SING, PAN, CODE:

ENHANCING AUDIENCE ENGAGEMENT THROUGH CHORAL, ALGORITHMIC, AND PHOTOGRAPHIC CROSSINGS

By

HAROLD SIKKEMA

Supervisor: Dr. David Ogborn

A Major Research Paper/Project

Submitted to the Department of Communication Studies and Multimedia

in Partial Fulfillment of the Requirements

for the Degree

Master of Arts

in Communication and New Media

McMaster University

Table of Contents

In my photography practice, I am preoccupied with the gestural qualities of Ontario's geology and hydrology. This engagement with pattern and process in the landscape spurs an inner artistic life, but also opens up potential for more collective concerns, and the involvement of audiences across a spectrum from passive observation to active artistic collaboration. I have found fruitful models of collective engagement in the community-oriented practices of the choral ensemble, and in the network-oriented context of mobile device art. In this project, I explore crossings of panoramic photography, choral music, and mobile device art, through the creation of four interactive artworks: *PeaceWeave*, *Sedimentary*, *ReptileChoir*, and *StreetSong*. Given an artistic vocabulary of flowing geo-panoramic photography, along with the transparent technical affordances of a networked mobile device platform, audience-performer engagement in choral contexts may be enhanced.

Modes (and media) of existence

When diverse media, materials and practices converge in art, the resulting work holds a productive tension. The present artistic inquiry involves crossings between multiple realms: photographic practice, choral music, and mobile device art. These are eclectic phenomena. Each exists on an independent trajectory, consists of unique languages, parameters, and assumptions, and affords unique resources and possibilities. To understand artistic amalgams of media, we may benefit from Bruno Latour's pluralism of modes. Recognizing the unique felicity and infelicity conditions of different modes of existence, Latour (2013) aims to clarify category mistakes that often result during conflicts of values. Lawyers and scientists, for example, have attachments to very different types of veridiction, but if their values are respected we need not reduce them to a single model. Beginning with the network, Latour zones in on continuities and hiatuses specific to each mode of existence. In this way "the social" (an elusive term) is reformulated around various human patterns and practices: law, politics, religion, etc. These modes, or ways of being, are productively analogous to ways of working. In the case of the four artworks I created for this project, I consider the 'modes' of three media: photography, choral music, and mobile device art. While these media do not necessarily constitute full blown modes of existence, they can be situated in terms of the existing modes, and also have "continuities and hiatuses" of their own. Contemporary choral music, for example, has linkages with the world of filmmaking, but remains a practice in its own right: the choir may rehearse independently of any film script. Photography, while always part of a wider political sphere, also has its own language generating its own kind of conversation. Thinking about these media in terms of modes of existence can help to situate artistic components within a wider frame.

Category mistakes happen when "the veracity of one mode is judged in terms of the conditions of veridiction of a different mode." (Latour, 2013, p.17) Whilst crossing modalities native to choral music with those of photography, we also risk running amok in this way. If the core concern of the choir — singing together — differs in value from the panoramic impetus — seeing further — we may yet respect each, and offer to each a new account of what their respective informants (singers and photographers) hold dear. Furthermore, if in choral music the base unit is a "beat," and if in digital imaging, the base unit is a "pixel," this does not automatically legitimate a mapping of one onto the other. The temptations of bifurcation and equivocation are for Latour a kind of cardinal sin: the work of an evil genius, Double Click. With these threads in mind, I will now the three realms of creation as they intersect in an iterative series of browser-based artworks: *PeaceWeave*, *Sedimentary*, *ReptileChoir*, and *StreetSong*.

Panoramic Photography

In a time when photography is ubiquitous (Hand, 2012) and the lifespan of an image is short, do panoramas persist? With billions of smartphones saturating the planet, photography is no longer a privilege, nor merely a means to remember, but a language for everyday communication. We have embraced the paradigm that images are disposable, designed to expire. Fifteen seconds has become a long time, making Warhol's fifteen minutes seem a panoramic eternity. The grand narratives of Ansel Adams' iconic photographs of mountains have given way to Erik Kessels' mountains of photographs (as in *24 Hrs in Photos* at FOAM, Amsterdam). Even David Hockney's 1982 photo collage works of the Grand Canyon seem quaint in light of the massive photocollage databank we know as Google Street View. And Andreas Gursky's *Rhine II* fetched a vast sum at auction not as any traditional panorama (though it has such a surface) but by virtue of astute manipulation: a landscape constructed from fragments. The panorama, if it exists at all, has been edited, curated, and stitched into our consciousness. Perhaps it is most present in this moment via the endless pan of the social media news feed, the monetized logical conclusion of an 'endless runner' videogame. Protagonists, from Nintendo's Mario onwards, have less and less agency.

While the ubiquity of imaging devices seems at first to put the means of production in the hands of the masses, the information networks and social media through which their value is determined are controlled by corporations. On the gallery side, the Hasselblad camera (used by activist Edward Burtinsky) remains beyond the budgets of most, but even here, the artist's true differentiation and success is attained politically: through negotiated access to industrial sites, and through networks of notoriety, and the artificial platforms of privilege. The photographic medium also remains political with respect to its use in and for representation. As a tool in the construction of meaning,

the photograph is discursively coded and decoded. Stuart Hall's point that "there is no intelligible discourse without the operation of a code" (2009, p. 167) applies as much to photography as to any medium.

I enjoy panoramas for the satisfaction of an ultra high resolution result, and for the enjoyment of the stitching process — for me a kind of ritual of looking — but I'm also interested in the narrative qualities of panoramas: not as over arching meta-narratives (though jet contrails do arch-over the dome of my particular heaven), but as traces of distinctly local trajectories, patterns, and processes. Photoshop's PhotoMerge tool makes it possible to stitch together ultra high resolution images without an ultra high end camera. The stitching process is performative: an integral constructivist assembly act. In practice, it's often more feasible to experience the wide sweep of a 60,000 pixel image via screen, rather than printed on monumental scale. There are economical and environmental reasons for this, but there is also the reality that screens are as ubiquitous as cameras. Smartphones and projection screens alike lie open as portals into panoramic metamedia.

In any photograph with typical proportions (such as 4:3 or 16:9) the eye moves through the space, according to its gestalt. In our optical scanning of the image, we seek a path from focal point to focal point, and thus curate for ourselves the experience of a narrative flow. In panoramic images, this flow spans a wider aspect, which involves a change in perception. As it moves proportionally from 1:2 to 1:10 to 1:n, the picture plane begins to resemble ever more closely a line, rather than a plane. To the mind's eye, it sets up a dichotomy of here and there, the temporality of a journey. Of course, the photograph remains distinct from the moving images of film, remaining "a neat slice of time, not a flow" (Sontag, 1977, p. 13). And yet, whether the eye moves along a wall-sized print, or whether the image pans across a palm-sized screen, the panoramic photo in particular does imply a kind of storied movement. I think of this as a scrape, an encounter, or better: a traversal. Panning images thus resonate for me with the more elemental kinds of movement and rhythm that I look for when shooting: from footpaths to riverbeds, I am preoccupied with linearities wherever they happen to emerge.

Like the river, the panorama is fraught with risk. It may be the lulling risk exemplified in the opening scene in Jennifer Baichwal's film Manufactured Landscapes (2006). Here, an 8-minute long transit across a Chinese factory deadpans the near equation of human and machine. Far from Sontag's 'neat slice of time', it punches us in the gut with a dehumanizing linearity. Panoramas can also risk being mistaken for the whole. We may be so seduced and taken by the IMAX theatre that we mistake it for the Grand Canyon itself. We may mistake the map for the mountain range. In its relentless Cartesian march, the one-dimensional panorama smacks of Latour's Double Click.

Spectacles with a softer sweep may be recalled by means of a media archaeology (Zielinski, 2006). Telegraph ticker tape is now obsolete, and the punched cards that once animated barrel organs now evoke nostalgia, but both survive in my panoramic consciousness. And while Robert Havell Sr.'s *Naturorama*(1825), aims to apprehend the River Thames through endless transposition — via a series of eighteen interchangeable aquatint cards — it nevertheless recollects a gentler pace of life (Bland et al., 2013, p. 173). If the present art partakes in too brash an enframing, too instrumental a setting-in-order of the world (Heidegger, 1954) and if, thereby, too much causality reigns, I find consolation in these fits and starts along an older (and slower) riverbank.

Choral Music

The choral score presents species of 'continuity and hiatus.' Notes and rests indicate the persistence and absence of vocal tone, the ins and outs of lungs, and the associated affect of breaths taken and given. Choral repertoire also typically includes that sort of continuity in which notes and rests are strung together into what we call 'songs,' punctuated also by the antiphonal hiatus known as 'applause.' And in the widest choral gyre, I find winding the continuity of rehearsal and the summer hiatus that inevitably follows a spring performance. When engaged on its own terms, choral music also makes the further demand that its unique felicity conditions be met. The singer must assign herself to a section: Soprano, Alto, Tenor, or Bass. To belong at all, she must acquiesce to the choir's structure and (sometimes essentialist) hierarchy. As Garnett points out, "assumptions about the 'natural' vocal range for one's sex can … work coercively" (2009 p. 67).

Choirs sing diverse repertoires for diverse reasons. While Toronto's Choir!Choir!Choir! sings the pop-politics of Beyoncé with enthusiasm, Howard Shore's choral rendition of "A Journey in the Dark" paints a more guttural chant, carrying the kind of gravitas we might associate with neo gothic stonemasonry. The choir retains at least the aftertaste of liturgy. The score is followed like a script, if not like scripture. The audience observes, the singers inhabit roles, and props (such as conducting batons, microphones) reveal identities and power relations. As a form of group singing, the choir forms a community. Whatever they come together to sing about, they form a community of practice: a group of people, who "share a concern or a passion for something they do and learn how to do it better as they interact regularly" (Wenger-Trayner, 2015).

What is learned by the choir is typically obedience to the score: the specific kind of text that choirs come together to read, interpret, and perform. Scores, like lanes in a street, tell us where it is safe to walk. They give little room for leeway, for subversive action, or for tactical resistance. In ritual contexts, this is typically by design. As Bell

notes, “Singing in unison imperceptibly schools the social body in the pleasures of and schemes for acting in accordance with assumptions that remain far from conscious or articulate” (1992, p. 215). Whatever its aesthetic outcome, the score remains a tool to exact compliance, and its linear staffs express panoramic inevitability: a virtual bobsled run.

But there are also other forms of choral music that do not rely on any centralized paradigm. Yun’s work with Jabber and choral improvisation, for example, evolves the definition of a choir beyond the model of a central conductor. (Yun, 2014) On the level of composition, John Stump’s visual scores evoke more than mere instruction. His densely printed *Faerie’s Aire and Death Waltz*[1] employs notes much as if they were brushstrokes on a canvas: a deliberate rather than accidental category mistake, and thus immune to Double Click. Varieties of self awareness thus remain latent in both the creation and the traversal of choral scores, much in the way that a concrete sidewalk presents to any curious pedestrian the impetus to playfully refuse stepping on the gaps.

While the choral score might involve panoptic aspects, it could as easily be understood as a boundary object (Starr & Griesemer, 1989) where notation constitutes a more democratic conversation. Boundary objects “have different meanings in different social worlds but their structure is common enough to more than one world to make them recognizable, a means of translation. The creation and management of boundary objects is a key process in developing and maintaining coherence across intersecting social worlds” (Starr & Griesemer, 1989, p. 393). The coherence we find in choral practice is negotiated around the score. We see how “a conductor, a music historian, and an involved listener may all use one score in completely different ways for different purposes, its multiplicity of function providing enough information for each user to happily and productively interact with the object” (Winget, 2008, p. 1879). In this view, the choir is more actively engaged in the construction of meaning, and less a passive recipient of ideological codes.

Garnett, while highlighting the way in which choral practice involves many people in a “fertile ground for the production of socio-musical meanings”, also examines choral practice through the discourses that surround it, applying a Foucauldian lens to its texts (including choral conducting manuals). She develops the critical thesis that choral rehearsal is a form of enculturation that “teaches choir members the accepted norms and values of the practice of choral singing in order that individuals may fulfill their roles competently within the group” (2009, p. 59). She

[1] https://lostinthecloud.files.wordpress.com/2010/03/faeries-aire.gif

further locates choral singing in terms of Anthony Giddens' notion of a 'lifestyle sector,' pointing to the choir as one of "several more or less distinct worlds, each with its own languages and repertoires of behaviours" (p. 60).

Pairing the notion of an ‘art world’ (Becker, 1982) with ‘musicking’ (Small, 2011), Crossley pictures a “music world” as engaged through, networks, conventions, resources, and places. Musical networks involve interactions and ties of various lengths and strengths. Conventions, like musical scales, help musicians coordinate. Particular distributions of resources and skills imply power relationships and kinds of interdependence. Places, too, profoundly inflect the making of music (Crossley, 2015, p. 474): a musical space emerges from “a sonic event, plus the architecture that shapes it.” (Blesser & Salter, p. 15) Applying this to choral milieus more specifically, I think of the bonds of commitment forged by practicing musicians, their agreement to observe the decorum of rehearsal, their varying and overlapping capacities, and the art venues, churches, and pubs in which they perform. All four of these considerations will diffract differently with the introduction of mobile devices into the mix. Smartphones change the way we fundamentally relate, demand new sensitivity to conventions (turn off your phones in rehearsal), afford us with new resources (Web Audio API, realtime score generation), and upend our sense of place as strictly physical by connecting us across distance.

Mobile Device Art

Smartphones are not a medium as such, but a portal into metamedia. Thus while McLuhan’s (1964) adage about the medium being the message might still hold to a degree, I am drawn to Lev Manovich’s (2014) understanding about how the software is the message, particularly when the software is orchestrating the experience of a participatory artwork.

Prior to pervasive broadband networking, mobile phones already offered an important platform for making interactive music and art, especially via their sensor capabilities (Essl & Rohs, 2009). Now that smartphones, connectivity, and accessible browser-based tools are all ubiquitous, mobile device art is flourishing. The Stanford Mobile Phone Orchestra, for example, has explored how audience participation via Apple iPhones and iPads, may open up a social musical experience that is both interactive and engaging. (Oh & Wang, 2011). Traditional musical contexts are reimagined, as in *Concert for Smartphones* which repositions the audience in a way analogous to a “classical concert for soloist and orchestra [using the] audience’s mobile devices as a medium for sound diffusion” (Bundin, 2016). The accessibility of tools is also a key concern. The Web Audio API (an emerging W3C standard) has been important in this respect, validating the web browser as a viable artistic platform. While the API is a “a sound

developer's interface rather than a general application developer's interface" (Wyse & Subramanian, 2013, p. 17) it has been made more artist-accessible through tools such as *WAAX* (Choi & Berger, 2013) and *Gibber*, a browser based coding environment for creating and distributing instruments. (Roberts et al., 2015, p.27). While the browser may be lacking in areas of timing and extensibility, especially when compared to other systems like Max/MSP and Puredata, its advantages are clear: it is naturally networked, community-supported, and (especially in terms of smartphones) accessible and portable. (Wyse & Subramanian, 2013, p.21).

It is a broad characteristic of New Media work that the user (or audience) may 'co-author' a work by means of interaction (Manovich, 2001). Freeman, for example, navigates these concerns in his work *Glimmer* by finding a comfortable balance of separation between audience and orchestra. (2005, p.757) In general, since I'm addressing the question of audience engagement, I want to keep this slippage of categories in mind. If the audience is redefined, then so too is audience engagement.

In each of the four artworks I created for this project, concerns have arisen about accessibility and ease of access. In choral and electroacoustic milieus alike, specialized resources and knowledges are required, and access limits (such as auditions) may be set in place. In interactive art, however, the focus is often on democratizing participation. The electronic music platform Pyxis Minor, for example, aims at a low barrier to entry, and a playful emphasis (Barraclough et al., 2015, p. 1) My work with *Reptile Choir* in particular similarly involves an interface that is play-oriented. And *StreetSong* likewise makes an effort to lower the bar of participation. By affording singers the combination of a musical score as well as a pitch cue, the access to non-musicians is opened up, and musical learning potential heightened.

PeaceWeave

To explore productive entanglements of panoramas, choirs, and smartphones, I created a series of four artworks. The first of these, *PeaceWeave*, omits smartphones, setting the stage with a simpler choral-photographic pairing. Scrolling visuals were synchronized with a choral repertoire performed by the McMaster University Choir, conducted by Dr. Rachel Rensink-Hoff. For each of ten works, a panning video displayed a scrolling sonogram interwoven with thematically appropriate panoramic photography. To keep singers in the centre, *PeaceWeave* employs two flanking monitors, a cue taken from a performance of Eric Whitacre's *Sleep* at Aberystwyth Arts Centre (Côrdydd, 2009). Real time adjustment of the playback rate temporally aligned the visuals with the choir, resulting in an emergent musical chronology: a river-like flow. Handwritten and photographic vignettes, contributed by participants during the

Perspectives on Peace initiative, were rendered into semi-opaque bubbles, and animated as if drifting downstream. To connect the bubble mementos with the momentum of choral transport, the conductor's gestural movements attenuated their throbbing via wireless sensors.

Figure 1: An overview of the performance configuration during PeaceWeave (Photo taken by K.J. Bedford, 2016)

Figure 2: An audience perspective during the PeaceWeave performance (Photo taken by K.J. Bedford, 2016)

***Figure 3**:* The basic flow of a panoramic sonogram across a viewport.

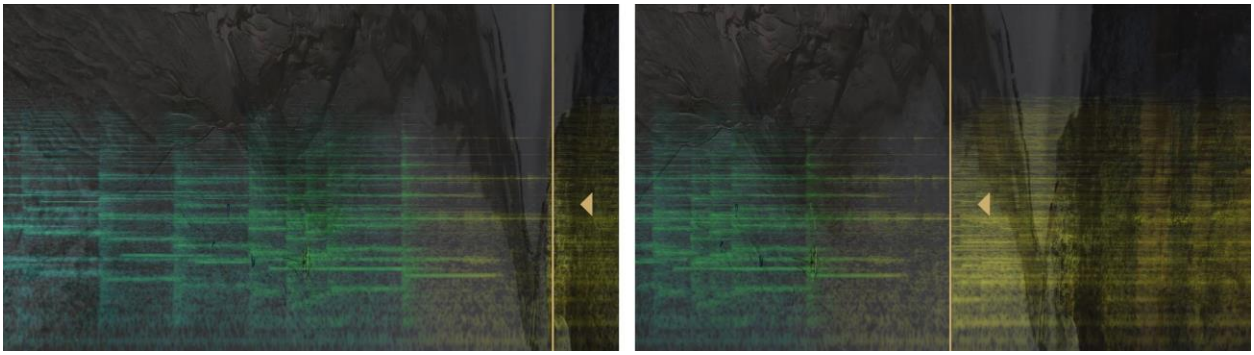

***Figure 4**: Screen captured visuals correspond to a swelling wall of sound in Timothy Corlis' Kyrie*

Panoramas — especially panning panoramas — lend themselves to representing a flowing musical temporality, their wide aspect ratio evoking a timeline. Moving across a viewport, the perceived "present" emerges from an anticipated future, and recedes into an imagined "past". This perceptual frame is culturally constructed, and like left- and right-handedness, negotiated by convention (McManus, 2004, p. 236). While on a social media feed, the "future" is often spatially *below* the screen, *PeaceWeave* traverses rightward, mirroring the traditional directionality of the score. Other approaches to traversal are explored in later works.

If *PeaceWeave*'s panning flow evoked a linear (perhaps even a Cartesian, or totalizing) inevitability, it also involved the complexity of photographic layering. The sonogram, while both minimal and predictably linear, offered a non-abrasive point of departure, overagainst more flashy forms of animation, that might distract from the choir. This base layer is the ground of a scrape between sonogram and photograph: the site where sonic representation meets the affect of image. The sonogram's painterly texture mingles here with the photo as in the works of Gerhard Richter, where photo-overlaid brushwork also creates a veil of abstraction. While the sonogram kept us in time, the photograph was free to do other things: to conjure an arena, to hold the tension of the music, to evoke a conceptual space, to present a tactile trajectory; to offer a geological sensibility as a pathway for a well-tuned traversal. Differing scroll rates between these blended layers heightened the sense of visual depth.

PeaceWeave's layered landscapes were diversely conceived. For Corlis' *Gloria*, a singular unbroken jet-vapour trail, crosses an expanse of hot pink sky, transposing the "In excelsis deo" of tradition into a postmodern key. For Ešenvalds' *Stars*, cosmic dust particles shimmer over the expanse of a frozen Lake Erie: an echo chamber for a song accompanied by wine glasses. In creating choral photoscapes, prior visual treatments of the score resonate in my mind: the graphic notation of Karlheinz Stockhausen in the 1970s, as well as John Cage's beautifully rendered scores.

Gritten and King (2006) have described an aesthetic quality in both conducting gesture as well as the agreement it achieves: when harmonious, that agreement engenders satisfaction among participants. (p.148) While this may hold true even when the conducted 'participants' are animated bubbles, the harmony is complicated in *PeaceWeave* by the bubbles' random appearance. Unscripted, these currents and eddies often juxtaposed (in sometimes conceptually provocative ways) with choral text and texture. While a John Lennon cameo on the opening of *Richte mich Gott* offered a new lens on Mendelssohn, a handwritten note urging "unity between the indigenous Assyrian people of Iraq/Syria Turkey/Iran and the rest of the Middle East" gave an inflection to *After the War* that its authors might not have anticipated. The bubble algorithm was thus jarring at times, but in a way that resonated with the inherent complexities of peace-building.

In *PeaceWeave*, temporal alignment of image and sound generally yielded a functioning 'audiovisual contract', enabling audio-viewers' to think "of sound and image as forming a single entity" (Chion, 1994, p.216). While the sonogram's slow crawl did not on its own create an immersive immediacy, the integration of additional layers — such as a stark pair of hand-drawn eyes in Lauren Bernofsky's *The Tiger* — enabled a further synchresis of key choral moments with panoramic texture. A future system might more thoroughly fuse conducting gesture into the visual flow for even greater synchronicity.

Sedimentary

Sedimentary was a live-coding act which I performed at McMaster's LIVELab alongside the Cybernetic Orchestra and friends at the April 7th edition of the LIVELab's *Series 10dB*. It combined photography and audience participation (via smartphone), with a poetic approach to live-coding. The base panorama here was a 30000-pixel wide sweep of the Niagara Escarpment, at Arkeldun Avenue in Hamilton (known as the Jolley Cut). Road widening in 1953 exposed a cross-section of layers, including the Grimsby and Thorold formations.[2] Shales, sandstones, and limestones, strikingly segmented into primary colours, here present an aesthetically and geologically significant

[2] See also https://www.science.mcmaster.ca/geo/outreach/road_trips/Jolley_Cut.pdf

dataset: the productive remnants of an ancient inland sea.

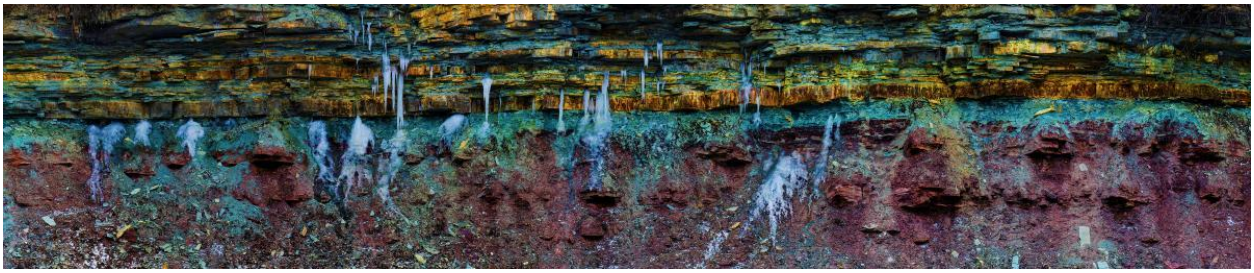

Figure 5*: A saturated excerpt from a wider panorama of the Niagara Escarpment in Hamilton at Arkeldun Ave*

Sedimentary fits into what Collins et al (2003) describe as live-coding: a musical performance that also involves real-time scripting. (p. 321) It has been created in view of principles of openness and quality fostered by live-coding communities like TOPLAP, wherein I have found helpful impetus to reveal 'the performer's mind' and 'the whole human instrument', and to 'transcend the program' by means of language. (Ward et al., 2004, p. 247) As we will see, Sedimentary employs a poetic subtext alongside a technical: pun is valued together with precision.

The web browser is the platform for both the performer (on a laptop) and the audience (on mobile devices). The laptop projects a control and display interface onto a large overhead screen. We see a scrolling escarpment, along with a series of 'lenses' that appear to hover over it. These lenses correspond to the image colours: red, blue, and yellow. Audience members visit a webpage, where they find a set of similarly coloured channel buttons, used to 'tune in' to a given lens. Devices are made to emit sound samples when their corresponding overhead lens encounters familiar pixels (achieved technically via HTML5, Canvas, CSS3, and jQuery). If the blue lens finds a blue rock, then the blue channel will sound. The audience thus self-organized into sections, each producing distinct output. Yellow mapped onto a bird-like whistle, blue onto a guttural stone scrape, and red onto a resonant rocky clink. In addition, the relative brightness or darkness of the encountered image data affected the sample playback rate, thus effecting a more expressive pitch space. Sounds played on audience mobile devices in the browser via sample-based synthesizers created using the Web Audio API.

To activate further musical emphasis and direction in *Sedimentary*, I triggered poetically named JavaScript functions that altered the behaviour of hovering lenses. (See Table 1). Functions were entered into a text area for that purpose, integrated over the display.

Function	**Effect**

goldenAge() theBlues() redShift()	Change to a gold/blue/red filtered version of the escarpment, thus highlighting colouristic segmentation in the image and activating the corresponding synth.
rockOn() rockOff()	Commence panning movement with optional speed argument Conclude panning movement
callMyBluff()	Trigger the “Primordeal” synth to “sigh” on all devices.
freeze() thaw()	Halt motion of traversal lenses (inspired by icicles) Resume motion of traversal lenses
breakOut()	Initiate interesting motion and rhythm in the traversal lenses.
getBackInLine()	Set traversal lenses into positions that align their colour with the image.

Table 1*: a selection of poetic function names from the Sedimentary performance.*

Sedimentary (along with the third and fourth works, *ReptileChoir* and *StreetSong)* is built on the technical affordances of apert, a server platform that enables artists to distribute JavaScript code and resources to participants’ mobile web browsers, and activate them through a centralized interface (Ogborn, 2016). Apert assumes the context of a free and open source development model, which values transparency. This is beneficial in collaborative projects for many reasons, include increased learning (from the actions of other developers) and heightened project visibility (Dabbish et al., 2012). It also saves time. With apert taking care of the complexities of network communication in the mobile browser, I was able to focus on the artistic aspects of my work. By linking my finished artworks back to the open source ecosystem, a mutual benefit is attained.

Musically, the work exhibited a choral texture: a tug between feathers and tectonics. Whistles contrasted with more deeply resonant rock scrapes and clinks. Efficient realtime distribution of signals effected a general synchronization of sounds, attenuated by small differences in latency and the speaker properties specific to each participating device. The sonic layering exhibited local imperfections, which nevertheless contributed to a blended (if not choral) texture. While none of these sounds involved collective of lungs or trachea, the unified outcome resonated for me with Daugherty’s description of choral sound as having "a nuanced life of its own apart from the discrete individual sound sources that contribute to it" (2001, p. 70).

Figure 6: *Testing audience interface using Nexus 7 tablets at the LIVELab*

Figure 7: *Timelapse showing audience smartphone interaction with Sedimentary at the LIVELab*

Reptile Choir

On May 13th 2016, I presented *ReptileChoir*, a youth-oriented interactive art exhibition, during Hamilton's monthly Art Crawl at the Factory Media Centre. Participants were prompted to log in via their smartphones, to a control interface, linked with a personalized avatar turtle appearing on a collective projector. The turtle would 'sing' pitches in different vowels, and navigate a panning photograph, responding to encountered highlights and shadows with octave jumps. Turtles could "vote" to control the direction of image flow, by shifting their position on the screen.

***Figure 8**: Browser-based UI for ReptileChoir as seen on a participant's smartphone (Photo: Harold Sikkema)*

***Figure 9**: Promotional materials for ReptileChoir: May 13th, 2016 at the Factory Media Centre*

Like *Sedimentary*, *ReptileChoir* relied on the server infrastructure of Apert. The key difference was that there was no discernable performer, nor any fixed length of time for a performance. Interaction, like the looping image chosen for the work, was ongoing. During the installation I alternated between monitoring the system for technical stability, and interacting with the participants.

I employed a relatively simple set of musical parameters in *ReptileChoir*. Having recently encountered univocalic constraints in Canadian Poet Christian Bök's bestselling *Eunoia*, I recorded each of the English vowels (A, E, I, O, U) in their long form across the twelve semitones of an octave, to build a base of 60 samples. Participants could select a 'favourite' vowel and hear it sung at a variable pitch: controlled by the position of their turtle along the vertical axis.

While visual rhythm is a property of images generally, I've found the most unique patterns in the index, gesture, and mark-making of organic forces. To the degree that rippling streams, cascading trees, and scaly fish, afford playful, integrated, and satisfying continuities of texture, they may also afford a unique musical potential. Complexity

in the ebb and flow of light, when transcoded through a keen interpretive lens, may yield just the sort of digital signal that can make for a potent acoustic gestalt. *ReptileChoir* features the Grindstone Marsh in Ontario's Hendrie Valley. I have invited its watery shadows and highlights, surface reflections, and plant matter to shape the sound and texture of a reptilian avatar experience.

