VIDGETS: The Development and Use of Interactive, Network Based Video Works

David Wolf
VIDGETS: The Development and Use of Interactive, Network Based Video Works

An exegesis submitted in fulfillment of the requirements for the degree of Master of Arts

David Wolf
BA (Honours)

SCHOOL OF APPLIED COMMUNICATION
PORTFOLIO OF DESIGN AND THE SOCIAL CONTEXT
RMIT UNIVERSITY

August 2007
DECLARATION

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the exegesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged.

David Wolf
This exegesis looks at interactive, network based video through the development and use of experimental works. The works are described in terms of the concepts, techniques and theories explored and their implications. It documents the formal and technical experiments which were used to investigate the affordances of network based video for real time interaction, and tracks the development of vidgets as audio-visual tools and instruments as they were increasingly designed for use in live VJ performances. I explain how used sound art and systems based art as frameworks for conceptualising interactive screen based works as an alternative to cinema, and devise the term ‘all-data’ as a means to describe this environment. Reflecting on the processes of action research used in the project, I identify and explore a number of techniques and attributes common to interactive, network based video production as a practice which may be used in a range of future projects.
ACKNOWLEDGEMENTS

Thanks to everyone who has assisted me in this project and in preparing this exegesis, in particular:

My supervisor Adrian Miles for his encouragement, patience and insight.

Sean Healy and the Plug n Play crew for giving me a place to play as often as I wanted to and for inspiring me with their pixels and bleeps.

Adrian Lucas for his friendship, frequent improvisational collaborations and noise hands.

Anna Wolf for the study sessions, proofreading and commas.

John Wolf for proofreading and support.

Timothy Webster for his friendship, many fruitful collaborations and chats.

Nicole Dominic for the excellent design under pressure.
# TABLE OF CONTENTS

## INTRODUCTION

<table>
<thead>
<tr>
<th>FINDING A FRAMEWORK: THE SHIFT FROM CINEMA TO SOUND ART</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive QuickTime and LiveStage Pro</td>
<td>10</td>
</tr>
<tr>
<td><em>Brick Style</em></td>
<td>13</td>
</tr>
<tr>
<td><em>REMOTE CONTROL 0.2</em></td>
<td>15</td>
</tr>
</tbody>
</table>

## THE SHIFT FROM STATIC TO DYNAMIC WORKS, DISCRETE OBJECTS TO DISTRIBUTED SYSTEMS

<table>
<thead>
<tr>
<th>XML</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Random Site</em></td>
<td>23</td>
</tr>
<tr>
<td>Tagging</td>
<td>24</td>
</tr>
<tr>
<td><em>QuickTime Flickr Photo Viewer</em></td>
<td>25</td>
</tr>
<tr>
<td>RSS and Atom Feeds</td>
<td>26</td>
</tr>
<tr>
<td><em>Vidget 3.5</em></td>
<td>29</td>
</tr>
</tbody>
</table>

## REAL TIME SYSTEMS, REAL TIME PROGRAMMING

<table>
<thead>
<tr>
<th>Quartz Kaleidoscope</th>
<th>43</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Bikelights: a Quartz Vidget</em></td>
<td>45</td>
</tr>
<tr>
<td><em>White Noise</em></td>
<td>51</td>
</tr>
</tbody>
</table>

## WHAT I HAVE LEARNT

| WHAT I HAVE LEARNT                                        | 55 |
INTRODUCTION
Background To My Practice: 
Film, Video, Sound and DVD Production

I began this research project with a background in film and video production, editing, sound design, and DVD authoring. In each of these roles I was using various pieces of software to generate and manipulate audio-visual content. Many of these programs were designed to replicate traditional analogue production tools such as film editing desks, audio mixers and tape recorders. This was reflected in both the user interface design of the software and in their bias towards the generation of fixed, linear, audio-visual sequences. Apple’s Final Cut Pro, for example, uses several metaphors in its user interface which refer back to traditional analogue filmmaking. A razor blade tool is used to trim shots as if they are strips of film. Audio faders are used on screen to set the levels of various audio tracks as they are mixed together. While such programs provided me with a number of features unique to the digital domain such as the capacity for non-destructive editing and multiple levels of ‘undo’, once a ‘final cut’ was completed the software generally output fixed, linear works intended for traditional broadcast or screening. To use Diane Balestri (1988) and Adrian Miles’ (2003) terminology, this was still a ‘hard copy’ approach. As I worked with these tools, I became fascinated by this process of combining and manipulating moving images and sounds on the computer. I began to think about the potential for audio-visual works that would allow the viewer to manipulate and sequence images and sounds, taking advantage of the interactivity and non-linearity afforded by digital technologies.

With my Honours project Twelve (Wolf, 2001) I began to explore the idea of producing a piece that would require the viewer’s interaction to ‘finish’ the work. I applied concepts such as Aarseth’s (1997) ‘cybertext’, and ‘ergodic literature’ to a narrative cinema work by creating a non-linear, interactive narrative short film on DVD. Using DVD motion menus, I created a branching structure of hyperlinks between scenes, requiring the user to choose their own path in order to progress through the narrative in a similar manner to Michael Joyce’s frequently-analysed literary hypertext Afternoon, a Story (1990). By creating a work which required user interaction I was trying in some ways to replicate my own experiences as a media producer. I wanted the viewer/user to feel as though they were active participants in the sequencing and construction of the narrative. While the DVD worked well as a collection of narrative fragments, on reflection, the interactive structure didn’t provide the viewer with the sense of agency and level of immersion for which I had initially aimed. This was partly a result of the limited number of scenes to be explored, partly a product of the inherent differences between text based narrative works and the televisual/cinematic, and partly a result of the technical limitations of the DVD format.
I decided that with this Masters project, rather than continuing to work with the limited interactivity afforded by the DVD Video specification, and the formal constraints of narrative cinema, I wanted to produce works which would run on computers and take advantage of their capacities for real time interactivity and network awareness. I wanted to explore the use of the networked computer as not only a production tool, replicating the paradigms of traditional analogue media, but as a new, interactive environment. In short, I wanted to explore what Adrian Miles refers to as ‘softvideography’ (2003).

**VJing and Sound Art Performance**

As I commenced this research project I began to organise and perform at live events featuring improvised noise music (sound art) and video art, (see Appendix 2 for a list of performances). These experiences influenced my project greatly in terms of both forming the research problems and shaping the ways in which solutions to these problems were investigated as the project progressed. Live video performance (VJing) and the development of vidgets (small scale video performance tools) soon became my primary arts practices and the basis for my research.

I found that as I experimented with the production of audio and video in a live performance context my fascination with processes of “remediation” increased. Bolter and Grusin define remediation as “the representation of one medium in another” and argue that it is “a defining characteristic of new digital media” (2000: 45). In both audio and video performance I was taking pre-existing clips, mixing and processing them with effects in real time. In many ways this process reflected what Bolter and Grusin refer to as the “twin logics of immediacy and hypermediacy” (2000: 5). They argue that our culture has contradictory desires for both immediacy (a process whereby mediation is rendered invisible, giving the audience a direct relationship to the object) and hypermediacy (where the processes of mediation are highlighted and multiplied). They note that “ideally, it wants to erase its media in the very act of multiplying them” (2000: 5). With my VJ performances in particular I was dealing with both the immediacy of manipulating images in real time in response to the music, and the distortion of these images via extremely hypermediated visual effects.

In preparation for both sound and video performances I would collect libraries of source material for use on the computer. For audio this involved collecting and producing sound files such as drum loops and field recordings. For video, this involved filming, editing and preparing short clips. In both cases these elements were combined and manipulated in an improvised performance context in front of an audience. One significant difference between my sound and video performances was the way in which the computer was used in the process. With audio performance I was able to use ‘off the shelf’ software such as Ableton’s Live to create custom arrangements of sounds and chains of audio effects. These could be varied, processed, ‘tweaked’ and interacted with via keyboard, mouse and MIDI controllers, manipulating live audio input and output in real time. For my VJ performances I was using two computers
running Apple’s Final Cut Pro software – a package designed for producing fixed, linear media. These two computer video sources were used in combination with a hardware video mixer to select, combine and effect the two sets of images in order to provide visual accompaniment to audio performance. The computer was being used as little more than a tape deck, not benefiting from its capabilities for real time interactivity or the fact that it could be connected to a network of other digital devices, either locally or through the internet.

In this project I was not aiming to create an application which would be a video equivalent of a complex and multipurpose audio program like Ableton Live. I did, however, want to create works which would let the user manipulate and combine images in real time on a regular computer without the need for additional hardware.

With my experiences in video performance in mind, I also wanted to explore the use of the network as a way of taking the library of audio-visual content I could access and manipulate and expanding it to include anything that was hosted online. For this project I wanted to create interactive works that could be used either in live improvised performance, or for private, self directed, recreational experimentation.

The problems to be explored

• How can new audio-visual works (vidgets) be created to explore and utilise the capacity for real time interactivity and network awareness found in everyday computer technology?
• What can we learn by looking at the use of real time interactivity and network/environmental awareness in the histories of media and art practice and how can these this inform the development of new works?
• What happens when video media art works shift from being fixed and self contained ‘objects’ to dynamic, distributed systems inviting user interaction and performance?

This project is centred on the development and use of ‘vidgets’: real time interactive, network aware audio-visual media art works designed specifically for use on computers. With recent advances in computer software and hardware development, works may be created which allow users to generate, source, combine and manipulate data which represent a range of audio-visual material. Works may allow the user to see and hear the results of their choices and interactions with this data as they are made in real time, without the need for specialised equipment, using regular desktop and laptop computers. With local and remote networking technologies and access to broadband internet connections, network aware works may be created which source a range of data from anywhere on the network. This data may represent images, sounds, text or other raw information which may be analysed and used to control the access, generation and manipulation of other audio-visual materials. Again, audio-visual media that takes advantage of this network connectivity may provide the user with an almost limitless range of source material to be interacted with, listened to and watched.
Practice Led Theory: 
Action Research and Research Through Design

In this project I have used a cycle of action based research, based on Donald Schön’s ideas of the reflective practitioner (1983) and Peter Downton’s (2003) thoughts on research through design. In the following chapters I will discuss works produced through the following cycle of steps:

- **Planning:** establishing the specific aims of each work as well as the themes and practices I am exploring;
- **Skills based research** (research for design): working out how can I achieve these goals using the technology at hand;
- **Production** (research through design): applying these skills, testing and exploring ideas through design and production;
- **Use and performance:** playing with the works, often in a performance context to see how they work, and / or reflect the initial aims;
- **Critical reflection:** Looking back not only at how the work reflected the initial aims, but also:
  - how the work fits in the greater scheme of media arts practice
  - how the processes at work are similar to or different from those of existing or historical works and how the study of these works may inform the project
  - the implications of the works, practice and theory future works and interactive, network based video as a practice

Rather than building towards one or two major works, with this project I chose to produce several smaller scale works and numerous performances. This research strategy complimented the granular, distributed nature of network based media and allowed me to investigate a wide range of techniques, technologies and ideas. It also allowed me to reflect on my productions and performances frequently as they were completed. This reflection assisted me in shaping and theoretically situating my creative practice as it developed.
Looking At Media Art Practices and Theory In Terms of Production Processes

A common strategy used by various new media theorists is to compare new and emerging media forms with their more established precursors and predecessors. Digital still images are often compared to photographs and paintings, hypertexts to printed literature, and various screen based media to cinema and television. Bolter and Grusin (2000: 11), for example write: “Remediation did not begin with the introduction of digital media. We can identify the same process throughout the last several hundred years of Western visual representation.” Similarly Lev Manovich’s The Language of New Media seeks to “analyze the language of new media by placing it within the history of modern visual and media cultures” (2001: 8).

This approach can be useful, however, many such comparisons may be of limited use in the context of this project in that they tend to focus primarily on the aesthetic and formal similarities between ‘old’ and ‘new’ media as finished works. In the context of this project I was interested in looking at similarities between old and new production processes and methods rather than the actual objects, programs and events themselves. As Smith and Dean note, while “critical theory has vastly opened up the analysis and understanding of reception of works of art, analysis of the creative process has tended to be undervalued in criticism and theory.” (1997: 29).

While a particular ‘new media’ form may look or sound similar to other traditional media works, or even use the same codes and conventions of audio visual communication, the processes used in their physical or virtual construction may be quite dissimilar. For example, images from a live video (VJ) performance may look like those of an experimental film. However, while they may look similar, the VJ is most likely selecting and manipulating the images as they are playing whereas the filmmaker most likely selected and assembled the shots one after another in preparation for screening days, weeks, months or years before the film is projected. Where a film may be reproduced, distributed and screened in multiple locations at different times whilst providing the audience with what is essentially the same sequence of images and sounds, no two VJ performances are ever exactly the same. A VJ improvises as he or she performs, interacting with audio-visual hardware and/or software in real time to produce a stream of images, usually accompanying a DJ or live musician’s sounds.
In this study I have chosen to look at the histories, theory and practice of a number of media and art practices in terms of their production processes. Over the course of this project I have found it useful to look at my own practice in terms of what can be learned from the histories of sound art and noise music (Russolo, 1986; Kahn, 1999) and the conceptual ‘systems art’ of the 1960s (Burnham, 1974). In particular I have looked at the ways in which these art movements have made use of real time systems, elements drawn from everyday environments and networks.

