EARplay

Exaggerated Acoustic Reality
play prototype app

Matt Adair
s3204584@student.rmit.edu.au

Overview

EARplay is an app developed for specific Android OS mobile phones and requires stereophonic microphones which listen to the surrounding environment and use it to form the basis of an abstracted and evolving soundscape listened to with headphones.

Synopsis

EARplay uses two microphones to create an abstracted soundscape from to the surrounding environment. General background ambience will pass through the app relatively untouched so as to provide a realistic context for the derived abstracted sounds. This will be created from the more prominent sounds as determined by a combination of the app audio engine and user settings. The intention is to add a creative sonic element to the surrounding environment of the user that augments and exaggerates the tonal qualities found within.

Function

EARplay is designed to make the user aware of the rich and complex sonic environment around them. To do this it mixes an accurate reproduction with an effected version, in much the same way as a musicians digital signal processing effects unit does. To further the basic effects lay people may be familiar with (delay, reverb, transpose, etc.) this app adds a more abstract element with a synthesis engine programmed to respond to tonal qualities with a high level of presence.

An example use will be the user commuting to work and, as they walk down the street to the train station, the app is invoked and then put to background while they read a news website. The app slowly increases the balance of its effected to dry audio ratio and the sounds of passing traffic soon get augmented by musical tones and melodies "attached" to cars. Footsteps are echoed and transformed into analogue synthesis drop sounds. The wind has harmonics added to it and a melody is also layered on top.

Design

Earplay consists of two main app windows that the user will see, with a third more advanced settings window available at the third touch. The primary window will load up after a splash screen with basic description is displayed for a few seconds. The main interface window is intended to be used as a form of screensaver with no touch inputs possible other than to invoke the settings dialog and exit buttons.

This main window uses an openGL library to draw shapes and waves that are intended to visualise what data the FFT process is producing. This process will operate at a lower priority than the audio processing and so, dependant on the device used, may have a framerate reduction from the target 30 fps. This is acceptable as it is intended to be primarily an audio app that runs in the background while the user can perform other phone related tasks if desired, or to just enjoy the sounds.

The visualisation is accurate enough to allow the user to "see" the sound wave augmentation as it happens and can be used to help understand the conceptual processes involved. The app engine will automatically reduce the framerate of this window if the latency rate for the audio crosses a programmed threshold. A setting control is also available on the advanced setting window that allows the user to manually reduce the frame rate in case they wish to retain the visual reminder that the app is running but do not want it to impact the audio to the default level. This could be useful if the user has changed the more advanced synthesis engine settings which may impact their particular device beyond the profile of the listed target devices (see appendix).

