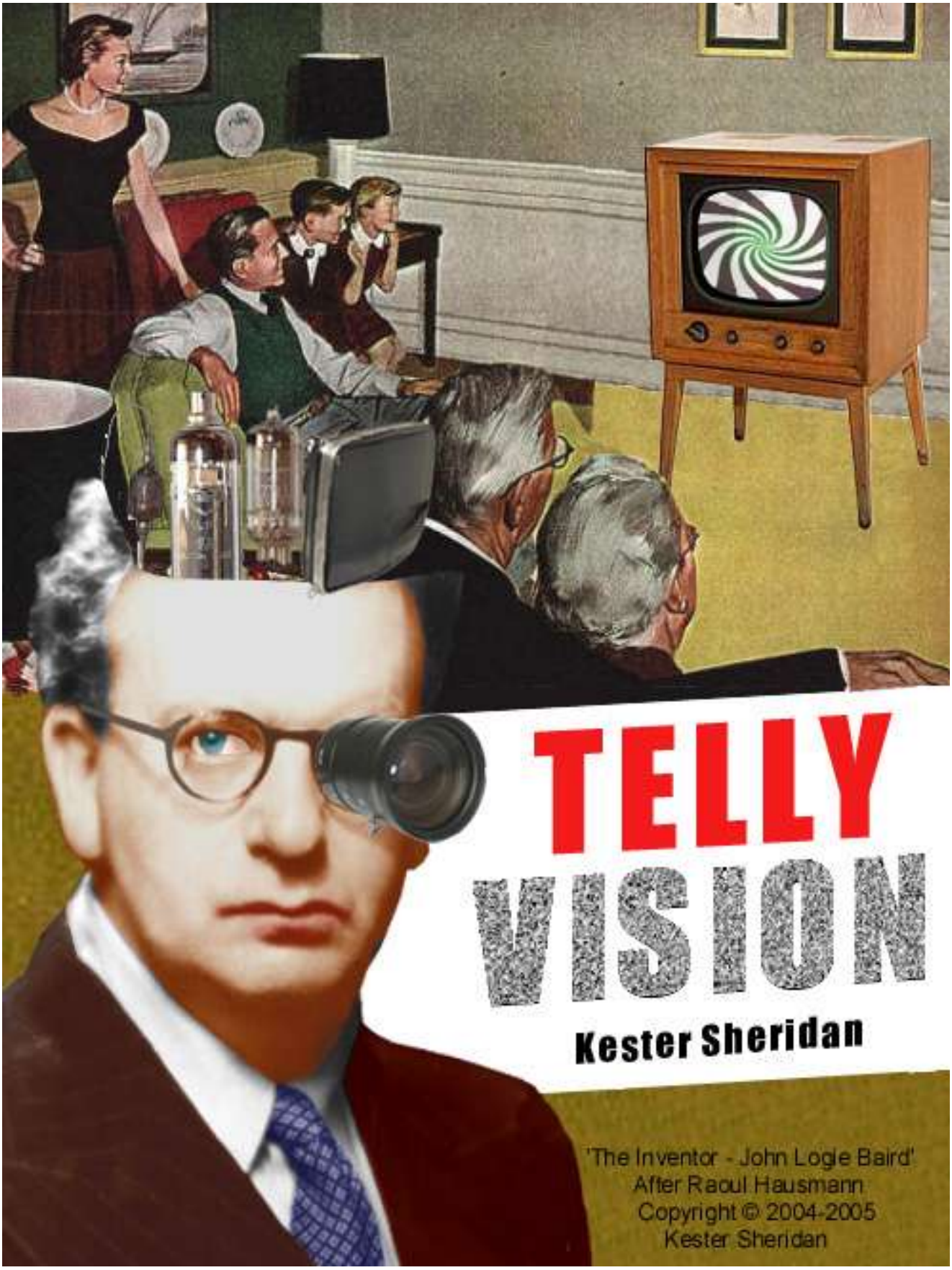
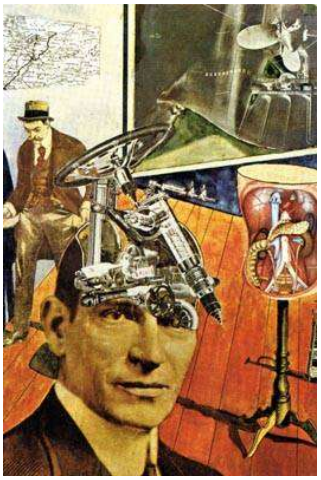
Report

