A Practical Study on the use of Emergent Behaviours towards the construction of Dynamic Independent Mapping Layers in Processing and Max/MSP.

Gregory White
6066402

University of East Anglia
School of Music
Music and Technology
W350

Sebastian Lexer
MUS-3Z1Y

Word Count: 8834
(Excluding appendices)

A research project submitted in partial fulfilment of the requirements for the Degree of Music.
May 2014

Acknowledgements

I would like to thank Sebastian Lexer for his continued support and understanding, not only in regards to this research project, but also throughout my three years of study at UEA. I must also thank Stephen Bennett for his suggestions and encouragement.

Table of Contents

Please Note:

The written work presented herein is presented in support of the practical exploration and research of my chosen subject. Please refer to the attached CD or https://github.com/gregcw/researchproject for examples of my practical work.

CD Contents

- Final Max Patch
- Interpolation Space Presets
- Final Processing Sketch
- Appendicised Code
  - Max-to-Processing Communication
    - 3rd Party
      - maxlink 0.36
      - mmo-visual-hacker-toolkit
      - MODIFIED_message_board

List of Figures

Research Project Documentation

1. Research Project Overview

Throughout the duration of this research project I focused on three main areas of research: creative coding with the Max/MSP and Processing languages; the control of audiovisual instruments through methods of parameter mapping; and generative and emergent processes in code-based artworks. Whilst my interest and research into these subjects is deep and goes beyond the confines of this course, this document will present an overview of only the information relevant to my practical work — granted I will retain some sources in the bibliography that, though they may not be directly referenced, nevertheless contributed to my overall understanding of the subjects at hand, and provide an indication of my broader engagement with them. Since much of the work conducted for this project was practical in nature, I have also included a diary of my progress to supplement my writing and where necessary I will refer to sections from it in this text.

This project began with my desire to research further into audiovisual instruments and the interactions that can occur between sound and image. This has been an interest of mine throughout my degree: searching for methods to combine my grounding in the visual arts with the audio processing skills learnt on this course, I have often implemented a visual component to my compositional and performance-based work.[1] Initially I set out to produce a completed generative audiovisual artwork or instrument, using a combination of the coding languages Processing (for visuals) and Max/MSP (for sound). But through my practical work I discovered new topics of great use and interest, which I decided to explore more thoroughly; as the project developed I saw it as an opportunity to investigate audiovisual instruments in greater detail, analysing how relationships between data, sound, and image can be built and controlled — particularly through methods of

[1] See my portfolio at http://www.whitenoises.co.uk/portfolio for examples.

parameter mapping and emergent systems — and decided to use this information as a foundation to inform my work post-degree.

I started working from a technical perspective — learning the basics of coding in Processing, object oriented programming, and so forth — which led me to discover concepts such as chaos vs. determinism and emergence. I then started thinking about how these technical processes and concepts could be applied artistically, and how they would result in a certain aesthetic. The culmination of my practical work is a set of concepts presenting how a particular algorithm, adapted from Matt Pearson's *Generative Art:A Practical Guide Using Processing*,[2] can be used as a "dynamic independent mapping layer"[3] for control of audiovisual instruments, using Ali Momeni and Cyrille Henry's 2006 article[4] as a framework. Due to the ineffective nature of attempting to convey moving image, sound, and their changing relationships through writing, I have submitted a video presentation of my concepts in aid of this text: in providing this video I aim to more effectively communicate my work and ideas than would be possible otherwise.

2. Languages

Once deciding on my areas of study, the first step of my research was to decide which coding languages I would use to carry out the practical exploration of these ideas. I felt that this research project would be a good chance to start investigating different coding languages and methodologies, with the aim of expanding my toolset to reveal new programming avenues for the

[2] Matt Pearson, *Generative Art: A Practical Guide Using Processing* (New York: Manning Publications Co., 2011).

[3] Ali Momeni and Cyrille Henry, 'Dynamic Independent Mapping Layers for Concurrent Control of Audio and Video Synthesis,' in *Computer Music Journal, Vol. 30, No. 1* (2006), 49-66.

[4] Ibid.

future. I chose to start learning Processing, a language with which I had no experience before this project, which I would use in conjunction with my existing skills in Max/MSP. The aim was to handle the control methods and audio processing in Max/MSP, whilst generating the visuals in Processing; not only would this give me the opportunity to learn a new language, but it would also cause me to research the different methods of communication between them and how information can be sent from one to another.

2.1 Processing

Consequently much of my time was dedicated to learning the basics of Processing, for which I drew upon a combination of four main sources: the aforementioned *Generative Art* book by Matt Pearson, the tutorials available on the official Processing website,[5] video tutorials by Jose Sanchez,[6] and *The Nature of Code* by Daniel Shiffman.[7] Most of my practical work follows Pearson's book, using the concepts and tutorials presented within as a basis for my final concepts (see section 5 of this document).

The decision to learn the Processing language was the result of a number of factors. First of all I wanted to learn a more conventional programming language of classes and functions to explore how it would affect my creative processes due to its text-based paradigm. With most of my coding experience being in Max/MSP, a language with a visual paradigm based on boxes and cables (see Figure 1), I wanted to challenge myself to a less-abstracted method of coding.[8] As well as providing

[5] Accessed at http://processing.org/tutorials.

[6] Jose Sanchez, *Processing Season 1* (2012), https://www.youtube.com/playlist?list=PL19223D55BA16ECDF, accessed 14 February 2014.

[7] Daniel Shiffman, *The Nature of Code: Simulating Natural Systems with Processing* (Magic Book, 13 December 2012).

[8] This is not to say that Processing is not an abstracted language — indeed it is a simplified and more accessible version of Java. However being a loop-based language that uses text as opposed to graphical objects, it is closer to lower-level languages like Java and C++ than Max/MSP is.

a different way of working, Processing would equip me with more transferrable skills: since it is a Java-based language, it can be used for creating web- and app-based work which I am interested in pursuing in the future. Therefore I saw the chance to use Processing as a worthwhile learning experience. Furthermore Processing is an open-source project built "for artists by artists",[9] with free access and a strong sense of community, and I wanted to be able to contribute in whatever small way possible towards its growth: for example I could post my findings of this research project on the forum section of the website,[10] which someone might then take and build upon to create something I would never have thought of — this sense of collaboration and partnership excites me. The forum is also full of members willing to help each other and answer questions, which in combination with the website's tutorials means there is an easy availability of help and educational resources available to beginners. All of these factors played an influence in my decision to use Processing as opposed to other programs or languages.

