Beat Browser

by

Jeffrey D. Goldenson

B.A. Architecture
Princeton University, 1999

SUBMITTED TO THE PROGRAM IN MEDIA ARTS AND SCIENCES
SCHOOL OF ARCHITECTURE AND PLANNING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JULY 2007

Author

Program in Media Arts and Sciences
July 6, 2007

Certified by

Christopher Schmandt
Principal Research Scientist
M.I.T. Media Laboratory
Thesis Supervisor

Accepted by

Andrew Lippman
Chairperson
Department Committee on Graduate Students

BEAT BROWSER

by

JEFFREY D. GOLDENSON

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning on July 6, 2007
in partial fulfillment of the requirements for the Degree of Master of Science

ABSTRACT

Beat Browser is a music browsing environment that delivers immediate audio feedback while browsing arbitrarily large music collections. The goal of Beat Browser is to give users a sense of exploring "live" and continuous audio while rapidly moving between sources by mouse. It appears their entire universe of music is playing all the time, whether they're there listening or not. Beat Browser's Universal Time Base architecture keeps a central clock running that manages the playback position of every piece of music launched, orchestrating this perceptual illusion.

Thesis Supervisor: Christopher M. Schmandt

Title: Principal Research Scientist, M.I.T. Media Laboratory

Thesis Committee

Thesis Advisor:

Christopher M. Schmandt
Principal Research Scientist
M.I.T Media Laboratory

Thesis Reader:

Barry Blesser, Ph.D.

Thesis Reader:

William Gardner, Ph.D.

Acknowledgements

First, I'd like to thank my advisor Chris Schmandt for providing really sharp insights and guidance throughout the research process. I'd also like to thank him for accepting me into the Speech and Mobility Group, I've had a great time here.

Linda Peterson, for being a fantastic, and unfailingly honest, guide through my experiences here at The Lab. Without her, I would not have had the experience I had.

My group mates for welcoming me and making me feel at home.

Kent Larson, for accepting me into the program.

I'd like to thank my readers, Barry Blesser and Bill Gardner. I've really appreciated our conversations and both of your input, both directly, and by example.

Robert Burkhardt, who's coding experience was invaluable.

My parents for their support.

My brother Andy, for being an engineer who always made time to bail me out of numerous situations. I'd also like to thank him for donating the hardware behind the Spatial Player research.

And finally my wife Natalie. Your perspective has been refreshing and your support awesome. Thanks partner, I couldn't have done it without you.

Table Of Contents

1. Introduction

Brian Eno

> *"In the 70's I came up with a word for this kind of music that more and more people were starting to do which I called Ambient Music. And that was quite a different idea [from narrative music], that was the idea of music as a sort of steady state condition that you entered, stayed in for a while, then left. Music as painting [rather] than as narrative.... It's closer to sitting by a river than to listening to an orchestra.*

Brian Eno, in conversation with Will Wright, "Playing With Time," The Long Now Foundation, June, 2006

The steady state nature of Ambient Music and the nature of Beat Browser form a useful parallel. Beat Browser is a kind of steady state music browsing environment, designed to be entered, stayed in for a while, then left.

Like Ambient Music, Beat Browser is always happening. It isn't stopped or started, it just *is*.

a) The Problem

Software music players are designed implicitly with *search* in mind. These tools are optimal when you know what you're looking for. They are a multi-step experience that may be broken down as follows:

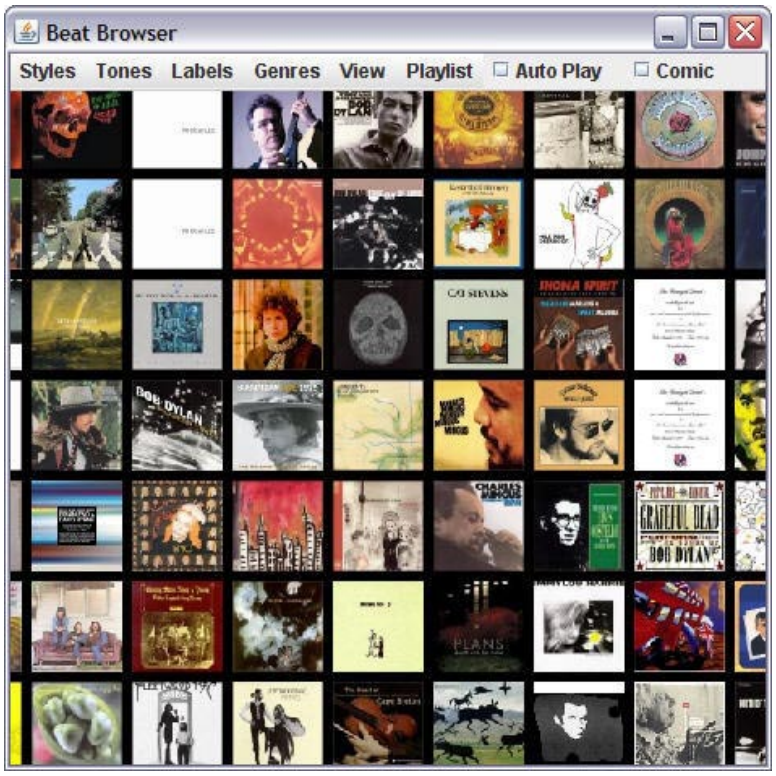
1. Establish what you'd like to listen to
   duration unknown

2. Scroll, navigate or begin typing to find that artist's name
   approx. 3 seconds
3. Select the song
   approx. 1 second
4. Commence playback
   approx. 1 second until the audio is audible

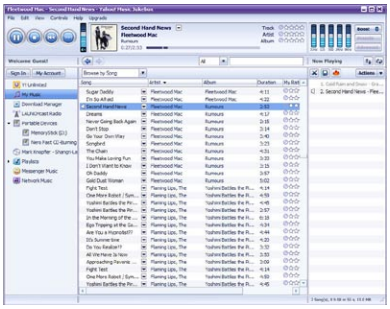
Cumulatively, this is a *~five second* delay between when you begin search to when you actually *hear* something. Beat Browser argues that these five seconds are the problem.

They quickly add up. These five seconds are full of reading, decision making and fine-motor mouse interaction. This focused cueing takes the fun out of any kind of music browsing or "shopping around." It's a lot of work that doesn't have anything to do with listening to music. And it's this work that discourages people from trying something new.

Beat Browser is explicitly designed with *browsing* in mind. It welcomes you not with a linear list, but with 2D field of album art. It ushers you into your collection with immediate musical response, letting you sift and surf until something catches your ear.



Beat Browser, displaying the basic welcome view



Yahoo! Music Unlimited

Apple's iTunes with CoverFlow, the album cover visualization, open

There are distinct scenarios where browsing is preferred to search. Beat Browser focuses on two often interrelated tasks: playlist building and music discovery. Playlist building is the act of compiling a selected set of songs for extended playback. Music discovery is finding new music you enjoy or uncovering old favorites long neglected.

These use cases are under served by current software music tools. Beat Browser's design addresses these needs specifically:

1. Instantaneous audio delivery for quicker discrimination
2. Fluid sound design to limit listener fatigue
3. Intuitive controls supporting casual, 2-axis gross-motor interaction

b.) Concept

> *"What would it sound like if every one of your songs, on every one of your albums was playing all the time?"*

This was the thought that inspired Beat Browser. Beat Browser creates the illusion that your entire universe of music is playing – beginning the moment you start-up the application. As a listener you skip between songs and albums as time flows forward in the universe, forming a kind of "river of music." Where you begin listening inside each song, album or playlist, is completely determined by how long you've been running Beat Browser.

c.) How It Works

This is the Halo open. The Halo is dynamically divided for each album, each wedge represents a track on the album.

At the outset of launching the application a central clock is started. Mousing over album covers or song wedges, representing its tracks, each piece of music asks this central clock, the Universal Time Base (UTB), how long Beat Browser has been open. If you've launched the application at 1:23:02 pm, and proceed to listen to an album until you mouse over another at 1:42:22 pm, this second album will begin 19:20 into the album. If we assume we're listening to a Pop album where every song is 3:30 long, we'll find ourselves approximately 1:52 seconds into the 6th song, minus the gaps between the songs.

If you're navigating songs, and not albums or you have been listening continuously for longer than the length of an album, Beat Browser takes the modulus to determine playback position. We'll use a songs as an example, but the same concept applies with albums.

Beat Browser takes the Universal Time Base value of 19:20 and divides this by the duration of some song, a 4:20 Elton John tune, for example. Beat Browser discards the integer value and takes the remainder as a percentage of the song length. This percentage of 4:20 is calculated as a time value, and playback begins at this point into the song.

This architecture constructs a "live" listening environment for users to enter, listen and build playlists. This architecture perpetrates the central illusion: that Beat Browser is steady state machine, playing the whole universe of available music, whether you're there to hear it or not.

2. In Context

a.) Music Access

There are two dominant paradigms in how people gain rights to listen to the music they listen to. Buying albums or singles in the store or on-line provides unfettered playback for personal use so long as anti-piracy and copyright legislation is upheld.

Subscription-based models, where one joins a music "club" of sorts, is the other paradigm. For a flat monthly fee, members gain access to the vast collection of recorded music. It's important to give this relatively recent music distribution channel more focus as the vastness of its library is perfectly suited for new music discovery.

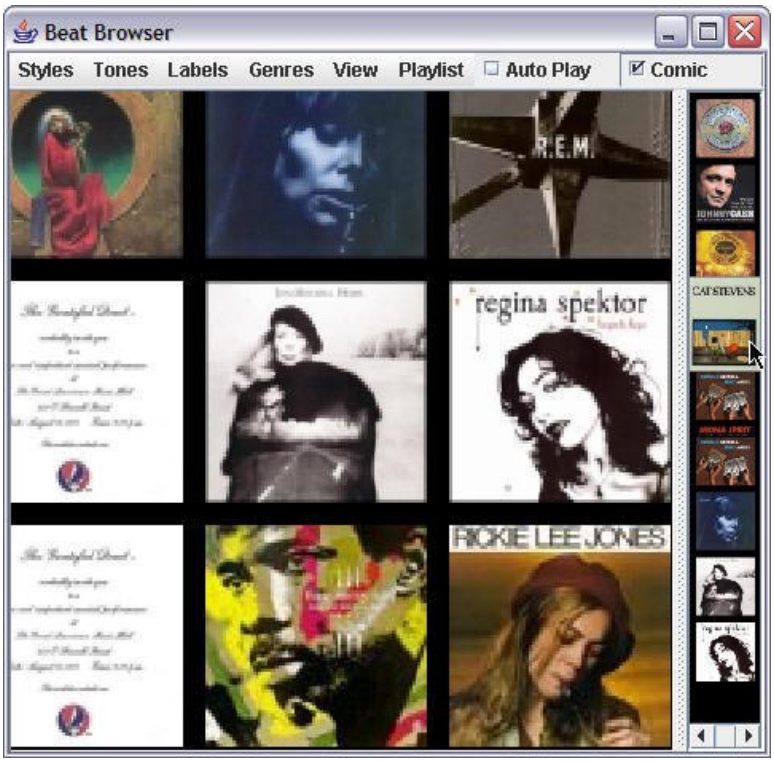
Currently Beat Browser only runs natively, but the ultimate fruition of the concept would be coupling Beat Browser with the true "universe" of music these clubs contain.

As a member you have access, but not control over, this universe. After password clearance, you gain permission to stream these files from the club's server to your local machine. This must occur over a live internet connection.

This is Real Music's Rhapsody listening service.

While you may listen as much as you like, you cannot save these files locally. Rhapsody, the market leader, refers to this concept colloquially as the "Jukebox in the Sky," as Real Networks' CEO Rob Glasser likes to describe it. It creates a new listening experience, inflected by a new business model.

Experimenting with Rhapsody for three months, I passed the initial swell



To the right of the image, the Beat Browser's playlist window is open.

of euphoria of streaming any of "millions of songs" on a whim, and after two months found I was hardly using it. I ran out of the songs I was craving to hear and didn't own. Occasionally I'd drop in to see what's new, rarely did I look into the archives.

