IMPLEMENTATION AND EVALUATION OF DYNAMIC LEVEL OF AUDIO DETAIL

GABRIEL DURR[1], LYS PEIXOTO, MARCELO SOUZA, RAISA TANOUE AND JOSHUA D. REISS

[1] *Centre for Digital Music, SEECS, Queen Mary University of London, London, UK*
g.carvalhodurr@stu13@qmul.ac.uk ; l.meneguellipeixoto@qmul.ac.uk ; m.soaresdesouza@stu13@qmul.ac.uk ; r.tanoue@stu13@qmul.ac.uk ; joshua.reiss@qmul.ac.uk

Sound synthesis involves creating a desired sound using software or algorithms and analysing the digital signal processing involved in the creation of the sound, rather than recording it. However, synthesis techniques are often too computationally complex for use in many scenarios. This project aims to implement and assess sound synthesis models with dynamic Level of Audio Detail (LOAD). The manipulations consist of modifying existing models to achieve a more or less complex implementation while still retaining the perceptual characteristics of the sound. The models implemented consist of sine waves, noise sources and filters that reproduce the desired sound, which could then be enhanced or reduced to provide dynamic LOAD. These different levels were then analysed in the time-frequency domain, and computational time and floating point operations were assessed as a function of the LOAD.

INTRODUCTION

Sound synthesis involves creating a desired sound using software or algorithms, rather than recording it. One of the most common and important applications is game audio, where synthesized sounds take up less memory than samples, are far less repetitive, and can be manipulated in far more creative ways.

Traditionally, game audio has been performed using samples of pre-recorded sounds. Early game consoles and personal computers had synthesiser chips that produced sound effects and music in real-time. However, as sample technology matured it overtook real-time synthesis because of its perceived realism [1].

While sample-based data requires most of the work and decisions to be done in advance, prior to execution, real-time sound synthesis is highly dynamic and flexible and many decisions are postponed until runtime. Unlike sampled sound, synthesised sound (and more generally, procedural audio as opposed to data driven audio) has a variable cost; the more complex the sound, the more work it requires.

A variable cost approach is well-established in rendering computer graphics, especially in games. Level of Detail (LOD) refers to the complexity of the rendering of an object or scene. Frame rendering time and memory utilization are improved in interactive visualization using simpler representations of an object [2]. Dynamic Level of Detail refers to the fact that the detail in the rendering can change based on how close a character is to whatever is being rendered.

In an early work [3], Clark discussed different approaches to reduce computation cost when rendering a scene and proposed an object hierarchy to represent a scene in different levels of detail. He took into consideration parameters such as how much space on the screen the object occupies, the field of view of the scene and whether or not the object or the viewer is moving.

We are concerned with investigating, implementing and testing an approach to dynamic Level of Audio Detail (LOAD). That is, synthesizing a sound such that we can choose how accurate and complex the synthesis is. By providing a variable level of detail, a complex sound scene can be synthesised with much less computation and storage than would be required to deliver, process and render a large number of samples.

Previous work approached the application of level of audio detail by reducing computation in sample-based spatialized rendering of sounds [4, 5] and using corpus-based granular synthesis [6].

Tsingos et al [4] proposed a pipeline composed of four steps for each time frame of audio processing. Firstly, all sound sources are sorted based on their binaural loudness. Perceptually inaudible sources that would be masked by the combination of other sources are then culled from the mix. A similar technique has been proposed for multitrack audio production [7, 8]. In step two, these remaining sources are then grouped into a predetermined number of clusters and a representative point source is created for each non-empty cluster. In step three, an equivalent sound signal is generated for each cluster and for each source, a number of operations is applied to the audio data. Finally, the pre-mixed signals and their representative point locations are used to feed audio rendering hardware or be rendered in software. The main limitation to this method is that it performs well for a few hundred sound sources, but cannot be scaled easily to the thousands.

In [5], Fouad et al propose that the real-time generation of the sound environment respects a predetermined computation time budget using buffers. They use three factors (the listener’s gaze, the intensity of the sound and the age of the sound) to prioritize the sound. The higher the priority the more detailed the sound will be. Once the buffers are filled the remaining samples that are missing are found through interpolation.

Schwarz et al [6] proposed three levels of detail for a scene, similar to our approach. However, their approach seeks to bring sound and image together, making it respond visually and sonically to a common stimulus instead of triggering a sound effect from a visual event or vice-versa, while our focus is only on synthesising the sounds.