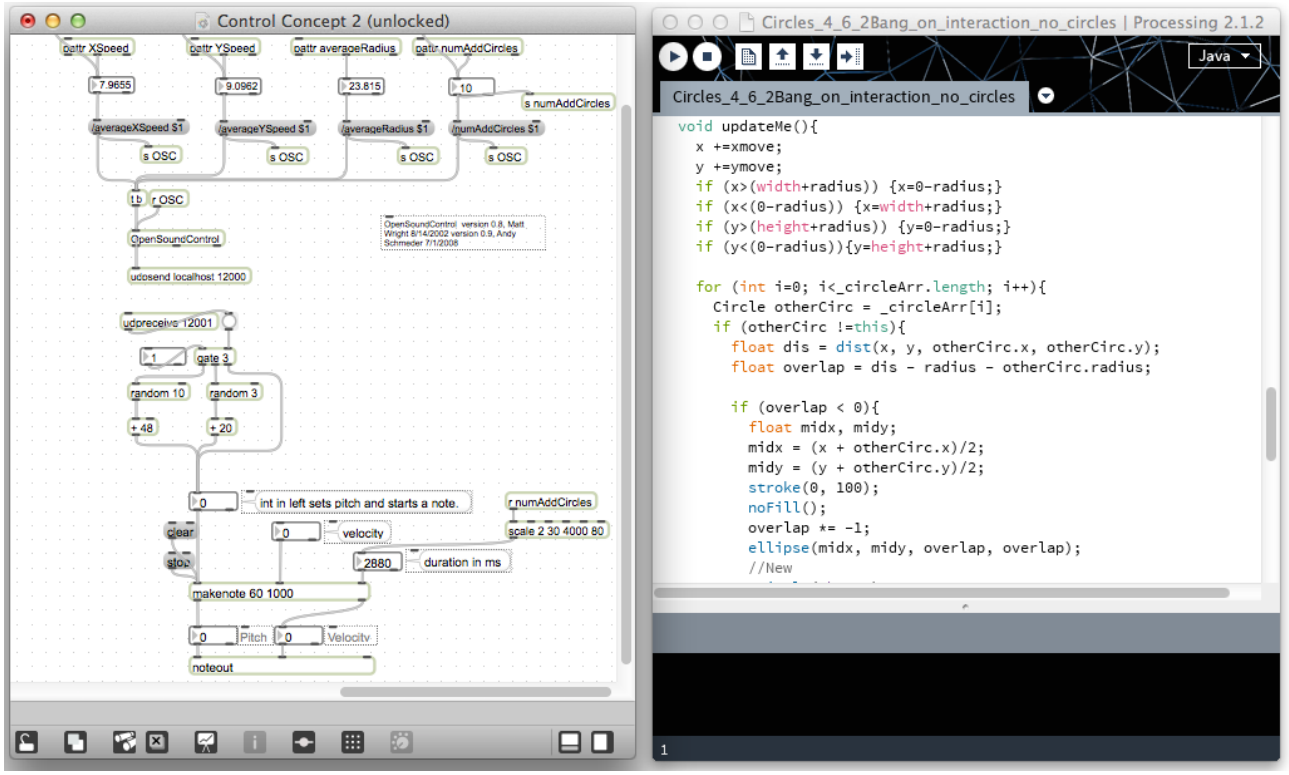

Figure 1: Comparing the paradigms used in Max/MSP (left) and Processing (right). The code shown is from my practical work, which can be found on the attached CD.

[9] Pearson, *Generative Art,* x.

[10] Accessed at http://forum.processing.org/two/.

2.2 Max/MSP

In conjunction with Processing, I made the decision to use Max/MSP. By using my existing skills and knowledge about audio processing and control methods in Max/MSP it relieved the pressure of having to learn about both audio and visuals in a new language, which would have been too much for me to have taken on given the scope of this project, and allowed me to work on the application of my skills in regards to parameter mapping (see section 3.3 of this document). But this decision was not merely a practical one: as mentioned above, by using Max/MSP and Processing in conjunction with each other I was forced to find ways for information to be sent back and forth between the programs, leading me to uncover a variety of different techniques and determine which one would be suitable for this particular project.[11]

3. Dynamic Independent Mapping Layers

Before detailing my practical work and the concepts that arose from it, I will first give an overview what I feel are the most relevant ideas I discovered when researching methods of controlling digital instruments and parameter mapping, presented by Ali Momeni and Cyrille Henry in their article *Dynamic Independent Mapping Layers for Concurrent Control of Audio and Video Synthesis.*[12] This article was particularly influential to me in this regard, and played a large part in my shift of focus from creating a completed artwork to exploring in more detail the different models and methods of controlling audiovisual instruments.

[11] See Progress Diary section 6.1, 7.2.

[12] Published in *Computer Music Journal, Vol. 30, No. 1* (2006), 49-66.

3.1 Article Outline

In their 2006 article for *Computer Music Journal*, Momeni and Henry aim to outline a number of concepts and techniques that enable "intimate and expressive control"[13] of audio and video synthesis simultaneously. They begin by breaking down the relationship between sound and image into three categories: "Most works can be characterised as [1] sound-to-image, [2] image-to-sound, or [3] concurrent generation of sound and image." It is clear that they believe concurrent control allows for a "rich" and "intrinsic connection" that is perhaps more sophisticated than the "unidirectional" relationships achieved through sound-to-image and image-to-sound control. Therefore they address an issue which had troubled me in my own work, of moving beyond a 1:1 relationship between image and sound that can quickly become predictable or even gimmicky. Whereas the majority of my previous audiovisual work could be categorised as sound-to-image, using sound to manipulate a video source, I wanted to use this research project as a chance to engage with methods that allowed for a deeper, more complex relationship. Consequently I found this an extremely helpful and influential article which has affected my work beyond the confines of this assignment (specifically my Solo Snare project for the Performance with New Technology module[14]), and will continue to guide my audiovisual work post-degree.

The primary method of achieving a rich concurrent control outlined by Momeni and Henry is that of a "dynamic independent visual mapping layer".[15] Whereas the methods of parameter mapping that I had previously been implementing were relatively basic, relying on the 1:1 mapping of a fader to increase values or using a button to trigger an event, Momeni and Henry present the

[13] Momeni and Henry, *Dynamic Independent Mapping Layers,* 49.

[14] More information about this project and its development can be found on my website: www.whitenoises.co.uk.

[15] Momeni and Henry, *Dynamic Independent Mapping Layers.*

idea of an intermediary dynamic mapping layer[16] that allows for a higher dimensionality of output, but still allows for expressive and accessible control through low-dimensional input.[17] This dynamic mapping layer provides a model or system which the performer influences through "classic"[18] mapping methods; the dynamic mapping layer then reacts to these influences, potentially providing a large set of information depending on its programmed behaviours. This information is then used to control parameters of video and audio synthesisers, again through classic mapping techniques. Therefore the performer is granted a greater degree of control or multiple parameters in both the audio and video realms, simultaneously.

3.2 Physical Models

Momeni and Henry provide a number of physical models as examples of dynamic mapping layers, particularly interested in mass-spring models.[19] While I did not choose to implement a mass-spring model in my concepts the methods of control they afford are much the same, so I feel it is worth explaining. The basis of a mass-spring model is the relationship between a group of linked objects (or masses), where changing the behaviour of the springs that connect them (length, resistance, and so forth) affects how the masses behave:

> Consider the example of a mass-spring structure made of a two-dimensional grid of 81 masses arranged in a 9 x 9 matrix [see Figure 5]. Adjacent masses in the *x*-*y* plane are

[16] Ibid., 50.

[17] Or to put it more simply, the aim of a dynamic mapping layer is to allow the simultaneous control of many parameters while only requiring a small degree of input from the performer.

[18] Momeni and Henry, *Dynamic Independent Mapping Layers*, 50.

[19] Ibid., 58.

linked with a spring, and the four corner masses are stationary, thus stretching the structure to form a kind of flexible surface.[20]

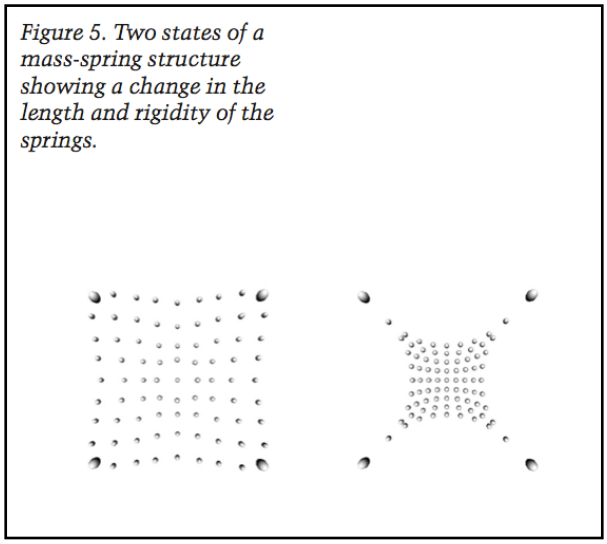

Figure 2: Example of mass-spring structure referenced in the above passage, from Momeni and Henry, Dynamic Independent Mapping Layers.

With this example model in place a number of methods for control are suggested, each affording a different level of sophistication.

The most low-low level manner of controlling the behaviour of this system would naturally lie in controlling the positions of each of the 81 masses, thus giving complete control and deterministic results. A second manner, […] would be to control one of the masses and allow its modelled physical interaction with the other part of the structure to drive the system's behaviour. A third method — one that benefits from features particular to mass-spring models — is to control with four input parameters the resistance, length, and damping constants — one for each end of a spring — for all the springs in the system. This control, in conjunction with the direct manipulation of the

[20] (Text and Figure) Ibid., 55.

> position of one of the masses — or any other way of adding energy to the system — allows for control of dramatic changes in the structure's behaviour.[21]

Here Momeni and Henry show how to go beyond basic control of a system that gives deterministic results, towards a method that yields more complex data but still remains manageable. The third method of control is most relevant to my concepts as it is concerned with manipulating the parameters of a set of objects, and the resulting changes in the emergent behaviours caused by this manipulation (see section 5 of this document).

