A location-based polyphonic binaural playback engine for recorded audio and text-to-speech for iOS

Thomas Resch[1,2]

[1] FHNW Academy of Music, Basel, Switzerland, E-Mail:thomas.resch@fhnw.ch
[2] Audio Communication Group, Technische Universität Berlin, Berlin, Germany

Abstract

The current smartphone generation is capable of calculating several binaural audio sources simultaneously. Yet all available applications for audio guides merely provide stereo playback. Furthermore, the software for creating the guides – usually a web interface - lacks the possibility for using a text-to-speech (TTS) engine in order to allow for fast prototyping. This paper describes the necessary components and a corresponding iOS demo application, which combine these technologies and enable the user to create a location-based, polyphonic binaural guide or game. The audio sources can be a combination of prerecorded audio and simple text, which will be reproduced by a TTS component. The localization is realized by using the built-in GPS and WLAN system; additionally, Bluetooth tags can trigger events indoors.

Introduction

The first documented handheld museum guide was built for the Stedlijk Museum in Amsterdam in 1952. Although the technique has improved dramatically, the principle used nowadays is still the same: people walk inside or outside; while walking, they hear a voice telling them where they are, or what they are looking at. Many software solutions already exist for these applications. Most vendors provide a web interface for creating the walks by combining GPS data with media files. The generated scripts then get stored on a server. A corresponding mobile client can access them and play back the media files, when the user arrives at a specified GPS position. If no further functionality is required, these solutions work perfectly well. However, beginning in the 1990's, artists such as Jannet Cardiff started using the same principle. But instead of hard facts, the participant of such a walk hears a story or music or – since the rise of the smartphone - can even see pictures or videos on the display. In other projects, for example in the Lautlots guide [1], which took place in 2013 in the german railway station in Basel, the participant even have to interact with the guide through gestures. Furthermore, a special type of game – the location based audio game – came to great attention recently with the game "Run! Zombie, Run!" [2].

The software described in this paper – the Real World Audio (RWA) game engine - enables developers and artists to build these applications for iOS. It may be used for simple guides but also provides the functionality for creating location-based audio games. The software is still under development and constantly subject to changes. Parts of it are available for download at [3].

Related Work

Due to the fact that this paper can be considered as a follow up on the paper about RWA by Resch [4], all literature and audio-guide-software mentioned in there are also relevant for this article. Additional examples of commercial tools for creating audio guides are texetera [5] or the mobile app from orpheo [6]. Both enable the user to combine GPS data with audio files; the latter one even provides augmented reality video features. But they lack the possiblity to realize program logic besides the localisation. A very detailed description of augmented audio reality including several use cases using head-related transfer functions (HRTFs) and head tracking is provided in [7]. The algorithm and the implementation used in the RWA software have been developed by Markus Hädrich for his binaural soundmap app [8].

The Desktop Application

The RWA-Creator Application is written in C++ with the Qt framework [9]. Although it is under development and constantly subject to changes, its architecture and main functionality are described accurately in [6]. Therefore – after a brief summary of the data structures and the workflow with the main graphic user interfaces (GUIs) – this section deals only with the new text-to-speech and simulation components.

An RWA script consists of at least one *scene*, which can be considered a state machine. A transition to a state is executed, if the necessary entry conditions are fulfilled (for example a certain GPS location); the corresponding state then becomes the active state of the mobile client (the person carrying the smartphone), and all assets of that state (in most cases merely audio files) will begin to play. From now on, the distance and the bearing between client and assets are calculated constantly for the binaural reproduction until the client enters a new state. In the case of simple mono or stereo playback, the distance might be used for adjusting loudness. Audio playback and signal processing for the binaural synthesis are realized with libPd [10] – a version of the Pure Data (Pd) [11] environment without a GUI.

The figure below shows a scene containing four states: a fallback state which is entered by default if no other state is active, two GPS states which will be entered, if the user arrives at a specified position, and one state, which will automatically become the active state after the specified time-out value of state 1. In addition to these cases, all types of entry and exit conditions may be used and combined: a Bluetooth beacon in range, a radio-frequency identity (RFID) chip or a specific orientation of the client.

