Sinus Synthese

Komplexpraktikum Audio

SS 2017

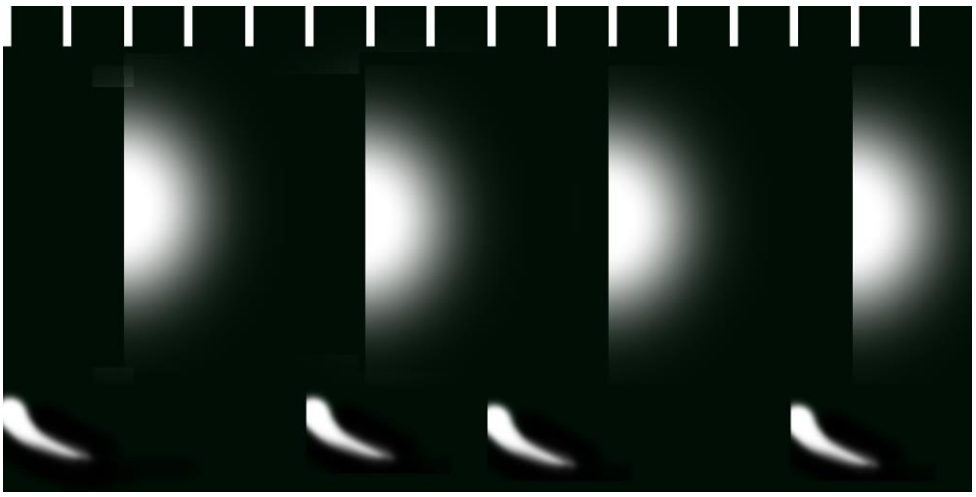
Dokumentation

Bettina Blasberg, Vincent Walter

Aufgabenstellung

Für das Komplexpraktikum haben wird die Aufgabe gestellt bekommen, mittels der Sinus Synthese ein Spektrogramm zu einem Sound zu synthetisieren. Dafür muss ein komplexer Klang erstellt werden, dass aus einer Addition von Sinuswellen unterschiedlicher Frequenzen und Amplituden besteht. Weiterhin sollte auch noch interaktive Elemente bei der Synthetisierung hinzugefügt werden. Wir dachten dabei an Änderungen der Frequenzen und Amplituden. Für die Erstellung dieses Programms haben wir die Bibliothek LibCinder verwendet.

Spektrogramm

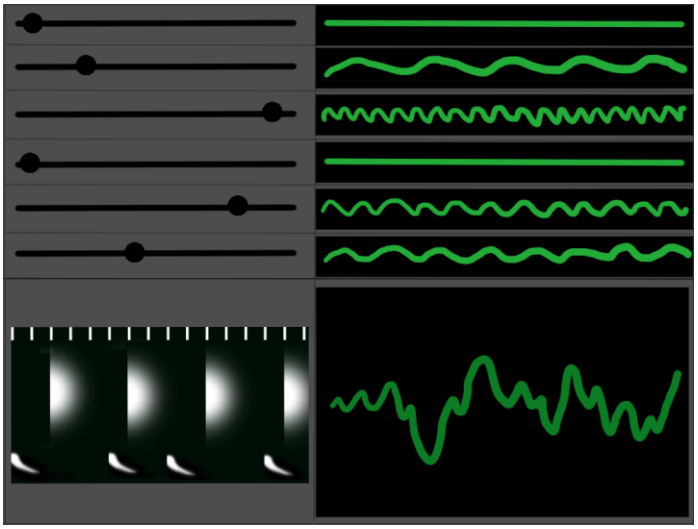


Konzepte

Im Laufe des Praktikums haben wir uns zwei mögliche Konzepte ausgedacht, um Spektrogramme zu bearbeiten.

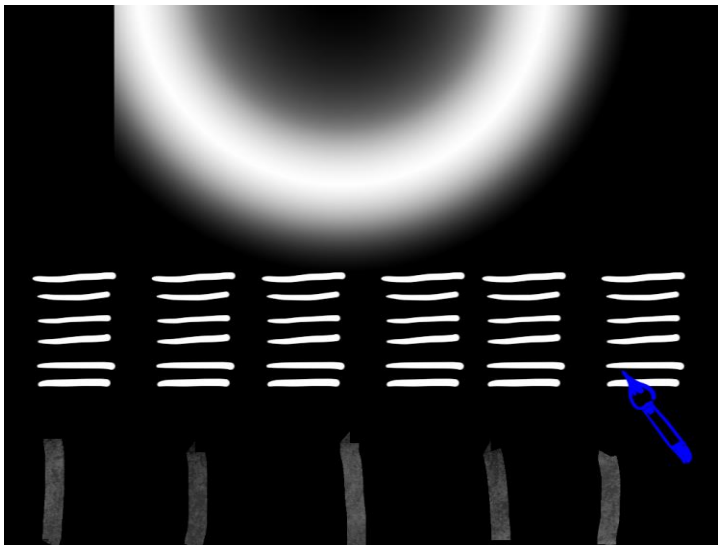
Live Manipulation der Sinus Oszillatoren