Concept art – main interface window

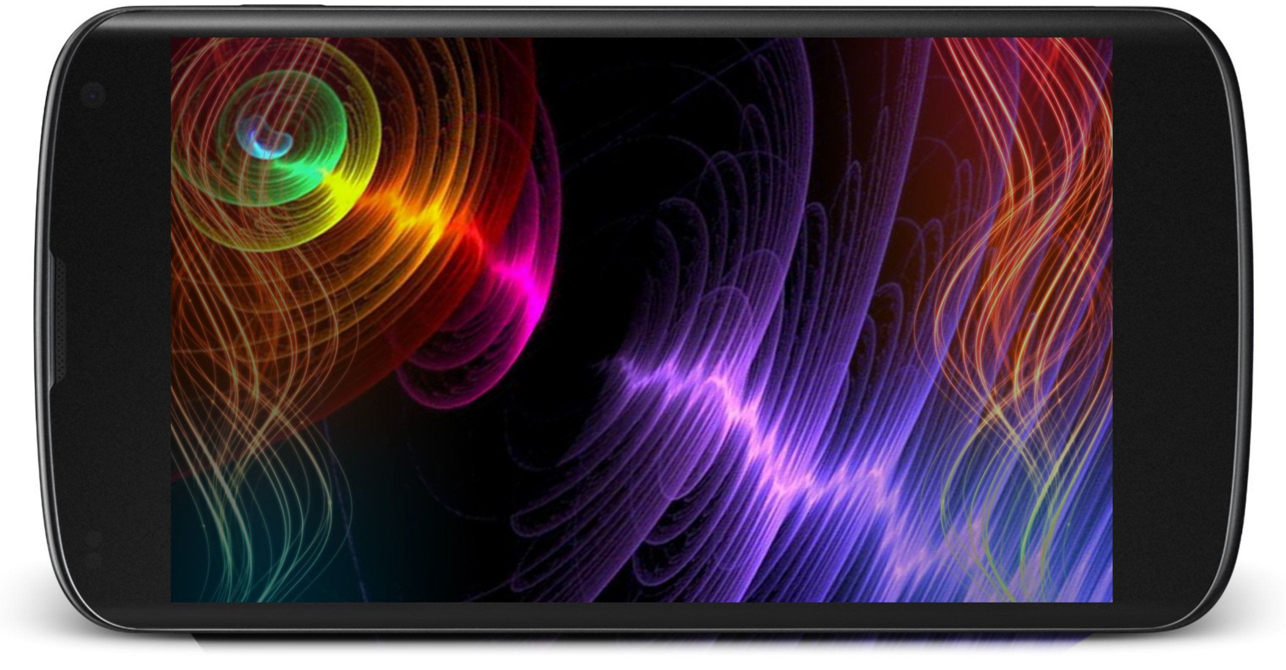
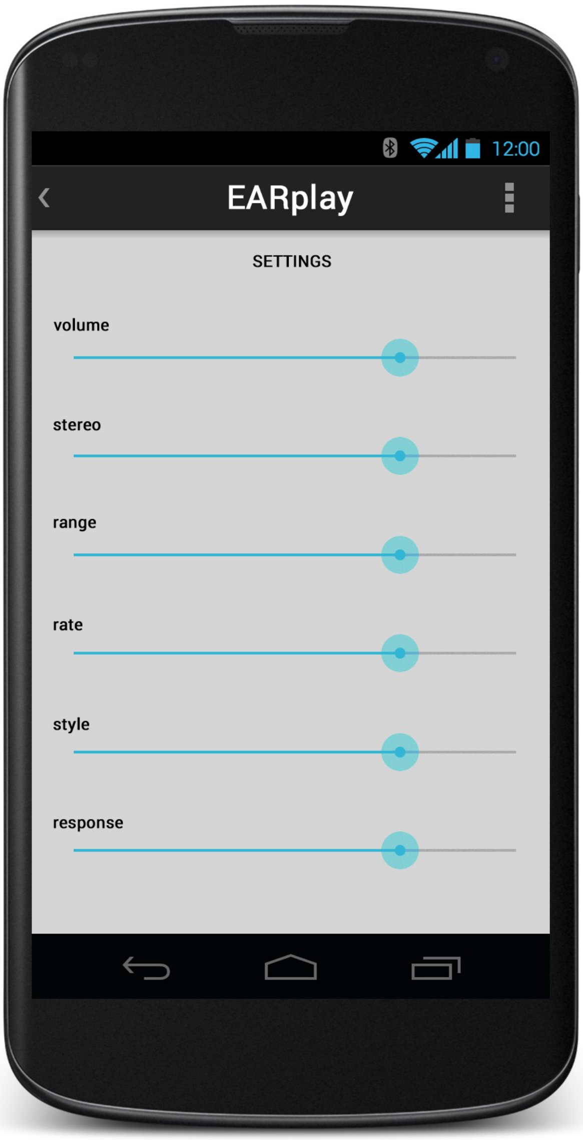