Somehow, this ocean of listening options was paralyzing. There were no visual representations of it, and it remained ungraspable and inaccessible. Then I started to let other Rhapsody members guide me. I quickly realized a better use for playlists than setting a mood, here they were maps to new or forgotten islands of music. Playlists inside Rhapsody have a different meaning. As a member, you have access to every piece of music all the time. All the music is in there, you just have to know where to look for it.

As Chris Schmandt has championed after more than two years as a Rhapsody listener, with this kind of music access at your fingertips, the *currency* of this environment becomes *playlists*.

b.) Building Playlists

Playlists are catalogs of time-based media ordered for sequential playback. This allows one to automate extended audio experiences consisting of two or more tracks – just hit start and forget about it.

Successful playlists are not easy to create. At best playlists score a mood or take the listener on a compelling journey. They can be optimized for a large number of factors, from environments to mindsets, aesthetics to beats per minute. They may also leverage higher level factors like

nostalgia and cultural context; it is all a balance between our "desire for repetition and desire for surprise"(Aucouturier, Pachet).

Automating playlist construction has drawn significant research attention within the past several years. The problem of computationally generating a playlist based on arbitrary criteria far exceeds the bounds of this discussion, but a brief survey of several prominent problem-solving strategies is useful.

The most commercially deployed strategy includes an initial query for "seed songs" from which a larger playlist may grow. Software identifies similar songs based on criteria ranging from timbral similarity (Logan Salomon), to tagging (Pandora Internet Radio and AudioScrobbler, described later) and applying machine learning from existing playlists (Platt, Burges). Pandora Internet Radio (www.pandora.com) asks you to select a seed artist. It then plays songs deemed similar by their staff of listeners – humans provide the brains in the loop.

Users create playlists quickly and simply in the Beat Browser environment. The feature set is limited to the basics, but the unique attributes of the music browser reveal themselves. Users can put songs or entire albums in the playlist with the touch of a button. They may roll-over the album art to hear the songs, and delete them with a touch of the same button that put them there. Playlists are saved in the .m3u format that has become standard and is recognized by all major music players. This is a simple text file format that is written in near natural language, legible to the untrained eye.

Building these playlists is a bit like listening, but in fast forward. Typically, users only want quick tastes of the music, and they want to be able to move through their library with agility. Most media players offer a "scrub" bar to slide to access playback points within the song, but this is an interaction-intensive, and consequently time-intensive task.

The goal with playslists is to craft the best mix possible in the least amount of time. Creating playlists in music player environments like iTunes or Rhapsody is work. There are many clicks involved in traversing artists, albums and songs. These programs were simply not designed with playlist building in mind. Consequently, it's not particularly fun and users don't make nearly as many of them as they could be.

Beat Browser focuses on creating an environment to help users build playlists. If Eno describes his approach as "music as painting" than Beat Browser may be seen as arraying all available musical colors to create a playlist palette, in service of composition.

This is Rhapsody's "Mixer" window, a staging area for songs considered for playlist integration.

3. Development

The roots of Beat Browser lie in a longer audio research trajectory. Barry Vercoe's class *Audio Perception in Humans and Machines*, in Fall '06, provided many of the foundation concepts. That semester I realized design focus could be brought to bear on a life-long curiosity with the relationship between sound and space.

As a child I enjoyed "listening" to walls when the lights were out, getting as close to where I felt I could "hear" them (experienced as an increase of pressure in my ears), without touching them. This game provided inspiration for the class project.

The completed ultrasonic sensor

a) Ultrasonic Sensor

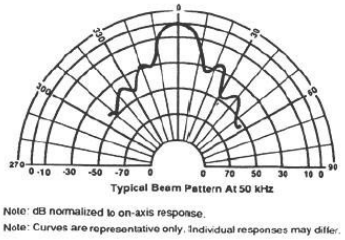
Objective
Using sound, create an experience where our ears and our proprioception converge to describe our environment.

This device employs echolocation to roughly determine the physical character of one's spatial environment in real-time. It uses a single Polaroid 6500, a 50 kHz ultrasonic transceiver to calculate time-of-flight, from which distance is determined.

These echo pulses are driven by an accompanying ranging module that incorporates simple filtering and selective echo exclusion capabilities. Desired echoes are translated into a distance value by a custom application running on a designated microcontroller. The microcontroller is then routed through my serial port and into the Max/

MSP audio programming environment. A Max "patch" then modulates the volume of a pre-recorded audio track proportionately based on the transceiver's proximity to walls or other large obstructions.

Once operational, the limits of affordable ultrasonic sensing became rapidly apparent: obstacles, if detectable, need to be near-perpendicular to the signal to ensure consistent readings; reflections can be very confusing; and varied material absorption really throws the data. Much of the noise could have been improved by the use of two sensors in a binaural setup. This would have provided necessary beam narrowing and was slated for the next design iteration, but confidence was low in ultrasonics in general – reflections were still going to be very limiting.



Spread of the Beam Pattern for the 50 kHz Polaroid 6500.

Modulating volume was implemented as a simple proof of concept. The idea stretched further into mapping sounds more directly on space. An envisioned implementation was to couple the range-finder with a radio tuning application, such that when your distance from an obstruction equaled the height of the waveform, from peak to trough, the system would tune into that band. It was an attempt to make the radio waves a little more "visible" and comprehensible. This seemed OK for a science museum, if it could even work, but it was hardly solving a problem.

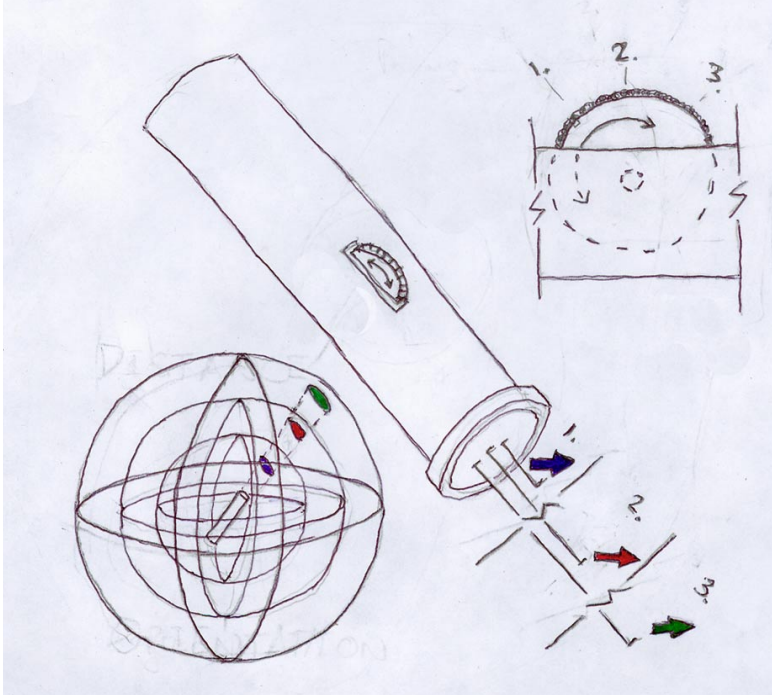
Exploratory sketch visualizing the acoustic concept.

It was this point where I began to think there may be utility in exploring memory and how we use spatial metaphors to aid our memory recall. My interest was what if we could associate data, or music, or any other digital stuff with a location in our physical space.

As the first prototype showed, an entirely new approach would be required if the device was to know much of anything about what it's looking at. This begged a question:

> *"If we couldn't know much about the real environment the device was sitting in, is there a way to create an imaginary environment around the device, but have the device running a map of this space on-board, to make it seem real?"*

Could the device be some other sensor that could describe an imaginary environment, a "wand" that could sense where some invisible structure around us?



This was an early sketch for Spatial Player. Concentric spheres, arbitrarily divisible into degree regions, become event triggers, a type of spherically arrayed buttons.

b) Spatial Player

Speculation
A 3-dimensional, mental model may be an intuitive alternate paradigm to interacting and organizing data on a 2D virtual desktop? We think and dream in 3 dimensions, why not interact with digital information in 3D?

Spatial Player emerged from these questions of how, and to what effect, one can associate audio with spatial position. Can you convincingly simulate the effect of shining an "audio flashlight" around some unreal space, building a mental picture of what is where, that can later be recalled?

This would require tracking where the "flashlight" was pointing. This meant using some sort of accurate heading sensor to monitor continuous orientation in real time.

It turns out that my brother is in the business of designing and building heading sensors for the Navy. These sensors are used in very long underwater towed arrays, long tails full of speakers that use heading sensors to provide extremely accurate orientation of each ping for submarine SONAR.

These sensors turn every which way, but always retain their orientation by sensing both gravity and the earth's magnetic field, in 3-axes. I

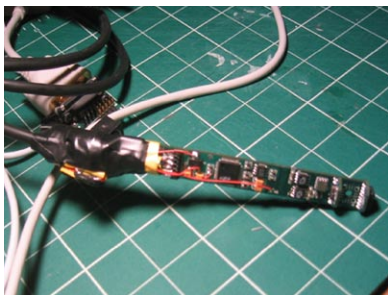
realized that the heading sensor could in fact be the actuator I was looking for – that "flashlight" to illuminate a virtual dimension of our construction.

The first step was to decide on a simplified scheme. This scheme was creating concentric bubbles around the user (*see previous sketch*). Spherical coordinates were chosen over cardinal, as these spheres would have divisible degree regions easier to imagine as places in space. The first application concept was to explode a simple file structure about the user, giving each element a spherical coordinate position.

Clymer's Terrella 6 sensor proved a good development platform, it had extreme sensitivity via 3 axes of accelerometers and three 3 axes of magnetometers. Using the earth's magnetic field as its anchor, Terrella's drift was very low. After adding a 3 position switch to increase input/output, the hardware backbone was in place to deliver readings hopefully consistent enough for users to build a mental model on.

About three months in total went into building a custom Java application to read in the data from the serial port, sufficiently smoothing it and using that data to control the navigation through a file structure of your music.

The first month was spent attaining consistent performance. This demanded multiple filtering schemes, dynamic re-zeroing by the user (a button-press to tell the device where you're facing) and several latency reduction strategies. Operating cooperatively, these strategies created a workable hardware prototype. Then came the issue of tightly fitting the hardware and software architectures to create a responsive and intuitive, spatially triggered music player. This took the remaining two (plus) months



This is basically what the raw sensors look like. To the left is the custom addition of a three-position switch, supporting:

Rock forward

Rock back

Press down

Here is the final, milled aluminum casing with two brass (non-magenetic) screws anchoring the milled switch-casing to the insided of the tube casing.

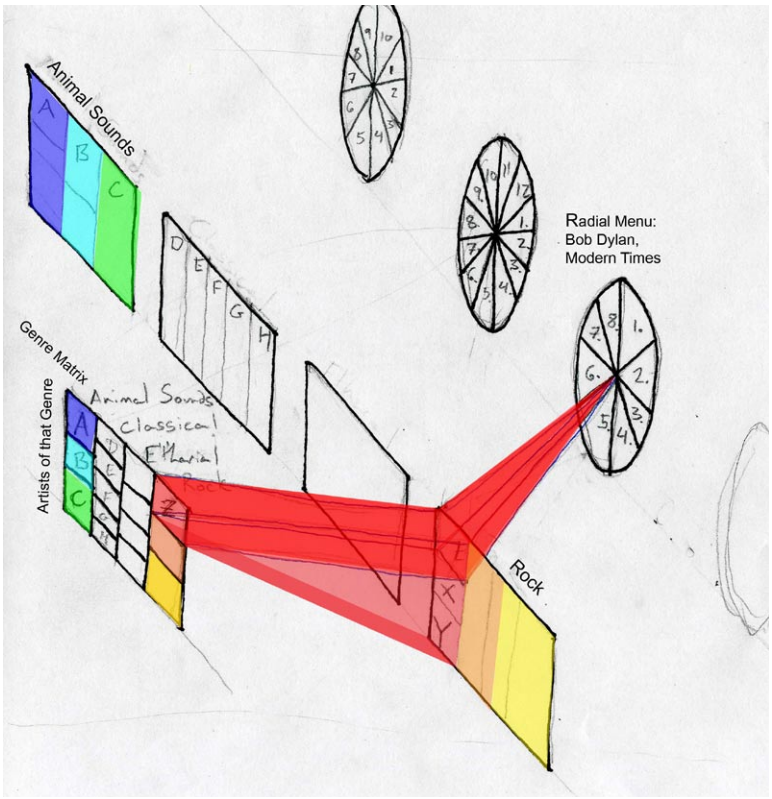
Java

All of the core software development during this research was with Java. Java's fast, there's a massive developer community, Sun provides good support documentation, and it's OS independent. The selection of the Java Media Framework (JMF), also developed by Sun Microsystems, seemed the appropriate choice as an audio rendering engine. While Sun discontinued development on the JMF in 2003, by that time the Java Media Framework was reliable and thoroughly documented.