The paper is organised as follows. Section 1 provides an analysis of sound synthesis engines and their effectiveness in applying LOAD. Section 2 presents how LOAD was applied to the chosen models. In Section 3, we analyse the results of sound synthesis with different models and different levels of LOAD. Section 4 describes a listening test that compared the perceived realism of the models with varying LOAD against each other and against recorded samples. Finally, Section 4 provides conclusions and a critical analysis, as well as discussion of future research directions.

1 ANALYSIS

Using [1] as a guideline, we aim to reconstruct and modify existing sound models to incorporate LOAD. Models described in [1] have been enhanced or extended previously by [9], [10] and [11], but to the best of our knowledge, not for the purpose of providing dynamic LOAD. The goal here is to have models close enough to the original sound so that they still sound like the effect being recreated but with less computation required, and to analyze how complex it would be to design an even more realistic model.

The models are studied regarding their components, such as oscillating waves, filters and operations. Our approach involves modifying frequencies, gains and even incorporating new blocks of codes leading to a better quality of sound as well as withdrawing some parts to keep it realistic but less complex.

Table 1 summarizes the elements of each model in [1], in order to categorize them in terms of the ease, appropriateness and effectiveness of applying LOAD. Not shown are those models that produce exactly the sound to be modified (pedestrian crossing, phone tone, DTMF tone, alarm…), where LOAD would most likely not be applicable. This table was created so we could analyze objectively the aspects of the models to critically choose the best ones to work with. The criteria of the choice was how difficult it would be to implement LOAD and how satisfying the outcome would be if we did.

2 IMPLEMENTATION

2.1 Model implementation

Based on the complexity of the models in Table 1, and the potential effectiveness and ease of implementing LOAD, three models were chosen for investigation: fire, bubbles, and wind. Fire was implemented using Simulink, a block diagram and model-based design tool for Matlab, chosen because of its simplicity and broad functionality. Also, performance tests such as computational time, spectrogram visualisations, number of operations and memory usage are easy to implement with Matlab. Bubble and wind models were implemented using PureData, an open source visual, block-based programming language mainly used for processing and generate sounds. It was chosen for these models because it allowed direct extension and adaptation of the code provided in [1].

Fire is a complex sound composed of many features. The synthesis model is mostly white noise filtered to achieve the right frequency levels. It is created after the sum of three different sounds: lapping, crackling and hissing. Although the sound of fire has more elements, these represent the dominant components. Hence, its implementation is made by generating all these features separately and then adding them together.

For bubbles, we considered bubbles rising from underwater until it reaches the surface where they pop and ring. The pitch of the bubble sound depends on its size; the larger the bubble the lower the sound. The model is based on two different elements, each one with a different envelope, pitch and amplitude of the bubbles. They depend on the size and quantities of bubbles. Also, different tones are used at the same time for the pitch as there are many different bubbles flowing.

Like the fire model, wind is composed of a series of features, such as a whistle, a howl, leaves moving, etc. To emulate this, we used noise and filters that generated pseudo-chaotic signals. Echo should appear since the direction of the wind depends on the obstacles. Also the air velocity should vary and that was obtained with low-frequency noise. The band-pass filters were responsible for providing the "whistle" while the wind through the leaves was obtained clipping sinusoidal waves.

2.2 Applying Level of Audio Detail

Each of the models has its own implementation and each generated sound has its own characteristics. So a different approach to applying LOAD was implemented for each model.

Table 1: Sound synthesis models from [1], their implementation, how LOAD might be applied, and the effect of applying LOAD.

