Francesco Redente

Production Analysis for Advanced Audio Project

Option - B

Date of submission: 28/08/2015

SAE INSTITUTE

Declaration

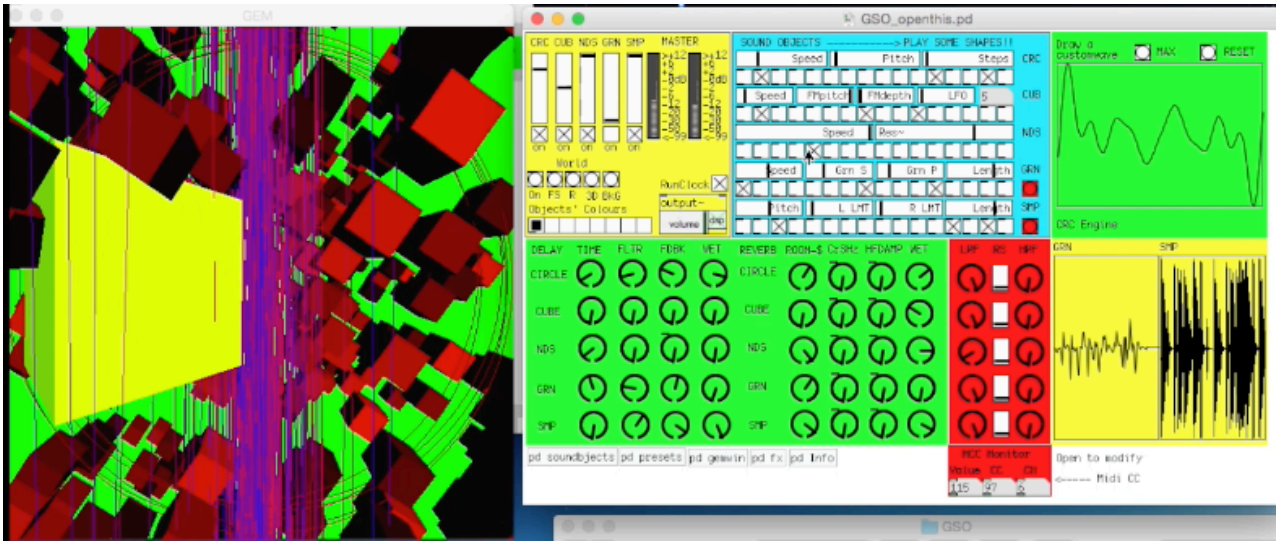
I hereby declare that I wrote this written assignment / essay / dissertation on my own and without the use of any other than the cited sources and tools and all explanations that I copied directly or in their sense are marked as such, as well as that the dissertation has not yet been handed in neither in this nor in equal form at any other official commission.

Table of Content

Introduction

The aim of this paper will be to present the reader with a descriptive analysis of the making of Gem Sound Objects (GSO). GSO is an audio-visual interactive musical instrument fully designed in Pure Data using the GEM library (Graphics Environment for Multimedia), sound synthesis and user interaction via midi controls changes (CC). It is designed to produce abstract audio visual compositions, inspired by Pierre Schaeffer 'sonorous objects' theories and experiments that led to *musique concrète*, which inspired artists, sound designers, composers and audio developers to think beyond the usual method of sound creation, tone manipulation and listening (Cox and Warner, 2004, p. 76).

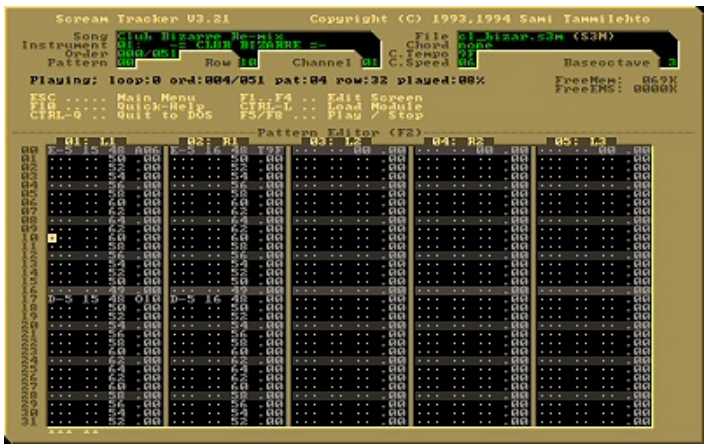
I will firstly outline the ideas which inspired it, sharing early drafts and the detailed research which motivated me to create the GSO software instrument (fig.1). I will then explain in detail how I managed the project, from the intricate technical requirements to how I reigned in my creative strategy to ensure that the objectives of the project were fulfilled within the time given. The Programming Implementation section will guide the reader through the code used to program the instrument followed by a guide for each section of the patch. Finally, my conclusion will assess the overall learning outcome of this practical assessment, evaluating both the final result to understand its strengths, limitations and potential, as well as my approach to the project in terms of strategy and time-management.



(fig.1, G.S.O, pic by Redente, 2015)

Concepts and ideas for GSO.

I have always been interested in computer programming, growing up in Italy I used to visit my neighbour Luca, a game programming student who used to make his own 2D games using the MS Dos operating system. At the time my computer skills were very poor, but my passion for music technology and sound made me question how I could use computers to make music. I was fascinated by Luca's programming skills and how easily he could write lines of text into a computer and somehow convert them into sounds. After a few months of visiting Luca he showed me Scream Trucker (Tamimlehto,1990, fig.2), a four track text based sequencer which sounded like nothing I heard before and it looked fairly easy to get sounds from. Without even knowing what synthesis or music theory were, I could write lines of text and the computer would convert them into synthesised sounds. Such an enriching experience somehow changed the way I thought of computers to this day, not only as a platform to write documents or play games, but rather as a machine capable of manipulating aural experiences. One of the reasons I started the audio production course at SAE was to finally learn computer programming. I could not wait to start the second year to research topics such as audio programming, user interaction and audio-visual experiences. I can say that I took the decision of making a Pure Data assessment two years ago, but I officially started working on it once the programming classes started, inspired by my early Scream Trucker memories which lead to 15 years of using DAW (Digital Audio Workstation) to produce my own music. The core idea to create a Pure Data patch started to materialise following a research paper I wrote during 501 (*An analysis on the process of creating live Electronic Music. Bringing composition and live performance together in modern Electronic Music*), in which I interviewed co-founder of Ableton Live Robert Henke. Henke exposed me to the unique compositional method he used to create Lumière, an audio-visual