3.3 Interpolation Spaces

They also argue that interpolation spaces can be used as an effective method of controlling these physical models, providing a number of implementations. An interpolation space is a virtual area, typically two- or three-dimensional, which contains several defined sets of information (for example, if the interpolation space were being used to directly control an audio synthesiser, each set of information might contain a frequency, the number of voices, and overall volume). The performer is then able to navigate this space, and as they do so they are provided with a smooth interpolation of the values defined (see Figure 3). Therefore they are able to achieve a high degree of control (using the data from the physical model) with a low degree of input (x-y or x-y-z coordinates, depending on the dimensionality of the space). This is the method of control I have employed in my practical work as outlined in section 5 of this document, and was also used in my final project for the Performance with New Technology module.[22] But whereas then I was using an interpolation space to directly alter the sound, in this project I am using it to control my dynamic

---

[21] Ibid.

[22] More information about my implementation of interpolation spaces in my Performance with New Technology project can be found in the post titled 'Performance with New Technology: Developing Final Performance' on my website: www.whitenoises.co.uk.

mapping layer concepts — the information from which can then be applied to video and audio synthesis concurrently.

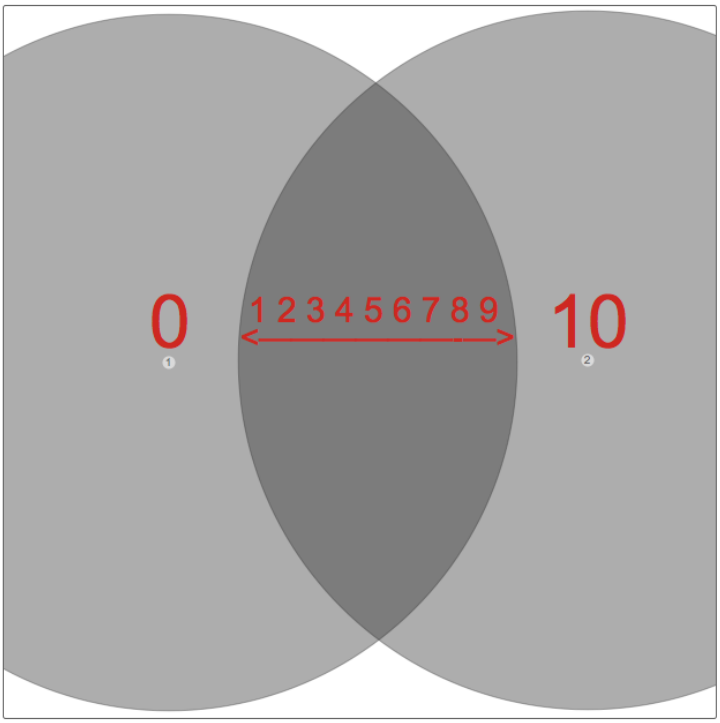

Figure 3: Illustrating interpolation of values between two points in a 2D virtual space, using the nodes *Max/MSP object.*

4. Generative Art

Matt Pearson's book *Generative Art* provided the foundation for me to learn how to code using Processing, and presented a number of interesting concepts along the way concerning generative methods. Generative art is loosely described as a merging of art and programming, "between man and machine"[23]; the programmer-artist defines a system of rules or behaviours that, when information is fed into them, produces unpredictable yet organically-evolving results. It is then up to us to determine what is of aesthetic value, adjust the rules accordingly, and continue on with the process. Pearson introduces a range of concepts that apply to the construction of generative

[23] Pearson, *Generative Art,* xviii.

systems, from randomness and noise to autonomy. But the concept that resonated most with me, and provided a basis for my practical concepts, was emergence.

4.1 Emergence

Pearson provides a succinct definition of emergence as "the observation of how complex and coherent patterns can arise from a large number of small, very simple interactions."[24] Providing the flocking behaviours of starlings over Brighton's West Pier as a real-world example, he teaches that a small set of rules on the micro level — "orienting themselves in relation to their immediate neighbours, seeking the centre of the flock, avoiding the paths of other birds," — can result in interesting behaviours on the macro level. Pearson is keen to stress that "these macro patterns aren't formed through any central design or intent: they're nothing more than *byproducts* of the local self-interested behaviours of the individuals collectively," — and perhaps that is why we find them so compelling.

Emergence is therefore not a phenomenon born out of creative coding,[25] and has been applied to nature, politics, and other art forms[26]; fine artist John Franzen's work *Each Line One Breath*[27] demonstrates how emergent results can be (literally) drawn from no more than a piece of paper and a pen: "Starting with a straight line, each sequence that follows is created by drawing another line that seeks to imitate its successor. Concentration and the repetitive act of inhaling and exhaling during each line is that which lets the lines evolve into logarithmic-like patterns and

---

[24] Ibid., 108.

[25] Indeed, Pearson notes "the term was first coined in the mid-19th Century by George Henry Lewes,"* far before the invention of the first computer.
* Ibid.

[26] Ibid., 110-111.

[27] John Franzen, *Each Line One Breath* (2011), http://www.johnfranzen.com/art/EACH_LINE_ONE_BREATH.html.

Figure 4: Each Line One Breath N° P 12 *(left) and* N° A 29 *(right), demonstrating emergent patterns.*[28]

layers."[29] However, the idea of using a set of simple rules to create complex results is obviously highly relevant to generative art. Pearson provides Craig Reynold's *Boids* algorithm as an example of how the natural emergent behaviours of animals flocking can be implemented in the world of code:

> Reynolds discovered that to produce a realistic flocking simulation in code, he needed only three rules:
>
> - *Separation* — Steer to avoid your immediate neighbours.
> - *Alignment* — Steer to align with the average heading of your immediate neighbours.
> - *Cohesion* — Steer toward the average position of your immediate neighbours.

[28] Franzen, *Each Line One Breath*, http://www.johnfranzen.com/art/EACH_LINE_ONE_BREATH.html.

[29] Ibid.

> More complex rules can be added (object avoidance, for example), but these three are all that are required to achieve a complexity comparable with the West Pier starlings.[30]

But it is Pearson's own implementation emergent behaviour that I have adapted to create my dynamic independent mapping layer concepts. I will now provide an overview of the code Pearson provides, and how I have used it as a foundation for these concepts.

5. Circles

Pearson's practical tutorial on emergence begins by introducing the idea of object-oriented programming[31] in order to create a system whereby circles of random sizes travelling in random directions are created when the user clicks the mouse. By defining a 'class', the programmer creates a set of properties that new objects — in this case, circles — will adhere to: size, colour, direction, speed and so on. With this class in place, Pearson demonstrates how it is possible to determine behaviours that occur when these objects interact, which, when observed en masse, are emergent: for example, the colliding circles could fade, they could leave a trace at the position of impact, or a new circle could be become present at the midpoint of their collision. This last interaction is the one I have implemented in my concepts.

5.1 Implementing Pearson's Code as a Mapping Layer

*For the video presentation of my mapping layer, please refer to the attached CD, or https://github.com/gregcw/researchproject.

---

[30] Pearson, *Generative Art,* 109.

[31] See Progress Diary section 4.2.

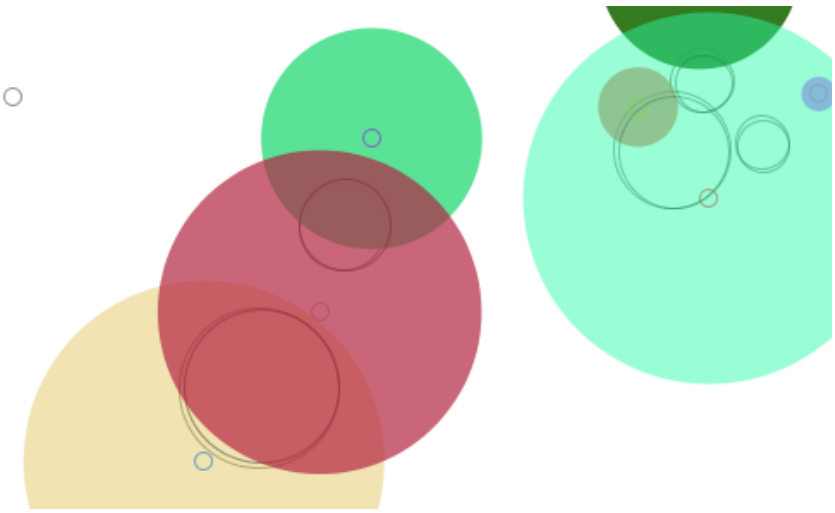

Figure 5: The result of Matt Pearson's code in which new circles appear at the midpoint of a collision (screenshot my own).

