JavaScript Synthesizer

Konzept

Das Ziel ist es eine Anwendung zu estellen die einen Synthesizer mit einem Sequencer in einer mobilfähigen Webpage darzustellen. Diese sollen beide über zwei unterschiedliche Ansichten steuerbar sein. Globale Kontrollen sollen auf beiden Ansichten verfügbar sein (Geschwindigkeit in BpM, Play/Pause/Stop). Die erste Ansicht ist die des Sequencers. Der Sequencer spielt den Synthesizer. Der Sequencer besteht aus 8x8 Feldern, die verschiedene Noten innerhalb eines Taktes darstellen, der in einem Loop läuft. Die Einstellungen des Synthesizers (Wellenform der Oscillatoren, Hüllkurvenform, Filtereinstellungen) sollen über eine Ansicht mit mehreren Drehreglern steuerbar sein.

Verwendete Techniken

Die Software soll hauptsächlich in Javascript geschrieben werden, sie Audiofunktionalitäten werden mit Hilfe der vom W3C entworfenen HTML5/ECMAScript5 Web Audio Api realisiert werden. Zur Darstellung und Steuerung werden HTML-Elemente verwendet, die mit Hilfe der jQuery Bibliothek vom JavaScript erzeugt werden. Benötigte Bibliotheken auf Clientseite:

- jQuery (DOM-Manipulation, EventHandling in der DOM)
- lodash (nützliche Hilfsfunktionen für Javascript z.B. clone, sort, map, isNaN, etc.)

Benötigte Bibliotheken/Software zur Entwicklung:

- git (Versionskontrolle)
- io.js oder node.js (JavaScript engine die ausführung von Javascript in der Konsole ermöglicht)
- npm (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 durchfuehren zu koennen muss

1. node.js oder io.js installiert sein
2. im Hauptverzeichniss der Software einmal der Befehl `npm install` durchgefuehrt werden (dieser installiert alle(!) benoetigten Module lokal in das Verzeichnis node_modules)
3. Mit dem Befehl `npm run build` wird sowohl das Javascript als auch das CSS gebaut.

Aufbau der Software

1. Modelle / Logik
    - Grundlegende Datentypen:
        - timeObjects: Objekte, die eine Zeit, eine Dauer und assozierte Daten enthalten.
        - TimeCollection: Eine Klasse deren Objekte eine Sammlung von timeObjects enthalten und verschiedene Funktionen enthalten um diese zu organisieren, bzw. gesammelt auf sie zugreifen zu können.
    - ObservableObject
        - Eine Basisklasse, die Funktionalitäten zum Erstellen von Events und Einhängen von Eventlistenern auf dieses Objekt ermöglichen.
    - StateExchangeObject
        - Eine Basisklasse, die das einfache auslesen und setzen aller für das reproduzieren der Objekte im aktuellen Zustand nötigen Parameter realisiert. Soll verwendet werden um es später einfach zu ermöglichen Einstellungen zum Beispiel in einer Datenbank zu speichern.
    - Instrument
        - Das Instrument ist eine Basisklasse, die grundlegende Funktionen eines Instruments in dieser Software darstellt. Momentan ist es nur geplant konkret Synthesizer vom Instrument abzuleiten, es soll jedoch die Möglichkeit im Auge behalten werden, später auch einen Sampler von dieser Klasse abzuleiten.
        - SingleVoiceOscillator --- ein Instrument, dass immer einen Ton zur gleichen Zeit abspielen kann. Keine Tonmodulation, lediglich Wellenform.
        - MultiVoiceOscillator --- ein Instrument, dass mehrere Töne gleichzeitig abspielen kann. Verwendet SingleVoiceOscillator.
        - Synthesizer --- enthält verschiedene Module, die als einzelne Klassen realisiert werden sollen:
            - Oscillatoren --- Verwendet ein oder mehrere MultiVoiceOscillatoren
            - Timer und Envelopeshaper --- Steuert das Timing der Noten und die Hüllkurve (Parameter einer Hüllkurve sind Attack, Sustain, Release, Decay, sie steuern das Anschwellen, Anhalten und Abschwellen eines Tones).
            - Filter --- Eine FilterNode ist bereits in der Web Audio Api enthalten, wahrscheinlich kann sie direkt verwendet werden und muss nicht weiter angepasst werden.
    - Player
        - Eine Basisklasse, die ein Modul representiert, das ein Instrument steuert (spielt).
        - 8x8 sequencer --- Ein Sequencer, der einen Takt in Achtel unterteilt und 8 Noten in einer einstellbaren Skala anbietet.
    - Clock --- Ein Objekt, das regelmäßig die Tonerzeugung anstößt und den Playern mitteilt wo das Playback sich gerade befindet und in welcher Geschwindigkeit gespielt werden soll. Informiert die Player außerdem über starten und stoppen des Playbacks.
    - Scale --- ordnet die Noten einer Tonleiter Frequenzen zu (dabei

entspricht der Wert 0 der Note C3, der Wert 2 der Note D3 usw.)

2. ViewControllers --- Diese Klassen übernehmen die Erzeugung des passenden HTMLs und der Eventhandler und verknüpfen diese mit der Logik.
    - playback
    - 8x8 sequencer view
    - synthesizer view
    - flip view - der Wechsel zwischen zwei Ansichten, hat keine Auswirkungen auf die Logik
3. LESS/CSS Die erzeugten Ansichten werden mit LESS/CSS gestyled

Zeitplanung:

Was?	erledigt am:	
1. Grundlegende Datentypen	05.03.2015	
2. StateExchangeObject	05.03.2015	
3. ObservableObject	05.03.2015	
4. SingleVoiceOscillator	12.03.2015	
5. MultiVoiceOscillator	12.03.2015	
6. Clock	19.03.2015	
7. Sequencer	21.03.2015	
--- Fertig: 25.03.2015 ---		
8. ViewController für Clock/Playback	21.03.2015	
9. ViewController für Sequencer	21.03.2015	
--- Fertig: 02.04.2015 ---		
9. Vorlaeufiges CSS (verschiebbar)	(06.04.2015)	
--- Fertig: 09.04.2015 ---		
10. Synthesizer	06.04.2015	-> Die wichtigsten Parts sind fertig.
--- Fertig: 30.04.2015 ---		
11. ViewController für Synthesizer	06.04.2015	
12. FlipView		
--- Fertig: 14.05.2015 ---		
13. Restliches CSS		
14. Bughunting		
--- Fertig: 28.05.2015 ---		
15. Buffer		
--- Fertig: 12.06.2015 ---		