(fig.2, Scream Trucker, pic by pouet.net, 2015)

experience fully designed by himself using Max/MSP, a laptop and some MIDI controllers. Live coding techniques using programming environments such as Extempore (Sorensen, 2013) and Tidal (Mclean, 2009) also grabbed my attention, but their level of user interaction, visual experience (considering my low level of programming skills) made me realise that a software like Pure Data would be a good start for me. During our programming classes we did lots of practical work, which really helped me to question the process of designing patches; what do I want to make? What do I need it for? Why people would want to use it? What tool is currently missing in my live set up? After few weeks of experimenting and making patches, which served as a learning process, I came across a picture of Schaeffer (fig.2) while reading *In Search of Concrete Musique*, a book advised to me by my tutor Jon. Reading about Schaeffer's concept of the *objet sonores* and listening methods proved to be a key inspiration to design GSO, an audio-visual interactive musical instrument designed to produce abstract audio visual compositions.

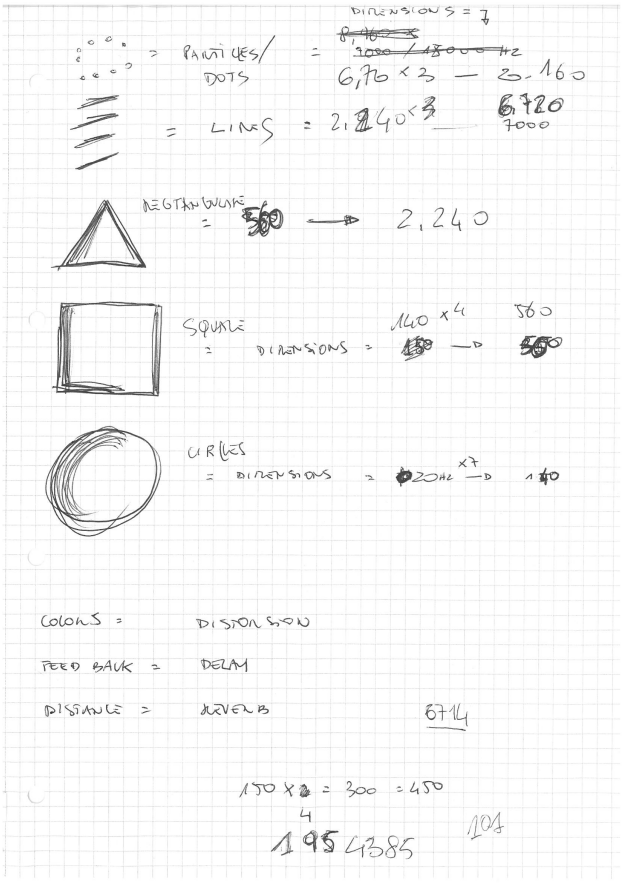
> *"In listening to sonorous objects [objets sonores] whose instrumentals causes are hidden, we are led to forget the latter and to take an interest in the objects for themselves."* (Schaeffer in Cox and Warner, 2004, p. 78).

(fig.3, Pierre Schaeffer, *Expérience de musique concrète*, pic by Institut National Audiovisuel, 2015)

Schaffer's theories inspired me to create visual objects (using Pure Date GEM library), with their geometries mapped to synthesis engines in Pure Data such as FM, Wavetable, Grain, Sampler and a random noise generator. Using a MIDI controller (sending control changes to PD) the user interacts with the GEM objects in order to generate sounds and modify their sonic forms, consequently focusing on live sound experimentation and where this audio-visual experience may lead to, both sonically and visually.

Methodology

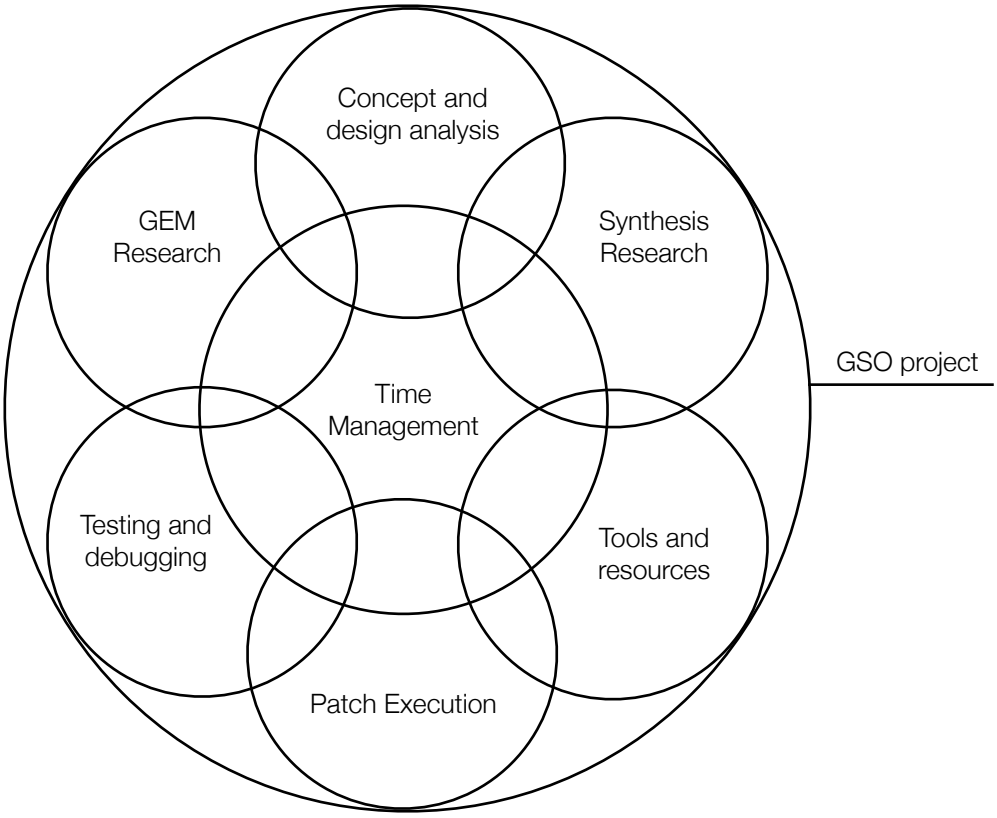
Once I decided what my patch was going to be and used for, I started planning a strategy to approach the different stages of the project. Attending the Pure Data classes helped me to familiarise myself with the software and the rationale needed in order to guide the programming in the right direction. Considering that my background experience in sound design and composition has always been spontaneous and practical, the Pure Data classes helped me to develop a more theoretical approach, or concept analysis, which facilitated the final practical exercise of building patches and testing the end result. The initial analysis helped me to envision the finished patch, giving me time to sketch ideas while also motivating me to start working on the patch's design and achieve tangible results.



