SoundScapeTK: A Platform for Mobile Soundscapes

Thomas Stoll
Kitefish Labs
Buffalo, NY, USA
tms@kitefishlabs.com

ABSTRACT

SoundScapeTK is a set of software tools for soundscape composition utilizing smartphones. Developed as an outgrowth of earlier technological solutions, the software is a simple system for developing and deploying a series of sounds placed in a physical space using GPS information from individuals' phones. The user experiences mixtures of these sounds as he/she moves about, with location information triggering all responses and interaction. The software is described in terms of its current core features, along with possibilities for extension and further capabilities.

1. INTRODUCTION

SoundScapeTK (SSTK) is an open source software project developing tools for soundscape composition utilizing smartphones. At present, SSTK is an iOS project, but Android or cross-platform versions are planned. The main focus of SSTK is to maintain a simple, effective, yet extensible software platform for composing and presenting mobile soundscape compositions. This paper outlines some examples of work that has influenced the design of SSTK, a thorough description of the software, the PureData-based audio engine, and other aspects of the system's design. Finally, some general ideas for the future development of the software are suggested.

2. PAST WORK

A number of mobile audio platforms have influenced the design of SSTK, in both direct and indirect ways. One direct predecessor of the software is Mscape [1] from Hewlett Packard (HP), used by artists such as Constance Fleuriot, Martin Reiser, Erik Conrad, and in cultural heritage projects[2]. In 2006, for instance, researchers from HP Labs worked with a team from the Historic Royal Palaces to create Escape from the Tower, a location-based mobile game at the Tower of London [3], in which players smuggled virtual escapees past the real Yeoman Warders, or 'Beefeaters'.

Mscape had its direct predecessor in MobileBristol [4], a project developed and supported by the University of Western England (circa 2004). Individual artists had pioneered similar applications in the prior decade including Teri Rueb —"Trace", 1999; "Drift", 2004—and Stefan Schemat—"Berlin Alexanderplatz 5.0", 1999. Some motivations and descriptions these works are documented in Stephen Wilson's book on "Information Arts" [5]. Other works from the pre-smartphone era that include advanced uses of technology in the form of laptops carried in backpacks [6] by participants have been presented. The equipment has shrunk from laptop to handheld, and evolved from D.I.Y. hardware to common off-the-shelf mobile phones.

Though these first projects employed custom software developed by Teri Rueb in collaboration with a variety of different programmers, she has continued to develop works in this genre that sometimes employ commercial software including MobileBristol—"Itinerant", 2005—and Mscape—"Core Sample", 2007. SoundScapeTK is the result of the author's work with Rueb since 2012. Not driven by the tools, Rueb instead is committed to developing compositions bas-ed on cues and inspirations drawn from the sites themselves and their social contexts. As such, each project demands a different set of technical solutions and functionalities. SSTK reflects basic functions of location-based sound playback as appropriate for the project for which it was first created—"No Places With Names", a 2012 collaboration with Larry Phan and Carmelita Topaha. It was also used in a new iOS version of her first GPS-based sound walk, "Trace", created in 1999 as a New Media Co-production with the Banff Center for the Arts. [1]

As the hardware has become more mainstream, the software includes some open source code in the form of the PureData-based audio synthesis engine, libpd [7]. A number of artistic applications have appeared in the first years of the smartphone era with Pd as a mobile audio engine. Most notable is RjDj [8], which produced an app that attempted to remix audio from listeners' smartphone microphones into their headphones. RjDj's attempt to sonify ambient audio into music is directly related to work from artists, notably Layla Gaye [9].

There are a number of albums or album extras that are released in the form of apps—see Daft Punk's iDaft [2] or Radiohead's PolyFauna [3] as examples. These apps are intended to add to the experience of the audio, and, as in these two examples, many feature tools for remixing. There is one musical group, Bluebrain, that has produced several location-based mobile apps [10] conceived as albums. These app-album hybrids are based at famous landmarks,

[1] For more information see http://www.teririeb.net.
[2] http://bit.ly/1m46o6h
[3] http://bit.ly/1jGl5sE

```
"1" : {
"type" : "sampler",
"sides" : 0,
"center" : [ 43.65072, -72.310644 ],
"radius" : 0.000395,
"sfIDs" : [ 1 ],
"attack" : 2000,
"release" : 5000,
"loop" :  1,
"cutoff" : 1,
"active" : 1,
"toactivate": [ 2, 3 ],
"lives" : 25
}
```

Figure 1. JSON code for a circular region, mapped to a sound file. This region loops, activates two other regions, and will play up to 25 times.

including the National Mall [4] in Washington, D.C.