In this exegesis I will discuss the production of several key works (vidgets) in relation to their exploration of real time interactivity and network awareness. I have divided what follows into three sections to focus on the three key themes which emerged through the research and which marked significant changes in my practice:

- The shift from cinema to sound art as a framework for looking at interactive screen based video work
- The shift from static to dynamic works – from self contained objects to distributed systems
- Real time programming as an improvisational practice

These changes included both methodological and theoretical shifts.
FINDING A FRAMEWORK:
THE SHIFT FROM CINEMA TO SOUND ART
Interactive QuickTime and LiveStage Pro

One of the initial aims of this project was to create works which would be easily accessible. This meant they should be designed for distribution via the internet, with small file sizes, and preferably the ability to run on both Mac OS and Windows computers. Rather than using specialised software which users would have to install and configure, I preferred that they could simply click on a link to run the works instantly. This led me to explore the use of Apple’s QuickTime file format as an architecture for interactive video.

QuickTime is bundled free with most computers, is cross platform and designed to work with networked data and user interaction. Before beginning this research, I had been using QuickTime formatted video and audio files in the creation of film, video and DVD projects. In this context, the files were treated simply as ‘assets’. A clip of sound or video could be imported into an editing program such as Final Cut Pro, combined, cut and manipulated until the piece was finished. The files were used like rolls of film and magnetic sound tape to be edited on a traditional Steenbeck editing machine. Working in this manner required the use of an external program to edit and manipulate the images and sounds.

With the development of vidgets, I wanted to integrate capacities for user interaction, manipulation of images and network awareness within the works themselves. Using Totally Hip’s LiveStage Pro as an authoring environment, I was able to create movies which could utilise QuickTime’s capacity to deal with user interaction and a range of data communications. This allowed me to make QuickTime ‘movies’ which function as simple, real time interactive programs.

While it is widely known as a container format for video and audio data, QuickTime was also designed as an extensible architecture for dealing with a range of interactive and time based media. As well as dealing with audio and visual tracks, the format includes a range of track types for dealing with text, still images, chapter information, MIDI music, and VR panoramas and objects. ‘Sprite tracks’ may contain programmatic scripts which deal with user interaction, the dynamic manipulation of audio-visual elements and the parsing of data from remote networked sources.

With Brick Style (March 2004) and Remote Control 0.2 (May 2004) I experimented with the creation of interactive QuickTime movies which could tap into the ‘digital background noises’ of the networked environment in order to use them to create real time interactive and network aware works. This was an area which lay outside of my previous experience as a film and video maker.
The two works I will discuss in this section, Brick Style (March 2004), and Remote Control 0.2 (May 2004) are pieces from a series of small scale proof of concept and exploratory works which explore the use of everyday computer software, in particular Apple’s QuickTime architecture, as an environment for real time interactive and network aware works. Through the development of these works I investigated the use of a range of real time and networked data and communications which are not traditionally associated with ‘video’ or ‘cinema’ but which are commonplace in everyday computer use. I specifically set out to explore the use of these interactive and network aware capabilities through the development of vidgets.

The ‘all-data’

In the environment of the networked computer all media signals, objects and applications are made up of data which may be accessed, copied, manipulated, synthesised and reproduced. There is effectively no difference between data which represents text, sound, still and moving images, keystrokes and mouse gestures, network information and even program code until it is rendered visible, audible or operational. I began to shift from working only with moving images and sounds to dealing with several other forms of data from this broad range. In order to inform and contextualise this significant change in practice I found it useful to look at the ways in which noise and sound artists have historically dealt with the concept of the ‘all sound’. Put simply, the ‘all sound’ is the idea that any sound, including noise or silence may be considered musically and incorporated into a musical work. The concept was initiated by sound artists such as Luigi Russolo and Pierre Schaeffer and taken to its logical extremes by John Cage (Kahn, 1999). As a reference to the ‘all sound’, I refer to both the networked digital environment and the palette of signals and objects accessible within it as the ‘all-data’. Sound artists had expanded the previously limited range of sounds available for use in musical practice to include the audible noises of the everyday environment. Similarly, I was interested in expanding the range of data forms used in video from the purely audio-visual to include a wider range of everyday data (the all-data) through the development of vidgets.

In his 1913 manifesto The Art of Noises Luigi Russolo wrote that “every manifestation of our life is accompanied by noise. The noise, therefore, is familiar to our ear, and has the power to conjure up life itself” (Russolo, 1986: p.27). Following Russolo, it could be argued that most aspects of daily life are now accompanied by some form of digital information transfer. Russolo’s noises were familiar by-products of the audibly noisy industrial age. Our ‘digital noises’ are now processes, transactions and trails in the all-data which operate in the background, facilitating the information age of today. Every time we move a mouse, type on a keyboard, visit a website or listen to a piece of music on the computer, data is exchanged between networked devices. As we increasingly use computers and other digital devices in our daily life our actions are manifested as the generation, access and manipulation of data. Many of these processes are silent and invisible to us, operating in the background as we become
accustomed to the everyday tasks they facilitate and ignoring them when they are not in use. When we download a movie trailer from a website we may be aware of navigating to a link, clicking on it and waiting for an exchange of data, however once the trailer is playing we are likely to stop physically interacting with the computer as we sit back and watch. If we begin to use computers and other digital devices not just as production tools or display media which replicate cinema or television, but as means to engage with this all-data we may expand video practice to include new interactive, networked forms.

Through the creation of vidgets, I am exploring the use of data associated with real time interaction and networked communications. These works explore the handling of data generated by user input such as the position of the cursor and mouse clicks, as well as inter-movie communication, the use of XML data from remote networked sources and programmatic animation. As I produced the following works, I learnt about what was technically possible in terms of real time interactivity and network awareness within the QuickTime architecture. This shaped the direction of subsequent works and gave me an idea as to the practical and conceptual changes which occur when dealing with the production of computer based interactive works. By identifying similarities between my own aims and methods and those from the histories and practice of noise and sound art, I was able to conceptualise a framework for my research and inform the development of new works.
**Brick Style**

*Brick Style* is designed as a very basic *Breakout* or *Pong* style paddle game. It features three on-screen elements, a square red ball, a grey rectangular paddle which follows the horizontal position of the user’s cursor, and a dark background. As the movie loads the red ball begins falling down from the top of the screen at a diagonal angle. The user must position the paddle below the ball using the mouse or trackpad to prevent it from hitting the bottom of the screen. If the user positions the paddle under the ball in time, the ball bounces back up, bouncing off the sides of the window and back down. The movie continues until the ball is missed or the user closes the window and quits.

The work was designed as a way of testing QuickTime’s capacity for dealing with real time user input. The stream of x and y co-ordinate data generated by the movement of the user’s cursor (which is usually ignored by videos playing on the computer) is interpreted by a script contained within the movie and used to control the movement of the paddle image. Coming from a background using the (extremely limited) DVD-Video format as a medium for interactive video, I was particularly interested in testing how well QuickTime dealt with real time user input. Rather than having the user of the work select from a series of menus of pre-prepared material to watch (as I had done with my Honours DVD *Twelve*), I wanted to create a work which required constant user input to proceed.
When viewed in the context of gaming, *Brick Style* is not new, but in terms of video on the computer it explores two main techniques from outside the realm of traditional film making: dealing with a real time stream of user input data, and the programmatic control and animation of objects on screen, based on this data. Unlike most traditional (non-interactive) QuickTime movies which feature a series of flat, pre-rendered video frames which play out according to a set timeline at a specific frame rate, regardless of what the user is doing, *Brick Style* is based on real time programmatic animation. This means the ball, paddle and background are discrete objects which are dynamically rendered on screen according to user input and scripted motion. Rather than being strictly pre-defined, the ball and the paddle are controlled by scripts which define rules for their movement over time, how to react to collisions between elements, and the user’s input, all as the movie plays. By using simple scripts to control objects rather than a series of flat, pre-rendered frames, a movie can potentially play forever without repeating, reacting dynamically to a range of user inputs and other network signals. *Brick Style* proved to be a useful test, showing that QuickTime could be highly responsive, controlling the movement of images based on user input in real time.

With my experience in live sound and video performance and the idea of the computer as a ‘noisy’ environment to be explored through software works, I was beginning to find that ideas and techniques from the history of sound art could be a useful way of informing and framing my work. By thinking of my work in this context I could also situate my work outside of cinema, narrative storytelling and gaming. Having found that I could produce works which could respond to user input in real time by moving simple images on screen, I wanted to test how QuickTime dealt with the manipulation of video images in real time. Rather than producing more game-like works, I decided to attempt to create a work which would function more like an instrument or machine.

A notable artist who’s work was particularly influential at this point in my research was John Whitney. Working in the emerging field of motion graphics and computer generated imagery from the 1950s onwards, Whitney sought to produce a ‘digital harmony’ between ‘visual music’ and sound (Whitney, 1980). In order to achieve this goal he developed custom hardware and software as tools with which to create his works. Quoted in Gene Youngblood’s *Expanded Cinema* (Youngblood, 1970: 207), Whitney wrote: “My computer program is like a piano. I could continue to use it creatively all my life.” The key difference, however, between the works I was beginning to create and Whitney’s, was that while his mechanical, computer controlled animation and optical film processes involved lengthy processes, my vidgets operated in real time. Where he used his ‘piano like’ software as a compositional tool, I was able to use my vidgets as live performance instruments.
REMOTE CONTROL 0.2

Remote Control 0.2 was designed as a prototype audio-visual instrument for live VJ performance. Having found that the QuickTime architecture was capable of responding to user input in real time through the use of sprites and scripting, I wanted to test its capabilities for the real time manipulation of video images. The work is made up of two QuickTime movies. The first movie functions as a set of clickable buttons and controls. When clicked, these buttons trigger scripts which send commands to a second ‘receiver’ movie, which functions as a video output window. Remote Control 0.2 uses a technique called ‘inter-movie’ communication to allow the two independent QuickTime movies to send and receive data from one another, allowing one movie to control the visual playback parameters of the other. The use of this technique continued the idea of expanding my video practice to include the use and manipulation of non-traditional data communications.

Figure 2 Remote Control 0.2’s Control Window (May 2004)

Figure 3, Remote Control 0.2’s Control Window (May 2004)
The division of the work into two windows references a range of standard media production software such as Final Cut Pro or Adobe After Effects. This software features separate widows for user input such as timelines, tools and effects and ‘preview’ or ‘output’ windows where the user can see the results of their actions. Using a separate window for output, Remote Control 0.2 allows the user to view the controls on one screen (for example, the built in display of a laptop) whilst displaying the output at full screen on a video projector. This was a significant development as it facilitated the use of the work and subsequent works, such as Vidget 3.5 (discussed in the following chapter), as live performance tools. In this case, I wanted to produce a machine or instrument which allowed the user to manipulate some of this data in a self-directed and performative manner in order to mix and effect video images.

In Remote Control 0.2 the ‘output’ movie is made up of three video tracks. These tracks feature pre-rendered video loops which are layered on top of one another and play continuously. Like layers in a photo editing application, such as Adobe Photoshop, in QuickTime, every visual track has an associated ‘blend mode’. By default, the blend modes of visual tracks in QuickTime are set to ‘dither copy’. This mode displays images at full opacity, meaning that when layered only the uppermost track is visible. By changing the blend mode, images may be displayed and mixed in a range of different ways. In this work I explored the use of the additive blend mode, which uses the visual track’s image to brighten the images below, a subtractive blend mode which inverts the image’s colours and a variety of opacity settings to mix images together. By clicking the buttons in the ‘sender’ movie, the three layers may be displayed, mixed and affected in real time.

Figure 4 Two images with various different blend modes applied
While video production applications such as Adobe After Effects and Final Cut Pro may allow users to effect and combine multiple layers of video, the key differences with works such as Remote Control 0.2 are that they operate in real time and employ a soft copy approach. While Final Cut Pro allows the user to control how layers are blended, the user cannot generally see the effects of these changes until the video has been rendered. This is partly because such applications are based on a traditional film editing and effects production workflow rather than that of a live video performance system or even live television broadcast. Just as optical effects must be printed onto film for projection, multiple layers of video must be rendered down to one final layer or ‘hard copy’ for output. With Remote Control 0.2 the three layers of video remain discrete soft video objects. All changes to blending modes are executed as the video files play and may be undone or adjusted repeatedly. The encoded video data itself is not being changed, only the variable data which determines how it is displayed on screen. This means that the user may receive instant feedback on their input and make endless changes without filling their hard disk drives with render files. In a VJ performance context real time feedback is essential and the ability to combine a given set of video elements in multiple ways is very useful. These techniques, first developed in Remote Control 0.2, were used in several subsequent vidgets as I explored the idea of creating soft video machines (See Vidget 1, Vidget 3, and Vidget 3.5 in Appendix 1).

Audio-visual Noise Machines

With Remote Control 0.2 and the several vidgets that followed, I was interested in exploring the development of audio-visual ‘noise machines’. These noise machines were designed as tools or instruments which could be used to interact with the all-data in a live video performance context. These works were intended to function much like the acoustic noise machines designed and built by Italian futurist noise musician Luigi Russolo. Russolo built Intonarumori, mechanical ‘Noise Intoners’ which were used to incorporate the noises of everyday industry into their musical performances. These instruments allowed performers to synthesise and manipulate a range of sounds from the percussive through to howls, roars, whistles and buzzes by turning cranks and shifting levers to manipulate the sounds’ pitches and tempos (Russolo, 1913: 10). Each instrument was defined by the particular range of sounds it produced. Where Russolo built mechanical devices in order to generate everyday noises from outside of the traditional musical realm, I created software devices in order to manipulate a range of everyday data from outside of traditional video practice. With Remote Control 0.2 this involved exploring the use of inter-movie communication, user input via mouse clicks and buttons, and the manipulation of graphic blend modes in real time. In the vidgets which followed I began to explore possibilities of allowing users to access and manipulate a wider range of user-contributed content and content generated by others on the network.
THE SHIFT FROM STATIC TO DYNAMIC WORKS, DISCRETE OBJECTS TO DISTRIBUTED SYSTEMS.
The next group of works, *Random Site, QuickTime Flickr Viewer, and Vidget 3.5* saw a significant shift in my practice. I moved from the production of audio-visual objects which included pre-defined audio-visual content which could be manipulated, to the creation of more open systems. These works dynamically loaded content from independent networked sources, separating the new vidgets from their content completely. I continued to explore the idea of developing audio-visual ‘machines’ or ‘instruments’ however these new works were defined by their functionality rather than their content.