Fig 1. EARplay sound wave visualisation running on main interface window

Concept art – settings windows



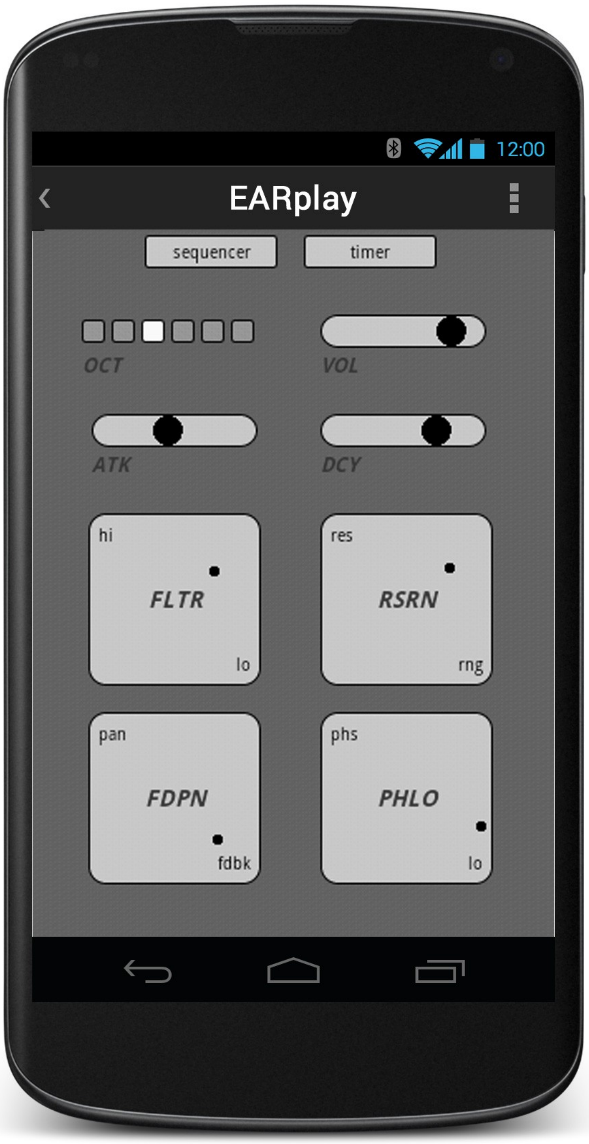


Fig 2. LEFT - EARplay main settings window showing slider controls for non-technical users

Fig 3. RIGHT – EARplay advanced settings window showing controls for the synthesis engine

Interactivity

It requires two microphones on a suitable Android device (see appendix). An alternative is for USB host mode enabled phones and an external stereo microphone. Minimum OS is Android 4.x. The target latency for this app is 50-100ms. A proper stereo effect is only possible with the device used in landscape mode, unless an external stereo microphone is used.

Three processes are run concurrently on the device, the first is the Audio-In that takes audio from the two microphones, provides some limited EQ and compression before then writing to a cache file. The second process reads the cache file and runs FFT comparisons looking for audible moments of interest, or blobs, and tags them for tracking in the third process. The third process analyses the tagged blobs as well as the surrounding audio and then uses this for the creation of a musical track consisting of waveform synthesis combined with the original audio recorded.

Several key settings will be presented to the user via a menu option and these will control major effects such as effect balance, response time and saturation. There is scope for more finite control via a secondary menu that would present access to settings such as musical styles, tempo, note on, note off, ADSR (attack, decay, sustain, release) envelopes, reverb, delay and assignable low frequency oscillators. These are considered to be more useful to musicians or those aware of synthesis theory but could prove to add a fun element to the lay user.

Market Analysis

Several apps are currently available that offer some similar but reduced functions to EARplay. Firstly is Inception The App which utilises PureData for audio manipulation and requires iOS 4.0. This app is more of a programmed score that uses GPS, accelerometer, ambient light sensor and microphone to change various musical parameters. Control is not finite at the synthesis level and as such is intended for the casual user who wishes to "induce dreams" in a game-like fashion.

Around Sound is available for Android devices and uses a similar technology of analysing surrounding audio and performing a task based upon certain criteria being met. The purpose of this app is to duck any audio currently playing and so allow the user to be alert to their surroundings.

A closer match to function is to be found in London artist Yuri Suzuki and his Sound Taxi that is equipped with a microphone to record the street sounds, manipulate them musically and play them back via the built-in speaker system. This, however, is the size of a London Black Cab and not mobile in the phone sense.

While these three examples offer some similarities to either scale or concept, they do not offer the simplicity and interaction of EARplay which is designed as an app that has at one level a basic, playable interface and at the other has more finite control typically found in a synthesiser.

Audience

Primarily this EARplay is aimed at musicians who may enjoy playing with the settings and experimenting with the app beyond its intended usage, such as using it as an effects unit during a live performance. This app may prove fun for the average user interested in a novel soundtrack based on their surroundings but it is assumed that long term use will be curtailed by high CPU and battery usage that is not desired in a mobile phone.