Spatial Interface

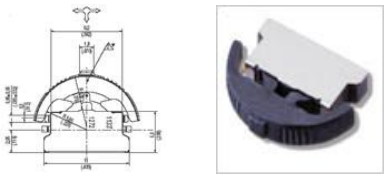
The goal of developing on the Terrella 6 was to create a scenario that supported the retrieval of situated "virtual objects" in real space. Over time and study, the shift was made from conceptualizing this environment as a sphere around you, to a wall in front of you. The reality remained that the system still used spherical coordinate mapping internally, but it was much more effective to communicate the wall metaphor to users. No matter that the diagram looked like a plane, the internal model still suffered the complexities of spherical mapping (*discussed later*.)

Early sketch of floating applications in space.



This is a diagram of the second interface structure. It was used to communicate Spatial Player's interface to users, though it belied that the system still employed spherical mapping internally.

A more mature interface, it now seemed more a kind of 2-axis, row/column, tree structure represented at the top level as a wall. This "wall" is divided into four longitudinal regions, where each region represents a genre. This allows quick browsing along two axes. Each genre is then divided into varying numbers of sections, depending on how many artists in that genre your library contains. Representative songs from artists of that genre are then arranged vertically.



ITT's three position scanner switch mounted to the front of Spatial Player

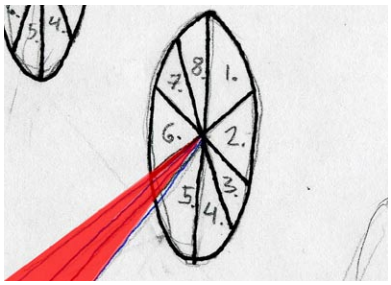
Once browsing, a brief downward press of the 3 position scanner switch, lets the user select, or "lock-on," to an artist's representative track, temporarily disabling the browsing interface. Pushing forward allows you to "step into" a particular artist's library or album. To "step back" to the parent directory, pull backward on the scanner switch. Repeat as necessary to continue stepping back.

Selecting one of these albums brings you to a terminal node. Here the interface shifts from that of an orthogonal grid to a radial clock face. All the songs are cast around this clock face structure – 12-1 o'clock is the beginning of the album, 6 is the halfway point and 11-12 the end.

This clock face menu structure was the most effective eyes-free interface explored. It was easy to verbally communicate the clock metaphor, and because of it's ingrained familiarity, users found it easy to envision in their mind's eye. This level of visual familiarity allows quick, intuitive

understanding and relative gestural confidence in the overall positioning scheme of tracks within an album. This was a glimpse at attaining some of the "eyes-free" functionality Spatial Player sought.

It was the success of the radial distribution of tracks led to our use of the donut menu in the Beat Browser.



Audio Rendering

It took another month and multiple software architectures tried and quit, before finally developing a sufficiently responsive and smooth listening experience. What quickly came into focus was the nature of human gesture and movement – it's gracefully "analog" nature, imprecision and whimsy. Mapping auditory events to these movements across time and space had to be similarly organic and forgiving.

Audio latency, and the drift associated, was quickly identified as the biggest problem users faced in building any kind of robust and trust-worthy mental map of the interface. Gestures tended to be *sweeping* in nature. If there was any latency during these sweeps, the user's hand would be long passed the region that triggered the audio event by the time the appropriate music played.

Gesture and response had to be instantaneously coupled or the edges of the regions would quickly become unidentifiable. It took the Java media players too long to switch between songs so each song was given its own media player. When triggered on-mouse-over, the designated player would begin the designated song. Instantaneous start-up was achieved the old fashioned way.

This brute force fix revealed a surprise – not all of the latency was due to bufforing or the opening of a new player or song. The contribution of the silent lead-in (usually on the scale of milliseconds) was not negligable. Ears don't blink.

The simplest patch was to launch all of the players simultaneously when Spatial Player launched – just play everything muted and un-mute according to pointer position. That way, all the songs would get over the lead-in after the first couple seconds.

In "solving" this issue, the idea of the Universal Time Base, the illusion of concurrent playback of your entire collection, emerged. The Universal Time Base is discussed at further length later as the central playback concept, which, appropriately began as a a percieved quick fix that had to be addressed later. Fittingly, it took playing with the system extensively before this patch revealed itself as a feature.

Sound Design

Testing quickly revealed that entering these songs already playing at full volume was too jarring to the ear. To ease this, volume ramps with dynamically tuneable time lengths, were implemented to control the attack.

Given that this interface was not visually reinforced, the goal was to use to manipulate sound to communicate as well, not just contour it to make it easy on the ear. In particular, users needed an intuitive way to determine their proximity to the edge of the currently occupied region. It was important to know if you were in the heart of the region, or about to fall out of it.

What was previously an invisible wall was re-imagined as a kind of plush upholstered banquette or diner booth. The seams were the region boundaries, the upholstered contour mirrored the gain contour. As pointer position approached the seam, the gain dropped to zero.

Based on the number of regions (which could be dynamically set to the users preference or file structure), a similar, dynamically calculated ellipse filled out the gain envelope.

Testing proved it was close, but a near-miss. The *MAX_VOLUME* region had too small an area, and at the edges the `currentVolume()` dropped off too quickly. The first problem was that if "locked" at a lower volume position, songs remained at this lower volume, even though they were selected expressly for the purpose of listening at full volume. In response, a gain "zoom" audio animation was tied to standard "lock" button functionality.

When you "lock" on a song, the song automatically ramps, or "zooms," into full volume. This meant taking the current volume of wherever you were in the region, and ramping that up to full volume. Or, if you were leaving the locked state, you would be ramped down to the appropriate volume of that song in your current position in the region. If you found yourself in a different region at this point, unlocking would ramp down the previously locked song to zero, and immediately pick you back up in an upward ramp into the current song at its region-appropriate gain value.

Second, the smaller *MAX_VOLUME* region dropped the average loudness of the interaction experience to a sub-optimal level. The *MAX_VOLUME* region had to be maximized, this made browsing around more fun. That said, communicating region-location information through gain was still a priority, testing had proven it intuitive and effective.

The response was to take the upholstery metaphor and stuff it. The edges became positive and negative sine curve volume ramps calculated as a percentage of the entire region to allow appropriate scalability. This percentage was a variable, which can be manipulated to affect slope grade. This flattened the top of the regions for a significantly longer stretch, making each region more spacious and forgiving to slight arm and wrist adjustments which had bothersome audible consequences of jitter using the older strategy. The sine curves at the edges also gave a nice, aural turn-of-the-lip that was more sonically interesting than a linear ramp transitions.

Tuning

Experimenting further with different region cuts, to increase the density, it became clear that the slight movements of the arm posed a significant problem to increasing the region count. This was most observable when moving along the vertical axis. (Recall: while mentally imagined as mapped to a wall, the actual computation model of the interface was mapped to a sphere.)

It is the natural decrease in region size as they approach the top of the sphere that quickly proved the limiting factor. Though lateral sweeps between the "Tropic of Cancer" and the "Tropic of Capricorn" could comfortably support many more slices of the pie (birds eye view), attempting to stay within the same column as the region areas shrink if you move up or down the rows was far too frustrating. Averaging approaches were tried to smooth out the path traced by the pointer, but this still did not overcome the problem. Chris recommended employing hysteresis, which appears to be the best strategy, but has yet to be successfully integrated.

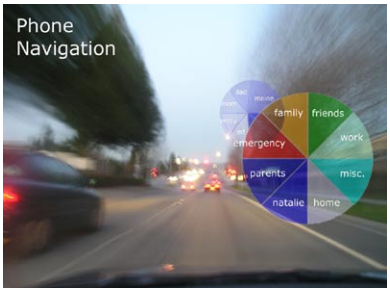
One of the other crucial features that improved overall performance was integrating on-the-fly re-zeroing. Dynamically setting the primeMeridian variable allowed users to re-orient the pointer to say that zero is where they are currently facing. The most important reason for this is the inconvenience in looking for something behind you. Users can't become a cat chasing its own tail.

Testing proved that it was much more natural to constrain the sphere to a rectangular region in front of you, as described earlier, that did not far exceed 45 degrees up, down, left or right orientation as this range of motion felt comfortable to the wrist. This region was entirely adjustable, and one could dynamically define the operable field, as well as the individual region size, as desired.

The first way the primeMeridian was set was with the press of the switch. The three positions on the scanner switch were highly valuable and the button press logically fit as the LockSong trigger. The solution was creating a scheme where if you "pushed against," or past, the edge of the region, the region would follow your pointing to redefine its edge location to the point at which you turn and swing in the other direction. That way, with a couple of swift sweeps side to side to either bounds, the active rectangular region would end up directly in front of you.

User Feedback from Sponsor Week. Disappointment.

The Spatial Player was operating as desired, incorporating all of the features described above that made for a pleasant listening experience, but this device was simply not an intuitive navigation tool for users. The "blue sky" thinking about spatial positioning and mnemonic strategies behind the project proposal yielded a disappointing instantiation. Spatial Player persistently met with a luke-warm "interesting..."



This super-imposed "pie and wedge" communicates what users would have in their mind's eye, not *what they would see literally. While driving, they point to a wedge to dial directly or to bring them to a sub-menu.*

Multiple scenarios, from navigating your cellphone numbers while you drive to music library navigation were used to help Lab sponsors understand the utility of sorting spatial menus with gesture. This attracted some interest, but trying it, the Spatial Player just didn't seem something you could put your trust in. There was always something missing.

The eyes-free interface paradigm, seemed just out of reach to everyone, myself included. Rather than assisting me in remembering some polar organizational structure of my construction, I found myself getting lost in Spatial Player's interface and not knowing how to find my way to solid ground. And if I couldn't find something, the question always loomed, was it my mistake or the device's?

Also tellingly, the overall direction of the project couldn't easily be communicated. Even today, I still can't craft a simple and concise description of the essence of the effort, let alone the device itself communicate its use and reveal its operation without explanation.

In this day an age, any interface that demands instruction is a doomed interface. You just have to *get it.*

Cell Phone Integration

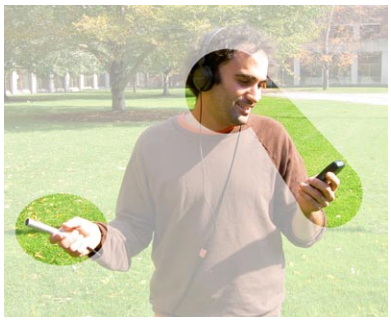
Far and away the most successful aspect of the project was the feature that I was most reluctant to implement. I had dreams of an intuitive, entirely display-free interface, but Chris lobbied for some visual feedback. He thought it'd be cool to have our device broadcasting album cover art, album title, artist and song title to a cell phone's display, something to hold for confirmation.

A Nextel iDen phone, provided by Motorola, was loaded with the .jpgs of the album covers. With the help of group-mate Jaewoo Cheung, a new Java class was incorporated to report album change events from my computer to Jaewoo's over the LAN. Jaewoo's computer dialed the cell phone directly, and once the connection was made, relayed the album change event over the cellular network. Via the nearest cell tower, the album change event was received by the user's phone along with the name of the new album cover to display. These instructions were then executed by a custom application, built by Jaewoo, that ran on the cell phone.

