3D audio source simulation on iOS devices

Orizio Riccardo, Rizzini Mattia, Zucchelli Maurizio

University of Brescia

1 August, 2014

Overview

The project goal is to realize a 3D audio simulator for the iPad.

- The interface allows to move the audio source around
- The interface also allows to change the orientation of the user (manipulating yaw, pitch and roll of the head)

Overview

The app is written in multiple languages:

Ruby, MATLAB A *Ruby* and a *MATLAB* scripts are used to pre-process 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 interface 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
- It is processed off-line and translated in three vectors containing the points' position, their HRTF and the result of the *Delaunay Triangulation*

Database Preprocessing - Delaunay Triangulation

- Determines a subdivision of the points' space in triangles
- Each triangle has 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 little effort
- The subdivision is performed thanks to a MATLAB script called by the Ruby one

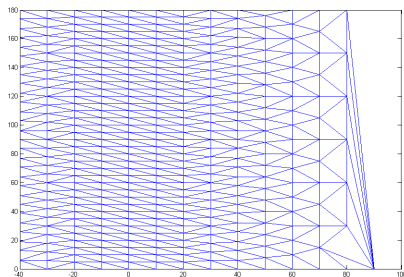# Database Preprocessing - Delaunay Triangulation

Figure 1: Triangulation

Database Preprocessing - Delaunay Triangulation

- The triangulation is in 2D, the coordinates are azimuth and elevation
- The distance is assumed to be 1 for every point
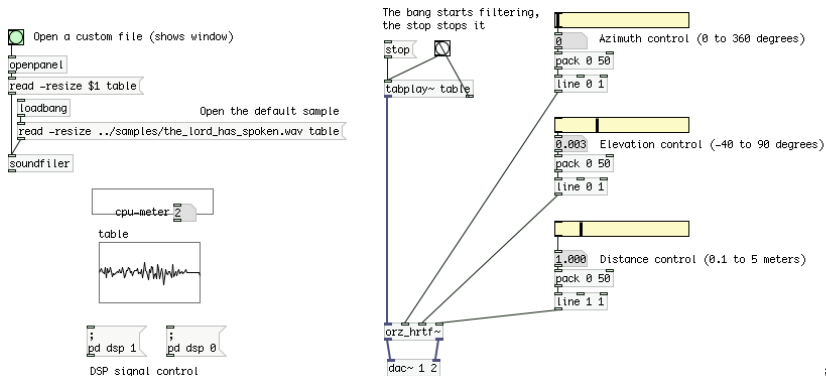
Outline

Pure Data patch

Figure 2: The patch used to test our external

Pure Data patch

- The block *orz_hrtf~* filters the signal in the given source's position
- The position is given in azimuth, elevation and distance from the user
- The resulting outlets are the left and right channel of the filtered signal
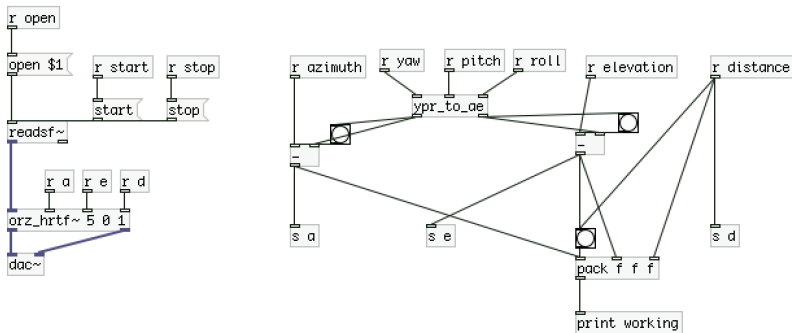
Pure Data patch

Figure 3: The patch used by the app

Pure Data patch

- The parameters are received from outside (*receive* blocks)
- The azimuth and elevation are modified accordingly to the given yaw, pitch and roll of the head
- A version with a simulated stereo source of the patch has been made
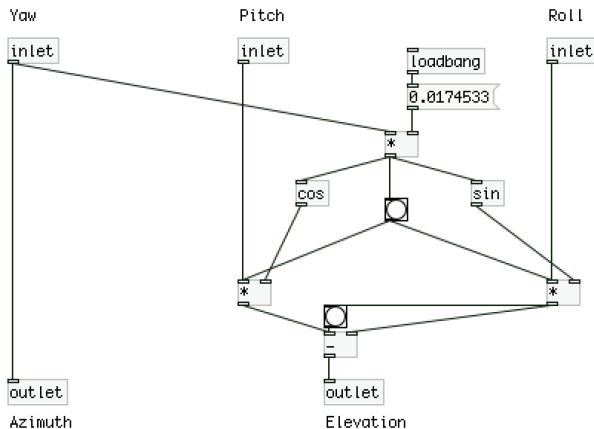
Pure Data patch