(fig.4, GSO first sketch, pic by Redente 2015)

Few of the questions which I kept on asking myself were: What am I missing from my current live set up? And what I am interested in? The GSO initial sketch helped me to understand how to relate images to sounds, most importantly I wanted to be able to modify sound synthesis parameters by manipulating images. I did not want to emulate any existing instrument, instead, I wanted to take a creative approach to sound synthesis with the help of images, each shape representing a sound engine with its own unique timbre and frequency range. I also planned to add a few elements of surprise to enhance the user experience; somehow connecting audio and video (e.g. video feedback to audio delay) parameters to create experimental effects while performing with the GSO.

(fig.5, pic by Redente 2015)



I divided the overall project into smaller steps, all of which formed a final system that helped me stay focused towards the final goal and also have a clear vision of what was needed to successfully complete the assessment within the given time (fig.5).

During the concept and analysis stages I consulted industry professionals, my tutors and some of my classmates (Cristian, Benedict and Luigi). Their feedback helped me to plan the initial research needed to be able to execute the programming required, and also to understand possible challenges and technical limitations I could have encountered along the way. In term of tools and resources needed to successfully accomplish the project, I decided to use my MacBook laptop to do all the programming, especially because I wanted the overall patch structure to fit perfectly on a 13” laptop, making it easy to load and control in both live and studio scenarios. To maximise user interaction and to learn how to implement it within the patch, I purchased a second hand MIDI controller (Beringeer BRC 2000) to be used as a live controller for the GSO.

Research

As the GSO project diagram shows (fig.5), I divided the research into two stages: the GEM research to cover the visual side of the project and the synthesis one to cover the sonic side of it. The GEM library comes ready to use as part of Pd-extended, I had no knowledge on how to use it, so I gave myself a few weeks to research and practice how to generate 3D images. My initial sketch (fig.4) really helped me to focus on what was important for the project: generate basic geometric images and somehow map some GEM parameters to a sound engine. I found many interesting papers on GEM and other projects which used a similar concept to mine. I started reading the Floss Manuals (online) then I moved on to more specific papers such as IOhannes M Zmölnig GEM (2003) and David Shimamoto's *Presentation material on GEM* (2002). I also researched John Whitney's works and his concept of "digital harmony" (Whitney, 1980).

> *"...all of the numbers...will arrive at some sort of harmonic relation among themselves...and as a result produce a more simple pattern...I was attracted by it primarily because this kind of harmonic phenomenon...is at the root [and] organization of music...the content of music is really motion, It [is] a matter of generating and resolving tensions via a process that is very much dynamic, a continuos matter of motion patterns, a kind of architecture in space and time..."* (Whitney in Wiggins, 2010)

Whitney's theories and numerous works, which he made using self-built computer systems, expanded my initial ideas and motivated me to make my project a reality. The research became quite exciting, I was able to get quick results and slowly elaborate how to best implement the initial ideas from my GSO sketch (fig.4) to Pure Data. Once I felt confident with the visual side of the project, I started researching sound synthesis and how the two could creatively come together. The Pure Data classes served me well for my research and practical purposes, and to further my research on sound synthesis and project management I read Tony Hillerson's book *Programming Sound with Pure Data* and sections of Andy Farnell's *Designing Sound,* online texts such as Johannes Kreidler's *Programming Electronic Music in Pd*, Miller Puckette's *Pure Data Documentations* and the Floss Manuals.

The reading helped me to familiarise myself with the specific terminologies involved, and also improved my mathematical understanding of programming sound in Pure Data, learning which [objects] were essential and how to creatively patch them to create a sonic system. I used those resources also during the execution of the patch, to learn how to implement my ideas and also to understand where and how to experiment with the programming. All the research I have done throughout this module, together with our Pure Data classes, expanded my understanding of mathematical music and programming. It also proved to me that both 3D graphics and sounds can be use to produce new types of instruments as well as sound design techniques capable of creating interactive audio-visual compositions.

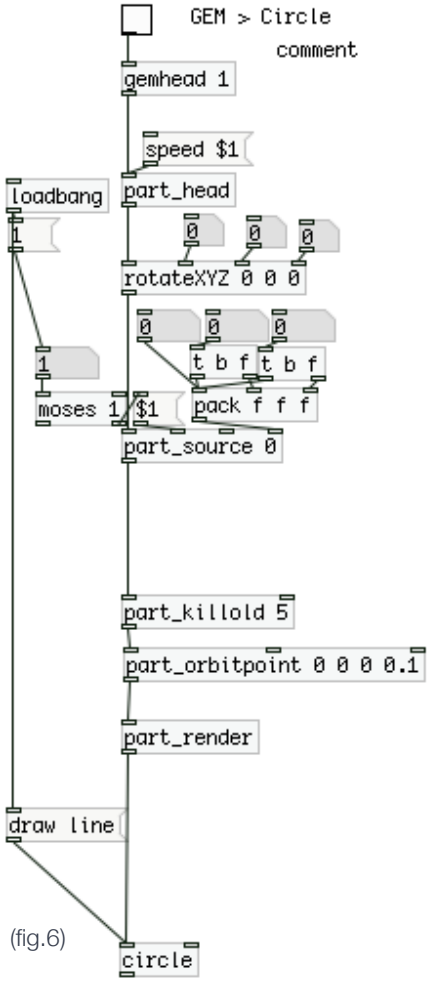
Programming Implementation

I planned to implement the programming in the most simple way possible, giving priorities to sound experimentation and user interaction. The main inspiration for such goals, and something which also helped me to focus throughout the creation of the patch, came from an interview with Morton Subotnick featured in the Buchla documentary preview;

