3D audio source simulation on iOS devices

Orizio Riccardo,
Rizzini Mattia,
Zucchelli Maurizio

University of Brescia

xx july 2014

Overview

The project goal is to realize a 3D audio simulator for the iPad, in which an audio source can be moved using a graphical interface. The interface also allows to change the orientation of the user (manipulating yaw and pitch of the head).

Overview

The app is written in multiple languages:

- **Ruby**, **Matlab** A *Ruby* and a *Matlab* scripts are used to preprocess the Database
- **Pure Data** The audio I/O is managed using a *Pure Data* patch
- **C++** The functional core of the patch is a PD external written entirely by us
- **Swift** The new Apple's language for iOS devices is used to develop the graphical inferface of the app, which communicates with the PD patch

Outline

Database Preprocessing

The KEMAR database is a list of HRTF recorded using a manikin with two microphones in place of the ears.
Each HRTF is a couple of 128-samples FIR filters (one per ear), associated to the position of the source at the moment of recording.

Database Preprocessing

The database is a textual file containing the data.
The database is processed offline, using two scripts that create a C++ header in which the data are defined as instance of two global vectors and an additional vector containing the result of the *Delaunay triangulation*.

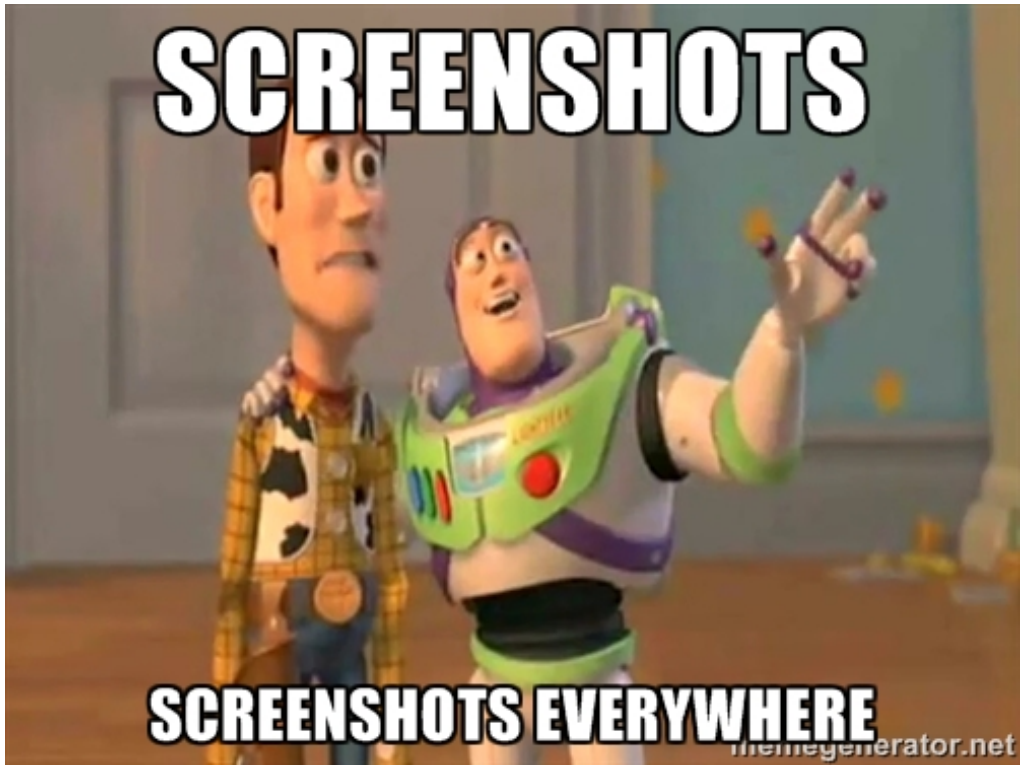
Database Preprocessing- Delaunay Triangulation

It is an algorithm that, given a set of points in a space, determines a subdivision in triangles having the points as vertices, such that no point is left inside a triangle.
This subdivision allows us to search for the three points that determine the triangle enclosing the source with less effort, compared to a brute force search.

Database Preprocessing- Delaunay Triangulation

The subdivision is performed thanks to a Matlab script.
The script is called by the Ruby one, which saves the result in the C++ header, alongside the original data.

