden und welche Zahl die größtmögliche Aussteuerung bestimmt.

Beispiel:

```
<CsInstruments>

     0dbfs     =     1
     sr        =     44100
     ksmps     =     32
     nchnls    =     2
```

Daraufhin werden globale Variablen mit dem Wert 0 initiiert und Instrumente definiert. Ein Instrument kann ein audio-generierendes oder audio-veränderndes Instrument sein (bspw. Reverb, Delay, Chorus etc.)

Eine Definition für ein Instrument beginnt stehts mit dem Befehl `instr` und einer folgenden Zahl und enden stets mit dem Befehl `endin`.

Beispiel:

```
instr 1

     aOsc           oscil          0.7, 440
     outs           aOsc, aOsc
```

```
endin

</CsInstruments>
```

Dieses Instrument wird nun, wenn es gerufen wird, eine Sinuswelle mit 70% des Maximalpegels und einer Frequenz von 440Hz wiedergeben. Man beachte, dass es zur Übersteuerung kommt, wenn gleichzeitig dasselbe oder ein weiteres Instrument mit 70% des Maximalpegels wiedergegeben werden würde.

Zwischen `<CsScore>` und `</CsScore>` teilt man CSound mit, welche Instrumente wann, wie lange und gegebenenfalls mit welchen Parametern wiedergegeben werden sollen.

Beispiel:

```
<CsScore>
     i1   0    0.5
</CsScore>
</CsoundSynthesizer>
```

So wird nun Instrument 1 sofort nach Ausführung des Codes gerufen und eine halbe Sekunde lang spielen.

Variablen

In CSound gibt es drei verschiedene Arten von Variablen. a, k und i Variablen. Die Arten unterscheiden sich lediglich darin, wie oft sie aktualisiert werden und sind essentiell um Code zu schreiben, der CPU-sparend ist.

- a Variablen: Diese Variablen werden mit jedem Sample aktualisiert, man benutzt Sie also, wenn man Audiosignale berechnet, da bei niedrigerer Abtastrate Artefakte entstehen würden
- k Variablen: Man kann selbst bestimmen, wie oft diese Variablen neu aktualisiert werden. Dies macht man mit dem Befehl `ksmps`. Setzt man diesen Wert beispielsweise auf 30, werden k Variablen jedes dreißigste Sample aktualisiert. Dieser Wert ist global, was bedeutet dass jede k-Variable gleich oft aktualisiert wird. Man benutzt sie, um bestimmte Werte zu verändern, die das Audiosignal beeinflussen. Beispielsweiße die Geschwindigkeit eines Vibratos oder die Cutoff-Frequenz eines Filters.
- i Variablen: Diese Variablen werden lediglich einmal beim Ausführen des Instruments initiiert und können danach nicht mehr verändert werden. Man benutzt i-Variablen bspw. für Frequenz oder Amplitude eines Instruments.

Nützliche Befehle

Möchte man beginnen in CSound zu arbeiten gibt es Befehle und Klassen, die den Start in die Entwicklung erleichtern. Hier eine kleine Auswahl mit Beispielen:

- `madsr`
  - Beinhaltet vier Argumente: Attack, Decay, Sustain, Release. Die Werte können zwischen 0 und 1 liegen.
  - Erstellt einen Envelope
  - Man kann Ihn durch Multiplikation mit dem gewünschten Signal anwenden

```
aDSR            madsr 0.1, 0.7, 0.5, 0.1
aSignal         *=        aDSR
```

- `lfo`
  - Ähnlich zu `oscil`, der einzige Unterschied besteht darin, dass `oscil` bei einer Amplitude von 1 eine Sinuswelle erzeugt, die zwischen -0.5 und 0.5 schwingt. `lfo` hingegen erzeugt bei einer Amplitude von 1 eine Sinuswelle, die zwischen 0 und 1 schwingt. Egal welche Amplitude man wählt: Standardmäßig ist der niedrigste Wert 0.
  - Beinhaltet zwei Argumente: Amplitude und Frequenz

```
aLFO            lfo       1, 1
```

- `rand`
  - Beinhaltet nur ein Argument: Amplitude
  - Erzeugt ein randomisiertes Rauschen
  - In Kombination mit einem Envelope kann man hiermit Percussion, wie beispielsweise eine Snare imitieren

```
aNoise          rand      0.4
```

- `moogladder`
  - Ein klassischer Tiefpassfilter der den des Moogs imitiert.
  - Beinhaltet drei Argumente: Signal, welches gefiltert werden soll, Grenzfrequenz und Resonanz (zwischen 0 und 1)

```
aFilter     moogladder     aSound, 1000, 0.3

            outs      aFilter, aFilter
```

- `reverbsc`
    - Ein 8 delay line stereo Feedback-Delay-Network Reverb
    - Beinhaltet vier Argumente: Einkommendes Signal Links, Einkommendes Signal Rechts, Feedbacklevel („Raumgröße“, zwischen 0 und 1) und Grenzfrequenz des Tiefpassfilters des Reverbs.
    - Im folgenden Beispiel wird das Signal am Ende 20% Wet wiedergegeben.

```
aVerbL, aVerbR      reverbsc       aSignalL, aSignalR,
0.9, 8000

outs 0.8*aSignalL+0.2*aVerbL, 0.8*aSignalR+0.2*aVerbR
```

Dies ist nur ein kleiner Auszug der Befehle und Klassen, zeigt aber, dass man bereits mit wenig Code in der Lage ist Audio zu programmieren. Für weitere Befehle kann man das bereits erwähnte Canonical CSound Reference Manual aufsuchen.