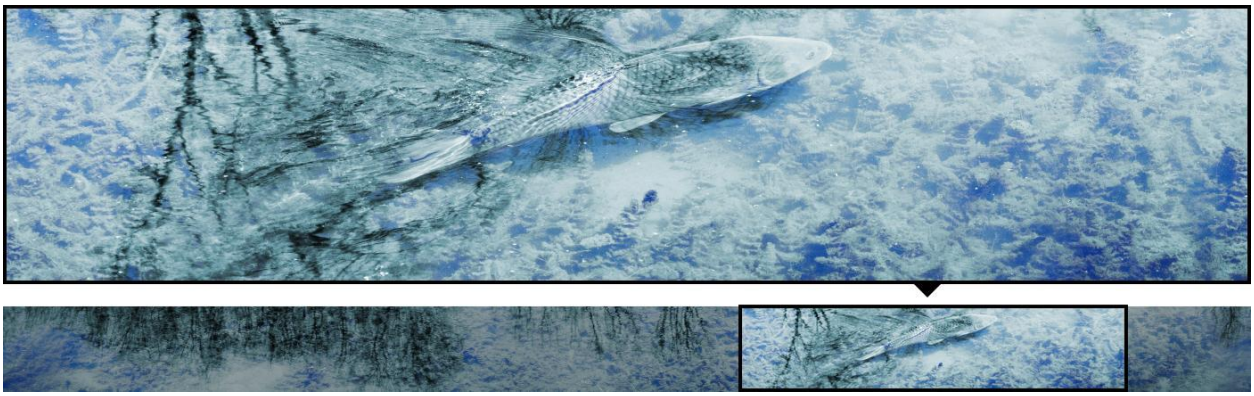

***Figure 10**: Excerpt from a wider panoramic image employed in ReptileChoir at the Factory Media Centre*

ReptileChoir was put to most interesting use by around 300 youth who attended the exhibit. Their engagement with the work was exploratory, exhibiting a focus on identity construction and interaction, rather than the creation of music. Youth used the interface to dramatically differentiate their avatars, using their "name" to engage in chat-like behaviours. In some mobile device art, "communication between the audience members is a by product of active engagement" (Oh & Wang, 2011, p.3), but with *ReptileChoir*, many participants made communication the focus of their experience, rather than a by product.

***Figure 11**: Children spawn a bale of turtles: the ReptileChoir (Photo: Débora Silva de Jesus)*

Keeping the visuals on a large collective screen keeps our collective heads up, and our attention outward: I see this as a fitting counterpoint to the more inward posture of engagement with a smartphone. In *ReptileChoir*, the smartphone is a control interface, following the paradigm of buttons rather than of any other kind of immersive/panning environment.

The irony that reptiles lack vocal chords is not lost on the fact that this installation is neither choral nor

performative, but primarily interactive and participatory. If the choral milieu is about 'singing together', Reptile choir has perhaps only kept the 'together'. Since *ReptileChoir* presents only a thin masquerade of a choral setting, it cannot claim to enhance engagement for any choral audience. But by subverting the terms of the choral milieu — perhaps at the cost of offending choral purists — it incidentally approaches an equally vibrant sort of interaction: that of the LAN party.

StreetSong

StreetSong is a new media choral performance which takes as its "score" the repurposed semiotics of Ontario road surfaces. In *StreetSong*, I pay attention to the familiar yellow language of dashed and solid lane markings, but also to the more organic (and less predictable) patterns of weathering, cracks, and tar-fill repairs. This visual language is brought about by a complex interplay of forces: historic settlement patterns, present-day urban planning, traffic, frost, erosion, and light conditions all contribute to the modalities of the surface.

Campbellville Road — near Mountsberg Conservation area in Milton, Ontario — offers a productive site for a dash-cam recording: a variegated array of tar-fill repairs make for an interesting dataset. It is also interpreted and apprehended by a particular traversal through that landscape. The dash cam video is converted to a photograph via a programmatically obtained cross section.

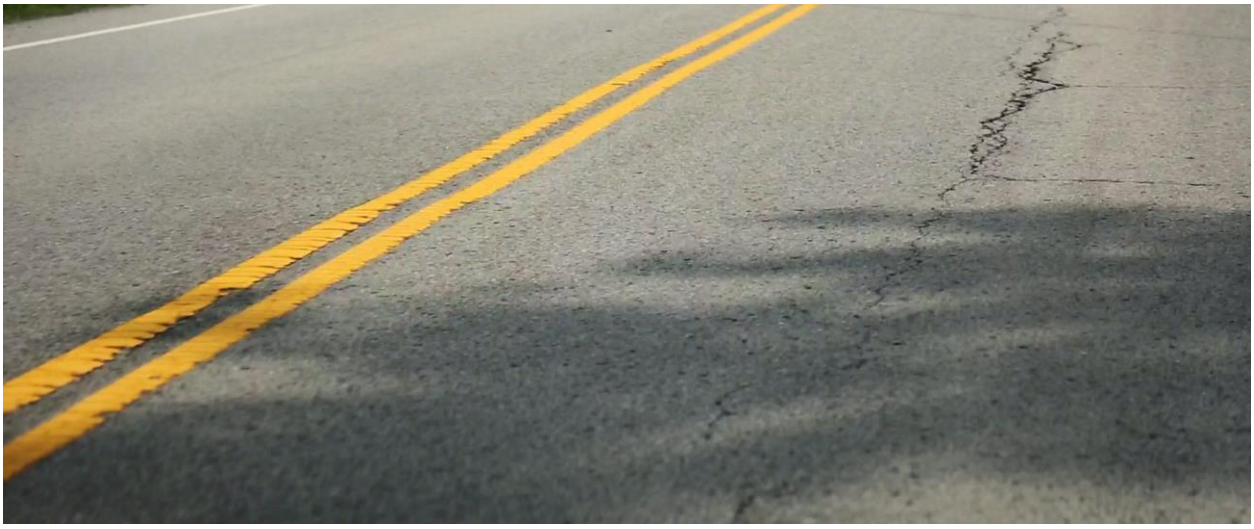

***Figure 12**: Video Still from a dash-cam recording along Campbellville Road*

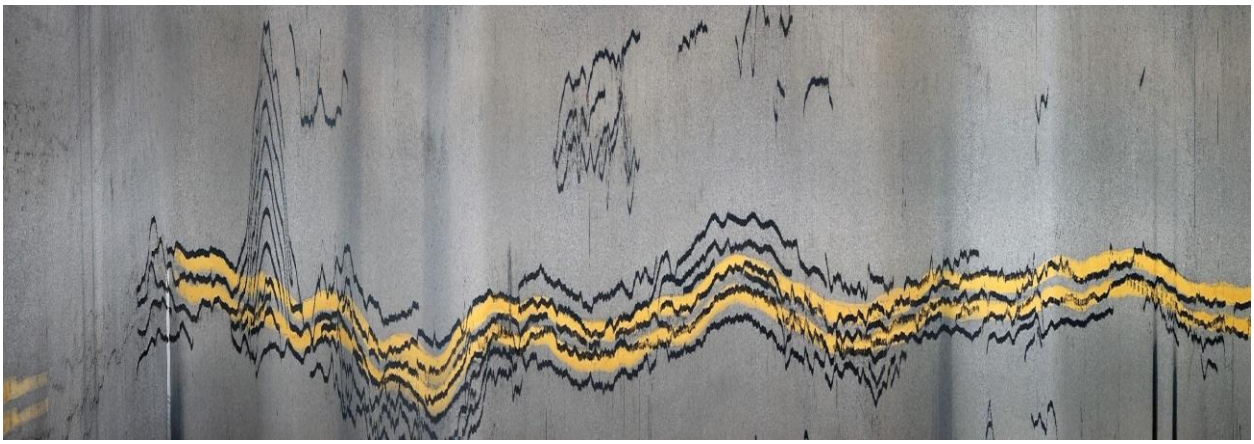

Figure 13*: Cross section of a dash-cam recording along Campbellville Road*

StreetSong begins with an overhead screen connected to the artist's laptop, displaying a cross-sectional panorama of a street. Audience members then browse to a URL, where they are invited to collaborate in the production by singing along. (Collaborative interfaces for conducting, composing and lyricizing are also in the works, existing currently via code only). Connected singers appear as avatars on the overhead display. We then begin to pan across the image as per a set conducting tempo. A series of parameters derived from the image are visualized as signal graphs overhead, whilst simultaneously triggering musical events on participants' phones. These algorithmic translations of the image data appear on audience screens as real time score notation, and sound on audience speakers as pitch cues. Lyrics, in keeping my vowel-emphasis in *ReptileChoir* are selected from long-vowel synonyms for street (A: Lane, E: Street, I: Drive, O: Road, U: Route). Like the previous works in this iterative process, technical affordances have been created using Apert, as well as the Web Audio API, JavaScript, CSS, HTML5 and JQuery. Musical parameters are broadcasted to all devices on an ongoing basis, providing them with the needed information to 'choose' a relevant note for the moment.

In traditional instruments, the control mechanism is often also the sound source — as with violin strings — but electronic instruments tend to separate the two. The core of an electronic instrument involves a mapping of input or sensor parameters to output or control parameters. This mapping tends towards simple one-to-one outcomes, as when a slider directly influences volume, or as in *Sedimentary*, when brightness directly influences pitch. It is also possible to make more complex parameter mappings: and these are facilitated by using a multi-layered mapping strategy. (Hunt et al, 2002, p. 429) In *StreetSong*, raw image data is mapped onto semantic, analytical, descriptive features in a first layer, is combined and reconfigured to map onto semantic musical categories in a second layer of mapping, and is finally translated from high-level semantic descriptions of the musical output into the concrete

controls of sound and image synthesis in a third layer of mapping. While one might be tempted to add the choir as a kind of 4th layer, a healthy respect for singers will see them as neither part of the instrument, nor as instrumental: their embodied interpretive capacities bring an important performative agency to bear on the realtime score and pitch cues.

In *StreetSong*, the mode of image traversal shapes the way multi-layered mapping is implemented. Whereas the first two works (*Sedimentary* and *Reptile Choir*) employed multiple lenses and avatars to define points of interest in the image, *StreetSong* represented a global 'now' with a moving vertical bar, spanning at any given moment across a single column of pixels. Within a given pixel column, properties were extracted, such as the number of lines/continuities, the thickness of those continuities, and the overall brightness. Processing these signals mathematically (primarily as slopes, standard deviations and averages) these values were then mapped onto semantically useful categories, like line count, line variation, peakishness and troughishness. These signals in turn inform musical parameters like pitchiness, chord dissonance, soprano boost, and bass boost. These musical parameters finally shape the way in which notes are selected for connected devices.

The sonic texture of *StreetSong* is somewhat aleatoric. Singers' notes overlap, starting and stopping at arbitrary moments (without special regard to synchronization). This approach is reminiscent of *echobo*, a 2012 iOS app through which audience members create sound with a harmonic, rather than a rhythmic consonance. (Lee, S. W., & Freeman, J., 2013, p.453) This arrangement is an aesthetic choice more than technical necessity. Natural choral latency generally exceeds network latency. Even so, the nature of the network does shape the kind of music we make with it. (Tanaka 2006, p.274).

The generation of a choral score in real time creates a productive conceptual tension with the traditional score. Real time notation systems are open ended, resulting in distinct variations at each performance, and inviting a rethinking of how algorithms, humans, and audiences relate. (Freeman, 2008, p. 34) In this situation the traditional choral rehearsal (practicing a particular set of notes in preparation for performance) becomes somewhat moot, as the notes will differ on each occasion. Rehearsal might still be useful, but would carry a new focus. Practices might tend towards a kind of bricolage, (Levi-Strauss, 1962, p.11) where participants iteratively expand their knowledge of how the system works, and their vocal capacity to sing with it, but also to contribute ideas for how the entirety of the system might be adjusted and expanded. Rehearsal would be less about rote, and more about playing and exploring. While the current focus of StreetSong is on the realtime score generation, future iterations will involve collaborators in further musical opportunities, including the creation of chord structure, and lyrical content.

To really enjoy a thing, it may be fruitful to slice it open: a kind of expansion by means of a flattening. So it is here with my cross sectioning a dashcam video: it gets at the productive interestingness of traversing a street by car. This, along with the networked slicing and dicing whereby the steering wheel becomes an asynchronous control-knob on a synthesizer, is simultaneously an algorithmic and an embodied artform. As much as we are here involving centralization, and enframing, and control, subtle agencies remain: of singers and lungs, of erosion and wind, of a mundane meandering over tar. That everyday practices retain a potential for creative and tactical production (in both the collaborative and individual aspects of this work) is to me an encouragement. On the street and in the choir, alternate algorithmic shortcuts are within reach: more than mere adherence to pavement markings. If it is at all possible to subvert the street, through a tactical and surreptitious creativity (De Certeau, 1984, p. 96) the meaning remains in the meander.

Conclusion

Networked smartphones may encourage the logic of mingling that already happens in traditional musical concerts when the distinct lines separating audiences and performer are intentionally blurred. (Lee & Freeman, 2013, p. 450). With or without such blurring, the experiences of participants may be enhanced. This enhancement may be costly, in that it may trivialize distinctions that are traditionally important. The technology may also leave us feeling disembodied. When musicians and computers mingle, the experience is quite immaterial, and the mind struggles to map what is happening. Adding networks and communication to this makes the situation even more difficult to track. (Rohrhuber, 2007, p.3). But there are ways to maximize clarity about how musicians and data co-create. The data visualization common to all my artworks taps into something less typical for a traditional choral context: a level of transparency about the creation process. In live-coding, the need for transparency is generally met through projection screens, on which "the negotiations of agency between performer and algorithm are made public" (Rohrhuber, 2007, p.3). But if we are going to interrupt milieus (such as the choir) where more thorough modes of veridiction are at play, an even greater measure of honesty seems warranted. Thus the live-coding TOPLAP manifesto's stated aversion to obscurantism, and the impetus to "Show us your screens" (Ward et al., 2004, p. 247) is fully applicable. Beyond transparency, I also strive to 'show screens' that approximate elegance and integration. By mean of these affordances, the panoramic inspiration, the derived data signal, the generated score, and the presence of musicians, are all opened up to participation in a concurrent flow. If this opening at all helps along a heightened awareness, the engagement of audiences and performers alike will have been enhanced.

References

Alexandraki, C., & Akoumianakis, D. (2010). *Exploring new perspectives in network music performance: The DIAMOUSES framework.* Computer Music Journal, 34(2), 66-83.

Arns, I. (2004). Read_Me, Run_Me, Execute_Me: Software and its Discontents, or: it's The Performativity of Code, Stupid'. *Read_Me: Software Art & Cultures-Edition,* 176-193.

Barraclough, T., Carnegie, D., Kapur, A. (2015) Pyxis Minor: App Design for Novel Social Music Experiences. In *Proceedings of the 21st International Symposium on Electronic Art ISEA 2015*

Becker, H. (1982). *Art Worlds*. Berkeley, CA: University of California Press.

Bell, C. (1992). *Ritual theory, ritual practice*. New York, NY: Oxford University Press.

Benjamin, W. (1968). The work of art in the age of mechanical reproduction. In H. Arendt (Ed.), *Illuminations* (pp. 217-252). New York, NY: Shocken Books.

Bennett, J. (2001). *The enchantment of modern life: attachments, crossings, and ethics.* Princeton, NJ: Princeton University Press.

Bland, B., Vookles, L., Snell, G., Hardy, P., & Dawson, T. (2013). *The panoramic river: The Hudson and the Thames*. Yonkers: Hudson River Museum.

Blesser, B., & Salter, L. R. (2009). *Spaces speak, are you listening?: experiencing aural architecture.* Cambridge, MA: MIT press.

Bundin, A. (2016). "Concert for Smartphones" [Abstract]. In Freeman, J., Lerch, A., Paradis, M. (Eds.), Proceedings of the *2nd Web Audio Conference (WAC-2016),* Atlanta.

Chion, M., & Murch, W. (1994). *Audio-vision: sound on screen*. New York: Columbia University Press.

Choi, H., & Berger, J. (2013). Waax: web audio API extension. Proceedings of the *International conference on new interfaces for musical expression*. Daejeon, Republic of Korea: Graduate School of Culture Technology, KAIST.

Collins, N., McLean, A., Rohrhuber, J., & Ward, A. (2003). Live coding in laptop performance. *Organised sound, 8*(3), 321-330.

Côrdydd. [Cordydd]. (2009, April 28). *Côrdydd, Sleep, Eric Whitacre* [Video file]. Retrieved from https://www.youtube.com/watch?v=RwqtR3xJMJw

Crossley, N. (2015). Music worlds and body techniques: On the embodiment of musicking. *Cultural Sociology, 9*(4), 471-492.

Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012, February). Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work* (pp. 1277-1286). ACM.

Daugherty, J. F. (2001). On the voice: Rethinking how voices work in a choral ensemble. *The Choral Journal*, *42*(5), 69-75.

De Certeau, M. (1984). Walking in the City in *The practice of everyday life.* Berkeley, CA: University of California Press (pp.91-110)

Essl, G., & Rohs, M. (2009). Interactivity for mobile music-making. *Organised Sound, 14*(02), 197-207.

Freeman, J. (2005). Large audience participation, technology, and orchestral performance. In *Proceedings of the 2005 International Computer Music Conference* (pp. 757-760).

Freeman, J. (2008). Extreme sight-reading, mediated expression, and audience participation: Real-time music notation in live performance.*Computer Music Journal*, *32*(3), 25-41.

Garnett, L. (2009). *Choral conducting and the construction of meaning: Gesture, voice, identity*. Burlington, VT: Ashgate Publishing Company.

Gritten, A., & King, E. (Eds.). (2006). *Music and gesture*. New York: Ashgate Publishing, Ltd..

Hall, S. (2009). Encoding/decoding. In M. G. Durham, & D. M. Kellner (Eds.), *Media and cultural studies: Keyworks*. Malden, MA: Blackwell Publishing Ltd.

Hand, M. (2012). *Ubiquitous photography*. Malden, MA: Polity.

Heidegger, M. (1954). The question concerning technology. In Hanks, C. (Ed.), *Technology and values: Essential readings,* Chichester, UK: Wiley-Blackwell, 99-113.

Hunt, A., Wanderley, M. M., & Paradis, M. (2003). The importance of parameter mapping in electronic instrument design. *Journal of New Music Research*, *32*(4), 429-440.

Latour, B. (2013). *An inquiry into modes of existence*. Cambridge, MA: Harvard University Press.

Lee, S. W., & Freeman, J. (2013). Echobo: audience participation using the mobile music instrument. Proceedings of the *International conference on new interfaces for musical expression*. Daejeon, Republic of Korea: Graduate School of Culture Technology, KAIST.

Strauss, C. L. (1962). *Savage mind*. Chicago, IL: University of Chicago.

McLuhan, M. (1964). Understanding Media. New York: McGraw Hill.

McManus, C. (2004). *Right hand, left hand: The origins of asymmetry in brains, bodies, atoms and cultures*. Cambridge, MA: Harvard University Press.

Manovich, L. (2001). *The language of new media*. Cambridge, MA: MIT press.

Manovich, L. (2014). Software is the Message. *Journal of Visual Culture*,*13*(1), 79-81.

Oh, J., & Wang, G. (2011). *Audience-participation techniques based on social mobile computing*. Ann Arbor, MI: Michigan Publishing, University of Michigan Library.

Ogborn, D. (2016) *Apert*.[Github repository] Retrieved from https://github.com/d0kt0r0/apert

Rebelo, P. (2009). Dramaturgy in the Network. *Contemporary Music Review*,*28*(4-5), 387-393.

Roberts, C., Wakefield, G., Wright, M., & Kuchera-Morin, J. (2015). Designing musical instruments for the browser. *Computer Music Journal*,*39*(1), 27-40.

Rohrhuber, J., de Campo, Alberto., Wieser, R., van Kampen, J., Ho, E., & Hölzl, E. (2007). Purloined letters and distributed persons. In *Music in the Global Village Conference,* Budapest, HU.

Small, C. (2011). *Musicking: The meanings of performing and listening.* Middletown, CT: Wesleyan University Press.

Sontag, S. (1977). *On photography*. New York, NY: Macmillan.

Starr, S. L., & Griesemer, J. R. (1989). Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science, 19*(3), 387-420.

Van Dijck, J. (2008). Digital photography: communication, identity, memory. *Visual Communication*, *7*(1), 57-76.

Ward, A., Rohrhuber, J., Olofsson, F., McLean, A., Griffiths, D., Collins, N., & Alexander, A. (2004). Live algorithm programming and a temporary organisation for its promotion. In *Proceedings of the README Software Art Conference*.

Wenger-Trayner, E., & Wenger-Trayner, B. (2015). *Introduction to communities of practice.* Retrieved from: http://wenger-trayner. com/introduction-to-communities-of-practice/.

Winget, M. A. (2008). Annotations on musical scores by performing musicians: Collaborative models, interactive methods, and music digital library tool development. *Journal of the American Society for Information Science and Technology*, *59*(12), 1878-1897.

Wyse, L., & Subramanian, S. (2013). The viability of the web browser as a computer music platform. *Computer Music Journal*, *37*(4), 10-23.

Yun, G. J., & Willingham, L. (2014). JABBLE! Choral Improvisation: A Model of Shared Leadership. *The Phenomenon of Singing*, *9*, 238-250.

Zielinski, S. (2006). *Deep time of the media: Toward an archaeology of hearing and seeing by technical means*. Cambridge, MA: MIT press.

Appendix 1: Code

All of the code written and used for this project is open source. It is available on the attached media, where it is organized into the following folders. The code assumes the context of a webserver configured with PHP and node.js.

- /Code-Apert-Server/ – Apert, Node.js, and Javascript for *Sedimentary, ReptileChoir, and StreetSong*
- /Code-Sedimentary/ – Display Code for *Sedimentary*
- /Code-ReptileChoir/ – Display Code for *ReptileChoir*
- /Code-StreetSong/ – Display Code for *StreetSong*
- /Code-PeaceWeave/ – Display Code for *PeaceWeave*

Portions of this code will be made available on my public GitHub repository at https://github.com/nsitu. What follows below is the code I created in JavaScript, Canvas, and HTML for *PeaceWeave*, *Sedimentary*, *ReptileChoir*, and *StreetSong*. I have also included a relevant Photoshop script.

PeaceWeave Display Interface

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>PAX - The Peace Within</title>
  <meta name="author" description="Harold Sikkema">
  <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Oxygen:400,300,700">
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
<div id="theBubbles"></div>
<div id="theContent" style="display: none;">
  <h1 id="theTitle">Title</h1>
  <p id="theSubtitle">theSubtitle</p>
</div>
<script type="text/javascript" src="jquery-1.12.1.min.js"></script>
<script type="text/javascript" src="jquery-ui/jquery-ui.min.js"></script>
<script type="text/javascript" src="gamepad.js"></script>
<script>
var thePlaylist=[];
thePlaylist[0]= {'thePath':'Tracks/muc-intro.mp4', 'theTitle': ' ', 'theSubtitle' : ' '};
thePlaylist[1]= {'thePath':'Tracks/03-Eternity/03-Eternity.mp4', 'theTitle': 'Tonight Eternity Alone', 'theSubtitle' :
'René Clausen'};
thePlaylist[2]= {'thePath':'Tracks/01-Kyrie/01-Kyrie.mp4', 'theTitle': 'Kyrie', 'theSubtitle' : 'Timothy Corlis'};
thePlaylist[3]= {'thePath':'Tracks/02-Gloria/02-Gloria.mp4', 'theTitle': 'Gloria', 'theSubtitle' : 'Timothy Corlis'};
thePlaylist[4]= {'thePath':'Tracks/04-Stars/04-Stars.mp4', 'theTitle': 'Stars', 'theSubtitle' : 'Ēriks Ešenvalds'};
thePlaylist[5]= {'thePath':'Tracks/05-Richte/05-Richte.mp4', 'theTitle': 'Richte mich, Gott', 'theSubtitle' : 'Felix
Mendelssohn'};
thePlaylist[6]= {'thePath':'Tracks/06-Tiger/06-Tiger.mp4', 'theTitle': 'The Tiger', 'theSubtitle' : 'Lauren Bernofsky'};
thePlaylist[7]= {'thePath':'Tracks/07-Lamb/07-Lamb.mp4', 'theTitle': 'The Lamb', 'theSubtitle' : 'John Tavener'};
thePlaylist[8]= {'thePath':'Tracks/08-After/08-After.mp4', 'theTitle': 'After the War', 'theSubtitle' : 'Mark Sirett'};
thePlaylist[9]= {'thePath':'Tracks/09-Rytmus/09-Rytmus.mp4', 'theTitle': 'Rytmus', 'theSubtitle' : 'Ivan Hrušovský'};
thePlaylist[10]= {'thePath':'Tracks/10-Muusika/10-Muusika.mp4', 'theTitle': 'Muusika', 'theSubtitle' : 'Pärt
Uusberg'};
thePlaylist[11]= {'thePath':'Tracks/muc-intro.mp4', 'theTitle': ' ', 'theSubtitle' : ' '};
function playVideo(playlistIndex){
  theVideo.src = thePlaylist[playlistIndex].thePath;
  $('#theTitle').html(thePlaylist[playlistIndex].theTitle );
```

```
  $('#theSubtitle').html(thePlaylist[playlistIndex].theSubtitle );
  $('#theContent').fadeIn().delay(5000).fadeOut();
}
var wiiData = [];
var wiiSum = [];
var wiiAvg = [];
var currentX, currentY, currentZ, previousX, previousY, previousZ;
var wiiCoefficient = 1;
var wiiExcitement = 1;
wiiData[0] = Array([0.1,0.1]);
wiiData[1] = Array([0.1,0.1]);
wiiData[2] = Array([0.1,0.1]);
var numImages = 395;
function theOtherList(theCurrent){  if (theCurrent == 'one'){ return 'two'; }  else{ return 'one'; } }
var imageLists = {  'one' : [],  'two' : []  };
for (i=0; i<numImages; i++){ imageLists['one'][i] = (i+1); }
var currentList = 'one';
function theImageNumber(){
   if ( imageLists[currentList].length < 1 ){ currentList = theOtherList(currentList); }
   var imgNumber = imageLists[currentList].splice( Math.floor(Math.random() * imageLists[currentList].length )
, 1); // Splice out a random element using the ri var
   imageLists[theOtherList(currentList)].unshift(theImageNumber);
   return ("0000" + imgNumber).substr(-3,3);
}
$( document ).ready(function() {
 // on pageload, play a generic mcmaster choir video
 var theVideo = $('<video />', { id: 'theVideo',  type: 'video/mp4',  controls: false });
 theVideo.insertAfter('#theBubbles');
 playVideo(0);
setInterval(function() {
 //if there's a wii connected get the overall acceration pulse+
 var pAverage;    pAverage = 1;
 if (typeof(theControllers) != "undefined" ){
  var pCurrent;
  for (j in theControllers) {
   var theController = theControllers[j];
   previousX = currentX;
   previousY = currentY;
   previousZ = currentZ;
   currentX = theController.axes[0];
   currentY = theController.axes[1];
   currentZ = theController.axes[2];
   wiiData[0].unshift( Math.abs(currentX - previousX) );
   wiiData[1].unshift( Math.abs(currentY - previousY) );
   wiiData[2].unshift( Math.abs(currentZ - previousZ) );
   if ( wiiData[0].length > 10 ){ wiiData[0].pop(); }
   if ( wiiData[1].length > 10 ){ wiiData[1].pop(); }
   if ( wiiData[2].length > 10 ){ wiiData[2].pop(); }
   wiiSum[0] = wiiData[0].reduce(function(a, b) { return a + b; });
   wiiSum[1] = wiiData[1].reduce(function(a, b) { return a + b; });
   wiiSum[2] = wiiData[2].reduce(function(a, b) { return a + b; });
   wiiAvg[0] = ( wiiSum[0] / wiiData[0].length );
   wiiAvg[1] = ( wiiSum[1] / wiiData[1].length );
   wiiAvg[2] = ( wiiSum[2] / wiiData[2].length );
   wiiExcitement = Math.min(4, wiiCoefficient * wiiData[2].reduce(function(a, b) { return a + b; }));
   //console.log(wiiExcitement);
```

