ImmLib - A new library for immersive spatial composition

Miguel Negrão

Sonic Arts Research Center, Belfast
Support from Portuguese Foundation for Science and Technology

September 17, 2014

Overview

- ImmLib is a new software library for spatial composition with grid-based loudspeaker systems in the context of computer sound synthesis and audio processing which places emphasis on immersiveness and a global approach to space.
- It implements techniques for dealing with multiple decorrelated, but perceptually similar, sound streams spatialized at different locations in space (a grid) with the aim of creating an expanded, broad or diffuse sound source with interesting musical spatial properties.

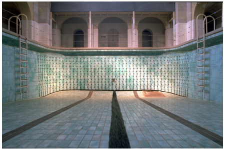
Overview

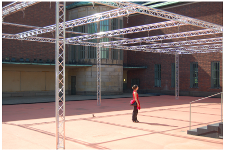
- ImmLib is a new software library for spatial composition with grid-based loudspeaker systems in the context of computer sound synthesis and audio processing which places emphasis on immersiveness and a global approach to space.
- It implements techniques for dealing with multiple decorrelated, but perceptually similar, sound streams spatialized at different locations in space (a grid) with the aim of creating an expanded, broad or diffuse sound source with interesting musical spatial properties.

Previous work - software

- Different approaches to working with sound objects created out of several individual decorrelated streams:
    - Decorrelation[Kendall(1995), Vaggione(2001)]
    - Spatial Swarm Granulation[Wilson(2008)]
    - Spectral Spatialization[Kim-Boyle(2008)]
    - Deduplication with band-pass filters, pitch-shifting or delays[McGee(2010)]
    - Spatio-Operational Spectral Synthesis[Topper et al.(2002)Topper, Burtner, and Serafin]
    - Image Based Spatialization[Iyo(2012)]
    - Particle systems

Previous work - sound works





- Examples of sound works (computer created and acoustic):
    - Terretektorh (Iannis Xenakis, 1966, For large orchestra of 88 players scattered among the audience).
    - SoundBits (Robin Minard, 2002, 576 channels of 1-bit audio, 576 piezo speakers, computer-controlled spatialization)
    - Pneumatic sound field (Edwin van der Heide, 2006, 42 pneumatic valves, computer-controlled spatialization)
    - Spaced Images with Noise and Lines (Eric Lyon, 2011, 8-channel composition, image based spatialization).

Model - overview

- ImmLib, implemented in SuperCollider, automates the process of creating decorrelated streams from a synthesis definition and provides mechanisms to create and control spatial patterns in a virtual surface by modulating synthesis parameters of the sound processes using different (but coherent) signals for each of the running instances.
- The main abstraction is the *Parameter Field* which defines ways to control the spatial patterns across space based on mathematical functions defined on a surface.

Model - overview

- ImmLib, implemented in SuperCollider, automates the process of creating decorrelated streams from a synthesis definition and provides mechanisms to create and control spatial patterns in a virtual surface by modulating synthesis parameters of the sound processes using different (but coherent) signals for each of the running instances.
- The main abstraction is the *Parameter Field* which defines ways to control the spatial patterns across space based on mathematical functions defined on a surface.

Simple example

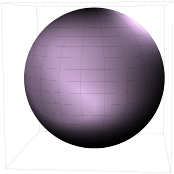


Figure : Function defined on surface

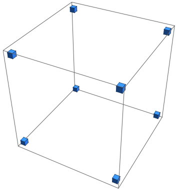
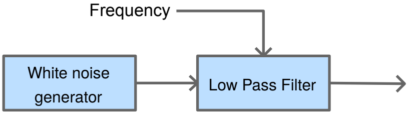
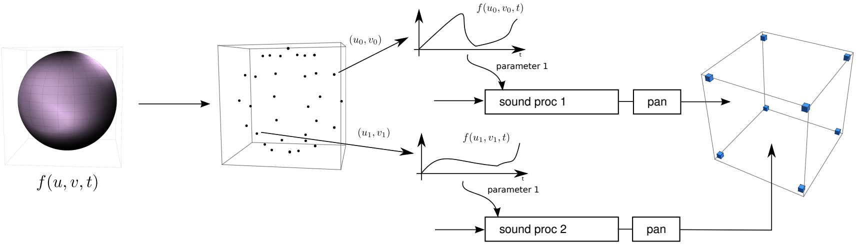