3. SOUNDSCAPETK

SoundScapeTK is a set of classes designed to provide a simple, yet extensible platform for soundscape composition using smartphones. Once activated, the app runs a routine that continuously queries the user's location at a set time interval (defaulting to once per second). Once a valid location with sufficient accuracy is detected, the location is hit-tested against an internal model of sounds mapped to regions in space, according to a special file loaded at launch. The map is a series of—potentially overlapping—circular regions, and once the software detects a"hit" for a region, the app plays the linked sound file or feeds the appropriate parameters to a synthesis routine.

This section describes the system in detail, including several practical tools and extensions to this very simple concept. The description of the software includes rules-based interactivity based on a special JavaScript Object Notation (JSON) data file, the handling of fading between regions' sounds, the use of Pd as sound engine, the use of the GUI for testing and monitoring purposes, and other aspects.

3.1 The GPSON File and Rules for Interaction

There are several basic functions that, in addition to the above summary, describe how SSTK works. Each sound file and the region to which it is mapped is maintained as part of the application's state. Further information fields attached to these lists are maintained as well. An instance of the app, which is also fully encapsulates a composition, is based around a GPSON file that describes the regions, mappings, and other parameters that make up the particular piece. GPSON files are JSON files with a certain arrangement of data expected by SSTK. Figure 1 shows an example region encoded within a GPSON file. Since this is the heart of the system, in terms of representing each region-to-sound mapping, a detailed explanation follows.

[4] http://bit.ly/1iQiE3x

At present, there is only one type of region shape. Each region description contains the latitude, longitude, and radius, all expressed in decimal longitude/latitude values, of the geographical region to which a sound is mapped. The mapping from region to sound file is accomplished by providing a list of sound file identifiers (sfIDs) that identify which sound file is too be triggered. At present, sound files must be named as zero-padded 2-digit numbers followed by the extension. In this case, the sound file with the key 1 points at a file bundled with the app named "01.wav" or "01.mp3". There are controls for the ramp times for both the attack and the release times. These times are expressed in milliseconds. These parameters and mappings are the most basic set of information for a region.

In addition to the basic information, there are several additional parameters. There are flags for looping and for activating the region—not every region needs to be active when the piece begins to run. The inactive regions would need to be activated in some way, and regions with lists of "toactivate" region indexes cause those regions' active flags to turn on when that first region is first entered. The "lives" setting contains the maximum number of times that the region can have its associated sound file triggered. Each time the region's associated sound file is played, even if not in its entirety, the number of lives is decremented. Once the number of lives is decremented to 0, the region becomes, for all intents and purposes, inactive. Looping, if enabled, will stop once the lives limit is reached.

Two final rules play an important role in the deployment and realization of a piece within SSTK. One governs the pausing and resumption of sound files' playback, and one defines the maximum polyphony. When a user is detected to be outside a region that has been playing sound, sound playback is either cut off or allowed to complete for the current traversal through whatever sound file is being played. If the "cutoff" flag is set to non-zero, the region's sound will be paused upon exit and resumed upon reentry. This behavior can be modified by changing the cutoff flag to a negative number, thus causing playback to begin again at time point 0 in the sound file upon future reentry to the region. The maximum polyphony is an implicit rule. Any time that a user "hits" a region while there are already the maximum number of sounds playing, region entry and playback of linked sounds is blocked by this rule. The composer is not limited in any way to a set maximum number of overlapping regions, so this implicit rule could be used to make an interesting unpredictable interaction within the rules of the system.

3.2 Overall Software Design

SSTK is a collection of Objective-C classes that make up a mobile app. These classes are somewhat modular, depending on the particular needs of a piece or depending on the sound design used. The default sound design, described below, is a polyphonic sampler with four voices. The major classes are listed in Table 1. While there are many modifications that can be made, the core functionality is completely contained within these classes, a few other code files, GPSON files, and the sound files them-

SSTKAppDelegate	Launches Pd, file operations.
SSTKViewController	Draw the interface, handle raw locations.
HTLPManager	Hit-testing and stateful playback logic.
LAPManager	Track location; post-process raw location data.
AudioFileRouter	Perform the actual message-sending that controls Pd.
LinkedCircleRegion	Encapsulate data associated with circular regions.
Linked SoundFile	Encapsulate data associated with sound files linked to regions.
MapViewController	View regions, current location, and current state information.