> *"...Music was this thing...[before the invention of the Buchla]...and still is to this day, when you say music you know what music is... I am looking for the day you do not know what music is..."* (Subotnick in Clarity Films, 2014)

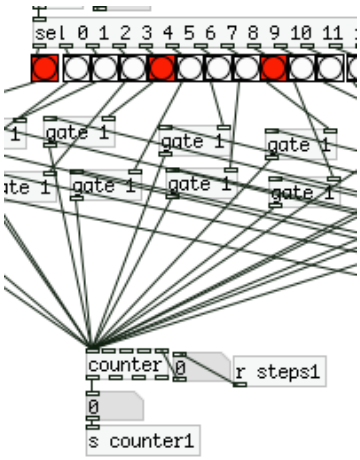
Those words made me feel good in terms of experimenting with the programming in order to create an instrument which was not intended to sound or look like any other. I started by first programming the 3D images using the GEM library, and once the first one was fully functional I then created the others. While programming the images, I also programmed a sequencer and the basic oscillators and connected all of them together. This gave me an instant feedback on the progress I was making, both visually and sonically. The following examples are taken from the GSO patches in Pure Data.

I first programmed a patch which produced a 2D image of a *Circle* (fig.6). It was simple to generate and gave me a chance to identify key parameters, such as rotation points, source points and size, which I then controlled via a sequencer (fig.7). The sequencer's counter was sent to both the circle geometry size and its sound engine, initially a simple oscillator (fig.8). Once this basic programming was tested and fully functional, I used the same method to program the remaining GEM patches, and made other basic oscillators for sonic tests.

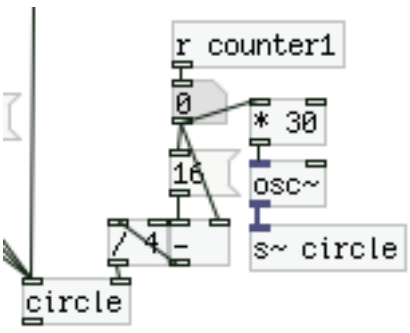


(fig.6)

The final GEM visual system consisted of a *Circle*, a *Cube* and three particles systems: *Lines*, *Cubes* and a *Sphere*. Sonically speaking, I already had a clear idea of creating five sound engines, as explained in the Concepts and Ideas for the GSO section. By programming the GEM systems first, I had the time to identify the key parameters which I wanted to use as a manipulation source for the sound synthesis. I then started to program the sound engines.

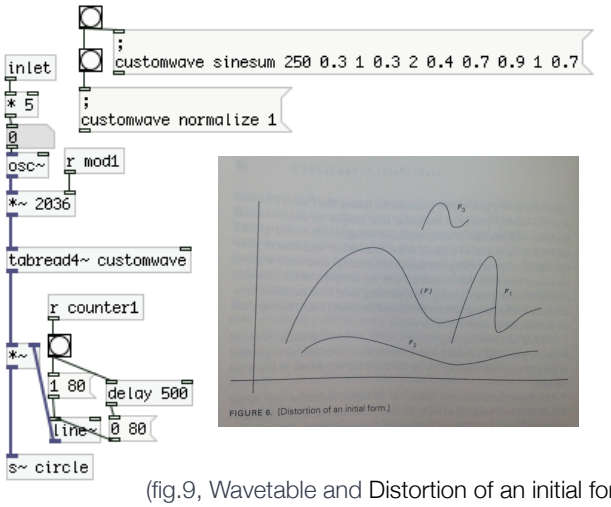


(fig.7,)



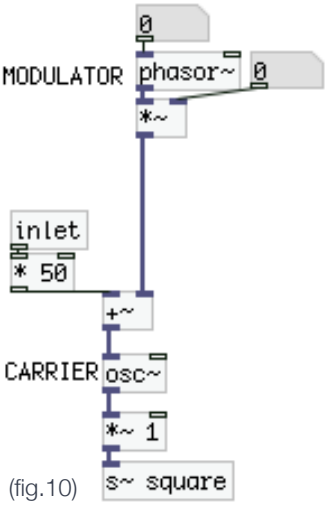
(fig.8,)

I replaced the basic oscillator previously programmed with a custom Wavetable engine (fig.9), FM synthesis (fig.10) , random noise generator (fig.11), Grain synthesis (fig. 12) and a Sampler (fig.13). To generate the *Circle's* sound engine I have used a [message] (sent to an array called customwave) containing values which represent the number of harmonics in a waveform. I chose to use a Wavetable method (fig.9) because I wanted to represent Schaeffer's example of 'Distortion of an initial form' (Schaeffer, 2012, p.40), which I discovered in his book *In Search of a Concrete Music*.

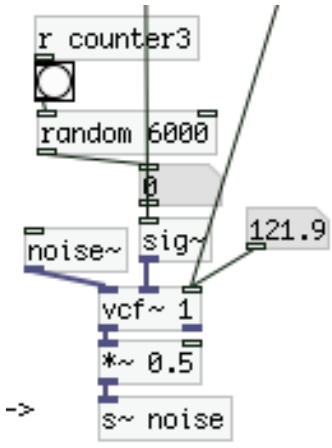


(fig.9, Wavetable and Distortion of an initial form)

Using receive and send [objects] I mapped some of their parameters to the final GUI (Sound Object Section in Functionalities), which also controlled key features of the GEM systems. I used multiplications and other math [objects] to distribute the frequency of each sound object across the audible frequency range.