As seen in the images, the sensor broadcasted the relevant track information onto the cellphone, co-opting its screen.

Artist, album and song data, displayed on Motorola's iDen phones.

In your right hand you swept the pointer in front of you, across the music regions, while your left hand, holding the cellphone displayed album, artist, and song names with little to no perceived latency. But during that time, the message took an unbelievably circuitous path.



All of the pieces of Spatial Player, working together.

Findings

The metaphors of boundaries, when only associated with gesture and audio feedback, without the assistance of visible boarders, proved endlessly slippery. This demanded fewer selectable regions. As you swept your hand from left to right, the genres would change as you past certain degree points. But these points couldn't be "held" in memory by users. Degree positions were not easily "fixable" to memory. The primary observed reasons were:

1) In using the wand, particularly because of its unencumbered nature, users wildly swung the device every which way, like a flashlight. With such whimsical engagement, they would fly through multiple regions. This made memory bindings between audio and gestural position very difficult.

   The form factor of the device may in part be to blame, a heavier wand commands slower movements (for inertial reasons if nothing else), but this added weight would also make the Spatial Player a less accurate simulation of future cell phone functionality.

2) The space in front of the user, divided into a sort of regional file structure, cannot be very dense with options because of the natural fluctuations in the hand movements of users. Particularly, in moving along the vertical axis, where the wedge size naturally decreases.

   Persistent unintentional column drift due to the spherical-geometry scheme (not the sensing hardware) eroded user confidence in what is already an inherently fragile mental map for file retrieval. Even with hysteresis, which could "force-swell" upper and lower regions, users would then have to overcompensate by several degrees from the actual bounds of the wedge if they actually wanted to move across along an upper or lower row. These "sticky" edges further complicate internal representation.

3) Averaging to the last 3 and 4 degree readings necessary to get the smoothest data produced too much lag, whereby it took the system a moment to catch up with pointer position. Gyro-stabilization is a probable, though very expensive, solution. It would approximately double the price of the hardware.

4) Multiple button capabilities were harder to learn without visual response.

5) Mental maps were constructed as relationships between

sounds, generally not formulated from gestural orientation or proprioception (e.g. “This song is next to that song”). We were asking users to create a mental model of a file structure, using only audio and gesture. This set the cognitive load higher than expected.

We could explore the use of broad, four-square regions where users point to different quadrants to decrease this load, but recognition at such a large scale could be done with much less expensive sensing.

Spatial Player Comments

After some reflection, two threads from the Spatial Player stood out for further research. First was integrating 3D audio spatialization. Wearing headphones, this would allow users to situate virtual sound sources in space. For the Spatial Player, this meant a song’s coordinate position may then match its perceived sound source location -- a song retrievable by pointing above-and-to-the-right, will seemingly emanate from there. This held hope as a way to create more compelling “bindings” between gestural interaction and auditory feedback.

The second thread was audio rendering efficiency. Audio had to play immediately and concurrently (to minimize quiet openings to songs) if users were to understand it as a boundary marker. Spatial Player overcame this by playing all of the songs once the application was launched, so that they all passed the start-up and song beginning stages as fast as possible, and then muting and un-muting them as you traverse regions. While playing all the songs simultaneously provided instantaneous audio feedback, it was no solution. It was neither efficient nor scalable. Could this criteria be met without sacrificing the interaction experience?

c) Beat Browser3D

Objective
Develop working 3D Audio on a simple platform with the intention of porting it to Spatial Player.

Beat Browser3d was built as a stripped down development platform to explore question one. Could 3D audio help Spatial Player become more intuitive.

Beat Browser3D was not originally intended for show – it was a simplified I/O that used a mouse instead of the sensor and it wasn’t “eyes-free.” There was an explicit visual representation of the album field the original Spatial Player asked users to imagine themselves.

It has a simple display matrix, mapped to a flat plane, not a sphere, where album covers represented audio that could be moused over (akin to the

Beat Browser3D with planar spatialization based on mouse position.

sweeping of the arm) to activate audio playback. The nine square grid of albums all played simultaneously, muting and un-muting according to mouse-focus.

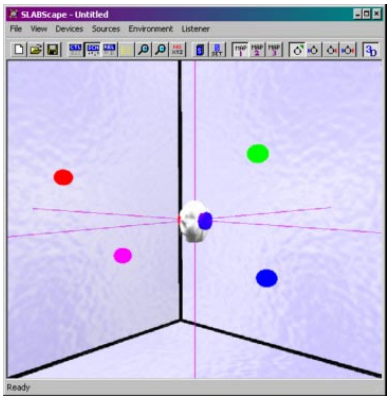
If you clicked an album, you committed to explore that album's songs. At that point all of the nine players running would close down, and one player for each song on the album was opened and represented as one wedge in a pie-menu, called a Halo (recall Spatial Player's clock face), where the wedges are dynamically divided based on track count.

The goal was no leap of faith or spike in cognitive load for this application, you were never operating without a visual net, Beat Browser3D was just the basics to keep focus on building a new audio architecture. But even this early, with just the mouse-over activation, crude concurrent playback and album-art based visual display, with no 3D audio in operation, Beat Browser3D's simplicity began to outshine the Spatial Player.

SLAB

Introducing spatialized 3D audio into Beat Browser3D exceeded the capabilities of the Java Media Framework. There are several open source 3D audio rendering packages out there. After lengthy comparison with

headphones, I chose NASA's SLAB (Sound LAB), spearheaded by Joel D. Miller at the Spatial Auditory Displays Lab at NASA Ames Research Center.



NASA's SLAB 3D audio spatialization suite includes a nice stand-alone application that was useful for experimentation before joining the two applications with code.

What set SLAB apart from the other open-source packages was that it was expressly constructed with psycho-acoustic experimentation in mind. Though the breadth of parameters incorporated into the system far exceeded needs, from constructing virtual sound spaces to manipulating the materiality and sound absorption co-efficients within them, this ultimate control was appealing when considering tuning or otherwise manipulating the prototyped listening environment.

The main issue with SLAB was that it was written in C++. This necessitated the design of custom Java Native Interface (JNI) to bridge SLAB methods and Beat Browser3D code.

This class was designed as a neutral JNI to support the complete functionality of SLAB's C++ interface. Each method or function was given a Java hook. Beat Browser3D's native interface supports far greater methods than are actually called, but it was ultimately the cleanest implementation. It was generalized code that put in place all the communication channels necessary for future expandability, and was Java code that could be contributed back to the SLAB project.

Implementation

In time, Beat Browser3D successfully communicated with SLAB. Each album cover was imagined as a square tile in plan (bird's-eye) view, with the user at its center. If the mouse veered to the left side of the tile, the sound source would follow. If the mouse then moved to the top of the tile, the sound source would appear to be emanating from ahead and to the left. It was an interesting effect, particularly when traversing selections in a lateral sweep: albums would go in one ear and out the other. Also, if the Halo was open, each song would emanate from the wedge position with the same logic, as though the entire album encircled you.

Find a video showing the prototyped version of these effects at:

http://web.media.mit.edu/~jdg/

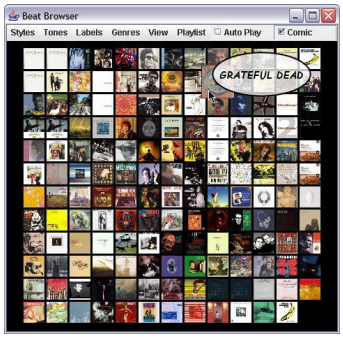
To achieve this effect, SLAB was converting these highly engineered stereo music compositions into mono. It was flattening them to then be able to position them. All 3D spatialization does this, it was just highlighted because the original signal was so heavily engineered and optimized for stereophonic playback conditions.

After experimentation, the conclusion was that while a source's location could be fairly well discerned in 3D space, the music itself sounded particularly 1D. The music was not ensconcing. It seemed worth further study for spatializing speech applications, particularly with the familiarity our ears have with spatialized speech sources, but not in this manifestation for music listening.

Besides, continuous playback, the unintentional fix, was showing itself to be the most compelling aspect of the entire audio research trajectory.

4. Beat Browser

Objective
Create a playback architecture that conveys the feeling of "live" and continuous audio, but does so with arbitrarily large libraries and remains computationally frugal.



Beat Browser, in comic mode

What seemed an incremental step from Beat Browser3D, proved the biggest design challenge: Get immediate mouse-over playback (no perceivable latency), without playing everything at once?

The first strategy explored was to "wake up the neighbors." Eight Media Players surrounding the current mouse position launched and quit on mouse movement. A *WAKE_UP_RADIUS* constant was created to easily tune this effect. When an album was in the radius, it began playing with gain = 0.0. Only when the mouse itself entered the album cover would the gain ramp up to 1.0. At the least, there were eight surrounding players running, but quickly four more were added to provide a two album lead orthogonally, while only one diagonally. That said, twelve players was not the efficiency desired.

And this strategy had latency too boot. The main attraction of the Beat Browser interface is its response. As Chris put it, it's about constructing a "live" environment. The illusion requires absolute freedom from latency, no matter how you play with it. But the wild, trans-library sweeps as users inevitably test the boundaries of experience, chocked the system. The edges of latency came much faster than the edges of the library itself (which were set as a respite of silence). Players weren't waking up in time, and it was still heavily processor intensive.

From Troubleshooting to the Final Design

Debugging the situation a little deeper, `MediaPlayer` revealed the time line of events that were occurring during the periods of latency. At any moment, JMF's `Controller` interface can be queried to find out what state its in. There is a life cycle to the `Controller`, and it has five methods available that induce life cycle state changes: realize, prefetch, deallocate, syncStart, and stop. To transition a `Controller` to the *Stopped, Prefetched,* or *Realized* state, you use the corresponding method: `stop(), prefetch(), or realize().`

Experimentation determined the latency between *Prefetched* to *Started* was undetectable on the 1.8Ghz processor used in development. Testing revealed there was no lag in accessing the `Clock,` which is called by `setMediaTime()` to control playback positioning. Beat Browser could go from *Stopped* (in *Prefetched*) to *Started* (at any point in the file) without latency. This pointed to the latency occurring somewhere in the state transition from *Started* to *Stopped*. The clog was in flushing the songs.

Further experimentation confirmed a processing period of 200 milliseconds is necessary to `stop()` players. It seemed counter-intuitive, but `MediaPlayer` took all the time to stop, not to start. Sun's choice in weighting the entirety of the processing demands in the *Stopped* state transition later proved itself a well thought out decision, but frustratingly it's not explained in any logical place in the documentation.

After finally learning what was happening under the hood of `MediaPlayer`, and that it was in the closing that the latency occurred, the final architecture was worked out.

Beat Browser hides this shutdown latency by creating a multi-threaded structure, dividing work into two tasks. Opening threads are created for the transition to the *Started* state and closing threads were created to transition to *Stopped* state.

The closing threads are where all the heavy lifting is done. Surfing Beat Browser at speed, there may be anywhere from five to seven closing threads in operation with just one opening thread. `MediaPlayers` are opened in one thread, and immediate playback is experienced, but the instant the mouse leaves that active region, that individual `MediaPlayer` object is passed to a closing thread which, after approximately 200 ms of work, puts the *Stopped* player in a cache.

The cache manages (limits) the creation of a new `MediaPlayers`. It stores all of the opened `MediaPlayers` resting in the *Stopped* state. The cache is the first place `SongPlayer()` looks for `MediaPlayers` to limit the resources and time allocated to creating them from scratch.