I decided to explore how this particular interaction could be applied artistically towards concurrent audio and video synthesis, acting as a dynamic independent mapping layer as outlined by Momeni and Henry. The first step was to enable control over the parameters of the circles class, in turn allowing me to influence the emerging behaviours. I decided to develop the control system in Max/MSP so that it could be used with the Mira touch controller iPad app,[32] resulting in a physical connection with the system that works toward an "intimate and expressive control,"[33] — which is the underlying principle that justifies using dynamic independent mapping layers to begin with. For this reason, combined with my aforementioned desire to utilise my audio processing skills learnt in Max/MSP, I had to research into methods of sending information back and forth between Processing and Max/MSP simultaneously.[34] After achieving the desired connection with OpenSoundControl, I was able to start combining my work on parameter mapping and instrument control (in Max/MSP) with what I had learnt about generative art and emergent behaviour (in Processing.

---

[32] Sam Tarakajian, Mira touch controller iPad app (2013), http://cycling74.com/products/mira/.

[33] Momeni and Henry, *Dynamic Independent Mapping Layers,* 49.

[34] See Progress Diary section 6.1, 7.2.

For all of the concepts presented, I decided to employ Momeni and Henry's concept of using interpolation spaces to smoothly transition between presets, which I had become familiar with through my final project for the Performance with New Technology module.[35] I adapted Pearson's code so that the values used for the circle class variables — size, speed and so forth — were sent from Max/MSP: This meant I could use an interpolation space to smoothly travel between sets of predetermined values, which would then be sent to Processing and affect the mapping layer. By altering the parameters of a continuously moving system, I am able to explore the emergent behaviours that occur when the objects interact: this is the basis of my concepts. Utilising emergent behaviours allows for a more complex and unpredictable final result than typical 1:1 mapping, but is still a controllable solution given the use of an interpolation space to move between predefined parameters. Or to put it more simply, I am able to provide a degree of control by changing parameters which in turn influences the emergent results. However, being new to Processing and still learning the logic it requires, I have not managed to achieve the exact response I desire from the mapping layer, which requires some additional alterations to Pearson's code.[36] But the greatest challenge I encountered was extracting information from the mapping layer to send back to Max/MSP for synthesis; whilst I have been able to yield some information (specifically, Processing communicates to Max/MSP when an emergent circle is created[37]), I am still unable to report values such as the x-y coordinates of the circles on screen,[38] which is vital to most of my concepts. Consequently I cannot present them as 'proof-of-concepts' yet, and I must develop them further.

---

[35] Again, more information about how I used interpolation spaces in this project can be found in the post titled 'Performance with New Technology: Developing Final Performance' on my website: www.whitenoises.co.uk.

[36] See Progress Diary section 8.1 for a breakdown of the changes desired and my attempts to implement them.

[37] See Progress Diary section 9.2, under model 2: 'Collision = sound'.

[38] See Progress Diary section 9.3.

I see a number of other issues with this system. Firstly, the values are only updated in Processing when the user clicks on mapping layer, instead of when they are altered in Max/MSP.[39] This severely limits the fluidity and expressiveness of control afforded by the system, and is the main obstacle I am currently facing. A less pressing issue, but one still considering, is that some of the mapping is still 1:1 — specifically the amount of circles, their x and y speeds, and size. These values are defined by the user in Max/MSP, and consequently will be fed directly to the audio and video synthesisers. Though these parameters do affect the mapping layer and therefore the information output as a whole, I feel it worth highlighting that they do not benefit directly from the affordances of dynamic mapping layers.

5.2 Concepts

The complete list of data I intend to extract from this mapping layer is as follows:

Original Circles	Emergent Circles
Amount	Amount
X Speed	X Speed
Y Speed	Y Speed
Size	Size
Direction of movement*	Direction of movement*
X-Y position*	X-Y position*
Density*	Density*
	Rate of change in size*

Figure 6: Information that could be extracted from mapping layer and used to control synthesis. An asterisk indicates values which are not yet reported.

This information can then be applied artistically towards the control of audio and visual synthesis in a variety of different ways; I will now describe three concepts for audio control, and three for video control.

[39] See Progress Diary section 8.1.

5.2.1 Audio Concept 1: Continuous Sound

Sound is continuously generated by the original circles, each corresponding to a voice, sample, or set of grains. Parameters such as pitch and playback speed could be mapped directly to any of the above variables: for example, the faster the circles move, the higher the playback speed of the sample it represents. Meanwhile the larger the circle's size, the greater the volume of its corresponding sound. Emergent circles represent a different synthesiser, instrument, or set of samples.

This is the most basic implementation of the mapping layer for synthesising, and I feel does not benefit enough from the emergent behaviours it was designed to exploit. Rather, there is too much 1:1 mapping present and the interaction is not rich or dynamic enough. Perhaps it would be more interesting to use the density of circles in a particular area, mapping each area to a sound. This may help avoid the system becoming too representational or predictable.[40]

5.2.2 Audio Concept 2: Collisions

Sonic events are triggered when collisions occur. These events could take the form of synthesised notes (by sending MIDI messages when emergent circles are created), predetermined patterns, or samples. The length of the sonic event is inversely proportional to the amount of circles: fewer circles means there is less chance of a collision occurring, so events are made longer to compensate, avoiding long silences; as the number of circles increases so too does the amount of collisions — consequently events become shorter and more percussive so as to prevent muddiness and provide the performer with a method of changing atmosphere.

[40] See Progress Diary 6.2.

This idea is still basic in concept, relying heavily on triggering events, but can be utilised in more interesting ways. It allows the user to employ a range of sounds, rather than just synthesising notes, including the use of precomposed motifs; therefore I feel it to be more powerful and applicable than the previous concept. It is also displays the benefits of using mapping layers in creating a direct relation between audio and sound, as it would be exceptionally easy to synchronise visual events with sonic ones by using the same triggers. However it is still too focused on the micro level, not taking into account the emergent behaviour on the macro level. Again, this could be improved somewhat by using density information, perhaps mapping events to areas of the screen and only triggering them once a certain amount of collisions have occurred in the corresponding area.

5.2.3 Audio Concept 3: Effect Control

This concept can be viewed as a combination and extension of the previous two. Sonic events are represented by the original circles on screen, as in the first concept. When collisions occur, an effect is applied to the corresponding events. For example, two circles — one representing a note on a synthesiser, the other representing a sample — collide, and for the duration of the emergent circle's lifespan, they become distorted. One of the effect's parameters, such as the frequency bands it affects or its wet level, could be controlled by the size of the emergent circle: in keeping with the example above, the distortion could start off affecting only the high frequencies, sweeping down to lower frequencies as the circle grows larger, before sweeping back up as it shrinks.

Alternatively, there could be two classes of circle: sounds and effects. When an effect collides with a sound, the wet level of the effect changes in accordance with the size of the emergent circle. This could produce an interesting emergent effect if the sound circles represented

frequency bands of the same audio source, as the effect would shift in and out of different parts of the sound.

I believe that this concept would work well in combination with a more developed system, but is not rich enough to stand on its own. More than the other concepts it is limited to the micro-level behaviours, though it may produce an interesting sonic effect on a macro-level.

5.2.4 Video Concept 1: Mapping Layer

As noted by Momeni and Henry,[41] visual mapping layers can also be used directly in video synthesis. The mapping layer could be implemented untouched or abstracted. Pearson suggests a number of abstractions, including the removal of the original circles so that only the emergent ones remain, popping in and out of view.[42] Another possibility he provides is removing the code that clears the screen between each frame, therefore allowing the emergent circles to build up and form larger patterns.[43]

I do not see much use in this particular mapping layer if left untouched, as the interactions do not change in nature and consequently become uninteresting rather quickly. But by abstracting the system and allowing the emergent patterns to become apparent, this could become a particularly effective solution.

[41] Momeni and Henry, *Dynamic Independent Mapping Layers,* 56.

[42] Pearson, *Generative Art,* 123.

[43] Ibid., 124-5.