Figure : 3D speaker system

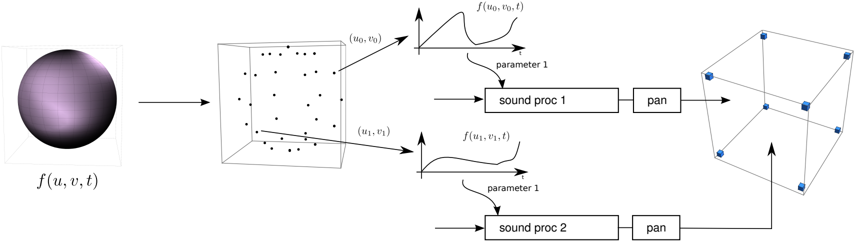


Model - sound processes - surfaces



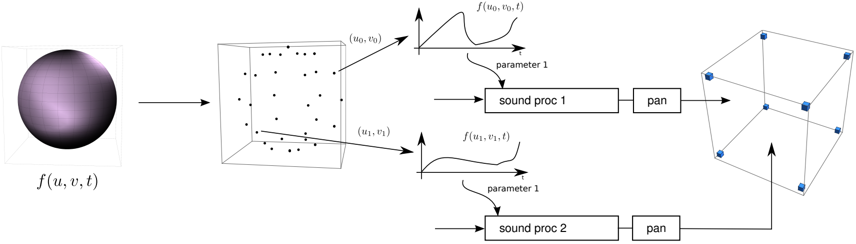
- *a* *sound process* → any type of sound synthesis or audio processing algorithm.
- Sound processes are played on a surface (e.g. a sphere, wall/plane).
- The imaginary surface is conceptualized as hovering over the loudspeaker grid.

Model - sound processes - surfaces



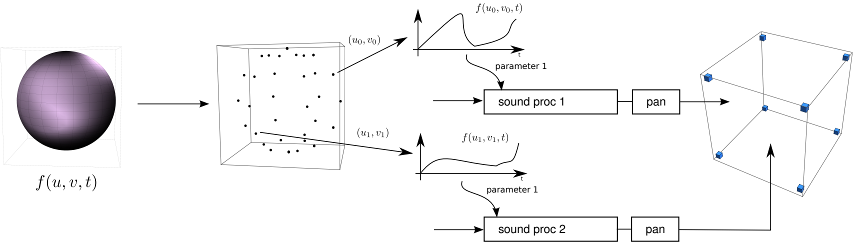
- *a* *sound process* → any type of sound synthesis or audio processing algorithm.
- Sound processes are played on a surface (e.g. a sphere, wall/plane).
- The imaginary surface is conceptualized as hovering over the loudspeaker grid.

Model - sound processes - surfaces



- *a sound process* → any type of sound synthesis or audio processing algorithm.
- Sound processes are played on a surface (e.g. a sphere, wall/plane).
- The imaginary surface is conceptualized as hovering over the loudspeaker grid.

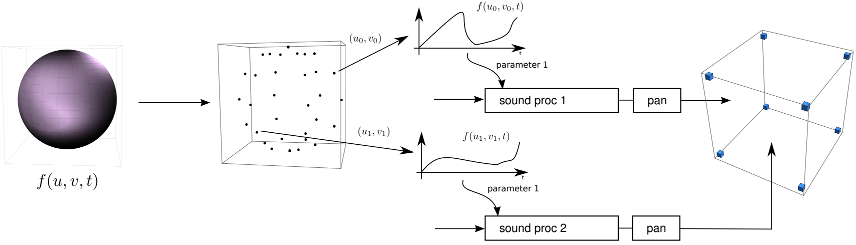
Model



- A sound process has *parameters* (e.g. frequency, amplitude, etc)
- parameter ↔ *parameter field* : Each parameter can be associated with a function $f(u,v,t)$ defined on the surface. e.g.:

$$f(u,v,t,c) = sin(2\pi c(u+v)t)$$

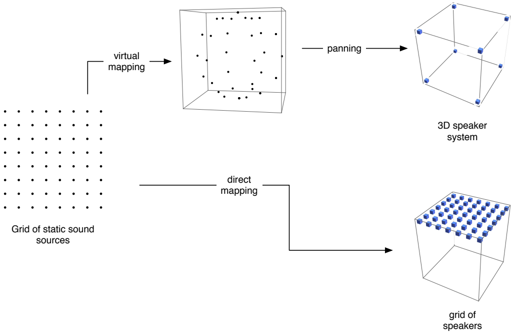
Model



- A sound process has *parameters* (e.g. frequency, amplitude, etc)
- parameter ↔ *parameter field* : Each parameter can be associated with a function $f(u,v,t)$ defined on the surface. e.g.:

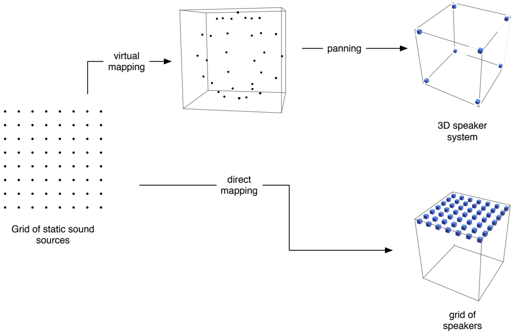
$$f(u,v,t,c) = sin(2\pi c(u+v)t)$$

Model - discretization



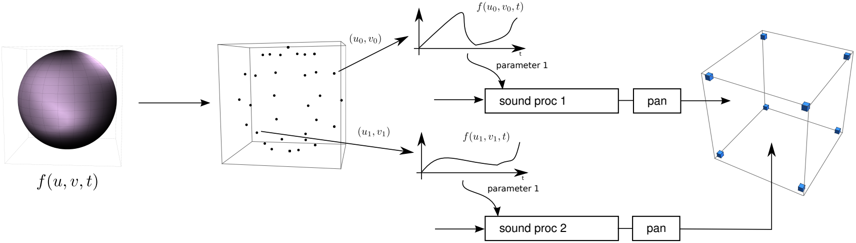
- The surface is discretized into multiple points.
- Each sound process plays simultaneously at all the points of the grid (loudspeakers or virtual panning).

Model - discretization



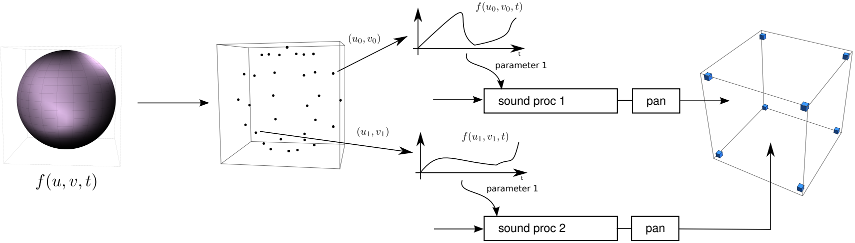
- The surface is discretized into multiple points.
- Each sound process plays simultaneously at all the points of the grid (loudspeakers or virtual panning).

Model



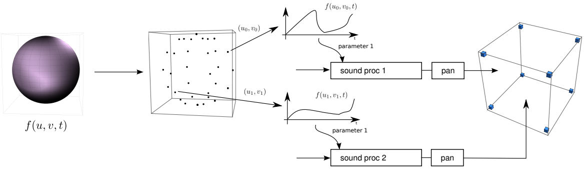
- At each point of the grid the parameter field is evaluated returning a (different) function of time (i.e. an envelope) $f(u_0, v_0, t)$.
- Each envelope is used to modulate the parameter of the corresponding synthesis process.

Model



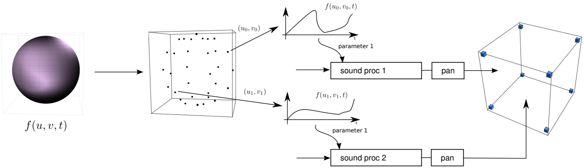
- At each point of the grid the parameter field is evaluated returning a (different) function of time (i.e. an envelope) $f(u_0, v_0, t)$.
- Each envelope is used to modulate the parameter of the corresponding synthesis process.

Model



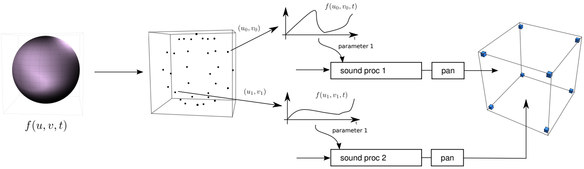
- The relationship over time of the different envelopes (created from the same parameter field) creates spatial patterns.
- By spatial patterns we mean the suggested movement or spatial impression that is perceived when one hears a complex sound source composed of many similar individual elements changing in time and spread in space .
- If the function is continuous, we usually get coordinated behaviour between all the envelopes.