Table 1. The classes that comprise SoundScapeTK.

selves.

3.3 Pd Sampler Interface

PureData (Pd) is used as the sound engine for SoundScapeTK and pieces built with the software in the form of libpd. libpd runs on a wide variety of mobile and non-mobile platforms and is, critically, designed to be run without it's patching GUI. The sound engine is programmed graphically as a patch and then loaded by the C framework. All interaction takes place through messaging. Within the graphical interface to Pd, with which most electronic music makers should be familiar, the messaging feature is a convenient way to send information from one part of a large patch to another. In libpd, the "sends" are C or Objective-C functions, and the corresponding receives can be seen in the patch that is loaded. The reverse situation takes place as well: information is sent from the loaded patch to a C function that receives data within the running app, often for debugging purposes.

With minimal effort, SSTK can be modified to accommodate a new Pd patch and, thus, a new sound engine. The current default within the code for SSTK is a sampler with four independent voices (see previous section). Figure 2 shows one voice of such a sampler. It is trivial to modify the patch to run a different number of voices and increase or decrease the maximum polyphony of the system. While there is an upper limit to the number of simultaneous voices—recall that this being run on a phone—there is no theoretical limit to the scaling of this feature. Likewise, there is no limit to the variety of sound designs that could be incorporated into SSTK. The current system, based on sound playback, is just one possible configuration of one design; there exist Pd patches for both the playback of MP3 and PCM files.

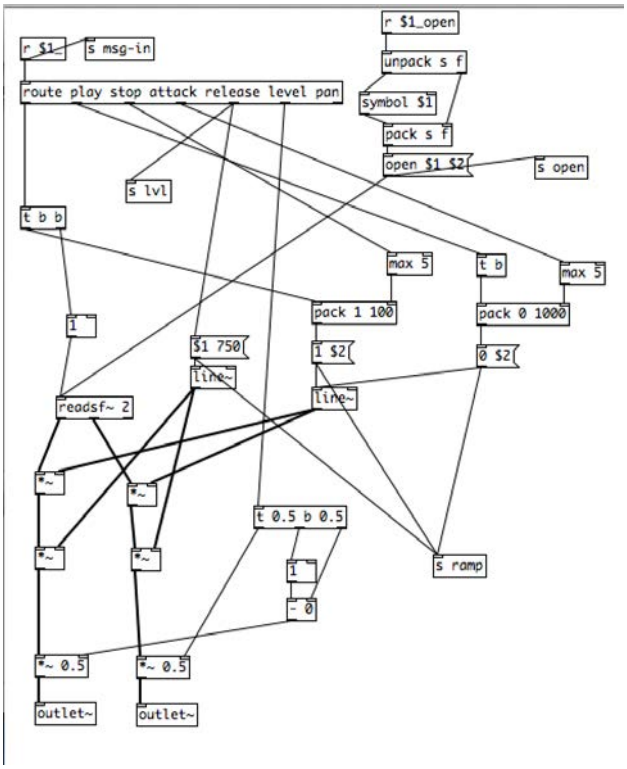

Figure 2. The Pd patch for one sampler voice. The send objects are passing information to Objective-C that is logged in order to debug the application.

3.4 Synthesis interface

The author has successfully deployed moderately complex synthesis-based sound design within an iteration of the app where geographical regions map to certain synthetic sounds and, furthermore, the location of the user within the region causes changes in synthesis parameters—for instance, the speed of a low frequency oscillator (LFO). In "The Wheel Within the Wheel" [5] , the user's location relative to the radii and/or rotational angle within a region controls the parameters of a polyphonic bank of modular synthesis voices. Each region visible in Figure 3 is actually 4-16 regions overlaid one upon the other. The relative angles and radii map to levels and low-frequency oscillation rates within each voice. The specific mappings are made within the piece's GPSON file.

