MiniSynth – eine mobilfähige WebApp

1. Beschreibung der Software

Der MiniSynth ist ein komplett in Javascript und HTML geschriebener Synthesizer kombiniert mit einem einfachen Sequencer, der in modernen Browsern sowohl auf Desktop-PCs, als auch auf mobilen Endgeräten (Smartphones, Tablets, etc) lauffähig ist.

Folgende Definitionen finden sich auf Wikipedia:

Synthesizer: Ein Synthesizer ist ein Musikinstrument, welches auf elektronischem Wege per Klangsynthese Töne erzeugt.

Sequencer: Ein Sequenzer ist ein elektronisches Gerät oder eine Software zur Aufnahme, Wiedergabe und Bearbeitung von Daten zur Erstellung von Musik. Kern eines Sequenzers ist die Speicherung und Übermittlung einer Partitur an einen Tonerzeuger.

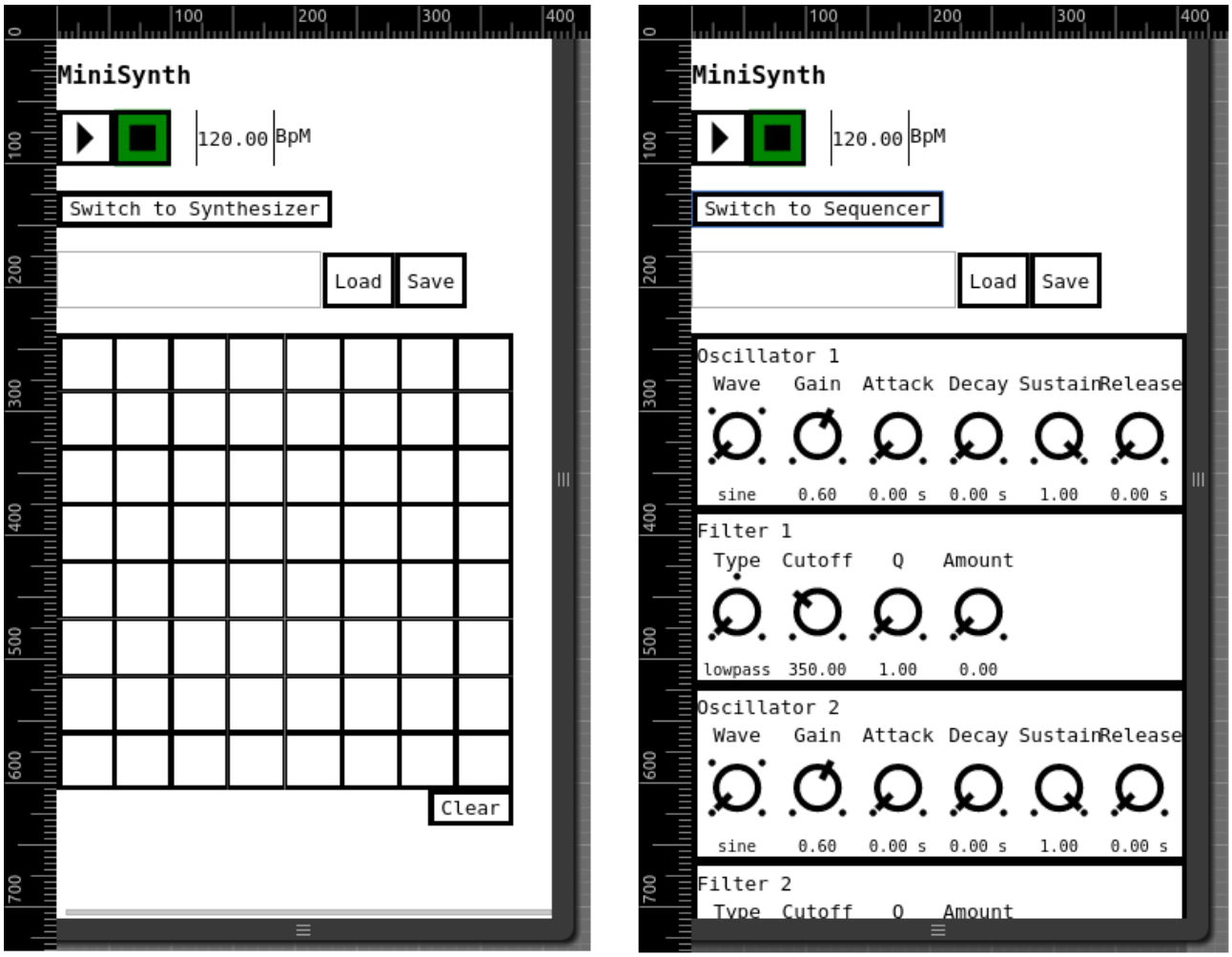

Abb 1: links die Sequenceransicht, rechts die Synthesizeransicht

Wird die Software im Browser gestartet, erscheint zunächst der Sequencer. Dort ist es dem Benutzer möglich verschiedene Noten innerhalb eines Taktes auszuwählen, die dann von dem Synthesizer in Klänge verwandelt werden. Die Noten werden in einer Schleife abgespielt. Auf

dem Schachbrettartigem Muster entspricht die X-Achse dann der Position innerhalb des Taktes und die Y-Achse entspricht der Tonhöhe.

Oberhalb des Sequencers befindet sich noch eine Speichern- und Ladefunktion für Notenmuster und ein Umschalter zum Synthesizer.
Außerdem eine Start- und Stopptaste ein Feld zur Tempoangabe, sowie ein Lautstärkeregler, diese Kontrollen sind auch in der Synthesizeransicht verfügbar.

Nach Druck auf den Umschalter zum Synthesizer erscheint eben dieser. Der Synthesizer arbeitet mit Hilfe von subtraktiver Klangsynthese (mehr zu diesem Thema: http://de.wikipedia.org/wiki/Subtraktive_Synthese).

