AIR WIITAR

MIDIOUTPUT

Hochschule für Angewandte Wissenschaften Hamburg

Fakultät Design, Medien und Information

Department Medientechnik

Studiengang Media Systems

Audio- Videoprogrammierung

WiSe 2014/15

Lars Krafft 2126500

Dozent Prof. Dr. Andreas Plaß

INHALT

EINLEITUNG

Der Projektname Air Wiitar leitet sich von dem englischen Air Guitar ab, was so viel wie Luftgitarre bedeutet und nimmt dabei Bezug zu der Wii einer Spielekonsole von Nintendo. Ziel des Projekts war es mithilfe einer Bewegungssteuerung und der Ausgabe von Midisignalen eine funktionierende Luftgitarre zu programmieren.

Diese Dokumentation befasst sich hauptsächlich mit der Erzeugung der Töne und dem Aufbau des dafür verantwortlichen Programs.

BIBLIOTHEKEN

Als externe Bibliothek wurde „drumstick“[1] verwendet die eine Vielzahl an Funktionen zum arbeiten mit Midi zur Verfügung stellt. Um einen besseren Einstieg in das Projekt zu finden wurde außerdem von Prof. Plaß erstellter Sourcecode[2] verwendet auf dem dieser Teil des Projekts zu großen Teilen basiert.

ANFORDERUNGEN

Aus langer Erfahrung mit der Wii und ihren Controlern war schon zu Beginn des Projekts klar, dass die Genauigkeit unseres fertigen Programs nicht für präzise Aktionen ausreichen würde. Es wurden daher schon früh Überlegungen angestellt, wie sich das präzise greifen auf einer Gitarre mit einer unpräzisen Eingabemethode abbilden ließ ohne dabei das Gefühl des Gitarrespielens vollständig zu verlieren.

Die Lösung bestand darin nicht alle Saiten des Instruments einzeln zu betrachten sondern immer ganze Akkorde zu spielen. Auch die maximale Anzahl der Akkorde sollte begrenzt sein. Es wurde beschlossen, dass vier Akkorde für die meisten bekannten Songs vollkommen ausreichen würden. Welche vier Akkorde gerade gespielt werden können musste allerdings frei wählbar sein.

Diese Angaben ermöglichten es einen ersten Überblick darüber zu bekommen was im Midioutput-Teil des Projekts programmiert werden musste. Durch weitere Anforderungen die speziell von den anderen Teammitgliedern kamen und dazu dienten ihnen den Umgang mit den Akkorden zu erleichtern kamen im Projektverlauf noch einige Änderungen und Ergänzungen hinzu.