Model	**Implementation**	**How to Apply LOAD?**	**Effect of LOAD**
Police	2 oscillators + clipping + band pass filter	Adding filter to simulate distortions of diaphragm, delay to simulate buildings	Computation becomes more complex
Telephone Bell	Few oscillators	Increasing or decreasing number of harmonics	
Fire	Combination of filtered white noise	Adding more or less effects, besides changing aspects of each effect	With each effect added more computation required to generate the sound
Bubbles	Sine wave generator + envelope generator	Changing some constants; adding the same model with different frequencies	Just changing a small detail
Running Water	Noise modulation (AM & FM)	Increasing or decreasing the number of rising sine waves and/or changing the parameters than it can sound like a harder or a softer flow	It could change the quality of the sound & switch between different kinds of running water
Pouring	"Bubbles" + "Running water" + Band pass filters (resonance)	Adding more variation in bubble sounds through adjustable number of filters	More computation because of the filters
Rain	Waveshaping noise into parabolic pulses	Change places drops fall; add coloured noise; change frequencies, resonance, threshold & amplitude; add sound of running water & wind; fade unused scenes as rain continues	More computation due to the different environments
Electricity	Mix of oscillators at close frequencies modulating a chirp impulse + short time comb filters + short impulses, noise & resonant bank filter	By adding LOAD to the different aspects that forms the sound, like adding noise, filtering & modulation.	
Thunder	Noise sources + delays + waveshaping + delay-based echo/reverb	Reducing echo component of sound, do not use polyphonic strike sound generator as it can only be heard in small distances from the thunder	Without echo, thunder would be less realistic but it would reduce CPU processing. Without polyphonic strike generator thunder would not sound perfect at short distances
Wind	Noise & filters (Low-frequency noise + amplitude-modulated wideband noise + narrow band pass filters)	Not using the subtle amplitude & frequency modulation/ Not delaying the excitations/Not use tree leaves sound	Without amplitude & frequency modulation wind would have less realisticm whistling. Without delay in excitations, stereo image would not be so realistic. Tree leaves sound not as good as described, so removing it is not a problem. Sound elements present in wind vary with ambience.
Switches	Parallel band pass filters + short delays + noise-like sources	Add some body resonance; Build more abstractions to change the type of the switch	More computation due to different switches
Clocks	Controlled "switches"	Adding more complexity to the control code, simulating a big or small clock body by adding delay & filtering.	More computation due to a more complex control code
Motors	Envelope generator + modulated noise source + raised cosine waveform	We could remove the noisy clicks, in this case we would not use a modulated noise source, decreasing CPU processing.	The motor sound would be less realistic but only when heard closely, the clicks can't be heard in a greater distance.
Cars	Phase splitting + wrapping + delays + filters (+ timewarp the waveguide + small noise)	Add inertia; Add timing & pulse width jitter; Define a set of different overtones; Set the controls differently	More computation when adding different controls & inertia
Fans	Noise source modulated with a narrow pulse + mildly resonant	Add a parameter to alter the mix of pulse, noise & Doppler shift, depending	Different sounds depending on observer's angle as well as produce other static

	band pass filter	on the observer's angle.	machinery sounds.
Jet Engine	5 oscillators (partial additive synthesis) + cascade of filters & nonlinear functions (noise source)	Varying control input of filter for flame. Adding more or less sinusoids to simulate the turbine. Varying the control input to simulate speed.	
Helicopter	Pulse generator & waveguide (engine) + impulse generator, flat amplitude noise source & movable comb delay (main rotor) + fixed pulse generator & band pass filter (tail rotor) + 3 sine oscillators (gearbox) + steep low pass filtered noise (distance filter)	Change the model related to the movement of the helice; Scale components differently;	Simple changes since you just add the same codes again or change scales based on the model used.
Footsteps	2 polynomial GRF curve generators + Noise filtered & clipped (+ envelope)	Use less or not use synthesisers to produce mixture of grass, soil, gravel or any combination	The transitions across changing terrains would not be too smooth. In the code it would be enough to remove some parts.
Insects	- Field cricket: synchronous AM (source – one phasor)	Add context	Computation will be needed, as the environment needs to be set
	- Field cricket 2: pulse + high resonance band pass filters		
	- Cicada: filtered noise source modulated with pulse wave		
Birds	Pulse waves + ring modulation for FM carrier + two parallel band pass filters	Birds are really complex animals, to be more accurate it would require intricate parameterisation of the syntax model	Really complex computation due to quantity of variables
Mammals	Pulse generators + cascade/parallel resonant band pass filters	A vocal tract model is essentially a set of band-pass filters arranged to mimic the changing diameter of a long passage. Using less filters is a solution to change the level of audio detail.	In the code it would be necessary to remove some blocks. The effect on the sound would be a lower quality
Guns	Short sine sweep (higher pitch) + short sweep at 100Hz + set of series band pass filters + short noise burst + envelope + distortion	Controlling the shell chirp to vary the form of the impulse. Adding more aspects to the sound with more or less filtering. Distortion can be applied too. Changing control constants.	More computational resources are needed if more aspects are added.
Explosions	chirp impulse + shaped noise + low-passed noise (+delays)	Changing the offsets	Minor
Rocket Launcher	Narrow band pass filters (tube model) + comb-filtered noise (rocket)	Unload empty tube behaves as a full wavelength resonator that generates a lot of harmonics. Using filters could reduce amount of harmonics. Another approach is to not use handling sounds.	The sound would have no significant changes with the filters. Although removing handling sounds would slightly affect quality. is the code has enough detail to add or remove parts.
Transporter	Two-stage FM using triangle waves + delays + envelope + filter bank (10 parallel band pass filters)	A simpler model can be made by adding less sounds	Less computational resources will be required as the model becomes simpler
R2D2	FM (or phase modulation) synthesis	A simpler model can be made by adding less sounds	Less computational resources will be required as the model becomes simpler
Red Alert	Sawtooth waveform + second harmonic + envelope + fixed delay-based resonators & sharp filters	Adding more resonant factors & changing constants.	The computational resources needed changes as the complex of the sound changes