```
    }
  }
}, 30);
});
var currentSize = 5;
var currentTrack = 0;
$(document).keydown(function(e) {
   switch(e.which) {
     case 107: currentTrack = Math.min( currentTrack + 1, thePlaylist.length - 1 ); playVideo(currentTrack); break;
// + Plus
     case 109: currentTrack = Math.max( currentTrack - 1, 0 ); playVideo(currentTrack); break;      // - Minus
      case 32: if ( theVideo.paused ){            theVideo.play(); } else{     theVideo.pause(); }   break;           //
spacebar pause/play
      case 38: theVideo.playbackRate = 1;       break;              // right arrow / rate up
      case 40: if ( theVideo.paused ){            theVideo.play(); } else{     theVideo.pause(); }  break;           // down
arrow  / pause/play
      case 37: theVideo.playbackRate = theVideo.playbackRate - 0.1; break;                  // up arrow
      case 39: theVideo.playbackRate = theVideo.playbackRate + 0.1;  break;     // right arrow
      case 66: bubbleBobble(); break;                                          // b
           case 67: $('#theBubbles').children().first().fadeOut().remove(); break;   // c
           case 48: currentSize = 0; bubbleBobble(); break;  // 0
           case 49: currentSize = 1; bubbleBobble(); break;  // 1
           case 50: currentSize = 2; bubbleBobble(); break;  // 2
           case 51: currentSize = 3; bubbleBobble(); break;  // 3
           case 52: currentSize = 4; bubbleBobble(); break;  // 4
           case 53: currentSize = 5; bubbleBobble(); break;  // 5
           case 54: currentSize = 6; bubbleBobble(); break;  // 6
           case 55: currentSize = 7; bubbleBobble(); break;  // 7
           case 56: currentSize = 8; bubbleBobble(); break;  // 8
           case 57: currentSize = 9; bubbleBobble(); break;  // 9
           case 188: theVideo.currentTime = Math.max(theVideo.currentTime - 10, 0 ) ; break;
           case 190: theVideo.currentTime = Math.min(theVideo.currentTime + 10, theVideo.duration); break;
   case 219:  wiiCoefficient = Math.max ( 0, wiiCoefficient - 0.2 );  console.log(wiiCoefficient); break; // [
   case 221:  wiiCoefficient = Math.min ( 1, wiiCoefficient + 0.2 );  console.log(wiiCoefficient); break; // ]
     default: return; // exit this handler for other keys
   }
   e.preventDefault(); // prevent the default action (scroll / move caret)
});
// the bubbles keep moving individually, but acquire the size of the most recent.
bubbleBobble = function(){
  var qq = new Date().getTime(); //unique
           var imgNumber =  theImageNumber();                  // between 1 and whatever
           var       imgSize = Math.floor((Math.random() * 50) + 1) + 200 + (currentSize * 50);  // between 200 and
400
  var rand = Math.random() * 1000+500,
      randother = Math.random() * 1000+500,
      randanother = Math.random() * 1000+500,
      randlow = Math.random().map(0,1,0.5,1);
           $('#theBubbles').append('<div id="bc_' + qq + '"><div id="b_' + qq + '"></div></div>');
           $('#b_' + qq).css('z-index', qq);
           //$('#b_' + qq).css('background-image', 'url(images/' + imgNumber + '.png)');
  $('#b_' + qq).css('background-image', 'url(800px/' + imgNumber + '.png)');
           $('#b_' + qq).css('background-size',  imgSize + 'px ' + imgSize + 'px' );
           $('#b_' + qq).css('width', imgSize + 'px' );
           $('#b_' + qq).css('height', imgSize + 'px' );
           $('#bc_' + qq).css('margin-left', Math.floor(1920 - imgSize) + 'px' );
```

```
setInterval(function() {
                    if ($('#bc_' + qq).length){
    // get global acceleration data
    if ( $('#bc_' + qq).css('margin-left').slice(0,-2) < ( imgSize * -1 ) ){   clearInterval();  $('#bc_' + qq).remove(); }
                    var       newSize = imgSize + Math.floor( Math.sin( new Date().getTime() / randanother  + 1000
          ) * 14 ) + ( wiiExcitement * 10 * randlow);
                    $('#b_' + qq).css('background-size',  newSize + 'px ' + newSize + 'px' );
                    $('#b_' + qq).css('width', newSize + 'px' );
                    $('#b_' + qq).css('height', newSize + 'px' );
                    $('#b_' + qq).css('top', Math.floor( Math.sin(new Date().getTime() / rand  +1000            ) * 20 +
40 + (10 * wiiExcitement ) )  );
                    $('#b_' + qq).css('transform', 'rotate(' + ( Math.sin(     new Date().getTime() / randother    ) * ( 5 +
( wiiExcitement * 2  )  ) )  + 'deg)');
                    $('#bc_' + qq).css('margin-left', function(n, v) { return  parseFloat(v) - 3; } );              // maybe
you can randomize the bubble flow rate?
   }
  }, 30);
};
</script>
</body>
</html>
```

PeaceWeave Display Interface CSS

```
/*!
Styles for full screen background video demo
*/
/* =============================================================================
  RESETS
============================================================================= */
html,
body,
div,
h1,
p,
a,
video {
  margin: 0;
  padding: 0;
}
/* =============================================================================
  HTML, BODY
============================================================================= */
html,
body {
  height: 100%;
}
body {
  font-size: 16px;
  font-family: "Oxygen", sans-serif;
  line-height: 1.5;
}
/* =============================================================================
  BUBBLES
============================================================================= */
#theBubbles {
```

```
  position: absolute;
  z-index: 5;
  margin: 0 auto;
  overflow: hidden;
  width: 100%;
  height: 100%;
  text-align: center;
}
#theBubbles  div{
            position: absolute;
}
#parent {
    left: 0;
    top: 0;
    width: 400px;
    height: 100%;
}

/* ======================================================================
  CONTENT
====================================================================== */
#theContent {
  position: absolute;
  top: 30%;
  z-index: 4;
  margin: 0 auto;
  width: 1920px;
  text-align: center;
}
#theTitle {
  margin-bottom: 24px;
  color: #ddd;
  font-size: 44px;
}
#theSubtitle {
  margin-bottom: 24px;
  color: #ddd;
  font-size: 22px;
}
/* ======================================================================
  Overlay
====================================================================== */
#theCanvas {
  position: fixed;
  z-index: 3;
  position: fixed;
  top: 50%;
  left: 50%;
  min-width: 100%;
  min-height: 100%;
  width: auto;
  height: auto;
  -webkit-transform: translate(-50%, -50%);
     -ms-transform: translate(-50%, -50%);
         transform: translate(-50%, -50%);
```

```
}
/* ================================================================================
  Overlay
================================================================================ */
#theOverlay {
  position: fixed;
  z-index: 2;
  position: fixed;
  top: 50%;
  left: 50%;
  min-width: 100%;
  min-height: 100%;
  width: auto;
  height: auto;
  -webkit-transform: translate(-50%, -50%);
     -ms-transform: translate(-50%, -50%);
         transform: translate(-50%, -50%);
  opacity: 0.5;
  background: red;
}
/* ================================================================================
  VIDEO
================================================================================ */
#theVideo {
z-index: 1;
  position: fixed;
  top: 50%;
  left: 50%;
  min-width: 100%;
  min-height: 100%;
  width: auto;
  height: auto;
  -webkit-transform: translate(-50%, -50%);
     -ms-transform: translate(-50%, -50%);
         transform: translate(-50%, -50%);
}
```

PeaceWeave Bubble Generation Script for Photoshop

```
// Adapted from a script by Trevor Morris (trevor@morris-photographics.com)
// enable double-clicking from Mac Finder or Windows Explorer
// this command only works in Photoshop CS2 and higher
#target photoshop
// bring application forward for double-click events
app.bringToFront();
//////////////////////////////////////////////////////////////////////////////
// main - main function
//////////////////////////////////////////////////////////////////////////////
function main() {
        // user settings
        var prefs = new Object();
        prefs.sourceFolder      = '~';  // default browse location (default: '~')
```

```
	prefs.removeFileExtensions = true; // remove filename extensions for imported layers (default: true)
	prefs.savePrompt           = false; // display save prompt after import is complete (default: false)
	prefs.closeAfterSave       = false; // close import document after saving (default: false)
	// prompt for source folder
	var sourceFolder = Folder.selectDialog('Please select the folder to be imported:',
Folder(prefs.sourceFolder));
	// ensure the source folder is valid
	if (!sourceFolder) {
		return;
	}
	else if (!sourceFolder.exists) {
		alert('Source folder not found.', 'Script Stopped', true);
		return;
	}
	// add source folder to user settings
	prefs.sourceFolder = sourceFolder;
	// get a list of files
	var fileArray = getFiles(prefs.sourceFolder);
	// if files were found, proceed with import
	if (fileArray.length) {
		importFolderAsLayers(fileArray, prefs);
	}
	// otherwise, diplay message
	else {
		alert("The selected folder doesn't contain any recognized images.", 'No Files Found', false);
	}
}
////////////////////////////////////////////////////////////////////////////
// getFiles - get all files within the specified source
////////////////////////////////////////////////////////////////////////////
function getFiles(sourceFolder) {
	// declare local variables
	var fileArray = new Array();
	var extRE = /\.(?:png|gif|jpg|bmp|tif|tga|psd)$/i;
	// get all files in source folder
	var docs = sourceFolder.getFiles();
	var len = docs.length;
	for (var i = 0; i < len; i++) {
		var doc = docs[i];
		// only match files (not folders)
		if (doc instanceof File) {
			// store all recognized files into an array
			var docName = doc.name;
			if (docName.match(extRE)) {
				fileArray.push(doc);
			}
		}
	}
	// return file array
```

```
	return fileArray;
}
////////////////////////////////////////////////////////////////////////////
// importFolderAsLayers - imports a folder of images as named layers
////////////////////////////////////////////////////////////////////////////
function importFolderAsLayers(fileArray, prefs) {
	// create a new document
	var newDoc = documents.add(1920, fileArray.length, 72, 'Imported Layers', NewDocumentMode.RGB,
DocumentFill.WHITE, 1);
	var newLayer = newDoc.activeLayer;
	// loop through all files in the source folder
	for (var i = 0; i < fileArray.length; i++) {
		// open document
		var doc = open(fileArray[i]);
		// get document name (and remove file extension)
		var name = doc.name;
		if (prefs.removeFileExtensions) {
			name = name.replace(/(?:\.[^.]*$|$)/, '');
		}
		// convert to RGB; convert to 8-bpc; merge visible
		//doc.changeMode(ChangeMode.RGB);
		//doc.bitsPerChannel = BitsPerChannelType.EIGHT;
		//doc.artLayers.add();
		//doc.mergeVisibleLayers();
		// rename layer; duplicate to new document
		var layer = doc.activeLayer;
		layer.name = name;
		layer.duplicate(newDoc, ElementPlacement.PLACEATBEGINNING);
		app.activeDocument = newDoc;
		var layerDup = newDoc.artLayers.getByName(name);
		MoveLayerTo(layerDup,0,i);
		newDoc.mergeVisibleLayers();
		app.activeDocument = doc;
		// close imported document
		doc.close(SaveOptions.DONOTSAVECHANGES);
	}
	// delete empty layer; reveal and trim to fit all layers
	// newLayer.remove();
	// newDoc.revealAll();
	newDoc.trim(TrimType.TRANSPARENT, true, true, true, true);
	// save the final document
	if (prefs.savePrompt) {
		// PSD save options
		var saveOptions = new PhotoshopSaveOptions();
		saveOptions.layers = true;
		saveOptions.embedColorProfile = true;
		// prompt for save name and location
		var saveFile = File.saveDialog('Save the new document as:');
		if (saveFile) {
```

```
                        newDoc.saveAs(saveFile, saveOptions, false, Extension.LOWERCASE);
                }
                // close import document
                if (prefs.closeAfterSave) {
                        newDoc.close(SaveOptions.DONOTSAVECHANGES);
                }
        }
}
////////////////////////////////////////////////////////////////////////////
// isCorrectVersion - check for Adobe Photoshop CS2 (v9) or higher
////////////////////////////////////////////////////////////////////////////
function isCorrectVersion() {
        if (parseInt(version, 10) >= 9) {
                return true;
        }
        else {
                alert('This script requires Adobe Photoshop CS2 or higher.', 'Wrong Version', false);
                return false;
        }
}
////////////////////////////////////////////////////////////////////////////
// showError - display error message if something goes wrong
////////////////////////////////////////////////////////////////////////////
function showError(err) {
        if (confirm('An unknown error has occurred.\n' +
                'Would you like to see more information?', true, 'Unknown Error')) {
                        alert(err + ': on line ' + err.line, 'Script Error', true);
        }
}
function MoveLayerTo(fLayer,fX,fY) {
  var Position = fLayer.bounds;
  Position[0] = fX - Position[0];
  Position[1] = fY - Position[1];
  fLayer.translate(-Position[0],-Position[1]);
}

// test initial conditions prior to running main function
if (isCorrectVersion()) {
        // remember ruler units; switch to pixels
        var originalRulerUnits = preferences.rulerUnits;
        preferences.rulerUnits = Units.PIXELS;
        try {
                main();
        }
        catch(e) {
                // don't report error on user cancel
                if (e.number != 8007) {
                        showError(e);
                }
```

```
        }
        // restore original ruler unit
        preferences.rulerUnits = originalRulerUnits;
}
```

PeaceWeave WiiMote Script for FreePIE (To map Wii onto virtual gamepad)

```
def the_motionplus():
 diagnostics.watch(wiimote[0].ahrs.yaw)
 diagnostics.watch(wiimote[0].ahrs.pitch)
 diagnostics.watch(wiimote[0].ahrs.roll)
 diagnostics.watch(wiimote[0].motionplus.yaw_down)
 diagnostics.watch(wiimote[0].motionplus.pitch_left)
 diagnostics.watch(wiimote[0].motionplus.roll_left)
 vJoy[0].rx = 16382 * wiimote[0].ahrs.yaw / 180
 vJoy[0].ry = 16382 * wiimote[0].ahrs.pitch / 180
 vJoy[0].rz = 16382 * wiimote[0].ahrs.roll / 180

def the_buttons():
   vJoy[0].setButton(0, wiimote[0].buttons.button_down(WiimoteButtons.A))
   vJoy[0].setButton(1, wiimote[0].buttons.button_down(WiimoteButtons.B))
   vJoy[0].setButton(2, wiimote[0].buttons.button_down(WiimoteButtons.One))
   vJoy[0].setButton(3, wiimote[0].buttons.button_down(WiimoteButtons.Two))
   vJoy[0].setButton(4, wiimote[0].buttons.button_down(WiimoteButtons.Minus))
   vJoy[0].setButton(5, wiimote[0].buttons.button_down(WiimoteButtons.Plus))
   vJoy[0].setButton(6, wiimote[0].buttons.button_down(WiimoteButtons.Home))

def the_acceleration():
 diagnostics.watch(vJoy[0].axisMax)
 diagnostics.watch(wiimote[0].acceleration.x)
 diagnostics.watch(wiimote[0].acceleration.y)
 diagnostics.watch(wiimote[0].acceleration.z - 9.81)
 vJoy[0].x = 16382 * wiimote[0].acceleration.x / 40
 vJoy[0].y = 16382 * wiimote[0].acceleration.y / 40
 vJoy[0].z = 16382 * (wiimote[0].acceleration.z ) / 40
  # ~ 40 m/s2 is the range of acceleration values (Four G-units).

if starting:
 wiimote[0].acceleration.update += the_acceleration
 wiimote[0].motionplus.update += the_motionplus
 wiimote[0].buttons.update += the_buttons
 wiimote[0].enable(WiimoteCapabilities.MotionPlus)
```

Sedimentary Mobile JavaScript

```
theAmp = 1;
function killBeeps(){
```

```
  theAmp = 0.001;
}
var had_touch = false;
var ac;
function fireWhenReady() {
   if (typeof SC != 'undefined') {
      setupSC();
   }
   else {
      setTimeout(fireWhenReady, 100);
   }
}
function setupSC(){
    var widgetIframe = document.getElementById('soundCloud'),
       widget       = SC.Widget(widgetIframe);
    widget.bind(SC.Widget.Events.READY, function() {
      console.log('ready');
      widget.bind(SC.Widget.Events.FINISH, function() {
        console.log('finished');
      });
      // get current level of volume
      widget.getVolume(function(volume) {
        console.log('current volume value is ' + volume);
      });
      // set new volume level
      widget.setVolume(50);
      // get the value of the current position
    });
}
/*if (navigator.userAgent.match(/(iPod|iPhone|iPad)/)) {*/

$( document ).ready(function() {
fireWhenReady();
  $('<iframe>', {
    src:
'https://w.soundcloud.com/player/?url=https://soundcloud.com/nsitu/thesilence&auto_play=false&buying=false&liking=false&download=false&sharing=false&show_artwork=false&show_comments=false&show_playcount=false&show_user=false&hide_related=false&visual=false&start_track=0&callback=true',
    id:  'soundCloud',
    frameborder: 'no',
    width: '100%',
    height: '166',
    scrolling: 'no'
  }).appendTo('body');
        var scapi = document.createElement("script");
        scapi.type = "text/javascript";
        scapi.src = "https://w.soundcloud.com/player/api.js";
        $("head").append(scapi);
/* }*/
```

```
 var s = document.createElement("script");
 s.type = "text/javascript";
 s.src = "http://nsitu.ca:8080/remote.js?tsh203";
 $("head").append(s);
  $('<div/>', { id: 'theGoldPill' }).appendTo('body').css({
     'width' : '50%', 'height' : '20%',
     'position' : 'absolute', 'top': '10%', 'left' : '25%',
     'background-color': 'rgba(170, 240, 38,0.5)',
     'border-radius' : '100px'
    }).click(function(){
     simple(440,theAmp);
     playGold = true; playBlue = false; playRed = false;
      $('#theRedPill').css({'border': 'none'});
      $('#theBluePill').css({'border': 'none'});
      $('#theGoldPill').css({'border': '10px solid rgba(134, 120, 15, 1)'});
    });
 $('<div/>', { id: 'theBluePill' }).appendTo('body').css({
    'width' : '50%', 'height' : '20%',
    'position' : 'absolute', 'top': '40%', 'left' : '25%',
    'background-color': 'rgba(38,38,170,0.5)',
    'border-radius' : '100px'
  }).click(function(){
     simple(440,theAmp);
    playBlue = true; playRed = false; playGold = false;
     $('#theRedPill').css({'border': 'none'});
     $('#theGoldPill').css({'border': 'none'});
     $('#theBluePill').css({'border': '10px solid rgba(15, 40, 80, 1)'});
  });
  $('<div/>', { id: 'theRedPill' }).appendTo('body').css({
     'width' : '50%', 'height' : '20%',
     'position' : 'absolute', 'top': '70%', 'left' : '25%',
     'background-color': 'rgba(170,38,38,0.5)',
     'border-radius' : '100px'
    }).click(function(){
       simple(440,theAmp);
       playRed = true; playBlue = false; playGold=false;
       $('#theRedPill').css({'border': '10px solid rgba(80,15,15,1)'});
       $('#theBluePill').css({'border': 'none'});
       $('#theGoldPill').css({'border': 'none'});
    });
 setInterval(function() { jigglePills(); }, 60 );

 });

var rand = Math.random() * 1000+500,
   randother = Math.random() * 1000+500,
   randanother = Math.random() * 1000+500,
   bankSize = 20,
   playRed = false,
```

```
    playBlue = false,
    playGold = false;
SimpleMonoSample = function(url) {
  this.url = url;
  var request = new XMLHttpRequest();
  request.open('GET',url,true);
  request.responseType = 'arraybuffer';
  var closure = this; // a closure is necessary for...
  request.onload = function() {
     var data = request.response;
     ac.decodeAudioData(data, function(x) {
       console.log("buffer loaded");
       closure.buffer = x; // ...the decoded data to be kept in the object
     },
     function(err) {
       console.log("error decoding buffer for" + url);
     });
  };
  request.send();
  this.gain = ac.createGain();
  this.gain.connect(ac.destination);
  this.playing = false;
}
SimpleMonoSample.prototype.play = function (amp,dur,rate,startPos) {
  if(amp == null) amp = 1;
  if(rate == null) rate = 1; // if 2nd argument not given, defaults to 1
  if(startPos == null) startPos = 0.0; // if 3rd arg not given, defaults 0
  if(this.playing == false) { // only play if not already playing
    this.playing = true;
    this.source = ac.createBufferSource();
    this.source.playbackRate.value = rate;
    this.source.buffer = this.buffer;
    this.source.connect(this.gain);
    var now = ac.currentTime;
    this.source.start(now,startPos);
    this.gain.gain.setValueAtTime(0,now);
    this.gain.gain.linearRampToValueAtTime(amp,now+0.003); // 3 ms fade-in
    if (dur > 0){
      this.gain.gain.linearRampToValueAtTime(amp,now+dur-0.003);  // hold
      this.gain.gain.linearRampToValueAtTime(0,now+dur);  // 3 ms fade-out
    }
    var closure = this;
    setTimeout(function() {
      closure.playing = false; // make synth available again...
    },(dur*1000)+250); // ...a quarter second after envelope finishes
  } else console.log("warning: attempt to play synth that was already playing");
}

function apertInitialize() {
```

```
// the purpose of this is for synths to be created after the audio context exists
 theSamples = {
  collideBank : new Array(bankSize),
  redBank : new Array(bankSize),
  blueBank : new Array(bankSize),
  goldBank : new Array(bankSize)
 }
 for(var n=0;n<bankSize;n++) {
  console.log("created synth " + n);
  // RockScrape4 short rough scrape
  // RockScrape10 long quiet scrape
  // RockScrape11 clink clink
  // RockScrape20 louder scrape
  var collideTracks = ["PrimeOrdeal1.wav"];
  var redTracks = ["RockScrape11.wav", "RockScrape19.wav"];
  var blueTracks = ["RockScrape20.wav"];
  var goldTracks = ["Whistle1.wav","Whistle12.wav","Whistle13.wav"]
  var collideFile = collideTracks[Math.floor(Math.random()*collideTracks.length)];
  var redFile = redTracks[Math.floor(Math.random()*redTracks.length)];
  var blueFile = blueTracks[Math.floor(Math.random()*blueTracks.length)];
  var goldFile = goldTracks[Math.floor(Math.random()*goldTracks.length)];
  theSamples.collideBank[n] = new SimpleMonoSample(collideFile);
  theSamples.redBank[n] = new SimpleMonoSample(redFile);
  theSamples.blueBank[n] = new SimpleMonoSample(blueFile);
  theSamples.goldBank[n] = new SimpleMonoSample(goldFile);
 }
}
function redPulse(){
 $('#theRedPill').css('background-color', 'rgba(170,38,38,1)' ).animate({'background-color':'rgba(170,38,38,0.5)'},
100);
}
function bluePulse(){
 $('#theBluePill').css('background-color', 'rgba(38,38,170,1)' ).animate({'background-color':'rgba(38,38,170,0.5)'},
100);
}
function goldPulse(){
 $('#theGoldPill').css('background-color', 'rgba(170,240,38,1)' ).animate({'background-
color':'rgba(170,240,38,0.5)'}, 100);
}

function jigglePills(){
 $('#theRedPill').css('transform', 'rotate(' + ( Math.sin(          ( new Date().getTime()+rand) / 2000          ) * 5 )  +
'deg)');
 $('#theBluePill').css('transform', 'rotate(' + ( Math.sin(          (new Date().getTime()-randanother) / 2000  ) * 5 )  +
'deg)');
 $('#theGoldPill').css('transform', 'rotate(' + ( Math.sin( (        new Date().getTime()+randother ) / 2000     ) * 5 )  +
'deg)');
}
function collideSound(amp,dur,rate,startPos) {
```

```
 killBeeps();
 //redPulse();
 var n;
 for(n=0;n<bankSize;n++) {  if( theSamples.collideBank[n].playing==false ) break; }
 if(n<bankSize) {   theSamples.collideBank[n].play(amp,dur,rate,startPos);  }
 else console.log("warning: all collide synth instances already playing");
}
function redSound(amp,dur,rate,startPos) {
 killBeeps();
 if (!playRed) return;  redPulse();  var n;
 for(n=0;n<bankSize;n++) {  if( theSamples.redBank[n].playing==false ) break; }
 if(n<bankSize) {   theSamples.redBank[n].play(amp,dur,rate,startPos);  }
 else console.log("warning: all red synth instances already playing");
}
function blueSound(amp,dur,rate,startPos) {
 killBeeps();
 if (!playBlue) return;  bluePulse();  var n;
 for(n=0;n<bankSize;n++) {  if( theSamples.blueBank[n].playing==false ) break; }
 if(n<bankSize) {   theSamples.blueBank[n].play(amp,dur,rate,startPos);  }
 else console.log("warning: all blue synth instances already playing");
}

function goldSound(amp,dur,rate,startPos) {
 killBeeps();
 if (!playGold) return;  goldPulse();  var n;
 for(n=0;n<bankSize;n++) {  if( theSamples.goldBank[n].playing==false ) break; }
 if(n<bankSize) {   theSamples.goldBank[n].play(amp,dur,rate,startPos);  }
 else console.log("warning: all red synth instances already playing");
}

function simple(freq,amp) {
 apertStartAudio();
        var sine = ac.createOscillator();
        sine.type = 'sine';
        sine.frequency.value = freq;
        var gain = ac.createGain();
        sine.connect(gain);
        gain.connect(ac.destination);
        sine.start();
        // envelope
        var now = ac.currentTime;
        gain.gain.setValueAtTime(0,now);
 gain.gain.linearRampToValueAtTime(amp,now+0.005); gain.gain.linearRampToValueAtTime(0,now+0.405);
        // schedule cleanup
        setTimeout(function() {
                sine.stop();
                sine.disconnect(gain);
                gain.disconnect(ac.destination);
        },1000);
```

