Assessment: Computer Music

There are two parts to the assessment.

1. Task 1 (Sound Synthesizer with GUI) Programming project with 1000 word write-up. Submitted Thursday Week 7 SPRING term (50%)
2. Task 2 (Algorithmic Composition) Programming project with 1000 word write-up. Submitted Assessment Block 2 (50%)

All submissions will be electronic, via the Study Direct site for the module.

Programming Tasks:

You will have to submit everything required to compile and run the programs, with the writeup. It is strongly suggested you zip everything up into one submission file. The time of submission will be strictly adhered to as recorded on Study Direct.

When you submit your SuperCollider code, don't forget to include any sound samples and to indicate any third party dependencies! (do not use obscure third party libraries unless you have cleared with me that I can get hold of them). You must either use Mac OS X, or your work must be compatible in principle with cross-platform compilation. Direct assessment will be undertaken of your code; it must run out of the box.

Note that there is an upload limit on Study Direct. If you need very large audio files as part of the project, you should place them online within another zip (e.g. just one download link) and provide a link to download this in a README document with your submission. The only external data that will be accepted in this fashion is auxiliary audio or video data; no programming code should be placed externally.

Each task should be accompanied by a 1000 word writeup in a PDF document (you can export as PDF from Word, for example) detailing your solutions to the exercise. You will be expected to contextualise your work with respect to the computer music literature as necessary; do not forget to include evidence of your wider reading and the relationship of your work to existing computer music projects.

Individual Task Details (to be further discussed during the module)

1. **GUI based/interactive synth**. Create a sound synthesizer with a GUI control panel which demonstrates one or more of subtractive, additive, modulation, sample-based or granular synthesis.

Example: Model a basic analogue synthesizer by providing a set of waveforms which can be chosen or mixed in as sources. These are passed through a filter, with envelope controls for the filter cutoff, and for amplitude over time. Notes are triggered by MIDI input, or by a button.

Marks Breakdown:

Proportion of marks out of 100	Being tested	How to get high marks
10	Coding neat and commented	10 = industry level commenting and formatting
10	Code runs out of the box and is difficult to break, despite a large codebase	10 = no problems, no crashes, easy to get going, but a large codebase. Note that a three line program will not score well here since there is no challenge in getting it to run!
20	Write-up describes the solution accurately, and is well contextualized, with appropriate references	Show that you understand the sound synthesis theory, and provide evidence of wide ranging further reading which places this system in the context of computer music research and projects.
20	Sound synthesis algorithms correctly implemented and go beyond basic examples	Your coding of the sound synthesis shows you fully understand how to implement the algorithms, and that you have extended beyond the basic components to a living synthesizer.
20	Fully functional and well laid out GUI	The interface is neatly and sensibly laid out, with a command of user interface design and facilities appropriate to the synthesizer's capabilities.
20	Overall user experience: how inspiring is the synthesizer?	The marker will rate the quality and range of output options allowed by the system, and your overall creative response to the task.

2. **Algorithmic composition**. Make an algorithmic composition using SuperCollider. You may either do your own scheduling, perhaps via {}.fork, or make use of the Patterns library. All sounds must also be created in SuperCollider, and one or more effects units should be used in addition.

Example: Create an algorithmic drum machine for producing automatic techno. Synthesized kick, snare and hihat sounds are created, with sample playback for the toms. An accompanying bassline is synthesized with lead line and chord stabs. A reverberation unit is used, taking differential amounts of each source part.

Marks Breakdown:

Proportion of marks out of 100	Being tested	How to get high marks
10	Coding neat and commented	10 = industry level commenting and formatting
10	Code runs out of the box and is difficult to break, despite a large codebase	10 = no problems, no crashes, easy to get going, but a large codebase. Note that a three line program will not score well here since there is no challenge in getting it to run!
20	Write-up describes the solution accurately, and is well contextualized, with appropriate references	Show that you understand the possibilities of algorithmic composition and its theory, and provide evidence of wide ranging further reading which places this system in the context of computer music research and projects.
20	Algorithmic composition correctly implemented and goes beyond basic examples	Your coding of the algorithmic composition shows you fully understand how to implement the algorithms, and that you have pushed the envelope in your response.
20	Variation of musical output	There is significant engagement with a particular style, and exciting musical outputs result which substantially vary between program runs
20	Sound synthesis and effects	The choice of sound synthesis techniques and effects will be marked independently of the algorithmic structure itself, rating the quality of output sound and the means used to obtain it.

Assessment criteria for exercises: UG

85-100% Original and exciting creative solutions showing significant effort in their construction, which deeply acknowledge and analyse the network of prior art and technology upon which they rest.
70-85% Individual solutions with some original and well-built code are presented which are carefully contextualized with respect to other work, demonstrating competent integration of computer music principles.
60-70% Good ideas are pursued, with a working practical instantiation and broad appreciation of context.
50-60% The solutions show some evidence of an individual idea, without a fully convincing realisation, and partial understanding of the computer music background.
40-50% Minimal functioning solutions are provided which are poorly contextualized, with at least some redeeming feature of note.
30-40% Basic constructions are provided which show no real attempt to engage with the creative task, little appreciation of context, and may not fully function.
15-30% Solutions are presented but are incomplete and do not function.
0-15% Little or nothing has been accomplished.