Authors:
Rich Eakin (reakin@ucsd.edu)
Trevor Muzzy (trevor.muzzy@gmail.com)

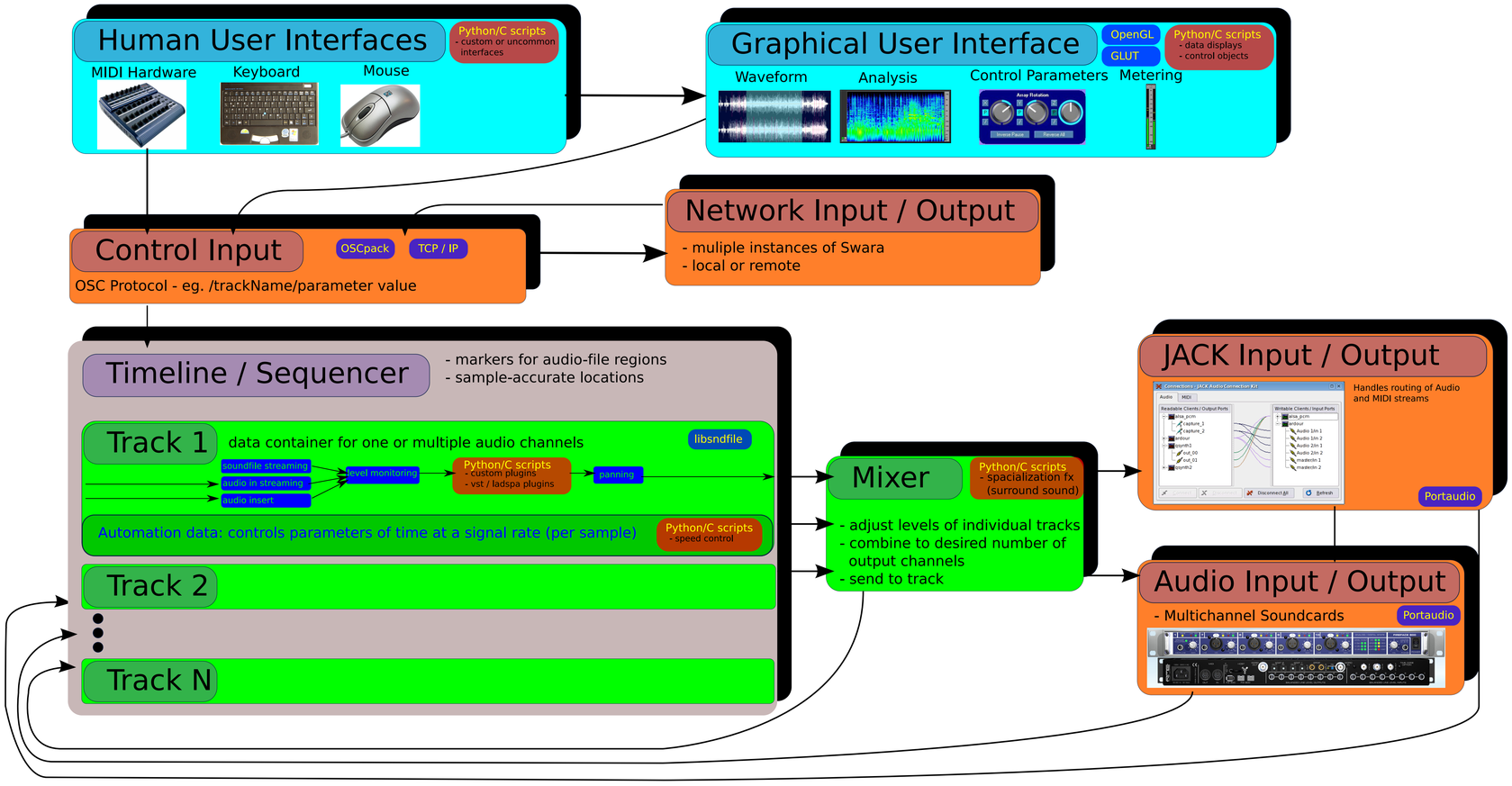
Advisor:
Tom Erbe (tre@music.ucsd.edu)