```
};
```

Sedimentary Display Interface HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Escarpment Traversal</title>
  <meta name="author" description="Harold Sikkema">
  <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Oxygen:400,300,700">
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
<textarea id="theCode"> </textarea>
<div id="doneCode"></div>
<div id="theFrame">
  <div id="theRedPill"></div>
  <div id="theBluePill"></div>
  <div id="theGoldPill"></div>
</div>
<script type="text/javascript" src="jquery-1.12.1.min.js"></script>
<script type="text/javascript" src="jquery.color.js"></script>
<script>
var ws;
function sedimentary(){  collide(); }
function setup(theServer) {
  window.WebSocket = window.WebSocket || window.MozWebSocket;
if (theServer == null){
theServer = 'nsitu.ca';
}
  var url = 'ws://' + theServer + ':8000';
  console.log("attempting websocket connection to " + url);
  ws = new WebSocket(url);
  ws.onopen = function () { console.log("websocket connection opened"); };
  ws.onerror = function () { console.log("ERROR opening websocket connection"); };
  ws.onmessage = function (m) {
    var data = JSON.parse(m.data);
    if(data.type =='all') { console.log("all " + data.name); }
    else if(data.type =='refreshCount') {
      /*document.getElementById('refreshCount').textContent = data.count;*/
    }
    else if(data.type =='clientCount') {
      /*document.getElementById('clientCount').textContent = data.count; */
    }
    else {
      console.log("received WebSocket message of unknown type = " + data.type);
    }
  }
}
function getPassword() {
  //var thePassword = 'tsh203';
```

```
  var thePassword =  '***';
   return thePassword;
}
function sendAll() {
  var password = getPassword();
  if(password == null) return;
  var name = document.getElementById('allName').value;
  if(name == null) return;
  var arg1 = document.getElementById('allArg1').value;
  var arg2 = document.getElementById('allArg2').value;
  var arg3 = document.getElementById('allArg3').value;
  var arg4 = document.getElementById('allArg4').value;
  var args = [];
  if(arg1 == "") arg1 = null;
  if(arg2 == "") arg2 = null;
  if(arg3 == "") arg3 = null;
  if(arg4 == "") arg4 = null;
  if(arg1 != null && arg2 == null && arg3 == null && arg4 == null) { args = [arg1]; }
  else if(arg1 != null && arg2 != null && arg3 == null && arg4 == null) { args = [arg1,arg2]; }
  else if(arg1 != null && arg2 != null && arg3 != null && arg4 == null) { args = [arg1,arg2,arg3]; }
  else if(arg1 != null && arg2 != null && arg3 != null && arg4 != null) { args = [arg1,arg2,arg3,arg4]; }
  var m = { password: password, request: 'all', name: name, args: args };
  var n = JSON.stringify(m);
  ws.send(n);
}
function load(path) {
  var password = getPassword();
  if(password == null) return;
  var path = document.getElementById('load').value;
  if(path == null) return;
  if(path == "") return;
  var m = { password: password, request: 'load', path: path };
  var n = JSON.stringify(m);
  ws.send(n);
}
function refresh() {
  var password = getPassword();
  if(password == null) return;
  var m = { password: password, request: 'refresh' };
  var n = JSON.stringify(m);
  ws.send(n);
}
function testOn() {
  var password = getPassword();
  if(password == null) return;
  var m = { password: password, request: 'all', name: 'testOn', args:[] };
  var n = JSON.stringify(m);
  ws.send(n);
}
function testOff() {
  var password = getPassword();
  if(password == null) return;
  var m = { password: password, request: 'all', name: 'testOff', args:[] };
  var n = JSON.stringify(m);
  ws.send(n);
}
```

```
Number.prototype.map = function (in_min, in_max, out_min, out_max) {
  return (this - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

// set some Global Variables.
theImageDirection = -1;
theImageSpeed = 5;
var blueIntervals;
var redIntervals;
var goldIntervals;
var canvasInterval;
  function loadImage(url){
    $("<img/>")
      .on('load', function() { makeCanvas(this); console.log("image loaded correctly"); })
      .on('error', function() { console.log("error loading image"); })
      .attr("src", url)
      .css('display', 'none')
      .appendTo('#theFrame');
  }
  loadImage('images/sedimentary-1080.jpg');
  function makeCanvas(someImage){
      $('canvas').remove(); // kill  old canvasses
            //Make a New Canvas
            theCanvas = $("<canvas/>").appendTo('#theFrame');
      theCanvas[0].width = someImage.width;
      theCanvas[0].height = someImage.height;
            theContext = theCanvas[0].getContext('2d');
            theContext.drawImage(someImage, 0, 0, someImage.width, someImage.height);
    }
    function saturate(){
      loadImage('images/sedimentary-saturated-1080.jpg');
    }
    function redShift(){
      loadImage('images/sedimentary-red-1080.jpg');
    }
    function theBlues(){
      loadImage('images/sedimentary-blue-1080.jpg');
    }
    function goldenAge(){
      loadImage('images/sedimentary-gold-1080.jpg');
    }
    function freeze(){
      isFrozen = true;
    }
    function thaw(){
      isFrozen = false;
    }
    function rockOn(theSpeed){
      if (! theSpeed) { theSpeed = 1; }
      clearInterval(canvasInterval);
      canvasInterval = setInterval(function() {
        $('canvas').css('margin-left', function(n, v) {
          var newMargin = parseFloat(v) + (theImageDirection * theSpeed);
          if (newMargin < (0 - $('canvas').width() + 1920) ) { newMargin = 0 - $('canvas').width() + 1920;
theImageDirection = 1; }
```

```
        if (newMargin > 0) { newMargin = 0; theImageDirection = -1; }
        return  newMargin;
       } );
      }, 60);
    }
    function oneEighty(){
      theImageDirection = (-1 * theImageDirection);
    }
    function rockOff(){
       clearInterval(canvasInterval);
    }
function theColourDrainedFromHerFace(){
  reset();
}
function reset(){
  theBlueZone = 'standard';
  theRedZone = 'standard';
  theGoldZone = 'standard';
  theBlueSpeed = 1;
  theGoldSpeed =1;
  theRedSpeed = 1;
  loadImage('images/sedimentary-1080.jpg');
  kenneth(1);
}
function faster(increment){
  if (!increment) { increment = 1; }
  theBlueSpeed+=increment;
  theGoldSpeed+=increment;
  theRedSpeed+=increment;
}
function slower(increment){
  if (!increment) {increment = 1; }
  theBlueSpeed-=increment;
  theGoldSpeed-=increment;
  theRedSpeed-=increment;
}
function stronger(increment){
  if (increment == null) { increment = 1;   }
  theBluePower = Math.min (10, theBluePower + increment);
  theGoldPower = Math.min(10, theGoldPower + increment);
  theRedPower = Math.min(10, theRedPower + increment);
  clearInterval( blueInterval );
  blueInterval = setInterval(function() {       blueBeats();        }, 1000 / theBluePower  );
  clearInterval( goldInterval );
  goldInterval = setInterval(function() {       goldBeats();        }, 1000 / theGoldPower  );
  clearInterval( redInterval );
  redInterval = setInterval(function() {        redBeats();         }, 1000 / theRedPower  );
}
function weaker(increment){
  if (increment == null) {increment = 1; }
  theBluePower = Math.max (1, theBluePower - increment);
  theGoldPower = Math.max (1, theGoldPower - increment);
  theRedPower = Math.max (1, theRedPower - increment);
  clearInterval( blueInterval );
  blueInterval = setInterval(function() {       blueBeats();        }, 1000 / theBluePower  );
  clearInterval( goldInterval );
```

```
  goldInterval = setInterval(function() {       goldBeats();        }, 1000 / theGoldPower  );
  clearInterval( redInterval );
  redInterval = setInterval(function() {        redBeats();         }, 1000 / theRedPower  );
}

  happyTrailsEnabled = false;
  isFrozen = false;
            // set up randomness for lens wobble
             var rand = Math.random() * 1000+500,
        randother = Math.random() * 1000+500,
        randanother = Math.random() * 1000+500;

  theBlueSpeed = Math.floor(Math.random() * 3) + 1;
  theBluePower = 1;
  theBlueZone = 'standard';
  theBlueDirection = 1;
  blueOffsetX = 0;
  blueOffsetY = 0;
  theRedSpeed = Math.floor(Math.random() * 3) + 1;
  theRedPower = 1;
  theRedZone = 'standard';
  theRedDirection = 1;
  redOffsetX = 0;
  redOffsetY = 0;
  theGoldSpeed = Math.floor(Math.random() * 3) + 1;
  theGoldPower = 1;
  theGoldZone = 'standard';
  theGoldDirection = 1;
  goldOffsetX = 0;
  goldOffsetY = 0;
  function blueSpeed(speed){  theBlueSpeed = speed;  }
  function redSpeed(speed){  theRedSpeed = speed;  }
  function goldSpeed(speed){  theGoldSpeed = speed;  }
  function bluePower(bluePower){
    if (!bluePower){ bluePower = 1; }
    theBluePower = bluePower;
    clearInterval( blueInterval );
    blueInterval = setInterval(function() {     blueBeats();        }, 1000 / bluePower  );
  }
  function redPower(redPower){
    if (!redPower){ redPower = 1; }
    theRedPower = redPower;
    clearInterval( redInterval );
    redInterval = setInterval(function() {      redBeats();         }, 1000 / redPower  );
  }
  function goldPower(power){
    if (!power){ power = 1; }
    theGoldPower = power;
    clearInterval( goldInterval );
    goldInterval = setInterval(function() {     goldBeats();        }, 1000 / power  );
  }
function sedimentaryRock(){
  rockOn(8);
  blueZone();
  redZone();
  goldZone();
```

```
  bluePower(2);
  redPower(8);
  goldPower(4);
  redSpeed(12);
  blueSpeed(12);
  goldSpeed(12);
}
  function breakOut(){
    isFrozen = false;
    blueZone('standard');
    redZone('standard');
    goldZone('standard');
    bluePower(12);
    redPower(6);
    goldPower(0.5);
    redSpeed(4);
    blueSpeed(8);
    goldSpeed(12);
  }
  function silence(){
    freeze();
    rockOn(1);
    redPower(0.0001);
    bluePower(0.0001);
    goldPower(0.0001);
  }
function sound(){
  thaw();
  rockOn(3);
  redPower(1);
  bluePower(1);
  goldPower(1);
}
  function getBackInLine(){
    reset();
    blueZone();
    redZone();
    goldZone();
  }
  function blueZone(theZone){
    if(!theZone){ theZone = 'middle'; }
    theBlueZone = theZone;
  }
  function redZone(theZone){
    if(!theZone){ theZone = 'lower'; }
    theRedZone = theZone;
  }
  function goldZone(theZone){
    if(!theZone){ theZone = 'upper'; }
    theGoldZone = theZone;
  }
  function blueMove(){
    if (typeof(theCanvas) == "undefined" ) return;
    if (!isFrozen) {
      if (theBlueZone == 'middle' ){ $('#theBluePill').css('top', Math.floor( Math.sin(          (new Date().getTime() +
rand ) / ( 5000 / theBlueSpeed) ) * 50 + 50 ) + 400 + 'px' ); }
```

```
    else{ $('#theBluePill').css('top', Math.floor( Math.sin(      (new Date().getTime() + rand ) / ( 5000 /
theBlueSpeed) ) * 440 + 440 ) +'px' ); }
    $('#theBluePill').css('left', function(n, v) {
     var xOffset = parseFloat(v) + (theBlueDirection * theBlueSpeed);
     if (xOffset < 0) { xOffset = 0; theBlueDirection = 1; }
     if (xOffset > 1800) { xOffset = 1800; theBlueDirection = -1; }
     return  xOffset;
    } );
   }
   $('#theBluePill').css('transform', 'rotate(' + ( Math.sin(       new Date().getTime() / randother    ) * 5 )  + 'deg)');
   blueOffsetX = ( parseFloat(theCanvas.css('margin-left')) * -1 )  +  parseFloat($('#theBluePill').css('left')) + 10 ,
   blueOffsetY =  parseFloat($('#theBluePill').css('top') ) + 50;
  }
  setInterval(function() { blueMove(); }, 60 );
  function redMove(){
   if (typeof(theCanvas) == "undefined" ) return;
   if (!isFrozen) {
    if (theRedZone =='lower'){ $('#theRedPill').css('top', Math.floor( Math.sin(   ( new Date().getTime() -
randother) / (5000 / theRedSpeed )  ) * 50 + 50 ) + 700 + 'px' ); }
    else { $('#theRedPill').css('top', Math.floor( Math.sin(      ( new Date().getTime() - randother) / (5000 /
theRedSpeed )  ) * 440 + 440 ) +'px' ); }
                $('#theRedPill').css('left', function(n, v) {
                        var xOffset = parseFloat(v) + (theRedDirection * theRedSpeed );
                        if (xOffset < 0) { xOffset = 0; theRedDirection = 1; }
                        if (xOffset > 1800) { xOffset = 1800; theRedDirection = -1; }
                        return  xOffset;
                } );
   }
   $('#theRedPill').css('transform', 'rotate(' + ( Math.sin(       new Date().getTime() / randother    ) * 5 )  + 'deg)');
   redOffsetX = ( parseFloat(theCanvas.css('margin-left')) * -1 )  +  parseFloat($('#theRedPill').css('left')) + 100 ;
   redOffsetY =  parseFloat($('#theRedPill').css('top') ) + 50;
  }
   setInterval(function() { redMove(); }, 60 );
   function goldMove(){
    if (typeof(theCanvas) == "undefined" ) return;
    if (!isFrozen) {
     if (theGoldZone =='upper'){ topPosition = Math.floor( Math.sin(   ( new Date().getTime() - randanother) /
(5000 / theGoldSpeed )  ) * 50 + 150 );  }
     else { topPosition = Math.floor( Math.sin(      ( new Date().getTime() - randanother) / (5000 / theGoldSpeed
)  ) * 440 + 440 ) ; }
      $('#theGoldPill').css('top', topPosition + 'px' );
      $('#theGoldPill').css('left', function(n, v) {
       var xOffset = parseFloat(v) + (theGoldDirection * theGoldSpeed );
       if (xOffset < 0) { xOffset = 0; theGoldDirection = 1; }
       if (xOffset > 1800) { xOffset = 1800; theGoldDirection = -1; }
       return  xOffset;
      } );
     }
    $('#theGoldPill').css('transform', 'rotate(' + ( Math.sin(      new Date().getTime() / randanother          ) * 5 )  +
'deg)');
    goldOffsetX = ( parseFloat(theCanvas.css('margin-left')) * -1 )  +  parseFloat($('#theGoldPill').css('left')) + 100
;
    goldOffsetY =  parseFloat($('#theGoldPill').css('top') ) + 50;
   }
   setInterval(function() { goldMove(); }, 60 );
```

```
  function redBeats(){

    var redPixelData = theContext.getImageData(parseFloat(redOffsetX), parseFloat(redOffsetY), 50, 50).data,
      redRGB = {r:0,g:0,b:0},
      redCount = 0;
    for (var i=0;i<redPixelData.length;i+=4) {  redRGB.r += redPixelData[i];  redRGB.g += redPixelData[i+1];
redRGB.b += redPixelData[i+2];  redCount ++;  }
    redRGB.r = ~~(redRGB.r/redCount);                  redRGB.g = ~~(redRGB.g/redCount);
        redRGB.b = ~~(redRGB.b/redCount); // floor values
    var theHSL = rgbToHsl(redRGB.r, redRGB.g, redRGB.b);
    var theHue = theHSL[0].map(0,1,0,360);
    if ( (theHue > 300 || theHue < 20) && typeof(ws) != "undefined"){
     $('#theRedPill').css({'background-color' : 'rgba(190,58,58,1)' ,'border':'0 solid white', 'borderWidth':5 }
).animate({'background-color':'rgba(170,38,38,0.5)', 'borderWidth':0}, 50);
      var thePass = getPassword();
      var m = { password: thePass, request: 'all', name: 'redSound', args: [1,1,theHSL[2].map(0,1,0.2,3.8)] };
                  ws.send(JSON.stringify(m));
      if (happyTrailsEnabled && !isFrozen ){
       var theTrailTop = parseFloat($('#theRedPill').css('top')) + 50 + 'px';
       var theTrailLeft = parseFloat($('#theRedPill').css('left')) + 100 + 'px';
       var theTrailWidth = theHSL[2].map(0,1,25,50);
       var theTrailHeight = theTrailWidth / 2;
       $('<div/>', { id: 'redBeat_' + Math.floor(Math.random()*1000) }).appendTo('#theFrame').css({
          'width' : theTrailWidth + 'px', 'height' :  theTrailHeight + 'px',
          'position' : 'absolute', 'top': theTrailTop, 'left' : theTrailLeft,
          'background-color': 'rgba(170,38,38,0.9)',
          'border-radius' : '100px'
        }).fadeOut(10000, function(){ $(this).remove();  });
      }
      // draw colour trace rectangle
      //theContext.fillStyle = "rgba(" +redRGB.r + ",0,0,"+ redRGB.r.map(0,255,0.5,1)+")";
      //theContext.fillRect( redOffsetX, redOffsetY,  redRGB.r.map(0,255,20,60) ,   redRGB.r.map(0,255,20,60));
    }
  }
  function blueBeats(){
    var bluePixelData = theContext.getImageData(parseFloat(blueOffsetX), parseFloat(blueOffsetY), 50,
50).data,
            blueRGB = {r:0,g:0,b:0},
                  blueCount = 0;
                for (var i=0;i<bluePixelData.length;i+=4) {  blueRGB.r += bluePixelData[i];  blueRGB.g +=
bluePixelData[i+1];  blueRGB.b += bluePixelData[i+2];  blueCount ++;  }
                  blueRGB.r = ~~(blueRGB.r/blueCount);                  blueRGB.g = ~~(blueRGB.g/blueCount);
        blueRGB.b = ~~(blueRGB.b/blueCount); // floor values
    var theHSL = rgbToHsl(blueRGB.r, blueRGB.g, blueRGB.b);
    var theHue = theHSL[0].map(0,1,0,360);
    if ((theHue > 170 && theHue < 240 ) && typeof(ws) != "undefined"){
     var theBrightness =  (blueRGB.r+ blueRGB.g+blueRGB.b) / 3;
     $('#theBluePill').css({'background-color' : 'rgba(58,58,190,1)','border':'0 solid white', 'borderWidth':5}
).animate({'background-color':'rgba(38,38,170,0.5)', 'borderWidth':0}, 50);
      //var m = { password: "123", request: 'all', name: 'generativeSaw', args: [blueishness.map(0,30, 440,880),
blueishness.map(0,30, 0,0.3)] };
      var thePass = getPassword();
      var m = { password: thePass, request: 'all', name: 'blueSound', args: [1,8,theBrightness.map(0,255,0.5,1.8)] };
      ws.send(JSON.stringify(m));
      //theContext.fillStyle = "rgba(0,0," +blueRGB.b +","+blueRGB.b.map(0,255, 0.5,1)+")";
                  //theContext.fillRect( blueOffsetX, blueOffsetY, blueRGB.b.map(0,255, 20,60) ,
```

```
blueRGB.b.map(0,255, 20,60));
      if (happyTrailsEnabled && !isFrozen ){
        var theTrailTop = parseFloat($('#theBluePill').css('top')) + 50 + 'px';
        var theTrailLeft = parseFloat($('#theBluePill').css('left')) + 100 + 'px';
        var theTrailWidth = theHSL[2].map(0,1,25,50);
        var theTrailHeight = theTrailWidth / 2;
        $('<div/>', { id: 'blueBeat_' + Math.floor(Math.random()*1000) }).appendTo('#theFrame').css({
            'width' : theTrailWidth + 'px', 'height' :  theTrailHeight + 'px',
            'position' : 'absolute', 'top': theTrailTop, 'left' : theTrailLeft,
            'background-color': 'rgba(38,38,170,0.9)',
            'border-radius' : '100px'
          }).fadeOut(10000, function(){ $(this).remove();  });
       }
      }
    }
     function goldBeats(){
     var goldPixelData = theContext.getImageData(parseFloat(goldOffsetX), parseFloat(goldOffsetY), 50,
50).data,
         goldRGB = {r:0,g:0,b:0},
         goldCount = 0;
     for (var i=0;i<goldPixelData.length;i+=4) {  goldRGB.r += goldPixelData[i];  goldRGB.g +=
goldPixelData[i+1];  goldRGB.b += goldPixelData[i+2];  goldCount ++;  }
     goldRGB.r = ~~(goldRGB.r/goldCount);              goldRGB.g = ~~(goldRGB.g/goldCount);
          goldRGB.b = ~~(goldRGB.b/goldCount); // floor values
     var theHSL = rgbToHsl(goldRGB.r, goldRGB.g, goldRGB.b);
     var theHue = theHSL[0].map(0,1,0,360);
     if ((theHue > 34 && theHue < 80) && typeof(ws) != "undefined"){
        $('#theGoldPill').css({'background-color':'rgba(220,240,58,1)', 'border':'0 solid white', 'borderWidth':5})
          .animate({'background-color':'rgba(170,240,38,0.5)', 'borderWidth':0}, 50);
        //var m = { password: "123", request: 'all', name: 'simpleSaw', args: [goldOffsetY.map(0,800, 466.164,
261.626),theGold.map(0,1,0.01,0.04)] };
        var thePass = getPassword();
        var m = { password: thePass, request: 'all', name: 'goldSound', args: [1,2,theHSL[2].map(0,1,0.2,6.8)] };

        ws.send(JSON.stringify(m));
        if (happyTrailsEnabled && !isFrozen ){
          var theTrailTop = parseFloat($('#theGoldPill').css('top')) + 50 + 'px';
          var theTrailLeft = parseFloat($('#theGoldPill').css('left')) + 100 + 'px';
          var theTrailWidth = theHSL[2].map(0,1,25,50);
          var theTrailHeight = theTrailWidth / 2;
          $('<div/>', { id: 'goldBeat_' + Math.floor(Math.random()*1000) }).appendTo('#theFrame').css({
              'width' : theTrailWidth + 'px', 'height' :  theTrailHeight + 'px',
              'position' : 'absolute', 'top': theTrailTop, 'left' : theTrailLeft,
              'background-color': 'rgba(170,240,38,0.9)',
              'border-radius' : '100px'
            }).fadeOut(10000, function(){ $(this).remove();  });
        }
      }
    }
redInterval = setInterval(function() {        redBeats(); }, 1000);
blueInterval = setInterval(function() {       blueBeats();       }, 1000);
goldInterval = setInterval(function() {       goldBeats();       }, 1000);
function callMyBluff(){
  silence();
  freeze();
  setTimeout( function(){ collide(); }, 2000);
```

```
}
function collide(){
  var thePass = getPassword();
  var m = { password: thePass, request: 'all', name: 'collideSound', args: [2,2,Math.random().map(0,1,0.5,0.9)] };
  ws.send(JSON.stringify(m));
  console.log('collision');
}
function happyTrails(){
  happyTrailsEnabled = true;
}
function sadTrails(){
  happyTrailsEnabled = false;
}
function kenneth( pulsesPerSecond ){
  var theMilliseconds = 1000/pulsesPerSecond;
  clearInterval( redInterval );
  clearInterval( blueInterval );
  clearInterval( goldInterval );
  redInterval = setInterval(function() {       redBeats(); }, theMilliseconds );
  blueInterval = setInterval(function() {      blueBeats();       }, theMilliseconds );
  goldInterval = setInterval(function() {      goldBeats();       }, theMilliseconds );
}
/**
 * Converts an RGB color value to HSL.
 * Assumes r, g, and b are contained in the set [0, 255] and
 * returns h, s, and l in the set [0, 1].
 */
function rgbToHsl(r, g, b){
    r /= 255, g /= 255, b /= 255;
    var max = Math.max(r, g, b), min = Math.min(r, g, b);
    var h, s, l = (max + min) / 2;
    if(max == min){
        h = s = 0; // achromatic
    }else{
        var d = max - min;
        s = l > 0.5 ? d / (2 - max - min) : d / (max + min);
        switch(max){
            case r: h = (g - b) / d + (g < b ? 6 : 0); break;
            case g: h = (b - r) / d + 2; break;
            case b: h = (r - g) / d + 4; break;
        }
        h /= 6;
    }
    return [h, s, l];
}
$('#theCode').keydown(function (e) {
  if (e.ctrlKey && e.keyCode == 13) {
    if ( e.target.value ) { eval(e.target.value); }
    $("<div></div>").html( e.target.value).appendTo('#doneCode').delay(20000).fadeOut();
    e.target.value = '';
    //$('#theCode').css("background-color", "#FFFF9C").animate({ backgroundColor: "#FFFFFF"}, 1500);
  }
  if (e.ctrlKey && e.keyCode == 38) {
    // reveal previous line of code
    $('#theCode').val( $('#doneCode').children().last().html() );
  }
```

```
});
</script>
</body>
</html>
```

ReptileChoir Mobile JavaScript

```
theAmp = 1;
function killBeeps(){  theAmp = .1; }
var audioReady = false;
var theSamples = {};
var theVoices = new Array(16);
var compressor;
var gain;
var choirOn = true; // not actually using this yet but an off switch would be nice
// local places to hold variables current top margin as per display interface/apert
var turtleName = 'theTurtle';
var theColor;
var theVowel = 'u';
var theMotion;
var theTop = 0;
var theLightness = 0.5;
Number.prototype.map = function (in_min, in_max, out_min, out_max) {
  return (this - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
function wsIsUp(){
  if (typeof(ws) == 'object'){
    if( ws.readyState){ return true; }
  }
  return false;
}
/* I use a hidden soundcloud widget as a hack for iOS devices to make webAudio work. */
function soundCloudSetup(){
  if (typeof (SC) == 'undefined') { setTimeout(soundCloudSetup, 100); return; }
  var widgetIframe = document.getElementById('soundCloud');
  var widget       = SC.Widget(widgetIframe);
  widget.bind(SC.Widget.Events.READY, function() {
    $('#scContainer').css({'transform': 'scale(2)'});
     widget.bind(SC.Widget.Events.PLAY, function() {
       $('#soundCloud').css({'marginTop': '200px'});
       $('#scContainer').css({'backgroundPosition': '50% 50%'});
     });
     widget.bind(SC.Widget.Events.FINISH, function() {
        $('#scContainer').remove();
        audioReady =true;
      });
  });
}
```

```
function selectButton(element){
  var theClass = $(element).attr('class');
  if ( wsIsUp() ) { apertMemorySet( theClass, $(element).html() ); console.log('set ' + theClass + ' to ' +
$(element).html() ); }
  if (theClass == 'turtleVowel') { theVowel = $(element).html(); }
  else{ console.log('ws is not up'); }
  $('.'+theClass).css({'borderWidth' : '2px', 'margin' : '10px'});
  $(element).css({'borderWidth' : '8px', 'margin' : '4px'});
}
/* Names for turtles */
function randomName(){
  var items = [
    "Bailey", "Bella ", "Max ", "Lucy ", "Charlie ", "Molly ", "Buddy ", "Daisy ", "Rocky",
    "Maggie ", "Jake", "Sophie ", "Jack", "Sadie", "Toby", "Chloe", "Cody", "Bailey", "Buster",
    "Lola", "Duke", "Zoe", "Cooper", "Abby", "Riley", "Ginger", "Harley", "Roxy", "Bear",
    "Gracie", "Tucker", "Coco", "Murphy", "Sasha", "Lucky", "Lily", "Oliver", "Angel", "Sam",
    "Princess", "Oscar", "Emma", "Teddy", "Annie", "Winston", "Rosie", "Sammy", "Ruby"
  ];
  var item = items[Math.floor(Math.random()*items.length)];
  return item;
}
function interfaceSetup(){
  if (!audioReady) {    setTimeout(interfaceSetup, 100); return; }
  apertStartAudio();
  //initialize?
    gain = db(12); // a global gain value, notated in dB
    // added a global compressor
    compressor = ac.createDynamicsCompressor();
    compressor.threshold.value = -3;
    compressor.ratio.value = 10;
    compressor.attack = 0.005;
    compressor.release = 0.005;
    compressor.connect(ac.destination);
 theSamples = {
    a : new Array(12),
    e : new Array(12),
    i : new Array(12),
    o : new Array(12),
    u : new Array(12)
  }
  theVowels = ['a', 'e', 'i', 'o', 'u'];
  theNotes = ['C', 'Db', 'D', 'Eb', 'E', 'F', 'Gb', 'G', 'Ab', 'A', 'Bb', 'B'];
  for(i in theVowels){
    for (j in theNotes){
      if (typeof(theSamples[theVowels[i]][j]) == 'undefined'){
        theSamples[theVowels[i]][j] = new SampleBuffer(theVowels[i]+'-'+theNotes[j] +'.wav');
      }
    }
  }
```

```
// initialization
for(var n=0;n<16;n++){ theVoices[n] = new Voice(); }
turtleName = randomName();
$('body').append(
 $('<div id="myNameIs" />').css({  'width': '100%' , 'height': '100px'}). append(
  $('<img src="hurtle-right-small.gif" />').css({ 'float' : 'left' }),
  $('<div>My name is </div>').css({
   'font-size' : '24px', 'color' : '#aaa', 'float' : 'left', 'margin' :'15px',
   'font-family' : '"Oxygen", sans-serif'  }),
  $('<input id="theTurtleName" value="'+turtleName+'">').css({
   'padding' : '5px', 'border' : '2px solid #ccc', 'width' : '400px', 'float' : 'left','margin' :'5px',
   'font-size' : '30px', 'color' : '#333', 'font-family' : '"Oxygen", sans-serif'
  }),
  $('<div id="muteButton"></div>').css({
   'background' : 'url(mute-button.png) no-repeat -50px 0px', 'float' : 'left',
   'width' :'50px', 'height':'50px'}).
   click(function(){
      if (choirOn == true){
         choirOn = false;
         $(this).css({'backgroundPosition': '0px 0px'});
      }
      else{
        choirOn = true;
        $(this).css({'backgroundPosition': '-50px 0px'});
      }
   })
 ),
 $('<div id="swimContainer" />').css({
   'width': '100%', 'height' : '80px', 'background' : '#eee' }). append(
  $('<div>I traverse land and sea </div>').css({
   'font-size' : '24px', 'color' : '#aaa', 'float' : 'left', 'margin' :'15px',
   'font-family' : '"Oxygen", sans-serif'  }),
  $('<input type="hidden"  id="theMotion" >'),
  $(' <button type="button" class="turtleMotion">up</button> ').css({
     'background': '#fff', 'border':'2px solid #ccc', 'float' : 'left', 'margin': '10px',
     'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
     click(function(){
        selectButton(this);
     }),
  $(' <button type="button" class="turtleMotion">down</button> ').css({
     'background': '#fff', 'border':'2px solid #ccc', 'float' : 'left', 'margin': '10px' ,
     'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
     click(function(){
        selectButton(this);
     }),
  $(' <button type="button" class="turtleMotion"  >left</button> ').css({
     'background': '#fff', 'border':'2px solid #ccc', 'float' : 'left', 'margin': '10px' ,
     'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
     click(function(){
```

```
        selectButton(this);
      }),
    $(' <button type="button" class="turtleMotion">right</button> ').css({
      'background': '#fff', 'border':'2px solid #ccc', 'float' : 'left', 'margin': '10px' ,
      'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
      click(function(){
        selectButton(this);
      }),
    $(' <button type="button" class="turtleMotion">hover</button> ').css({
      'background': '#fff', 'border':'2px solid #ccc', 'float' : 'left', 'margin': '10px' ,
      'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
      click(function(){
        selectButton(this);
      })
  ),
  $('<div id="colorsContainer" />').css({
     'width': '100%', 'height' : '80px', 'background' : '#eee' }). append(
    $('<div>I like </div>').css({
     'font-size' : '24px', 'color' : '#aaa', 'float' : 'left', 'margin' :'15px',
     'font-family' : '"Oxygen", sans-serif'  }),
    $(' <button type="button" class="turtleColor">red</button> ').css({
      'background': '#fff', 'border':'2px solid red', 'float' : 'left', 'margin': '10px',
      'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
      click(function(){
        selectButton(this);
      }),
    $(' <button type="button" class="turtleColor">orange</button> ').css({
      'background': '#fff', 'border':'2px solid orange', 'float' : 'left', 'margin': '10px' ,
      'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
      click(function(){
        selectButton(this);
      }),
    $(' <button type="button" class="turtleColor">yellow</button> ').css({
      'background': '#fff', 'border':'2px solid yellow', 'float' : 'left', 'margin': '10px' ,
      'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
      click(function(){
        selectButton(this);
      }),
    $(' <button type="button" class="turtleColor">green</button> ').css({
      'background': '#fff', 'border':'2px solid green', 'float' : 'left', 'margin': '10px' ,
      'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
      click(function(){
        selectButton(this);
      }),
    $(' <button type="button" class="turtleColor">blue</button> ').css({
      'background': '#fff', 'border':'2px solid blue', 'float' : 'left', 'margin': '10px' ,
      'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
      click(function(){
        selectButton(this);
```

```
    }),
  $(' <button type="button" class="turtleColor">indigo</button> ').css({
    'background': '#fff', 'border':'2px solid indigo', 'float' : 'left', 'margin': '10px' ,
    'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
    click(function(){
      selectButton(this);
    }),
  $(' <button type="button" class="turtleColor">violet</button> ').css({
    'background': '#fff', 'border':'2px solid violet', 'float' : 'left', 'margin': '10px' ,
    'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
    click(function(){
      selectButton(this);
    })
),
$('<div id="vowelsContainer" />').css({
   'width': '100%', 'height' : '80px', 'background' : '#eee' }). append(
  $('<div>My favourite vowel is </div>').css({
   'font-size' : '24px', 'color' : '#aaa', 'float' : 'left', 'margin' :'15px',
   'font-family' : '"Oxygen", sans-serif'  }),
  $(' <button type="button" class="turtleVowel">a</button> ').css({
    'background': '#fff', 'border':'2px solid #ccc', 'float' : 'left', 'margin': '10px',
    'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
    click(function(){
      selectButton(this);
    }),
  $(' <button type="button" class="turtleVowel">e</button> ').css({
    'background': '#fff', 'border':'2px solid #ccc', 'float' : 'left', 'margin': '10px' ,
    'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
    click(function(){
      selectButton(this);
    }),
  $(' <button type="button" class="turtleVowel">i</button> ').css({
    'background': '#fff', 'border':'2px solid #ccc', 'float' : 'left', 'margin': '10px' ,
    'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
    click(function(){
      selectButton(this);
    }),
  $(' <button type="button" class="turtleVowel">o</button> ').css({
    'background': '#fff', 'border':'2px solid #ccc', 'float' : 'left', 'margin': '10px' ,
    'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
    click(function(){
      selectButton(this);
    }),
  $(' <button type="button" class="turtleVowel">u</button> ').css({
    'background': '#fff', 'border':'2px solid #ccc', 'float' : 'left', 'margin': '10px' ,
    'font-size' : '24px', 'color' : '#333', 'padding':'5px 15px' }).
    click(function(){
      selectButton(this);
    })
```

```
    ),

    $('<div/>', { id: 'theRedPill' }).css({
      'width' : '100px', 'height' : '100px',
      'background-color': 'rgba(170,38,38,0.5)',
      'border-radius' : '100px'
    }).click(function(){
        simple(440,theAmp);
    })
  ).css({'background': '#ddd'});
}
$( document ).ready(function() {
$('body').css({'backgroundColor' : '#f5f5f5'});
    $('<div>', {id: 'scContainer'} ).appendTo('body').
      css({margin: '100px auto', width:'50px', height:'50px', background: 'url(ajax-loader.gif) no-repeat 20px 50%',
overflow: 'hidden' });
    $('<iframe>', {
       src:
'https://w.soundcloud.com/player/?url=https://soundcloud.com/nsitu/silence&auto_play=false&color=2266&&buyin
g=false&liking=false&download=false&sharing=false&show_artwork=false&show_comments=false&show_playc
ount=false&show_user=false&hide_related=false&visual=false',
       id:  'soundCloud',
       frameborder: 'no',
       scrolling: 'no'
     }).appendTo('#scContainer').css({'marginTop':'-5px', 'marginLeft':'-5px'});
      var scapi = document.createElement("script");
      scapi.type = "text/javascript";
      scapi.src = "https://w.soundcloud.com/player/api.js";
      $("head").append(scapi);
      soundCloudSetup();
    interfaceSetup();

  });

var rand = Math.random() * 1+5,
    randother = Math.random() * 1+5,
    randanother = Math.random() * 1+5,
    bankSize = 2;
SampleBuffer = function(url) {
  var request = new XMLHttpRequest();
  request.open('GET',url,true);
  request.responseType = 'arraybuffer';
  var closure = this; // a closure is necessary for...
  request.onload = function() {
    ac.decodeAudioData(request.response, function(x) {
      console.log("sample " + url + "loaded");
      closure.buffer = x;
    });
  };
```

```
  request.send();
}
db = function(x) { return Math.pow(10,x/20); } // a function to calculate amplitude given dB
Voice = function() {
  this.gain = ac.createGain();
  this.gain.connect(compressor);
  this.gain.gain.setValueAtTime(0,ac.currentTime);
  this.playing = false;
}
/* theVowel is aeiou, theNote is a number between 0 and 11 */
Voice.prototype.play = function(aVowel, aNote, aRate) {
  if (!choirOn) return;
  //defaults
  if(aVowel == null) aVowel = 'a';
  if(aNote == null) aNote = 1;
  if(aRate == null) aRate = 1;
  this.source = ac.createBufferSource();
  if(theSamples[aVowel][aNote].buffer == null) return;
  this.source.buffer = theSamples[aVowel][aNote].buffer;
  this.source.playbackRate.value = aRate;
  this.source.connect(this.gain);
  this.source.start();
  var now = ac.currentTime;
  this.gain.gain.setValueAtTime(0,now);
  this.gain.gain.linearRampToValueAtTime(db(gain),now+0.5);
  this.gain.gain.linearRampToValueAtTime(0,now+1.0);
  this.playing = true;
  var closure = this;
  setTimeout(function() {
    closure.playing = false;
    closure.source.disconnect(closure.gain);
  },1500);
}
function turtleSong() {
  var n;
  for(n=0;n<16;n++) {
    if(theVoices[n].playing == false)break;
  }
  if(n==16) {
    console.log("no voices are available right now");
    return;
  }
  /* to get the pitch, map a range of heights */
  if (theTop != null){  theNote = Math.floor(theTop.map(780, 0, 0, 11));  }
  else{   theNote = Math.floor(Math.random()*11);  }
  var derivedRate;
  console.log(theLightness);
  /*derive an octave from the current lightness*/
  if (theLightness != null){
```

```
    derivedRate = 1;
    if (theLightness > 0.66){ derivedRate = 2; }
    if (theLightness < 0.33){ derivedRate = 0.5; }
  }
  theVoices[n].play(theVowel, theNote, derivedRate);
}
function apertReceivedGet( msg ){
  if(msg.key =='top' && typeof( msg.value) != 'undefined'){  theTop = msg.value; }
  if(msg.key =='lightness' && typeof( msg.value) != 'undefined'){  theLightness = msg.value; }
}
function apertInitialize() {
// the purpose of this is for synths to be created after the audio context exists
  // ask apert where we are
  setInterval(function() {
    if ( wsIsUp() ) {
      apertMemorySet('theTurtleName', $('#theTurtleName').val() );
      apertMemoryGet('top');
      apertMemoryGet('lightness');
    }
  },137);
  setInterval(function() {
    turtleSong();
  },250);
}
function apertMemoryGet(key) {
  if (wsIsUp()){
    /* doesn't this message go to everyone? */
    var m = { request: 'get', key: key};
    var s = JSON.stringify(m);
    ws.send(s);
  }
}
function simple(freq,amp) {
  apertStartAudio();
        var sine = ac.createOscillator();
        sine.type = 'sine';
        sine.frequency.value = freq;
        var gain = ac.createGain();
        sine.connect(gain);
        gain.connect(ac.destination);
        sine.start();
        // envelope
        var now = ac.currentTime;
        gain.gain.setValueAtTime(0,now);
  gain.gain.linearRampToValueAtTime(amp,now+0.005); gain.gain.linearRampToValueAtTime(0,now+0.405);
        // schedule cleanup
        setTimeout(function() {
                sine.stop();
                sine.disconnect(gain);
```

```
                gain.disconnect(ac.destination);
        },1000);
};
```