Cabbage

Cabbage ist ein Framework, um Audio-Software zu entwickeln. Der Code basiert auf CSound und hat wenige zusätzliche Befehle, um aus einfachem CSound Code ein eigenständiges Programm oder VST3 Plugin zu erstellen.

Der Texteditor von Cabbage besitzt die oben angesprochene Highlighting Funktion nicht, weshalb es sich empfiehlt, den Code zuerst in Sublime oder einem ähnlichen Texteditor zu entwickeln. Anschließend kann man den Code in den Cabbage Editor einfügen, eine GUI erstellen und den Code um Befehle erweitern, die die Handhabung mit dem GUI ermöglichen.

Möchte man seinen CSound Code um eine GUI mit Cabbage erweitern um ihn als eigenständiges Programm oder VST3 zu verwenden, ergänzt sich der Code um eine weitere Bracket, welche vor <CsSynthesizer> geschrieben wird: Zwischen `<Cabbage>` und `</Cabbage>` befindet sich der Code, der sich auf das GUI bezieht. Dazu gehören verschiedene Arten von Slidern, Bildern sowie Größe und Farbe des Plugins:

Beispiel:

```
<Cabbage> bounds(0, 0, 0, 0)

form caption("Test") size(1200, 700), colour (255, 255,
255, 255) pluginId("def1"), guiMode("queue")

rslider bounds(400, 300, 60, 60) channel("VibratoSpeed")
range(0, 10, 0.1, 1, 0.001) colour(0, 0, 0, 0)
text("Speed") textColour(0, 0, 0, 255) trackerColour(255,
231, 0, 255)

</Cabbage>
```

Das Fenster des Programms trägt den Titel „Test“ und ist 1200x700 Pixel groß. Es besitzt einen Rotations-Slider auf der Stelle 400, 300, ist 60 Pixel breit wie lang und trägt die Beschreibung „Speed“. Der Channel unter dem seine Werte später aufgerufen werden können trägt den Namen „VibratoSpeed“. Er kann Werte zwischen 0 und 10 annehmen und ist auf 3 Nachkommastellen genau.

Möchte man den Wert im späteren CSound Code aufrufen und verwenden benutzt man:

```
kVibratoSpeed               cabbageGetValue ("VibratoSpeed")
```

Möchte man das Programm als VST3 in seiner DAW verwenden kann man es in Cabbage als VST3 exportieren. Zu der VST3-Plugin Datei erhält man eine .csd Datei. Beide Dateien müssen mit allen anderen verwendeten Dateien (bspw. Bildern oder Audiodateien) unter Programme > Common Files > VST3 abgelegt werden.
Startet man nun seine DAW neu und scannt die Plugins kann man das Plugin wie alle anderen VST-Plugins finden und benutzen. Außerdem sind in Cabbage alle verwendeten Werte automatisierbar.

Ausblick: CSound Web-IDE

Eine aktuelle Weiterentwicklung von CSound ist die open-source Web-IDE. Ihr Ziel ist die Programmierung von CSound im Browser, ohne zusätzliche Software, zu ermöglichen.

Zudem ist es hier Möglich den Code, den andere User geschrieben haben, einzusehen, zu benutzen und für eigene Projekte zu verwenden oder mit einem anderen User zusammen an einem Projekt weiter zu arbeiten. Diese Möglichkeiten helfen enorm den Wissensaustausch in der Community zu beschleunigen und motiviert die User. Die Website ist ähnlich wie eine Social Media Plattform aufgebaut, mit Profilen und Beiträgen. Die Beiträge sind hier jedoch keine Bilder oder Texte sondern die CSound Projekte an denen der User momentan arbeitet. Man kann anderen Usern folgen und sieht deren Fortschritte im persönlichen Feed.

Die Beiträge werden erst ohne Code und mit einem Play Button angezeigt. Drückt man „Play“ wird der Code ausgeführt und man kann ihn sich anhören. Interessiert man sich für den Code dahinter, kann man den Beitrag anklicken und erhält den gesamten Code, der in diesem Projekt geschrieben wurde.

Auch wenn CSound mittlerweile eine alte Programmiersprache ist, birgt diese Plattform meiner Meinung nach viel Potential, gerade wegen des Austauschs und eines ausgeprägten Community Gefühls.

Die Website befindet sich immer noch in der Entwicklung, funktioniert allerdings bereits ohne größere Probleme.

Links

Csound Homepage

Canonical CSound Reference Manual

Csound Web-IDE

Sublime

Cabbage