As a result of this shift I began to find that aspects of the ‘noise machine’ metaphor I had been using to frame my work were at odds with a number of the dynamic and networked characteristics I was exploring through the new works. While each of Luigi Russolo’s *Intonarumori* were defined by the particular range of sounds they could generate, my new works were no longer limited to the presentation of a fixed set of predefined images. While Russolo’s work was a useful reference which shaped the development of vidgets and the concept of the all-data, I began to look for new ways to situate and conceptualise the interaction between my works, user input and other data from the network. As I developed the following works, I began to re-examine my initial definitions of the two key terms I was exploring: network awareness and real time interactivity. In this section, I discuss the changes in my conception of these terms as they were informed by the production of *Random Site, QuickTime Flickr Viewer, and Vidget 3.5*. 

XML

XML is a simple but very powerful 'markup-language' which may be used to store, transmit and describe a range of data. It is generated and read by a wide array of networked applications, devices and services, operating in the background of a multitude of inter-machine data communications. XML allows a range of different applications and online services, operating on different platforms, to transmit, receive and interpret information. Since the XML format is based on plain text characters, it is possible to create files in a number of different ways. Files may be written manually using text editor applications, programmatically through custom scripts running on web servers using languages such as PHP, or through a range of pre-existing web services and applications. As this was my first dealing with the format, I wrote the XML file used by Random Site manually with a text editor.

Figure 5 An excerpt from the XML file read by Random Site

<site>
  <SiteName>Radiohead TV</SiteName>
  <URL>http://www.radiohead.tv/</URL>
</site>
<site>
  <SiteName>dpwolf/blog</SiteName>
  <URL>http://hypertext.rmit.edu.au/~dpwolf/blog/</URL>
</site>
<site>
  <SiteName>Electundra</SiteName>
  <URL>http://www.electundra.com/</URL>
</site>
*Random Site*

*Random Site* is a work which investigated two key techniques associated with network awareness: the separation of a work’s function and its ‘content’; and the use of XML (Extensible Markup Language) as a means of storing and retrieving data. While *Random Site* may look more like a simple html web page than what we traditionally think of as ‘video’ on the computer, it is in fact a QuickTime movie. The movie features a number of black text elements on a blank grey background. Rather than being a fixed, rasterised image, as is usually the case for text on screen in cinema or video, these textual elements are editable text tracks. At the top right of the window is a bold, blue, underlined word ‘Randomise’. When clicked, this button triggers a script which loads in a separate file containing a list of websites. Once loaded, the script randomly chooses one of the sites from the list and displays the web site’s name and URL on screen. The web site’s URL functions as a link which, when clicked, launches the site in a web browser. *Random Site* could be described as a software ‘widget’, a small ‘mechanical’ device or gadget with a simple function. The work is comprised of two elements, a QuickTime movie file and an XML file, located at a particular URL (http://dpwolf.net/blog/sites.xml.)

*Random Site* explores the idea that rather than containing a fixed, predefined collection of images and sounds, on the computer a ‘video’ (in this case a QuickTime movie) may function as an empty container into which a range of media may be loaded. This content data may be selected based on user input, embedded scripts and/or external databases. By using a remote XML file to define it’s content, a single work may be designed so as to present an almost infinite array of different media content. By changing the data in the XML file, the contents of the movie may be edited, updated or changed completely without having to change the movie file itself. While *Random Site* deals primarily with text and simple links to web pages, by developing scripts which allow QuickTime to parse XML, the piece lays the groundwork for works which dealt with a range of still and moving images. In subsequent works such as *QuickTime Blogroll* (See Appendix 1) and *QuickTime Flickr Viewer*, Vidget 3.5 and my performance at *Electrofringe 2005* (See Appendix 2), I began to use remote data generated by third party online services to source text and images for real time manipulation.
Tagging

One of Flickr’s most interesting and useful features is the use of ‘tagging’ as a user defined method for categorising and searching for images. A ‘tag’ is a keyword or short phrase which may be used to classify a photo. Rather than using a fixed set of categories to classify images, Flickr allows users to define their own tags. A single photo may be ‘tagged’ with multiple terms and each of these terms may be used to search for other identically tagged images. Its extensive database of images and searchable tag metadata made it an ideal source to be accessed in this work. Another of Flickr’s useful network based features utilised by QuickTime Flickr Viewer, is the ability to automatically generate RSS and Atom feeds of recently uploaded photos in real time.
QuickTime Flickr Photo Viewer

QuickTime Flickr Viewer was another significant proof-of-concept which expanded the idea of creating a work which contains no fixed audio-visual ‘content’ but instead provides the user with a means of accessing and presenting content from remote networked sources. Through the production of this work I investigated the shift from creating self contained, author defined works to the development of more open, network aware systems. The work functions as a system in abstract, facilitating the exchange of data between the user and a remote external source, providing the user with a wide range of potential content. I found that this kind of abstraction is one of the defining features of the networked computer environment I was exploring in this project.

QuickTime Flickr Viewer allows the user to search and browse through photos from Flickr, a complex web service designed for photo sharing and social networking which allows users to upload photographs, add comments and make annotations. By entering a search term into a text field in QuickTime Flickr Photo Viewer and pressing enter, the user may view the ten most recent images uploaded to the site which were ‘tagged’ with that particular term. The photos appear as thumbnail images on the left hand side of the viewer. When clicked, the images appear at a larger size in the ‘viewing area’ on the right.

Flickr’s combination of user defined tagging as a method of categorisation and the automatic generation of feeds for each tag allowed me to create QuickTime Flickr Viewer. The work accesses and interprets these feeds and presents the images referred to for viewing. Using tags entered by the user, URLs for the associated feeds of images are composed. When the user enters a tag into the text field and presses enter, the tag is copied by a script and used to assemble the URL for Flickr’s corresponding feed of images. The URL uses the tag as an “attribute” when querying Flickr’s database of images. For example, the tag “cow” is used to produce the URL “http://api.flickr.com/services/feeds/photos_public.gne?tags=cow&format=rss200”. This feed is loaded by the movie and parsed by a script to extract the locations of the first ten images referenced.
RSS and Atom Feeds

The RSS (standing for Rich Site Summary, RDF Site Summary or Really Simple Syndication) and Atom groups of standards were originally designed to facilitate the syndication of text based articles from news websites and blogs. As the popularity of text syndication increased, the specifications were soon extended to syndicate the distribution of a range of other data such as photos, audio files (podcasts), video and software updates. Subsets of the XML specification which I had begun to use as a way of storing and accessing data in Random Site, feeds (whether RSS or Atom) generally contain a web site or service’s most recent entries, date-stamped with time information and links to the original source pages. They feature the basic content without styling, menus or lists of older, archived content. Rather than being intended for human readers directly, feeds are intended for computer to computer communication. RSS and Atom feeds are designed to be read by stand alone ‘news reader’ applications or web based services which separate out the individual entries contained within the feed and present them to the user as individual items.

Feeds are usually generated dynamically by content management systems and other web services. Rather than being manually written, the feeds are assembled automatically via scripts running on a web server. Dynamic content management systems and other web services generally store their content in a database, and generate individual pages and feeds on demand. This means that whenever a new item is added to the site’s database, such as a new photo uploaded to Flickr, it is also instantly added to the associated feeds. A feed is accessed via a URL which effectively functions as a query, asking to check the site’s database for recently added items and returning the results. Rather than being a permanent link to specific content, a feed’s URL is a link to a query, retrieving recent items from a certain source or category. A single website may generate multiple feeds for different categories of content. For example, a news site may generate separate feeds for world news, local news, politics, entertainment and sport. Checking the feed on different days, or even different times of the day may retrieve different stories from that category. To use a newspaper as an analogy, rather than asking for a particular article (eg. ‘Show me the story written by X which can be found on a particular page of a particular issue from a certain date’) the feed’s URL asks for a more general and time specific answer (eg. ‘What are today’s headlines?’ or ‘What stories are in the sports section today?’). In the case of the Flickr feeds used by QuickTime Flickr Viewer the question is, ‘What are the ten most recently uploaded images which are tagged with this term?’.
One major limitation of QuickTime's XML handling capabilities is the inability to extract the “attributes” used in RSS and other XML schemas. In XML data can be stored as content between tags (for example, \p{tag}\p{content}/tag\p{}) or as attributes in quotes (for example, \p{item}\p{value}="attribute" \p{}/). Consequently I had to write my own XML parser script to extract the image information from the Flickr RSS feed. Once extracted, the image URLs were stored by the movie as variables for use in the display of thumbnails and full sized images. When images are uploaded to Flickr, several different sized files are automatically generated and stored on Flickr’s servers (thumbnail, square, small, medium and large). These files are named according to their size by adding a particular suffix to the filename (“_t.jpg” for thumbnail, “_m.jpg” for medium and so on). QuickTime Flickr Viewer appends each of the image’s URL’s to refer to thumbnail sized versions and loads them as a grid of sprite images. These sprites act as buttons which, when clicked, run scripts which load the full sized versions of the images into a movie track via another modification to the image’s URL.
As I explored the use of remote sources of data from the web such as RSS feeds, I began to realise that real time interactivity and network awareness and access were interrelated phenomena. I found that thinking of a URL as a 'universal' fixed location to access a networked resource or service belied the fact that any URL is really a command. Unlike a physical location, a URL is really a query. This query is passed from machine to machine over the internet until it reaches (or fails to reach) its intended destination where it may be executed and trigger the sending of data in return (Galloway, 2004). Where a description of a physical location, such as a latitude and longitude value, is fixed and has a direct relationship with the object to which it refers, a URL is open to the uncertainty and variability of networked systems. These systems may be designed so as to return different responses to the same query depending on the time or date of the query, geographical location, the user’s identity, or any number of programmatic operations. These operations may be performed by web applications and scripts running on the server in real time. Once a response to the query has been loaded, the resulting data must be interpreted and prepared for presentation by an application such as a web browser, which in turn runs inside an operating system. Changes may be introduced by the manipulation of any one of these processes or protocols.
**Vidget 3.5**

Continuing and expanding on the idea of creating a computer based video work which contains no fixed ‘content’, Vidget 3.5 functions as an interactive system for accessing, manipulating and presenting a variety of audio-visual content, all chosen by the user at runtime. Vidget 3.5 was designed primarily as a VJ tool for use in my own live performances. It was used and tested in several performances beginning in late 2004 including *Plug n Play*, *Electrofringe 2004* and *Straight Out of Brisbane 2004* (see Appendix 2 for more information on performances). The work was based on the combination of several techniques and ideas tested and explored individually in previous proof of concept vidget works. It brought these individual functions together to form a more complex and flexible interactive tool suited to VJ performance. I wanted to incorporate the real time layering and manipulation functionality of my previous VJ orientated works with the photo searching capabilities of *QuickTime Flickr Viewer*. I also wanted to develop a straightforward and accessible user interface and a more flexible system for defining libraries of video clips.
One of the problems with my earlier VJ performance orientated works was that their user interfaces were overly complicated and quite difficult to use in a performance context. This was especially apparent when comparing works such as Vidget 1 (Appendix 1) with simpler works like Brick Style and QuickTime Flickr Photo Viewer, which were easier to use but limited in terms of functionality. With Vidget 1 in particular, the task of selecting a video clip, loading it as one of three visual layers and manipulating the ways in which the layers were blended was a slow, complicated process involving several steps. Each of these steps involved clicking on a separate button and the user was given very little indication of what they would get when they clicked. The interface was based on text based buttons alone, and required the user to choose a clip, then select a destination and finally ‘activate’ their choices.

Since there was no real-time feedback between the user’s actions and their outcomes with these intermediate steps, the work was difficult to use in live performance where timing and control are important. As a consequence I also had difficulty explaining its use to others. With Vidget 3.5 I wanted to create a new, simpler user interface. This would make it easier to learn and would allow the user to concentrate on their performance or play without being distracted or confused by the interface itself.

Another limitation of my early experimental VJ performance works was that the selection of video ‘content’ was fixed at the time of authoring, incorporated into either the controller or output movie files. This meant that rather than being able to quickly collect a library of clips at the time of performance, users (myself included) were limited to playing with a selection of clips which I had prepared as I was creating the program. Whenever I wanted...
to incorporate new content I would have to edit, re-compile and build the works from their original source files in LiveStage Pro. This meant that other users (who most likely did not own a copy of LiveStage Pro) were forced to use my visual content rather than their own. As well as being designed for my own use in live performance, Vidget 3.5 was intended for distribution online. I wanted users to be able to easily play with the work in performances or self-directed experimentation using their own video content.

In order to solve these problems I developed Vidget 3.5 with both a new user interface and a new method for specifying the work’s content by merging a range of techniques developed and tested in previous works. In particular the new interface blended the use of sprites with dynamically loaded thumbnail images (first used in QuickTime Flickr Viewer) with the use of an external XML file to define the movie’s content (first explored in Random Site). The new interface used a grid of thumbnail images to represent both still images and video clips. As well as visually representing the video clips, I created the new interface so as to allow the user to “drag and drop” the thumbnail images onto “targets” for each of the three video layers, loading their respective video clips or still images in the corresponding layer of the output movie. This new drag and drop interface was based on sprite animation and the handling of real time mouse movement and clicks (first tested in Brick Style).