ReptileChoir Display Interface

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Reptile Choir</title>
  <meta name="author" description="Harold Sikkema">
  <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Oxygen:400,300,700">
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
<textarea id="theCode"> </textarea>
<div id="doneCode"></div>
<div id="theFrame">
</div>
<script type="text/javascript" src="jquery-1.12.1.min.js"></script>
<script type="text/javascript" src="jquery.color.js"></script>

<script>
var ws;
var apertMemorySnapshot;
var apertMemoryOld;
function wsIsUp(){
  if (typeof(ws) == 'object'){
    if( ws.readyState){
      return true;
    }
  }
  return false;
}
function setup(theServer) {
  window.WebSocket = window.WebSocket || window.MozWebSocket;

if (theServer == null){
theServer = 'nsitu.ca';
}
  var url = 'ws://' + theServer + ':8000';
  console.log("attempting websocket connection to " + url);
  ws = new WebSocket(url);
  ws.onopen = function () { console.log("websocket connection opened"); };
  ws.onerror = function () { console.log("ERROR opening websocket connection"); };
  ws.onmessage = function (m) {
    var data = JSON.parse(m.data);
    if(data.type =='all') { console.log("all " + data.name); }
    else if(data.type =='refreshCount') {
      /*document.getElementById('refreshCount').textContent = data.count;*/
    }
    else if(data.type =='clientCount') {
      /*document.getElementById('clientCount').textContent = data.count; */
```

```
    }
    else if(data.type == 'dump') {
      apertMemoryOld = apertMemorySnapshot;
      apertMemorySnapshot = data.result;
    }
    else {
      console.log("received WebSocket message of unknown type = " + data.type);
    }
  }
}
function getPassword() {
  var thePassword =  '***';
  return thePassword;
}
function sendAll() {
  var password = getPassword();
  if(password == null) return;
  var name = document.getElementById('allName').value;
  if(name == null) return;
  var arg1 = document.getElementById('allArg1').value;
  var arg2 = document.getElementById('allArg2').value;
  var arg3 = document.getElementById('allArg3').value;
  var arg4 = document.getElementById('allArg4').value;
  var args = [];
  if(arg1 == "") arg1 = null;
  if(arg2 == "") arg2 = null;
  if(arg3 == "") arg3 = null;
  if(arg4 == "") arg4 = null;
  if(arg1 != null && arg2 == null && arg3 == null && arg4 == null) { args = [arg1]; }
  else if(arg1 != null && arg2 != null && arg3 == null && arg4 == null) { args = [arg1,arg2]; }
  else if(arg1 != null && arg2 != null && arg3 != null && arg4 == null) { args = [arg1,arg2,arg3]; }
  else if(arg1 != null && arg2 != null && arg3 != null && arg4 != null) { args = [arg1,arg2,arg3,arg4]; }
  var m = { password: password, request: 'all', name: name, args: args };
  var n = JSON.stringify(m);
  ws.send(n);
}
function load(path) {
  var password = getPassword();
  if(password == null) return;
  var path = document.getElementById('load').value;
  if(path == null) return;
  if(path == "") return;
  var m = { password: password, request: 'load', path: path };
  var n = JSON.stringify(m);
  ws.send(n);
}
function refresh() {
  var password = getPassword();
  if(password == null) return;
  var m = { password: password, request: 'refresh' };
  var n = JSON.stringify(m);
  ws.send(n);
}
function testOn() {
  var password = getPassword();
  if(password == null) return;
```

```
  var m = { password: password, request: 'all', name: 'testOn', args:[] };
  var n = JSON.stringify(m);
  ws.send(n);
}
function testOff() {
  var password = getPassword();
  if(password == null) return;
  var m = { password: password, request: 'all', name: 'testOff', args:[] };
  var n = JSON.stringify(m);
  ws.send(n);
}
Number.prototype.map = function (in_min, in_max, out_min, out_max) {
  return (this - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
// set some Global Variables.
theImageDirection = -1;
theImageSpeed = 5;
var canvasInterval;
            // set up randomness for lens wobble
             var rand = Math.random() * 1000+500,
       randother = Math.random() * 1000+500,
       randanother = Math.random() * 1000+500;
  function loadImage(url){
    $("<img/>")
      .on('load', function() { makeCanvas(this); console.log("image loaded correctly"); })
      .on('error', function() { console.log("error loading image"); })
      .attr("src", url)
      .css('display', 'none')
      .appendTo('#theFrame');
  }
  loadImage('images/carpe-diem-continuous.jpg');
  function makeCanvas(someImage){
     $('canvas').remove(); // kill  old canvasses
            //Make a New Canvas
            theCanvas = $("<canvas/>").appendTo('#theFrame');
     theCanvas[0].width = someImage.width;
     theCanvas[0].height = someImage.height;
            theContext = theCanvas[0].getContext('2d');
            theContext.drawImage(someImage, 0, 0, someImage.width, someImage.height);
    }

    function rockOn(){
      clearInterval(canvasInterval);
      canvasInterval = setInterval(function() {
        $('canvas').css('margin-left', function(n, v) {
          var newMargin = parseFloat(v) + (theImageDirection * theImageSpeed);
          if (newMargin < (0 - $('canvas').width() + 1920) ) { newMargin = 0; }
          if (newMargin > 0) { newMargin = 0 - $('canvas').width() + 1920; }
          /*if (newMargin < (0 - $('canvas').width() + 1920) ) { newMargin = 0 - $('canvas').width() + 1920;
theImageDirection = 1; }
          if (newMargin > 0) { newMargin = 0; theImageDirection = -1; }*/
          return  newMargin;
        } );
      }, 60);
    }
    function oneEighty(){
```

```
    theImageDirection = (-1 * theImageDirection);
  }

  /* this needs to set some memory values like top margin. */
  function turtleBeat(el){
   /* calculate top */
   apertMemorySetFor(el,'top', parseFloat($('#'+el).css('top') ) );
   theOffsetX = ( parseFloat(theCanvas.css('margin-left')) * -1 )  +  parseFloat($('#'+el).css('left')) + 100 ;
   theOffsetY =  parseFloat($('#'+el).css('top') ) + 50;
   var thePixelData = theContext.getImageData(parseFloat(theOffsetX), parseFloat(theOffsetY), 25, 25).data,
      theRGB = {r:0,g:0,b:0},
      theCount = 0;
   for (var i=0;i<thePixelData.length;i+=4) {
      theRGB.r += thePixelData[i];
      theRGB.g += thePixelData[i+1];
      theRGB.b += thePixelData[i+2];
      theCount ++;
   }
   theRGB.r = ~~(theRGB.r/theCount);
   theRGB.g = ~~(theRGB.g/theCount);
   theRGB.b = ~~(theRGB.b/theCount); // floor values
   var theHSL = rgbToHsl(theRGB.r, theRGB.g, theRGB.b);
   apertMemorySetFor(el,'lightness', theHSL[2] );
  }

/**
 * Converts an RGB color value to HSL.
 * Assumes r, g, and b are contained in the set [0, 255] and
 * returns h, s, and l in the set [0, 1].
 */
function rgbToHsl(r, g, b){
  r /= 255, g /= 255, b /= 255;
  var max = Math.max(r, g, b), min = Math.min(r, g, b);
  var h, s, l = (max + min) / 2;
  if(max == min){
     h = s = 0; // achromatic
  }else{
     var d = max - min;
     s = l > 0.5 ? d / (2 - max - min) : d / (max + min);
     switch(max){
        case r: h = (g - b) / d + (g < b ? 6 : 0); break;
        case g: h = (b - r) / d + 2; break;
        case b: h = (r - g) / d + 4; break;
     }
     h /= 6;
  }
  return [h, s, l];
}
$('#theCode').keydown(function (e) {
 if (e.ctrlKey && e.keyCode == 13) {
  if ( e.target.value ) { eval(e.target.value); }
  $("<div></div>").html( e.target.value).appendTo('#doneCode').delay(20000).fadeOut();
  e.target.value = '';
  //$('#theCode').css("background-color", "#FFFF9C").animate({ backgroundColor: "#FFFFFF"}, 1500);
 }
 if (e.ctrlKey && e.keyCode == 38) {
```

```
    // reveal previous line of code
    $('#theCode').val( $('#doneCode').children().last().html() );
   }
});
function turtleUpdate(theId, t){
  if (typeof(t.theTurtleName) != 'undefined'){
    /* if the turtle is new create an avatar*/
    if (! $('#'+theId).length > 0){
      $('#theFrame').append(
        $('<div class="turtleAvatar" id="'+theId+'"/>').css({
        'top':Math.floor(Math.random()*500)+'px',
        'left':Math.floor(Math.random()*500)+'px'})
      );
    }
    $('#'+theId).html('<span>'+ t.theTurtleName+'</span>');
    if (typeof(t.turtleMotion) != 'undefined' ){
      var turtleSpeed = 4;
      if(t.turtleMotion == 'up'){ $('#'+theId).css({
          'transform': 'rotate(270deg)',
          'top': function(n, v) {            return  Math.max(( parseFloat(v) - turtleSpeed ), 0) ;            }
        });
        /*$('#'+theId+' span').css({      'transform': 'rotate(90deg)'    });*/
      }
      if(t.turtleMotion == 'down'){ $('#'+theId).css({
          'transform': 'rotate(90deg)',
          'top': function(n, v) {            return  Math.min(( parseFloat(v) + turtleSpeed ), 780) ;        }
        });
        /*$('#'+theId+' span').css({      'transform': 'rotate(270deg)'    });*/
      }
      if(t.turtleMotion == 'left'){ $('#'+theId).css({
          'transform': 'rotate(180deg)',
          'left': function(n, v) {            return  Math.max(( parseFloat(v) - turtleSpeed ), 0) ;            }
        });
        /*$('#'+theId+' span').css({      'transform': 'rotate(180deg)'    });*/
      }
      if(t.turtleMotion == 'right'){ $('#'+theId).css({
          'transform': 'rotate(0deg)',
          'left': function(n, v) {            return  Math.min(( parseFloat(v) + turtleSpeed ), 1620) ;       }
        });
        /*$('#'+theId+' span').css({      'transform': 'rotate(0deg)'    });*/
      }
    }
    //only update color if it's different from before?
    if (typeof(t.turtleColor) != 'undefined'  ){
       if (t.turtleColor == 'red'){  $('#'+theId).css({
         'background': 'rgba(255, 0, 0, 0.2) url(hurtle-red-250.gif) no-repeat 50% 50%',
         'border': '5px solid rgba(255, 0, 0, 0.7)'}) ;
       }
       if (t.turtleColor == 'orange'){  $('#'+theId).css({
         'background': 'rgba(255, 165, 0, 0.2) url(hurtle-orange-250.gif) no-repeat 50% 50%',
         'border': '5px solid rgba(255, 165, 0, 0.7)'});
       }
       if (t.turtleColor == 'yellow'){  $('#'+theId).css({
         'background': 'rgba(255, 255, 0, 0.2) url(hurtle-yellow-250.gif) no-repeat 50% 50%',
         'border': '5px solid rgba(255, 255, 0, 0.7)'});
       }
```

```
      if (t.turtleColor == 'green'){  $('#'+theId).css({
       'background': 'rgba(0, 128, 0, 0.2)  url(hurtle-green-250.gif) no-repeat 50% 50%',
       'border': '5px solid rgba(0, 128, 0, 0.7)'});
      }
      if (t.turtleColor == 'blue'){  $('#'+theId).css({
       'background': 'rgba(0,0, 255, 0.2) url(hurtle-blue-250.gif) no-repeat 50% 50%',
       'border': '5px solid rgba(0,0,255, 0.7)'});
      }
      if (t.turtleColor == 'indigo'){  $('#'+theId).css({
       'background': 'rgba(75, 0, 130, 0.2) url(hurtle-indigo-250.gif) no-repeat 50% 50%',
       'border': '5px solid rgba(75, 0, 130, 0.7)'});
      }
      if (t.turtleColor == 'violet'){  $('#'+theId).css({
       'background': 'rgba(238, 130, 238, 0.2) url(hurtle-violet-250.gif) no-repeat 50% 50%',
       'border': '5px solid rgba(238, 130, 238, 0.7)'});
      }
    }
  }
}
function getAvatars(){
  var avatarIds =[];
  $('.turtleAvatar').map(function(){  avatarIds.push($(this).attr('id')); });
  return avatarIds;
}
function apertMemorySetFor(id,key,value) {
  var password = getPassword();
  if(password == null) return;
  var m = { password: password, request: 'setFor', id: id, key: key, value: value};
  var n = JSON.stringify(m);
  ws.send(n);
}
function apertMemoryPoll(){
  if (wsIsUp()){
    var password = getPassword();
    if(password == null) return;
    var m = { password: password, request: 'dump' };
    var n = JSON.stringify(m);
    ws.send(n);
  }
  /*get existing avatars*/
  var theAvatars = getAvatars();
  /*update avatars with snapshot data*/
  for (someTurtle in apertMemorySnapshot){
    for (aId in theAvatars){ if (theAvatars[aId] == someTurtle){ delete theAvatars[aId]; } }
    turtleUpdate(someTurtle, apertMemorySnapshot[someTurtle]);
  }
  /*dissapear disconnected avatars*/
  for (aId in theAvatars){
    $('#'+theAvatars[aId]).remove();
  }
  setTimeout(apertMemoryPoll, 60 );
}
function canvasManagement(){
  var theAvatars = getAvatars();
  var thePositions = [];
  for (aId in theAvatars){
```

```
    var str = $('#'+theAvatars[aId]).css('left');
    str = str.substring(0, str.length - 2);
    thePositions[aId] = parseFloat(str);
   }
   if ( thePositions.length ){
    var sum = thePositions.reduce(function(a, b) { return a + b; });
    var avg = sum / thePositions.length;
    console.log(thePositions);
    console.log(avg);
    if (avg < 810){
      theImageDirection = 1;
    }
    else{
      theImageDirection = -1;
    }
    theImageSpeed = Math.abs(avg - 810).map(0,810, 1, 20);
   }
   setTimeout(canvasManagement, 300 );
}
function turtleSong(){
   var theAvatars = getAvatars();
   for (aId in theAvatars){  turtleBeat(theAvatars[aId]); }
   setTimeout(turtleSong, 200 );
}
$( document ).ready(function() {
   setup();
   rockOn();
   apertMemoryPoll();
   canvasManagement();
   turtleSong();
});
</script>
</body>
</html>
```

ReptileChoir Display Interface CSS

```
body {
   font-size: 16px;
   font-family: "Oxygen", sans-serif;
   line-height: 1.5;
}
::selection {
   background: rgba(170,138,138,0.5);/* WebKit/Blink Browsers */
}
::-moz-selection {
   background: rgba(170,138,138,0.5); /* Gecko Browsers */
}
#theCode {
   position: absolute;
   z-index: 12;
```

```
  top: 40%;
  padding: 40px;
  border: none;
  overflow: hidden;
  width: 1840px;
  height: auto;
  text-align: center;
  font-size: 100px;
  color: #fff;
  font-family: "Oxygen", sans-serif;
  opacity: 0.8;
  background-color: rgba(170,138,138,0);
  text-shadow: 4px 4px #000000;
}
#doneCode{
  position: absolute;
  z-index: 11;
  top: 0px;
  right: 0px;
  padding: 15px;
  width: 1000px;
  font-size: 24px;
  height: 1000px;
  overflow: hidden;
  color: #fff;
  text-align: right;
}
#doneCode div{
  float:right;
  clear: both;
  background-color: rgba(170,138,138,0.5);
  padding: 5px;
  margin-bottom: 5px;
}
/* ================================================================
  FRAME
================================================================ */
#theFrame {
  position: absolute;
  z-index: 3;
  margin: 0 auto;
  overflow: hidden;
  width: 100%;
  height: 100%;
  text-align: center;
}

#theRedPill, .turtleAvatar {
            z-index: 9;
```

```
 /*transform: scale(0.5);*/
            font-family: "Oxygen", sans-serif;
            background: rgba(255, 255, 255, 0.1) url('../hurtle-right-250.gif') no-repeat 50% 50%;
            padding: 20px;
   font-size: 24px;
            color: white;
   width: 250px;
   height: 250px;
   position: absolute;
          top: 300px;
          left: 100px;
border-radius: 150px;
}
```

StreetSong Mobile Javascript

```
theAmp = 1;
function killBeeps(){  theAmp = .1; }
var audioReady = false;
var theSamples = {};
var theSinger = {};
var theScore = {
 'harmonicPace': 0,
 'chordDissonance' : 0.3,
 'polyphony':0.3,
 'concreteChord': [],
 'pitchSpreadUpper':0.5,
 'growl':0.1,
 'lyric':'street',
 'sopranoBoost': 0.5,
 'bassBoost': 0.5,
 'punctuation': 0.1,
 'dynamic':0.5,
 'duration':0.2,
 'brightness':0.4,
 'progress':0
}; // updated moment by moment.
var chordSet={
 0: [64,69], 1: [60,64,69], 2: [60,65,69], 3: [64,65,69],
 4: [60,64,65,69], 5: [60,64,65,69,71], 6: [60,62,64,65,67,69,71],
 7: [60,62,64,65,67,68,69,71]
};
var wordSet={
 0:'lane',1:'street',2:'drive',3:'road',4:'route'
};
var compressor;
var gain;
// true: auto fetch notes continually. play button becomes pause button.
// false: 1 note at a time.
var autoTraverse = true;
```

```
// enable singing
var isSinging = false;

// the regular scale fits ~c2 to g5 but maybe you can go beyond?
var midiNotes = {
   40:'E2 ledger',41:'F2',42:'F2 sharp',43:'G2', 44:'G2 sharp', 45:'A2', 46: 'A2 sharp', 47: 'B2',
   48: 'C3', 49: 'C3 sharp', 50:'D3',51:'D3 sharp',52:'E3',53:'F3',54:'F3 sharp',55:'G3', 56:'G3 sharp', 57:'A3', 58: 'A3
sharp', 59: 'B3',
   60: 'C4 ledger', 61: 'C4 sharp ledger', 62:'D4',63:'D4 sharp',64:'E4',65:'F4',66:'F4 sharp',67:'G4', 68:'G4 sharp',
69:'A4', 70: 'A4 sharp', 71: 'B4',
   72: 'C5', 73: 'C5 sharp', 74:'D5',75:'D5 sharp',76:'E5',77:'F5',78:'F5 sharp',79:'G5', 80:'G5 sharp', 81:'A5 ledger',
82: 'A5 ledger sharp'
  };
var theNotes = ['C', 'Db', 'D', 'Eb', 'E', 'F', 'Gb', 'G', 'Ab', 'A', 'Bb', 'B'];
var theColor;
var theVowel = 'u';
var theRole;
var theTop = 0;
var keysExist = false;
var ppmSliderExists = false;
var staveExists = false;
var pixelsPerMinute = 600;
var previousPitch;
Number.prototype.map = function (in_min, in_max, out_min, out_max) {
  return (this - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
function wsIsUp(){
  if (typeof(ws) == 'object'){
    if( ws.readyState){ return true; }
    else{  console.log('WebSocket not ready');  }
  }
  else{ console.log('WebSocket object does not exist'); }
  return false;
}

/* I use a hidden soundcloud widget as a hack for iOS devices to make webAudio work. */
/*function soundCloudSetup(){
  if (typeof (SC) == 'undefined') { setTimeout(soundCloudSetup, 100); return; }
  var widgetIframe = document.getElementById('soundCloud');
  var widget       = SC.Widget(widgetIframe);
  widget.bind(SC.Widget.Events.READY, function() {
    $('#scContainer').css({'transform': 'scale(2)'});
     widget.bind(SC.Widget.Events.PLAY, function() {
       $('#soundCloud').css({'marginTop': '200px'});
       $('#scContainer').css({'backgroundPosition': '50% 50%'});
     });
     widget.bind(SC.Widget.Events.FINISH, function() {
        $('#scContainer').remove();
        audioReady =true;
```

```
    });
  });
}*/
// derive a chord from currently lit keys
// set the chord in apert shared public memory.
function setChord(){
  var newChord = [];
  for(i in theNotes){
    if ($('#'+theNotes[i]).hasClass('active')){
      newChord.push(parseFloat(i) + 60);
      // Derivatives of the base chord are created elsewhere
      // but could opt to do it here.
      // newChord.push(parseFloat(i) + 60 + 12);
      // newChord.push(parseFloat(i) + 60 - 12);
    }
  }
  newChord.sort();
  console.log(newChord);
  apertMemorySetPublic( 'theChord', newChord );
}
// toggle the piano key state
function toggleButton(element){
  if ($(element).hasClass('active') ){
    $(element).removeClass('active');
  }else{
    $(element).addClass('active');
  }
  setChord();
}
//
function selectButton(element){
  var theClass = $(element).attr('class');
  if ( wsIsUp() ) {
      apertMemorySet( theClass, $(element).html() );
      console.log('set ' + theClass + ' to ' + $(element).html() );
  }
  $('.'+theClass).css({'borderWidth' : '2px', 'margin' : '10px'});
  $(element).css({'borderWidth' : '8px', 'margin' : '4px'});
}
/* Names for singers */
function randomName(){
  var items = [
    "Aretha Franklin", "Ray Charles", "Elvis Presley", "Sam Cooke", "John Lennon",
  "Marvin Gaye", "Bob Dylan", "Otis Redding", "Stevie Wonder", "James Brown",
  "Paul McCartney", "Little Richard", "Roy Orbison", "Al Green", "Robert Plant",
  "Mick Jagger", "Tina Turner", "Freddie Mercury", "Bob Marley", "Smokey Robinson",
  "Johnny Cash", "Etta James", "David Bowie", "Van Morrison", "Michael Jackson",
  "Jackie Wilson", "Hank Williams", "Janis Joplin", "Nina Simone", "Prince",
  "Howlin' Wolf", "Bono", "Steve Winwood", "Whitney Houston", "Dusty Springfield",
```

```
 "Bruce Springsteen", "Neil Young", "Elton John", "Jeff Buckley", "Curtis Mayfield",
 "Chuck Berry", "Joni Mitchell", "George Jones", "Bobby Bland", "Kurt Cobain",
 "Patsy Cline", "Jim Morrison", "Buddy Holly", "Donny Hathaway", "Bonnie Raitt",
 "Gladys Knight", "Brian Wilson", "Muddy Waters", "Luther Vandross", "Paul Rodgers",
 "Mavis Staples", "Eric Burdon", "Christina Aguilera", "Rod Stewart", "Brant Bjork",
 "Roger Daltrey", "Lou Reed", "Dion DiMucci", "Axl Rose", "David Ruffin",
 "Thom Yorke", "Jerry Lee Lewis", "Wilson Pickett", "Ronnie Spector", "Gregg Allman",
  "Toots Hibbert", "John Fogerty", "Dolly Parton", "James Taylor", "Iggy Pop",
  "Steve Perry", "Merle Haggard", "Sly Stone", "Mariah Carey", "Frankie Valli",
  "John Lee Hooker", "Tom Waits", "Patti Smith", "Darlene Love", "Sam Moore",
  "Art Garfunkel", "Don Henley", "Willie Nelson", "Solomon Burke",
  "The Everly Brothers", "Levon Helm", "Morrissey", "Annie Lennox", "Karen Carpenter",
  "Patti LaBelle", "B.B. King", "Joe Cocker", "Stevie Nicks", "Steven Tyler",
  "Mary J. Blige", "Beyoncé"];
 var item = items[Math.floor(Math.random()*items.length)];
 apertMemorySet('theName', item );
 return item;
}
/*  */
function interfaceSetup(){
 if (!wsIsUp()  ) {    setTimeout(interfaceSetup, 100); return; }