5.2.5 Video Concept 2: Video Synthesis

Here the screen area could be broken down into several subsections, each representing a particular visual object or parameter, such as colour, saturation, or opacity. When collisions occur in these subsections, the corresponding object is synthesised, or the corresponding parameter is affected in accordance with the data produced by the emergent circles; for example, the size of the emergent circle could be directly related to the size of an object or the saturation of an image. In the event of multiple collisions occurring in the same area simultaneously, whereby the area represents a parameter, the data used to affect the parameter's value will be taken as the average from all collisions.

This method is limited by the amount of subsections one can divide the screen area into. However it could bring to attention the density of collisions occurring in particular areas, and hence emergent behaviours.

5.2.6 Video concept 3: Video Warping

This final concept uses the information from emerging circles to control affects applied to a separate video source. One can imagine that the video source is placed on top of the visual mapping layer; where emergent circles appear, an effect is applied. The overall emergent effect will be of moving streaks that bulge in and out, altering the image before disappearing again.

A more sophisticated version of this concept is provided by Momeni and Henry,[44] whereby the behaviour of the mapping layer deforms a membrane, "whose displacement from the z-axis was mapped to the repositioning of the corresponding pixel of the source image."

[44] Momeni and Henry, *Dynamic Independent Mapping Layers,* 56-57.

My version of this concept is perhaps too gimmicky and superficial, and I do not see a great potential for its application beyond demonstrating a proof-of-concept. However, Momeni and Henry's version, particularly if viewed in constant motion, could produce an arresting psychedelic visual effect.

5.3 Evaluation of Mapping Layer

I will now evaluate my adaption of Pearson's code as a dynamic independent mapping layer, in accordance with the benefits as specified by Momeni and Henry.[45] In doing so, I present a number of ways my system could be extended so as to become more effective.

5.3.1 Performing Low-Dimensional to High-Dimensional Mappings

> The primary goal of our proposed model of mapping is to allow exploration and expressive control of a high-dimensional parameter space using a low-dimensional gestural controller. This involves some form of dimensional scaling that allows one to transform *n* inputs to *m* outputs.[46]

By implementing an interpolation space containing sets of predetermined values, I am able to scale two inputs (x and y coordinates) to as many outputs as I desire. Currently I am sending using four outputs which I am attempting to map to average speed in the X direction, average speed in the Y direction, average circle size, and number of circles. Momeni and Henry's intention is that significantly more outputs can be utilised, but currently I am investigating the relationships that can be altered using just these four parameters. If necessary, I could develop this further by utilising

[45] Ibid., 54-57.

[46] Ibid., 54.

other parameters in Pearson's code (like circle colour) or introduce new ones to grant me an even greater degree of control.

The "exploration and expressive control" of the system is enhanced by my use of the Mira iPad app. The user is presented with a square area which will track the movement of up to five fingers simultaneously, reporting their x and y position in the interpolation space; they can move about this space by dragging, resulting in a smooth interpolation, or tapping, allowing for sudden drastic changes in position. Currently I am only using data from one finger, but if I wanted to expand this control method by using the possibilities afforded by tracking five fingers I could do so in numerous ways: perhaps most obviously I could calculate an average position taking into account all of the active fingers, using this data as the final position in the interpolation space; or else I could report gestures such as swipes and pinches[47] that trigger certain preset movements to occur — for example, swiping to the right with three fingers could cause my position in the interpolation space to move from the leftmost point to rightmost in exactly three seconds, thus allowing precise timing of events.

5.3.2 High-Level Control

> Because we attempt to control a complex independent system with low-dimensional input,it is imperative that the behaviour of our algorithmic independent mapping layer be controllable at a high level.[48]

[47] The help files for the `Mira.mt.fingers` Max object provide a starting point for such gesture recognition, but might need more development in this instance.

[48] Momeni and Henry, *Dynamic Independent Mapping Layers,* 55.

I am implementing a high level of control since I am not, for example, dictating the position of each circle, which Momeni and Henry label "low-level." Instead I attempt to affect the overall characteristics of the system — average speed, average size, and so forth. However I believe that a higher level of control could be achieved by constructing a way of "adding energy to the system" to more directly influence its behaviour. Applying one of their examples, I could enable the movement of one circle, in conjunction with my existing control, which would then alter the behaviours of the others: for instance, this controllable circle could cause the others to bounce off of it, altering their paths and allowing users to manipulate the collisions that occur (or even prevent them from occurring).

5.3.3 Time-Variable Behaviour of Structure

> The first goal of this approach to mapping is to allow exploration of a greater subspace of a rich audio/video instrument's complete parameter space. […] The second overarching goal is to integrate a sense of time in the mapping layer itself […] the user puts energy into the system by affecting the structure in some way with a controller; once left alone, the structure's behaviour is a natural continuation of its previously controlled behaviour.[49]

I feel that this area is where my system is weakest: it lacks dynamism. If left alone, the system does indeed produce data, but its behaviour does not change over time without human input; therefore I do not feel it is correct to label it as a *dynamic* independent mapping layer. However, Momeni and Henry provide two possible solutions. First of all is the ability to create a subspace of the overall system, which moves over time. Data is only collected from this small area as opposed to the whole

[49] Ibid., 56.

mapping layer: for example, if my system produced an area of 1200x900 pixels, I could create a subspace of 400x300 pixels, effectively limiting my data potential output to a third of the total; starting at the top left, this window could move over to the bottom right throughout the duration of the piece. Yet I do not feel that this would be a particularly effective solution to apply to my mapping layer as it currently stands, since there is no particular variation across the screen — the movement of the circles is random. If, however, I built a method of attracting the circles to particular points on screen — like a moveable centre of gravity — I would be afforded a degree of control over their overall placement or density, which could then allow interesting results if viewed from a moving window.

The second solution posited by Momeni and Henry is to enable the "natural continuation of previously controlled behaviour." Continuing with the idea of a gravitational field, this could be implemented in my mapping layer by enabling a more direct control over individual circles. By placing a subset of circles in an orbit around a central point, they would continually move in a predictable pattern. I could then alter their speed or how strongly they are affected by the gravitational pull in order to affect their orbit, which could then gradually return to its original state over time. Meanwhile, another subset of circles could be travelling normally, unaffected by the gravity as in my presented concepts; when these circles interact with those in orbit, collision data could be reported.

5.3.4 Visual Mapping Layers

> We are especially interested in dynamic mapping layers that have a graphical metaphor and thus a visual representation that can act as an interface for the mapping layer. […]

> We believe that a visual paradigm and representation of the mapping layer eases exploration and expressive control of the system.[50]

My mapping layer is visual in nature, displaying the interactions and behaviours of the objects defined in the code. Momeni and Henry observe the added benefit of applying visual mapping layers directly to the video synthesis: "Varying degrees of abstraction from the direct visualisation of the mapping layer […] can serve as a continuum of possibilities for video generation." Indeed this is the case with my concepts, as displayed above. Furthermore the interpolation space aids in making the system "intuitively easy to understand and manipulate" by clearly marking distinct areas containing sets of information, allowing the user to navigate as they see fit.

5.3.5 An Intrinsic Link Between Generated Audio and Video

> Central in our approach to expressive control of synthesis is the concurrent control of video and audio using a single independent mapping layer. We seek a rich relationship between generated audio and video.[51]

In all of my concepts, the audio and video synthesis are controlled simultaneously by the same mapping layer. Consequently I achieve an "intrinsic connection between the audio and the video that is a direct result of their common sources of control," as opposed to the unidirectional sound-to-image or image-to-sound mappings I am trying to move away from.

---

[50] Ibid.

[51] Ibid., 57.

6. Conclusion

To conclude I will remind the reader that throughout the duration of this research project I have focused on three areas of study: learning the basics of Processing and how to use it in combination with Max/MSP; advanced control of audiovisual instruments that goes beyond sound-to-image or image-to-sound relationships; and generative and emergent processes in code-based artworks. These topics were not viewed in isolation, but have been combined in the form of my practical work. I have presented a number of concepts outlining how Matt Pearson's code could be implemented as a mapping layer for concurrent audio and video synthesis, using Momeni and Henry's article as a framework. Moreover I have suggested a number of ways in which the mapping layer could be extended so as to more effectively address Momeni and Henry's concerns.