Der hier zu sehende Synthesizer besteht aus 2 Oszillatoren, 2 Filtern und einem Lautstärkeregler. Die Frequenzen der von den Oszillatoren erzeugten Wellen hängen von denen im Sequencer eingestellten Noten ab, durch Drehregeler lassen sich die Wellenform und die Hüllkurvenform der erzeugten Klänge verändern. Die Klänge der Oszillatoren werden jeweils in einen Filter geleitet. Jeder Filter enthält einen Amountregler, der bestimmt, wieviel des Klangs gefiltert, bzw. durchgelassen wird. Außerdem lassen sich der Typ des Filters, eine Cutoff-Frequenz und eine Resonanz Q einstellen. Danach werden die Klänge zusammengeführt und ausgegeben.

Auch der Synthesizer hat eine Speichern- und Ladefunktion um gemachte Einstellungen später am gleichen Gerät wieder abrufen zu können.

2. Softwaredesign der Software

Die Software ist nach dem Prinzip der Objektorientierten Programmierung (OOP) designt, wobei ein besonderer Wert auf spätere Weiterentwicklungen wie z.B. ein Samplebasierter Drumcomputer oder eine Keyboardklaviatur gelegt wurde.

Zu diesem Zweck wurden die allgemeinen Basisklassen Player und Instrument entworfen. Der Player ist lediglich für die Erzeugung von Noten zuständig, kann aber selber keine Töne erzeugen – dies macht das Instrument. Sowohl der Player, als auch das Instrument definieren allgemeine Schnittstellen, die in Kindklassen implementiert werden. Konkrete Kindklassen sind OneBarStepSequencer und ModularSynth.

Zur Soundsynthese verwendbare Module können zu dem ModularSynth hinzugefügt werden, dieser kann theoretisch aus einer beliebig miteinander verknüpften Struktur aus Modulen bestehen. Die einzelnen Module bilden verschiedene Komponenten der subtraktiven Soundsynthese ab. Zur Zeit sind folgende Module verfügbar: SoundGenerator (Oszillator), PassFilter (filtert Klänge), Amount (bestimmt den Anteil, der durch ein anderes Modul geroutet wird, bzw. einfach unverändert durchgelassen wird) und Gain (Lautstärkereglung).

Die Klasse Clock liefert in sich selbst adjustierenden Intervallen ein Event "tick" an, das verwendet wird um den Player aufzufordern den jeweils nächsten Takt an das Instrument zu senden. Die Clock verwaltet das Tempo und kann auch die Wiedergabe stoppen und starten.

