Project Aims

A Graphical Tool for Hybrid Modelling Auralisation

- produce a program to simulate acoustics of virtual environments
  - a realistic modelling reverb
  - offline, not real-time
- should be usable by musicians, not just programmers
  - needs to be friendly and fast

Note: software generates impulse responses for use with convolution reverbs

Background

- my undergraduate research concentrated on 'geometric' solutions to this problem
  - modelling sound as rays

Note: rays are

emitted in straight lines from some source

perfectly reflected at boundaries

An impulse (single sample) is added to the output file for each reflection

time depends on the distance the ray has travelled, speed of sound

volume depends on the materials at the reflections, attenuation due to air

Ray Tracing

- works great for high frequencies
- works less well for low frequencies
  - the wavelength is of the same order as the size of the reflecting surfaces

Note: 20Hz wave has a wavelength of 17m, most rooms aren't made up of 17mˆ2 surfaces

also can't accurately simulate wave effects

diffraction, cancellation

tested on drum kit samples, bass drums wouldn't behave

either overpowering or not present

Usability Issues

- wasn't very friendly
  - *musician*: 'I need a reverb'
  - *me*: 'Why not try the program I'm writing, it's got a simple command-line interface, you just give it two config files and a model file…'
  - *musician*: *snoring sounds*

Note: musicians want to be doing the music

not reading manual files for UNIX programs

What's the Solution?

- combine the ray tracer with another model which can handle low frequencies
- add a graphical interface

Note: My focuses:

building this secondary model

making sure it's as flexible as the ray tracer

mic modelling

material modelling

combining the two models

the GUI is less of a 'focus' - I'm not looking to innovate here, but it's necessary if I want the program to have a future

The Waveguide Model

- approximates the 3D wave equation (good)
- quite expensive to calculate at high frequencies (bad)
  - but we don't need high frequencies

Note: behaves like air in an enclosed space would

complexity scales with the cube of the maximum modelled frequency

relatively cheap for low frequencies

takes for ever if you model the whole audible spectrum in a big room

How Does it Work?

- divide the air in the room into equally-spaced 'nodes'
  - nodes are linked to adjacent nodes

Note: can divide the room differently depending on how many connections each node should have

cubic layouts have six adjacent nodes, two along each axis

octahedral, dodecahdral topologies with 8, 12 nodes

tetrahedral topology has 4

- each node has an associated air pressure
- the next (at the next time-step) air pressure at a given node depends on
  - the previous pressure at that node
  - the current pressures at adjacent nodes
- for this reason…
- tetrahedral topology is (arguably) most efficient
  - only four connections per node
  - fewer memory lookups, additions per node

tetrahedral mesh structure

Note: it can be cubically aligned, so it's still quite simple to generate the mesh

finding node neighbors is a bit more complicated than in other topologies

- update all the pressures in the room in one go
- gives us a map of the air pressure in the room at sequential steps in time
- have to update all nodes once per time-step
- write the air pressure at any node to an audio file
  - this is the impulse response or reverb

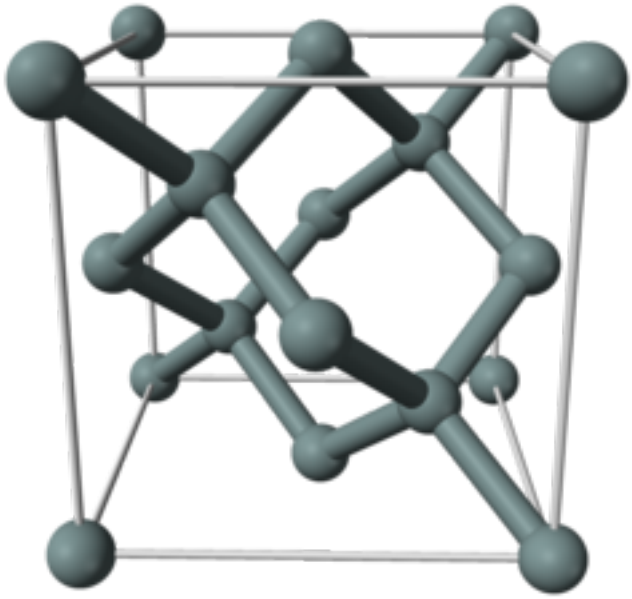

Figure 1:

Optimisation

- nodes can be updated in any order or even at the same time
- graphics hardware is designed for doing lots of calculations in parallel
- so let's do that

Unique Features

- other auralisation software (odeon, catt-acoustic, EAR) tends to use geometric methods on the CPU
- hybrid approach on the GPU is much rarer
- once complete, should be faster and have better low-frequency performance than comparable systems
- open-source, GPL licensed! `github.com/reuk/waveguide`

How Far Have I Got?

library for generating and simulating tetrahedral waveguide meshes

post-processor for directional receivers

run it several times on the same waveguide output!

(multichannel output from *coincident* mic capsules needs only one waveguide run)

refactoring

improving the ray tracing library, removing duplication between the two libraries

visual debugger * view the waveguide output * and (some of the) diffuse ray paths * this may become the graphical interface for the program

What's in the future?

boundary modelling

automatically combining outputs, rather than doing it by ear testing!

Quick Demo?