I believe that I have indeed made progress in understanding Processing and more conventional programming languages, using techniques such as object-oriented programming. However I do not yet feel as comfortable with the logic required to realise many (though not all) of my ideas as I do in Max/MSP; though this is to be expected given how much more experience I have with the latter. Throughout my practical work I was able to problem-solve and develop solutions much quicker and without as much research in Max/MSP, whereas with Processing I struggled. Moreover I still feel that the visual paradigm that Max/MSP implements is easier to follow than Processing's conventional one, and I find that I am able to remember the thought processes and intentions behind the code more; I believe that, in my case, this is because Max/MSP enables me to arrange processes and objects in a very precise way that represents the flow of information, whereas Processing has me scrolling back and forth between chunks of text. Furthermore in Processing I found the difference between how a human and a computer interpret data to be more of an issue: the computer requires everything to be declared and defined precisely, with very specific instructions on how to complete a task, whereas I found myself jumping ahead —

for example, once defining the circles class and adjusting its parameters, I expected to be able to observe them straight away; however they were only created conceptually in the computer's memory, and it had to be instructed to draw them (again, very specifically). This is a problem I came across many times, which I feel is easier to avoid in Max/MSP since one can clearly see the flow of data: if an object creating circles is not linked to a video window, you know you are not going to be able to observe them. I do recognise that these problems are perhaps more common to the beginner, and with time will become less frequent as I become accustomed to the demands of the language.

I am most satisfied with my research into concurrent control of audiovisual instruments, as I feel I have found a deep source of information and inspiration for future work in Momeni and Henry's article. Directly addressing my desire to move beyond mapping sound analysis to video control and vice versa, they provide not only a conceptual framework but also practical examples of how to implement these ideas successfully. I fully intend to further explore the use of mass-spring models in particular, and indeed have already downloaded pre-built models to analyse and adapt to my own needs. Furthermore the bibliography contains a wealth of writing concerning parameter mapping and digital instruments, many of which I have obtained from their respective sources for further reading.

Additionally I believe Matt Pearson's *Generative Art* was successful in providing an introduction and basic understanding of generative art and a range of the concepts associated with it, including noise and randomness, autonomy, and fractals. I feel ready to address the topics I found most interesting — particularly emergence — in more detail, for which I will have to look elsewhere. Pearson successfully applied his teaching of the basics of Processing by exploring these concepts, though I do feel that his explanations were a little unclear at times, perhaps assuming too

much familiarity with traditional programming conventions at certain points. Consequently I had to use other sources, such as Jose Sanchez's video tutorial series[52], to clarify some concepts. Moreover I do not feel that I have explored the use of emergent behaviours in mapping layers as successfully as I had hoped, becoming more fixated on figuring out how to retrieve data from the micro-level behaviours. My intention was to observe this data on a larger scale to see the patterns that emerge on the macro-level, but my relative inexperience with Processing prevented me from getting this far.

In evaluating my research project I feel that I was perhaps overambitious initially, in attempting to create a dynamic generative artwork using a new language. And though I am pleased with the progress that I have made and knowledge I have gained in the three areas specified, I do feel that each could have been a project on its own, and that tackling them all at once has meant having to sacrifice progress in some areas at the expense of others. Where I feel most dissatisfied is my inability to provide working proof-of-concepts that display my ideas exactly as intended; though as I develop my skills in Processing, this will surely become a manageable task.

[52] Sanchez, *Processing Season 1* https://www.youtube.com/playlist?list=PL19223D55BA16ECDF.

Bibliography

Bayle, Julien. *Approaches of A/V Performances.* http://julienbayle.net/works/workshop/approaches-of-av-performances/. Accessed on 6 November 2013.

Franzen, John. *Each Line One Breath.* 2011. http://www.johnfranzen.com/art/EACH_LINE_ONE_BREATH.html.

Gillies, Marco. Yee-King, Matthew. Grierson, Mick. *Creative Programming for Digital Media & Mobile Apps.* https://class.coursera.org/digitalmedia-001/lecture. Accessed 20 November 2013.

Grosse, Darwin. 'Podcast 017: Julien Bayle'. *Art + Music + Technology Podcast. https://itunes.apple.com/gb/podcast/art-+-music-+-technology/id736102938?mt=2.* 3 February 2014.

Grosse, Darwin. 'Podcast 024: Ali Momeni'. *Art + Music + Technology Podcast. https://itunes.apple.com/gb/podcast/art-+-music-+-technology/id736102938?mt=2.* 3 February 2014.

Hunt, A. and Kirk, R. 'Mapping Strategies for Musical Performance'. In *Trends in Gestural Control of Music*. Eds. Wanderley, M.M. and Battier, M. Ircam Centre Pompidou: 2000.

Magnusson, T. 'Designing Constraints: Composing and Performing with Digital Musical Systems'. In *Computer Music Journal, Vol. 34, No. 4.* Winter 2010. pp. 62-73.

Momeni, A. and Henry, C. 'Dynamic Independent Mapping Layers for Concurrent Control of Audio and Video Synthesis'. In *Computer Music Journal, Vol. 30, No. 1.* 2006. pp. 49-66.

Momeni, A. and Wessel, D. 'Characterizing and Controlling Musical Material Intuitively with Geometric Models'. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*. Montreal, Canada: 2003. pp. 54–62.

Pearson, Matt. *Generative Art: A Practical Guide Using Processing.* New York: Manning Publications Co. 2011.

Sanchez, Jose. *Processing Season 1*. 2012. https://www.youtube.com/playlist?list=PL19223D55BA16ECDF. Accessed 14 February 2014.

Shiffman, Daniel. *The Nature of Code: Simulating Natural Systems with Processing.* Magic Book, 13 December 2012.

Sokol, Zach. *Synesthetic Sensory Stimulation With Ryoichi Kurokawa*. http://thecreatorsproject.vice.com/blog/video-sensory-overload-with. Published 26 November 2013. Accessed 15 December 2013.

Tarakajian, Sam. *Mira touch controller iPad app*. 2013. http://cycling74.com/products/mira/.

Researched Works

Akten, Memo. *Amoeba Dance.* http://www.memo.tv/amoeba-dance/. Accessed 10 November 2013.

Burton, John. *Boxing Day Java Script in Max/MSP.* http://leafcutterjohn.com/?p=1391. Published 27 December 2010. Accessed 19 November 2013.

Burton, John. *The SOUND of MONEY Algorithmic Max/MSP experiment based on the Price Of Gold 1957-2013.* http://leafcutterjohn.com/?p=1865. Published 19 November 2013. Accessed 19 November 2013.

Ikeda, Ryoji. *test pattern [n°5].* https://vimeo.com/68597939. Published July 2013. Accessed 21 November 2013.

Kurokawa, Ryoichi. *Oscillating Continuum.* http://www.ryoichikurokawa.com/project/oc.html Published 2013. Accessed 15 December 2013.

Rosati, Franz. *Streakline #1.* http://www.franzrosati.com/streakline-1/. Published 2013. Accessed 10 November 2013.

Tsutsui, Masato. *Bridge.* https://vimeo.com/7908215. Published 2010. Accessed 10 November 2013.

Tsutsui, Masato. *mtb.* http://adsr.jp/download/patch/. Accessed 10 November 2013.

APPENDICES

Progress Diary

Throughout the duration of my practical work in Processing, I kept an informal progress diary. Here I have selected the relevant entries that show the maturation of my skills and ideas implemented in the code submitted on CD. In my practical work I learnt about functional and conceptual aspects of coding, which lead me to research generative art and audiovisual systems.

The intent behind this diary is to give an indication of my developing thought processes that resulted from both the practical coding and my research, which mostly takes the form of academic articles and exploration of relevant artists or works. Furthermore, by presenting this information in a separate document I am able to free up the main written component of this research project to avoid it becoming too convoluted or technical; when needed, I have referred to this diary in the Research Project Documentation.

1. Monday 17th February 2014

1.1 Emergence

- Introduced to concept of emergence
- Emergence: "This phenomenon, whereby a simple rule set at a low level creates organised complexity on a higher level, is called emergence."[53]
- "When you create a large number of small agents with simple behaviours, collectively they can become capable of producing a more complex, emergent, behaviour on the macro level." Interesting! Possible way to go for final composition.
- "In recent decades, the concept of emergence has found a new relevance and wider popularity through its application to Complexity theory, first by John Holland in his book Emergence

[53] Matt Pearson, *Generative Art: A Practical Guide Using Processing* (New York: Manning Publications Co., 2011), 108.

(1989), and later for a more popular audience by Steven Johnson in a 2001 book of the same name."