$("head").append(
 $("<style type='text/css'>" +
  "body{"+
   "font-family: sans-serif;"+
   "background-color: #f5f5f5;"+
  "}"+
  "button { "+
   "background: #fff; border: 4px solid #ccc; float: left; margin: 10px;" +
   "font-size: 24px; color: 333; padding: 5px 15px;"+
  "} " +
  ".container {"+
   "width: 100%; height: 360px; background: #eee;"+
  "}"+
  ".active{"+
   "border: 4px solid red;"+
  "}" +
  ".titleLabel{"+
   "font-size: 24px; color: #aaa; float: left; margin: 15px;"+
  "}"+
  "input {"+
   "padding: 5px; border: 2px solid #ccc; width: 400px; float: left;  margin: 5px; " +
   "font-size: 30px; color: #333;"+
  "}"+
  ".theNote {"+
   "position: absolute; padding: none; "+
  "}"+
```

```
"#C, #D, #E, #F, #G, #A, #B {"+
  "height: 360px; width: 14.285%;"+
  "z-index: 20;"+
  "background: #fff;"+
  "color: #000;"+
"}"+
"#Db, #Eb, #Gb, #Ab, #Bb {"+
  "height: 240px;"+
  "width: 8.074%;"+
  "z-index: 21;"+
  "background: #000;"+
  "color: #fff;"+
"}"+
"#C { left: 0%; }"+
"#Db { left: 8.903%; }"+
"#D { left: 14.285%; }"+
"#Eb { left: 25.88%; }"+
"#E { left: 28.57%; }"+
"#F { left: 42.855%; }"+
"#Gb { left: 51.139%; }"+
"#G { left: 57.14%; }"+
"#Ab { left: 67.288%; }"+
"#A { left: 71.425%; }"+
"#Bb { left: 83.644%; }"+
"#B { left: 85.71%; }"+
".ui-slider{"+
  "clear: left; margin: 20px; width: 50%;"+
"}"+
".staffLine{"+
  "border-bottom: 2px solid #000; width: 100%; height:13%;"+
"}"+
".trebleClef{"+
  "position: relative; width: 100%; height: 50%; "+
  "background: url(trebleClef.png) no-repeat 30px 0px; "+
  "background-size: auto 98%;"+
"}"+
".bassClef{"+
  "position: relative; width: 100%; height: 50%; "+
  "background: url(bassClef.png) no-repeat 20px 41%; "+
  "background-size: auto 48%;"+
"}"+
".dynamic{"+
  "visibility: hidden;"+
  "position: absolute; "+
  "height: 100%; "+
  "width: 50px; "+
  "top: -44%; "+
  "left: -50%; "+
  "background-size: auto 100%; "+
```

```
  "background-repeat: no-repeat;"+
"}"+
".ff .dynamic{"+
  "visibility: visible; " +
  "background-image: url(ff.png); "+
"}"+
".f .dynamic{"+
  "visibility: visible; " +
  "background-image: url(f.png); "+
"}"+
".mf .dynamic{"+
  "visibility: visible; " +
  "background-image: url(mf.png); "+
"}"+
".mp .dynamic{"+
  "visibility: visible; " +
  "background-image: url(mp.png); "+
"}"+
".p .dynamic{"+
  "visibility: visible; " +
  "background-image: url(p.png); "+
"}"+
".pp .dynamic{"+
  "visibility: visible; " +
  "background-image: url(pp.png); "+
"}"+
".aLyric{"+
  "margin-left: 60%; "+
  "position: absolute; width: 100px; height: 13%; "+
  "text-align: center; "+
  "font-weight: bold; "+
  "position: absolute; "+
"}"+
" .trebleClef .aLyric{"+
  " top: 98%; "+
"}"+
" .bassClef .aLyric{"+
  " top: -5%; "+
"}"+
".symbol{"+
  "visibility: hidden;"+
"}"+
".sharp .symbol{ "+
  "visibility: visible; " +
  "position: absolute; "+
  "height: 200%; "+
  "width: 50px; "+
  "top: -44%; "+
  "left: 0; "+
```

```
  "background: url(sharp.png) no-repeat; "+
  "background-size: auto 100%; "+
"}"+
".flat .symbol{ "+
  "visibility: visible; " +
  "position: absolute; "+
  "height: 150%; "+
  "width: 50px; "+
  "top: -55%; "+
  "left: 0; "+
  "background: url(flat.png) no-repeat; "+
  "background-size: auto 100%; "+
"}"+
"#singerContainer{"+
  "background: white; height: 400px; padding-top: 20px; "+
"}"+
".notes{ float: left; width: 100%; height: 100%; }"+
" .aNote { "+
    "margin-left: 60%; "+
    "background-size: auto 100%;"+
    "position: absolute; width: 100px; height: 13%; "+
    "background-position: 50% 50%; "+
    "background-repeat: no-repeat; "+
  " } "+
" .whole { "+
    "background-image: url(wholeNote.png);"+
  " } "+
" .percussive { "+
    "background-image: url(percNote.png);"+
  " } "+
" .rest { "+
    "background-image: url(wholeRest.png);"+
  " } "+
  ".lineNote{ left: 300px; }"+
  ".spaceNote{ left: 400px; }"+
" .G5 { top: 8%; }"+
" .F5 { top: 15%; }"+
" .E5 {  top: 22%; }"+
" .D5 { top: 29%; }"+
" .C5 {  top: 36%; }"+
" .B4 { top: 43%; }"+
" .A4 {  top: 50%; }"+
" .G4 { top: 57%; }"+
" .F4 {  top: 64%; }"+
" .E4 { top: 71%; }"+
" .D4 {  top: 78%; }"+
" .C4 {  top: 85%; }"+
/* Bass Clef Here */
" .B3 { top: 8%; }"+
```

```
    " .A3 {  top: 15%; }"+
    " .G3 { top: 22%; }"+
    " .F3 {  top: 29%; }"+
    " .E3 { top: 36%; }"+
    " .D3 {  top: 43%; }"+
    " .C3 {  top: 50%; }"+
    " .B2 { top: 57%; }"+
    " .A2 {  top: 64%; }"+
    " .G2 { top: 71%; }"+
    " .F2 {  top: 78%; }"+
    " .E2 { top: 85%; }"+
    " .D2 {  top: 92%; }"+
    " .C2 {  top: 99%; } "+

    ".line { visibility: hidden; } " +
    ".ledger .line { "+
      "visibility: visible; "+
      "margin: 0 auto; "+
      "height: 50%; width: 80%; border-bottom: 2px solid #000; "+
    "} "+
  "</style>")
);

  $('body').append(
    $('<div id="myNameIs" />').css({  'width': '100%' , 'height': '100px'}). append(
      $('<div class="titleLabel">I am </div>'),
      $('<input id="theName" value="'+ randomName() +'">')
        .on('input',function(){
          console.log('here');
          apertMemorySet('theName', $(this).val() );
          console.log('there');
        })
    ),
    $('<div id="swimContainer" />').css({
       'width': '100%', 'height' : '80px', 'background' : '#eee' }). append(
      $('<div class="titleLabel">I will </div>'),
      $('<input type="hidden"  id="theRole" >'),
      $('<button type="button" class="theRole">Compose</button> ').
        click(function(){
            selectButton(this);
            $('.container').hide();
            $('#composerContainer').show();
            if ( keysExist == false ){
              for (j in theNotes){
                $('#composerContainer').append(
                  $('<button type="button" class="theNote" id="'+theNotes[j]+'"></button> ').
                    click(function(){  toggleButton(this);  })
                );
              }
```

```
            keysExist = true;
          }
        }),
    $('<button type="button" class="theRole">Conduct</button> ').
      click(function(){
        selectButton(this);
        $('.container').hide();
        $('#conductorContainer').show();
        if ( ppmSliderExists == false ){
          $("#ppmSlider").slider({
            value: pixelsPerMinute, min: 60, max: 2400, step: 10,
            slide: function(event, ui) {
              $('#pixelsPerMinute').val(ui.value);
              if ( wsIsUp() ) {
                apertMemorySetPublic( 'pixelsPerMinute', ui.value );
                console.log('set pixelsPerMinute to ' + ui.value );
              }
            }
          });
          ppmSliderExists = true;
        }
      }),
    $('<button type="button" class="theRole">Sing</button> ').
      click(function(){
        apertStartAudio();
        loadSamples();
        var lowNote = Math.floor(Math.random()*24)+36;
        var range = 12 + Math.floor(Math.random()*12);
        theSinger = new VirtualSinger(lowNote,lowNote+range);
        selectButton(this);
        $('.container').hide();
        $('#singerContainer').show();
        apertMemoryGetPublic('pixelsPerMinute');
      }),
      $('<button type="button" class="theRole">Lyricize</button> ').
        click(function(){
          selectButton(this);
          $('.container').hide();
          $('#lyricContainer').show();
        })
  ),
  $('<div class="container" id="composerContainer" />').css({
      'display' : 'none' }). append(
    $('<div class="titleLabel" >Chords </div>')
  ),
  $('<div class="container" id="conductorContainer" />').css({
      'display' : 'none' }). append(
    $('<div style="height: 80px;">'+
      '<label class="titleLabel" for="pixelsPerMinute">Pixels Per Minute</label>'+
```

```
      '<input  id="pixelsPerMinute" value="' + pixelsPerMinute + '" />'+
    '</div>'+
    '<div id="ppmSlider" />'
  )
),
$('<div class="container" id="singerContainer" />').css({
   'display' : 'none' }). append(
  $('<div class="trebleClef">'+
     '<div class="notes"></div>'+
     '<div class="spacer" style="width: 100%; height:8%;"></div>'+
     '<div class="staffLine"></div>'+
     '<div class="staffLine"></div>'+
     '<div class="staffLine"></div>'+
     '<div class="staffLine"></div>'+
     '<div class="staffLine"></div>'+
     '<div class="spacer" style="width: 100%; height:27%;"></div>'+
  '</div>'+
  '<div class="bassClef">'+
     '<div class="notes"></div>'+
     '<div class="spacer" style="width: 100%; height:8%;"></div>'+
     '<div class="staffLine"></div>'+
     '<div class="staffLine"></div>'+
     '<div class="staffLine"></div>'+
     '<div class="staffLine"></div>'+
     '<div class="staffLine"></div>'+
     '<div class="spacer" style="width: 100%; height:27%;"></div>'+
  '</div>'),
  $('<div id="autoTraverse"></div>').css({
    'background' : 'url(autoTraverse.png) no-repeat -50px 0px', 'float' : 'left',
    'width' :'50px', 'height':'50px'}).
    click(function(){
       if (autoTraverse == true){
          autoTraverse = false;
          $(this).css({'backgroundPosition': '0px 0px'});
       }
       else{
         autoTraverse = true;
         $(this).css({'backgroundPosition': '-50px 0px'});
       }
    }),
    $('<div id="playPause"></div>').css({
     'background' : 'url(playPause.png) no-repeat 0px 0px', 'float' : 'left',
     'width' :'50px', 'height':'50px'}).
     click(function(){
        if (isSinging == true){
           isSinging = false;
           $(this).css({'backgroundPosition': '0px 0px'});
        }
        else{
```

```
            startSinger();
            isSinging = true;
            $(this).css({'backgroundPosition': '-50px 0px'});
          }
        })
        // if you were to do staff.hide()
        // you might call it a staff retreat?
    /*$('<canvas id="theStaff" width=700 height=100"></canvas>')*/
  ),
  $('<div class="container" id="lyricContainer" />').css({
     'display' : 'none' }). append(
    $('<div style="height: 80px;">'+
     '<label class="titleLabel" for="theLyrics">Lyrics</label>' +
     '<input  id="theLyrics" value="OOoo-ahh" /> ' +
    '</div>')
  )
).css({'background': '#ddd'} );


}


$( document ).ready(function() {


/*  if (navigator.userAgent.match(/(iPod|iPhone|iPad)/)) */
 $("<link>").attr({ rel: "stylesheet", type: "text/css", href: "jquery-ui.min.css" }).appendTo('head');
 var jqui = document.createElement("script");
 jqui.type = "text/javascript";
 jqui.src = "jquery-ui.min.js";
 $("head").append(jqui);

/*
 $('<div>', {id: 'scContainer'} ).appendTo('body').
  css({margin: '100px auto', width:'50px', height:'50px', background: 'url(ajax-loader.gif) no-repeat 20px 50%',
overflow: 'hidden' });
 $('<iframe>', {
   src:
'https://w.soundcloud.com/player/?url=https://soundcloud.com/nsitu/silence&auto_play=false&color=2266&&buyin
g=false&liking=false&download=false&sharing=false&show_artwork=false&show_comments=false&show_playc
ount=false&show_user=false&hide_related=false&visual=false',
    id:  'soundCloud',
    frameborder: 'no',
    scrolling: 'no'
  }).appendTo('#scContainer').css({'marginTop':'-5px', 'marginLeft':'-5px'});
  var scapi = document.createElement("script");
  scapi.type = "text/javascript";
  scapi.src = "https://w.soundcloud.com/player/api.js";
  $("head").append(scapi);
```

```
    soundCloudSetup();*/
    interfaceSetup();
  });
SampleBuffer = function(url) {
  var request = new XMLHttpRequest();
  request.open('GET',url,true);
  request.responseType = 'arraybuffer';
  var closure = this; // a closure is necessary for...
  request.onload = function() {
    ac.decodeAudioData(request.response, function(x) {
      console.log("sample " + url + "loaded");
      closure.buffer = x;
    });
  };
  request.send();
}
// a function to calculate amplitude given dB
db = function(x) { return Math.pow(10,x/20); }


function apertInitialize() {
  setInterval(function() {
    if ( wsIsUp() ) {
      // no need to run intervals :
      // the interface is the source of action
    }
  },137); // frequent enough to feel reponsive.
}

function apertMemorySetPublic(key,value) {
  // call this in your code to set a key-value pair in a shared memory
  // entries are unique to each client/browser
  if (wsIsUp()){
    var m = { request: 'setPublic', key: key, value: value};
    var n = JSON.stringify(m);
    ws.send(n);
  }
}
function apertMemoryGetPublic(key) {
  if (wsIsUp()){
    /* doesn't this message go to everyone? */
    var m = { request: 'getPublic', key: key};
    var s = JSON.stringify(m);
    ws.send(s);
  }
}
function apertMemoryGet(key) {
  if (wsIsUp()){
    /* doesn't this message go to everyone? */
```

```
    var m = { request: 'get', key: key};
    var s = JSON.stringify(m);
    ws.send(s);
  }
}
// there is no apertReceivedGetPublic. Should there be?
function apertReceivedGet( msg ){
  console.log('Received Get ');
  console.log(msg.value);

}
function apertReceivedSendTo( msg ){
  console.log('Recieved Direct Message ' + msg);
  //JSON.parse(msg);?
  if( typeof( msg) != 'undefined' ){

  }
}
function midiToFreq (midi, tuning) {
  return Math.pow(2, (midi - 69) / 12) * (tuning || 440)
}
function notate(x){
  var theClasses = '';
  var theStaff = '.trebleClef';
  var theDelay = 2000;
  if (x['midiNote'] == 'rest'){
    theClasses += 'rest';
    theClasses += ' '+midiNotes[72]; // put the rest on C5
    theWord = ' ';
  }
  else if (x['isPitched'] == false )  {
    theClasses += 'percussive';
    theClasses += ' '+x['dynamic']; // mf still applies here.
    theClasses += ' '+midiNotes[74]; // put the X on D5
    theWord = x['hmm'];
  }
  else{
    theClasses += 'whole';
    theClasses += ' '+x['dynamic'];
    theClasses += ' '+midiNotes[x['midiNote']]
    var theWord = x['lyric'];
    if (x['midiNote'] < 60  )  {  theStaff = '.bassClef';   }
    theDelay = x['dur']*1000;
  }
  console.log('notating ' + x['midiNote'] + ' ' + x['dynamic'] );
  //alert (theClasses + ' ' + theWord + ' ' + theStaff);
  $('<div class="aNote">'+
      '<div class="dynamic"></div>'+
      '<div class="symbol"></div>'+
```

```
      '<div class="line"></div>'+
    '</div>')
    .addClass(theClasses)
    .appendTo( theStaff )
    .animate({ marginLeft: '-=30%' }, theDelay, function(){ $(this).remove(); });
  $('<div class="aLyric">'+theWord+'</div>')
    .appendTo( theStaff )
    .animate({ marginLeft: '-=30%' }, theDelay, function(){ $(this).remove(); });
}

function setVal(theKey, theValue){
  if (theValue != null){
    window[theKey] = theValue;
    console.log('setVal ' + theKey +' = ' + theValue);
  }
}
SampleBuffer = function(url) {
  var request = new XMLHttpRequest();
  request.open('GET',url,true);
  request.responseType = 'arraybuffer';
  var closure = this; // a closure is necessary for...
  request.onload = function() {
    ac.decodeAudioData(request.response, function(x) {
      console.log("sample " + url + "loaded");
      closure.buffer = x;
    });
  };
  request.send();
}
function loadSamples(){
  if(typeof(ac) == 'undefined'){ setTimeout(loadSamples, 80); return; }
  theSamples = {
   'rrr':  new SampleBuffer('rrr.wav'),
   'shh':  new SampleBuffer('shh.wav')
  };
}
function startSinger(){
  if(typeof(ac) == 'undefined'){ setTimeout(startSinger, 80); return; }
  if ( !theSinger.started ) { theSinger.start();  }
}
var VirtualSinger = function (lowestNote,highestNote) {
  if (lowestNote==null) lowestNote = 40;
  if (highestNote==null) highestNote = 82;
  this.lowestNote = lowestNote;
  this.highestNote = highestNote;
  this.started = false;
  this.lastRelease = 0; // a useful timestamp
  this.dynamicsSet = ['pp','p','mp','mf','f','ff'];
  this.pitchWobble = randomElement([-3,-2,-1,0,1,2,3]);
```

```
  this.sine = ac.createOscillator();
  this.oscGain = ac.createGain();
  this.wavGain = ac.createGain();
  this.compressor = ac.createDynamicsCompressor();
  this.sine.type = 'sine';
  this.sine.connect(this.oscGain);
  this.oscGain.gain.setValueAtTime(0,ac.currentTime);
  this.compressor.threshold.value = -3;
  this.compressor.ratio.value = 10;
  this.compressor.attack = 0.005;
  this.compressor.release = 0.005;
  this.oscGain.connect(this.compressor);
  this.wavGain.connect(this.compressor);
  this.compressor.connect(ac.destination);
  this.sine.start();
}
// start singing.
VirtualSinger.prototype.start = function () {
  this.started = true;
  var closure = this;
  setTimeout(function() {
    closure.callback();
  },70+Math.floor(Math.random()*130));
  // singers shouldn't all start at the same time.
  // is this to create a musical texture?
}
VirtualSinger.prototype.callback = function () {
  if(this.wantANote() ){
    var note = this.getANote(this.lastNote);
    this.singANote(note);
    this.lastNote = note;
  }
  this.start(); // loop start > callback > start > etc.
}

VirtualSinger.prototype.wantANote = function(){
  //alert(isSinging + ' ' + this.lastRelease + ' ' +ac.currentTime);
  if ( isSinging ){
    if (this.lastRelease == 0 && ac.currentTime == 0) return true;
    if (this.lastRelease < ac.currentTime) return true;
  }
  return false;
}
VirtualSinger.prototype.singANote = function(x) {
  notate(x);
  if (x['midiNote'] =='rest'){
    var now = ac.currentTime;
    this.lastRelease = now+2.0;
    return;
```

```
  }
  else if(x['isPitched'] == false){
    this.source = ac.createBufferSource();
    this.source.buffer = theSamples[x['hmm']].buffer;
    this.source.playbackRate.value = 1; // you could alter this by pitch.
    this.source.connect(this.wavGain);
    this.source.start();
    var now = ac.currentTime;
    this.wavGain.gain.setValueAtTime(0,now);
    this.wavGain.gain.linearRampToValueAtTime(db(-6),now+0.5);
    this.wavGain.gain.linearRampToValueAtTime(0,now+1.0);
    this.lastRelease = now+1.0;
    var closure = this;
    setTimeout(function() {
      closure.source.disconnect(closure.gain);
    },1500);
  } //end nonPitched note;
  else{

    //else, if the note has a pitch
    var duration = x['dur'];
    // override duration if staccato.
    if (x['isStaccato']){  duration = 0.25; }
    var amp = db(x['amp']);
    // override amplitude if accented
    if (x['isAccented']){  amp = db(-6); }
    var freq =  midiToFreq(x['midiNote']) + this.pitchWobble;
  //console.log(x['midiNote']+' '+freq);
    var attack = 0.1;
    var decay = 0.1;
    var now = ac.currentTime;
    var sustainFrom = now + attack;
    var sustainUntil = now + duration - decay;
    var stopPoint = now + duration;
    this.sine.frequency.linearRampToValueAtTime(freq, sustainFrom);
    this.oscGain.gain.setValueAtTime(0,now);
    this.oscGain.gain.linearRampToValueAtTime(amp, sustainFrom);
    this.oscGain.gain.linearRampToValueAtTime(amp, sustainUntil);
    this.oscGain.gain.linearRampToValueAtTime(0,stopPoint);
    this.lastRelease = stopPoint;
  } // end pitched note
} //end singANote()
VirtualSinger.prototype.canSing = function(midiNote){
  if ( this.highestNote >= midiNote && this.lowestNote <= midiNote ) return true;
  return false;
}
VirtualSinger.prototype.isBass = function () {
  if ( (this.highestNote - 60) < (60 - this.lowestNote) ) return true;
  return false;
```

```
}
VirtualSinger.prototype.isSoprano = function () {
 if ( (this.highestNote - 60) > (60 - this.lowestNote) ) return true;
 return false;
}
VirtualSinger.prototype.getANote = function (lastNote) {
 var isAccented = false;
 var bs = theScore['bassBoost'];
 if ( this.isBass() ) {
  if (bs > 0.75) {  isAccented = true; }
  else if (bs > 0.25) {
   if (Math.random() > bs){   isAccented = true;  }
  }
 }
 var ss = theScore['sopranoBoost'];
 if ( this.isSoprano() ) {
  if (ss > 0.75) {  isAccented = true; }
  else if (ss > 0.25) {
   if (Math.random() > ss){   isAccented = true;  }
  }
 }
 var midiNotes = theScore['concreteChord'];
 //midiNotes is full of undefineds and NaNs?
 var goodNotes = [];
 for (aNote in midiNotes){
  if ( this.canSing(midiNotes[aNote]) ){
   goodNotes.push(midiNotes[aNote]);
  }
 }
 if (goodNotes.length > 0){
  var midiNote = goodNotes[Math.floor(Math.random()*goodNotes.length)];
 }
 else{
  midiNote = 'rest';
 }
 // are you really going to pick a note randomly?
 // yes, for now.

 // notes are between 0.5 and 4 seconds
 var duration = theScore['duration'].map(0,1,1,4);
 var dynamicIndex = parseInt(theScore['dynamic'].map(0,1,0,6.1));
 var dynamic = this.dynamicsSet[dynamicIndex];
 var amplitude = parseInt(theScore['dynamic'].map(0,1,-48,-6));
 var isPitched = true;
 var hmm = '';
 if (theScore['brightness'] < 0.25){
    isPitched = false;
    hmm = 'shh';
 }
```

```
  if ( Math.random() <  theScore['growl'] ){
    isPitched = false;
    hmm = 'rrr';
  }

  var isStaccato = false;
  if (theScore['punctuation'] > 0.5){ isStaccato = true; }
  return {
    'midiNote':midiNote,
    'dur':duration,
    'amp':amplitude,
    'dynamic':dynamic,
    'isStaccato': isStaccato,
    'isAccented': isAccented,
    'isPitched' : isPitched,
    'lyric' : theScore['lyric'],
    'hmm': hmm
  };
}

function randomElement(items){
  var item = items[Math.floor(Math.random()*items.length)];
  return item;
}
// given an array arr, and a number of elements n,
// return a new array with n random elements from arr
function getRandomElements(arr, n) {
    if (n > arr.length) return arr;
    newArr = [];
    while(n--){
        var key = Math.floor(Math.random() * arr.length);
        newArr.push(arr[key]);
        arr.splice(key,1);
    }
    return newArr;
}

//=======SIMPLE SYNTH
function simple(freq,amp,duration) {
  if (duration == null ){ duration = 0.5; }
  console.log(duration);
            var sine = ac.createOscillator();
            sine.type = 'sine';
            sine.frequency.value = freq;
            var gain = ac.createGain();
            sine.connect(gain);
            gain.connect(ac.destination);
            sine.start();
            // envelope
```