The Android operating system has in the past suffered from poor implementation of the audio APIs as well as direct access to audio hardware but this is slowly being rectified with a more mature release in the 4.x branch. The latency difficulty has meant that there are few apps available in Google Play that offer EARplay's combination of real time analysis and synthesis. Interesting and experimental music software has ordinarily been available to the user of the iOS system with its stricter hardware control and better software implementation. Despite this, the Android system is the number one mobile smartphone OS in the world and as such can be seen as a potentially large market for novel and fun audio apps.

Original Selling Points

- interactive
- explore the audible environment
- responds to the users surrounding
- inspire curiosity
- low level competition in the chosen market
- unique soundscape generation
- useful addition to musician's creative studio

Appendix

Tested device list -

Asus	Nexus 7
HTC	Desire, One M7, One S, One XL
LG	Nexus 4
Samsung	Galaxy Note II, Galaxy S2, Galaxy S3, Galaxy S4(i9505), Google Nexus S
Sony	Xperia S

3D spectrum of sound waves -
(product of time over frequency)

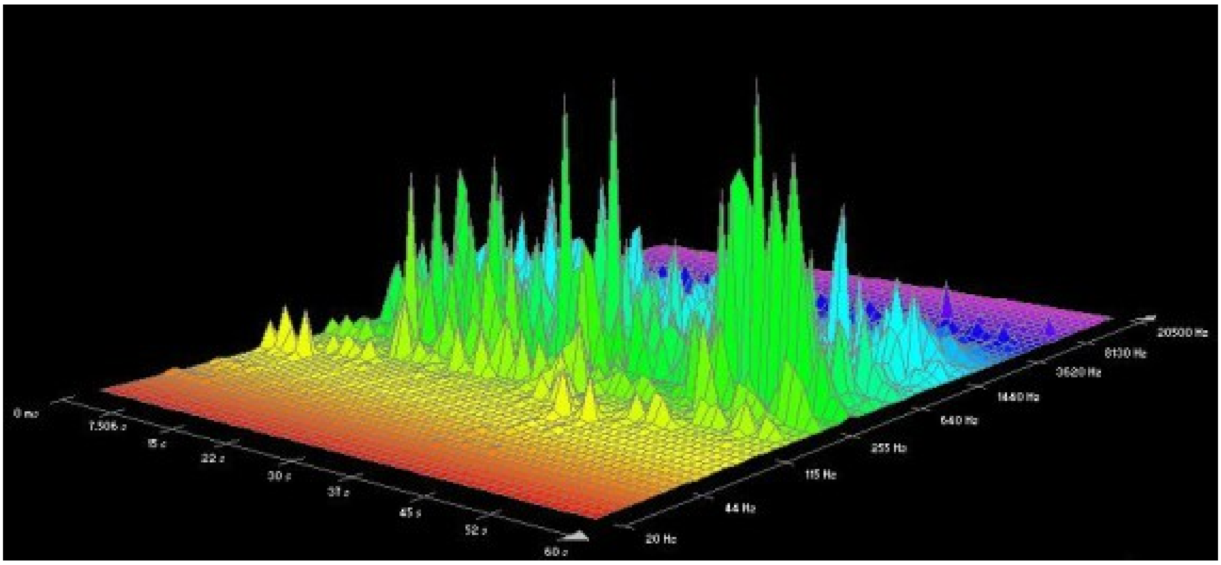


Fig 4 Graphic showing example FFT analysis of peak sounds with frequency and duration