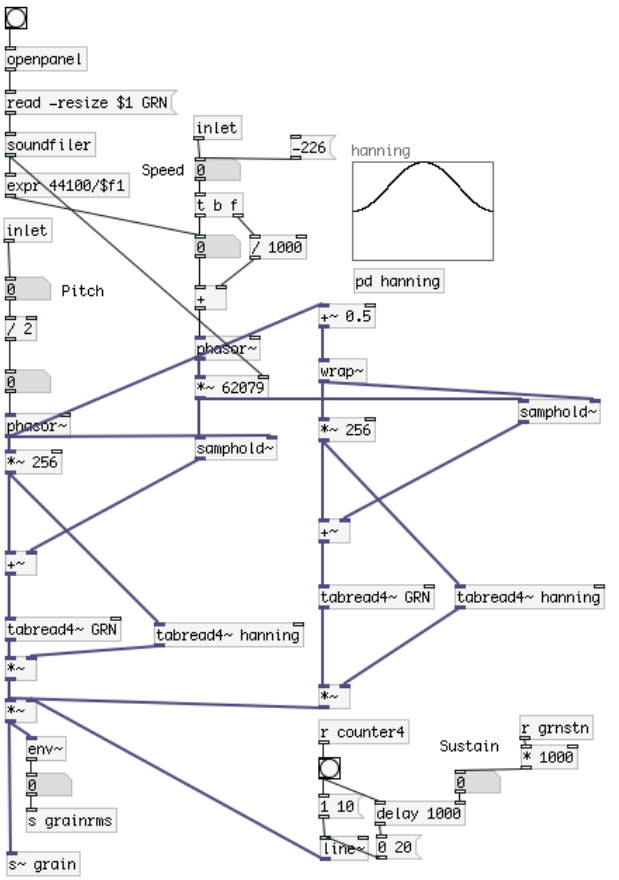


(fig.10)



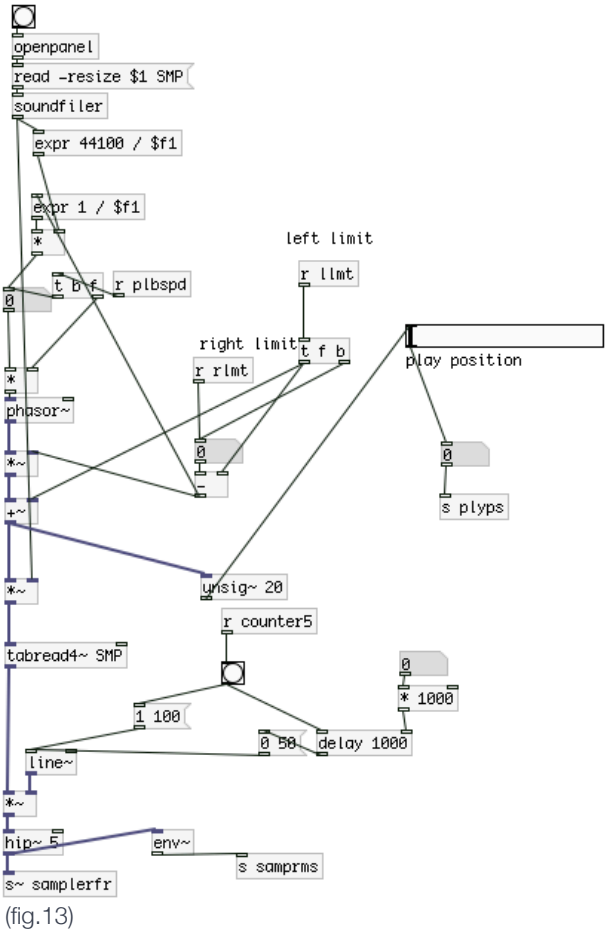
(fig.11)

The *Sphere's* sound is based on a grain synthesis engine. I learned to program this patch by reading Johannes Kreidler's book *Programming Electronic Music in Pd*. This method was also covered by our lecturer during the Pure Data classes. I then added a [line~] object (triggered by the *Sphere's* sequencer), and via the properties message I mapped the gains' pitch and speed to the GUI. I also added an [env~ object] to obtain RMS values from the audio signal output, which I used to modify the size of the *Sphere* (fig.12).



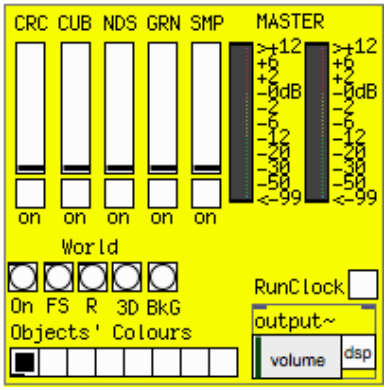
(fig.12)

The *Cube* sound consisted of a basic sampler which I also learned to program by reading Kreidler's book. I used a similar method used for the other sound object to modify its parameters. This patch worked successfully but moving the left and right limit in real time created artefacts in the audio signal (fig.13). I researched how to fix this problem but I could not find a working solutions which suited the patch. My tutor advised me not to worry about this issue.