Goldsmiths
UNIVERSITY
OF LONDON



Abstract

There is a growing trend in technology for convergence. Modern devices such as mobile phones do not have just one function but many - combining the functionality of a telephone, a camera and an mp3 player all into one device. This trend can also been seen in home entertainment with the convergence of the computer and television into one single entertainment centre. For me this convergence of the television set and the computer provides an interesting opportunity for the artist to truly 'interrupt live television', allowing us to apply the techniques developed by video artists to the live medium of television and so interrupt the expectations of the spectator for a medium that is taken so much for granted. This report details the design and development of the 'Telly Vision' application which provides the tools for me as an artist to create such art work which challenges the spectator's view of the medium. The report also includes the design and development of the 'Media Montage' application in which I wanted to explore further some of the concepts I had developed in the 'Telly Vision' project.



0.0.0.1About Front Page Image

The image on the front page is an image I have created after Raoul Hausmann's photomontage entitled 'Tatlin At Home' (1920). John Logie Baird (1888 – 1946) was one of the pioneers of the television, transmitting the world's first moving image on 30th October 1925[1] and so I thought it was only appropriate to have parts of a television; a series of valves and cathode ray tubes making up part of his head like a cyborg. In the background a family can be seen watching television and although from the 1950's the scene could be taken from any era afterwards including modern day. On the television there is a hypnotic pattern, as television seems to have that effect on people who are 'glued' to watching it for many hours each night.

[1] http://en.wikipedia.org/wiki/John_Logie_Baird John Logie Baird entry in Wikipedia, the free encyclopedia

Table of Contents

1.Introduction

1.1Art and Technology

Art is based on technology or at the very least technology has held a fascination for the artist through the centuries and allowed the artist to expand their horizons in terms of their art. Indeed art history can be seen through the innovation of technology from the invention of oil paints, pigment ground into a medium of oil usually linseed oil in the 15th Century; the camera obscura in the 16th Century – an optical device allowing an artist to accurately trace a scene or image onto a canvas in a darkened room; the invention of the collapsible zinc paint tube by John G. Rand in 1841[2] so allowing artists like the Impressionist to paint outside (plein air); through to the invention of photography and film. Thereby it is the logical progression of art to harness the technology of the time and so logically computers are now being used for this purpose. If one takes the Royal Academy Summer Exhibition as a barometer of the art world today then computer art is moving into the mainstream with some prominent artists such as Michael Craig-Martin now showing works (see figure 1.1) that are created and exhibited on computers. This is opposed to many other works such as digital prints that although they have been developed on computers are displayed by more conventional means as digital prints on paper, which has been the norm in recent times.

Figure 1.1: 'Becoming', Michael Craig-Martin (2001)
Computer and 19" LCD unit, vector drawings and bespoke software
Exhibited at Royal Academy Summer Exhibition 2005

With this in mind, my intentions were to try and create an artwork that would not just fit into the category of 'computer art' but be able to be seen in a wider context. For this purpose it would need to be immediate and striking enough so that the fact it was

[2] http://www.humanitiesweb.org/human.php?s=g&p=a&a=i&ID=980

either on a computer or made with a computer was irrelevant to the layperson seeing it in a gallery environment. As the computer, for all of its wider acceptance as a tool, for me computer art is still seen as the poor cousin to more conventional and traditional forms of art such as painting and photography. I decided to try and create a computer-based artwork using television as its core. I would try and manipulate the television image by adding text and images over the top. This would be impossible to accomplish without using a computer but still give the opportunity to disguise the fact that it was even on a computer at all.