Figure 4: The sub-patch used to map yaw, pitch and roll into azimuth and elevation
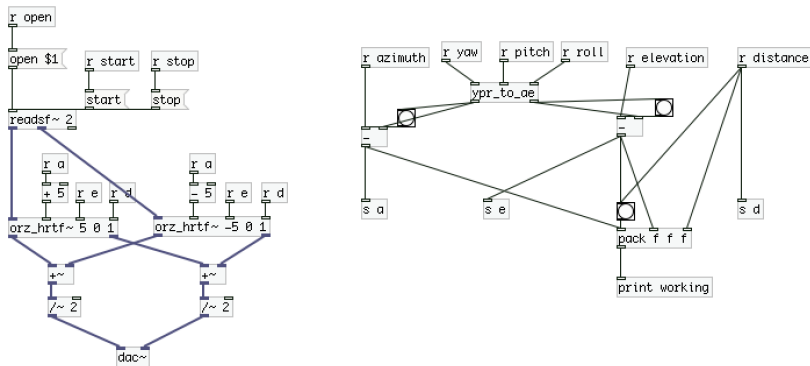
Pure Data patch

Figure 5: The simulated stereo patch used by the app

Outline

1 Overview

2 Database Preprocessing
- Delaunay Triangulation

3 Pure Data patch

4 Processing external
- Structure
- HRTF interpolation
- Filtering
- Distance

5 The iPad interface

Processing external - Structure

- A Pure Data external must be written in C or in wrapped C++
- A *setup* method is called when the block is loaded in the patch to initialize inlets, outlets and callback methods
- A *new* method instantiates the internal data of the class
- A callback method to handle the DSP signal must be present, in our case it is called *perform*

Processing external - Structure

The *perform* method works this way

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

Processing external - HRTF interpolation

- The distance between the centre of each triangle and the source is used as an heuristic to find the triangle enclosing the source

- The correct one will be the one that produces positive coefficients for the source's HRTF interpolation

Processing external - HRTF interpolation

$$\text{HRTF}_{s,l,r} = \sum_{i=0}^{2} \frac{g_i}{\left(\sum\limits_{j=0}^{2} g_j\right)} \cdot \text{HRTF}_{i,l,r}$$

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

$$H = [point_0 | point_1 | point_2]$$

Processing external - Filtering

- The filtering is a convolution of the signal with the two computed HRTFs
- Past filters are used to smoothen the transition from one filter to another

Processing external - Distance

- The HRTF is considered to be distance invariant for distances higher than one meter

- The effect attenuation caused by the distance is simulated using the *Inverse Square Law*

$$I(r) = \frac{I_0}{r^2}$$

- For lower distances the whole interpolation process becomes much harder

Outline

The iPad interface

- Three views of the user's head and the source
- The user's head can be turned and the source moved with a drag and drop
- The sound is automatically adjusted to the given position by communicating the new parameters to the PD patch
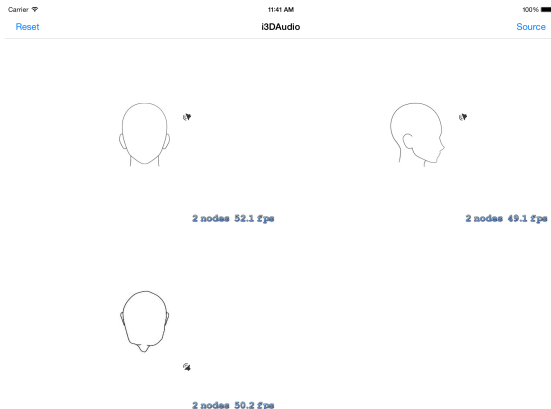
The iPad interface

Figure 6: The app running on an iPad

Conclusions

Figure 7: Suggested grade: 30