Model



- The relationship over time of the different envelopes (created from the same parameter field) creates spatial patterns.
- By spatial patterns we mean the suggested movement or spatial impression that is perceived when one hears a complex sound source composed of many similar individual elements changing in time and spread in space .
- If the function is continuous, we usually get coordinated behaviour between all the envelopes.

Model



$f(u,v,t)$

(u_0,v_0)

(u_1,v_1)

$f(u_0,v_0,t)$

parameter 1

sound proc 1

pan

$f(u_1,v_1,t)$

parameter 1

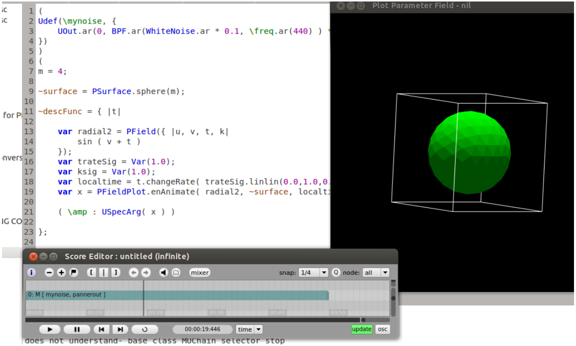
sound proc 2

pan

- The relationship over time of the different envelopes (created from the same parameter field) creates spatial patterns.
- By spatial patterns we mean the suggested movement or spatial impression that is perceived when one hears a complex sound source composed of many similar individual elements changing in time and spread in space .
- If the function is continuous, we usually get coordinated behaviour between all the envelopes.

ImmLib - Implementation

- SuperCollider library
- Standalone visualizer (OpenGL + Haskell)

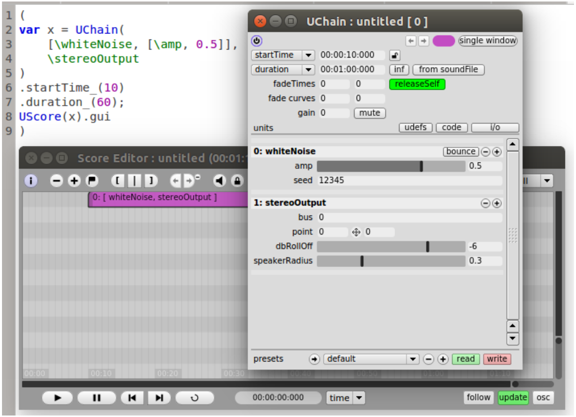


```
(
Udef(\mynoise, {
    UOut.ar(0, BPF.ar(WhiteNoise.ar * 0.1, \freq.ar(440) ) *
})
)
(
m = 4;

~surface = PSurface.sphere(m);

~descFunc = { |t|

    var radial2 = PField({ |u, v, t, k|
        sin ( v + t )
    });
    var trateSig = Var(1.0);
    var ksig = Var(1.0);
    var localtime = t.changeRate( trateSig.linlin(0.0,1.0,0.
    var x = PFieldPlot.enAnimate( radial2, ~surface, localti

    ( \amp : USpecArg( x ) )

};
```

```
does not understand- base class MUChain selector stop
```

ImmLib

- Implemented on top of Unit Lib (from Game of Life WFSCollider) + FPLib.



```
(
var x = UChain(
    [\whiteNoise, [\amp, 0.5]],
    \stereoOutput
)
.startTime_(10)
.duration_(60);
UScore(x).gui
)
```

UChain : untitled [0]

startTime 00:00:10:000

duration 00:01:00:000

fadeTimes 0 0

fade curves 0 0

gain 0

units

0: whiteNoise

amp 0.5

seed 12345

1: stereoOutput

bus 0

point 0 0

dbRollOff -6

speakerRadius 0.3

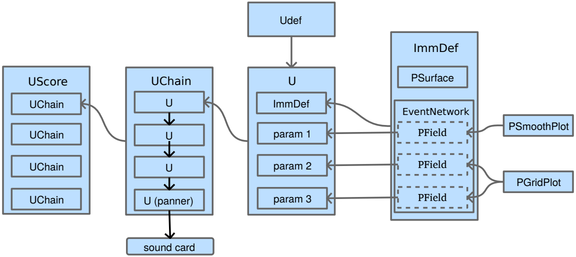
presets default

Score Editor : untitled (00:01:

0: [whiteNoise, stereoOutput]

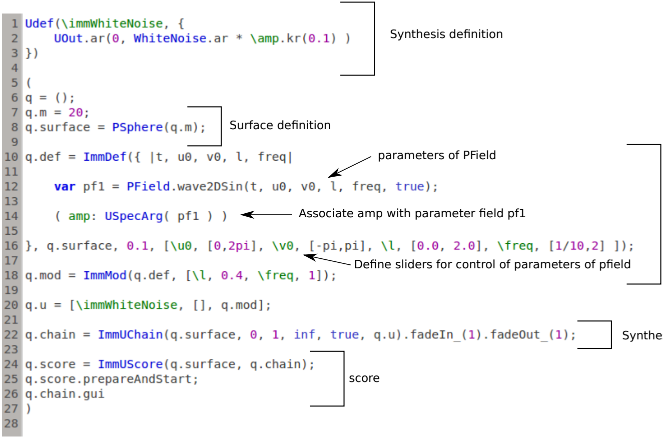
00:00:00:000

SuperCollider implementation



- An ImmDef connects parameter fields with synthesis parameters of the unit (args).
- Surface → PSphere, PPlane (grid points + functions to calculate distances in the surface)

Syntax



```
Udef(\immWhiteNoise, {
    UOut.ar(0, WhiteNoise.ar * \amp.kr(0.1) )
})

(
q = ();
q.m = 20;
q.surface = PSphere(q.m);

q.def = ImmDef({ |t, u0, v0, l, freq|

    var pf1 = PField.wave2DSin(t, u0, v0, l, freq, true);

    ( amp: USpecArg( pf1 ) )

}, q.surface, 0.1, [\u0, [0,2pi], \v0, [-pi,pi], \l, [0.0, 2.0], \freq, [1/10,2] ]);

q.mod = ImmMod(q.def, [\l, 0.4, \freq, 1]);

q.u = [\immWhiteNoise, [], q.mod];

q.chain = ImmUChain(q.surface, 0, 1, inf, true, q.u).fadeIn_(1).fadeOut_(1);

q.score = ImmUScore(q.surface, q.chain);
q.score.prepareAndStart;
q.chain.gui
)

```

Synthesis definition

Surface definition

parameters of PField

Associate amp with parameter field pf1

Define sliders for control of parameters of pfield

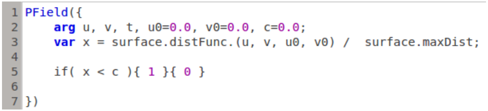
parameter fields

Synthesis processes

score

Parameter Fields

- Can be defined explicitly by functions:



```
PField({
    arg u, v, t, u0=0.0, v0=0.0, c=0.0;
    var x = surface.distFunc.(u, v, u0, v0) /  surface.maxDist;

    if( x < c ){ 1 }{ 0 }

})
```

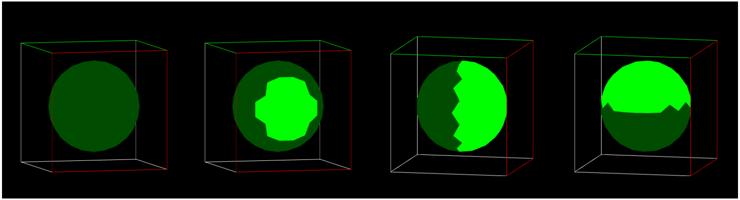
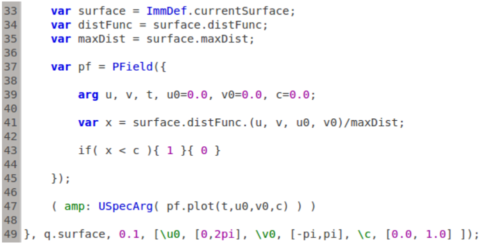


Figure : $(u_0, v_0) = (0, 0),\ c = 0,\ c = \frac{1}{4},\ c = \frac{1}{2}$and $(u_0, v_0) = (0, \frac{\pi}{2}),\ c = \frac{1}{2}$