- "[Flocking of birds] has been modelled in computing, most famously by Craig Reynolds in his Boids algorithm, first published in 1987. Reynolds discovered that to produce a realistic flocking simulation in code, he needed only three rules:
    - Separation—Steer to avoid your immediate neighbours.
    - Alignment—Steer to align with the average heading of your immediate neighbours.
    - Cohesion—Steer toward the average position of your immediate neighbours.
- More complex rules can be added (object avoidance, for example), but these three are all that are required to achieve a complexity comparable with the West Pier starlings."[54]
- See also http://processing.org/learning/topics/flocking.html

1.2 Object-Oriented Programming

- To explore this particular concept I needed to look into object-oriented programming
- "It's difficult to get far with flocking algorithms, and other forms of emergent code, without a **basic understanding of object-oriented approaches to programming**."[55]

1.3 Simple Code, Complex Results

- "Second, within a methodology whereby you work with the simple logic of the programming language, where you encourage artworks to grow from logical seeds, and you wish to evolve a complexity sufficient to be visually interesting, the principles of emergence seem to be an ideal tool for **getting complex results from simple code**."[56]

---

[54] Ibid., 109.

[55] Ibid., 110.

[56] Ibid., 111.

- "We haven't relied on randomness or noise to introduce the generative element to the code in this chapter. All we've done is apply a level of abstraction sufficient to introduce an unpredictable complexity. Simple rules creating complex results."[57]
- See what happens when you add noise/randomness

---

2. Wednesday 26th February 2014

2.1 Autonomous Agents

- So far quite interested in emergence, autonomy. Systems that have a simple set of rules, from which complex behaviour emerges.
- "The difference between an object and an agent, in programming parlance, is that agents, specifically, observe and interact with their environment. They may have more sophisticated behaviours, such as goals, beliefs, and prejudices…This is what we mean when we speak of autonomy: the capability for something, whether human, monkey, software construct, robot, or supermodel, to make its own decisions."[58]
- Game of Life - "A simple black or white dot is only a starting point. Just as you did with multidimensional Perlin noise back in chapter 5, where you used the noise value as rotation, shape, size, and alpha values, you can try similar rendering experiments using a cell's properties. In the two examples in figures 7.3 and 7.4, I've used the number of live neighbours a cell has, and the number of frames it has remained alive, to determine the size, rotation, and colour of a shape drawn at that cell's location. You may like to try similar experiments."[59]

---

[57] Ibid., 126.

[58] Ibid., 127.

[59] Ibid., 137.

- Game of Life is basically a rule set, and one of many. It can be used in different ways as a framework to be built upon.

2.2 Concurrent Audio and Video Synthesis

- I have reached a problem: So far the work I have been following has only been around making a generative visual piece. Is it possible to have art which is generative AND audiovisual? Should I specialise into one?
- Yes it is! **'Dynamic Independent Mapping Layers for Concurrent Control of Audio and Video Synthesis' - Ali Momeni and Cyrille Henry**: utilising generative systems as dynamic mapping layers that control both audio and visual elements.[60]
- Two key examples: Mass-spring physics models and interpolations systems in perceptual spaces
- "Most works can be characterised as (1) sound-to-image, (2) image-to-sound, or (3) concurrent generation of sound and image".[61] This concurrent generation of sound and image is what I am interested in (and indeed explored (1) and (2) in the Sound for Image and Performance with New Technology second-year modules).
- This approach to mapping "involves an independent layer of algorithms with time-varying behaviour that is affected, explored, or observed in some way with gesture."

3. Saturday 1st March 2014

3.1 Concept for Piece - Physical Models

- Physical mass-spring model used to control grains of granular synth.

[60] Ali Momeni and Cyrille Henry, 'Dynamic Independent Mapping Layers for Concurrent Control of Audio and Video Synthesis,' in *Computer Music Journal, Vol. 30, No. 1* (2006), 49-66.

[61] Ibid., 49.

- Interpolation space used to control parameters of the model
- Way of controlling masses undecided. Interpolation space between states? Directly control one mass?
- Try looking at the externals mentioned in Momeni/Henry article and trying to recreate them in processing/use them for inspiration

3.2 Artist Example — egrégore by chdh

- [From chdh website] “COLLECTIVE BEHAVIOURS: Group behaviour is an interesting field of research due to the great complexity of evolution and forms generated by the interaction of subjects in a crowd. A crowd of identical elements can create surprising shapes and movements. This is the case in some schools of fish or flock of birds: each element adapts its speed and position based on those of its neighbours. From these elementary movements rises structures organised in shapes whose diversity is a reflect of the behaviour they are based on. A global shape is created by the sum of individual wills: noisy lines or clouds turn into fractals from chaotic dynamics. All these elements can have a common purpose, be opposed in different subgroups or evolved independently”
- “In egregore, there is only one composed form, a macro structure born thanks to a crowd of micro-elements creating a visual and sound space”
- “In one continuous gesture, the elements evolves from a chaotic movement to a consistent organism.”
- Possible method of control:

  “GENERAL USAGE

The egregore instrument is divided in five different behaviors, click on the on/off button to activate them separately and move the faders to change the state."[62]

---

4. Monday 3rd March 2014

4.1 Particle Systems

- Decided to stop working on the tutorials in *Generative Art* (Matt Pearson) and look further into particle models. In order to gain a deeper understanding than is provided by Pearson, I am looking at *The Nature of Code* by Daniel Shiffman.
- "The goal of this book is simple. We want to take a look at something that naturally occurs in our physical world, then determine how we can write code to simulate that occurrence."

4.2 Classes in Processing

- Working with classes: "All classes must include four elements: name, data, constructor, and methods. (Technically, the only actual required element is the class name, but the point of doing object-oriented programming is to include all of these.)" [63]
- "Class names are traditionally capitalised (to distinguish them from variable names, which traditionally are lowercase)"
- Using objects: Step 1 - Declare, Step 2 - Initialise, Step 3 - Functions
- "Calling a function inside of an object is accomplished via dot syntax: variableName.objectFunction(Function Arguments)"

---

[62] chdh, 'README,' in egregore source, http://sourceforge.net/projects/chdh-egregore/files/, accessed 1 March 2014.

[63] Daniel Shiffman, *Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction* (Morgan Kaufmann, August 2008).

4.3 Concept for Piece: Physical Models

- Could use boids as the physical model, using the xy position of each boid to control grains
- Example: http://forum.processing.org/two/discussion/2964/how-to-get-an-objects-xy-location-on-screen/p1
- See boids max/msp patch

---

5. Wednesday 5th March 2014

5.1 PROBLEM

- After starting work on following *The Nature of Code,*[64] it seems perhaps too optimistic to build my own particle/boids/mass-spring system in processing within the time left, especially given that they can be quite complex.
- Furthermore I am finding several objects built in Max for boids/mass-spring models (such as those in the Momeni & Henry paper) which I could utilise instead.
- Or another option could be to use one of the tutorials from *Generative Art* as a starting point
- This would allow me to explore the ways in which such systems could be implemented artistically

5.2 Concept for Piece: Methods of Controlling Audio & Video Synthesis

- I could have an interpolation space defining how boids/particles/objects react - determining distances between each, their speeds etc.

[64] Daniel Shiffman, *The Nature of Code: Simulating Natural Systems with Processing* (Magic Book, 13 December 2012).

- I could then use a different method of control to move the attraction point for boids around - e.g. an x-y control, where each boid = a grain/set of grains in a granular synth.
- I could then take the x-y position, speed/acceleration, direction of movement, etc. for each boid (look at the Momeni & Henry paper for examples) and use this information to control the audio & video.
- Or if using a spring-based model, use the "resistance, length, and damping constants (one for each end of a spring)."[65]

---

6. Thursday 6th March 2014

6.1 Communication between Max/MSP and Processing

- Decided to start looking at how to exchange information between Max/MSP and Processing
- Looked at *oscP5* library[66] for Processing and followed the *Sending and Receiving OSC Data Using Processing* tutorial on the CodaSign website,[67] but couldn't figure out how to send OSC messages to appear in Max — only in Processing
  - UPDATE: I have managed to adjust my Processing sketch so that `bang` messages can be sent to Max over OSC
- I managed to achieve a basic level of communication, sending messages between Processing and Max by using the *Maxlink* externals[68]

---

[65] Momeni and Henry, *Dynamic Independent Mapping Layers,* 55.

[66] Andreas Schlegel, *oscP5 0.9.8.* (19 December 2011) http://www.sojamo.de/libraries/oscP5/, accessed 6 March 2014.