Where QuickTime Flickr Photo Viewer sourced it’s thumbnail images (100 x 75 pixel jpeg files) from Flickr, with Vidget 3.5 I needed to manually create my own jpeg thumbnails to represent the video clips with which I was dealing. In order to create a neat grid of images I chose to use the same image resolution for my own jpeg thumbnails, manually exporting still frames from QuickTime Player for each clip. Similarly, where QuickTime Flickr Photo Viewer used a dynamically generated RSS feed to define it’s contents, with Vidget 3.5 I created my own simple XML database to specify a library of clips to be loaded and manipulated.
This external XML file was used to list both the clips of video and their thumbnails. I used a file naming convention whereby a single name was used to specify each QuickTime movie and it’s jpeg thumbnail image. The file’s extension was used to differentiate between the two formats. By naming my files in this manner I was able to store a simple list of the file names (without extensions) in the XML file. For example, a clip featuring a cow would be entered into the XML file simply as “<clip>cow</clip>”, it’s corresponding movie file would be named “cow.mov”, and it’s thumbnail image would be “cow.jpg”. By using this naming convention I was able to keep this XML file as simple as possible, meaning that users could easily edit and create their own libraries of clips with a text editor without needing a complex knowledge of the XML format.

The controller movie in Vidget 3.5 uses code adapted from Random Site and QuickTime Flickr Viewer to read and parse this XML file, storing the files’ names as indexed variables. The movie then loads in a grid of up to 25 thumbnail images as sprite images, adding the “.jpg” extension to each file name to locate the images. Like Remote Control 0.2 the work uses one QuickTime movie as a set of interactive controls and another as an output, using inter-movie communication to send data from one movie to another. Rather than using text based buttons to trigger commands from the controller movie for loading clips in the output movie (as I had done in Vidget 1) or clickable thumbnails (as used in QuickTime Flickr Viewer), I developed a new drag and drop interface. This interface used techniques adapted from Brick Style to track the position of the user’s cursor, mouse clicks and the boundaries of each thumbnail sprite. Continuing from Remote Control 0.2 and Vidget 1, Vidget 3.5 allows the user to mix up to three layers of video together. As well as changing the transparency and blend modes of the three visual layers, the work allows the user to manipulate the playback speeds of each clip individually. Rather than mixing the images down into one single layer, the work remains ‘softvideo’–infinitely editable and malleable. Incorporating the Flickr photo search capabilities I developed in QuickTime Photo Viewer, the work allows the user to search for images, view preview thumbnails and then mix them together with other still and moving visuals.
In the case of *QuickTime Flickr Viewer*, I had explored the idea of framing the work as an interactive system, designed to facilitate the exchange of everyday data from one source to another. Through the development of *Vidget 3.5* I took this process one step further by re-adapting and building on my own previous pieces of code. This created an internal “system of systems”, stringing existing scripts and functions together in new ways to create new works. This was a particularly important factor in the development of my practice which furthered my understanding of the modularity afforded by the interactive, network based video production process. I began to look for alternative ways of framing and situating my practice in order to inform and reflect upon this approach.
REAL TIME SYSTEMS,
REAL TIME PROGRAMMING
System Esthetics [sic] and networks of interaction

As this project progressed I increasingly defined my works as networked systems and collections of interrelated functions operating together in real time. I found that Jack Burnham’s writing on post-formalist sculpture and conceptual art provided a context with which to locate and reflect upon my own work. Burnham is an artist, art critic and theorist who wrote on the influence of new technology on the world of sculpture and conceptual art in late 1960s North America. In his paper *Systems Esthetics*, Burnham writes: “We are now in transition from an object-orientated to a systems-orientated culture. Here change emanates, not from things, but from the way things are done.” (1974a:16). Rather than defining art works simply as physical objects, he argues that they are products of larger art systems and networks. He writes, “… in essence, all institutions which process art data, thus making information, are components of the work of art” (1974b, 27). With this rationale, Burnham argued that art may be created by co-ordinating pre-existing systems inherent in the everyday environment and framing them as works. Vidgets may be designed so as to use pre-existing data, standards, services and protocols in novel ways. In the all-data environment explored throughout this project, any given work relies upon the combination and co-ordination of various technologies, systems and services, all working together to function. Burnham argues “The art object is, in effect, an information ‘trigger’ for mobilizing the information cycle.” (1974b: 28). Similarly, a vidget functions as a trigger, managing the transfer of data between systems and devices in a digital information cycle.
Reflecting on the use of vidgets as real time systems for improvised video performance, I was interested in situating the concept of the ‘real time’ in art theory and practice. Having established that systems of interaction and communication were vital aspects of network awareness and data access, I found that real time interactivity, similarly, was largely dependent on environmental awareness.

In *Real Time Systems* (1974b) Jack Burnham discusses the real time nature of system and processed based art. Burnham defines real time systems from a technical, engineering perspective as “systems which gather and process data from environments, in time to effect future events within those environments.” (1974b: 29). To illustrate this definition, he lists a number of pioneering real time systems of the late 1960s: “SAGE, the first computer-based air defense system; Project Mercury, the first real time digital support system for space flight; Telefile, the first online banking system; and SABRE, the first computerized airline reservation system...” (Burham, 1974b: 29). Such systems allow the user to observe the current status of events, make changes to them, and instantly see or hear the results.

While these systems may be highly automated, they may also provide the user with the opportunity to improvise. For artists working with technology, this is often a vital part of the creative practice. As Smith and Dean note: “A very simple definition of artistic improvising is that it is the simultaneous conception and performance of a work” (1997: 3). “Pure improvisation, therefore, takes place at the intersection of performance and creativity ... This involves an attentiveness to the present moment, so that creativity is a response to the here and now” (1997: 26). In terms of my own practice, this instant feedback afforded by real time systems is vital for VJ performance.
Quartz Composer and the shift to real time programming

At the end of April 2005, well into my research, a new version of the Mac OS was released. The new operating system, Mac OS 10.4 “Tiger”, featured a number of fundamental changes in the way all graphics and video were rendered on screen. This would have a great effect on my subsequent works. Dubbed Core Image and Core Video, these new graphical processing methods were designed to utilise the powerful GPUs (Graphical Processing Units) of modern computers to accelerate the display and manipulation of all still and moving images. Rather than using the computer’s main CPU (Central Processing Unit) to render graphics and video, Mac OS 10.4 allows developers to easily design software which processes images in real time using the graphics card. This was achieved by dealing with all images on screen as surfaces in a three dimensional space using the OpenGL specification, the GPU’s highly optimised native language.

Initially, I was not sure how the changes would affect my practice. I had made a conscious decision to use LiveStage Pro and QuickTime as an authoring environment due to its cross platform capabilities, network aware functionality and the fact that it was ‘everyday’ software, not requiring the user to perform any special installation or configuration. While this low level system optimisation was interesting, from the beginning of this project I was not aiming to develop stand alone software for Mac OS and had no time to learn a completely new programming language such as Objective C in order to create such applications.

These changes to the operating system would not have been useful to me in this project if not for the release of Quartz Composer, a real time, visual programming environment and developers tool for processing and rendering graphical data. Included free with Mac OS 10.4 as part of the ‘Developers Tools’ package, it is specifically designed to utilise Core Image and Core Video functionality. Quartz Composer allows the user/developer to quickly and easily create “compositions” – real time interactive systems for generating, accessing, processing and manipulating images. Quartz Composer uses a visual patching interface similar to those employed by real time audio software and multimedia performance environments such as Reaktor and Max/MSP. These visual programming based interfaces allow developers to create complex interactive works rapidly by connecting modular patches rather than writing text based code.
Visual programming in Quartz Composer

To simplify the use of Core Image and Core Video functions, Quartz Composer uses a visual programming paradigm which is built on the use of ‘patches’. Patches are functional objects which have inputs and/or outputs, and which may be connected together in chains to perform set operations. Just as a guitarist may plug their instrument into a delay pedal, a distortion pedal and then into an amplifier to create a chain of effects, Quartz Composer allows the user to plug a series of patches together in order to generate, process and display images. Based on the values of their input parameters, patches may output images, numbers or other data which may, in turn, be used as the input for subsequent patches. For example, in order to adjust the brightness and contrast of an existing QuickTime movie, the output of an ‘Image with Movie’ patch may be connected to a ‘Color Controls’ patch which, in turn, is connected to a ‘Billboard’ patch for rendering on screen. Just as each device in a guitar effect chain may have controls for tone, distortion, or delay time which affect the sounds as they pass through, each patch may have a number of input parameters which influence the flow of data or visual effect as it passes through the chain.

Figure 12 the Quartz Composer editor window showing a basic composition
Instances of patches are placed in an editor window, a visual programming workspace, and linked together by dragging the output port of one patch to the input port of another to create a visual connection. Unlike guitar pedals which are only designed to be linked together by connecting audio outputs to audio inputs in order to process audio signals, patches in Quartz Composer may be connected to deal with a range of non-visual data such as mathematical functions, live audio input as well as images. This data may be used to control the access, generation and manipulation of still and moving images. For example, in order to automate the manipulation of a movie’s brightness over time, a ‘LFO’ (Low Frequency Oscillator) patch may be used to generate a stream of numbers which oscillate within a certain range at a particular rate. This patch’s output signal may be connected to the ‘Brightness’ parameter input of the ‘Color Controls’ patch. The clip’s brightness will increase and decrease as it plays on screen. Just as a chain of guitar effects must be connected to an amplifier to make a sound, a composition in Quartz Composer must contain at least one patch classed as a ‘renderer’ in order for the composition to execute and display on screen. Again, just like a chain of audio devices, a Quartz Composer composition runs in real time, beginning as soon as it is patched together, reacting instantly to parameter changes.

In LiveStage Pro real time user input, such as tracking the mouse position, was handled by scripts set to run using the ‘idle event handler’. The frequency at which these ‘idle’ events were repeatedly executed depended on a number of factors. This frequency could be manually set to a high rate, resulting, for example, in accurate tracking across time, however this significantly increased the CPU usage of the movie, potentially dropping the overall frame rate. For this reason, the QuickTime architecture was not suited to the use of highly variable time based user interface elements such as sliders when being used to control CPU intensive tasks. In comparison, the Quartz Composer environment was designed particularly to deal with this type of input. In LiveStage Pro I had to be careful in utilising the idle event handler sparingly. In Quartz Composer almost all events and functions were executed continuously with no loss of frame rate.

This capacity for instant real time feedback and fine control of variables over time made the programming environment particularly suited to creating works which were highly ‘tweakable’, much like audio synthesisers.
Two practices into one: Real time programming and performance

Before I began to explore the use of Quartz Composer, the focus of my creative practice had centred on two complimentary, but quite separate activities:

- The creation of vidgets. These were interactive works and VJ tools produced through the development and assembling of functions and systems from the digital environment. These systems were structured and framed as distinct, network aware works, largely independent of content and designed for real time interactivity in performance and play.

- Audio-visual (VJ) performances, accompanying live musicians, sound artists and DJs with live improvised video projections. These performances were created through the use of vidgets, manipulating the playback of both pre-composed and found visual elements in real time.

The use and testing of each vidget in live performance contexts informed the development of subsequent works, however, these two activities were completely distinct, occurring at different times in different places. Since I was creating vidgets with LiveStage Pro, which was built on the development of written programmatic scripts, I needed to finish, debug and compile the works before I could test them. As I developed skills and knowledge in this area the challenge became the development of logical systems which linked functions and pieces of code together. While my video performance work was improvisational in terms of the selection and manipulation of visual content, it was limited to the use of pre-existing functions and systems developed at the time of programming. As the complexity of these QuickTime based vidgets increased with works such as Vidget 3.5, I had less opportunity to be flexible in my programming without completely starting from scratch.

With the adoption of Quartz Composer as a programming environment, the distinction between making vidgets and using them in performance began to blur. The programming itself became a real time operation, allowing for instant feedback, improvisation and ‘live coding’ (Ward et al, 2004). As I experimented with this new programming method I explored Quartz Composer’s capacity for the real time manipulation of images. Much of this experimental practice occurred as part of live improvised performance, either in private, or at Plug n Play, a weekly audio-visual event run by Melbourne based audio-visual artist Sean Healy and myself (See Appendix 2 for a list of performances). Through a process of live coding I was able to quickly explore and then master the use of the different sources, functions and effects. I patched together compositions in a live performance context whilst viewing the results as they were projected on screen. Whenever I came up with an interesting visual effect or method for patching I was able to save the composition for re-use and incorporation into new works. Many techniques featured in my Quartz Composer based works were originally developed through this form of improvised performance.
Quartz Kaleidoscope

Quartz Kaleidoscope was one of my earliest works developed in Quartz Composer. Just as I had originally explored the use of the QuickTime architecture through the development of works such as Vog 001, Inter-Movie Text Communication, and QuickTime Blogroll (see Appendix 1), Quartz Kaleidoscope was designed as a proof of concept to investigate the use of Quartz Composer as an authoring environment for the creation of real time interactive, network aware video works. In particular, Quartz Kaleidoscope was designed to experiment with Quartz Composer’s patching based, visual programming interface, it’s ability to dynamically manipulate video images in real time, and its capacity for remotely sourcing video content.

My early works developed with LiveStage Pro required me to extensively read the user manual and learn the QScript written programming language by following tutorials before I could get my own works to function at all. In contrast, once I had an understanding of Quartz Composer’s visual programming interface I was able to quickly create experimental works, improvising by patching together chains of nodes to see how they functioned.