Parameter Fields - Interaction

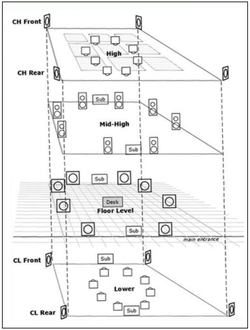
- Parameters u_0, v_0 and c can be changed in real-time from GUI, MIDI, OSC, etc.



```
    var surface = ImmDef.currentSurface;
    var distFunc = surface.distFunc;
    var maxDist = surface.maxDist;

    var pf = PField({

        arg u, v, t, u0=0.0, v0=0.0, c=0.0;

        var x = surface.distFunc.(u, v, u0, v0)/maxDist;

        if( x < c ){ 1 }{ 0 }

    });

    ( amp: USpecArg( pf.plot(t,u0,v0,c) ) )

}, q.surface, 0.1, [\u0, [0,2pi], \v0, [-pi,pi], \c, [0.0, 1.0] ]);
```

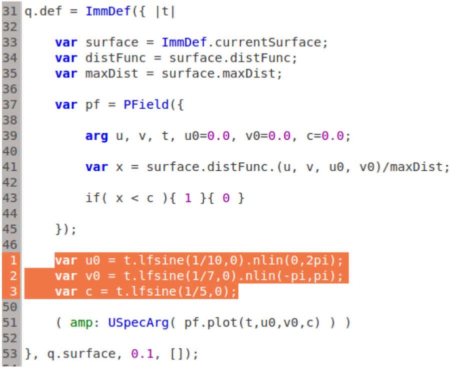
Parameter Fields - Interaction

- Parameters u_0, v_0 and c can be changed in real-time from GUI, MIDI, OSC, etc.
- demo 1 - Sonic Lab, 32 speakers



Parameter Fields - Modulation

- Parameters u_0, v_0 and c can can also be modulated with sine or saw waves or envelopes.



```
q.def = ImmDef({ |t|

    var surface = ImmDef.currentSurface;
    var distFunc = surface.distFunc;
    var maxDist = surface.maxDist;

    var pf = PField({

        arg u, v, t, u0=0.0, v0=0.0, c=0.0;

        var x = surface.distFunc.(u, v, u0, v0)/maxDist;

        if( x < c ){ 1 }{ 0 }

    });

    var u0 = t.lfsine(1/10,0).nlin(0,2pi);
    var v0 = t.lfsine(1/7,0).nlin(-pi,pi);
    var c = t.lfsine(1/5,0);

    ( amp: USpecArg( pf.plot(t,u0,v0,c) ) )

}, q.surface, 0.1, []);
```

- demo 2

Predefined Parameter Fields

- The library comes supplied with a number of predefined pfields:

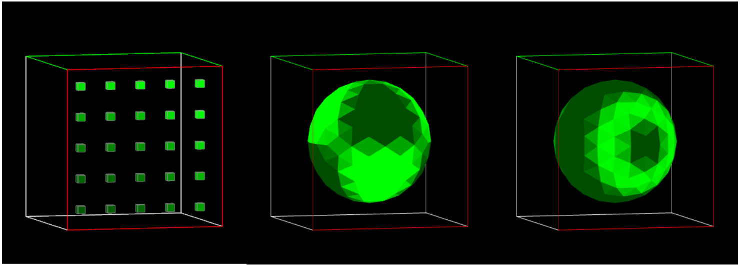
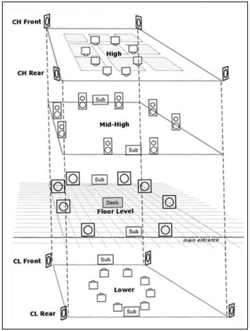
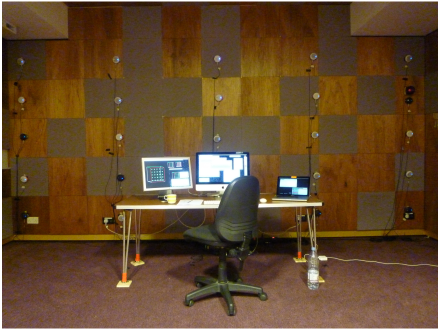