Reverb visualisation -

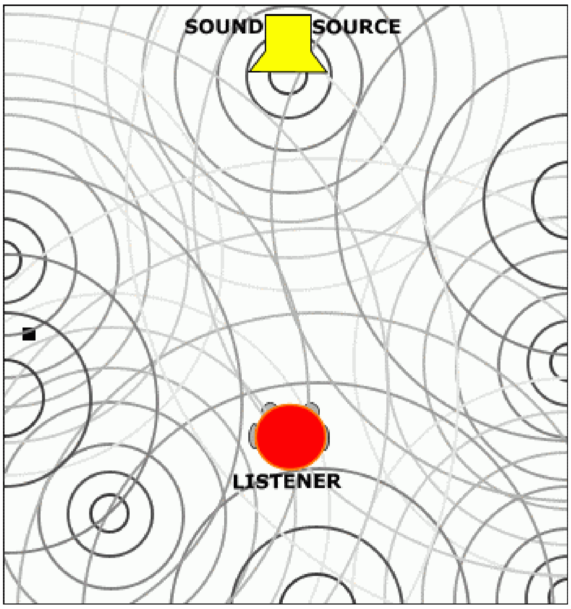
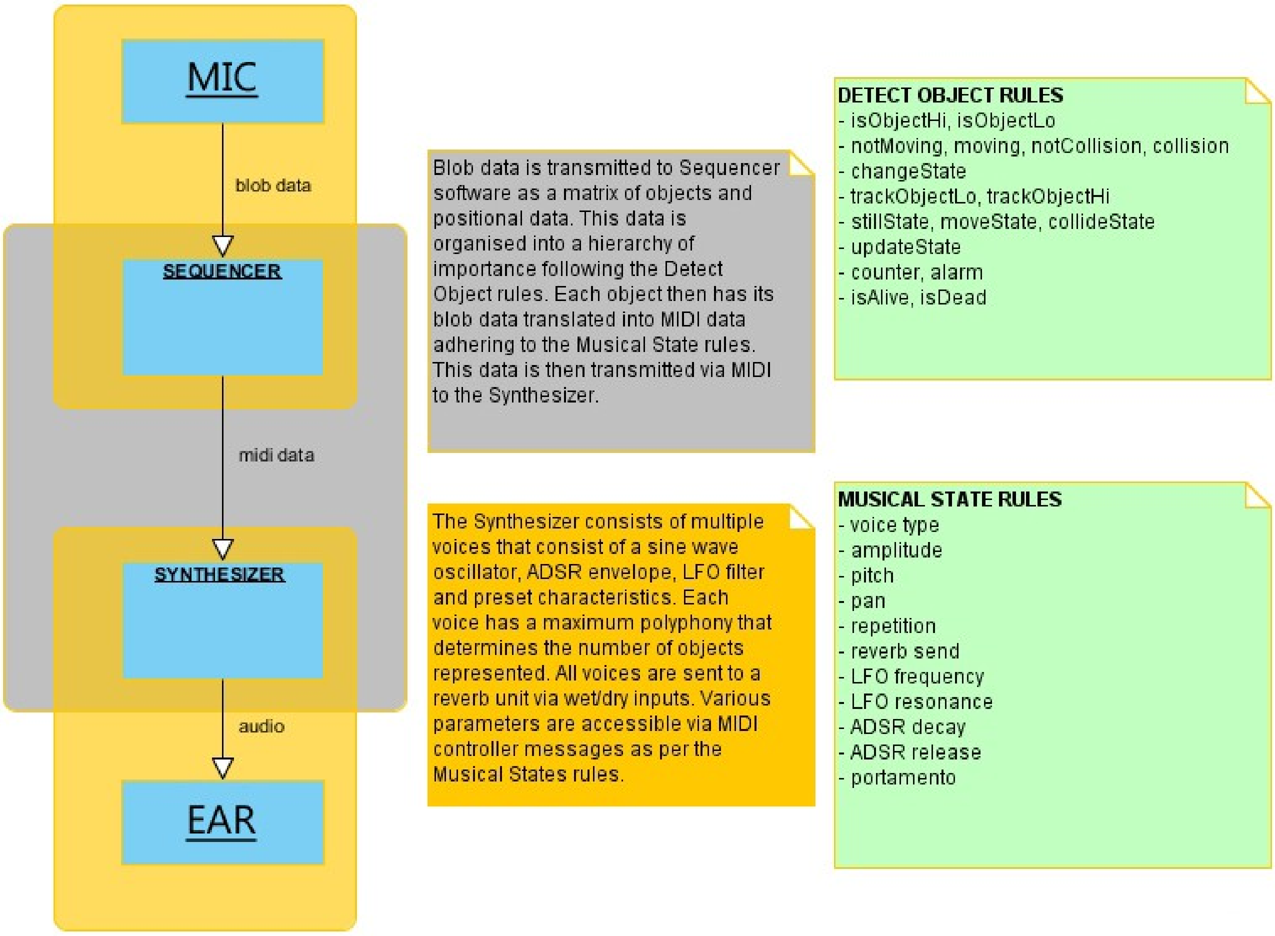


Fig 5. Graphic showing example of major sound sources and reverberations around a user

EARplay software concept chart -



Blob data is transmitted to Sequencer software as a matrix of objects and positional data. This data is organised into a hierarchy of importance following the Detect Object rules. Each object then has its blob data translated into MIDI data adhering to the Musical State rules. This data is then transmitted via MIDI to the Synthesizer.