3.5 Composition Interface

SoundScapeTK would be rather limited without an interface for the composer to test ideas and pieces. With this in mind, we have introduced visualization and editing capabilities to the core app. There are a number of ways to update the information contained within the GPS score (see above). First, the GPSON file may be replaced by the user, without recompiling, by downloading new files through a settings or preferences menu/window. This ability to swap versions of the map for a piece accelerates one's workflow. Rapid development and deployment of sounds is achieved

[5] http://bit.ly/1ssZlUL

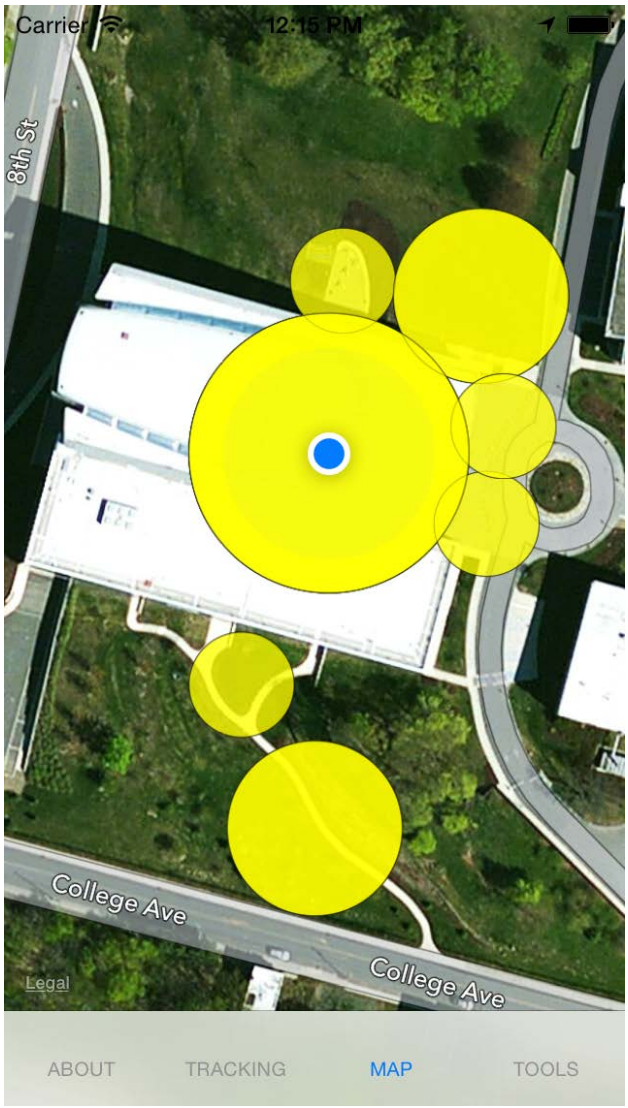

Figure 3. Several circular file regions used in the piece "Wheel".

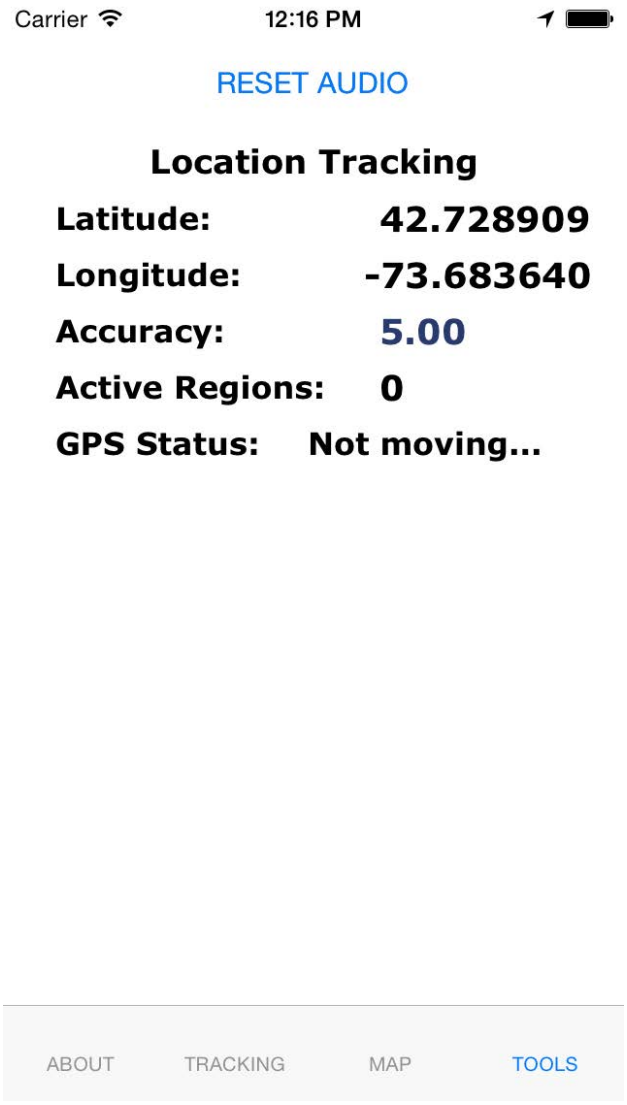

Figure 4. The main interface for an iOS app "The Wheel Within the Wheel" built using SSTK.

through a system (optional, used mainly for testing) relying on the Amazon Simple Storage Service (S3) services. Such a feature may be turned on within an instance of the app by changing several flags, setting up an S3 instance, and inputing the corresponding access data into a preference pane. Other synchronized, cloud-based services will be incorporated in future versions.