Die TimeCollection-Klasse bietet Methoden zur Verwaltung von zeitbezogenen Daten (z.B. Noten auf einer Zeitleiste).

Klassen, die von Observable angeleitet sind, haben die Möglichkeit Events zu senden, die mit

Eventlistenern abgefangen werden können.

Abgeleitet von der Observable-Klasse ist die Klasse ChangeFiring. Diese liefert die Möglichkeit Eigenschaften mit set und get zu setzen. Dabei werden automatisch Events gesendet, wenn sich der Wert geändert hat. Durch die bindProperty-Methode können zwei Werte von ChangeFiring-Objekten miteinander verknüpft werden (d.h. ändert sich der eine Wert wird automatisch der andere aktualisiert).

Die StateExchangable-Klasse bietet ein einfaches Interface zum Abfragen und Setzen des aktuellen Zustandes eines Objektes an. Dies wird unter anderem zum Speichern und Laden von Zuständen von Objekten verwendet.

Es gibt mehrere HTML-Klassen, die einzelne Anzeigeobjekte repräsentieren, z.B. die RotaryControl-Klasse (Drehregler).

Die Controllerklassen erzeugen einerseits die logischen Klassen, als auch entsprechende Anzeige zu diesen und andere Controllerklassen, die für andere Parts der Software zuständig sind. So erzeugt zum Beispiel der SoundGenerator-Controller ein Modul vom Typ Soundgenerator und mehrere Drehregler um das Modul zu steuern. Das zentrale Element ist der Main-Controller.