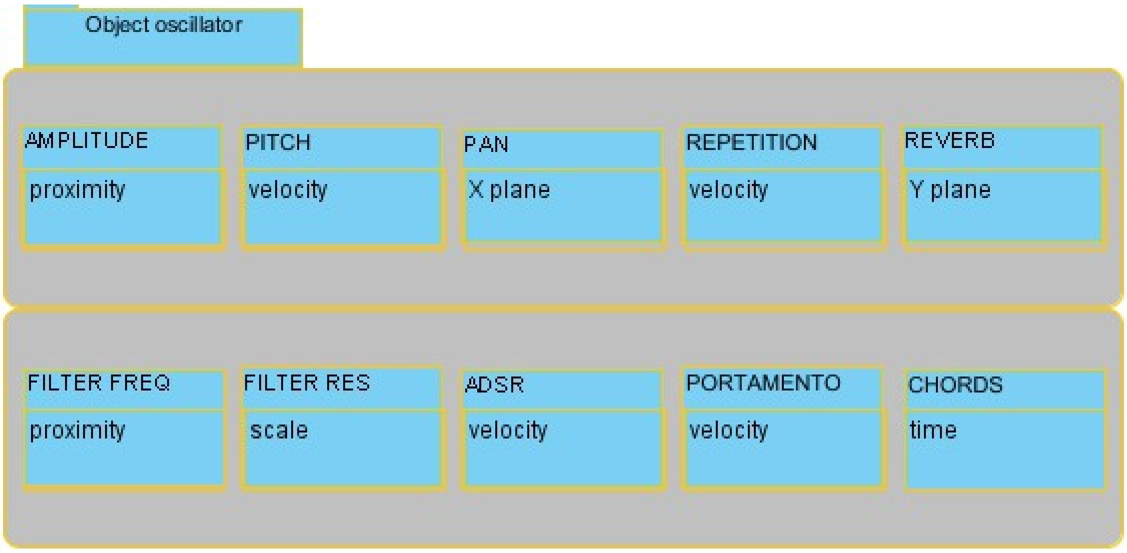
The Synthesizer consists of multiple voices that consist of a sine wave oscillator, ADSR envelope, LFO filter and preset characteristics. Each voice has a maximum polyphony that determines the number of objects represented. All voices are sent to a reverb unit via wet/dry inputs. Various parameters are accessible via MIDI controller messages as per the Musical States rules.

DETECT OBJECT RULES
- isObjectHi, isObjectLo
- notMoving, moving, notCollision, collision
- changeState
- trackObjectLo, trackObjectHi
- stillState, moveState, collideState
- updateState
- counter, alarm
- isAlive, isDead

MUSICAL STATE RULES
- voice type
- amplitude
- pitch
- pan
- repetition
- reverb send
- LFO frequency
- LFO resonance
- ADSR decay
- ADSR release
- portamento

Fig 6. EARplay flow chart of software functions and purposes

Musical state machine rules -


Object oscillator

AMPLITUDE	PITCH	PAN	REPETITION	REVERB
proximity	velocity	X plane	velocity	Y plane

FILTER FREQ	FILTER RES	ADSR	PORTAMENTO	CHORDS
proximity	scale	velocity	velocity	time

Fig 7. EARplay synthesis parameters for an audible object blob as determined by the app

