Jari Kleimola 2016-05-12

HOBA

HOBA (HRTFs On-demand for Binaural Audio) framework enables custom HRTFs in browser based spatial audio rendering. HRTFs are loaded using WAVH transfer format and rendered using Web Audio API, which is extended with two APIs: HOBA.SpatialPanner is a polyfill (or a dop-in replacement) for the forthcoming Web Audio API SpatialPanner node. It extends Web Audio API with custom HRTF rendering and spatial coordinate system. HOBA.SpatialSource is loosely modeled after gaming and VR platforms that encapsulate audio source and panner in a single entity. It may also be operated remotely, e.g., from Unity, via OpenSoundControl (OSC).

The distribution contains three demos with source comments. Note that the demos need to be served via http.

- hoba-pannerdemo explains the simple SpatialPanner API
- hoba-oscdemo goes deeper into the SpatialSource and OSC
- hoba-explorer connects 3D audio with 3D visuals

HOBA SpatialPanner API

```
var actx = new AudioContext()
var p = new HOBA.SpatialPanner(actx)
p.loadHRTF(url)                       // url to wavh file
p.elevationRange                      // {min,max} in degrees
p.xfadeduration                       // interpolation xfade in secs
p.setSPosition( a,e,r );              // spherical
p.setCPosition( x,y,z );              // cartesian
p.positionA|E|R                       // get and set
p.positionX|Y|Z                       // get and set
p.orientationX|Y|Z                    // get and set
p.setOrientation( x,y,z );            // cartesian
p.connect(target[,outport,inport])    // see web audio api docs
p.disconnect([target,outport,inport]) // see web audio api docs
p.panningModel                        // fixed to “HRTF”
p.coneInnerAngle,OuterAngle,OuterGain // see web audio api docs
p.distanceModel,rolloffFactor         // see web audio api docs
p.refDistance,maxDistance             // see web audio api docs
```

loadHRTF() returns a promise, which resolves with HRTF measurement grid geometry. The geometry contains arrays of 2D points (in azimuth – elevation space), 3D points (in Cartesian), and the triangulation of 2D space. See demos/hoba-pannerdemo.html for usage info. Implementation can be found in hoba-spatialpanner.js and hoba-waa.js files. Requires scripts

• libs/hoba-spatialpanner-api.min.js	// hoba framework
• libs/three.min.js	// for vector computation
• libs/Delaunay.min.js	// for triangulation

HOBA SpatialSource API

HOBA.SpatialSource API exposes HOBA spatialization framework to both external (OSC) and in-browser (JS) applications. OSC messages are received via websocket connection and parsed into method calls to the HOBA framework. The API is implemented in two parts: there can be any number of HOBA.SpatialSource instances, but only one HOBA.AudioListener. The description below first documents the OSC messages, then their JavaScript counterparts, and finally the top level HOBA namespace.

HOBA.SpatialSource

Sound sources are identified with the (int) N attribute in the OSC address string. For instance, "/source/0/position" sets the coordinates of sound source #0. The performance of the rendering environment defines max N. By convention, "/source/N/sound" creates a new sound source if N is new, and deletes it if the id parameter is set to -1. Setting id to 0 stops the sound. Sound id may be changed to any valid soundpool.id during runtime, even while playing.

/source/N/position	"s" a e r : float	// spherical
/source/N/position	"c" x y z : float	// cartesian
/source/N/sound	id : int	// links to soundpool.id
/source/N/gain	gain : float	// custom gain, 0..1 linear

Corresponding Javascript API is

```
var s = new HOBA.SpatialSource(id,opt) // zero-based unique id
     opt.pos: {a,e,r}                  // initial position
     opt.sound                         // links to soundpool.id
     opt.element                     // or <audio> element id
s.connect(target)                    // see web audio api docs
s.disconnect()                       // see web audio api docs
s.start()                            // start sound
s.stop()                             // stop sound
s.setSPosition( a,e,r );             // spherical
s.setCPosition( x,y,z );             // cartesian
s.sound                              // change sound id
s.gain                               // change gain, 0..1 linear
s.panner                             // get SpatialPanner node
```

Note that all HOBA.SpatialPanner attributes and methods are available via SpatialSource.panner accessor. setSPosition method returns an index to the HRTF measurement grid triangulation array, which points to a triangle used in interpolation. setCPosition returns this index embedded into an object, which also contains azimuth, elevation and distance of the point in spherical space.