By plugging patches together on screen in this way I was able to create a composition which played a QuickTime movie with a kaleidoscopic effect. By attaching an LFO to the ‘angle’ input of the kaleidoscope effect I was able to automate the rotation of the effect. Testing another of Quartz Composer’s real time capabilities, I was able to have the composition react to the volume of live audio input by zooming the image in and out. In terms of testing Quartz Composer’s real time manipulation of images, I found that Quartz Kaleidoscope clearly illustrated its advantage over the older QuickTime architecture with highly responsive performance, excellent frame rates and complex visual effects capabilities.
After authoring QuickTime based vidgets which were designed for networked distribution and to facilitate the sourcing of remote video files, I was keen to compare and contrast Quartz Composer’s network access capabilities. While QuickTime Player is designed to manage the dynamic loading of remote files, unfortunately Quartz Composer is not. In an attempt to work around this limitation, with Quartz Kaleidoscope I used a QuickTime reference movie, which in turn, pointed to a movie file hosted on the RMIT Hypertext web server. The remote movie was eventually played, however, since Quartz Composer is only designed for playing back local files, the composition stuttered until loading was complete.

While this method for loading remote video content worked, I did not return to its use due to the stuttering playback problems. As an alternative, I found that rather than attempting to load remote video material through Quartz Composer itself, I could simply download the files through a web browser or QuickTime Player, save them to a local folder and drag and drop them into the composition for use. Since Quartz Composer was deeply integrated into the operating system, I found many examples of this type of simple but effective functionality as my research progressed.

In my initial experiments I had been using Quartz Composer as both a programming and playback environment for performance and construction of experimental works. Working in this manner allowed for easy access to any parameter from any part of the composition for manipulation in performance. One limitation, however, was that the interface used a floating “inspector” window which was designed primarily for the manipulation of one value at a time.

In live performance, I often wanted the ability to quickly change multiple settings quickly. While I could do this by selecting a number of input parameters and “publishing” them as master controls which would appear together in one inspector window, this process was not elegant. If I was to distribute my works online so that others could use them easily, I decided that a more intuitive user interface would be required.
Bikelights: a Quartz Vidget

With *Bikelights: a Quartz Vidget*, I began to explore the use of Quartz Composer and Xcode, Apple’s integrated development environment for constructing and compiling applications, to create a prototypical, stand-alone, video performance application. It was designed to separate ‘the work’ from it’s content just as I had in earlier QuickTime based works such as *Random Site*, *QuickTime Flickr Viewer* and *Vidget 3.5* (see previous chapter). Much like *Remote Control 0.2*, this work served as a proof of concept, exploring the development of a new type of user friendly interface and new production processes.
Beginning as a basic Quartz composition, Bikelights was initially developed through live experimentation. Through this process of improvised experimentation I designed and constructed an interactive system for selecting from a bank of five QuickTime movies allowing for their display in a number of different ways. This composition featured the ability to adjust a clip’s playback speed, brightness, contrast and saturation and the use of various different ‘renderers’ which would display images on separate layers. Building on techniques tested in Quartz Kaleidoscope, the composition was designed to use live audio input to manipulate the positioning of these layers of video.

Rather than simply releasing an interactive Quartz Composer file for others to use, I wanted to utilise the programming environment’s links to the Xcode software development tool to explore the creation of a stand-alone application with a customised user interface. In particular I wanted to develop a simple way for users to load their own video content into the application and control it’s playback via a series of on screen controls. After reading a tutorial on Apple’s developer website (Apple Computer, Inc. 2005b), I was able to proceed with creating my first stand-alone application.
In order to map each of the composition’s input parameters to customised user interface elements for user control, I selected and “published” each input, naming them uniquely so that they could be addressed programatically. Using Xcode, I imported the Quartz composition into a new project and was able to create a new ‘window’ element with Interface Builder, a graphical user interface development tool. Using Interface Builder I created a range of standard Mac OS user interface elements such as sliders, check boxes and text fields and ‘bound’ them to each of the published inputs, addressing them by name. The project was then compiled and assembled in Xcode.

This process produced a stand alone application which would run on any computer running MacOS 10.4 without the need for the user to install any additional software. Running as a stand alone application, I found that it was possible to rapidly load QuickTime movies into Bikelights by dragging them from the desktop into the text field locations, thus allowing users to easily manipulate their own video content. Through the development of Bikelights I found that by using Quartz Composer and Xcode I was able to quickly create custom video applications which were easy to use and share. This paved the way for the creation of more complex performance applications with the ability to mix multiple visual sources at once.
In June 2005 I was invited to perform live video as part of two international arts festivals in Melbourne. Remains To Be Seen (see Appendix 2 for details) was organised as part of the Melbourne International Animation Festival (MIAF) in order to showcase Melbourne’s live video artists and audio-visual performers. A few days later, I performed with fellow Melbourne VJs Jean Poole (Sean Healy), keith_d (Keith Deverrell), Lindsay Cox and the VJ Zoo collective (Kat Black and Jasper Cook) from Perth as part of Liquid Architecture 6 (LA6), a festival of sound art (see Appendix 2 for details). In preparation for these two performances I decided to produce a new Quartz based vidget, MIAF 2 LA 6. Bikelights had served well as a proof of concept, investigating the processes involved in the construction of a stand alone application from a Quartz Composer composition, however, with this next work I intended to create a more flexible tool designed for video performance.
Taking *Bikelights* as a starting point, I added several new features as I created *MIAF 2 LA 6*. These new features were intended to create a more flexible performance tool. This new work featured two banks of clips which could be selected for playback two at a time, where *Bikelights* only allowed one video source. Manipulating two clips at once, the images could be layered and combined in various ways. Like *Bikelights*, the video images could be connected to various renderers as additive layers or to a “billboard” renderer as a subtractive mask. Each bank of clips featured a series of general purpose effects such as brightness and contrast which could be enabled and manipulated or switched off, bypassing the effect. Another new feature included in *MIAF 2 LA 6* was the ability to playback a sequence of still images from a given folder location. By dragging and dropping a folder of images into the ‘Folder Images’ field in the main window the user could cycle through the images contained, setting the rate of playback via a text input field. Depending on the rate chosen by the user this could result in a slow ‘slideshow’ effect or rapid animation of the images.

In use at *Remains To Be Seen* and *Liquid Architecture 6*, I found that the video playback performance was quite remarkable. I had previously compressed my video source material to a low resolution with a non-CPU intensive codec such as h.263 in order to achieve acceptable frame rates in QuickTime based vidgets. By comparison, with *Quartz Composer* based works running on computers with suitable graphics cards I found I was able to use full frame sized video captured straight from a DV camera without a noticeable slowdown. Released via my website, the work generated significant interest from VJs around the world.
From this point on, the *MIAF 2 LA 6* vidget became my primary VJ tool for general use in performance (see Appendices 1 and 2 for a listing of all of the performances made using this work). This was partly due to the fact that it was designed as a simple, general purpose tool for presenting and mixing video content, and partly due to the *Quartz Composer* architecture’s capacity to play compositions within compositions. Creating and saving new experimental compositions in *Quartz Composer* developed through live coding, I was able to manipulate and play these works back from within the *MIAF 2 LA 6* user interface. This meant that rather than focusing my attention on the creation of an increasingly complex application in order to add new features, as I had done through the development of *Vidget 3* and 3.5, I was able to continue the development of new, small scale experimental works. These generative and interactive works such as *Gadget Saver*, and the *Atmospheric Image Sequence* series (see Appendix 1) could be mixed with video elements, image sequences or other compositions whilst retaining interactive controls such as MIDI and live audio and video input. This process represented another aspect of the highly modular nature of the *Quartz Composer* environment.

**Specialised Real Time Performance Environments**

While the *MIAF 2 LA 6* vidget has served as my main general purpose VJ performance tool to this day, I have since developed a number of works which are designed for particular events as specialised performance instruments. Unlike most of my previous vidgets, these new works were not intended for distribution online but specifically for use in my own performances. This allowed me the freedom to use a range of idiosyncratic processes and controls which would be difficult to replicate on others’ machines and problematic to explain to other users. These tools or ‘performance environments’ feature highly customised user input, utilising technologies such as MIDI keyboards and controllers and Bluetooth mobile phones.

Unlike previous vidgets, these works also generally incorporated predefined libraries of visual source material and generative systems which produce a specific range of images and styles. These works represented a return to my earlier aims, in exploring the development of audio-visual noise machines in the tradition of Luigi Russolo. Whilst the use of these performance environments is still based on improvised performance, these specialised environments facilitated the repeated performance of certain ‘sets’ with distinctive aesthetics.
White Noise

The first of these specialised performance environments, *White Noise*, was inspired by the exhibition of the same name curated by Mike Stubbs which was held at the Australian Centre for the Moving Image from August through to October 2005. An exploration of minimalism in the digital age, Stubbs writes:

“*White Noise* is an exhibition in real-time. With varying degrees of luminosity, tempo and volume the artworks invite the audience into a space that is both physical and reflective: not to view pictorial representations of something, or document another time or place, but to invite us in to the here and now.” (2005).

The exhibition featured numerous mesmerising screen based works in which simple lines would slowly morph or shift. While most of the works in the exhibition were pre-recorded video loops, with *White Noise* I wanted to create similar minimal, abstract effects through the development of a real time interactive system designed for performance.
To achieve this, I created a video feedback loop which was continuously manipulated by a Quartz Composer composition. The computer’s analogue composite video output was connected to the input of a digital video camera whilst a digital firewire cable connected the camera back into the computer. Processing this live video source in real time via a Quartz Composer composition, the resulting images were sent out of the computer’s video output, looping back into the camera. In order to manipulate this finely balanced feedback system, I used a USB MIDI controller with a series of MIDI knobs and sliders. These controls were mapped to manipulate various parameters such as the colour channels (red, green, blue and alpha) of several video layers, and the width of a halftone filter effect.

The White Noise performance environment generated minimal, abstract images which constantly evolved and oscillated between brightness and darkness. It was particularly suited to accompanying highly textural noise performances as it allowed me to generate and manipulate images on screen with a fine degree of control and fluidity. Using the various MIDI controllers I was able to reflect subtle changes in the sound performances with changes in colour and form. It has been used in several performances in collaboration with sound artists such as Somaya Langley, Simulus and Doktor Robotnik in Melbourne and Newcastle (see Appendices 1 and 2 for more details on performances).
Gadget Performance Environment

Gadget was developed for use in a collaborative audio-visual performance with sound and new media artist Somaya Langley at an event in Canberra in early 2006. The work featured a combination of generative video systems running in real time with 3D particle effects, specially shot location based footage, colour, and accumulation effects processing. Building upon the idea of creating a specialised performance environment with a particular aesthetic, Gadget was designed to accompany an audio performance featuring highly reverberant field recordings of public spaces. In order to visually reflect this sonic environment, I developed an ‘accumulation’ effect which functioned much like a ‘tracer’ or slow shutter speed setting on a traditional video camera. Using this effect, moving objects would leave trails as they travelled across the screen. The work built upon White Noise’s use of MIDI controls, allowing me to finely control the rate at which images were accumulated and the duration of their time on screen before fading out. MIDI controls were also used to manipulate various aspects of a 3D particle system generating hundreds of circles of varying colour and transparency swirling and spinning in a 3D environment. This effect was designed to compliment Langley’s use of a gestural performance interface, producing a highly integrated audio-visual experience. As well as complimenting the sonic performance through visual effects, Gadget featured a live camera feed of Langley as she moved her arms to manipulate the sounds she was generating. The work also included specially shot footage of a notable site in Canberra which previously housed a

Figure 22 Gadget Performance Environment in use at Electrofringe 2006 with sound/new media artist Somaya Langley (October 2006)
community radio station and was being demolished to make way for an apartment block. As well as locating the performance in a particular time and place relevant to the audience, the footage was designed to accompany the social commentary implied in Langley’s sonic focus on public spaces.

The first performance of this work was highly successful, leading to a second performance, featured as part of Electrofringe 2006 in Newcastle. Rather than replaying the footage shot earlier for the Canberra performance, I spent a day travelling around Newcastle recording footage of another notable local landmark which was being re-developed as well as images from the coast and docks. Again, this situated the work in it’s performance locale, reflected Langley’s performance focus and resonated with the local audience.

This work represented a synthesis of previous techniques developed through this research project and a new direction as a complete audio-visual performance incorporating elements of both improvisation and a repeatable structure. It also represented a return to the use of more representational video material. Over the course of this study my practice has shifted from the production of narrative film and video works, through the development of interactive, network based works as formal and technical experiments, to the production and use of vidgets as audio-visual tools and instruments designed for live performance. As my practice continues to evolve I am interested in exploring new and hybrid forms of interactive, network based video work in both a live performance context and as distributed online works.
WHAT I HAVE LEARNT
In this concluding chapter I describe the distinctive aspects of interactive, network based video practice I have identified through the development and use of vidgets. Much of this exegesis refers to the changes which have occurred in my own practice as it has evolved into the development and use of vidgets as live video performance (VJ) tools. In this chapter however, I wish to reflect upon the knowledge and methods generated through this research and how it may be applied to interactive, network based video in general.