In diesem Konzept ist eine UI vorhanden, die in zweite Teilbereiche aufgeteilt ist. Der obere Bereich ist zuständig für die Bestimmung der Frequenzen und zeigt die Oszillatoren an. Dafür sind auf der linken Seite Schieberegler, die erlauben, Frequenzen zu bestimmen. Die rechte Seite zeigt zu jedem Oszillator die entsprechende Sinuskurve erstellt nach vordefinierten Frequenz. Im unteren Bereich ist auf der linke Seite das Spektrogramm zu sehen und auf der rechten Seite die Summer der Sinuswellen. Ein Play-Button ermöglicht, den Sound abzuspielen.



Komponieren durch Malen

In diesem Konzept sollte eine Zeichenoberfläche erstellt werden. Der Nutzer kann dann mit verschieden Pinsel und Graustufen ein Bild malen. Dieses Bild wird anschließend abgetastet und als Sound wiedergegeben.

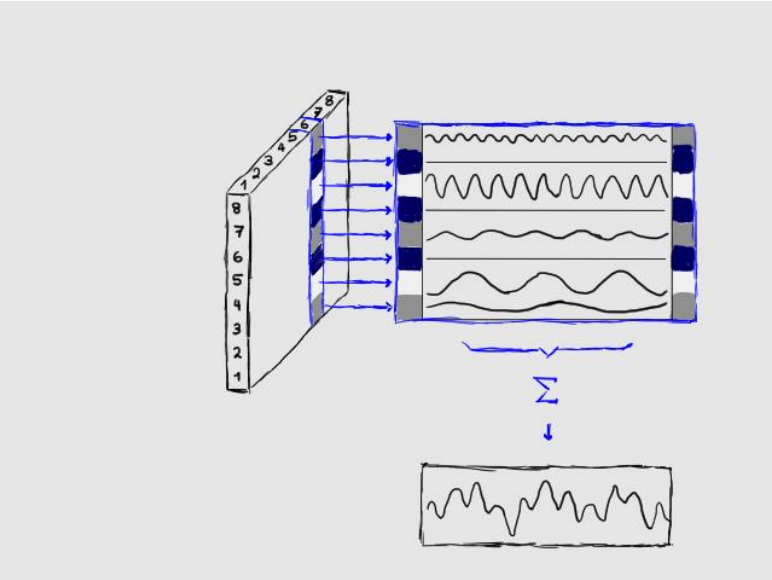


Wir haben uns für das erste Konzept entschieden, da wir uns auf die Abtastung eines vorhandenen Spektrogramms fokussiert haben.

Vorgehen

Um ein Spektrogramm abzutasten und als Ton wiederzugeben, haben wir uns folgende Schritte ausgedacht:

- Bild einlesen
- Bild in Spalten zerlegen
- Sinusoszillatoren für abzutastende Pixel erstellen
- Einzelne Pixel der Spalte abtasten (Amplitude)
    - Schwarz: Amplitude = 0
    - Weiß: Amplitude = 1
- Summe der Pixelspalte = komplexer Klang



LibCinder hat schon vordefinierte Funktionen, um Bilder einzulesen und abzutasten. Pro Zeile wird ein Oszillator generiert, um ein komplexen Klang zu erzeugen. Die Pixel der vorhandenen Bilddatei wird nach ihren Alphawert gefragt, um die Amplitude zu bestimmen. Ein Vektor enthält die Information einer Pixelspalte des Spektrogramms. Wenn diese Vektoren zu Nodes (engl. Knoten) anhand der LibCinder Bibliothek generiert und danach addiert werden, entsteht ein komplexer Klang.