For fire, all features were generated by filtering and modulating white noise. Hence, adding more or less filters to the feature models or changing the modulation parameters was used to obtain dynamic Level of Audio Detail (LOAD). The first approach significantly affected the computational time needed to produce the sound while the second one did not. Another way to implement LOAD is to sum together two or more fire sound sources, each one with all three features implemented, and filter them in order to get different resonant frequencies from each one. This improves realism but increases computational time.

In the bubble model, the quantity of bubbles is determined by a random number within a given range. Our first approach to adding LOAD was to reduce or increase this range by a simple change of parameters. Our second approach was reducing the quantity of tones used for bubble pitches. The sound with the highest quality uses four different tones; in this case it has to process the same function four times. Using less tones slightly reduces the quality of the sound, but results in less functions running simultaneously.

Since the wind sound is essentially different kinds of noise modulated by filters that alter the direction and velocity, to apply LOAD we added more noise modulated in different ways to compose the ambience. The noise wasn't randomly added; filters were applied that would specifically create different elements, like leaves moving, objects rolling etc.

3 EVALUATION

For each model, we applied three levels of LOAD, as explained in Section 2.2. In Figure 1, spectrograms are shown for low, medium and high LOAD fire sounds. We can see that the more quality the sound has, the denser the spectrogram. The small number of filters for low LOAD generates a sound with frequencies in almost every part of the spectrum. As more filters are added, certain frequency ranges become emphasised, until the high LOAD where it is possible to note the cleaner sound.

Figure 2 depicts spectrograms of synthesised bubble sounds for low, medium and high LOAD, respectively. The spectrogram for the low level of audio detail has less frequency components than the one representing the sound with high level of detail. This occurs because the worst quality sound has fewer tones generators. The spectrogram for the midlevel LOAD should have the same amount of frequency components as low LOAD but it does not. This may have happened because the size of each bubble is randomly generated, thus influencing the frequency used. Also, bubbles are produced with random variations in timing, which may lead to the frequency content of bubbles adjacent in time appearing smeared, thus resulting in overlapping bubble tones in the spectrogram.