One of the major themes of this research has been an attempt to look at interactive, networked video practice in terms of its relationship to artistic practices other than cinema. This strategy has allowed me to identify significant similarities between real time interactive, networked, video practice and sound art and systems based art. The use of conceptual frameworks and techniques adopted from these fields has informed the development of my own practice considerably. In this final chapter it is useful now, however, to refer back to traditional film making in order to reflect upon the differences between the two practices, and in order to describe some of the ways in which film makers may utilise interactive, network based techniques to explore the development of new hybrid forms.
Abstraction

Through the production of vidgets, I have identified computational abstraction as one of the key characteristics which sets interactive, network based video practice apart from traditional film production. Abstraction refers to the use of variables and programmatic functions in order to deal with images and sounds, their selection, manipulation and presentation. Interactive, network based video producers often deal with abstract, programmatic ‘placeholders’ and scripts rather than specifying a particular series of audio-visual elements. By designing audio-visual systems in abstract, works may be designed so as to automatically include images and sounds from a variety of sources in a multitude of combinations and orders programmatically based on user input and the functions of the systems designed. Depending on the user’s actions and choices, every use of an interactive, network based video work has the potential to result in a unique audio-visual outcome. The work itself may therefore be defined more as an interactive system which facilitates the selection, manipulation and presentation of audio-visual elements rather than the actual images and sounds produced.

Using a systems esthetics approach, the traditional film making process can be described as an art-making system whereby a series of finite creative and pragmatic choices are made in order to determine the ways in which audio-visual material is created. The key difference between this process and those at work in interactive, network based video production, however, is that in film making all of these decisions are finalised specifically, long before the film is screened. Writers, directors, cinematographers, performers, editors and sound designers each have their input as the production unfolds, but the result is the construction of a single ‘final cut’ which is duplicated for screening. With interactive, network based video works in contrast, many (if not all) of the decisions as to the selection, treatment and presentation of source media may be left open until the time of use, screening or performance when they are made in real time. The ‘open-ness’ of these choices is dependent on the design of the user interface and underlying functionality developed by the producer of the work.

Vidgets may be designed so as to give the user limited options, allowing them to choose only between pre-defined media at particular times. This approach may be useful in the creation of works which include narrative based elements which need to appear in a certain order or in a particular combination. Alternatively, more open systems can be designed to allow the user to specify and manipulate a range of media sources with greater flexibility. Such works may take the form of encyclopaedic collections and databases of media or more general purpose video applications. Works may also be designed to incorporate live camera and sound feeds from local or remote sources. The use of these techniques potentially allows the user to manipulate whatever audio-visual signals are being sent and may be used effectively in hybrid forms of live theatre or dance, or in installation art works.
While my own practice has centred on the development of vidgets as systems designed primarily for VJing, the same techniques of abstraction may be used in the development of a variety interactive, network based video works. Using different sources of content, programmatic functionality and user interface, systems could be developed in order to create interactive, network based works which deal with elements of traditional cinematic narrative, video art, live theatre and dance, documentaries, news, installation or hybrid forms. For example, rather than presenting visual content based on aesthetics such as rhythm and colour as a VJ using a vidget does in live performance, a ‘documentary vidget’ could be designed so as to allow the user to browse through and select content based on attributes such as interview subject, location or time. The vidget itself could provide a rhizomatic structure whereby interviews, montages and voice overs are selected, combined and sequenced programatically based on their content.

**Granularity**

In order for vidgets to function most effectively on the network and to allow the interactive manipulation of media elements, this process of abstraction requires content to be dealt with with a fine degree of granularity. By keeping these files small and separate, (rather than combining them to form one monolithic media object) they may be transmitted over the network more quickly and efficiently. By breaking content down into small, dynamically accessed units, works may be developed so as to not waste bandwidth, sourcing only the elements used. Files may be sourced from a variety of locations, loaded and combined on demand. For example, raw video footage may be split up into a collection of individual shots and sound files. Unlike traditional motion picture production which generally requires shots to be physically spliced or copied together with sounds mixed down, on the network content can be referenced directly from a variety of discrete sources. Elements of a work which are shot in London may be uploaded onto a server there, while other shots filmed in Melbourne may be stored on a server in the USA. By referencing and accessing each of these files separately they need never exist on the same machine until the time of playback. Since each file is addressed directly, shots may be easily replaced or updated without effecting other elements. Using a finely granular collection of source media, a ‘soap opera vidget’ could be designed so as allow the user to access content from a library of media which is updated episodically. This content could be filtered programatically to allow the user to follow one particular character’s path through the entire story or to dynamically trigger ‘flashbacks’ to scenes from previous episodes on demand. Such a work could be structured like a social networking web site such as *Myspace* or *Facebook*, in which the relationships between characters could be explored as a means for navigating the story.
Tagging and Metadata

As well as being broken up into manageable parts in order to be easily accessed, arranged and combined programmatically, media files require metadata. A popular method for handling this task used by several notable web services such as Flickr, and del.icio.us, tagging allows for the extensible and non-hierarchical classification of media elements. As described earlier in this document, tagging is a process by which keywords or phrases are used to annotate and classify media objects. These tags can refer to any aspect of the file, such as it’s colour, tone, duration, themes, objects or characters within the media. Rather than designing a complete hierarchical schema for categorising media before populating the categories, tagging allows users and developers to continually add new categories as they add new content. Tagged metadata can be stored within the media file itself or an external database which relates the tagged information to the object. Once a collection of source media has been broken down and tagged, systems may be developed to allow the user to select, sequence and combine these elements programmatically. For example, a vidget could be designed so as to automatically retrieve and play a collection of music tagged ‘melancholy’ or ‘joyous’ from an internet radio station web service such as Last.fm. These songs could accompany a slideshow of similarly tagged photographs or a montage of tagged video shots.

Distributed Systems, Content Management and Mash-ups

Just as the ‘content’ used in interactive, network based video works may be heterogeneous, granular and pieced together from widely distributed sources, so too may the systems themselves. Works can be created through the collection of pre-existing sub-systems and online services. Continuing with the theme of abstraction, online services may be virtually ‘plugged into’ each other in order to function together as one system. For example a ‘travel vidget’ could be designed so as to display video reports on various travel locations on a scalable, searchable world map. Rather than having to design complex online mapping software and a full content management system to create the work from scratch, the producer could configure a system whereby a number of pre-existing web services work together in order to create a hybrid work or ‘mashup’ (Merril, 2006). The online mapping capacity of Google Maps could be used in combination with YouTube’s video content management and storage system, for example, in order to rapidly develop a new work. Rather than designing the underlying systems, the job of the interactive, networked video producer in this case would be to design, configure and manage the flow of data from one system and responding to the user’s input. For example a system could be developed which would allow the user to enter the name of a city in order to bring up a map of the city, populated with icons representing location based video reports. Clicking on these icons could load the video reports into a ‘viewer’ window.
Interactive, Network Based Video Practice

On reflection, one of the most significant outcomes of this research project has been my understanding of interactive, network based video production as a practice. The nature of works produced in this emerging field may be extremely diverse, potentially ranging from tools and instruments intended for use in live performance through to educational resources, narrative and documentary works. Works may be based in a myriad of locations and mediums ranging from desktop and laptop computers to mobile devices and installations. While the works themselves may be diverse in form and function, they share a number of common production processes which define the practice. In each case, the interactive, network based video producer must frame the work conceptually in terms of its audience of users, the nature and source of its content, its interactive functions, and the intended outcomes of its use.

One of the most important decisions to be made by the producers of interactive, network based video works is to define the intended users of the work. This choice frames the scope and nature of the production. The producer must ask: ‘Is it intended for personal use? For distribution to a wider audience? For VJs? For travellers? What level of technical knowledge does the intended user have? What are the potential users’ expectations and needs?’ The answers to these and similar questions will inform the selection of platform, user interface and technology used for the work.

Two interrelated factors which must be decided upon in order to define a new work are its content and function. The interactive, network based video producer must decide on the type of content the work will deal with as well as how this material will be handled. They must ask questions such as: ‘Will the content be specially produced or generated, re-mediated from existing sources, provided by the user or served by external systems? How will the work allow the user to access and manipulate the source material? How frequent will user interactions be and what effects will they have?’ Again, based on the answers, the interactive, network based video producer will be able to select appropriate technologies and approaches to the work.

As well as defining the work’s intended users, content and functionality, the producer must decide on the intended outcomes and uses of the work. In order to define these qualities the producer must ask questions such as: ‘Will it be used in live performance? As part of a theatre show? As an educational resource? As entertainment? As a storytelling device? Will the work be re-used or is it intended for single use? What would characterise a successful use of the work?’ The answers to these types of questions will help refine the shape of the work.

Once the work has been framed through this series of defining choices, the producer may assess the technical and functional requirements of the system. This process involves the critical assessment of various platforms, formats and programming languages in terms of their strengths and limitations for realising the aims of the production. Pre-existing web services, APIs (Application Programming Interfaces), scripts and classes may be evaluated and tested for functionality and inter-operability as works are developed and configured.
Future Directions

I am currently working on the production of a number new audio-visual performance works which utilise various real time video processing techniques from my ‘performance environment’ works such as White Noise and Gadget. These new works have an emphasis on the incorporation of live physical performance as well as the use of everyday network based technologies. These works continue to explore an improvisational ‘musical’ approach to performance and development with the addition of more representational audio-visual elements.

I am also developing a series of experimental templates and plug-ins which allow users to present and browse through video on blogs in new ways. Rather than locating video objects in a primarily text based blogging system, these works allow the user to explore links between fragments of video in context visually. Using WordPress (a popular blogging content management system) as a base these works will continue my exploration of tagging as a non-hierarchical means for categorising content. The experiments may result in a number of different outcomes ranging from systems for organising and presenting documentary content online, through to the development of more poetic experiments, presenting multiple video elements on screen side by side.

The ‘art’ of interactive, network based video production lies in the negotiation and incorporation of technological and art systems, theory and practice. As Burnham writes: “Scientists and technicians are not converted into artists, rather the artist becomes a symptom of the schism between art and technics. Progressively, the need to make ultrasensitive judgments as to the use of technology and scientific information becomes ‘art’ in the most literal sense.” (1968: 30). As I have made these ‘judgements’ over the course of the research project I have found that the cycle of action research has been particularly suited to working in this field. Moving on from this project, my own interactive, network based video research and practice continues in new directions, continuing my reflective practice.
References


Appendix 1
WORKS
**Name**  
Vog 001

**Date**  
December 2003

**Description**  
A loop of pre-recorded analogue video feedback plays endlessly with an electronic drum pattern as accompaniment. The movement of the user’s cursor from left to right and back manipulates the playback speed and direction of the loop in real time, playing at either double speed forward or half speed backwards.

**Aims**  
To investigate interactive QuickTime’s capacity for real time user interaction via mouse movement, QScript and LiveStage Pro as an authoring environment and the networked distribution of interactive media.

**Outcomes and Implications**  
I found the QuickTime environment to handle playback control based on mouse movement very effectively. The Qscript scripting language was difficult to master at first, however it proved to be a quite capable development environment for use in subsequent works.

**Production Software**  
LiveStage Pro, Live, Final Cut Pro

**Runtime Environment**  
QuickTime Player

**Remote Data Services**  
Vog 001 was designed for networked distribution via embedding in a web page. The work loads its video content from a separate remote file.

**User Input**  
mouse movement

**Related Works (Subsequent)**  
Brick Style, Image pan series, Quartz Kaleidoscope, White Noise

**Blog Post**  
http://dpwolf.net/blog/2003/12/vog001/
**Name**
Brick Style

**Date**
March 2004

**Description**
A simple paddle based game.

**Aims**
To further explore the QuickTime architecture’s capacity for dealing with real time user input.
To investigate the programmatic animation of sprites on screen.

**Outcomes and Implications**
QuickTime handled the real time animation of on screen sprites, collision detection and user input via mouse movement well. The sprite handling scripts developed through the production of this work were useful reference points for the development of more complex user interfaces in subsequent works.

**Production Software**
LiveStage Pro, Adobe Photoshop

**Runtime Environment**
QuickTime Player

**User Input**
mouse movement, clickable objects

**Related Works (Previous)**
Vog 001

**Related Works (Subsequent)**
QuickTime Flickr Photo Viewer, Vidget 3

**Blog Post**
http://dpwolf.net/blog/2004/03/brick-style/
Click me to send

BANGBANGBANGBANGBANG
ANGANGANGANGANGANGANG
Name
Inter-movie Text Communication

Date
April 2004

Description
When clicked, one movie sends a text string to a text track in another movie.

Aims
A proof of concept work exploring QuickTime’s capacity for sending information between two separate movies.

Outcomes and Implications
Using inter-movie communication skills developed through the production of this work I was able to create more complex works composed of separate ‘controller’ and ‘output’ movies.

Production Software
LiveStage Pro

Runtime Environment
QuickTime Player

User Input
mouse clicks

Related Works (Subsequent)
Random Site, Text to String, Vidget 1

Blog Post
http://dpwolf.net/blog/2004/04/inter-movie-text-communication/
Name
Text to String

Date
May 2004

Description
A proof of concept work exploring QuickTime’s handling of real time text input. A script copies text from a user-editable text field and repeatedly adds it to another field, resulting in an abstract flow of characters flowing across and down the screen.

Aims
A proof of concept work exploring QuickTime’s capacity for handling text input.

Outcomes and Implications
This test proved highly successful and led to the use of text input in several subsequent works, particularly those dealing with search functionality.

Production Software
LiveStage Pro

Runtime Environment
QuickTime Player

User Input
text entry via text fields

Related Works (Previous)
Inter-movie Text Communication

Related Works (Subsequent)
QuickTime Flickr Photo Viewer, Vidget 3, Vidget 3.5

Blog Post
http://dpwolf.net/blog/2004/05/text-to-string/
**Name**
Remote Control 0.2

**Date**
May 2004

**Description**
A sketch towards the development of a QuickTime based VJ performance tool.

**Aims**
A proof of concept work exploring QuickTime’s capacity for dealing with different blend modes.

**Outcomes and Implications**
This test proved highly successful and led to the use of blend mode manipulation techniques in several VJ orientated works.

