sequence.me

Overall, excellent job.

40

Alex Glasser, Harrison Kelly, Ian Wixon
GUI II — Project Proposal
February 13th, 2015

Table of Contents

Goal Statement

Sequence.me is an audio sampler and step sequencer that targets web browsers. Using SoundJS and/or the Web Audio API, sequence.me enables users to arrange sound samples to create instrumental loops. These loops, known as *patterns*, can then be organized into full songs that users can arrange to their liking. The application allows users to control the volume and panning of individual sounds to create dynamic, unique compositions.

Features and Components

User Interface

The sequence.me user interface aims to be simple to use for the average user while providing functionality that experienced individuals can take advantage of. The envisioned layout is reminiscent of existing software[1], so the learning curve will not be steep for those who are accustomed to other audio software. The user interface is implemented using jQuery and jQuery UI.[2]

A sequence.me song consists of three layers: tracks, patterns, and the song. Tracks and patterns are reflected in the sequence.me step sequencer and pattern editor, which allows users to choose which sounds they are using (*tracks*), as well as the order in which the sounds are played (*patterns*). The step sequencer provides a view of the currently loaded tracks, their settings (volume, panning, mute), and the grid structure that the user manipulates to control their playback (figure 1)

[1] FL Studio is being used as an influence for the user interface. http://www.image-line.com/flstudio/
[2] http://jquery.com, http://jqueryui.com

sequence. Multiple patterns will eventually be controlled by selecting which pattern to modify from a dropdown menu.

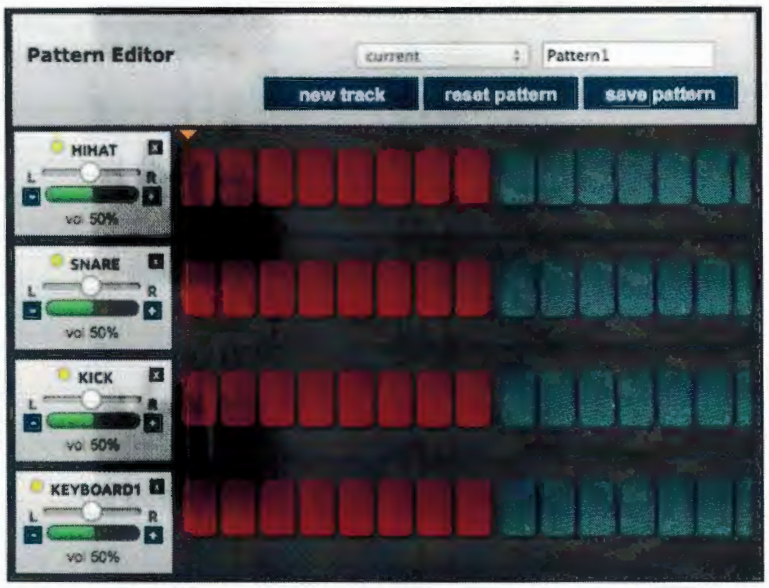

Figure 1. *The step sequencer allows users to sequence the order in which tracks are played within a pattern.*

Each created pattern can then be placed in the song editor, a structure that will allow arranging patterns into a full song. The song editor will provide users with a graphical overview of when each pattern will play and for how long. This is not yet implemented, but is shown in the wireframe below.

Another aspect of the user interface is known as the *transport*, a common element in all audio software. The transport allows the user to control audio playback. Figure 2 shows ~~Below is~~ a tentative ~~screenshot~~ design of the sequence.me transport. Future additions will include radio buttons to select between pattern and song playback, as well as rewind and fast-forward buttons to control playback location.

Please number all figures and refer to them by number in your text to connect your narrative to the illustrations. I noted this in your draft.

Figure 2. *The transport allows users to control audio playback.*

Finally, the user interface allows users to control song settings, such as tempo, time signature, and how many measures a pattern consists of. The user settings panel (Figure 3) will be expanded to include other controllable settings such as master volume.

Figure 3. *The user settings panel allows users to control song settings.*

As shown in the above screenshots, we have begun prototyping the layout for the sequence.me user interface. This will be extended as depicted in the wireframe ~~below~~. in Figure 4.