SongPlayer

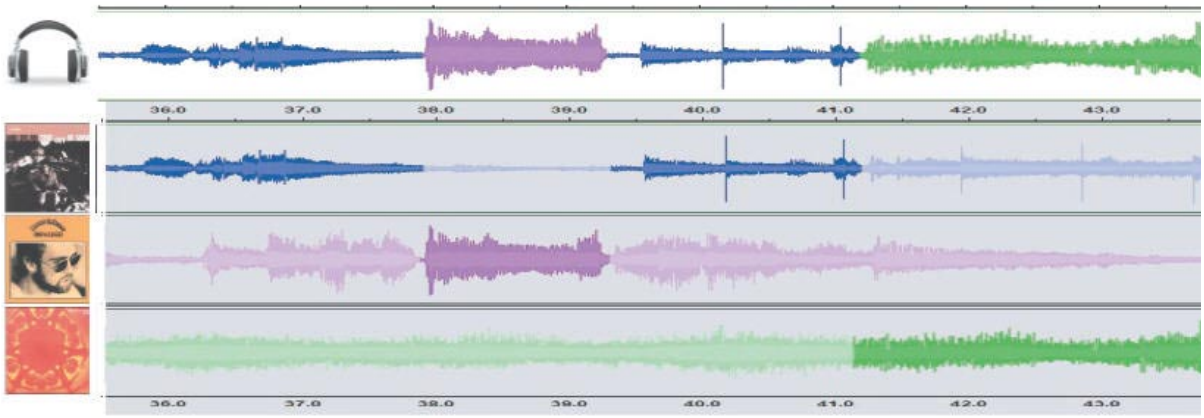
`SongPlayer()` can be thought of as a telephone switchboard in its architecture. It creates the "jack" to each piece of music that the `MediaPlayer` can be plugged into.

`SongPlayer.java` class is the heart of Beat Browser's playback engine, handling the entirety of the playback responsibilities. An instance of `SongPlayer()` is generated for each album and individual song in your collection. For my audio library, over a thousand `SongPlayers()` are created on start-up. `SongPlayer()` can be imagined as the "jack" in an arbitrarily scalable telephone switchboard.

Just as each song has a `SongPlayer()` plug, each song has both album art and a donut wedge as its visual representation. These graphical representations bound a display region that the `BeatBrowser.java` class monitors. When one of these areas is traversed, the `(main)` class determines the song's appropriate coordinates and communicates them as index values that "light up" the appropriate `SongPlayer()` "jack." `SongPlayer()` queries the cache for a `MediaPlayer` to render the audio. If there aren't any, `SongPlayer()` will create a new player and playback begins.

The bottleneck of the earlier efforts was trying to do all this changing over one line – *Starting* and *Stopping* one `MediaPlayer` in one thread. In this scheme, changing the songs could could only happen after the elapsed time between when `stop()` is called and when the `TransitionEvent` is posted by the `Controller`, indicating the `MediaPlayer` is in the *Stopped* state, typically 200 milliseconds.

Multi-threading divided and conquered the latency, but still incurs a cost. It's resource intensive to create a seamless audio experience when users rapidly surf, but these spikes quickly fall off after songs are listened to for any length of time, and abandoned `MediaPlayers` can begin to fall into the cache in a 200ms cascade.



The Universal Time Base is represented graphically above. In this particular image, we see the user's listening focus (bright) and their navagation through time articulated: first, the she tried a Bob Dylan, then went to Elton John for a spell before returning to Bob Dylan and committing to a "locked" state. This allowed her to mouse around until she happened to notice Boards of Canada, and double-clicked on their album. thereby unlocking Dylan, and engaging in the new record without rolling back over Elton.

UTB

It was crucial to maintain Beat Browser's illusory concurrent playback – it was a subtle, yet convincing twist on current media players. It was this attribute, which Chris defined as the Universal Time Base (UTB)

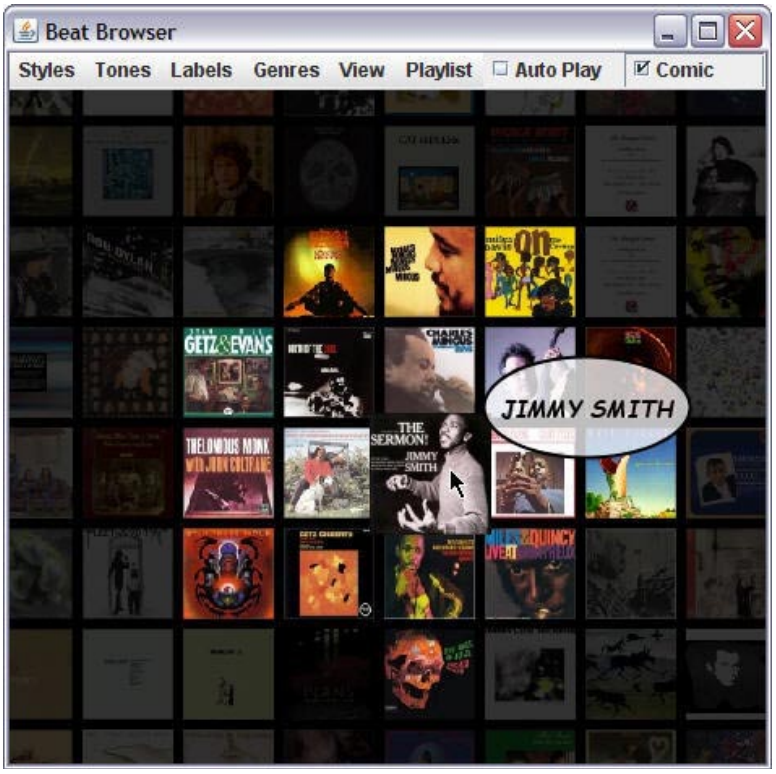
which gave the interface its immediately responsive, old-media (*think radio*) feel.

The Universal Time Base was integrated once latency between jumping between players was without perceptable consequence. The UTB is another name for the central `Clock` interface for the application. `SongPlayer()` queries the Universal Time Base, calling `getMediaTime()`. `SongPlayer()` then performs the modulus of each song, taking the remainder as the distance in seconds into the track that playback should begin from.

It is just as feasible to take the modulus of the entire album, or some created playlist, going as far into that sequence, be it the first or 15th track, as the elapsed time dictates. `SongPlayer()` then takes this calculation and uses the "setter" method `setMediaTime()` to designate playback position. `volumeRamp()` is called and playback becomes audible. It is this sequence of events that conveys the central, river of music metaphor, the illusion that your universe of music is playing at once.

Features

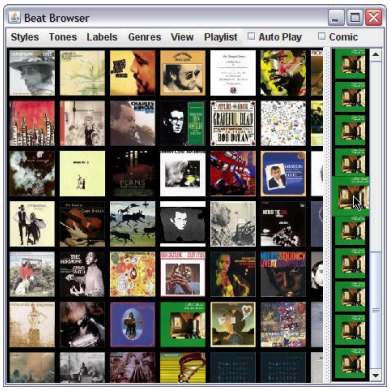
Beat Browser.java also trolls through your file structure, looking for .xml data associated with albums and songs. This is a standard meta-data format provided by AllMusic's All Media Guide (www.allmusic.com) and allows the organization of albums and songs, dynamically managing data allows `MatrixDisplay()` to redraw album covers in clusters of relevancy on-the-fly.



The genre clustering in action: Jazz

Auto Play emulates scan on the radio. It steps into each visible album alphabetically. If you have clustered a certain subset of albums, Auto Play will only step through those selections in alphabetical order, as those are the only visible choices.

Finally, songs or entire albums can be throw into the playlist with the right mouse button. A right click with the Halo closed adds the entire album in focus, a right click in one of the Halo's wedges will add just that song. Mousing over selections in the playlist plays them back for review. Right clicking the in the side window removes them from the list.

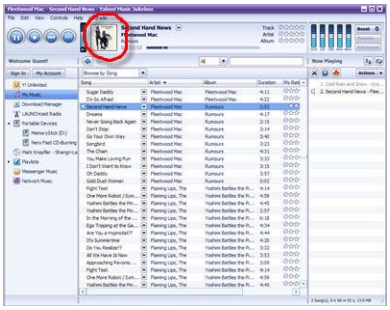


Nick Drake's Five Leaves Left, in its entirety, in the playlist

Appearance

The most striking visual attribute of the major players in this space is their lack of cover art. All of these environments for music listening construct an elaborate frame through which to view your music as a text list. Exploring music has become a highly literate affair.

The notable exception is iTunes' CoverFlow, orignally designed by the folks at Steel Skies (www.steelskies.com). This "breeze-through" album art navigation window begins to cede some screen real-estate to cover art, but it in no way competes with the Apple® look and feel.



What percentage of this screen is devoted to the album art?

Music players and music browsers shouldn't be about the brands. These managing and rendering application environments need to get out of the way. Artists have spent a lot of energy assembling the covers. Album art is the perfect face to your interface – a collage wholly dependent on each unique library of music.

Let the album art "speak" for the artists, like Beat Browser let's the music speak for itself.

5. Design Principles

Several decisions and strategies were consistently reinforced throughout these projects. "Design Principles" catalogs these audio browsing factors so they may find broader applicability.

Combat Fatigue

No jarring sounds; Surprises are tiring.

"Everything in Modulation"

Don't peak normalize; Set perceived loudness levels equal across the library.

Beat Browser uses pervasive volume ramps with adjustable slopes to uniformly contour the attack of new sounds, and pre-processes its audio library with the help of the MP3Gain project.
http://mp3gain.sourceforge.net/

High Exposure to Time Ratio

Visually: Tile selections to maximize screen space, provide zoom-out for 10,000 ft views plus easy zoom-in for confirmation.

Aurally: Implement audio animation defaults or other "Lazy Susan"-style interfaces.

Gesturally: maximize response for minimal physical activity.

Visually, Beat Browser employs the zoom +/- feature set; aurally, "Auto Play" is browsing on auto-pilot; and gesturally and a single gestural trajectory opens and closes multiple selections

Lazy Susans spin on bearings, bringing the table to you. Why aren't they more popular?

No Latency

Motivation is fragile, patience fleeting: instantaneous audio start-up is a necessity.

If this means using audio thumbnails that sacrifice quality for start-up time, *do it*.

Beat Browser eliminates latency through multi-threaded player management, looping and UTB start-up. The Universal Time Base strategy of dropping in the middle of songs skips the silence when songs play from the beginning.

Ramps

...volume ramps are kinda like ellipses...

...swelling or fading...

...keep all things gradual...
...smoothly transition between passing streams...

...or acoustincally communicate a "zoom" or "motion" effect...

Ramps are everywhere in Beat Browser, segueing between two gain levels, fading in and out of graphics and for swelling them. Every hard edge, of every kind, uses or tried to use ramps to smooth it.

Time is Elastic

Play with time in time-based media.

Are we "Live"? Or is this a pre-view or post-view?

Beat Browser's Universal Time Base (UTB) architecture simulating simultaneous playback is one strategy. Songs and or albums begin playback according to the time elapsed since Beat Browser's launch.

Loops are Your Friend

Loops limit silence.

Don't consider time linear.

Looping is an integral playback property of Beat Browser. When using visual signifiers for audio information (e.g. album covers representing album tracks) looping individual tracks "underneath" their representative image maintains the feeling of movement in the interface. In the Universal Time Base's "music as river" effect, looping is a central ingredient.

A Good Browser is an Ecology

Design pathways for flexible, easy and independent dialog between objects, people and things.

Create the sandbox for self-organization to occur.

Beat Browser uses metadata stored in standardized .xml file formats per the All Music Guidelines, www.allmusic.com. This is the "personal" information shared albums use to form clusters of similarity. While not implemented, tagging support is needed to allow alternate naming and organizing conventions contributable by any author.

Accommodate Serendipity

Design-in opportunities for surprise, orchestrate as many "situations" of overlap and proximity as possible.

Sweeping across the matrix, albums shift in and out of phase, and lyrics that begin in one album bleed into the next and the next.

A couple of clustering filters put Lionel Ritchie's "All Night Long" right next to KISS' "Rock n' Roll All Nite." Mousing between Ritchie's chorus, "All night long, all night...." to KISS' "I wanna Rock n' Roll! All nite!..." triggered:

Lionel Ritchie and KISS, together again

KISS' shouts of "I wanna Rock n' Roll!.." segueing fluidly into Ritchie's soulful "....all night, all night; All night long....."

Fast and Loose + Slow and Ordered

Simultaneously; Encourage fast and loose interactivity and accomodate slow and ordered engagement.