Funktionalität

Das Programm besteht aus vier Anzeigen: Die Definition der Frequenzen, die Anzeige der Sinusschwingungen der Oszillatoren, die Anzeige des abgetasteten Bildes und die Summe die der Sinusschwingungen. Die Anzeige der Definition der Frequenzen hat Links die Beschriftungen und rechts die dazugehörige Zahl. Das Bestimmen passiert, wenn die Zahl mit der Maus geklickt und dann per Tastatur verändert wird. Es besitzt auch einen Plus- und Minus-Symbol, wodurch die Werte Schrittweise addiert oder subtrahiert werden können. Die Anzeige der Sinusschwingungen besteht aus Rechtecken und darin wird die Schwingung angezeigt. Die Reihenfolge der Rechtecke entspricht die der Frequenzwerten. Sobald der Wert eines Oszillators verändert wird, ändert sich auch die entsprechende Schwingung. Unter der Anzeige der Frequenzwerten befindet sich das Bild. Rechts daneben und unter der Anzeige der Schwingungen wird die summierte Anzeige der Sinusschwingungen dargestellt. Wenn die Sinusschwingungen sich ändern, ändert sich die Anzeige der Summe. Wenn auf dem Play-Button gedrückt wird, wird die entsprechende Musik zu der der addierten Sinusschwingung abgespielt.

Probleme

Die Abtastung nach LibCinder ist etwas komplexer, da es nur einen Iterator gibt, der jede Zeile abtastet und nicht jede Spalte. Das konnte aber gelöst werde, indem die abgetasteten Elemente in einem Vektor bestehend aus Vektoren, die abgetastete Pixel als Spalte speichert, eingefügt werden. Es ist zu rechenintensiv und komplex, pro Zeile einen Oszillator zu erzeugen. Deshalb wurden für dieses Konzept nur 20 Oszillatoren generiert. Die zugehörigen Frequenzen zu erzeugen war ein weiteres Problem. Ursache war eine Fehldenken: die Frequenzen einfach nach entsprechenden Zeilen addieren und dann normieren. Das erzeugte zu hohe Frequenzen, die nicht hörbar sind. Deshalb wechselten wird auf den Ansatz, wo ein Frequenzbereich definiert wird. Dieser Frequenzbereich wird logarithmisch berechnet und unterteilt. Dabei mussten wird achten, keine allzu hohen Frequenzwerten einzusetzen.
Nach vielen Prüfen stellte sich auch heraus, dass die Pixel keine Alphawert gespeichert haben. Das konnte gelöst werden, indem die Luminanz anhand der RGB-Werte jedes Pixels ausgerechnet wird. Ein weiteres Problem ist das Abspielen der Spalten. Die Abspielgeschwindigkeit ist viel zu schnell und unsere bisherigen Lösungsansätze haben bei diesem Problem nicht geklappt. Die Lösungsansätze waren folgende:

- einen Thread erzeugen, um den Rechner beim Erzeugen der Töne zum Schlafen zu bringen
- Delay-Node von LibCinder, um das Abspielen der kommenden Spalten zu verzögern
- Apply-Ramp von LibCinder, um das Abspielen auf eine Frequenz und Abspielzeit definieren zu können

Da Vincent und ich mit verschiedenen Ansätzen die Aufgabe bewältigen wollten, hatte jeder von uns eine andere Ausgabe an Tönen. Der Ansatz mit dem Thread hat zwar nicht einwandfrei funktioniert, aber es konnte teilweise den Sound eines Bildes erzeugen. Im jetzigen Projekt ist es auch noch nicht möglich einen Ton zu erzeugen, da der Ansatz mit dem Thread nicht weiterentwickelt wurde. Die Manipulation der Frequenzen und entsprechenden Sinusschwingungen ist auch nicht möglich. LibCinder bietet bei der Angabe der Params eine Callback-Funktion, welche aus einer Lambdafunktion besteht. Dieses Lambda kann jedoch keine Vektoren abfangen oder sie als Parameter einsetzen. Dadurch ist es auch nicht möglich, die Oszillatoren und entsprechenden Schwingungen zu aktualisieren.

Zusammenfassung und Ausblick

Ein weiterer Faktor zu dem Ergebnis dieses Praktikums ist die fehlende Motivation, die wir beide hatten. Schon allein der Grund, keinen vernünftigen Ton zu erzeugen war richtig demotivierend. Man dachte sich alles Mögliche aus, und trotzdem klappte nichts. Hinzu kommt noch, dass die Dokumentation von LibCinder überhaupt ausreichend ist. Es werden zwar zu jeder Klasse die Funktionen aufgelistet, jedoch gibt es dazu keine vernünftigen Erklärungen. Die dazu bereitgestellten Tutorials sind auch keine große Hilfe, denn die aufgeführten Funktionen sind entweder veraltet oder nicht mehr verfügbar. Trotz Demotivation konnte ich viel bezüglich der C++-Programmierung und Anwendung von der LibCinder Bibliothek lernen. Mit besseren Kenntnissen wäre es wohl möglich, komplexere Anwendungen zu erstellen, die nicht nur Musiker ansprechen, sondern auch Zeichner. Der Gedanke, durch Zeichnungen Kompositionen zu erstellen wäre für viele Künstler ein ansprechendes Thema.