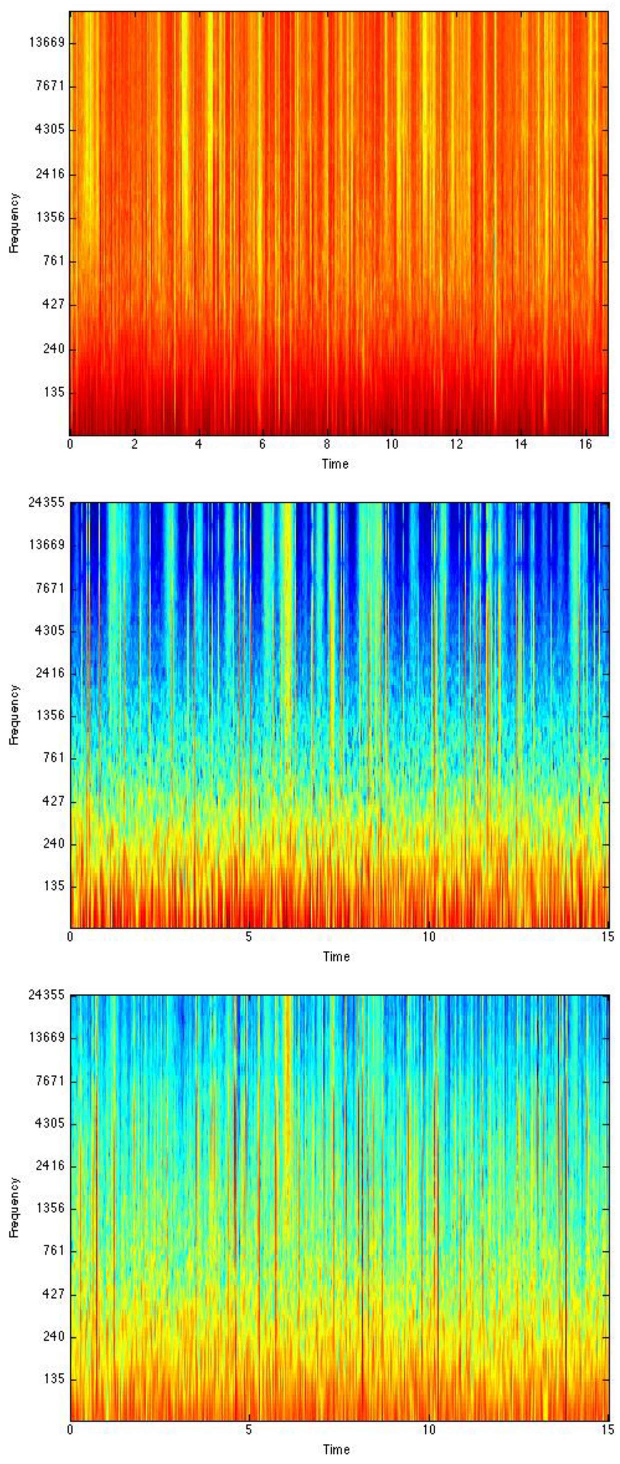

Figure 1: Spectograms of fire sounds for low (top), medium (middle) and high (bottom) LOAD.

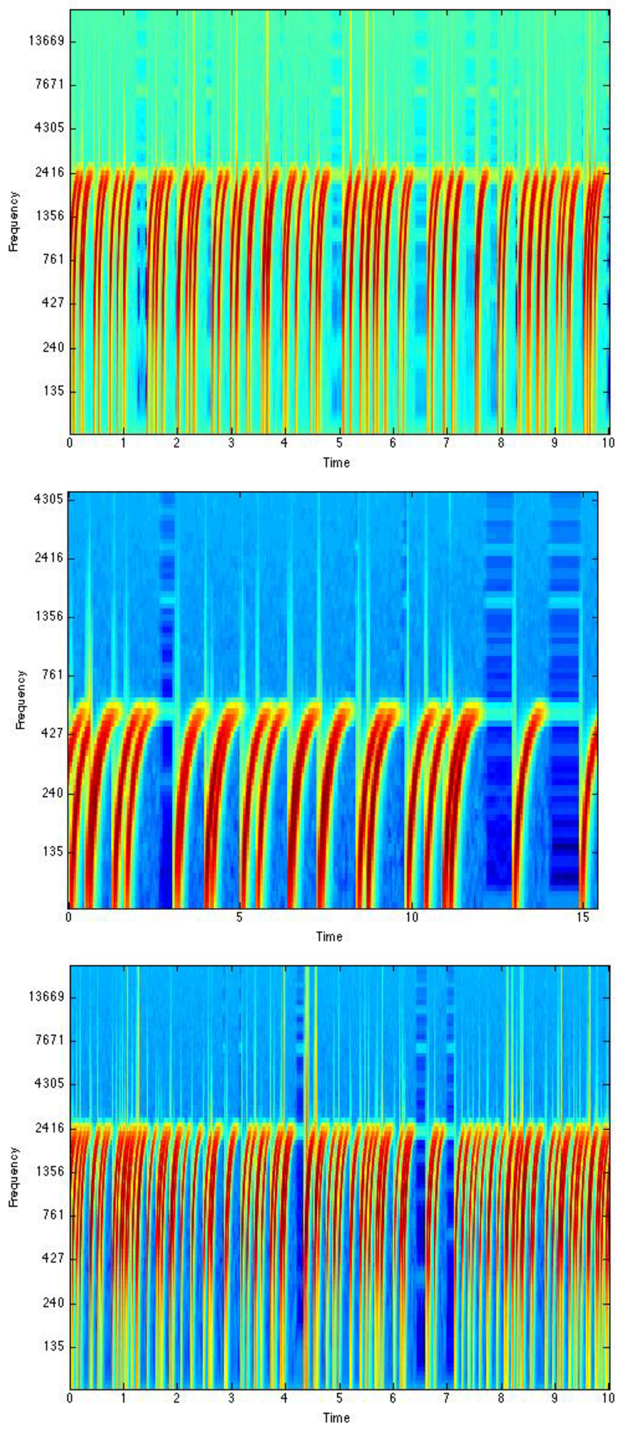

Figure 2: Spectograms of bubble sounds for low (top), medium (middle) and high (bottom) LOAD.