```
  var attack = 0.005;
  var decay = 0.1;
        var now = ac.currentTime;
  var sustainFrom = now + attack;
  var sustainUntil = now + duration - decay;
  var stopPoint = now + duration;
  console.log(now);
  console.log(sustainFrom);
  console.log(sustainUntil);
  console.log(stopPoint);
  // the duraion isn't ideal. why?
        gain.gain.setValueAtTime(0,now);
  gain.gain.linearRampToValueAtTime(amp, sustainFrom);
  gain.gain.linearRampToValueAtTime(amp, sustainUntil);
  gain.gain.linearRampToValueAtTime(0,stopPoint);
        // schedule cleanup
        setTimeout(function() {
              sine.stop();
              sine.disconnect(gain);
              gain.disconnect(ac.destination);
        },duration*1000);
};
```

StreetSong Display Interface

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>StreetSong</title>
  <meta name="author" description="Harold Sikkema">
  <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Oxygen:400,300,700">
  <link rel="stylesheet" href="jquery-ui-1.11.4.custom/jquery-ui.min.css">
  <link rel="stylesheet" href="css/style.css">
</head>
<body>

<!-- uncomment to reenable livecode interfaec -->
<!--<textarea id="theCode"> </textarea> -->
<!--<div id="doneCode"></div>-->

<div id="theFrame">
  <div id="progressBar">
    <div id="progressUnits"></div>
    <div id="progressPercent"></div>
    <div id="progressAction">Loading</div>
  </div>
  <div id="theNow"></div>
  <div id="theShade"></div>
</div>
<div id="theAvatars"></div>
```

```
<div id="playPause"></div>
<!-- load jquery -->
<script type="text/javascript" src="jquery-1.12.1.min.js"></script>
<!--<link rel="stylesheet" href="http://code.jquery.com/ui/1.12.0/themes/base/jquery-ui.css">
<script src="http://code.jquery.com/ui/1.12.0/jquery-ui.js"></script> -->
<!--<script type="text/javascript" src="jquery-ui-1.11.4.custom/jquery-ui.min.js"></script>-->
<!--<script type="text/javascript" src="jquery.color.js"></script>
<!--<script type="text/javascript" src="jquery.caret.js"></script>-->
<!--<script type="text/javascript" src="gamepad.js"></script>-->
<script>
var ws;
var apertSnapshot;
var apertSnapshotOld;
var apertStartAudioAlreadyCalled = false;
var theCanvas;
var theCanvasWidth;
var       theContext;
var isSinging = false;
var pixelsPerMinute = 600;
var pixelsTraversed = 0;
var theNow = 0;
var theChord = [];
var newChord = false;
var fullChord = [];
var theScore = {
  'harmonicPace': 0,
  'chordDissonance' : 0.3,
  'polyphony':0.3,
  'concreteChord': [],
  'pitchSpreadUpper':0.5,
  'growl':0.1,
  'lyric':'street',
  'sopranoBoost': 0.5,
  'bassBoost': 0.5,
  'punctuation': 0.1,
  'dynamic':0.5,
  'duration':0.2,
  'brightness':0.4,
  'progress':0
}; // updated moment by moment.

// a set of chords arranged from least to most dissonant
var chordSet={
  0: [60,64,67], 1: [60,65,69], 2: [65,69,72], 3: [62,66,69],
  4: [60,64,67,71]
};
/*var chordSet={
  0: [64,69], 1: [60,64,69], 2: [60,65,69], 3: [64,65,69],
  4: [60,64,65,69], 5: [60,64,65,69,71], 6: [60,62,64,65,67,69,71],
  7: [60,62,64,65,67,68,69,71]
};*/
// a set of street-themed lyrics:
// alternates:  alley, way, trail, track, avenue, blvd, course, highway
var wordSet={
  0:'lane',1:'street',2:'drive',3:'road',4:'route'
```

```
};
//musically useful vowels
var vowelSet=[
   'ă','ā','ĕ','ē','ĭ','ī','ŏ','ō','ŭ','ū'
];
//musically useful consonants
var consonantSet={
   0:"Sss", 1:"Ttt", 2:"Rrr",3:"Eee",4:"Ttt"
};

var theSamples = {};
var theColumns = []; /*columns in the image*/
var theRanges = [];  /* vertical spans of interesting pixels*/
var theOverlaps = [];
var theWeakOverlaps = [];
var theActiveOverlaps = [];
var theContinuities = []; /*store data about continuities here */
var theSignals = []; /*store data about image here */
var theGlobals ={};
theGlobals['sdMax'] = {0:0,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0};
theGlobals['slMax'] = 0; //slopeMax
theGlobals['pkMax'] = 0; //slopeMax
theGlobals['trMax'] = 0; //slopeMax
theGlobals['ysdMax'] = 0; //yPos standard deviation max
theGlobals['ccMax'] = 0; //continuity count
theGlobals['ccyMax'] = 0; //continuity count
theGlobals['ccdMax'] = 0; //continuity count
theGlobals['stMax'] = 0; //smooth thickness max
theGlobals['styMax'] = 0; //smooth thickness yellow max
theGlobals['stdMax'] = 0; //smooth thickness dark max
theGlobals['rsMax'] = 0; // reachsum max
theGlobals['maxReach'] = 1;
theGlobals['maxSlope'] = 0;
theGlobals['minSlope'] = 0;
theGlobals['maxAccel'] = 0;
theGlobals['minAccel'] = 0;
theGlobals['maxRangeSpan'] = 20; // like window.maxRangeSpan
theGlobals['minRangeSpan'] = 10; // but is actual (not a filter def)
var cachedContinuities = null; /* may be populated from server to override data mining process. */
var cachedSignals = null; /* may be populated from server to override data mining process. */
var cachedGlobals = null; /* may be populated from server to override data mining process. */
var cacheUsed = false;

//used by loadImage() and also to index stringified continuities.
//traversal.jpg
// 259A4596-crop.jpg
//259A4596-crop2k.jpg
//fold.jpg
//259A4596-side.jpg
//259A4551-side-1080-stretched.jpg
//var theImageFile = '259A4596-crop2k.jpg';
//var theImageFile = 'traversal.jpg';
var theImageFile = '259A4596-side.jpg';
var currentColumn = 0; //start at x-position on the left
var lastDrawnColumn = 0;
var cCount = 0;         /* a  tally of continuities */
```

```
var minRangeSpan = 4; // ranges must span at least this many pixels
var maxRangeSpan = 200; // ranges wider than this are probably ghosts
var minContinuityXSpan = 20; // skip short continuities
var keepNotes = 6; //lower this for possible performance gain
function wsIsUp(){
  if (typeof(ws) == 'object'){
    if( ws.readyState){
      return true;
    }
  }
  return false;
}
// there is no apertReceivedGetPublic. Should there be?
function apertReceivedGet( msg ){
  //console.log('Received Get ');
  //console.log(msg.value);
  if( typeof( msg.value) != 'undefined'){
      window[msg.key] = msg.value;
      if (msg.key == 'theChord'){
          newChord = true; //we got notes!
      }
  }
}
function setup(theServer) {
  window.WebSocket = window.WebSocket || window.MozWebSocket;
  // *** NOTE: port should not be hardwired in the line below!!!
  // location.hostname evaluates to "" on windos when loading web pages directly from the filesystem.
if (theServer == 'LIVELab'){
  theServer = 'intramuros.mcmaster.ca';
}
if (theServer == null){
//theServer = '127.0.0.1' ;
//theServer = '104.131.34.122';
theServer = 'nsitu.ca';
// theServer = 'intramuros.mcmaster.ca';
//theServer = '130.113.48.96';
//theServer = location.hostname;
}
  var url = 'ws://' + theServer + ':8000';
  //var url =  theIntraIP + ':8000';
  //var url = 'ws://nsitu.ca:8000';
  console.log("attempting websocket connection to " + url);
  ws = new WebSocket(url);
  ws.onopen = function () { console.log("websocket connection opened"); };
  ws.onerror = function () { console.log("ERROR opening websocket connection"); };
  ws.onmessage = function (m) {
    var data = JSON.parse(m.data);
    if(data.type == 'all') {
      //console.log("/all " + data.name + " " + data.args);
      var name = data.name;
      if(data.args.length == 0) eval(name + "()");
      else if(data.args.length == 1) eval(name + "(data.args[0])");
      else if(data.args.length == 2) eval(name + "(data.args[0],data.args[1])");
      else if(data.args.length == 3) eval(name + "(data.args[0],data.args[1],data.args[2])");
      else if(data.args.length == 4) eval(name + "(data.args[0],data.args[1],data.args[2],data.args[3])");
      else if(data.args.length == 5) eval(name + "(data.args[0],data.args[1],data.args[2],data.args[3],data.args[4])");
```

```
      else if(data.args.length == 6) eval(name +
"(data.args[0],data.args[1],data.args[2],data.args[3],data.args[4],data.args[5])");
      else console.log("warning: too many arguments in all message, apert is unfinished software, so sorry, try again
later");
      // should probably check to make sure the function exists first, also!...
    }
    else if(data.type =='refreshCount') {
      /*document.getElementById('refreshCount').textContent = data.count;*/
    }
    else if(data.type =='clientCount') {
      /*document.getElementById('clientCount').textContent = data.count; */
    }
    else if(data.type == 'get'){
      /* if you recieve a get message */
      console.log('Apert Received Get:');
      console.log(apertReceivedGet);
      if (typeof(apertReceivedGet) == 'function') {
        apertReceivedGet(data);
      }
    }
    else if(data.type == 'dump') {
      apertSnapshotOld = apertSnapshot;
      apertSnapshot = data.result;
    }
    else {
      console.log("received WebSocket message of unknown type = " + data.type);
    }
  }
  //
}
function getPassword() {
  var thePassword =  'tsh203';
  return thePassword;
}
function load(path) {
  var password = getPassword();
  if(password == null) return;
  var path = document.getElementById('load').value;
  if(path == null) return;
  if(path == "") return;
  var m = { password: password, request: 'load', path: path };
  var n = JSON.stringify(m);
  ws.send(n);
}
function refresh() {
  var password = getPassword();
  if(password == null) return;
  var m = { password: password, request: 'refresh' };
  var n = JSON.stringify(m);
  ws.send(n);
}
// the function below should be called once
// to create a valid audio context. For things to work on iOS,
// this function needs to be called from a user interaction event.
function apertStartAudio() {
  if(apertStartAudioAlreadyCalled)return;
```

```
  apertStartAudioAlreadyCalled=true;
  try {
    window.AudioContext = window.AudioContext||window.webkitAudioContext;
    ac = new AudioContext();
    console.log("created new audio context");
  }
  catch(e) {
    alert('Web Audio API is not supported in this browser');
  }
  if (typeof apertInitialize == 'function') {
    console.log('calling apertInitialize...');
    apertInitialize(); // call initializer function provided by specific loaded JavaScript
    console.log('returned from apertInitialize');
  }else{
    console.log('There was no apertInitialize function. You can make one if you like!');
  }
  apertSilentNote(); // create a silent synth to unmute audio on iOS
  // ac = new (window.AudioContext||window.webkitAudioContext)();
}
// a silent note, to be triggered by touch event calling apertStartAudio to unmute iOS audio
function apertSilentNote() {
            var sine = ac.createOscillator();
            sine.type = 'sine';
            sine.frequency.value = 440;
            var gain = ac.createGain();
            sine.connect(gain);
            gain.connect(ac.destination);
            sine.start();
            var now = ac.currentTime;
            gain.gain.setValueAtTime(0,ac.currentTime);
}

function testOn() {
  var password = getPassword();
  if(password == null) return;
  var m = { password: password, request: 'all', name: 'testOn', args:[] };
  var n = JSON.stringify(m);
  ws.send(n);
}
function testOff() {
  var password = getPassword();
  if(password == null) return;
  var m = { password: password, request: 'all', name: 'testOff', args:[] };
  var n = JSON.stringify(m);
  ws.send(n);
}
Number.prototype.map = function (in_min, in_max, out_min, out_max) {
  return (this - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
function loadImage(url){
  $("<img/>")
    .attr('crossOrigin', "Anonymous")
    .on('load', function() { makeCanvas(this); console.log("image loaded correctly"); })
    .on('error', function() { console.log("error loading image"); })
    .attr("src", url)
    .css('display', 'none')
```

```
    .appendTo('#theFrame');
}
function makeCanvas(someImage){
  $('canvas').remove(); // kill  old canvasses
            //Make a New Canvas
            theCanvas = $("<canvas/>").appendTo('#theFrame');
  theCanvas[0].width = someImage.width;
  theCanvas[0].height = someImage.height;
            theContext = theCanvas[0].getContext('2d');
            theContext.drawImage(someImage, 0, 0, someImage.width, someImage.height);
  theCanvasWidth = someImage.width; //global reference;
  setupIndeces();
  loadData(); //check cache for existing data
  findRanges(); // build new data if needed.
  chordCycle();
}
function loadData(){
  var request;
  request = $.ajax({
    url: "storeImageData.php",
    type: "GET",
    data: {fileName:theImageFile}
  });
  request.done(function (response, textStatus, jqXHR){
    if (textStatus == 'success'){
      console.log("Cached Data Found");
      //console.log(response);
      if (response == 'ERROR'){  console.log(jqXHR);  }
      else{   var dataArray = JSON.parse(response);
        cachedSignals = dataArray[0];
        cachedContinuities = dataArray[1];
        cachedGlobals = dataArray[2];
      }
    }
    else{
      console.log('Server Error.');
    }
  });
}
function saveData(){
  var request;
  var theString = JSON.stringify([theSignals,theContinuities,theGlobals]);
  request = $.ajax({
    url: "storeImageData.php",
    type: "POST",
    data: {json: theString, fileName:theImageFile}
  });
  request.done(function (response, textStatus, jqXHR){
      if (textStatus == 'success'){
        //console.log(response);
        if (response == 'OK'){
           console.log('Cache data saved to server');
        }
      }
      else{
         console.log('ERROR: could not save cache data to server');
```

```
        //console.log(jqXHR);
      }
  });
}
/*=========================================================*/
var Range = function (xPos, yStart, ySpan, rColor) {
  this.xPos = xPos;
  this.yStart = parseInt(yStart);
  this.ySpan = ySpan;
  this.yCentre = this.yStart + parseInt(ySpan / 2);
  this.Color = rColor;
  theRanges.push(this);
  this.uid = theRanges.length - 1;
  // add reference in column index:
  if (typeof(theColumns[xPos]) != 'undefined'){
    theColumns[xPos]['theRanges'][this.uid] = this.uid;
  }
  theGlobals['minRangeSpan'] = Math.min(theGlobals['minRangeSpan'], this.ySpan);
  theGlobals['maxRangeSpan'] = Math.max(theGlobals['maxRangeSpan'], this.ySpan);
  this.theContinuity = null; //id number of continuity.
}
Range.prototype.findOverlapWith = function(anotherRange){
  if (this.Color != anotherRange.Color){ return; }
  var yStartA = this.yStart;
  var ySpanA = this.ySpan;
  var yStartB = anotherRange.yStart;
  var ySpanB = anotherRange.ySpan;
  var pixelsInCommon = getPixelOverlap(yStartA, ySpanA, yStartB, ySpanB);
  if (pixelsInCommon.length > 0){
    var yStart = pixelsInCommon.reduce(function(a, b, i, pixelsInCommon) {return Math.min(a,b)});
    var ySpan = pixelsInCommon.length;
    new Overlap(this.uid, anotherRange.uid, yStart, ySpan);
  }
}
function getPixelOverlap(yStartA, ySpanA, yStartB, ySpanB){
  var pixelsInCommon = [];
  for (k = yStartA; k < yStartA+ySpanA ; k++){
    for (l = yStartB; l< yStartB+ySpanB; l++){
      if (k == l) { pixelsInCommon.push(k); }
    }
  }
  return pixelsInCommon;
}
/*=======OVERLAP====================*/
var Overlap = function (rangeOneId, rangeTwoId, yStart, ySpan) {
  this.rangeOneId = rangeOneId;
  this.rangeTwoId = rangeTwoId;
  this.xPosA = theRanges[rangeOneId].xPos;
  this.xPosB = theRanges[rangeTwoId].xPos;
  this.yStart = yStart;
  this.ySpan = ySpan;
  this.yCentre = yStart + parseInt(ySpan / 2);
  this.isActive = true; // until proven otherwise
  theOverlaps.push(this);
  this.uid = theOverlaps.length - 1;
  // add reference in column index:
```

```
  theColumns[this.xPosA]['theOverlaps'][this.uid] = this.uid;
}
Overlap.prototype.resolveContestsWith = function(anotherOverlap){
  if (this.rangeOneId == anotherOverlap.rangeOneId){
    thisY = parseInt(theRanges[this.rangeTwoId].ySpan);
    thatY = parseInt(theRanges[anotherOverlap.rangeTwoId].ySpan);
    /*alert( 'resolve: o'+this.uid + 'r1='+this.rangeOneId+' and o'+ anotherOverlap.uid+'r1 ='
+anotherOverlap.rangeOneId+'   thisY='+thisY+' thatY='+thatY);*/
    if (thisY >= thatY){
      //alert( 'Overlap '+this.uid+' and '+anotherOverlap.uid+'have range '+this.rangeOneId+'in common.'+ 'Range'+
this.rangeOneId + ' overlaps with Range:' + this.rangeTwoId + ' via Overlap '+this.uid + 'Range' +
anotherOverlap.rangeOneId +' overlaps with Range:' +anotherOverlap.rangeTwoId + 'via Overlap '
+anotherOverlap.uid + 'Range'+ this.rangeTwoId + ' ySpan is. '+ thisY+ 'and Range' + anotherOverlap.rangeTwoId
+ 'ySpan is ' + thatY + 'therefore setting theOverlaps['+anotherOverlap.uid+'].isActive =false');
      theWeakOverlaps.push(anotherOverlap.uid);
    }
  }
  else if (this.rangeTwoId == anotherOverlap.rangeTwoId){
    thisY = theRanges[this.rangeOneId].ySpan;
    thatY = theRanges[anotherOverlap.rangeOneId].ySpan;
    //alert( 'resolve: o'+this.uid + 'r2='+this.rangeTwoId+' and o'+ anotherOverlap.uid+'r2 ='
+anotherOverlap.rangeTwoId+'   thisY='+thisY+' thatY='+thatY);
    if (thisY >= thatY){
      theWeakOverlaps.push(anotherOverlap.uid);
    }
  }
}
/*======CONTINUITY===========================*/
var Continuity = function (rangeId) {
  this.data = {}; // you could remove this but its tricky: you do need to loop through it later
  this.sd = {};
  theContinuities.push(this);
  this.uid = theContinuities.length - 1;
  if (theRanges[rangeId]['Color'] == 'd'){
    this.isYellow = false; this.isDark = true;
    var r=0 + Math.floor(Math.random()*80);
    var g=0 + Math.floor(Math.random()*80);
    var b=0 + Math.floor(Math.random()*80);
  }
  else if (theRanges[rangeId]['Color'] == 'y'){
    this.isYellow = true; this.isDark = false;
    var r=255 - Math.floor(Math.random()*80);
    var g=200 - Math.floor(Math.random()*80);
    var b=0 + Math.floor(Math.random()*80);
  }
  this.c = [r,g,b];
  this.addRange(rangeId);
}
Continuity.prototype.addRange = function(rangeId){
  var xPos = theRanges[rangeId].xPos;
  theRanges[rangeId].theContinuity = this.uid;
  theColumns[xPos]['theContinuities'][this.uid] = this.uid;
  var slope = 0;
  var accel = 0;
  if (typeof(this.data[xPos-1]) != 'undefined'){
    var slope = (theRanges[rangeId]['yCentre'] - this.data[xPos-1][3]) * -1;
```

```
    gMax('maxSlope',slope);
    gMin('minSlope', slope);
    var accel = Math.abs(slope - this.data[xPos-1][4]);
    gMax('maxAccel',accel);
    gMin('minAccel', accel);
   }
   this.data[xPos] = [
    rangeId,
    theRanges[rangeId]['yStart'],
    theRanges[rangeId]['ySpan'],
    theRanges[rangeId]['yCentre'],
    slope,
    accel
   ];
   /*
   rangeId, yStart, ySpan,   yCentre
   slope, accel, smoothSlope, smoothslopeABS
   */
   //how can you get overlap span in here?
}
Continuity.prototype.compute = function(i, method){
   var theData = [];
   for (xPos in this.data){ theData.push(this.data[xPos][i]); }
   if (method == 'stdDev'){
    this.sd[i] = stdDev(theData);
    theGlobals['sdMax'][i] = Math.max(theGlobals['sdMax'][i], this.sd[i]);
   }
}
// gMax and gMin update global Max and Min for a given value.
function gMax(theVar, theValue){
   if (theVar != null){
    theGlobals[theVar] = Math.max(theGlobals[theVar], theValue);
   }
}
function gMin(theVar, theValue){
   if (theVar != null){
    theGlobals[theVar] = Math.min(theGlobals[theVar], theValue);
   }
}
// display progress bar
function setProgress(percent, message){
   $('#progressBar').css({'width':percent+'%'});
   $('#progressUnits').html("%");
   $('#progressPercent').html(percent);
   $('#progressAction').html(message);
}
function setupIndeces(){
   for (var i = 0; i < theCanvasWidth; i++ ){
    theColumns[i] = {};
    theColumns[i]['theRanges'] = [];
    theColumns[i]['theOverlaps'] = [];
    theColumns[i]['theActiveOverlaps'] = [];
    theColumns[i]['theContinuities'] = [];
    theSignals[i]={}; // this is used for signals
    theSignals[i]['cc'] = 0; //continuity count
    theSignals[i]['ccy'] = 0; //continuity count yellow
```

```
    theSignals[i]['ccd'] = 0; //continuity count dark
    /*
    // this is the same thing as thicness.
    theSignals[i]['pc'] = 0; // pixel count
    theSignals[i]['pcy'] = 0; // pixel count yellow
    theSignals[i]['pcd'] = 0; // pixel count dark
    */
    theSignals[i]['l'] = 0; // lightness
    theSignals[i]['st'] = 0 ; // smooth thickness.
    theSignals[i]['sty'] = 0 ; // smooth thickness yellow
    theSignals[i]['std'] = 0 ; // smooth thickness dark
    theSignals[i]['rs'] = 0 ; // reach sum
    theSignals[i]['ysd'] = [0] ; // yposition standard deviation
    theSignals[i]['sl'] = [0] ; // slope
    theSignals[i]['pk'] = [0] ; // peakishness
    theSignals[i]['tr'] = [0] ; // troughishness
  }
}
/*=========IMAGE PROCESSING ====================*/
function findRanges(){
  var i = currentColumn; // starts at 0;
  var columnRanges = [];
  var darkRanges = [];
  var yellowRanges = [];
  var thePixelData = theContext.getImageData(i, 0, 1, 1080).data;
  var yStartDark = null;
  var yStartYellow = null;
  var theRGB = {r:0,g:0,b:0}; // tally for average
  for (var j=0;j<thePixelData.length;j+=4) {
    theRGB.r += thePixelData[j];
    theRGB.g += thePixelData[j+1];
    theRGB.b += thePixelData[j+2];
    var someHSL = rgbToHsl(thePixelData[j], thePixelData[j+1], thePixelData[j+2]);
    if (isDark(someHSL)){
      if (yStartDark == null){  yStartDark = j/4;    }
      if (typeof(darkRanges[yStartDark]) == 'undefined'){
        darkRanges[yStartDark] = 0;
      }
      darkRanges[yStartDark] +=1;
    }
    else{ yStartDark = null; }
    if (isYellow(someHSL)){
      if (yStartYellow == null){  yStartYellow = j/4;    }
      if (typeof(yellowRanges[yStartYellow]) == 'undefined'){
        yellowRanges[yStartYellow] = 0;
      }
      yellowRanges[yStartYellow] +=1;
    }
    else{ yStartYellow = null; }
  }
  for (yStart in darkRanges){
    if (darkRanges[yStart] > minRangeSpan){
      if (darkRanges[yStart] < maxRangeSpan){
        new Range(i, yStart, darkRanges[yStart], 'd');
      }
    }
```

```
  }
  for (yStart in yellowRanges){
   if (yellowRanges[yStart] > minRangeSpan){
    if (yellowRanges[yStart] < maxRangeSpan){
     new Range(i, yStart, yellowRanges[yStart], 'y');
    }
   }
  }
  /*get average RGB*/
  theRGB.r = ~~(theRGB.r/1080);
  theRGB.g = ~~(theRGB.g/1080);
  theRGB.b = ~~(theRGB.b/1080); // floor values
  var theHSL = rgbToHsl(theRGB.r, theRGB.g, theRGB.b); // average HSL
  theSignals[i]['l']=theHSL[2];
  var thePercent =  parseInt( (i / theCanvasWidth) * 80 + 5);
  if (thePercent > 10){ var theMessage = "Finding Ranges"; }
  else{ var theMessage = "Wait"; }
  setProgress(thePercent, theMessage);
  if (cachedContinuities != null
   && cachedSignals != null
   && cachedGlobals != null){
   var r = confirm("Use Cached Data?");
   if (r == true) {
    theContinuities = cachedContinuities;
    theSignals = cachedSignals;
    theGlobals = cachedGlobals;
    cacheUsed = true;
    setProgress(98, "Finishing Up");
    setTimeout(finishUp, 2);
   } else {
    cachedContinuities = null;
    cachedSignals = null;
   }
  }
  // if no continuitiesare loaded (e.g from  cache), keep going
  if (theContinuities.length == 0){
   if (i<(theCanvasWidth-1)){
    currentColumn ++;
    setTimeout(findRanges, 2);
   }
   else{
    setProgress(86, "Finding Overlaps");
    setTimeout(findOverlaps, 2);
   }
  }
}
function findOverlaps(){
 for (var i = 0; i < theCanvasWidth; i++ ){
  for (rangeA in theColumns[i]['theRanges']){
   try{
    for(rangeB in theColumns[i+1]['theRanges']){
     theRanges[rangeA].findOverlapWith(theRanges[rangeB]);
    }
   }catch(e){}
  }
 }
```

```
  setProgress(90, "Resolving Contests");
  setTimeout(resolveContests, 2);
}
function resolveContests(){
  for (var i = 0; i < theCanvasWidth; i++ ){
    var overlapsA = theColumns[i]['theOverlaps'];
    var overlapsB = theColumns[i]['theOverlaps'];
    for (overlapA in overlapsA){
      for(overlapB in overlapsB){
        if (overlapA != overlapB){
          theOverlaps[overlapA].resolveContestsWith(theOverlaps[overlapB]);
        }
      }
    }
  }
  for (var w in theWeakOverlaps){
    theOverlaps[theWeakOverlaps[w]].isActive = false;
  }
  setProgress(95, "Finding Continuities");
  setTimeout(findContinuities, 2);
}
function findContinuities(){
  for (v in theOverlaps){
    if (theOverlaps[v]['isActive']==true){
      var ct;
      var ctId = theRanges[theOverlaps[v].rangeOneId].theContinuity;
      if (ctId == null){
        ct = new Continuity(theOverlaps[v].rangeOneId);
      }
      else{
        ct = theContinuities[ctId];
      }
      ct.addRange(theOverlaps[v].rangeTwoId);
    }
  }
  var deleteThese = [];
  for (w in theContinuities){
    var keyCount = Object.keys(theContinuities[w].data).length;
    if (keyCount < minContinuityXSpan){ deleteThese.push(w); }
  }
  for (x in deleteThese){
    theContinuities.splice(deleteThese[x]-x, 1);
  }
  setProgress(97, "Calculating Slopes");
  setTimeout(calculateSlopes, 2);
}
function calculateSlopes(){
  for (w in theContinuities){
    for (xPos in theContinuities[w].data){
      theSignals[xPos]['ysd'].push(theContinuities[w].data[xPos][1]); //ypos standard dev

      // mark the extent to which subsequent ranges overlap.
      var nearbyYSpans = [theContinuities[w].data[xPos][2]];
      var nearbySlopes = [theContinuities[w].data[xPos][4]];
      var pSlopes = []; //preceding
      var fSlopes = []; // following
```

```
    for (i=1; i < 50; i++){
      try{
        var fSlope = theContinuities[w].data[parseInt(xPos)+i][4];
        nearbySlopes.push(fSlope);
        fSlopes.push(fSlope);
      } catch(e){}
      try{
        var pSlope = theContinuities[w].data[parseInt(xPos)-i][4];
        nearbySlopes.push(pSlope);
        pSlopes.push(pSlope);
      } catch(e){}
      try{ nearbyYSpans.push(theContinuities[w].data[parseInt(xPos)+i][2]);
      } catch(e){}
      try{ nearbyYSpans.push(theContinuities[w].data[parseInt(xPos)-i][2]);
      } catch(e){}
    }
    // todo: standard deviaiton in yPosition?
    // todo: smooth yPosition?
    var troughishness = 0;
    var peakishness = 0;
    var pSlopesSum = arrSum(pSlopes);
    var fSlopesSum = arrSum(fSlopes);
    var slopeDiff = Math.abs(pSlopesSum) + Math.abs(fSlopesSum)
    if (pSlopesSum > 0 && fSlopesSum < 0){
      var peakishness = slopeDiff;
    }
    if (pSlopesSum < 0 && fSlopesSum > 0){
      var troughishness = slopeDiff;
    }
    theSignals[xPos]['pk'].push(peakishness);
    theSignals[xPos]['tr'].push(troughishness);


    var smoothThickness =  average(nearbyYSpans);
    var smoothSlope = average(nearbySlopes);
    var smoothSlopeABS = nearbySlopes.reduce(function(sum, a) { return Math.abs(sum) + Math.abs(a)
},0)/(nearbySlopes.length||1);

    theSignals[xPos]['sl'].push(Math.min(smoothSlopeABS, 30));
    theContinuities[w].data[xPos].push(smoothSlope);
    theContinuities[w].data[xPos].push(smoothSlopeABS);
    theContinuities[w].data[xPos].push(smoothThickness);
    theSignals[xPos]['st'] += smoothThickness;
    if (theContinuities[w].isYellow){
      theSignals[xPos]['sty'] += smoothThickness;
    }
    if (theContinuities[w].isDark){
      theSignals[xPos]['std'] += smoothThickness;
    }
    var reachPixels = 0;
    var keyCount = Object.keys(theContinuities[w].data).length;
    for (var i = 1; i < keyCount; i++){
      if (typeof(theContinuities[w].data[parseInt(xPos)+i]) !="undefined"){
        var yStartA = theContinuities[w].data[xPos][1];
        var ySpanA = theContinuities[w].data[xPos][2];
        var yStartB = theContinuities[w].data[parseInt(xPos)+i][1];
```

```
        var ySpanB = theContinuities[w].data[parseInt(xPos)+i][2];
        var theOverlap = getPixelOverlap(yStartA, ySpanA, yStartB, ySpanB);
        if (theOverlap.length == 0) { break; }
        reachPixels++;
       }
      }
      gMax('maxReach',reachPixels);
      theContinuities[w].data[xPos].push(reachPixels);
      theSignals[xPos]['rs'] += reachPixels; //increment reach sum
    }
    for (i = 0; i < 9; i++){
      //calculate standard deviations for all signals
      theContinuities[w].compute(i, 'stdDev');
    }
  }
  // i need the average standard deviation of yPosition.
  setProgress(98, "Calculating Signals");
  setTimeout(calculateSignals, 2);
}
function calculateSignals(){
  // these signals apply globally (vs in one continuity only).
  for (w in theContinuities){
    for (xPos in theContinuities[w].data){
      theSignals[xPos]['cc']++;
      if (theContinuities[w].isYellow)  theSignals[xPos]['ccy']++;
      if (theContinuities[w].isDark)  theSignals[xPos]['ccd']++;
    }
  }
  // store max ContinuityCount and max smoothThickness
  for (xPos in theSignals){
    theSignals[xPos]['ysd'] = stdDev(theSignals[xPos]['ysd']);
    theSignals[xPos]['sl'] = average(theSignals[xPos]['sl']);
    theSignals[xPos]['pk'] = average(theSignals[xPos]['pk']);
    theSignals[xPos]['tr'] = average(theSignals[xPos]['tr']);
    theGlobals['pkMax'] = Math.max(theSignals[xPos]['pk'], theGlobals['pkMax']);
    theGlobals['trMax'] = Math.max(theSignals[xPos]['tr'], theGlobals['trMax']);
    theGlobals['slMax'] = Math.max(theSignals[xPos]['sl'], theGlobals['slMax']);
    theGlobals['ysdMax'] = Math.max(theSignals[xPos]['ysd'], theGlobals['ysdMax']);
    theGlobals['ccMax'] = Math.max(theSignals[xPos]['cc'], theGlobals['ccMax']);
    theGlobals['ccyMax'] = Math.max(theSignals[xPos]['ccy'], theGlobals['ccyMax']);
    theGlobals['ccdMax'] = Math.max(theSignals[xPos]['ccd'], theGlobals['ccdMax']);
    theGlobals['stMax'] = Math.max(theSignals[xPos]['st'], theGlobals['stMax']);
    theGlobals['styMax'] = Math.max(theSignals[xPos]['sty'], theGlobals['styMax']);
    theGlobals['stdMax'] = Math.max(theSignals[xPos]['std'], theGlobals['stdMax']);
    theGlobals['rsMax'] = Math.max(theSignals[xPos]['rs'], theGlobals['rsMax']);
    // no need  lightness Max, because it's already [0,1]
  }
  // TODO: calculate standard deviation for global signals.
  setProgress(99, "Finishing Up");
  setTimeout(finishUp, 2);
}
/*============SETUP CHOIR====================*/
var VirtualSinger = function (lowestNote,highestNote) {
  if (lowestNote==null) lowestNote = 40;
  if (highestNote==null) highestNote = 82;
  this.lowestNote = lowestNote;
```

```
  this.highestNote = highestNote;
  this.singing = false;
  this.lastRelease = 0; // a useful timestamp
  this.dynamicsSet = ['pp','p','mp','mf','f','ff'];
  this.pitchWobble = randomElement([-3,-2,-1,0,1,2,3]);
// boolean singing
// dont make a new note if already singing
// set it to ready a bit later than the minimum necessary.
  this.sine = ac.createOscillator();
  this.oscGain = ac.createGain();
  this.wavGain = ac.createGain();
  this.compressor = ac.createDynamicsCompressor();
  this.sine.type = 'sine';
  this.sine.connect(this.oscGain);
  this.oscGain.gain.setValueAtTime(0,ac.currentTime);
  this.compressor.threshold.value = -3;
  this.compressor.ratio.value = 10;
  this.compressor.attack = 0.005;
  this.compressor.release = 0.005;
  this.oscGain.connect(this.compressor);
  this.wavGain.connect(this.compressor);
  this.compressor.connect(ac.destination);
  this.sine.start();
  // sometimes you just need to restart the browser!!
}
// start singing.
VirtualSinger.prototype.start = function () {
  var closure = this;
  setTimeout(function() {
    closure.callback();
  },70+Math.floor(Math.random()*130));
  // singers shouldn't all start at the same time.
  // is this to create a musical texture?
}
VirtualSinger.prototype.callback = function () {
  if(this.wantANote() ){
    var note = this.getANote(this.lastNote);
    this.singANote(note);
    this.lastNote = note;
  }
  this.start(); // loop start > callback > start > etc.
}

VirtualSinger.prototype.wantANote = function(){
  //alert(isSinging + ' ' + this.lastRelease + ' ' +ac.currentTime);
  if (isSinging && !this.singing){
    if (this.lastRelease == 0 && ac.currentTime == 0) return true;
    if (this.lastRelease < ac.currentTime) return true;
  }
  return false;
}
VirtualSinger.prototype.singANote = function(x) {
  if(x['isPitched'] == false){
    this.source = ac.createBufferSource();
    console.log(theSamples);
    this.source.buffer = theSamples[x['hmm']].buffer;
```

```
    this.source.playbackRate.value = 1; // you could alter this by pitch.
    this.source.connect(this.wavGain);
    this.source.start();
    var now = ac.currentTime;
    this.wavGain.gain.setValueAtTime(0,now);
    this.wavGain.gain.linearRampToValueAtTime(db(-6),now+0.5);
    this.wavGain.gain.linearRampToValueAtTime(0,now+1.0);
    this.lastRelease = now+1.0;
    var closure = this;
    setTimeout(function() {
      closure.source.disconnect(closure.gain);
    },1500);
  }
  else{
    var duration = x['dur'];
    // override duration if staccato.
    if (x['isStaccato']){  duration = 0.25; }
    var amp = db(x['amp']);

    // override amplitude if accented
    if (x['isAccented']){  amp = db(-6); }
    if (x['midiNote'] =='rest'){ return;  }
    var freq =  midiToFreq(x['midiNote']) + this.pitchWobble;
  //console.log(x['midiNote']+' '+freq);
    var attack = 0.1;
    var decay = 0.1;
    var now = ac.currentTime;
    var sustainFrom = now + attack;
    var sustainUntil = now + duration - decay;
    var stopPoint = now + duration;
    this.sine.frequency.linearRampToValueAtTime(freq, sustainFrom);
    //this.sine.frequency.setValueAtTime(freq,now);
    this.oscGain.gain.setValueAtTime(0,now);
    this.oscGain.gain.linearRampToValueAtTime(amp, sustainFrom);
    this.oscGain.gain.linearRampToValueAtTime(amp, sustainUntil);
    this.oscGain.gain.linearRampToValueAtTime(0,stopPoint);
    this.lastRelease = stopPoint;
  }


}
VirtualSinger.prototype.canSing = function(midiNote){
  if ( this.highestNote >= midiNote && this.lowestNote <= midiNote ) return true;
  return false;
}
VirtualSinger.prototype.isBass = function () {
  if ( (this.highestNote - 60) < (60 - this.lowestNote) ) return true;
  return false;
}
VirtualSinger.prototype.isSoprano = function () {
  if ( (this.highestNote - 60) > (60 - this.lowestNote) ) return true;
  return false;
}
VirtualSinger.prototype.getANote = function (lastNote) {
  var isAccented = false;
  var bs = theScore['bassBoost'];
```

```
if ( this.isBass() ) {
  if (bs > 0.75) {  isAccented = true; }
  else if (bs > 0.25) {
    if (Math.random() > bs){   isAccented = true;  }
  }
}
var ss = theScore['sopranoBoost'];
if ( this.isSoprano() ) {
  if (ss > 0.75) {  isAccented = true; }
  else if (ss > 0.25) {
    if (Math.random() > ss){   isAccented = true;  }
  }
}
var midiNotes = theScore['concreteChord'];
//midiNotes is full of undefineds and NaNs?
var goodNotes = [];
for (aNote in midiNotes){
  if ( this.canSing(midiNotes[aNote]) ){
    goodNotes.push(midiNotes[aNote]);
  }
}
if (goodNotes.length > 0){
  var midiNote = goodNotes[Math.floor(Math.random()*goodNotes.length)];
}
else{
  midiNote = 'rest';
}
// are you really going to pick a note randomly?
// yes, for now.

// notes are between 0.5 and 4 seconds
var duration = theScore['duration'].map(0,1,1,4);
var dynamicIndex = parseInt(theScore['dynamic'].map(0,1,0,6.1));
var dynamic = this.dynamicsSet[dynamicIndex];
var amplitude = parseInt(theScore['dynamic'].map(0,1,-48,-6));
var isPitched = true;
var hmm = '';
if (theScore['brightness'] < 0.25){
    isPitched = false;
    hmm = 'shh';
}
if ( Math.random() <  theScore['growl'] ){
  isPitched = false;
  hmm = 'rrr';
}

var isStaccato = false;
if (theScore['punctuation'] > 0.5){ isStaccato = true; }
return {
  'midiNote':midiNote,
  'dur':duration,
  'amp':amplitude,
  'dynamic':dynamic,
  'isStaccato': isStaccato,
  'isAccented': isAccented,
  'isPitched' : isPitched,
```

```
    'lyric' : theScore['lyric'],
    'hmm': hmm
   };
}
var Choir = function () {
  this.singers = new Array(15);  //or, 50, or 100
  for(var x=0;x<this.singers.length;x++) {
    var lowNote = Math.floor(Math.random()*24)+36;
    var range = 12 + Math.floor(Math.random()*12);
    this.singers[x] = new VirtualSinger(lowNote,lowNote+range);
  }
}
Choir.prototype.start = function () {
  for(var x=0;x<this.singers.length;x++) {
    this.singers[x].start();
  }
}
SampleBuffer = function(url) {
  var request = new XMLHttpRequest();
  request.open('GET',url,true);
  request.responseType = 'arraybuffer';
  var closure = this; // a closure is necessary for...
  request.onload = function() {
    ac.decodeAudioData(request.response, function(x) {
      console.log("sample " + url + "loaded");
      closure.buffer = x;
    });
  };
  request.send();
}
function chordCycle(){
    // calculate interval for next cycle
    var theHertz = theScore['harmonicPace'].map(0,1,0.1, 1);
    var theInterval = (1/theHertz)*1000;  // ms
    // Pick a chord
    var numChords = Object.keys(chordSet).length;
    var chordNumber = parseInt(theScore['chordDissonance'].map(0,1,0,(numChords+0.1) ));
    var baseChord = chordSet[chordNumber];

    // expand octaves
    var fullChord = [];
    for (someNote in baseChord){
      fullChord.push(baseChord[someNote] - 24);
      fullChord.push(baseChord[someNote] - 12);
      fullChord.push(baseChord[someNote]);
      fullChord.push(baseChord[someNote] + 12);
      fullChord.push(baseChord[someNote] + 24);
    }
    // poll connected singers here to get a sensible range.
    var maxNote = 60 + parseInt(theScore['pitchSpreadUpper'].map(0,1,4,22));
    var minNote = 60 - parseInt(theScore['growl'].map(0,1,4,22));

    // minimum and maximum midi notes
    for (aKey in fullChord){
      if (fullChord[aKey] > maxNote ){
        //console.log('deleted' + fullChord[aKey]);
```

```
      fullChord.splice(aKey, 1);
    }
    if (fullChord[aKey] < minNote ){
      //console.log('deleted' + fullChord[aKey]);
      fullChord.splice(aKey, 1);
    }
  }
  fullChord.sort().reverse();

  // randomly reduce fullChord to match pitchCount
  // when polyphony is too low this makes for empty chords.
  var pitchCount = parseInt(theScore['polyphony'].map(0,1,2,24));
  var concreteChord = getRandomElements(fullChord, pitchCount);
  theScore['concreteChord'] = concreteChord;
  var wordIndex = Math.floor(theScore['progress'].map(0,1,0, (Object.keys(wordSet).length - 0.01)));
  var theWord = wordSet[wordIndex];
  theScore['lyric']=theWord;
  var theMinNote = Math.min.apply(Math, concreteChord);
  var theMaxNote = Math.max.apply(Math, concreteChord);
    // STRUCTURE
    // chordCycle() updates as per harmonicPace:
      // theScore[concreteChord]
      // theScore[lyric]
    // draw() and sing() update as per xPos:
      // sopranoBoost, bassBoost, punctuation,
      // dynamic, duration, polyphony, etc.
    // broadcast() will send theScore via apert continuously
      // (for now local singers reference it directly.)


  // repeat to fill 1080p?
  /*
  while (fullChord.length < 36){
    fullChord = fullChord.concat(fullChord);  //repeat
  }
  while (fullChord.length > 36){
    fullChord.splice(fullChord.length -1, 1); // remove last element
  }
  $('#theNow div').remove();
  for (someNote in fullChord){
    var cc = parseInt(fullChord[someNote].map(theMinNote, theMaxNote, 0, 255));
    $('<div id="_'+ someNote +'" class="square" style="'+
        'background-color:rgba('+cc+','+cc+','+cc+',0.5);'+
      '">'+ fullChord[someNote] +'</div>').appendTo('#theNow');
  }
  */
  setTimeout(  chordCycle, theInterval );

}
/*
// get the average of a value for all continuities at a given xPos
function contAvg(xPos, array, index){
  theValues = [];
  for (w in theContinuities){
    try{ theValue.push(theContinuities[w][array][xPos][index]);
    }
```

```
    catch(e){}
  }
  return average(theValue);
}
// get the sum of a value for all continuities at a given xPos
function continuitySum(xPos, array, index){
  theValue = 0;
  for (w in theContinuities){
    try{ theValue += theContinuities[w][array][xPos][index];   }
    catch(e){}
  }
  return theValue;
}
// get the standard deviation of a value for all continuities at a given xPos
function continuityStdDev(xPos, array, index){
  theValues = [];
  for (w in theContinuities){
    try{ theValue.push(theContinuities[w][array][xPos][index]);
    }
    catch(e){}
  }
  return stdDev(theValue);
}*/
/*=============== DRAW and SING ====================*/
// draw visualisations
function draw(xPos){
  for (w in theContinuities){
    if (typeof( theContinuities[w].data[xPos]) != 'undefined' ){
      // The Range Itself
      var rgb = theContinuities[w].c;
      theContext.fillStyle = 'rgba('+rgb[0]+','+rgb[1]+','+rgb[2]+',0.5)';
      theContext.fillRect( xPos, theContinuities[w].data[xPos][1], 1, theContinuities[w].data[xPos][2] );
      // TODO reach of shade areas?
      // TODO: sum of thicknesses of all continuities?
      // (a.k.a.  sum of all in-range pixels)
      /* TODO lineExpression */
      // like lineWeight, but account for
      // diminished thickness of larger smoothSlope
      // Thickness
      //var theThickness = theContinuities[w].data[xPos][2];
      //var thickYPos = theThickness.map(theGlobals['minRangeSpan'], theGlobals['maxRangeSpan'],900, 700);
      //theContext.fillRect( xPos, thickYPos , 3, 3);
      /*
      // Reach
      var theReach = theContinuities[w].data[xPos][9];
      var reachYPos = theReach.map(0, theGlobals['maxReach'],900, 700);
      theContext.fillRect( xPos, reachYPos , 3, 3);
      // Smooth thickness.
      var theSThickness = theContinuities[w].data[xPos][8];
      var thickSYPos = theSThickness.map(theGlobals['minRangeSpan'], theGlobals['maxRangeSpan'],900, 700);
      theContext.fillRect( xPos, thickSYPos , 3, 3);
      // Standard Deviation of yPosition
      var sdyPos = theContinuities[w].sd[2].map(0, theGlobals['sdMax'][2], 800, 600);
      theContext.fillRect( xPos, sdyPos, 3, 3 );
      //Smooth Slope
      var smoothSlope = theContinuities[w].data[xPos][6];
```

```
    if (smoothSlope > 0){ var yPoint = smoothSlope.map(0, theGlobals['maxSlope'], 540, 0); }
    if (smoothSlope < 0){ var yPoint = smoothSlope.map(0, (theGlobals['maxSlope']*-1), 540, 1080); }
    if (smoothSlope == 0){ var yPoint = 540; }
    theContext.fillRect( parseInt(xPos), yPoint, 3, 3 );
    try{
      var smoothSlopeABS = theContinuities[w].data[xPos][7];
      var yPoint = smoothSlopeABS.map(0, theGlobals['maxSlope'], 1080, 54);
      theContext.fillRect( parseInt(xPos), yPoint, 5, 5 );
    }catch(e){}
    */
  }
 }

 /* Draw parameters */
 var theParams = Object.keys(theScore);
 var graphHeight = parseInt(1080 / theParams.length);
 for (ai in theParams){
  var theParam = theParams[ai];
  var theValue = theScore[theParam];
  var ayMin = ai*graphHeight;
  var ayMax = ai*graphHeight + graphHeight;
  var ayMid = parseInt(ai*graphHeight + (graphHeight / 2));
  if (typeof(theValue) == 'number' && !isNaN(theValue)){
   var ayPos = theValue.map(0, 1, ayMax, ayMin);
   if (typeof(ayPos) != 'undefined'){
    theContext.fillStyle = 'rgba(255,255,255,0.2)';
    theContext.beginPath();
    theContext.arc(xPos, ayPos, 2, 0, 2 * Math.PI, false);
    theContext.fill();
    //theContext.fillRect( xPos, ayPos, 3, 3 );
   }
  }
  $('#'+theParam).remove();
  $('<div id="'+theParam+'">'+theParam+' '+theValue+'</div>').css({
    'position':'absolute',
    'z-index':'20000',
    'top':ayMid+'px'
  }).appendTo('body');
 }
 lastDrawnColumn = xPos;
}
//  update theScore for column xPos
// broadcast theScore to connected devices.
function sing(xPos){
  // SIGNALS
  var theCount = theSignals[xPos]['cc']; //continuity count
  var theCountYellow = theSignals[xPos]['ccy']; //continuity count yellow
  var theCountDark = theSignals[xPos]['ccd']; //continuity count dark
  var theLightness = theSignals[xPos]['l'];
  var theST =   theSignals[xPos]['st']; // smooth thickness
  var theSTY =   theSignals[xPos]['sty']; // smooth thickness yellow
  var theSTD =   theSignals[xPos]['std']; // smooth thickness dark
  var theRS =   theSignals[xPos]['rs']; // reach sum
  var theYSD =  theSignals[xPos]['ysd']; // standard deviation of y position
  var theSlope =  theSignals[xPos]['sl']; // avg slope of all continuities
  var thePeakishness =  theSignals[xPos]['pk']; // avg peakishness
```

```
    var theTroughishness =  theSignals[xPos]['tr']; // avg Troughishness
    theScore['progress'] = xPos.map(0, theCanvasWidth, 0,1);
    theScore['harmonicPace'] = theSlope.map(0, theGlobals['slMax'], 0,1);
    theScore['polyphony'] = theCount.map(0,theGlobals['ccMax'], 0, 1);
    theScore['chordDissonance'] = theST.map(0,theGlobals['stMax'], 0, 1);
    // we used to have pitchSpread, but it is redundantly like chordDissonance
    theScore['pitchSpreadUpper'] = theSTY.map(0,theGlobals['styMax'], 0, 1); // sum of yellow Ranges
    theScore['growl'] = theSTD.map(0,theGlobals['stdMax'], 0, 1);
    //theScore['sopranoBoost'] = theSTY.map(0,theGlobals['styMax'], 0, 1);
    theScore['sopranoBoost'] = thePeakishness.map(0,theGlobals['pkMax'], 0, 1);
    //theScore['bassBoost'] = theSTD.map(0,theGlobals['stdMax'], 0, 1);
    theScore['bassBoost'] = theTroughishness.map(0,theGlobals['trMax'], 0, 1);
    theScore['punctuation'] = theYSD.map(0,theGlobals['ysdMax'],0,1);
    theScore['dynamic'] = theST.map(0,theGlobals['stMax'], 0, 1);
    theScore['duration'] = theRS.map(0,theGlobals['rsMax'], 0, 1);
    theScore['brightness'] = theLightness;
    apertMemorySetPublic( 'theScore', theScore );
}
function finishUp(){
  setProgress(100, "Done!");
  $('#progressBar').fadeOut(2000);
  if (!cacheUsed){    saveData();   }
}

// traverse the image
function animate(){
  if (isSinging == false) return;
  var theInterval = 42;         // 42ms translates into ~24 fps.
  var pixelsPerFrame = pixelsPerMinute / 60 / 1000 * theInterval;
  //var pixelsPerFrame = beatsPerMinute / 60 / 1000 * pixelsPerBeat * theInterval;
  pixelsTraversed += pixelsPerFrame;
  if ( (pixelsTraversed + 1920) > theCanvasWidth ){
    pixelsTraversed = 0;
    isSinging = false;
  }
  var newMargin = pixelsTraversed * -1;
  $('canvas').css('margin-left', newMargin + 'px');
  var theNow = ( pixelsTraversed / (theCanvasWidth - 1920) ) * 1920;
  $('#theNow').css('left', parseFloat(theNow) + 'px');
  $('#theShade').css('width', parseInt(1920 - parseFloat(theNow))+'px');
  for (i = lastDrawnColumn; i < parseInt(theNow+pixelsTraversed); i++){
    //alert(i+' ' +theNow+' '+pixelsTraversed+' '+newMargin);
    draw(i);
    sing(i);
  }
  setTimeout(animate, theInterval );
}
function isDark(someHSL){
  if (someHSL[2] < 0.25){ return true; }
    return false;
}
function isYellow(someHSL){
  // yellow is between 35 and 55 on a scale of 360
  // this maps onto ie 0.09 and 0.152 on a 0-1 scale
  // cconserative: if (someHSL[0] > 0.0972 && someHSL[0] < 0.152777
  if (someHSL[0] > 0.0962 && someHSL[0] < 0.162777
```

```
    && someHSL[1] > 0.35
    && someHSL[2] > 0.35){ return true; }
  return false;
}
function midiToFreq (midi, tuning) {
 return Math.pow(2, (midi - 69) / 12) * (tuning || 440)
}
//calculate amplitude given dB
db = function(x) { return Math.pow(10,x/20); }
// Converts an RGB color value to HSL.
// [r,g,b] in set [0, 255],  [h,s,l] in set [0,1]
function rgbToHsl(r, g, b){
  r /= 255, g /= 255, b /= 255;
  var max = Math.max(r, g, b), min = Math.min(r, g, b);
  var h, s, l = (max + min) / 2;
  if(max == min){  h = s = 0;  }
  else{
    var d = max - min;
    s = l > 0.5 ? d / (2 - max - min) : d / (max + min);
    switch(max){
      case r: h = (g - b) / d + (g < b ? 6 : 0); break;
      case g: h = (b - r) / d + 2; break;
      case b: h = (r - g) / d + 4; break;
    }
    h /= 6;
  }
  return [h, s, l];
}
function stdDev(values){
 var avg = average(values);
 var squareDiffs = values.map(function(value){
  var diff = value - avg;
  var sqrDiff = diff * diff;
  return sqrDiff;
 });
 var avgSquareDiff = average(squareDiffs);
 var stdDev = Math.sqrt(avgSquareDiff);
 return stdDev;
}
function arrSum(data){
 var sum = data.reduce(function(sum, value){
  return sum + value;
 }, 0);
 return sum;
}
function average(data){
 var sum = data.reduce(function(sum, value){
  return sum + value;
 }, 0);
 var avg = sum / data.length;
 return avg;
}
//return one random element from the supplied array.
function randomElement(items){
 var item = items[Math.floor(Math.random()*items.length)];
 return item;
```

```
}
// given an array arr, and a number of elements n,
// return a new array with n random elements from arr
function getRandomElements(arr, n) {
   if (n > arr.length) return arr;
   newArr = [];
   while(n--){
      var key = Math.floor(Math.random() * arr.length);
      newArr.push(arr[key]);
      arr.splice(key,1);
   }
   return newArr;
}
/*
$('#theCode').keydown(function (e) {
  if (e.ctrlKey && e.keyCode == 13) {
    if ( e.target.value ) { eval(e.target.value); }
    $("<div></div>").html( e.target.value).appendTo('#doneCode').delay(20000).fadeOut();
    e.target.value = '';
    //$('#theCode').css("background-color", "#FFFF9C").animate({ backgroundColor: "#FFFFFF"}, 1500);
  }
  if (e.ctrlKey && e.keyCode == 38) {
    // reveal previous line of code
    $('#theCode').val( $('#doneCode').children().last().html() );
  }
});*/
function avatarUpdate(theId, av){
  /* if the avatar is new create an avatar*/
  if (! $('#'+theId).length > 0){
    $('#theAvatars').append(
      $('<div class="avatar" id="' + theId + '"></div>')
    );
  }
  if (typeof(av.theName) != 'undefined'){
    $('#'+theId).html('<span>'+ getAcronym(av.theName) + '</span>');
  }
  else{
    $('#'+theId).html('<span> ' + theId +' </span>');
  }
  if (typeof(av.theRole) != 'undefined' ){
    //metro Gnome gets special treatment
    if (av.theRole == 'Conduct'){
      $('#'+theId).css({   'marginTop' : '0px', 'paddingTop' : '50px'   });
    }
    $('#'+theId).css({
      'background' : 'url("images/'+ av.theRole +'.png") no-repeat 50% 0px',
    });
  }
  else{
    $('#'+theId).css({
      'background' : 'url("images/Unknown.png") no-repeat 50% 0px',
    });
  }
}
function apertMemoryGetPublic(key) {
  if (wsIsUp()){
```

```
    /* doesn't this message go to everyone? */
    var m = { request: 'getPublic', key: key};
    var s = JSON.stringify(m);
    ws.send(s);
    console.log('getPublic ' + key);
   }
   else{
    console.log('failed to getPublic ' +key);
   }
}
function getAcronym(avName){
  var matches = avName.match(/\b(\w)/g);
  if (matches != null){ return matches.join(''); }
  return '';
}
function getAvatars(){
  var avatarIds =[];
  $('.avatar').map(function(){  avatarIds.push($(this).attr('id')); });
  return avatarIds;
}
/* this is triggered whenever a public memory value is set */
// this might happen when ... ?
// this was so long ago. hmmm.
function setVal(theKey, theValue){
  if (theValue != null){
    if (theKey == 'theChord') newChord = true;
    window[theKey] = theValue;
    // console.log('setVal ' + theKey +' = ' + theValue);
  }
}
function apertMemorySendTo(id, value) {
  var password = getPassword();
  if(password == null) return;
  var m = {  password: password, request: 'sendTo', id: id, value: value};
  var n = JSON.stringify(m);
  ws.send(n);
}
function apertMemorySetPublic(key,value) {
  // call this in your code to set a key-value pair in a shared memory
  // entries are unique to each client/browser
  if (wsIsUp()){
    var m = { request: 'setPublic', key: key, value: value};
    var n = JSON.stringify(m);
    ws.send(n);
  }
}
function apertMemorySet(key,value) {
  // call this in your code to set a key-value pair in a shared memory
  // entries are unique to each client/browser
  if (wsIsUp()){
    var m = { request: 'set', key: key, value: value};
    var n = JSON.stringify(m);
    ws.send(n);
  }
}
function apertMemorySetFor(id,key,value) {
```

```
  var password = getPassword();
  if(password == null) return;
  var m = { password: password, request: 'setFor', id: id, key: key, value: value};
  var n = JSON.stringify(m);
  ws.send(n);
}
function apertMemorySetForAll(key,value) {
  if (wsIsUp()){
    var password = getPassword();
    if(password == null) return;
    var m = {  password: password, request: 'setForAll', key: key, value: value};
    var n = JSON.stringify(m);
    ws.send(n);
  }
}
function setupChoir(){
  /*
  if(typeof(ac) == 'undefined'){ setTimeout(setupChoir, 80); return; }
  theSamples = {
   'rrr':  new SampleBuffer('rrr.wav'),
   'shh':  new SampleBuffer('shh.wav')
  };
  theChoir = new Choir;
  theChoir.start();*/
}
function initialize(){
    if (!wsIsUp()) {    setTimeout(initialize, 100); return; }
    setupChoir();
    console.log('Getting Initial Values');
    apertMemoryGetPublic('pixelsPerMinute');
    apertMemoryGetPublic('theChord');
}
function apertMemoryPoll(){
  // this is called on a rhythm
  if (wsIsUp()){
    apertMemorySet('theRole', 'Display');
    apertMemorySet('theName', 'Disp.');
    var password = getPassword();
    if(password == null) return;
    var m = { password: password, request: 'dump' };
    var n = JSON.stringify(m);
    ws.send(n);
  }
  /*get existing avatars*/
  var theAvatars = getAvatars();
  /*update avatars with snapshot data*/
  for (someAvatar in apertSnapshot){
    // skip the public memory
    if (someAvatar != 'public'){
      avatarUpdate(someAvatar, apertSnapshot[someAvatar]);
      for (uId in theAvatars){
        if (theAvatars[uId] == someAvatar){ delete theAvatars[uId]; }
      }
    }
  }
  for (uId in theAvatars){   $('#'+theAvatars[uId]).remove();  }
```

```
  setTimeout(apertMemoryPoll, 60 ); // why 60?
}
$( document ).ready(function() {
  loadImage('images/'+theImageFile);
  setup();
  initialize();
  apertMemoryPoll();
});
$('#playPause').
   click(function(){
      if (isSinging == true){
         isSinging = false;
         $(this).css({'backgroundPosition': '0px 0px'});
      }
      else{
        isSinging = true;
        apertStartAudio();
        animate();
        $(this).css({'backgroundPosition': '-50px 0px'});
      }
   });
</script>
</body>
</html>
```