1.2Brief Background

Technology is converging. Mobile phones are converging with cameras and audio players. The new 'iPod' boasts not only being able to play music but also being able to show music videos as well, just as Microsoft's new X-Box 360 games console boasts of being an entertainment centre. Whilst in the home the speed of convergence has perhaps been less acute, there are still plans a foot for this convergence to take place. Microsoft's 'Windows Media Center' operating system (see figure 1.2) allows you to watch movies and TV, listen to the radio and music, and look at photos all on the same device namely the computer. The computer is equally not restricted to showing these things on standard CRT monitors as most modern graphics cards now come with TV-out features. This is usually an S-video type jack designed to send a high-quality signal to a TV (see figure 1.3). This allows computers to use conventional televisions for their display (described by Microsoft as the 10-foot experience[3]) whereas previously they were restricted only to specifically designed monitors (the 2-foot experience). This therefore becomes a truly multimedia experience that can be delivered on near cinematic proportions as the largest LCD or Plasma television currently commercially available is 65"[4].

Figure 1.2: Screenshot from Windows XP Media Center Edition 2005

[3] Taken from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/directx/TechnicalArticles/MCE.asp
[4] Taken from http://www.i4u.com/article2217.html and http://www.i4u.com/article2071.html

Figure 1.3: ATI Radeon Graphics card showing TV Out

From a cultural standpoint it is interesting that probably the two greatest transmitters of what Richard Dawkins in The Selfish Gene (1976) coined as cultural memes, a unit of imitation such as a tune, catchphrase, idea etc are being combined. Television since the 1950's has spread relentlessly around the world. The Himalayan kingdom of Bhutan had banned television until June 1999, since its introduction the residents of the capital, Thimphu, now say they are glued to the telly for several hours a day watching soap operas beamed from across the Indian border[5]. Although in Bhutan, this could be similar scenario anywhere in the world.

In the 1982 horror film Poltergeist, it is via the television set that the spirits from another dimension choose to make contact with Carol Anne, the five year-old daughter of Steve and Diane Freeling (see figure 1.4). In one of the film's most memorable scenes, Carol Anne becomes trapped inside the television with her distraught parents communicating with her through a particular channel on the TV. This can be seen as a metaphor for many of our lives where we have ourselves become 'trapped inside the TV' with much of our shared common experiences coming via the television set. In some of the remotest parts of the world, people will have been able to share in the trials and tribulations of Ross, Rachael, Joey, Chandler and Phoebe from the American television series 'Friends', which has been transmitted in at least 46 countries including Iran and India[6].

Figure 1.4: A scene from the movie, Poltergeist (1982)
Directed by Tobe Hooper; with Steven Spielberg as co-producer and co-writer

[5] Taken from 'TV conquers remote Bhutan', Geeta Pandey, 10 March 2005 (http://news.bbc.co.uk/1/hi/world/south_asia/4332357.stm)
[6] http://en.wikipedia.org/wiki/Friends#.22Friends.22_in_other_countries

The Internet via the computer has had a similar effect since its emergence in the mid 1990's. Although this technology has been enabling allowing new radical ideas to been transmitted around the world, there has certainly been some degree of cultural homogenisation via both the television and computer.

Television has come to represent the banal. The image of television is instantly recognisable therefore can be used as a metaphor in art such as in the work of Nam June Paik. Paik's relationship with technology has always been driven by a sense of wonder and play. For him, new technologies are tools that allow him to play with the world. This familiarity has led him to surround himself with technical equipment in an ironic fashion with a family of televisual robots (father, mother, aunt, uncle, children etc), which he presented in a variety of different forms from 1986 onwards[7] (see figures 1.5, 1.6, 1.7).