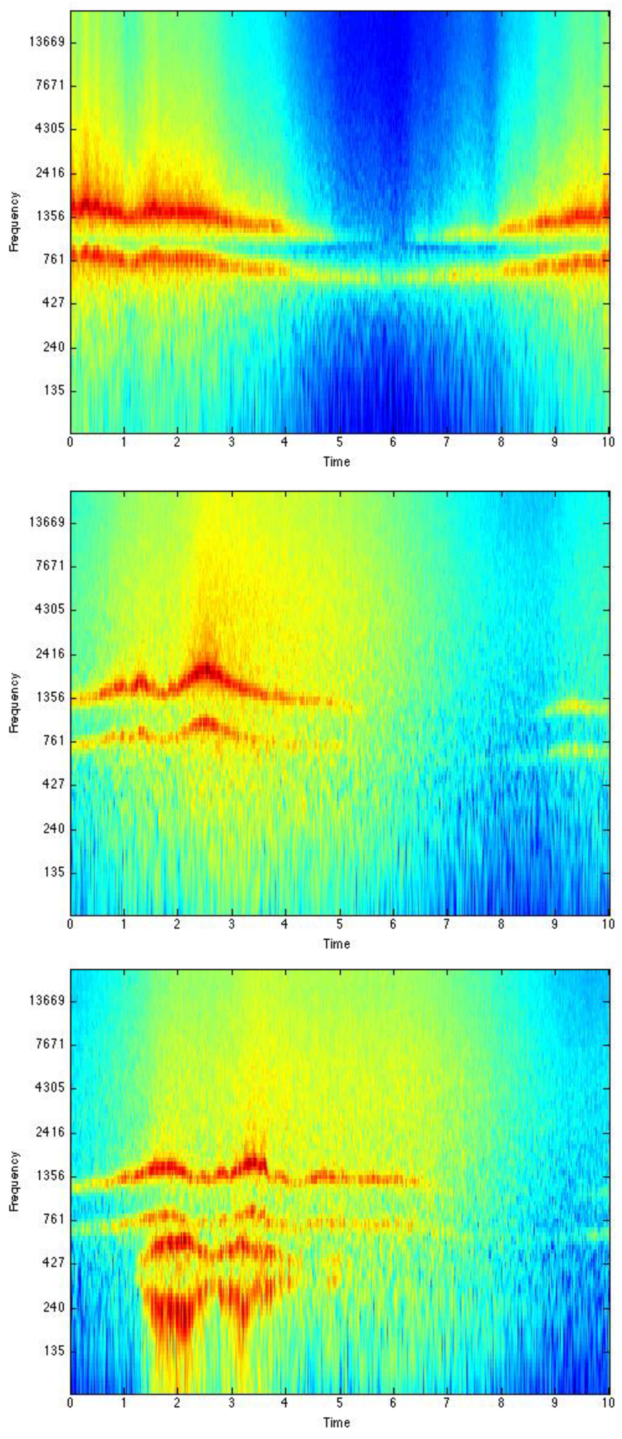

Figure 3: Spectogram of wind sounds for low (top), medium (middle) and high (bottom) LOAD.

In Figure 3, spectrograms are depicted for wind sounds with varying LOAD. We can see significant differences in the distribution of frequency content in each case. The differences exist because the noise distribution depends on LOAD. The higher quality sound has more noise since in that case, the wind sound is generated using a combination of noise sources and filters.

For objective measurement of computational performance, the number of operations the program performed and the computational time required to run the code were determined. These tests were performed on the fire model, implemented in Matlab Simulink, because these quantities are more easily assessed in Matlab than in PureData. Figure 4 shows similar behaviour for both computational time and number of operations. Both analyses are consistent with the expected results, and suggest that considerable savings can be achieved when low LOAD is used.

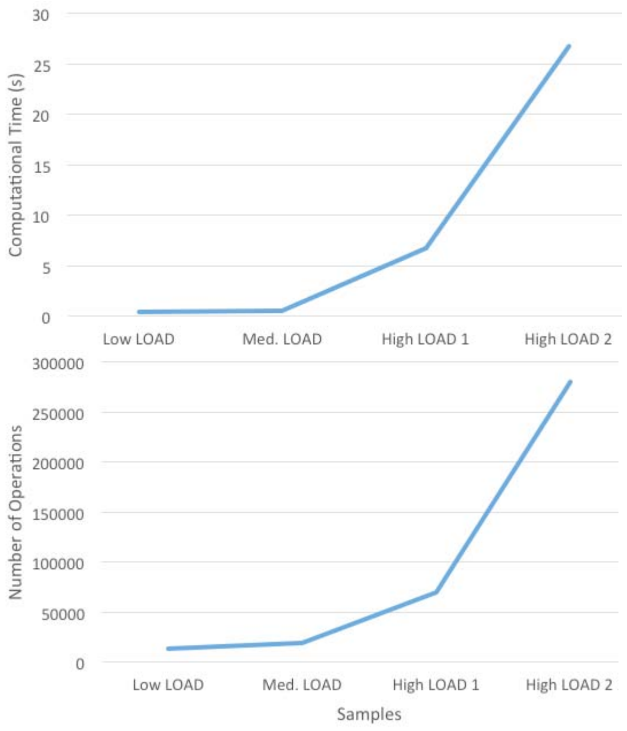

Figure 4: Computational time (top) and number of operations (bottom) of the fire model for different levels of LOAD.