Database Preprocessing- Delaunay Triangulation



Outline

Pure Data patch

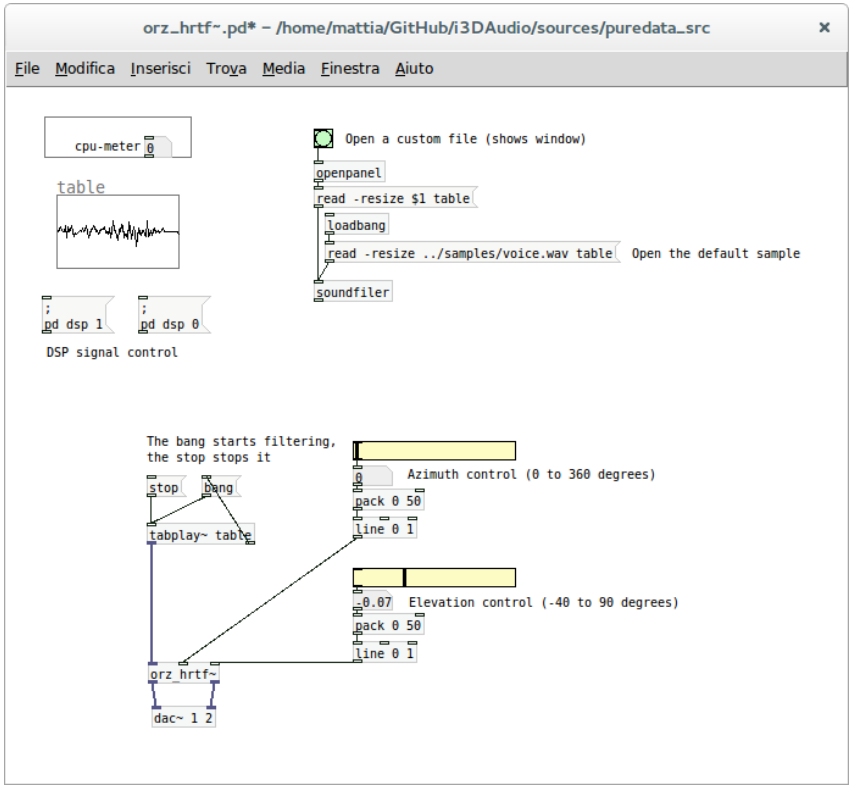


Figure 1 : The patch used to test our external

Pure Data patch

The input signal, along with the source's actual azimuth and elevation, are provided as inlets to the block orz_hrtf~, which produces as outlets the left and right channel of the filtered signal. The whole filtering is performed by that block, implemented as C++ external.

Pure Data patch

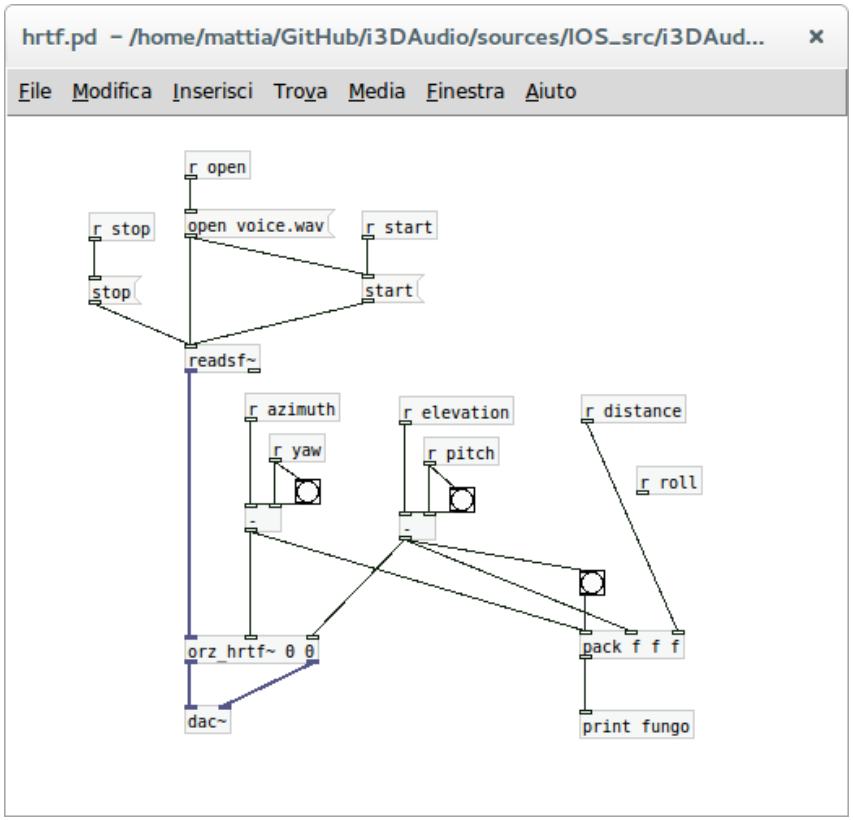


Figure 2 : The patch used by the app

Pure Data patch

The patch is similar to the one in figure , but this time the signals, as well as the parameters, are received from somwhere else (using the r blocks).
Also, the azimuth and elevation are modified accordingly to the given yaw and pitch of the head.