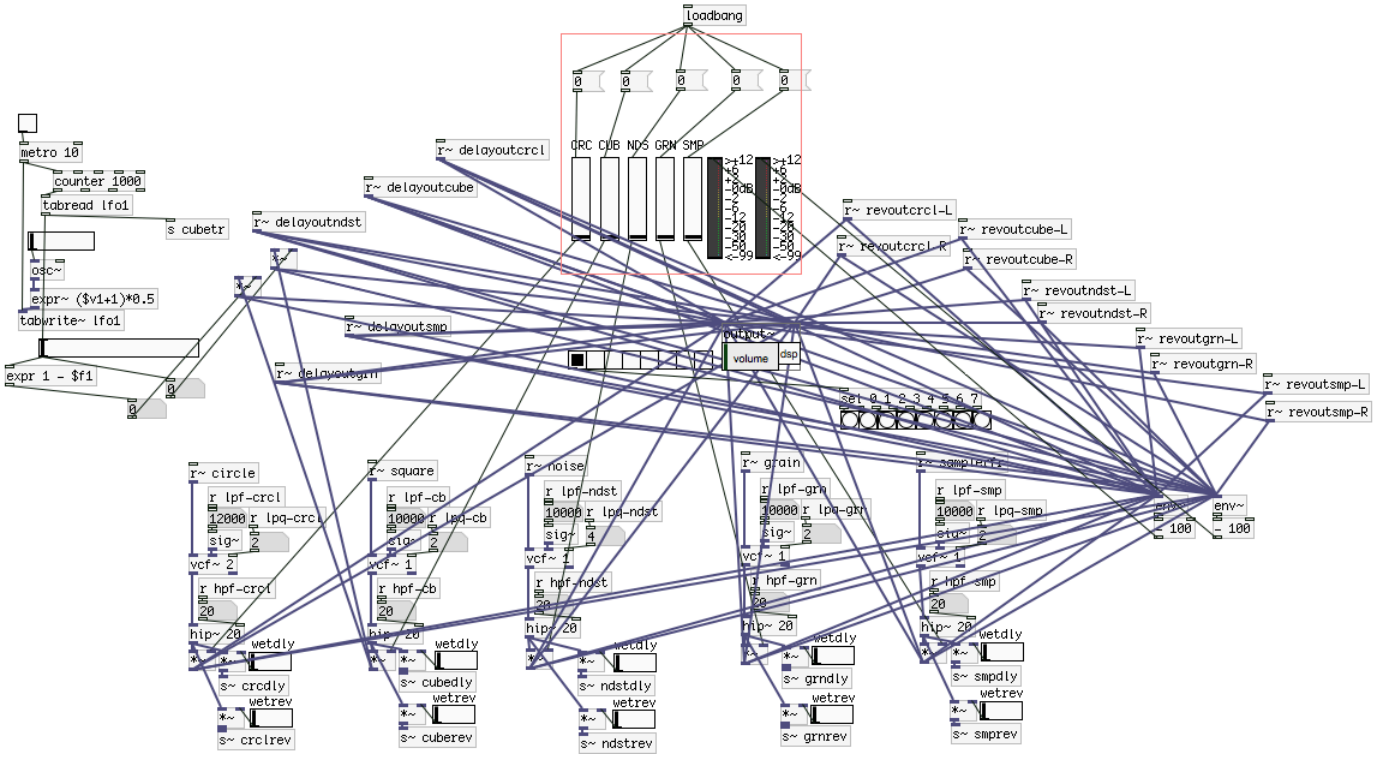


(fig.13)

To manage the overall signals within one patch, I programmed a master section which consisted of a sub patch receiving signals from each of the sound objects (fig.14). In the master sub patch I also programmed the routings of the effects such as reverb, delay, filters and an LFO (controlling the *Cube's* pan). I then designed a GUI (fig.15) to control the level of each sound object, added a toggle switch to start all the sequencers and some features from the GEM window rendering subpatch [pd gemwin].

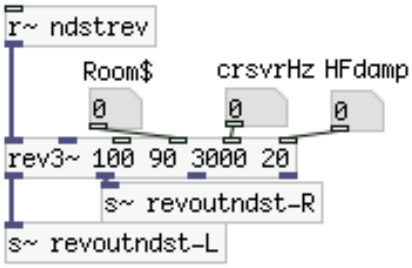


(fig.15)



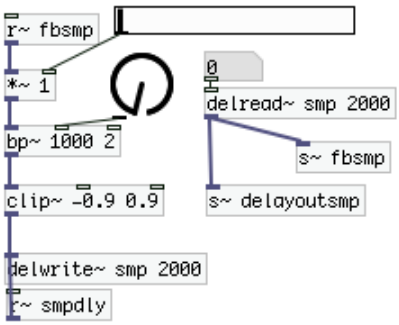
(fig.14)

For the reverb effect I used an [object] called [rev3~], which comes as part of the Pd extended library. This object allowed me to control the room size, crossover frequency and high frequency damping via a number box (fig.16).



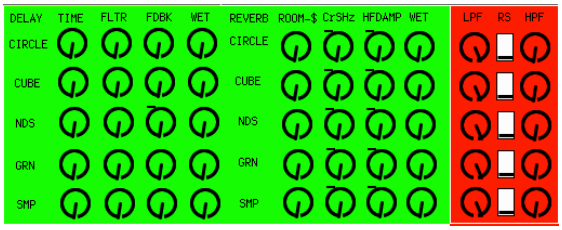
(fig.16)

For the delay effect I programmed a tape delay for each sound object. Our tutor covered this topic during one of our Pure Data classes. I programmed one delay during the class and saved it as an abstraction and copied the same subpatch and labelled each [delwrite~] and [delread~] with the corresponding sound object's name. Finally I added GUI controls for feedback, filter frequency and delay time (fig.17).

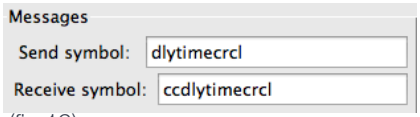


(fig.17)

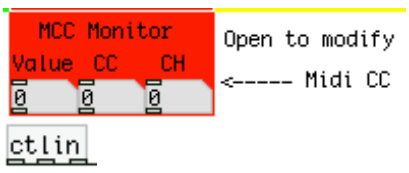
To keep the same style for all the GUI controls, I also designed a section to control each of the sound objects effects (fig.18). Using the property's messages I easily managed to send all the GUI pots' values to each effect parameter, without having to use physical connections (fig.19). I also made a MIDI Control Chance Monitor subpatch (fig.20), consisting of multiple [ctlin objects] receiving MIDI control change values from the Beringeer BRC 2000 controller. This allowed me to put the final touch to the final GSO GUI, making it fully interactive and controllable in both live and studio scenarios.



(fig.18)



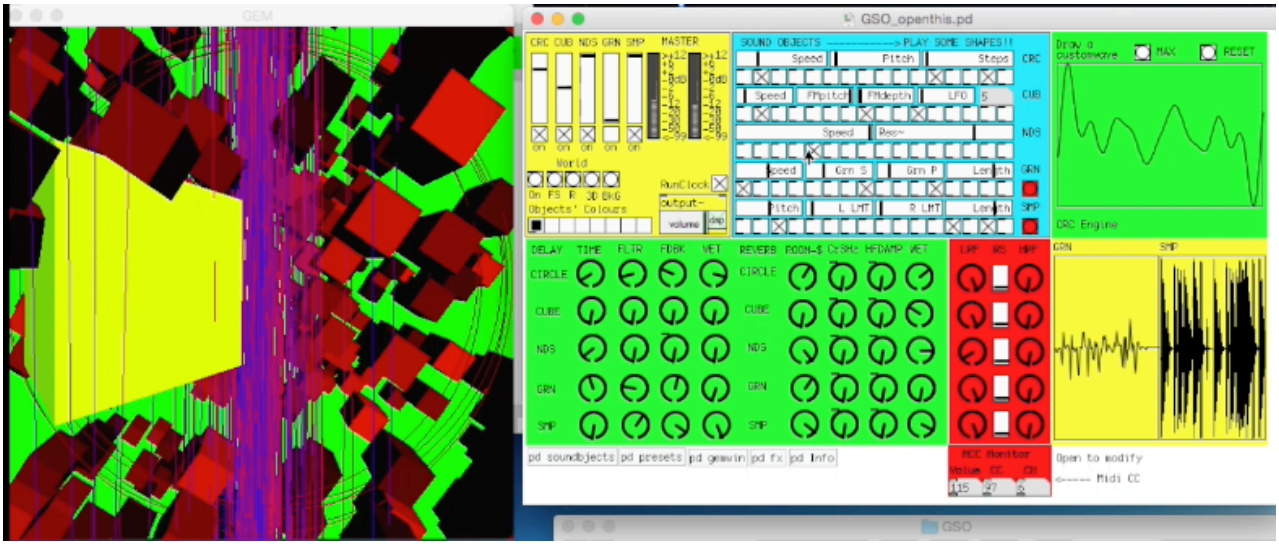
(fig.19)