The framework keeps track of all SpatialSource objects, and exposes the following functions:

- HOBA.addSource(src)   // adds SpatialSource src
- HOBA.removeSource(id)   // removes SpatialSource
- HOBA.getSource(id)   // returns SpatialSource

See demos/hoba-oscdemo.html for usage info. Implementation can be found in hoba-spatialsource.js file. SpatialSource API is built on top of SpatialPanner API, so it requires all scripts described for the panner API. In addition, following scripts are required:

- libs/hoba-spatialsource-api.min.js   // hoba framework
- libs/osc-browser.min.js   // for OSC (optional)

HOBA.AudioListener

There's only one listener in the model. The defaults for position and direction are given below in comments. HRTF is defined as a HOBA wavh file url, which in the browser can point to the cloud or to localhost. HRTF can be changed on the fly while rendering.

/listener/position	x y z : float	// [1 0 0]
/listener/front	x y z : float	// [0 0 -1]
/listener/up	x y z : float	// [0 1 0]
/listener/hrtf	url : string	

Corresponding Javascript API is

```
var a = HOBA.listener;
a.position                        // get: vector3D, set [x,y,z]
a.front                           // get: vector3D, set [x,y,z]
a.up                              // get: vector3D, set [x,y,z]
a.cartesian2spherical(x,y,z)      // returns {a,e,r}
a.spherical2cartesian(a.e.r)      // returns {x,y,z}
```

HRTF loading is routed to SpatialPanner API, which stores it in a shared prototype.

HOBA.soundpool

soundpool is a collection of samples. It is usually configured in the rendering environment (browser), but may also be maintained using OSC. Sound can be a pcm wav or in any compression format supported by the rendering environment (i.e., mp3 and aac should be ok for the browser). Note: the first id in the pool should be 1 (not 0).

For each sample:

id : int	// /source/N/sound binds to this, starts at 1
url : string	// sample file url
gain: float	// distance model scaling, 0..1 linear

soundpool may be configured using the OSC messages below.

/sound/id	url: string gain: float	// create new sound
/sound/id	(no args)	// remove sound

Corresponding Javascript API is

```
HOBA.addSound(id,url,gain=1)
HOBA.removeSound(id)
HOBA.getSoundpoolID(soundname)
```

additional HOBA namespace functions

```
HOBA.init(hrtfurl [,actx])            // framework init
HOBA.startOSC(url, callback)          // see hoba-oscdemo.html
HOBA.stopOSC()                        // stops receiving OSC
HOBA.createSpatialPanner()            // convenience function
```

HOBA.init() is the preferred way of using SoundSource API. HRTF url argument is passed to SpatialPanner.loadHRTF function, while actx argument gives an optional Web Audio API AudioContext (created internally if not given). See hoba-oscdemo and hoba-explorer for usage info.

OSC

OSC messages are delivered usually on top of UDP. Browsers cannot communicate through UDP directly, but use websockets for similar functionality. HOBA osc2ws bridge handles the required protocol conversion. Follow the steps below to install and run the bridge.

prerequisites

1. install node.js
2. open OSX terminal or Windows command prompt and type

```
cd osc2ws
npm install ws
npm install osc
```

running

```
node osc2ws.js
```

OSC messages received from port 7400 are now routed to the browser websocket port 8081. These defaults can be modified in osc2ws.js code. To test the connectivity, launch browser (Chrome recommended) and browse to demos/hoba-oscdemo.html. The demos folder also contains hoba-oscdemo.pd patch for testing.