4 LISTENING TEST

A listening test was performed in order to evaluate if the sounds synthesized by our modules were realistic and how the different levels of LOAD would be perceived, or if they would be perceived at all.

The test consisted of a multiple stimulus evaluation using the Audio Perceptual Evaluation toolbox for MATLAB [12]. Ten test subjects were chosen, and they each had to answer three multi-stimulus questions. Each question focused on one of the three sound types; fire, bubbles and wind. In each question, the test subject was asked to rate five audio samples in terms of how realistic they sounded. Three of the samples were created by our modules, which represented low, medium and high LOAD configurations. The other two samples were recordings of the actual sound. The interface is shown in Figure 5 and the results are plotted in Figure 6.

As expected, the recorded samples had the best mean ratings in all cases, because our synthesis techniques did not cover the whole range of complexity of each sound.

For fire sounds, low LOAD resulted in significantly lower perceived realism. Though high LOAD was not perceived to be more realistic than medium LOAD, both medium and high LOAD fire sounds were close in perceived realism to the first recorded sample.

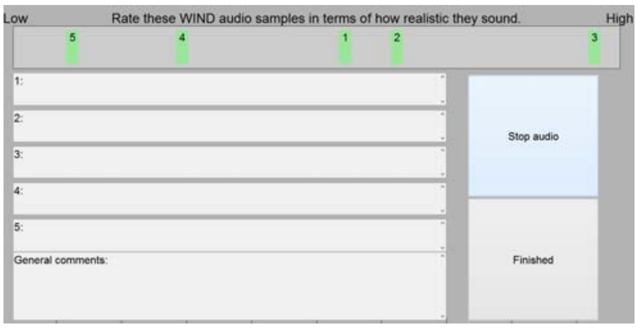
Low
Rate these WIND audio samples in terms of how realistic they sound.
High
5
4
1
2
3
1:
2:
3:
4:
5:
General comments:
Stop audio
Finished

Figure 5: User interface for the multi-stimulus perceptual evaluation listening test.

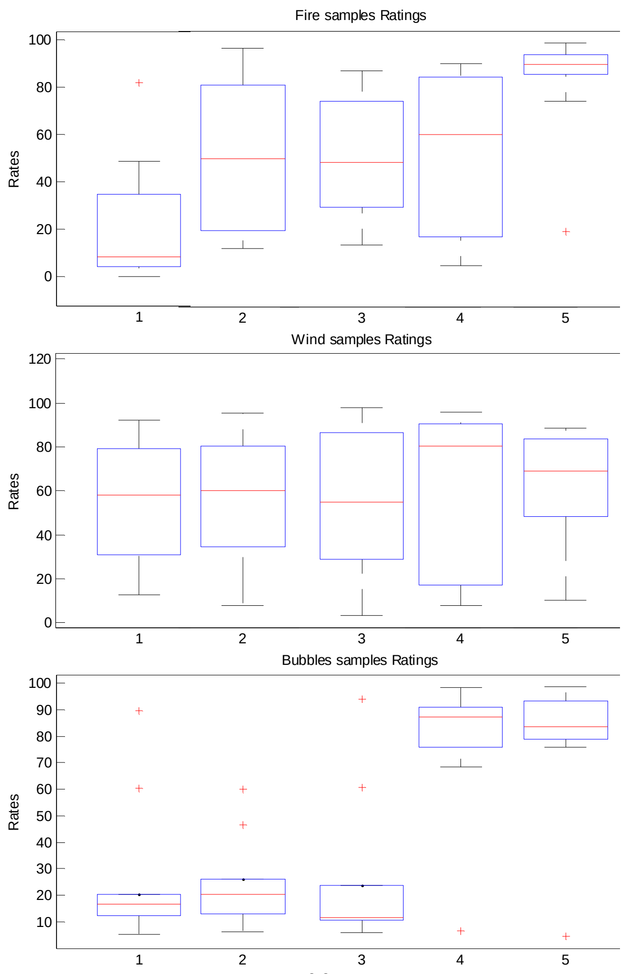

Figure 6: Subjective ratings of fire (top), wind (middle) and bubble (bottom) sounds. 1- low LOAD, 2- medium LOAD, 3– high LOAD, 4– recorded sample 1, 5– recorded sample 2. The box represents the 25-75% interquartile region, black lines represent the range excluding outliers, and crosses represent outliers.