Detect object state machine rules -

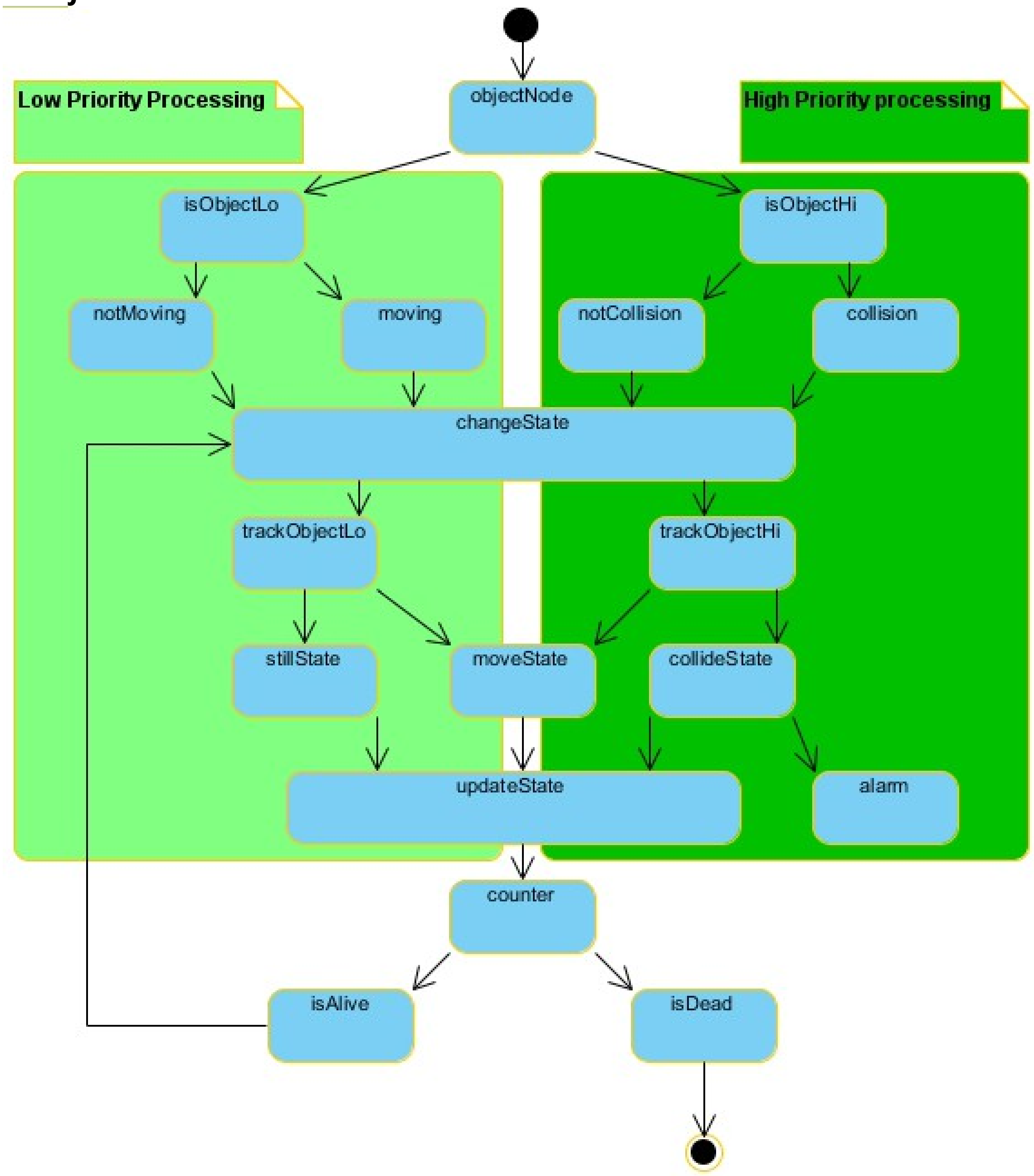


Fig 8. EARplay processing state machine of given audible object blob lifecycle

References

OS comparison - http://www.synthtopia.com/content/2013/07/30/why-android-the-1-mobile-platform-wont-get-great-music-apps/

Puredata visual programming language - http://puredata.info/

Inception The App - https://itunes.apple.com/app/inception-the-app/id405235483?mt=8

Around Sound - https://play.google.com/store/apps/details?id=com.MarksThinkTank.AroundSound&hl=en

Sound Taxi - http://yurisuzuki.com/works/sound-taxi-2/