3.6 Testing and Verification

The user is able to visualize the map overlaid with sound regions and can display current state information—see Figure 3—about each region on the map. in a different view, status feedback tools are available—see Figure 4. In a version compiled for composition, the composer/creator is able to enter edit mode at any time and change the size and location of regions. The rules for interaction and state tracking are also available for editing within the app.

SSTK is testable in a virtual way as well. The map viewer interface allows the user to navigate using tap gestures while the automated location tracking is switched off. This test-

ing interface allows the composer to check combinations of sound regions and to virtually walk the map in real time in order to rapidly experience the kinds of behavior the system will exhibit when deployed in the real world. Test interactions can even be automated with standard mobile debugging tools.

3.7 Persistance and Connectivity

In all iterations of SSTK, each individual user's state is maintained within the individual's app. As part of the testing process for several pieces, different strategies for logging of real time data were tried. The simplest involves keeping a list within the app that logs each action—when the location changes, when files are triggered, etc.—and making that list available in some form outside the app. The author's system, which works quite well, takes advantage of the ability to launch an email editor with the contents of this log file and send an email (usually to oneself, when testing) with a log of your most recent activity. In the future, this functionality will allow for the offline recreation of a sound walk. There are many more possible uses for persistent state data and the communication of data to and from the user, as discussed below.

4. FUTURE EXTENSIONS

There are a number of features that have been considered or tested experimentally, but are not yet included in the open source release of SSTK. There are additional shapes for regions and possibilities for interaction rules. For instance, rectangular regions are partially implemented within the codebase of SSTK and could be fully reintegrated into the system with a few lines of code. Further parameters will likely be added in the future to control sound playback; for instance, the phone's gyroscope data might effect directional mixing of multiple sound sources in a composition.

Since the app already tracks the state of a user's interaction with a sonic map, it would be trivial to share that data with other users or with a central server. Likewise, a server could dynamically feed information to the app about new sound locations or change regions based on other users' interaction. The central server can act as a gateway or a central controller for an interactive experience among individual smartphone users.

5. OPEN SOURCE SOFTWARE

A development version of the software is available at GitHub [6]. While individual pieces are forked from this version of the software, enhancements and bug-fixes will be merged into the main development tree as they happen.

6. REFERENCES

[1] S. Stanton, R. Hull, P. Goddi, J. Reid, B. Clayton, T. Melamed, and S. Wee, "Mediascapes: Context-aware multimedia experiences," *IEEE Multimedia*, vol. 13, no. 3, pp. 98–105, 2007.

[2] H. Labs. (2008) mscapefest locates to belfast. [Online]. Available: http://www.hpl.hp.com/news/2008/oct-dec/mscapefest.html

[3] ——. (2006) Escape from the tower of london. [Online]. Available: http://www.hpl.hp.com/news/2006/oct-dec/tower.html

[4] M. Williams, C. Fleuriot, K. Facer, J. Reid, R. Hull, and O. Jones, "Mobile bristol: A new sense of place," in *Proceedings of UbiComp 2002*, 2002, p. 27.

[5] S. Wilson, *Information arts: intersections of art, science, and technology*. MIT press, 2002.

[6] I. for Unstable Media. (2000) Sonic interface. [Online]. Available: http://v2.nl/archive/works/sonic-interface

[7] P. Brinkmann, P. Kirn, R. Lawler, C. McCormick, M. Roth, and H.-C. Steiner, "Embedding pure data with libpd," in *Proc Pure Data Convention 2011*, 2011.

[8] RjDj. (2014) Sonic experiences—rjdj. [Online]. Available: http://www.rjdj.me

[9] L. Gaye, R. Mazé, and L. Holmquist, "Sonic city: the urban environment as a musical interface," in *Proceedings of the 2003 conference on New interfaces for musical expression*. National University of Singapore, 2003, pp. 109–115.

[10] C. Richards, "Go with music's flow," *The Washington Post*, p. C.1, 5 2011.

[6] `http://github.com/kitefishlabs/SoundScapeTK`