Figure : gradient, spherical harmonics and wave2DSin

- demo 3

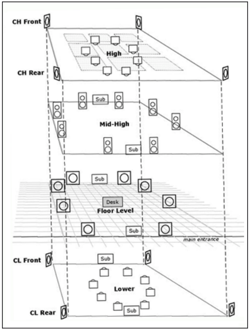
Empirical evaluation

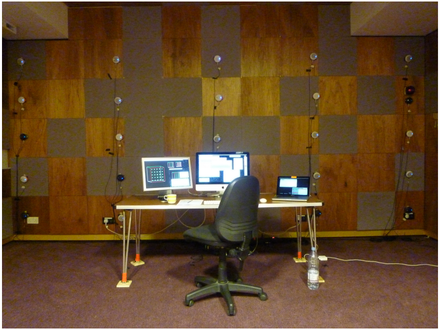




- Sonic Lab of the Sonic Arts Research Center (Belfast)
  - 4 rings of 8 speakers = 32 speakers; surface: sphere.
- 5 x 5 grid of portable battery-powered speakers mounted on a wall; 25 speakers; surface: plane.

Empirical evaluation





- Each pfield was experimented with using different sound processes (noisy, pitched, synthetic, file based, etc.) and different values of the pfield parameters.
- 3 composers at SARC have started using the library for their own work.

observations

- The mono sound processes must be decorrelated at each point (otherwise precedence effect destroys immersiveness):
  1. Non-deterministic unit generators (clicky sounds work well).
  2. Deterministic UGens with large enough range for the parameters.
  3. file playback:
     - send through decorrelation unit
     - different start times
     - further processing taking into account 1 and 2.

observations

- What a plot shows not always corresponds to what one hears.
- The agreement between the plot and what is heard is higher with slower movements.

observations

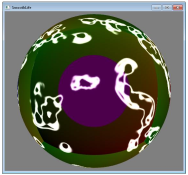
- Which parameter a pfield modulates makes a big difference.
    - modulate amplitude -> move the sound
        - going from point *a* to point *b*
        - going from nowhere to everywhere
        - bifurcating
    - modulate another parameter -> move a property of a sound (e.g. brightness).

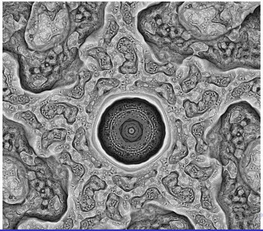
Future Work

- Adding more pfields to the library.
- Adapting it to make use of other panning systems (Ambisonics, binaural).
- Events: Use pfields as probability density functions for event creation.

Future Work

- The system could be easily extended to use 3D grids of speakers (e.g. 5 x 5 x 5 with 2m spacing).
  - $f(u,v,t) \to f(x,y,z,t)$
  - Direct output to speaker or Distance-Based Amplitude Panning.
- Complex systems (Reaction-diffusion, Multi-Scale Turing patterns, Continuous Cellular Automata) - rich behaviour from few parameters.





Thank you

- Thank you.

Image-based spatialization.
In *Proceedings of the International Computer Music Conference*, pages 200–203, Ljubljana, Slovenia, 2012.
URL http://www.icmc2012.si/.

Gary S. Kendall.
The decorrelation of audio signals and its impact on spatial imagery.
Computer Music Journal, 19(4):71–87, December 1995.
ISSN 0148-9267.
doi: 10.2307/3680992.
URL http://www.jstor.org/stable/3680992.

D. Kim-Boyle.
Spectral spatialization - an overview.
In *Proc. Int. Computer Music Conf*, Belfast, 2008.
doi: http://hdl.handle.net/2027/spo.bbp2372.2008.086.

URL http://classes.berklee.edu/mbierylo/ICMC08/defevent/papers/cr1549.pdf.

Ryan McGee.
Sound Element Spatializer.
MS thesis, University of California, 2010.
URL http://soundspatializer.com/writing/McGee_SES_Thesis.pdf.

D. Topper, M. Burtner, and S. Serafin.
Spatio-operational spectral (sos) synthesis.
In *Proceedings of the International Conference on Digital Audio Effects*, Hamburg, Germany, 2002.
URL http://www2.hsuhh.de/ant/dafx2002/papers/DAFX02_Topper_Burtner_Serafin_SOS_synthesis.pdf.

H. Vaggione.

Composing musical spaces by means of decorrelation of audio signals.
In *Proceedings of the DAFx Conference on Digital Audio Effects*, 2001.
URL http://www.csis.ul.ie/dafx01/proceedings/papers/Addendum1%20-%20Vaggione.pdf.

S. Wilson.
Spatial swarm granulation.
In *Proceedings of the International Computer Music Conference*, Belfast, N. Ireland, 2008.
URL http://eprints.bham.ac.uk/237/1/cr1690.pdf.