Appendix 2: Video Documentation
Supporting videos have been produced for each of the four artworks. In some cases, video documentation records a live performance and in other cases, it is merely a demonstration.

PeaceWeave
PeaceWeave was part of the *Peaces of McMaster Showcase* held at the Donaldson Family Marketplace on April 5th 2016, and helped to mark the conclusion of *Perspectives on Peace*: a year-long initiative of the McMaster President's Office. Documentation of the event (held on April 5th 2016) is accessible as a video file (PeaceWeave-Documentation.mp4) on attached media, and online at https://youtu.be/EeZ77XaQ6uU. The running time is 36:33. WiiRemote technical documentation is also accessible as a video file (PeaceWeave-Wii.mp4) on attached media, and online at https://youtu.be/sil7Jo3Y2WE. The running time is 1:54.

The choral set list for *PeaceWeave* was as follows: (1) Tonight Eternity Alone - René Clausen, (2) Kyrie - Timothy Corlis, (3) Gloria - Timothy Corlis, (4) Stars - Ēriks Ešenvalds, (5) Richte mich Gott - Felix Mendelssohn, (6) The Tiger - Lauren Bernofsky, (7) The Lamb - John Tavener, (8) After the War - Mark Sirett, (9) Rytmus - Ivan Hrušovský, (10) Muusika - Pärt Uusberg. The video files that serve as the photographic pan for each track (without the Bubble Overlays) are available on the attached media located in the folder titled /PeaceWeave-Tracks/.

As a more tangible addendum to the *PeaceWeave*, I also produced a series of aluminum prints with the same title, also in the context of the *Perspectives on Peace* initiative. At the time of writing these prints have been installed outside the President's Office in Gilmour Hall at McMaster University, and may be viewed online at http://www.nsitu.ca/blog/2016/07/peaceweave/.

Sedimentary
Documentation of the event (held on April 7th 2016) is accessible as a video file (Sedimentary-Documentation.mp4) on attached media, and online at https://vimeo.com/162170474. The running time is 4:46. Use the password iclc2016 to access Vimeo.

ReptileChoir
Documentation of the event (held on May 13th 2016) is accessible as a video file (ReptileChoir-Documentation.mp4) on attached media, and online at https://youtu.be/bfb90hAcBZo. The running time is 2:09. A more technical demonstration is also accessible as a video file (ReptileChoir-Demonstration.mp4) on attached media, and online at https://youtu.be/Pxyn-aTNhcE. The running time is 5:45.

StreetSong: A technical demonstration is accessible as a video file (StreetSong-Demonstration.mp4) on attached media, and online at https://youtu.be/7WuFXfWijOw. The running time is 9:27. At time of writing, no choral recordings have yet been made. However, a second video is available that records a traversal through StreetSong, including combined pitch cue audio for six voices. It alternates between the overhead display and the mobile device display. This video is accessible as a video file (StreetSong-Pitch-Cues.mp4) on attached media, and online at https://youtu.be/3ZeGV5IggDY. The running time is 3:55.

Appendix 3: Photo Documentation

Photos from each of the four artworks created for this project are available on the attached media in the following folders. They include the various panoramas involved. Thanks to Caroline Tabah for the photos of *Sedimentary* at the LIVELab. Thanks to K.J. Bedford and Debora Jesus for the Photos of *PeaceWeave*.

- /Photos-PeaceWeave/
- /Photos-Sedimentary/
- /Photos-ReptileChoir/
- /Photos-StreetSong/

I anticipate that this and other photo documentation will be published on my blog at http://www.nsitu.ca

Acknowledgements

Thanks to Dr. David Ogborn for supervising this project
Thanks to Dr. Paula Gardner for being my second reader
Thanks to The Factory Media Centre, and Amy McIntosh, and Vanessa Crosbie Ramsay
Thanks to the McMaster LiveLab and Dan Bosnyak, and Susan Marsh-Rollo
Thanks to AVTek Productions
Thanks to the McMaster President's Office, and to Teddy Saull
Thanks to the McMaster University Choir and Dr. Rachel Rensink-Hoff
Thanks to Earth Wind and Choir.