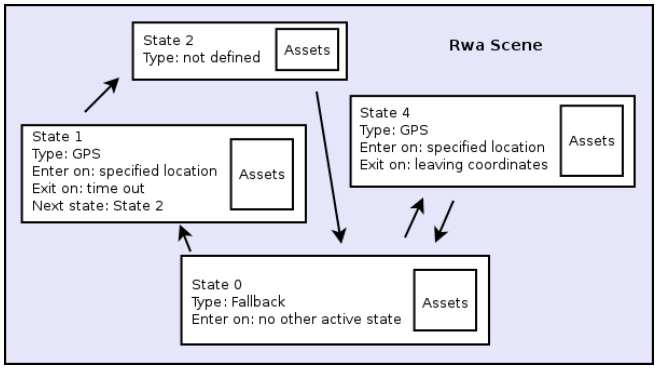

Figure 1: The data structure of an RWA scene containing four states and their possible transitions

On startup, the RWA Creator opens the most important windows automatically: the map view (for creating and editing GPS states with the mouse), the scene view (where also non-GPS states are editable), the state view and the TTS editor. Assets can be added to a state via drag and drop into the state view or by using the TTS editor.

The text-to-speech component uses the CMU Festival engine, developed at the University of Edinburgh [12]; it is bound to RWA through its C++ interface. The connected GUI consists of a text editor, a menu for selecting the voice, and three buttons for generating, prelistening and adding the speech to the currently active state. Festival implements *sable,* a TTS markup language, which provides tags for manipulating intonation, pitch and tempo [13]. They can be incorporated directly into the text as shown in the picture below.

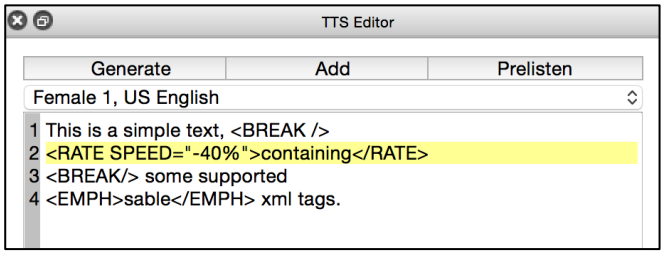

Figure 2: The RWA TTS editor. It provides access to CMU Festival, which implements a sable markup interpreter.

After activating the simulator – with the play button in the upper left corner of the map view – the game loop starts with a refresh rate of 10 Hz and (at least) one client will appear on the map; it is depicted as a black arrow with a name below and can be moved with the mouse. The game loop consists of three functions: *setEntityState()* determines, whether a client is fulfilling the entry or exit conditions for a state. *processAssets()* activates all assets and depending on certain attributes (such as playback type, loop mode or crossfade time), the matching Pd patcher is loaded for the audio playback and signal processing. The *sendData2activeAssets()* function sends the necessary data to already activated assets. In order to enable testing on the real hardware without being forced to take a walk outside, client and simulator can communicate via OSC: after sending a register message to the RWA Creator, the simulator can send location data back to the client.

The iOS Client

The iOS client should be considered a demo app with limited functionality. Despite its unfinished state, it is capable of importing and executing an RWA script and supports binaural playback. The source code including several scripts can be downloaded at [3]. The game loop is similar to that of the simulator as described above; the C++ code was ported to the Swift programming language, audio playback and signal processing are realized with the same Pd patchers.

The client has two views: the first view lists all available scripts. Currently they have to be compiled into the app; at a later point of development, scripts will be downloaded from a corresponding RWA server. The second view allows the user to send a register message to the RWA Creator in order to receive simulated location data and start/stop the game loop. If no register message has been send, localization with GPS and WLAN starts automatically with the game loop. Additionally to longitude and latitude, the orientation of the smartphone is used to calculate the bearing between asset and client. In a future version, a small microcontroller-based headtracker will be responsible for measuring the head position.

Results

Besides many virtual walks, which were examined with simulated location data only, a real walk containing ten states with altogether 40 audio assets has been tested outdoors several times. It is set around the Zionskirchplatz in Berlin. The 5th state forces the client to play four binaural sources simultaneously; all other states vary between one and three assets.

Figure cannot be displayed temporarily (June 2019)

Figure 3: The RWA map view. The screenshot shows a test walk around the Zionskirchplatz in Berlin.