Outline

Processing external

A Pure Data external is structured in a very precise manner and must be written in C (or wrapped in C++).
A *setup* method is called when the block is loaded in the patch, and it must initialize inlets, outlets and callback methods.
A *new* method instantiates the internal data of the class after its creation by setup.
A callback method to handle the DSP signal must be present, in our case it is called *perform*.

Processing external

The perform method receives the signal and azimuth and elevation parameters, then proceeds in three separate steps:

- Find the points that form the triangle which encloses the source
- Determine the coefficients for the HRTF interpolation and calculate the HRTF in the source's position
- Filter the signal separately with the left and right HRTFs

Processing external- HRTF interpolation

We use the distance between the triangles and the source as a first estimate of the probability of enclosing the source, thus reducing the set of triangles in which we have to search the correct one. The correct one will be the one that produces positive coefficients for the source's HRTF interpolation.

Processing external- HRTF interpolation

The interpolation is performed as linear combination of the HRTFs of the points with the coefficients, computed as

$$g = H^{-1} \cdot s \tag{1}$$

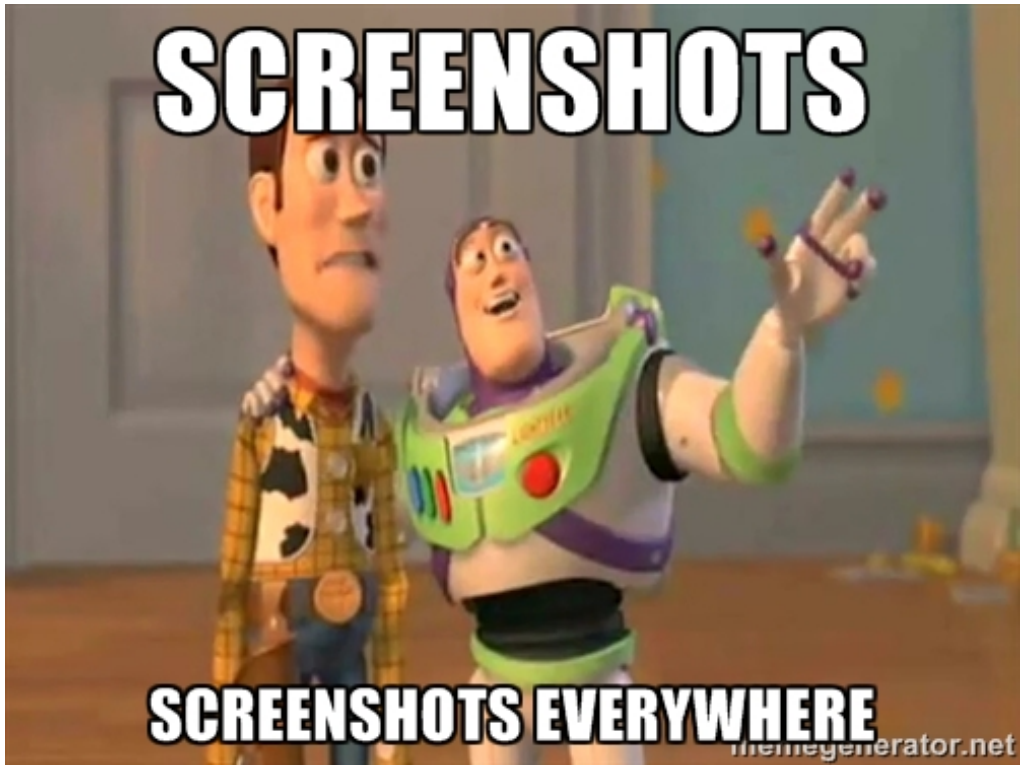
and normalized to their own summation.
Where H is a matrix having the points' coordinates as column, and s is a vector containing the source's coordinates.

Processing external- Filtering

The filtering is done by performing a convolution of the signal with the two computed HRTFs.
Past filters are kept as external's attributes to smoothen the transition from one filter to another

Outline

The iPad interface



The iPad interface

The iPad interface