Casual engagement with Beat Browser could be flying through your library with the mouse at a party, looking for a quick musical fix that'll make everybody happy, setting the mood just right when you're out of ideas.

Simultaneously, allow for more structured interaction, such as playlist construction, where a the goal is to create a well-formed and expressive composition of music selections.

Scale Independent, Platform Independent Operation

Support small scale and small screen interaction.

Exploit large screens at large scales.

Software is running on a wide range of hardware. Beat Browser can be zoomed on to fit an entire collection, with full functionality, to the screen size available. Album titles and other text reliant displays are not scale independent like album covers are, point sizes only go so small. Color schemes and general features of cover images remain recognizable at very small scales.

Fun

If it's not fun, what's the point?

If Beat Browser's not fun, what's the point?

The Power of Timbre

"Pop musicians compose with timbre. Pitch and harmony are becoming less important."

Daniel Levitin,
James McGill Professor of Behavioral Neuroscience and Music
McGill University

Daniel Levitin, Ph.D.

Timbral qualities are amplified with Beat Browser's lateral freedom and side-by-side UTB playback.

Beat Browser's metadata-constrainable "Auto Play" provides perceived mood and timbre filtering, without signal processing.

Encourage Discovery

Lead users into the unfamiliar, the new, and the forgotten.

Set the barrier to new music discovery to null – in time and effort.

Beat Browser sorts with 2D proximity easily allowing consequence- free roll-over taste-tests.

Tuning

Everything a variable, from constants, to volume to animation intervals.

Every interval, level, and rate is adjustable. Audio interfaces are an instrument, of sorts. They need to be tuneable as such.

Boundaries and Edges

The boundary between regions is fertile for aural demarcation.

Beat Browser has explored volume ramps, delay and blurring, and panning across the Left and Right channels to communicate song boundaries. Ultimately, contours to milliseconds of silence were chosen.

Fluidity

Your ears don't blink.

This applies to everything.

Consider Silence

Silence should be easy and fluid to engage, but not by accident.

Silence shouts louder than a looped song heard over and over again

(This works when the repeating segment is of a sufficient length, broken records shout the loudest of all.)

Sweeping into background of the window triggers silence in Beat Browser. Loops are the default, keeping everything in motion.

6. Related Work

a) Spatial Player

Stephen Brewster, et. al.

Beyond Cohen and Ludwig, it has really been the remarkable work of Stephen Brewster and colleagues at the University of Glasgow's Multimodal Interaction Group, that has truly advanced the state of the art in gestural audio interfaces. Particularly, their work "Audio Clouds," presented at CHI 2005, has deeply explored the spatial positioning of different objects and applications to create intuitive, "eyes-free" computer interfaces. Brewster's work, outlined in broad vision in "Multi-modal 'Eyes-Free' Interaction Techniques for Wearable Devices," incorporates radial menus, spatial bindings and auditory icons in creating a robust, "eyes-free," (display-independent) navigation system. These are many of the themes Spatial Player explored.

A conceptual sketch of Brewster's work, created by his lab at the University of Glasgow.

Brewster's work also includes spatial audio processing for added effect. Referred to as 3D audio this is the method of dynamically positioning sound sources at specific locations with respect to the listener. With the use of headphones, this creates the convincing illusion of event locations all around you at a surprisingly high level of resolution along the horizontal plane.

Determining sound source location along the median plane is much more difficult. If a sound is emanating from a point five feet away, at a 45 degree incline, or 45 degree decline, it is far more difficult for listeners to discern with high confidence or precision.

For any acceptable discrimination along this plane and higher resolution

on the horizontal, head related transfer functions (HRTFs) must be created for each user. This means building individual profiles of the unique body characteristics, particularly with respect to head, ears and shoulders, as they play an essential role in sculpting the uniquely familiar beam reflections we have learned to associate with location. And even with professionally created HRTFs, the vertical resolution nowhere near approaches that of the horizontal axis, let alone natural listening.

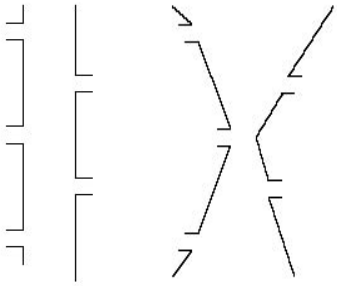
Outside of audio rendering, a similarity that sets the Spatial Player and Brewster's Audio Clouds from the current mass market products, is the use of magnetometers coupled with accelerometers. An expensive route, this creates a tool that constantly calibrates itself based on the external anchor of the Earth's magnetic field. Consequently, both are highly drift-resistant.

Chris Schmandt

The other landmark paper during this research process was Schmandt's "Audio Hallway." Initially Audio Hallway was intriguing in its exploitation of familiar architectural metaphors, re-imagined in solely acoustic aspect. Audio was processed to construct a virtual environment. Users could walk up and down via head movement, passing open doors out of which different sound streams spilled. These rooms, containing news broadcasts and other audio content, could then be entered for deeper, more intimate engagement.

It was with the subsequent design improvements to craft a more realistic and familiar user experience that the most personally compelling aspects of this work revealed themselves. Schmandt identified and addressed several curious sound design and audio perception issues encountered in designing convincing, non-visual architectural spaces. One observation in particular had interesting implications:

> *"With increasing distance from the head, doors are positioned further to the sides, to help maintain lateral acoustic discrimination. [With this modified hallway, people] ...understood the hallway metaphor, they were more easily able to experience the desired spatial audio configuration."*
>
> Audio Hallway, Schmandt, 1998



To the right is the audio transformation, visually articulated, necessary to create the audible illusion of a standard hallway, seen at left.

To create the visual illusion of a hallway, one-point perspective dictates parallel lines convergence on a central vanishing point. To create the auditory illusion of a hallway, Audio Hallway identified a curious perceptual phenomena: the most effective audio transforms to achieve an acoustic interpretation of perspective were in fact an inversion of visual perspective. Audio converges to point at the listener's head, not in the distance.

In Schmandt's scenario, the walls dynamically converge on your position in the hallway, not a distant point approaching the horizon. Acoustic fall-off is exacerbated, receding both ahead and behind. This wrinkle highlights the near-range cues we expect from our ears vs. the

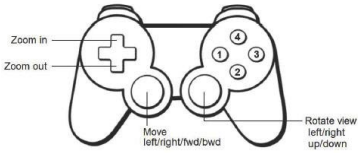
predominantly distance-oriented feedback from our eyes. In fact, to compensate for the lack of visual accompaniment, for the sighted user to consistently interpret and maintain this mental hallway model, the gain on natural experience had to be turned way up.

Traditional one point visual perspective, seen in elevation. Image courtesy of Jerome Agel.

b) Beat Browser

Inside Academia
Knees, et. al.

In 2006 Knees et. al. introduced an extraordinarily feature-rich application (unnamed, we'll call it the Knees Player). In the Knees Player, you fly through your audio library, visually represented as a set of islands at sea, via a video game control pad. It is an immersive browsing interface atop an individually generated topographic self-organizing map (SOM) based on your music collection.

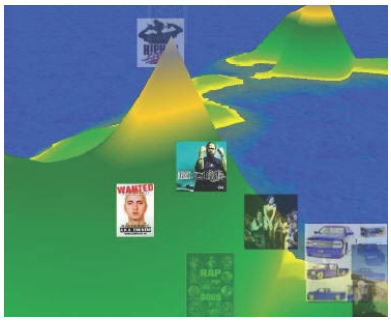


The Sony Playstation video game controller the authors use to navigate the Knees Player environment.

The SOM is constructed through signal analysis to cluster similar sounding pieces together, forming islands and mountain peaks of acoustic terrain. As you navigate your audio collection, the closest sounds are delivered using 5.1 surround sound. This creates an 3D audio environment that reinforces the topography.

Response to the Knees Player

This work, I feel, may be considered the extreme in the music browser realm. The number of effects and facets incorporated is so numerous, the Knees Player seems more about the experience of the browser itself, than about the discovery of music. The Knees Player takes many of the threads introduced in previous works in the fields of music information retrieval and audio navigation and weaves them into a unified experience. Works I am referring to include, the SoundSpace browser (Tzanetakis, 2001) which explored multiple phenomena regarding 3D spatialization, Pampalks concept (2003) of Islands of Music, and similarity functions and visualizations explored with the Sonic Browser (Brazil).

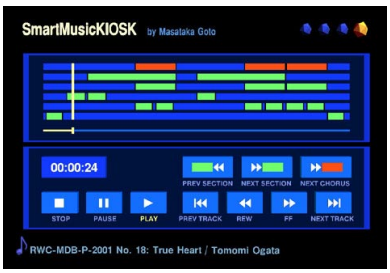


The self-organizing map (SOM) coupled with album cover art.

While 3D Audio creates novel experiences, as we've discussed, it necessitates flattening audio streams into mono output, which, in concert with multiple sound sources streaming at you creates an unfamiliar effect – though perceived in 3D, the music feels 1D.

We have often experienced the chatter of voices at a cocktail party, and picked out a word or phrase, but most of us have never heard the cacophony of competing musical streams. They muddy themselves in ways much harder for the unfocused ear to disentangle without attention. While the 3D experience is enhanced visually and aurallywith significant richness, musically (assuming the sound sources are professionally mixed compositions) it sounds thin and vacuous, where a single stereophonic source is ensconcing.

Goto, et. al.



The abstracted song structure view of Goto's SmartMusicKIOSK

The SmartMusicKIOSK, work conducted by Goto et. al., is an interesting approach to browsing. The SmartMusicKIOSK is designed as an in-store listening station. The author identifies that during "short trial listening" in music stores, customers often search out the chorus or 'hook' of a song using the fast-forward button.

SmartMusicKIOSK visualizes more advanced song structure information as a surprisingly intuitive "music map" to the lay eye, providing a color coded time line to further guide exploration in an intuitive way. The "next section" function automatically jumps to the beginning of specific sections relevant to that song's structure. As described above, audio analysis also supports "next chorus" functionality, indicated by the red fast forward button. While this is a very good project for intra-song browsing, it does not make significant additions to higher level browsing.

Responses to SmartMusicKIOSK

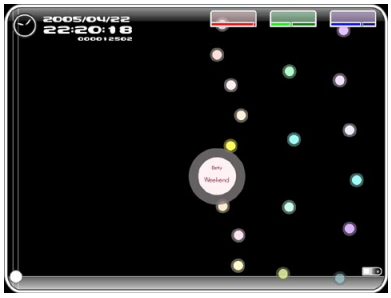
What makes SmartMusicKIOSK worth particular note, byond its visual component, is its sound design response to the observed inefficiencies of in-store music listening. It may be good for a certain type of browsing, but it ultimately doesn't attain the fun. It represents a different perspective, a more goal-oriented (towards the chorus) than music-oriented (recognizing the browsing experience itself as a musical composition).

This application also brings up the a frequent question new users have about Beat Browser: what is the value of the UTB architecture as opposed to looped chorus'? This was experimented with early on. When songs were queued to loop on chorus playback, there was a repetition to Beat Browser. There wasn't the feeling of *flow*. When you start listening from the chorus every time, you're always cresting the climax when you arrive on the selection, and the ride it down to the verse. The repetition of this scripted segment compromised the spontaneity and "live" feel of the UTB playback. The experience quickly turned stale, where the UTB Beat Browser remained fresh. There is a subtle and unique feeling when you realize on coming back to an album, the music has been playing right along, even though you weren't listening.

Musicream

Musicream (Goto and Goto, 2005) is perhaps the best music browsing environment reviewed. It incorporates a music-as-liquid metaphor and visualizes three "taps" at the top of the screen in red, green and purple, representing three descriptive parameters. The taps drop music "pieces" at a rate of your control. These pieces gradually fall down the screen, in response to some slow physics. They may also be "sticked and sorted" in clusters to introduce looser organizational relationships.

By clicking on the bubbles, you expand them, activating playback. You can easily jump, or "scrub," to any position within a track by selecting a point on the circular time line just inside the bubble's boundary. These bubbles are color coded and may be strung together like pearls, in varying orders to create playlists. You may also create "meta playlists," or playlists of playlists, with similar ease. You may also activate an "Auto Play"-like functionality by setting a bar which wakes the bubbles into playback from the beginning as they pass over it. And finally, a wide selection of "history" functions allows you to retrace the steps you've made in a very intuitive, and visual way.