**Production Software**
Macromedia Fireworks, LiveStage Pro, Adobe Photoshop, Final Cut Pro

**Runtime Environment**
QuickTime Player

**Related Works (Subsequent)**
Vidget 1, Vidget 3, Vidget 3.5

**Blog Post**
http://dpwolf.net/blog/2004/05/remote-control-02/
<table>
<thead>
<tr>
<th>Random Site Linker</th>
<th>Randomise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Radiohead TV</td>
<td></td>
</tr>
<tr>
<td>URL: <a href="http://www.radiohead.tv/">http://www.radiohead.tv/</a></td>
<td></td>
</tr>
</tbody>
</table>
Name
Random Site

Date
June 2004

Description
A proof of concept work which explores QuickTime’s capacity to read and parse external XML data files. A movie dynamically loads data from an XML file containing a list of websites and their URLs, chooses and displays one at random.

Production Software
LiveStage Pro

Runtime Environment
QuickTime Player

Local Data Sources
A separate XML file, hosted locally or on a web server.

User Input
mouse clicks, data entry via hand coded XML

Related Works (Previous)
Inter-movie Text Communication

Related Works (Subsequent)
QuickTime Blogroll, Vidget 3

Blog Post
http://dpwolf.net/blog/2004/06/random-site-an-xml-reading-quicktime-linker-movie/
Name: Jonah Brucker Cohen : coin-operated.com
URL: http://www.coin-operated.com/
Info: Very cool network artist / researcher
**Name**  
QuickTime Blogroll

**Date**  
June 2004

**Description**  
A proof of concept work which extends on Random Site’s ability to read a custom written XML file, parsing a dynamically generated RSS feed from online web service blogrolling.com

**Production Software**  
LiveStage Pro, Adobe Photoshop

**Runtime Environment**  
QuickTime Player

**Remote Data Services**  
Blogrolling RSS feed

**User Input**  
mouse clicks, data entry via 3rd party web services

**Related Works (Previous)**  
Random Site

**Related Works (Subsequent)**  
QuickTime Flickr Photo Viewer

**Blog Post**  
http://dpwolf.net/blog/2004/06/quicktime-blogroll/
<table>
<thead>
<tr>
<th>Channel</th>
<th>Clip Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>strobecircular2</td>
<td>(3.1 MB)</td>
</tr>
<tr>
<td></td>
<td>strobecircular</td>
<td>(2.4 MB)</td>
</tr>
<tr>
<td></td>
<td>steam</td>
<td>(1 MB)</td>
</tr>
<tr>
<td></td>
<td>snow</td>
<td>(6.1 MB)</td>
</tr>
<tr>
<td></td>
<td>redmozbright</td>
<td>(516 KB)</td>
</tr>
<tr>
<td></td>
<td>redmoz</td>
<td>(876 KB)</td>
</tr>
<tr>
<td></td>
<td>rainfill</td>
<td>(1.2 MB)</td>
</tr>
<tr>
<td></td>
<td>pycore</td>
<td>(2.8 MB)</td>
</tr>
<tr>
<td></td>
<td>invhorizonticks</td>
<td>(972 KB)</td>
</tr>
<tr>
<td></td>
<td>invdorzn</td>
<td>(900 KB)</td>
</tr>
<tr>
<td></td>
<td>greenscreenroll</td>
<td>(5.7 MB)</td>
</tr>
<tr>
<td></td>
<td>greenmozaic2</td>
<td>(2.5 MB)</td>
</tr>
<tr>
<td></td>
<td>greenmozaic</td>
<td>(4 MB)</td>
</tr>
<tr>
<td></td>
<td>greencircular</td>
<td>(2.2 MB)</td>
</tr>
<tr>
<td></td>
<td>greenblur</td>
<td>(1 MB)</td>
</tr>
<tr>
<td></td>
<td>citylightsinclassic</td>
<td>(1 MB)</td>
</tr>
<tr>
<td></td>
<td>citylightsinwack</td>
<td>(1.7 MB)</td>
</tr>
<tr>
<td></td>
<td>dripsonblack</td>
<td>(1.1 MB)</td>
</tr>
<tr>
<td></td>
<td>citylightsshakefstr</td>
<td>(1.1 MB)</td>
</tr>
<tr>
<td></td>
<td>citylightsshakeblur</td>
<td>(684 MB)</td>
</tr>
<tr>
<td></td>
<td>citylightsshakeblurblur</td>
<td>(476 MB)</td>
</tr>
</tbody>
</table>

**Other Channels:***

<table>
<thead>
<tr>
<th>Channel</th>
<th>Clip Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Do it!*
**Name**
Vidget 1

**Date**
July 2004

**Description**
An experimental QuickTime based VJ performance tool.

**Aims**
An early sketch of a VJ application for use in live performance.

**Outcomes and Implications**
While successfully bringing together several techniques into the creation of a performance tool, the basic user interface was in need of major improvement.

**Production Software**
LiveStage Pro, Final Cut Pro

**Runtime Environment**
QuickTime Player

**Related Works (Previous)**
Remote Control 0.2, Inter-movie Text Communication

**Related Works (Subsequent)**
QuickTime Flickr Photo Viewer, Vidget 3

**Related Performances**
Segmentation Fault 4

**Blog Post**
http://dpwolf.net/blog/2004/07/vidget-1-an-interactive-networked-vj-application-for-quicktime/
Name
QuickTime Flickr Photo Viewer

Date
September 2004

Description
A network aware photo browser. This work allows the user to enter a search word to browse for tagged images from photo sharing service Flickr.

Aims
A proof of concept work implementing the parsing of RSS feeds as a means for accessing media files.

Outcomes and Implications
This work was highly effective not only technically, but as a means for demonstrating the concept of the network based video work. It led to the incorporation of similar search functions into several VJ tools which followed.

Production Software
LiveStage Pro

Runtime Environment
QuickTime Player

Remote Data Services
Flickr RSS (API)

User Input
text entry via text fields, data entry via 3rd party web services, clickable objects

Related Works (Previous)
Vidget 1, QuickTime Blogroll, Brick Style

Related Works (Subsequent)
Electrofringe 2005 3D/Feedback/Flickr/MIDI setup, Vidget 3, xmlarticle.php

Blog Post
http://dpwolf.net/blog/2004/09/quicktime-flickr-photo-viewer/
**Name**
Inspiration

**Date**
August 2004

**Description**
An experimental internet radio project produced with Hannah Miller and Kate Eccles.

**Production Software**
QuickTime Player, LiveStage Pro, Final Cut Pro, Adobe Photoshop

**Runtime Environment**
QuickTime Player

**Related Works (Previous)**
Vidget 3

**Blog Post**
http://dpwolf.net/blog/2004/11/inspiration/
Name
Vidget 3

Date
September 2004

Description
QuickTime based VJ performance tool with thumbnail based graphical user interface and Flickr photo search functionality

Production Software
Macromedia Fireworks, LiveStage Pro

Runtime Environment
QuickTime Player

Remote Data Services
Flickr

Local Data Sources
All movies and still images are externally referenced and dynamically loaded based on a separate XML file

User Input
text entry via text fields, real time video input (via external file), drag and drop sprites, data entry via hand coded XML, data entry via 3rd party web services

Related Works (Previous)
Vidget 1, Random Site, QuickTime Flickr Photo Viewer, Brick Style

Related Works (Subsequent)
Inspiration, Vidget 3.5

Related Performances
...Frame Rate, This is Not Art 2004, Segmentation Fault (2nd November 2004), Plug n Play (September 2004)

Blog Post
http://dpwolf.net/blog/2004/09/vidget-3/
Name
Vidget 3.5

Date
December 2004

Description
An updated version of Vidget 3. A network aware VJ performance tool

Production Software
Macromedia Fireworks, LiveStage Pro

Runtime Environment
QuickTime Player

Remote Data Services
Flickr

User Input
text entry via text fields, real time video input (via external file), drag and drop sprites, data entry via hand coded XML, data entry via 3rd party web services

Related Works (Previous)
Vidget 3

Related Performances
Straight Out of Brisbane, Electrofringe 2004, Digital Fringe Festival Mobile Projection Unit

Blog Post
http://dpwolf.net/blog/2004/12/vidget-35/
**Name**  
Quartz Kaleidoscope

**Date**  
May 2005

**Description**  
An early experiment testing Quartz Composer’s capacity for real time programming, manipulation of images and dynamic loading of video files from remote sources.

**Aims**  
Investigate Quartz Composer as a development environment for real time interactive, network aware video. Specifically, to test the use of remote video sources, the new real time video effects and user input. To compare and contrast with the authoring of interactive QuickTime movies with LiveStage Pro.

**Outcomes and Implications**  
Found that whilst possible, the playback of remotely sourced video through Quartz Composer was poor as it had no provision for waiting until the complete file had finished downloading before playback. While this was a limitation, the capacity for real time programming, complex visual effects and new methods for user input inspired the further exploration of Quartz Composer as an environment.

**Production Software**  
QuickTime Player, Quartz Composer

**Runtime Environment**  
QuickTime Player, Quartz Composer, MacOS 10.4 or Later

**Remote Data Services**  
A QuickTime Movie hosted remotely on RMIT web server is accessed via a local QuickTime reference movie.

**Local Data Sources**  
QuickTime reference movie.

**User Input**  
real time audio level input, live coding / patching

**Related Works (Previous)**  
Vog 001

**Related Works (Subsequent)**  
3D Particle Screen Saver
**Name**
Bikelights: a Quartz Vidget

**Date**
June 2005

**Production Software**
Xcode, Quartz Composer, Interface Builder

**Runtime Environment**
MacOS 10.4 or Later

**User Input**
text entry via text fields, real time video input (via external file), real time audio level input, on screen sliders, on screen check boxes, drag and drop files from MacOS Finder

**Related Works (Subsequent)**
MIAF 2 LA 6: a Quartz based Vidget

**Blog Post**
http://dpwolf.net/blog/2005/06/bikelights-a-quartz-widget/
Name
MIAF 2 LA 6: a Quartz based Vidget

Date
June 2005

Description
A stand alone application designed for VJ performance which allows the user to select manipulate and present their own video content in real time.

Production Software
Xcode, Quartz Composer, Interface Builder

Runtime Environment
MacOS 10.4 or Later

Local Data Sources
All video and still image content is dynamically loaded from local storage devices.

User Input
text entry via text fields, real time video input (via external file), real time audio level input, popup colour selector, on screen sliders, on screen check boxes, drag and drop files from MacOS Finder, automatic scanning of images in a folder

Related Works (Previous)
Bikelights: a Quartz Vidget

Related Works (Subsequent)
Electrofringe 2005 3D/Feedback/Flickr/MIDI setup, Gadget Performance Environment

Related Performances
Share (New York), Melbourne International Animation Festival presents: Remains To Be Seen, Melbourne Fringe Festival / Kiss My After Effects, Liquid Architecture 6, Fly by Wire, Distorted Festival, 5ive 6ix (September 2006), 5ive 6ix

Blog Post
http://dpwolf.net/blog/2005/07/miaf-la-6-a-quartz-vidget/
Name
3D Object Plugin Experiments

Date
August 2005

Production Software
Quartz Composer, Custom Quartz Composer Plugins from quartzcomposer.jp

Runtime Environment
QuickTime Player, Quartz Composer

Related Works (Subsequent)
3D Cube Inertia Series, Electrofringe 2005 3D/Feedback/Flickr/MIDI setup

Blog Post
http://dpwolf.net/blog/2005/08/3d-objects-in-quartz-composer/
Monkey!
**Name**
Experiments with mDimension’s Quartz Composer plugin for Safari

**Date**
August 2005

**Description**
Embedding Quartz Compositions in web pages, sending and receiving variables via JavaScript

**Production Software**
SubEthaEdit, Quartz Composer

**Runtime Environment**
Safari, mDimension Quartz Composer plugin for Safari

**Remote Data Services**
Flickr

**Blog Post**
**Name**  
White Noise

**Date**  
September 2005

**Description**  
Analogue / Digital video feedback system with real time midi controls.

**Production Software**  
Quartz Composer

**Runtime Environment**  
Quartz Composer

**User Input**  
real time video input (direct), MIDI, live coding / patching

**Related Works (Previous)**  
Vog 001

**Related Works (Subsequent)**  
Electrofringe 2005 3D/Feedback/Flickr/MIDI setup, Gadget Saver, TBA Performance Environment

**Related Performances**  
Simulus @ Plug n Play, Electrofringe 2005 (with Doktorb Robotnik), Canvas City, Bits of Clay & David Sevo with dpwolf @ Glitch

**Blog Post**  
**Name**  
3D Particle Screen Saver

**Date**  
October 2005

**Description**  
A generative screen saver featuring coloured circular particles rotating in a 3D environment and reacting to audio input in real time.

**Production Software**  
Quartz Composer, Macromedia Fireworks

**Runtime Environment**  
QuickTime Player, Quartz Composer, MacOS 10.4 or Later

**User Input**  
real time audio level input

**Related Works (Previous)**  
Quartz Kaleidoscope

**Related Works (Subsequent)**  
3D Cube Inertia Series, Atmospheric Image Sequence Series, Gadget Saver

**Blog Post**  
http://dpwolf.net/blog/2005/10/3d-particle-screen-saver/
ElectroFringe
NEW MEDIA ARTS FESTIVAL 2005
Name
Electrofringe 2005 3D/Feedback/Flickr/MIDI setup

Date
October 2005

Production Software
SubEthaEdit, Quartz Composer, PHP

Runtime Environment
Quartz Composer, PHP

Remote Data Services
Flickr API, custom PHP script generating RSS feeds

User Input
text entry via text fields, real time audio level input, MIDI, live coding / patching, data entry via 3rd party web services

Related Works (Previous)
White Noise, QuickTime Flickr Photo Viewer, MIAF 2 LA 6: a Quartz based Vidget, 3D Object Plugin Experiments

Related Works (Subsequent)
Atmospheric Image Sequence Series, Painterly Ultra-Widescreen Series, xmlarticle.php

Related Performances
Electrofringe 2005 (with Simon Gorman)
Name
xmlarticle.php

Date
October 2005

Description
A php script which queries the Flickr API and returns a custom built RSS feed of images based on a tag for use in Quartz Compositions.