(fig.20)

GSO functionalities

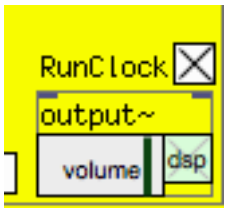
Welcome to GEM SOUND OBJECTS (GSO fig.21), a performing instrument capable of creating abstract audio-visual compositions. Some of the sound objects' parameters have been mapped to modify the geometry of the objects (world). This method will hopefully encourage user interaction to perform audio-visual live compositions.



(fig.21, G.S.O, pic by Redente, 2015)

MASTER SECTION

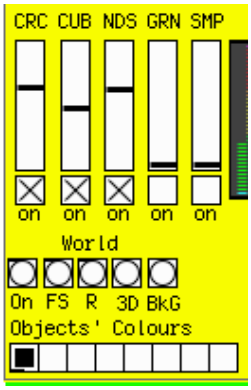
To start, activate RunClock, this will trigger all the five sequencers within the patch. To activate the DSP simply increase the output~ slider (Volume) to a desired level (fig.22).



(fig.22)

CRC, CUB, NDS, GRN and SMP control the volume of each sound object, while ON will activate their visual representation in the GEM window (WORLD section, fig.23).

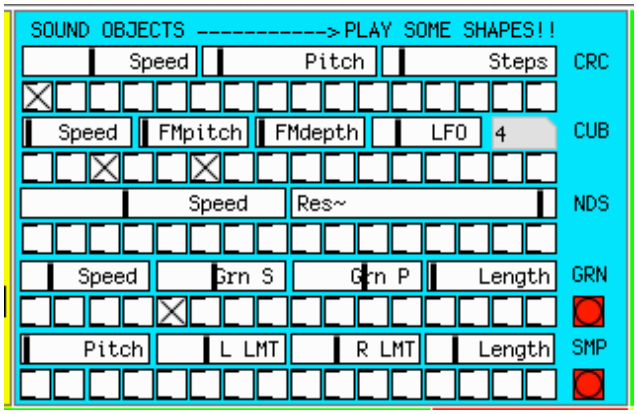
WORLD: ON (create small GEM window), FS (create full screen window-second screen), R (reset-destroy the GEM window), 3D (World's light), BkG (changes the background colour).



(fig.23)

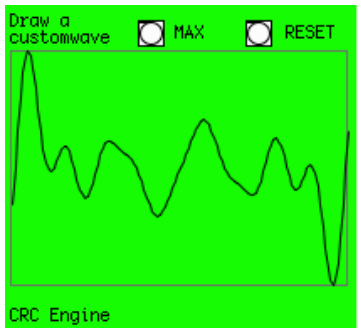
SOUND OBJECTS SECTION

Each sound object correspond to a dedicated sound engine, which uses a sequencer to trigger its envelope (fig.24).



(fig.24)

CRC: A custom wave engine can be drawn in using the mouse' pointer. MAX will maximise its amplitude, RESET will reset the waveform its default status (fig. 25).

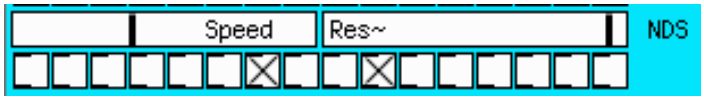


(fig. 25)

CUB: FM engine, the sliders modify pitch and depth. Speed modifies the sequencer tempo while the number box counts the number of steps (fig. 26).

(fig. 26)

NDS: Random Noise generator, Speed modifies the sequencer tempo. Res~ controls its [vcf~] resonance (fig. 27).

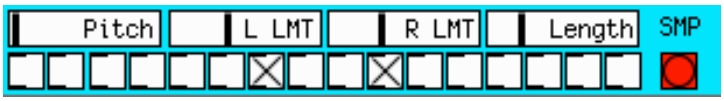


(fig. 27)

GRN: Grain synthesis engine. Press GRN (red bang) to load an audio file. Speed modifies the sequencer tempo. Grn S controls Grain Speed, Grn P controls Grain pitch, Length controls the decay of the envelope (fig. 28).

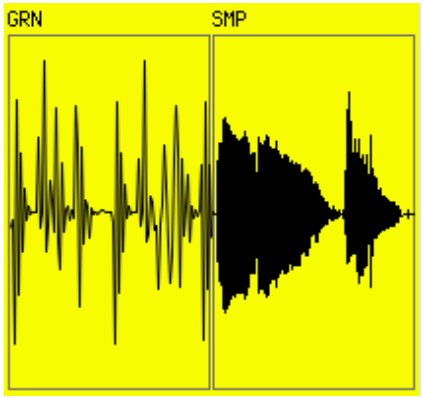
(fig. 28)

SMP: Sampler engine, Press SMP (red bang) to load an audio file. Length controls the decay of the envelope. L LMT controls the left limit of the sample while R LMT controls the right limit of the sample. Pitch controls the pitch of the sample (fig. 29).



