Audio UI Demo

Code and Sound Design by
Peter "pdx" Drescher
Twittering Machine
http://www.twittering.com
twittering@gmail.com
425.202.7721

The purpose of the **Audio UI Demo** is:

a) to demonstrate, and experiment with, techniques for creating system sound effects, as discussed in my "Annoying Audio" blog "Generating Audio UI" (using game audio techniques to make user interfaces sound more like they do in the movies), published by O'Reilly.com.

b) to update and enhance my C++/MacOSX programming skills, and increase my employment prospects (note: I am currently available for contracts, and full time positions).

Also note: the **Audio UI Demo** doesn't actually *do* anything, other than generate audio in response to keyboard entry. It is not a word processor, or web browser; rather, it is intended to mimic the sounds heard when people type at computers in the movies and on television.

Sound Sets

The program contains 8 sets of audio files:

⌘A	Beep:	sine wave beeps, a quarter-tone apart
⌘S	Blip:	very short square wave (~3ms), 25 cents apart
⌘D	Noise:	short clips of pink noise differentiated by resonant sweep
⌘F	Telemetry:	high-pitched sound of data transmissions, varied by tone
⌘G	Chirp:	a series of bird chirps (Twittering Machine)
⌘H	Keyboard:	recordings of "key clacks" on an old keyboard
⌘J	Chromatic:	short sawtooth wave, on a chromatic (half-step) scale
⌘K	Diatonic:	short muted guitar sound, on a diatonic (C Major) scale

Algorithms

Sounds are produced for each keyDown event based on the selected Algorithm:

⌘1	Monotone:	plays the same sound each time (note: if a mobile device makes any sound when typing, or when pressing numeric keys, it always uses this algorithm).
⌘2	Random:	aka "White Noise", each successive sound is selected at random from the set.
⌘3	1/f:	aka "Pink Noise", each successive sound is selected from the set using a fractal algorithm.
⌘4	Stochastic:	aka "Brownian motion" or "The Drunkard's Walk", each successive sound is selected by moving a small random distance up or down from the previous one.
⌘5	Weighted > Alpha:	each key is assigned a sound, alphabetically.
⌘6	Weighted > Qwerty:	each key is assigned a sound, based on the rows of the Qwerty keyboard.
⌘7	Weighted > ETAOIN:	each key is assigned a sound, based on the letter frequency of the English language.

KeyWords

search “<string>”	do a google search, and return the labels
hello	pick a canned response at random
goodbye	pick a canned response at random
clear	clears the screen
you know what I meant	this 'macro' was programmed into the first IBM mainframe I ever worked on, as a reminder that computers take everything literally.
fingerprint	a CSI-style fingerprint matching routine, to be implemented in the next version (when I figure out how to hack the FBI database :)
<any question>?	a "magic 8 ball" routine (random canned response)
list	prints out the list of keywords
ascii	prints out the ascii character set (useful for listening to sound sets and algorithms)

Speed

Determines the speed at which screen printing mode displays characters.

⌥⌘1 - Slow
⌥⌘2 - Medium
⌥⌘3 - Fast
⌥⌘4 - Zip!

Notes

• There are two types of errors:

“white alert” indicates a search was performed, but no matching documents were found. For example, try typing: *search “.”*

“red alert” indicates the computer is not connected to the Internet. Try unplugging your network cable (or turning off AirPort), then doing a search.

Each sound set contains its own related alert sounds.

• The Beep and Blip sounds sets are varied by pitch, separated by 25 cents (quarter tones). This produces “musical” variations, but within a limited range of notes. The Chromatic and Diatonic sets produce musical variations using a much wider range, for reference and experimentation. While they may be “interesting” from a computer-generated audio point of view, they would be less “useful” (i.e. quite annoying!) in an actual UI.

• The Noise, Telemetry, Chirp, and Keyboard sound sets consist of various short audio clips *not* ordered by pitch. Thus the Algorithms have a less obvious effect on the overall sound than with the pitched sets.

• The Telemetry set is probably the most successful as a UI effect (i.e. sounds like computer readouts in many movies), but Beep and Blip work as well. The Keyboard set can be almost comical, particularly at high Speed, and the Chirp set is included because, well, after all, my studio *is* named Twittering Machine ...

• The Monotone Algorithm is most commonly used in the real world, when typing produces any sound at all (as on mobile devices).

• The Random Algorithm is reminiscent of computer readouts from 1950’s science fiction movies (try typing “ascii” using Beep or Chromatic, Random, Medium).

• The 1/f Algorithm generates the most “musical” variations when used with the pitched sound sets. To understand why, refer to “White, Brown, and Fractal Music” by Martin Gardener.

• The Stochastic Alghorithm is possibly the least interesting when used with the pitched sound sets, in that it tires the ear quickly, like listening to radio static wavering up and down. However, it seems to have a more pleasing effect when used on the unpitched sets, because the sounds recur in changing patterns, producing a more ordered, yet less repetitious, sound than other algorithms.

• The Weighted Algorithms can be more “informative” to the user, in that the same sound will be produced by the same key each time, thereby creating unique audio patterns for each word entered or printed. While this seems like it might be useful, in fact, the ear rarely recognizes these patterns, unless they are distinctive and/or repeated (try typing: search “iiiiiiiiii”). Personally, I find the ETAOIN Algorithm the most “coherent”, but they all have a similar feel (mostly, letters being distinct from punctuation, depending on the sound set being used).