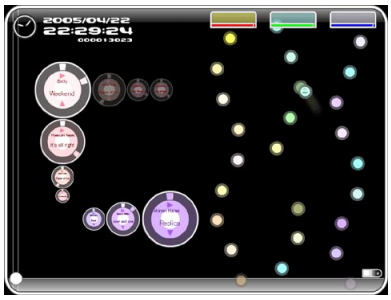


The basic view of Musicream, showing the taps dripping pieces of music, color-coded by genre, one of which is currently selected (enlarged) for listening.

Responses to Musicream

Goto et. al. have created a very nice suite of features, setting Musicream apart from players in the literature and in the marketplace. In particular, they introduce a visually intuitive representation of streaming (depicted in the origins of the name, music + stream).

As the musical pieces fall from the faucets above, your collection of music becomes animated in such a way as to foster fresh engagements with a tired library. This animation, organic in nature, facilitates discovery within your own collection. It creates a unique circumstance, a sort of eddy, within which users can drag a musical piece to snag like-songs through Goto's stickiness functions. Songs that are similar in signature, stick to each other, facilitating playlist construction.



Playlists are shown as pieces sticking to each other along different axes. The scrub bar is clearly visible lining the interior of the pieces.

Musicream makes no effort to incorporate album art into the interface. Consequently, the author's aesthetic dominates the experience. This creates a constraining effect. This is frustrating, as Musicream's contributions to playlist construction, with each song "a piece" that has certain physical properties that interact with dictate interactions with other "pieces," is very creative. It is a fantastic example of the power of creating an ecology of objects inside an application.

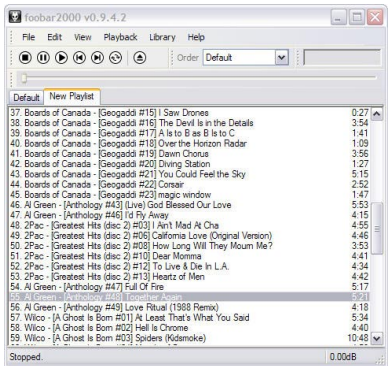
Outside Academia

Attention must be given to what has emerged in the marketplace and in the open source communities as much as that circulating in the academic and conference realms. This is particularly so when it comes to usability principles.

The earlier examples were selected because of their explorations into the browsing paradigm. These music players and websites do not do so so explicitly, but they hold lessons nonetheless.

Foo Player

The Foo Player, as the anonymity of the name may suggest, is one of the most basic players out there. It is extremely light and fast – startup time on my 1 Ghz machine was under a second. The GUI (Graphical User Interface) is all text, the complete antithesis to Beat Browser's



The Foo Player, the complete antithesis of Beat Browser visually.

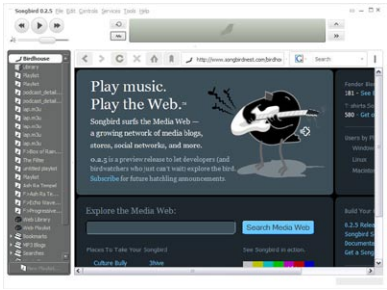
experience. This, however, is exactly what makes Foo interesting. It is *just* a player. It is lean, optimized for those times when you know exactly what you want. It is the quintessential search-based music player. It does include standard playlist capabilities and little else. The Foo Player draws very little power, making it a superior choice when multi-tasking with gaming or otherwise computationally-intensive processes.

iTunes

iTunes, the most recognizable of players, is both well designed visually and originator of the "component of a larger music eco-system" deployment model.

Apple's iTunes software has become the market leader for audio library management. A very attractive and intuitive application, iTunes ties together locally stored media seamlessly with the on-line iTunes music store. iTunes has set the standard in media management, organization and polish. It's easy to use and pretty format agnostic – rendering the GPL licensed .ogg file format. It also creates playlists automatically based on dimensions such most and least listened to, era, genre, etc. Playlist creation follows standard drag and drop procedure, but can be performed while listening to other music. iTunes is more than just an application, it is a music ecosystem for purchase, management and playback of music, extending even into portable hardware, namely the iPod and now the new iPhone.

Songbird



Songbird, an new open-source music player/web browser hybrid.

Songbird, a recently deployed open-source competitor to iTunes is the most significant open-source media player interface to emerge so far. It mimics Apple's look and feel, and provides the feature set one has come to expect. But Songbird takes a unique approach to constructing its own broader eco-system. They build their player atop the Mozilla web browser. As Songbird describes it, their player is in fact a music player to "Play the Web."

What their tagline is referring to is the growing number of music blogs that are emerging that post songs each day for your free download. Songbird is optimized for sites such as these, once you land on one, it immediately begins downloading all of the songs on the page and displaying them to you as part of your collection. What was once part of the web has fluidly found a new home in your personal library. You may then listen to them as you would any of your other tracks.

Songbird is addressing "browsing" in as much as web browsers are. Songbird is the frame through which to experience a limited web of music sites, which is a unique approach. And as the web becomes increasingly populated with media, this paradigm should prove itself further. Hopefully Songbird's user interface will evolve to keep up.

Last.fm

Falling in a slightly different category is Audioscrobbler. Audioscrobbler is the software behind Last.fm's web site (www.last.fm). Last.fm

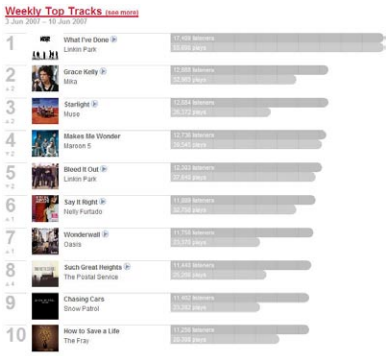
is a statistically-oriented browsing environment and music listening community. It is not just a music player per se, as its name suggests, Last.fm contains webcasts associated with each artist. It fills the playlists of all these "stations" with the primary artist of interest as the anchor, and mixes in other highly similar selections. Last.fm does not perform audio analysis, rather they perform very large scale data analysis. Data is gathered by little Audioscrobbler plug-ins, basically a satellite to the central server, that every Last.fm community member installs locally. All Audioscrobbler does is listen right along side of you with your permission.

Audioscrobbler is a lean plug-in versioned to work with all the major music players. It sits innocuously in the background grabbing the meta-data of each track you play. It keeps this log and uploads this data to Last.fm central, which in turn updates your individual profile.

Last.fm is about sharing. It is a website devoted to creating a listening community that can be browsed and experienced. Last.fm shows you what you've been listening to, how much, and recommends music you may like. It also points you toward the listening habits of other members of the last.fm community, helping people share their tastes – share their knowledge.

It creates a statistical mirror at both the individual scale, and the community-wide scale. One of its greatest qualities is that Audioscrobbler is symmetric. It shows you everything it takes, and it shows it to you in a pleasant, visual way with bar graphs and personal charts. They then repackage these profile statistics for easy posting to your blog or website, helping you share listening habits with an even larger audience.

This transparency has built trust with users. Users, along with the Last.fm central server, learn from their Last.fm's profile. It creates a very viral atmosphere for music and has become a highly trafficked site with a cultivated listening community. The amount of data they have "scrobbled," and that data's ability to make effective listener recommendations via individual artist webcasts or data clouds, is remarkable. If there is a lesson to be taken from last.fm and Audioscrobbler, it is the importance and power of transparency in community building applications.



Last.fm's statistically oriented nature is on view above: the aggregate listening habits of the entire community are displayed as Top-Ten lists, among other formats, on the home page.

P2P Radio

Peer to Peer Radio (P2P Radio) adheres to the letter of law, by streaming content from one user to another. Mercora (www.mercora.com) is P2P Radio devoted to the sharing of playlists. It is also an example of P2P Radio. You can't download any part of the webcast playlists, though downloads for purchase are made very easy.

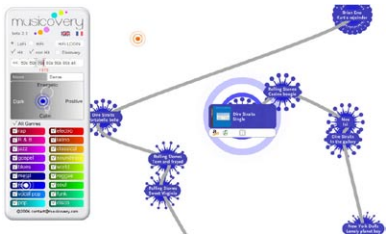
What P2P Radio websites do is create a virtual environment where communities of users can come to DJ for each other and those visiting the site. It is a remarkable, free resource. At its simplest, it is DJing over a distance, each member acting like their own internet radio station, with

This is the Stevie Wonder area of Mercora, where the list users in the right column is responsible for streaming the song titles in the left column.

Mercora hosting. You can kick off any of these playlists at your leisure, or jump to individual tracks within them. The difference between these communities and those of Rhapsody and Yahoo! Music Unlimited and other subscription services is that Mercora is just the host, providing the "airwaves." Mercora is only as good as its membership.

As a listener, you don't have complete choice. You can only access those songs members of the community have made available in their library. If you search for Bob Dylan, his whole catalog won't appear, just those songs community members are interested in including on their "webcasts."

Internet Radio



Musicovery is a nicely navigable example of internet radio, streaming at lower qualities, but ensuring near-instant playback.

Musicovery (www.musicovery.com) is a nice example of a graphically pleasing webcast interface that employs a similarity engine and instantaneous playback. Their strategy is to stream the music at very low quality, 32 kbs versions, offering a subscription for CD-quality streams. Web radio companies are allowed under law to broadcast whatever music they like without asking permission, as long as they pay copyright owners about seven one-hundredths of a cent per song streamed. But that open-ended invitation comes with restrictions: Internet Radio does not let listeners choose specific songs at specific times, they do not permit copying of broadcasts, and they have strict limitations on how many songs by a single artist they can play back to back.

Pandora (www.pandora.com), another popular option, can be remarkable in its ability to determine your tastes. Their propriety database of human generated metadata, collected as part of their "Music Genome Project," is very impressive. Pandora initially asks you to enter a song you like. They follow with their guess of something you will like, and from my experience, they are frequently correct. But if not, you click thumbs down to help the system hone its model of your taste.

One of the most heavily visited internet radio sites.

7. Future Work

Short Term

The Codec

If Beat Browser is to be deployable in any larger sense, abandoning the Java Media Framework will be necessary. There are two reasons for this:

The first, as described earlier, is that work on JMF ceased in 2003, leaving several newer open source formats such as .ogg, and .flac unsupported. Fraunhofer's Mp3 format remains propriety, but also the industry standard. Until this becomes license-free, full support of the work of the Vorbis Foundation (the folks behind .ogg) and others is important.

The second is that while Beat Browser is computationally intensive in only sharp spikes, the highs are too high. These peaks need to be brought down by using a leaner codec. The Java Media Framework is exactly what it says, a *media* framework, supporting both audio and video coding and decoding. Beat Browser doesn't need the added weight of `MediaPlayer`'s video functionality, particularly when multiple are being opened or closed near simultaneously.

Music Sorting

While Beat Browser does acquire metadata based on a global standard, it does not yet support tagging. Where Allmusic.com is based on the controlled vocabulary which is the heart of their top-down taxonomy,

tagging is the bottom-up ability to ascribe personal attributes to each piece of music. While it is necessary to have a default organizational scheme like allmusic in place, it is just as important to have manual override – the ability to organize your music based on your opinions.

There are also massive on-line data bases of tags available through open source websites such as Music Brainz (http://musicbrainz.org/) or commercial websites such as Audioscrobbler (www.audioscrobbler.net) with open APIs (Application Programming Interfaces). With these metadata reservoirs, you can organize your library based on the opinions of hundreds of thousands – creating what's come to be known as a "folksonomy." Access to this kind of information, and harnessing the algorithms put in place by these projects, Beat Browser will be able to organize your library along many more advanced dimensions.

Playlist Manipulation

Playlist functionality is in place, but there is a limited feature set. It is important to add drag and drop functionality to the playlist window to make reordering songs in a playlist intuitive. It may also be interesting to have the album art in the browsing window indicate that a song has been added to a playlist. Perhaps an album fades slightly when one of its songs are taken, or totally if all of its songs are. Finally, there should be more generous playlist view options. Users need to be able to minimize the browsing window in favor of a more comfortable playlist viewing and editing environment.