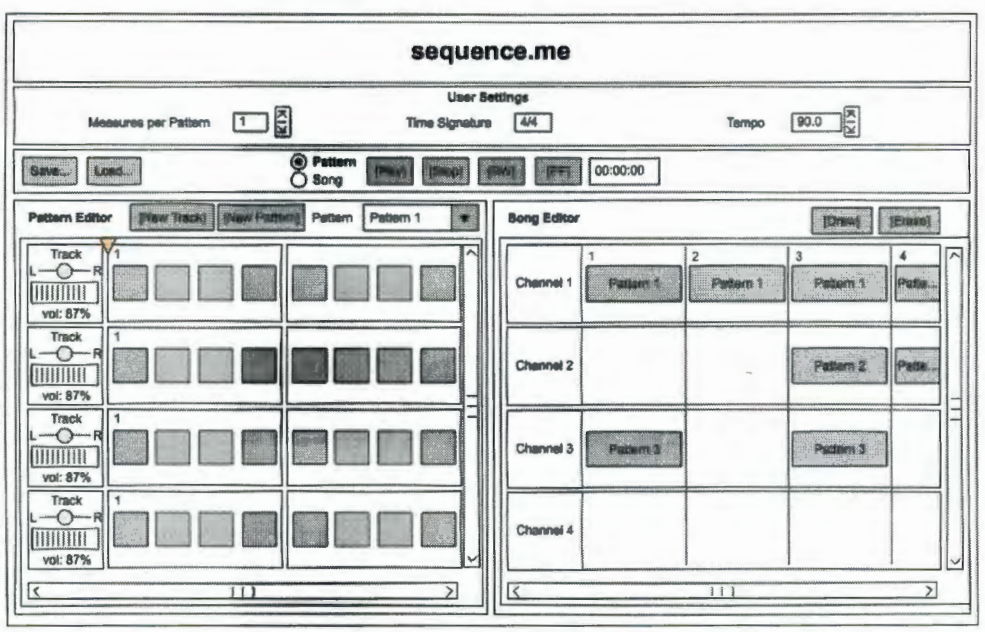

Figure 4. *Wireframe for the eventual sequence.me user interface.*

The song editor will be implemented to allow users to sequence patterns into full songs. Once implemented, sequence.me will provide users with the tools that they need to create musical compositions.

Audio Component

SoundJS[3] plays loaded sounds according to the current state of the step sequencer or song editor. It allows sound samples to be preloaded when the application loads, and later facilitates playing and stopping these sounds.[4][5]

The step sequencer controls a backend representation of the pattern that is being modified. This data structure is visible to the audio component and controls whether a sound will play on each step. Patterns are organized as a list of lists containing a sequence of ones and zeros that determine if the sound is played on that step.

An event handler will be implemented for the user interface to interact with the audio component and vice versa. This event handler will allow the user interface to send signals to the audio component without directly knowing how the audio is played.

Login & Registration

Login and registration will become an integral part of sequence.me. This system will be implemented using PHP, MySQL, and PHP-Login. Without registering, users will have limited functionality—pre-defined sound samples to work with, no save/load ability, and no community to share their work with.

---

[3] http://www.createjs.com/#!/SoundJS
[4] http://createjs.com/tutorials/SoundJS%20and%20PreloadJS/
[5] http://createjs.com/Docs/SoundJS/classes/Sound.html

Registering an account with sequence.me will enable the application's full feature-set. Songs and other user data will be saved into a MySQL database.

Saving & Loading

The ability to save and load created songs is essential to anyone composing digital music. Songs will be saved in JSON format including a title, tempo, sound names and locations, and other information associated with the user's song. These JSON files will be stored and associated with the user's account in our MySQL database. The files can be downloaded so that users can keep local copies for their own storage. Given development resources and adequate time, new versions of the JSON songs will be stored individually, not overwriting old versions. Users will then be able to track and recover their changes as they modify their compositions.

Value Add Features

There are several features that we would like to implement, given enough time. However, these are *strictly value-add features* and are not committed to by the sequence.me development team.

- **MP3/MIDI Export** — We are researching how to export songs to MP3[6] format and/or **MIDI**[7] format for the user to share with other producers. We have not yet proven that this is possible in JavaScript, but are exploring ways of exporting to these formats both client-side and server-side.