The workload for the 4-channel binaural synthesis is around 50% on an iPhone 6. The most expensive calculations are done within the two Pd objects earplug~ [14] and convolve~ (which is part of the Pd Vanilla distribution). The former implements the binaural convolution in the time domain, the latter performs an equal-segmented FFT convolution for the late room reflections. The workload could certainly be reduced, if the calculations were done with a more optimized DSP framework within a single Pd object.

If simulated position data is used, the binaural localization works flawlessly. Unfortunately, GPS and WLAN positioning is not stable enough for a static binaural scene. This could be optimized through the additional utilization of a step counter and compass – a solution which has been successfully realized in combination with Bluetooth for the Large Scale Indoortracking [15]. The libPd library runs on both platforms - iOS and OS X – mostly as expected. The only drawback during development was the discovery of stability issues on OS X, if Pd patchers were loaded and closed dynamically very often. To mitigate this problem, all patchers are loaded in advance. The additional memory required can be neglected and Pd offers the possiblity to turn off DSP processing of unused patchers with the switch~ object. Therefore this solution does not even require any more processing power.

Conclusion

In contrast to the many existing applications, the proposed software is not limited to simple location-based walks. The binaural synthesis enables the user to create augmented realty applications; state and asset attributes and the RWA scripting language (described in [4]) provide the necessary functionality for implementing game logic. The various editors are easy to use and do not require any advanced programming skills, which is especially important for artists (rather than software developers) working in this specific area. Although simulator and client are programmed in different languages, the signal processing for both is realized with the same DSP library – libPd. This eases the development of applications dramatically; if the performance of the targeted hardware is known, the development can be done almost exclusively within the simulator; of course a final test should be executed on actual hardware. A future release of the RWA Creator might include versions for browsers and/or mobile operating systems; however, due to the complexity of the software with its many editors and views, a desktop computer with at least one large display will probably remain the preferred working environment.

Literature

[1] M. Krebs, C. Stamm, T. Resch: Indoor Tracking - Technik und Kunst in engem Zusammenspiel. Fokus REPORT - FHNW (2013), 21-30

[2] Zombies, Run!, URL: https:// www.zombiesrungame.com/ [accessed 2016, March 30]

[3] Orpheo – Global Visitor Solutions, URL: http://www.orpheo.com [accessed 2016, March 30]

[4] T. Resch: RWA – A Game Engine for Real World Audio Games. Proceedings of the International Conference on New Interfaces for Musical Expression 2015 (2015), 392-396

[5] Github repository for RWA, URL: https://github.comk/funkerresch/rwaengine [accessed 2016, March 30]

[6] Textera - Audioguides, URL: http://www.textera.ch/ [accessed 2016, March 30]

[7] A. Härmä, J. Jakka, M. Tikander, M. Karjalainen, T. Lokki, J. Hiipakka, G. Lorho: Augmented Reality Audio for Mobile and Wearable Appliances, Journal of the Audio Engineering Society 52 (6) (2004), 618-639

[8] M. Hädrich, A. Lindau, S. Weinzierl: A Mobile App for Geolocalized, Dynamic Binaural Synthesis, Proceedings of the DAGA Conference 2015 (2015)

[9] Qt Homepage, URL: http://www.qt.io [accessed 2016, March 30]

[10] Libpd Homepage, URL: http://www.libpd.cc, [accessed 2016 March 30]

[11] Pure Data Homepage, URL: https://puredata.info [accessed 2016, March 30]

[12] Festival Homepage, URL: http:// http://www.cstr.ed.ac.uk/projects/festival/, [accessed 2016, March 30]

[13] R. Sproat, A. Hunt, M. Ostendorf, P. Taylor, A. Black, K. Lenz, M. Edgington: Sable: A Standard for TTS Markup, Proceedings of the Third ESCA/COCOSDA Workshop on Speech Synthesis (1998), 27-30

[14] earplug~ Download Site, URL: http://sourceforge.net/projects/pure-data/files/libraries/earplug~/earplug~-0.2.tar.gz/download, [accessed 2016, March 30].

[15] M. Krebs, T. Resch: A Simple Architecture for Server-based (Indoor) Audio Walks, Proceedings of the International Conference on New Interfaces for Musical Expression 2014 (2014), 269-272