Touch Screen

The immediacy of interaction may make Beat Browser a successful application in a touch screen environment If initial experimentation is encouraging a significant I/O remapping will be necessary. Touch screens only support a limited set of inputs. There isn't the pair of buttons and a scroll wheel, there's just your finger (assuming multi-touch is still a little ways in the distance). This may mean automating some features, such as opening the Halo with a sustained touch, and adding new interface elements like sliders to permit zoom control.

Long Term

Web Ready

Beat Browser seems naturally suited for surfing the massive music collections of the on-line music services such as Rhapsody. The barrier, however, is significant: redesign Beat Browser to operate with similar fluidity while pulling content from a remote server instead of the local hard drive. Many issues arise, particularly with respect to bandwidth limitations and buffering latencies.

Initial exploration points to using the Adobe Flash player to render the audio. Flash players are currently installed on over %95 of the world's computers. This means developing Beat Browser in the Flash development environment, or in some lower level language that ultimately reports to Flash.

Certainly dropping the sound quality as low as possible will be crucial to improve buffering. To approximate the UTB effect, Bill Gardner suggested quantizing the effect, downloading parts of each song in steps – every 15 seconds updating the buffer for "universal" playback beginning 15 seconds further in the songs. Multiple strategies will clearly have to be developed to convincingly reproduce Beat Browser over the web. That said, a web-based Beat Browser would take the concept to its logical conclusion.

8. Conclusion

Many decisions have been made throughout the process leading to the current Beat Browser interface. They all, however, followed both the design logic of the Universal Time Base playback architecture, and code logic in the 862 lines that comprise the core SongPlayer.java class. The other 5,000 make Beat Browser look and respond as it does.

Response overall has been positive. Beat Browser is both familiar, yet slightly unfamiliar. It is not any particular innovation, or invention, rather Beat Browser represents a strategy for more animated and intuitive media representation and interaction.

We have to recognize the importance of browsing, particularly as the amounts of information we confront increase. "You don't know what you don't know" goes the old saying. We need to create better tools to help people discover the things they don't know and rediscover things they've forgotten.

9. Bibliography

Aucouturier J. J., Pachet F. “Scaling Up Music Playlist Generation” Proceedings of IEEE International Conference on Multimedia and Expo, Lausanne 2002.

Brazil, E., Fernstrom M. “Audio Information Browsing with the Sonic Browser” In Coordinated and Multiple Views In Exploratory Visualization (CMV03), London, UK 2003.

Brewster, S., Lumsden, J., Bell, M., Hall, M. and Tasker, S. “Multimodal ‘Eyes-Free’ Interaction Techniques for Wearable Devices” In ACM Transactions on Computer-Human Interaction, 2003, p. 473-480.

Bull M. “Investigating the Culture of Mobile Listening: From Walkman to iPod” (pp. 131-151)
O’Hara K., Brown B. “Consuming Music Together: Social and Collaborative Aspects of Music Consumption Technologies.’ Computer Supported Cooperative Work, Volume 35.

Carey B. “Magical Thinking: Why Do People Cling to Odd Rituals?,” The New York Times, Mental Health & Behavior Section, January 23.

Cohen, M. and Ludwig , L., Multidimensional Audio Window Management. International Journal of Man -Machine Studies, 1991. 34: p. 319-336.

Csikszentmihalyi M. “Flow: the psychology of optimal experience”

HarperCollins, New York, New York, 1990.

Cunningham S. J., Reeves N., Britland M. “An Ethnographic Study of Music Information Seeking: Implications for the Design of a Music Digital Library” In Proceedings of the Joint Conference on Digital Libraries, pages 5–16, Houston, Texas, USA, May 2003.

Earl, P. and Potts, J. Latent Demand and the Browsing Shopper, Managerial and Decision Economics 21: 111 – 122 (2000) Copyright © 2000 John Wiley & Sons, Ltd.

Falk P., and Campbell C. Introduction, in The Shopping Experience (P. Falk and C. Campbell, eds.), Sage Publications, New York, 1997.

Ghias, A., Logan, J., Chamberlin, D. and Smith B.C. Query by humming: musical information retrieval in an audio database. In Proceedings of ACM Multimedia Conference’95 (San Francisco, California, November 1995).

Gibbons, D. “Ceiling Mounted Speakers” Home Theater Tune Up, April 2007, www.dgibbons.com.

Goto M. “SmartMusicKiosk: Music Listening Station with Chorus-Search Function” In Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology (UIST 2003), p. 31 – 40.

Goto M., Goto T. “Musicream: New Music Playback Interface For Streaming, Sticking, Sorting and Recalling Musical Pieces” In Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR ‘05), London, UK 2005.

Jones M., Jones S., “The Music Is The Message,” interactions, Volume 13 Issue 4, 2006.

Kim, Ja-Young and Belkin, N. J., Categories of music description and search terms and phrases used by non-music experts, in Michael Fingerhut (Editor) Proceedings of the Third International Conference on Music Information Retrieval: ISMIR (Paris, France, October 2002), pp. 209-214.

Knees P., Schedl M., Pohle T., and Widmer G. “An Innovative Three-Dimensional User Interface for Exploring Music Collections Enriched with Meta-Information from the Web” Proceedings of the 14th ACM International Conference on Multimedia (MM’06), Santa Barbara, California, USA, October 23-27, 2006.

Leong T., Vetere F., Howard S., “The Serendipity Shuffle” Proceedings of OZCHI 2005, Australia.

Leong T., Vetere F., Howard S., “Randomness as a Resource for Design” In Proceedings of DIS 2006, University Park, Pennsylvania.

Levitin, D. J., “This Is Your Brain on Music: The Science of a Human

Obsession," Dutton, Penguin USA, New York, New York, 2006.

Levitin, D.J., Neuroscientist, profiled in The New York Times, "Music of the Hemispheres," Clive Thompson, December 31, 2006.

Logan B., Salomon A. "A Music Similarity function based on signal analysis," in Proceedings of IEEE International Conference on Multimedia and Expo (ICME), 2001.

Mackinlay J. D., Zellweger P.T., Chignell M., Furnas G., Salton G., "Browing vs. Search: Can We Find a Synergy?" CHI Mosaic of Creativity, Colorado, 1995.

Magical Thinking, http://en.wikipedia.org/wiki/Magical_thinking

Marchionini G. "Exploratory Search: From Finding to Understanding, Communications of the ACM, April 2006/Vol 49, No. 4, pg. 41 – 46.

McLuhan M., Fiore Q. "The Medium is the Massage: An Inventory of Effects" Bantom Books, New York, New York 1967.

Moore R., Shaw J., Chipp K., "Eight years on: An extended model of online consumer behaviour" South African Journal of Business Management, 36(2) 95 – 103, 2005.

Morville, P. "Ambient Findability: What We Find Changes Who We Become" O'Reilly Media, Sebastopol, California, 2005.

Pampalk E. "Islands of Music: Analysis, Organization and Visualization of Music Archives." Master's Thesis, Vienna University of Technology, 2001.

Pampalk, E., Goto, M., *"MusicRainbow: A New User Interface to Discover Artists Using Audio-based Similarity and Web-based Labeling"*, in the Proceedings of the ISMIR International Conference on Music Information Retrieval, 2006.

Platt J. C., Burges C. J. C., Swenson S., Weare C., and Zheng A. "Learning a gaussian process prior for automatically generating music playlists." Advances in Neural Information Processing, volume 14, p. 1425-1432, 2002.

Reid J., Hull R., Cater K., Flueriot C. "Magic Moments in Situated Mediascapes" Proceedings of the 2005 ACM SIGCHI International Conference, Portland 2005.

Sawhney, N. and Schmandt, C., "Nomadic Radio: Speech and Audio Interaction for Contextual Messaging in Nomadic Environments" ACM Transactions on Computer-Human Interaction, 2000. 7(3): p. 353-383.

Tzanetakis G., Cook P. "Marsyas3D: A Prototype Audio Browser-Editor Using a Large Scale Immersive Visual Audio Display" In Proceedings of the International Conference on Auditory Display, 2001.

Instructional Map

Operable View commands:
Fit to Window
Select All

albums covers are acquired from the web, and resized to 75 x 75 pixels. This number can be increased with further graphics optimization

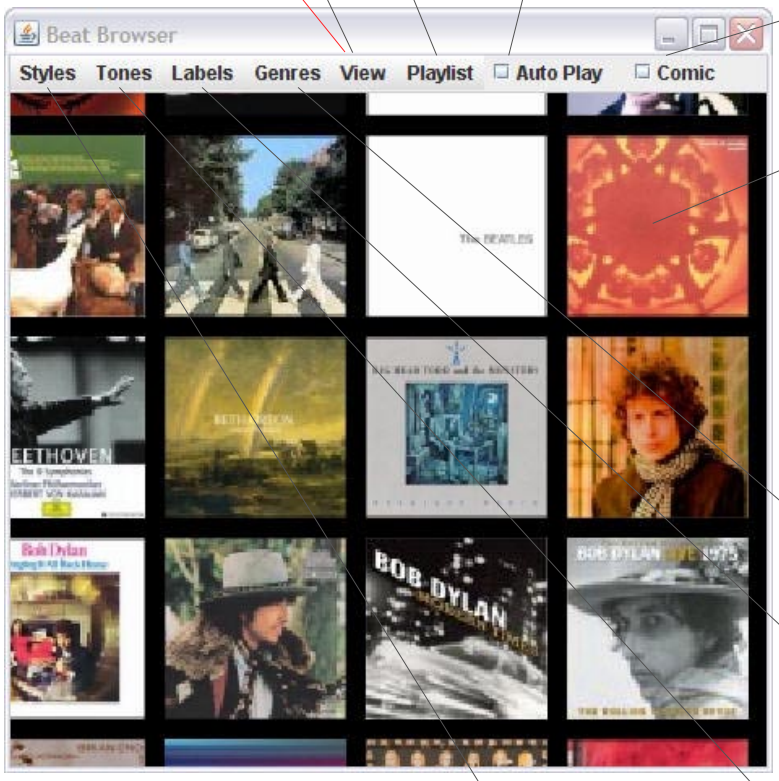
Auto Play is akin to the Scan feature on radios

Auto Play steps through the selections in your current view (all selected, genre, tone, etc.) with a 5 second interval

Unchecking Auto Play opens the Halo view

Important
To step out of an attribute view, e.g. Genre, Tone, etc.:
View > Select All
Repeat if necessary

opens comic view



albums covers are acquired from the web, and resized to 75 x 75 pixels. This number can be increased with further graphics optimization

These fields are dynamically populated by reading the album.xml data, created by AllMusic.com

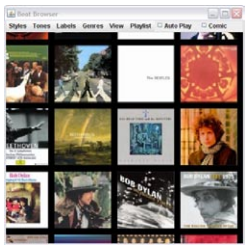
Genres, e.g.: Electronic, Folk,Jazz, Rock, Classical, World, R&B...

Labels, e.g.: Warner Bros., Epic, Sub Pop, Reprise, Polygram, Impulse, Verve...

Tones, e.g.: snide, innocent, sparse,bright, wistful, literate, ramshackle...

Styles, e.g.: anti-folk, contemporary folk, progressive country, british folk....

Standard View



Click album cover of choice to open up Halo View for track selection

zoom in and out

add album to playlist view

click and drag for pan

Halo View

scroll through Halo

click and drag for pan

add song to playlist

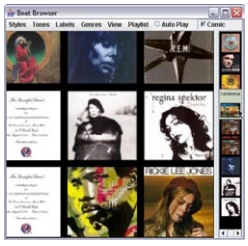
lock song / unlock song:

select wedge (visually verified) to automate album playback, disabling mouse motion events

restart song with keyboard

click anywhere to unlock, click again to exit Halo View

Playlist View



remove from playlist

Keyboard Function

1. If in Halo View, and Locked, press the left arrow key (keyboard) to begin song again, from the beginning.