Production Software
SubEthaEdit, PHP

Runtime Environment
PHP

User Input
data entry via 3rd party web services

Related Works (Previous)
QuickTime Flickr Photo Viewer, Electrofringe 2005 3D/Feedback/Flickr/MIDI setup

Related Works (Subsequent)
Atmospheric Image Sequence Series, Painterly Ultra-Widescreen Series

Related Performances
Electrofringe 2005 (with Simon Gorman)
**Name**
Boxed

**Date**
November 2005

**Description**
A video portrait designed for playback as part of a theatrical performance.

**Production Software**
QuickTime Player, Quartz Composer, Final Cut Pro

**Runtime Environment**
QuickTime Player

**Related Performances**
Boxed @ Short & Sweet

**Blog Post**
http://dpwolf.net/blog/2005/11/boxed-short-sweet/
STEVE LAW (Melb), DAVE WOLF (Melb)
KIMMO JENVONEN, SOMAYA LANGLEY
WARWICK LYNCH & ALEX THOROGOOD
A BI-MONTHLY NEW MEDIA EVENT BEGINNING DEC 7th, $5 AT THE DOOR

GADGET
EXPERIMENTAL SOUND & VISION PRESENTED BY MONOPOD
7pm, DEC 7th @ THE AUSTRALIAN CHOREOGRAPHIC CENTRE
GORMAN HOUSE ARTS CENTRE
Enquiries: 0418562540
**Name**  
Gadget Performance Environment

**Date**  
December 2005

**Description**  
A video performance tool, custom built for a specific performance featuring on screen controls, MIDI controls and video accumulation effects.

**Production Software**  
Xcode, Quartz Composer, Interface Builder

**Runtime Environment**  
MacOS 10.4 or Later

**User Input**  
real time video input (direct), on screen sliders, on screen check boxes, MIDI

**Related Works (Previous)**  
MIAF 2 LA 6: a Quartz based Vidget, Gadget Saver

**Related Works (Subsequent)**  
3D Cube Inertia Series, Atmospheric Image Sequence Series, Painterly Ultra-Widescreen Series, TBA Performance Environment

**Related Performances**  
Gadget (with Somaya Langley)
This macro patch reads its subpatches into an image.

Render in Image basically creates a new rendering destination in the form of an Image. The subpatches of this macro patch then render to that new destination instead of the original one. The resulting image can be used as an 'regular' image, for example, it can be passed to Core Image filters or used as a mask.

The dimensions of the image can be set in the asset's settings. If set to 0/0, the image will be as large as the bounding box of all subpatches together.
Name
Gadget Saver

Date
December 2005

Description
A revised version of 3D Particle Screen Saver with real time MIDI controls.

Production Software
Quartz Composer, Macromedia Fireworks

Runtime Environment
QuickTime Player, Quartz Composer

User Input
MIDI, live coding / patching

Related Works (Previous)
White Noise, 3D Particle Screen Saver

Related Works (Subsequent)
Gadget Performance Environment

Related Performances
Gadget (with Somaya Langley)
**Name**
Atmospheric Image Sequence Series

**Date**
March 2006

**Description**
A series of painterly generative video works which cycled through a series of still images displayed on a multitude of 2D surfaces in 3D space with tracer effects.

**Production Software**
Quartz Composer

**Runtime Environment**
QuickTime Player, Quartz Composer, Dataton Watchout

**User Input**
live coding / patching, automatic scanning of images in a folder

**Related Works (Previous)**
xmlarticle.php, Gadget Performance Environment, Electrofringe 2005 3D/Feedback/Flickr/MIDI setup, 3D Particle Screen Saver

**Related Works (Subsequent)**
Chinese Fire Dragon Parts I and II (with Simon Gorman), TBA Performance Environment

**Related Performances**
Plug n Play (September 2006), Fly by Wire, Similar & dpwolf & Horse Bazaar
Name
Painterly Ultra-Widescreen Series

Date
March 2006

Description
A series of painterly generative video works designed for projection across six seemed projectors at Melbourne bar Horse Bazaar.

Production Software
QuickTime Player, Quartz Composer

Runtime Environment
QuickTime Player, Quartz Composer, Dataton Watchout

User Input
live coding / patching

Related Works (Previous)
xmlarticle.php, Gadget Performance Environment, Electrofringe 2005 3D/Feedback/Flickr/MIDI setup

Related Works (Subsequent)
Chinese Fire Dragon Parts I and II (with Simon Gorman), TBA Performance Environment

Related Performances
Plug n Play (September 2006), Fly by Wire, Similar & dpwolf & Horse Bazaar
**Name**
Bluetooth Mobile TV Remote

**Date**
June 2006

**Description**
A simple remote control video switching system designed to use a mobile phone to select between video clips via Bluetooth.

**Production Software**
Quartz Composer, Bluetooth Remote Control

**Runtime Environment**
Sony Ericsson Mobile Phones with Bluetooth HID capabilities, Quartz Composer

**User Input**
keystrokes (Bluetooth mobile phone)

**Related Performances**
Electundra 2006
Name
3D Cube Inertia Series

Date
August 2006

Description
Experiments exploring the movement of a cube through 3D space.

Aims
To explore the use of JavaScript to smooth out rapidly changing user input data.

Outcomes and Implications
I found that I could create an ‘inertia’ effect which was particularly suited to smoothing changes in 3D rotation and positioning. This led to further experiments with the transformation of 3D particle systems via user input and JavaScript.

Production Software
Quartz Composer

Runtime Environment
Quartz Composer

User Input
live coding / patching, keystrokes (computer keyboard)

Related Works (Previous)
Gadget Performance Environment, 3D Particle Screen Saver, 3D Object Plugin Experiments

Related Works (Subsequent)
TBA Performance Environment
**Name**  
Image pan series

**Date**  
September 2006

**Description**  
Real time animation, randomly panning and scanning across high resolution still images. For use in live video performance.

**Production Software**  
Quartz Composer, Adobe Lightroom

**User Input**  
mouse movement, mouse clicks, live coding / patching

**Related Works (Previous)**  
Vog 001

**Related Performances**  
Plug n Play (September 2006), 5ive 6ix (September 2006)
Name
TBA Performance Environment

Date
October 2006

Description
A revised version of the Gadget Performance Environment work designed to use MIDI and computer keyboard controls and to work around software limitations imposed by a security update.

Production Software
Quartz Composer, Final Cut Pro

Runtime Environment
Quartz Composer

User Input
on screen sliders, MIDI, live coding / patching, keystrokes (computer keyboard)

Related Works (Previous)
White Noise, Painterly Ultra-Widescreen Series, Gadget Performance Environment, Atmospheric Image Sequence Series, 3D Cube Inertia Series

Related Performances
Share (New York), Electrofringe 2006 (with Somaya Langley), Digital Fringe Festival Mobile Projection Unit
Appendix 2
PERFORMANCES
<table>
<thead>
<tr>
<th>Year</th>
<th>Name</th>
<th>Date</th>
<th>Venue</th>
<th>Location</th>
<th>Description</th>
<th>Web Link</th>
<th>Works Used</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>Segmentation Fault</td>
<td>11 November 2003</td>
<td>Loop</td>
<td>23 Meyers Place, Melbourne, Victoria, Australia</td>
<td></td>
<td>Vidget 3</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>Segmentation Fault 2.0</td>
<td>20 January 2004</td>
<td>Loop</td>
<td>23 Meyers Place, Melbourne, Victoria, Australia</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>Segmentation Fault (3)</td>
<td>6 April 2004</td>
<td>Loop</td>
<td>23 Meyers Place, Melbourne, Victoria, Australia</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>Electundra 2004</td>
<td>June 2004</td>
<td>Loop</td>
<td>23 Meyers Place, Melbourne, Victoria, Australia</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>Segmentation Fault 4</td>
<td>6 July 2004</td>
<td>Loop</td>
<td>23 Meyers Place, Melbourne, Victoria, Australia</td>
<td></td>
<td>Vidget 1</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>Plug n Play (September 2004)</td>
<td>2 September 2004</td>
<td>Kent St Cafe</td>
<td>201 Smith St, Fitzroy, Victoria, Australia</td>
<td></td>
<td>Vidget 3</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>Segmentation Fault n+1</td>
<td>21 September 2004</td>
<td>Loop</td>
<td>23 Meyers Place, Melbourne, Victoria, Australia</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>Electrofringe 2004</td>
<td>October 2004</td>
<td>Cambridge Hotel</td>
<td>789 Hunter St, Newcastle West, NSW, Australia</td>
<td>Various VJ performances including collaborations with Tim Webster</td>
<td>Vidget 3.5</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>This is Not Art 2004</td>
<td>October 2004</td>
<td>Cambridge Hotel</td>
<td>789 Hunter St, Newcastle West, NSW, Australia</td>
<td></td>
<td>Vidget 3</td>
<td></td>
</tr>
<tr>
<td>Name:</td>
<td>Date:</td>
<td>Venue:</td>
<td>Location:</td>
<td>Works Used:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>----------------</td>
<td>-----------------------------------</td>
<td>-------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Segmentation Fault</td>
<td>2 November 2004</td>
<td>Loop</td>
<td>23 Meyers Place, Melbourne, Victoria, Australia</td>
<td>Vidget 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...Frame Rate</td>
<td>15 November 2004</td>
<td>Loop</td>
<td>23 Meyers Place, Melbourne, Victoria, Australia</td>
<td>Vidget 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Straight Out of Brisbane</td>
<td>December 2004</td>
<td>Straight Out Of Brisbane Festival Club</td>
<td>Fortitude Valley, Queensland, Australia</td>
<td>Vidget 3.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electundra 2005</td>
<td>June 2005</td>
<td>Duckboard House</td>
<td>91 Flinders Lane, Melbourne, Victoria, Australia</td>
<td>MIAF 2 LA 6: a Quartz based Vidget</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Melbourne International Animation Festival presents: Remains To Be Seen</td>
<td>25 June 2005</td>
<td>Duckboard House</td>
<td>91 Flinders Lane, Melbourne, Victoria, Australia</td>
<td>MIAF 2 LA 6: a Quartz based Vidget</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electrofringe 2005 (with Doktorb Robotnik)</td>
<td>1 October 2005</td>
<td>Newcastle TAFE Gallery</td>
<td>590-608 Hunter St, Newcastle, NSW, Australia</td>
<td>White Noise</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Date</td>
<td>Venue</td>
<td>Location</td>
<td>Works Used</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>------------------------</td>
<td>--------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 October 2005</td>
<td>Octapod</td>
<td>3/231 King Street, Newcastle, NSW, Australia</td>
<td>xmlarticle.php, Electrofringe 2005 3D/Feedback/Flickr/MIDI setup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 December 2005</td>
<td>Brown Alley</td>
<td>Brown Alley, Melbourne, Victoria, Australia</td>
<td>MIAF 2 LA 6: a Quartz based Vidget</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 February 2006</td>
<td>Corner Hotel</td>
<td>57 Swan St, Richmond, Victoria, Australia</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 March 2006</td>
<td>Kent St Cafe</td>
<td>201 Smith St, Fitzroy, Victoria, Australia</td>
<td>White Noise</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 March 2006</td>
<td>Horse Bazaar</td>
<td>397 Little Lonsdale St, Melbourne, Victoria, Australia</td>
<td>Painterly Ultra-Widescreen Series, Atmospheric Image Sequence Series</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18 March 2006</td>
<td>Loop</td>
<td>23 Meyers Place, Melbourne, Victoria, Australia</td>
<td>MIAF 2 LA 6: a Quartz based Vidget</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name: Electundra 2006</td>
<td>Works Used: Chinese Fire Dragon Parts I and II (with Simon Gorman), Bluetooth Mobile TV Remote</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date: June 2006</td>
<td>Venue: Loop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Location: 23 Meyers Place, Melbourne, Victoria, Australia</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: 5ive 6ix (September 2006)</th>
<th>Works Used: MIAF 2 LA 6: a Quartz based Vidget, Image pan series</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date: September 2006</td>
<td>Venue: Loop</td>
</tr>
<tr>
<td>Location: 23 Meyers Place, Melbourne, Victoria, Australia</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Date: September 2006</td>
<td>Venue: Kent St Cafe</td>
</tr>
<tr>
<td>Location: 201 Smith St, Fitzroy, Victoria, Australia</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: Electrofringe 2006 (with Somaya Langley)</th>
<th>Works Used: TBA Performance Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date: 1 October 2006</td>
<td>Venue: Newcastle TAFE Gallery</td>
</tr>
<tr>
<td>Location: 590-608 Hunter St, Newcastle, NSW, Australia</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: Digital Fringe Festival Mobile Projection Unit</th>
<th>Works Used: Vidget 3.5, TBA Performance Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date: 7 October 2006</td>
<td>Venue: Various streets around Melbourne</td>
</tr>
<tr>
<td>Location: Melbourne, Victoria, Australia</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name: Share (New York)</th>
<th>Works Used: TBA Performance Environment, MIAF 2 LA 6: a Quartz based Vidget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date: 4 March 2007</td>
<td>Venue: Reboot</td>
</tr>
<tr>
<td>Location: 37 Avenue A, New York, NY 10009</td>
<td></td>
</tr>
</tbody>
</table>

2007