Results were less conclusive for the other models, however. All results for wind sounds had large variance in ratings, resulting in an inability to distinguish performance. Only the first recorded wind sound had noticeable better perceived realism than other samples, but the large range of ratings for this sample implies that this result cannot be considered definitive. For the bubble sounds, the model does not generate sufficient complexity, and hence all synthesised samples were rated very low in comparison to the recorded samples. Hence, one cannot extract meaningful information regarding the perceived realism of the synthesised bubble sounds with varying LOAD.

5 CONCLUSIONS

In this work we sought to understand the mechanisms for creating a sound with variable Level of Audio Detail. We were able to create and manipulate three sound models, apply varying LOAD on each of them, and synthesise samples that in some cases have a level of perception close to reality.

However, the results obtained in the listening test were not satisfying. The different Levels of Audio Detail were not ranked as expected. We suspect that this is due to the oversimplicity of the synthesis models and the challenging nature of performing an insightful listening test.

For further work, LOAD should be implemented such that LOAD has a continuum of values, and high LOAD approaches the full level of realism of a recording of the actual sound. Thus subjective evaluation could compare the full spectrum from crude to highly realistic synthesis. This would also allow thorough investigation of how the dynamic aspects should be implemented without perceptual artifacts being introduced as LOAD changes.

One further challenge with the approach presented in this paper was that each sound is synthesised using a unique approach. Thus the implementation of LOAD differs for each sound. An alternative would be to explore how LOAD can be applied in a consistent manner to a very general synthesis technique. For example, if sounds are synthesised using an example recording as a template, then a sinusoids plus noise model might be applied, and LOAD can be directly linked to the number of sinusoids in the synthesis, regardless of the sound source.

In this paper, the subjective evaluation was only in the form of comparison between the proposed models and recorded samples. Better benchmarking, and more accurate assessment of the state of the art, would be achieved by including other synthesis techniques from the literature for comparison. Only then would it become clear how different approaches to sound synthesis, with or without LOAD, compare to each other.

REFERENCES

[1] Farnell, A., Designing Sound. MIT Press, 2010.

[2] Funkhouser, T., and Sequin, C. Adaptive Display Algorithms for Interactive Frame Rates during Visualization of Complex Virtual Environments. Computer Graphics (SIGGRAPH), Los Angeles, CA, 247–254, 1993.

[3] Clark, J. H., Hierarchical Geometric Models for Visible Surface Algorithms. Communications of the ACM, Oct. 1976 19 (10). P 547-554, 1976.

[4] Tsingos, N., et al Perceptual Audio Rendering of Complex Virtual Environments. ACM Transactions on Graphics (TOG), proceedings of ACM SIGGRAPH, New York, NY, 249-258, 2004.

[5] Fouad, H., et al. Perceptually Based Scheduling Algorithms for Real-time Synthesis of Complex Sonic Environments. Proceedings of the 1997 International Conference on Auditory Display (ICAD'97), Xerox Palo Alto Research Center, Palo Alto, USA, 1997.

[6] Schwarz, D., et al. Sound Level of Detail in Interactive Audiographic 3D Scenes. Proceedings of the International Computer Music Conference (ICMC), Huddersfield, UK, 2011.

[7] Kleczkowski, P., Selective Mixing of Sounds. 119th AES Convention, New York, Oct. 2005.

[8] Tsilfidis, A., et al, Hierarchical Perceptual Mixing. 126th AES Convention, Munich, Germany, May 2009.

[9] S. Hendry and J. D. Reiss, "Physical Modeling and Synthesis of Motor Noise for Replication of a Sound Effects Library," 129th AES Convention, San Francisco, 2010

[10] C. Heinrichs and A. McPherson, "Mapping and interaction strategies for performing environmental sound," IEEE VR Workshop on Sonic Interactions for Virtual Environments, Minneapolis, USA, 2014

[11] C. Heinrichs, et al., "Human performance of computational sound models for immersive environments," The New Soundtrack Journal, 2014.

[12] B. De Man, J. D. Reiss, "APE: Audio Peceptual Evaluation Toolbox for MATLAB," 136th Audio Engineering Society Convention, Berlin, April, 2014.