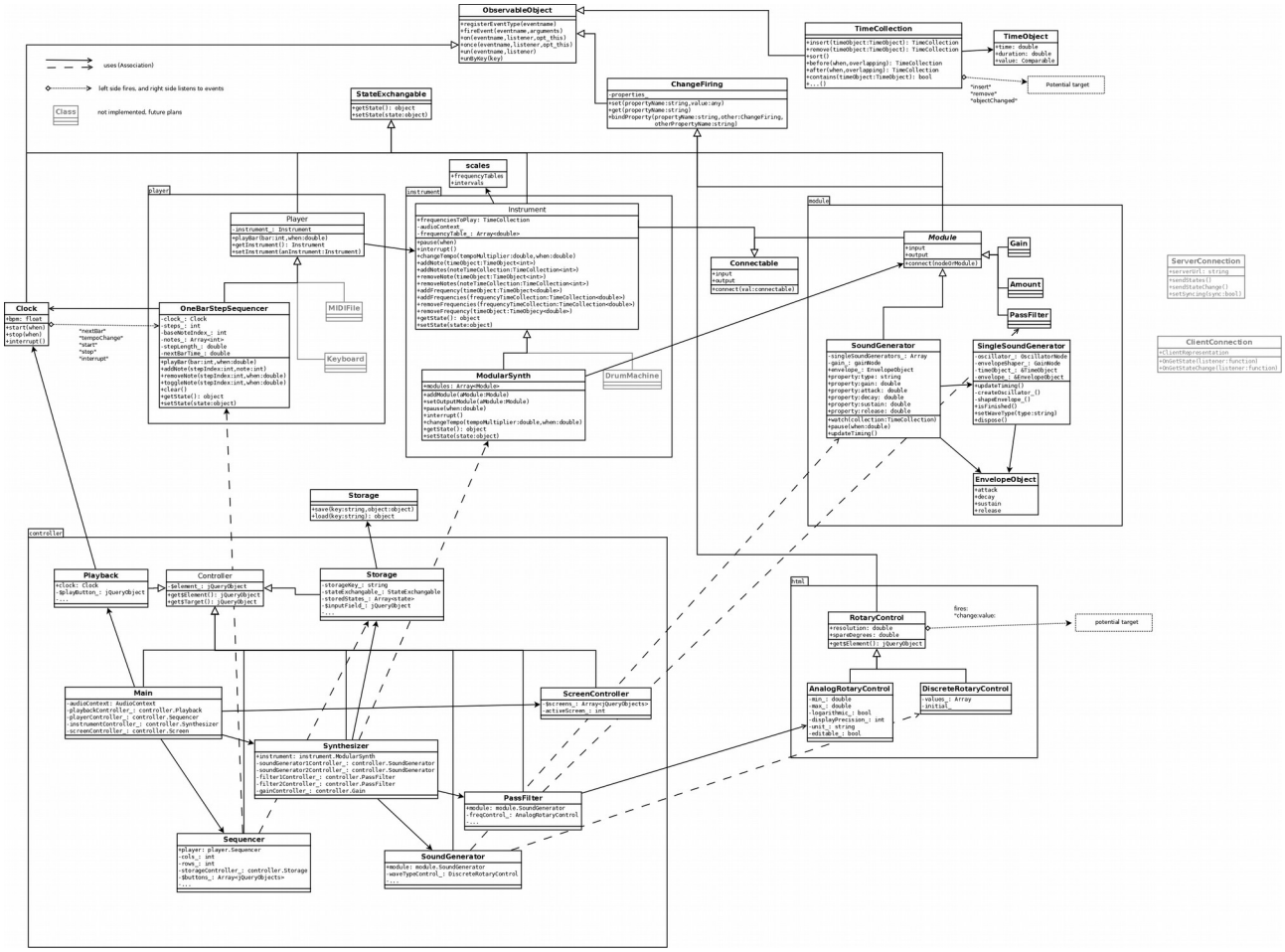

Abb2: Das Klassendiagramm (liegt auch als png bei)

Ein Großteil der Klassen sind in Namespaces organisiert, die im Diagramm als Kästen um die Klassen herum dargestellt sind und auch den Unterordnern im src-Verzeichnis entsprechen.

3. Verwendete Techniken

Die Software ist hauptsächlich in Javascript geschrieben, die Anzeige erfolgt durch HTML. Die Sounderzeugung und -modulation wird mit Hilfe der vom W3C entworfenen HTML5/ECMAScript5 Web Audio Api realisiert (http://webaudio.github.io/web-audio-api/).

Es werden die Bibliotheken jQuery und lodash verwendet. Jquery vereinfacht die Erzeugung und Manipulation der DOM-Struktur von Webpages. Lodash liefert Lowlevelfunktionen zum Beispiel zum Clonen von Objekten oder zum Sortieren von Listen.

Zur Entwicklung wurden folgende Programme/Bibliotheken verwendet:
- git (Versionskontrolle)
- io.js oder node.js (Ermöglicht die Ausführung von Javascript in der Konsole)
- npm (node-Paketverwaltung)
- uprocess (Durch Schlüsselwörter gesteuertes Zusammenfügen von mehreren Javascriptdateien zu einer fertigen Datei)
- less (Eine Syntax um CSS einfacherer und strukturierter in mehreren Dateien darzustellen, wird ebenfalls zu einer Datei zusammengefügt)

Um einen Buildprozess durchführen zu können muss
1. node.js oder io.js installiert sein
2. im Hauptverzeichniss der Software einmal der Befehl "npm install" durchgeführt werden (dieser installiert alle benötigten Bibliotheken lokal in das Verzeichnis node_modules)
3. Mit dem Befehl "npm run build" wird sowohl das Javascript als auch das CSS zusammengefügt und in das build Verzeichnis geschrieben.

4. Probleme und Lösungen

Bei der Entwicklung der Software tauchten an verschiedenen Stellen Problem auf, dabei erwies sich vor allem das Timing von Noten als nicht einfach, da durch das ungenaue Timing der JavaScript-Intervall Methode, erzwungenermaßen alle Noten immer im Voraus an die präzise Audioengine gesendet werden mussten. Dies erforderte einen relativ hohen Verwaltungsaufwand vor allem in Hinsicht darauf, dass alle gesendeten Noten im Falle einer Tempoänderung aktualisiert werden mussten.

Ein weiteres Problem war das Abspielen von mehreren Tönen gleichzeitig, da ein Oscillator der WebAudioApi immer nur ein Ton zu einer Zeit abspielen kann. Die Lösung hierfür war es für jeden Ton einen eigenen Oscillator zu erzeugen. Wenn nun jedoch Parameter von bereits erzeugten Oscillatoren geändert werden, müssen diese alle upgedatet werden.