[67] Unknown, *Sending and Receiving OSC Data Using Processing* (March 2012), http://learning.codasign.com/index.php?title=Sending_and_Receiving_OSC_Data_Using_Processing, accessed 6 March 2014.

[68] Jesse Kriss, *MaxLink version 0.36,* (4 March 2010), http://jklabs.net/maxlink/index.html%3Fpage=downloads.html, accessed 6 March 2014.

6.2 Concept for Piece: Observing Behaviour

- I talked to Sebastian about the idea of using boids/swarming behaviours to control audio/video parameters. He warned me that in his experience, it is very easy to become predictable when using these techniques, if the boids follow you around like a dog.
- Therefore, why not use something like Computer Vision to analyse the boids flying around and track the motion, density etc. of groups that emerge — looking at and using *emergent* behaviours rather than individual behaviours of single boids (e.g. one boid = one grain in a granular synth).
- This way the relationship is not so simple or gimmicky, instead using the more complex behaviours of the system to influence the audio/video.

---

7. Tuesday 18th March 2014

7.1 Concept for Piece: Circles

- I could utilise Matt Pearson's tutorial on emergence that uses circles as a dynamic independent mapping layer
- The emerging circles could be used to control audio and video synthesis, as well as the original circles if applicable

7.2 Communication between Max/MSP and Processing

- The method of communication I had established using the Maxlink externals was not working correctly

- As such I continued researching other ways of communicating between Max and Processing, using OpenSoundControl. I decided to take another look at the *oscP5* library in Processing, and came across the *MMO Visual Hacker Toolkit*[69]. This pointed me towards a Max object called `OpenSoundControl`[70] that works in conjunction with the code provided in the *Toolkit*.
- Using the *MMO Visual Hacker Toolkit* code I have managed to get Max/MSP and Processing to communicate in such a way as to control parameters in the Processing sketch by using a Max patch. The Max patch I am using currently is very simple: it is an x-y controller which currently just controls two of the three values I have defined (these being average speed, average size, and number of circles).
- Eventually the x-y controller will control my position in an interpolation space (the interpolation space in this case being a nodes Max/MSP object), with each node in the interpolation space containing presets for each of the three parameters. In this way, the control system will work much like the system used in my final Performance with New Technology project; but there, instead of using an x-y controller, I used two softpot sensors (one controlling the x value, and one the y value).[71]
- In the Max patch, both the x-y control and nodes objects are within a Mira frame. This means they can be seen/controlled using the Mira iPad app

[69] Berkeley Multi-Media Orchestra, *MMO Visual Hacker Toolkit* (Berkeley, California: 2013), https://github.com/derekrazo/mmo-visual-hacker-toolkit, accessed 18 March 2014.

[70] Matt Wright & Andy Schmeder, *OpenSoundControl,* (Berkeley, California: CNMAT, 1996) http://cnmat.berkeley.edu/downloads, accessed 18 March 2014.

[71] More information about my implementation of interpolation spaces in my Performance with New Technology project can be found in the post titled 'Performance with New Technology: Developing Final Performance' on my website: www.whitenoises.co.uk.

8. Monday 24th March 2014

8.1 Developing Sketch

- Circles sketch is partly working with Max control (radius works, `numCircles` and speed partly work) but:
  - Need to have number of circles update when value is changed, not when mouse released
    - UPDATE: Tried achieving this by moving the `drawCircles()` function from `void mouseReleased()` to `void draw()`. However, this means that new circles are added every time the program loops and the system quickly becomes unstable. This area needs more development.
  - Need to find out how to delete circles — `numCircles` decides how many circles are added, not how many are on screen (I have temporarily renamed it `numAddCircles` to clarify).
    - UPDATE: Realised that this might be possible by changing the array size, therefore changing how many circles can be constructed in the system's memory.
    - UPDATE: `Array` size cannot be changed, but found out about `ArrayList` — the size of which can be altered.
    - Have switched out the `Array` for an `ArrayList`. It partially works, but have had trouble getting the `drawCircles()` function to work as it requires some adjustment.
    - Once this has been sorted I will be able to see the circles more clearly and work on how to delete them when the `numCircles` parameter is reduced.
  - Need to have speed affect all circles, not new ones
    - I think to achieve this I must change the `averageXSpeed` and `averageYSpeed` to affect the `ArrayList` rather than the `Circle` class; at the moment they are affecting the class, which is called when constructing *new* circles, as opposed to the `ArrayList` which contains all circles.

9. Wednesday 2nd April 2014

9.1 Control

- Make sure that as soon as someone uses the x-y controls, they can perceive a difference: that their control has an effect on the system

9.2 Circle relationships

- Are the circles that appear through collision really 'emergent *behaviour*'? Or it more that you are manipulating a model, which results in certain behaviours emerging?
- Develop your thoughts on the ways the circles interact, how these interactions can be farmed for data, and how this data can be applied artistically.
  - Data available from original and emergent circles:
    - Speed
    - Direction
    - Size
    - X-Y position
    - Density
    - Rate of change in size (applies to emergent circles only)
  - Three model ideas:
    1. 'Continuous sound' model: Sound is continuously generated by the original circles. Emergent circles effect this sound (could be just one, or multiple effects) depending on above data.

2. ‘Collision = sound’ model: Sound is generated when emergent circles are created. Type of sound depends on above data.
    - This can be achieved by sending a `bang` message from Processing to Max when a collision occurs
    - My sketch sends a bang message when a collision occurs — however, it sends them constantly while circles are overlapping instead of just once, because the function to send it is in an `if` loop.
    - Pitch could be controlled by x-y position of collision? At the moment it is just random.
    - Length of note inversely proportional to amount of circles to avoid long silences (when few circles) and too much going on (when many circles)
3. ‘Sound byte/effect’ model: Original circles are sound bytes (grains/parts of samples etc.) When they collide an effect is applied (controlled by data from the emergent circle). Or, original circles are split into two types: bytes and effects. Bytes are affected only when they collide with an effect (again, controlled by data from emergent circle).
    - Think how these could be combined.
    - Is the circles sketch the visual component? Or is it the dynamic control layer which affects a separate visual source?

- Create a presentation showing these concepts, on how the data can be harvested and used artistically
- What these models have in common is how the changing behaviour of a body can be used to control audio/visual parameters: i.e. they are dynamic mapping layers[72].

72

9.3 Extracting x-y positions

- Have looked into methods of extracting the x-y positions of the circles. I have found some help in an old thread on the Processing forums: http://forum.processing.org/two/discussion/2964/how-to-get-an-objects-xy-location-on-screen/p1
- I think I need to look further into `ArrayLists` and how to retrieve information from them on demand
- Alternatively I could try taking the x and y float values defined in the Circles class.

Diary Bibliography

Berkeley Multi-Media Orchestra. *MMO Visual Hacker Toolkit.* Berkeley, California: 2013. https://github.com/derekrazo/mmo-visual-hacker-toolkit. Accessed 18 March 2014.

chdh. 'README.' In egregore source. http://sourceforge.net/projects/chdh-egregore/files/. Accessed 1 March 2014.

Kriss, Jesse. *MaxLink version 0.36.* 4 March 2010. http://jklabs.net/maxlink/index.htm%3Fpage=downloads.html. Accessed 6 March 2014.

Momeni, A. and Henry, C. 'Dynamic Independent Mapping Layers for Concurrent Control of Audio and Video Synthesis'. In *Computer Music Journal, Vol. 30, No. 1.* 2006. pp. 49-66.

Pearson, Matt. *Generative Art: A Practical Guide Using Processing.* New York: Manning Publications Co. 2011.

Schlegel, Andreas. *oscP5 0.9.8.* 19 December 2011. http://www.sojamo.de/libraries/oscP5/. Accessed 6 March 2014.

Shiffman, Daniel. *The Nature of Code: Simulating Natural Systems with Processing.* Magic Book, 13 December 2012.

Shiffman, Daniel. *Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction.* Morgan Kaufmann, August 2008.

Unknown. *Sending and Receiving OSC Data Using Processing.* March 2012. http://learning.codasign.com/index.php?title=Sending_and_Receiving_OSC_Data_Using_Processing. Accessed 6 March 2014.

Wright, Matt & Schmeder, Andy. *OpenSoundControl.* Berkeley, California: CNMAT, 1996. http://cnmat.berkeley.edu/downloads. Accessed 18 March 2014.