(fig. 29)

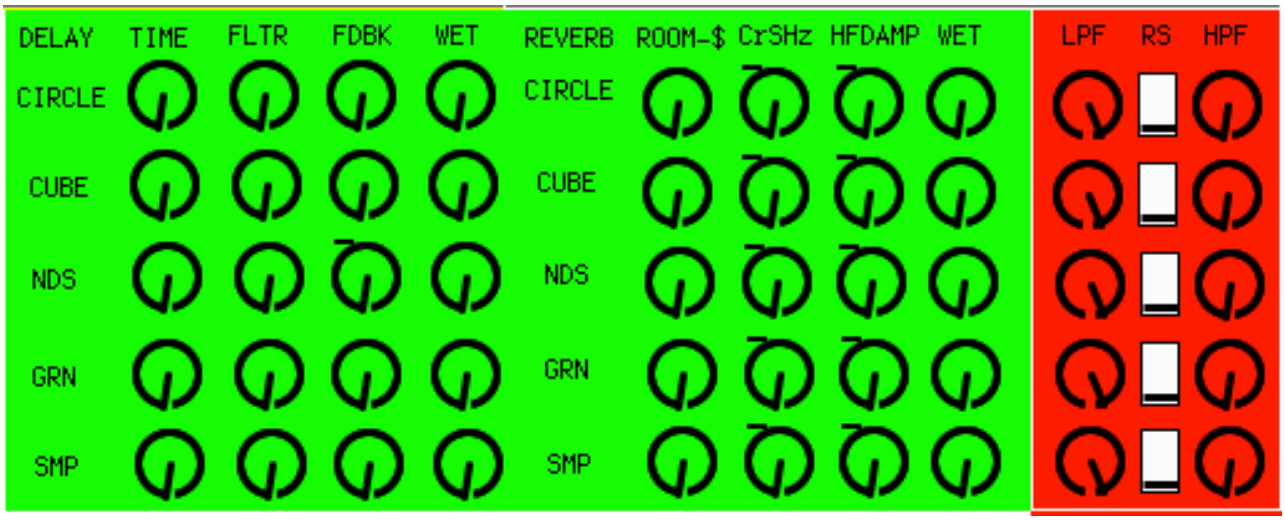
Both GRN and SMOP have dedicated arrays to view the samples once they have been loaded (fig. 30).



(fig. 30)

EFFECTS

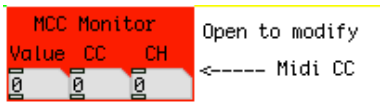
Each sound object has DELAY - REVERB - LPF (low pass filter) - RS (resonance) - HPF (high pass filter). DELAY and REVERB are Pre Volume fader (fig. 31).



(fig. 31)

For a best user experience is advised to use this patch with an external MIDI controller (e.g. Beringeer BRC 2000, fig. 32). Full MIDI implementation can be found in MCC patch (fig. 33).

(fig. 32)



(fig. 33)

Conclusions

I believe that the overall project, starting from the initial concept and ending with a video tutorial, was a success. I didn't expect to be able to think, design and code the system on my own. The final result is very similar to what I sketched at the beginning, but although I wanted to add some extra features, I did not have the time to implement them. Working with the GEM library was easy at first, but later became very challenging to debug, due to glitches and not many solutions to fix them (e.g. one single [part_colour] object interfering with all the other GEM objects). However I did not lose motivation at any point, because I was able to obtain constant feedback from the programming I was doing, both visually and sonically. While making the final patch (MIDI Control Change), I feel that I could have been tidier when programming the first patches. This is a lesson I will keep in mind for future projects. One area of improvement for this project would be to be able to further extend the adaptive capabilities of the instruments by including gestural control of the objects.

The visual element of the patch inspired me to complete the rest of the programming such as the master section, effects and the MIDI control change patches. The feedback I gathered from the presentation proved Pierre Schaeffer's point about the primacy of the sonorous object over its instrumental causes, which was the inspiration for making the GSO instrument. The making of this patch also helped me to develop a new method of analysis which can help to address technical challenges and optimise creative strategies. I feel that my technical and time management skills have improved by learning to program this instrument, giving me the knowledge required to undertake my major project, which I am looking forward to starting.

Bibliography

Clarity Films (2014) *BUCHLA documentary preview*. Available at: https://vimeo.com/96808503 (Accessed: 17 July 2015).

Cox, C., Warner D., (ed.) (2004) *Audio Culture : Readings in Modern Music. 2nd edn.* New York: Bloomsbury Academic

Farnell, A. (2010) *Designing Sound*. Cambridge, MA: MIT Press.

FLOSS Manuals (en) (no date) Available at: http://en.flossmanuals.net/ (Accessed: 03 July 2015).

Institut National Audiovisuel (no date) *Expérience de musique concrète [Coffret Pierre Schaeffer]*. Available at: http://www.institut-national-audiovisuel.fr/sites/ina/medias/upload/grm/mini-sites/schaeffer/co/experience.html (Accessed: 14 July 2015).

Kreidler, J. (2013) *Programming Electronic Music in Pd*. Available at: http://www.pd-tutorial.com/ (Accessed: 07 July 2015).

Henke, R. (2014) *Home page* [Online] Available at: http://roberthenke.com/ (Accessed: 02 November 2014)

Hillerson, T. (2014) *Programming Sound With Pure Data Make Your Apps Come Alive With Dynamic Audio*. United States: The Pragmatic Programmers.

MacLean, A. (2009) *Tidal* [Computer programming language]. Available at: http://yaxu.org/tidal/ (Accessed: 17 November 2014)

Puckette, M. (no date) *Pd Documentation*. Available at: http://msp.ucsd.edu/Pd_documentation/ (Accessed: 27 June 2015).

Schaeffer, P. (2013) *In Search of a Concrete Music*. United States: University of California Press

Sorensen, A. (2011) *Extempore* [Computer programming language]. Available at: http://extempore.moso.com.au/ (Accessed: 17 November 2014)

Tamimlehto, S. (1990) *Scream Trucker* [Music software packages]. Available at: http://www.hitsquad.com/smm/programs/Scream_Tracker/ and http://www.soundonsound.com/sos/jul04/ articles/pcmusician.htm (Accessed: 27 November 2014)

Whitney, J. (1980) *Digital Harmony: On the Complementarity of Music and Visual Art - John Whitney - Hardcover*. Peterborough, NH: McGraw-Hill Companies, The.

Wiggins, W. (2010) *John Whitney on 'The Screening Room'.* Available at: https://www.youtube.com/watch?v=BaW4DTKNflA (Accessed: 01 July 2015).