KLASSEN

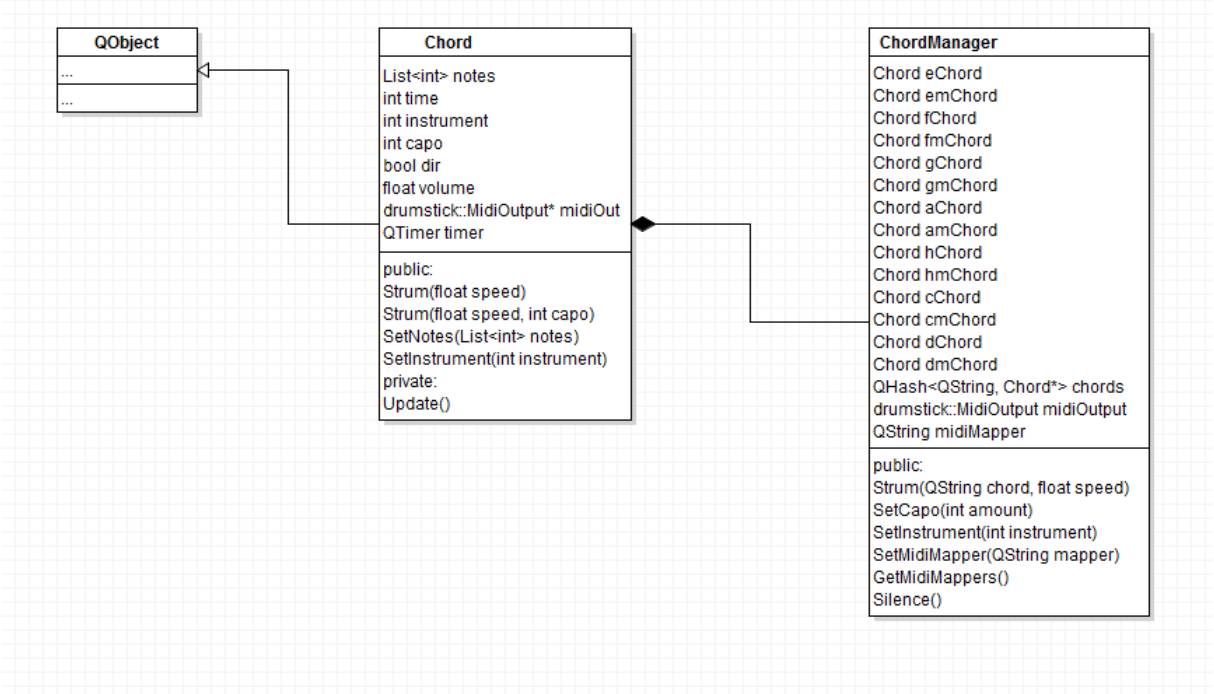

Bild1: UML-Diagramm des Midioutputs

CHORD

Die Chord-Klasse symbolisiert einen Akkord also eine Gruppe von Tönen die zusammen abgespielt einen angenehmen Klang ergeben. In der Musiktheorie sind Akkorde sehr berechenbar und es wäre daher ein Leichtes einen Algorithmus zu programmieren, der automatisch Akkorde generiert. Da sich das Projekt allerdings so stark wie möglich an einer Gitarre orientieren sollte sollten auch die Akkorde genau denen einer Gitarre entsprechen.

Anders als z.B. bei einem Piano bei dem Akkorde häufig nur aus drei Tönen bestehen setzen sie sich bei einer Gitarre aus vier bis sechs Tönen zusammen. Dies liegt an der Bauweise der Gitarre. Mit ihren sechs Saiten bieten sich Akkorde mit sechs Tönen an. Allerdings hat jede Saite nur eine begrenzte Anzahl von Tönen, die sinnvoll für einen Akkord genutzt werden können. Deshalb hat der D-Akkord auf der Gitarre beispielweise nur vier Töne. Eine weitere Eigenschaft der Gitarre ist, dass der tiefste Akkord nicht etwa das C sondern das E ist.

Mit diesen Eigenschaften im Hinterkopf erschien es einfacher die sieben Akkorde von Hand einzugeben als einen Algorithmus zu entwickeln, der alle Eigenschaften der Gitarre berücksichtigt.

Bei einer echten Gitarre werden die Töne eines Akkords nie zeitgleich gespielt, da sich die Hand der Reihe nach entweder von oben oder von unten über die Saiten bewegt. Diese Bewegung kann mal schneller und mal langsamer geschehen und genau dieses Verhalten sollte auch in unserer Wiitar abgebildet werden.

Um die zeitliche Verzögerung zu erzeugen wurde auf QTimer zurückgegriffen und eine Update-Methode implementiert. Um das Timer-Event mit der Chord-Klasse zu verknüpfen war es nötig, dass die Chord-Klasse von dem QT eigenem QObject erbt. Dies brachte die Einschränkung mit sich, dass Instanzen der Chord-Klasse nicht kopiert werden sondern nur als Pointer übergeben werden können.

Jeder Instanz von Chord beginnt in regelmäßigen Abständen die eigene Update-Methode sobald die Stum-Methode aufgerufen wurde. Bei jedem Aufruf wird dabei der nächste Ton des Akkords angespielt. Durch ändern der QTimer-Geschwindigkeit ist es so möglich die programmierten Akkorde genau die echten mal schnell und mal langsam anzuspielen. Um das Gefühl einer echten Gitarre noch weiter zu imitieren wird zusätzlich die Lautstärke der Töne an die Geschwindigkeit angepasst.

CHORDMANAGER

Das Anlegen aller Akkorde von Hand wurde ursprünglich im Code des Mainwindows erledigt. Außerdem wurden die Akkorde fest mit Buttons verbunden. Da der Midioutputcode in ein anderes Projekt übernommen werden sollte wurde die ChordManager-Klasse implementiert.

Diese Klasse hat die Aufgabe die Akkorde anzulegen und zu verwalten außerdem ermöglicht sie das spielen von Akkorde nur durch Angabe eines Strings. Nach der Implementierung dieser Klasse musste sie nur noch in einem anderem Projekt initialisiert werden um alle Funktionen des Midi-Outputs sinnvoll gekapselt zur Verfügung zu stellen.

Um das anspielen von Akkorden über einen String zu ermöglichen wurden die Akkorde innerhalb des Chordmanagers in einem QHash<QString, Chord*> abgelegt. Bevor der entsprechende String geladen wird, wird er auf die Symbole „#“ und „b“ durchsucht um Halbtöne zu erkennen. Sollte eines der Symbole gefunden werden wird der Akkordnormal ausgewählt allerdings mit einem virtuellen Kapodaster aufgerufen um alle Töne des Akkords um einen Halbton zu erhöhen oder zu erniedrigen.

Das erniedrigen von Tönen ist bei einer echten Gitarre mit einem Kapodaster nicht möglich wäre allerdings häufig eine praktische Funktion – alternativ muss die echte Gitarre herunter gestimmt werden – deshalb wurde beschlossen die Funktionalität in der Software zu ermöglichen auch wenn sie nicht der Realität entspricht.

QUELLEN

[1] drumstick: http://sourceforge.net/projects/drumstick/

[2] Prof. Plaß Sourcecode: https://github.com/MediaSystems/avprg/tree/master/midi