Swara

Designing a Multi-Platform, Multi-Purpose Sound Editor

Abstract

The creation of a multi-platform, software audio engine could benefit many different fields of multimedia production that require customized operations on sound. It would be ideal for an experimental sound engineer or musician to have a sound-editor well structured for common operations on multiple sound streams, such as mixing surround sound audio, while still maintaining freedom for experimental algorithms that could provide new and interesting audio effects. Swara is the sketch work of an audio engine that aims to be capable of handling multi-channel audio in real-time, able to link with or embed in other multimedia applications and extended with user-defined scripts.



Overview of Features

Seperation between graphical interface and audio engine

Provides multi-computer operation, where multiple sound engines (possibly remote) can be controlled by a single or multiple graphics layers

Network communication between applications

The seperation is achieved by using a standard network protocol (in the form of OSC - Open Sound Control) to allow both the graphical and audio layers to communicate with other applications.

Cross-Platform

Swara is built from the ground up as a multi-platform application, tested on Max OS X, Ubuntu Linux, and Windows XP. This is achieved using only cross-platform API's and coding graphical objects at a low level

Extendable unit design using scripts

Allows access to the heirarchy of methods within the sound editor, as well as allowing for extendability of all but the core elements of Swara

Unlimited channels / tracks

An open design is used to support multi-channel applications, which may need any number of tracks that hold audio or channels that allow for audio input and output.

Applications

Experimental music composition and production

Swara is meant to be, at the same time, both easy to operate as a conventional sound editor and capable of complex, experimental processes. This is achieved by creating an intuitive graphical interface that is seperated from the audio engine it is controlling. The engine can then be extended in any number of ways using scripts, which can also then be represented on the graphical layer.

Can be used as an embedded application

Multimediasts involved in areas such as virtual reality, gaming, cinema, and others can benifit from Swara's multi-channel support and scriptability, while being included in the code of other existing applications that are adept to the field at hand.

Used in concurrence with other audio applications

Swara can also be linked to other applications that process audio, allowing one to take advantage of the best features of both.

Progress

The bulk of our time this summer was spent coding the framework and main architecture of Swara. The work was split between the audio engine and the graphical interface, which is appropriate for the overall structure of the application. Below is an overview of the main modules created so far. It is obvious that there is still much work to do!

Audio Engine

SwaraApp -

Main instantiation of application. Contains synthesizer.

Synthesizer -

Audio processing functions. Uses PortAudio API, which is compatible with all main computer architectures. PortAudio also supports JACK, allowing interconnectability between applications. This makes it possible to use and test Swara at a very early age. Processes audio using a callback routine

Unit -

Main unit generator modules. Contains a linked list of pointers to the audio block, making real-time signal processing efficient. All unit generators inherit this class, which so far includes a sinewave module and amplitude gain module, used to test the functionality of the audio engine.

Graphical Interface

Window -

Contains all GLUT windowing and event handling functions necessary to create and take input from a window. All events get handed up the hierarchy to the panel.

Panel -

Container for controls. Visually, a panel occupies a portion of a window. Panels have assignable background, foreground, highlight and text colors.

Control -

Represents a basic control with only one value. Controls have assignable background, foreground, highlight and text colors.

Slider -

Contains the functionality necessary to create a horizontal slider object. Mouse click and drag events are handled and used to redraw the control. Redraw happens either by mouse input or an internal value change.

Lastly, GNU Autiomake tools (Autoconf, Audomake, Autoscan) were incorporated to ensure the application is cross-platform compliant from the start. With these tools, one can configure Swara for compilation on any number of computer architectures, including but not limited to Mac OS X, Windows XP, Windows Vista, and many different Linux builds.

University of California San Diego