[6] http://mpgedit.org/mpgedit/mpeg_format/MP3Format.html
[7] http://www.ccarh.org/courses/253/handout/smf/

- **User Profile Pages** — We would eventually like to provide users with their own profile pages and unique URLs to facilitate sharing songs with others. This page would display a user's created songs with download links, contact information, and other pertinent information that the user would like to provide to others.
- **Community Forum** — These user profile pages could then be used in implementing a community forum in which users can share their songs and provide feedback to other community members. This would be a "final touch" feature that may not be implemented, but would be a nice feature to have, given enough time.

User Description

Our user base will primarily consist of hobbyist audio producers—individuals who are interested in creating digitally sequenced music. To use the application, users will need a rudimentary understanding of basic musical concepts such as tempo, measure, and time signature. Users will have a better experience with existing knowledge of how audio software works in general, but a help system will be provided to diminish the learning curve associated with sequence.me. We envision the vast majority of our user base to be hobbyists, with some professionals using the product to gauge its strength. Our user base will also consist of listeners who can listen to updates from sequence.me artists.

Potential Issues

Audio — Timing

SoundJS eliminates timing issues associated with the step sequencer, but does not currently provide us with a way to dynamically play new sounds as they are placed on the step sequencer. Currently, the pattern must be stopped and restarted for the updated pattern to be reflected in playing audio. This problem is caused by how SoundJS orders sound playback—sounds are queued with a delay value that we are calculating based on tempo and current step in the step sequencer.[8] Sounds are queued almost instantly and then the job of playback is handled by SoundJS, which obeys the delay offset but does not take into account whether the playing pattern has been modified since being queued. This problem is being investigated and may require overhauling how sounds are triggered for playback. We may need to decide what is more important—perfect playback timing or dynamically updating pattern playback.

Community Forum / User Profiles — Storage Space

Implementing a community forum and user profiles will allow sequence.me to become a social platform. However, allowing users to upload their own sound samples and store exported MP3 sounds will quickly become a large strain on our infrastructure. This is being taken into consideration as we discuss file size restrictions, song expiration dates, and other ways to limit the burden on our servers if the application becomes popular.

---

[8] http://createjs.com/Docs/SoundJS/classes/Sound.html#method_play

Acceptance Criteria

Our team has defined goals that can be thought of as the basic acceptance criteria for our project. Sequence.me must provide the following functionality to be considered functionally complete:

- Users are able to create patterns of sounds and arrange them using the song editor.
- User interface controls tempo and pattern/song composition. Audio component is notified of these changes automatically.
- Audio component is able to play patterns that the user modifies during playback. No starting/stopping is required for playback to recognize pattern changes.
- Login and registration functionality provides the user with a means to save and load their compositions.
- Users are able to download JSON files containing their song. Users are able to upload JSON files created with sequence.me and recover work that was saved locally.

Tentative Development Schedule

Due Date	Task	Alex	Harrison	Ian
Jan 27, 2015	Prototype basic functionality	●	●	●
Feb 3, 2015	Commit prototyped code to git, begin implementing UI—Audio interface code	●	●	●
Feb 10, 2015	UI—Audio interface working	●	●	
Feb 10, 2015	Prototype login/registration functionality			●
Feb 17, 2015	Determine feasibility of exporting songs to MP3/MIDI – begin implementing	●		●
Feb 17, 2015	UI—Audio interface complete, tested	●	●	
Feb 17, 2015	Login/registration complete			●
Feb 17, 2015	JSON save/load complete – associated with login	●	●	●
Feb 24, 2015	Alpha version committed and public	●	●	●
April 7, 2015	*MP3/MIDI export complete*	●		●
April 9, 2015	*MP3/WAV sample upload complete*	●	●	●
April 12, 2015	Beta version committed and public	●	●	●
April 20, 2015	*User profiles / community forum complete*	●	●	●
April 29, 2015	Final version committed and public	●	●	●
April 29—May 1, 2015	Final testing, bug fixes	●	●	●

Dates/tasks in italics pertain to nice-to-have features.

Good.