Figure 1.5: 'TV violoncelo', Nam June Paik (1971), Mixed Media

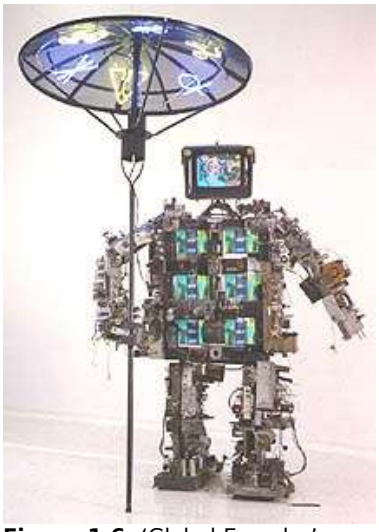


Figure 1.6: 'Global Encoder' Nam June Paik (1994), Mixed Media

Figure 1.7: 'TV Cello', Nam June Paik (2000), Mixed Media

The artist Oonagh Hegarty capitalises on the banal nature of television particularly daytime television in her piece '28,710' (see figure 1.8), in which a number of 20mm x 20mm squares of sticky tape have been overlaid over the entire screen of a second-hand portable TV. The surface impedes our ability to read the photographic image as transparent; its surface is literally rendered opaque. The effect of blurring her source material is to endlessly frustrate our attempts to know the bodies represented, or to possess the image. The artist describes her work as "a battle of control – a need to pin down the properties of a personal experience of the mass media, and a struggle to define an individual encounter with an entity that cannot be contained.[8]"

[7] pg 224 'Digital and Video Art' Florence de Mèredieu, Chambers 2005
[8] Taken from 'The Analysis of Beauty' Northern Gallery for Contemporary Art, 23 July - 11 September 2004 (http://www.ngca.co.uk/home/default.asp?id=46)

Figure 1.8: '28,710', Oonagh Hegarty (2003) mixed media.

Wolf Vostell and Nam June Paik invented video art in the 1960's by 'appropriating' the television set by subverting and disrupting its normal operation[9]. In the introduction to the exhibition 'The Analysis of Beauty' at the Northern Gallery of Cotemporary Art in Sunderland (23rd July – 11th September 2004), the effect of Oonagh Hegarty's piece, '28,710' is described as "almost like watching an unexpected interruption to live television[10]". For me the converging of the television set and the computer provides an interesting opportunity to truly 'interrupt live television', allowing us to apply the techniques developed by video artists to the live medium of television. And which in turn can be displayed on a television set via the TV-out of a computer graphics card so interrupting the expectations of the spectator for a medium that is taken so much for granted.

The vocabulary of the video artist has been developed over the forty or so years since its birth in the 1960's. Jean-Christophe Averty and Max Debrenne experimented with the first graphic effects on television images for the monthly variety shows "Histoire de sourire" (Story of Smiling) and "Les Raisins verts" (Green Grapes) on France's first channel in 1963. The video artist, Jean-Luc Godard experimented with slow motion sequences, text insertion, multiple images, and dividing the screen into segments.

The Swiss video artist Pipilotti Rist piece 'I'm Not The Girl Who Misses Much', 1986 (see figure 1.9) is seen as a seminal work. Rist's classic video takes on rock music with its own tools, pushing pop's repetitive strategies and representations of women to absurd lengths. Footage of the artist chanting the piece's title (a line adapted from The Beatles' song Happiness is a Warm Gun) is replayed at high and low speeds, with obscuring video effects, blurring into an almost painterly procession of images. Rist's manipulation renders her voice into a parody of female hysteria and her body into a grotesquely dancing doll.

[9] pg 9 'Digital and Video Art' Florence de Mèredieu, Chambers 2005
[10] Taken from 'The Analysis of Beauty' Northern Gallery for Contemporary Art, 23 July - 11 September 2004 (http://www.ngca.co.uk/home/default.asp?id=46)

Figure 1.9: Video Stills from 'I'm not the girl who misses much' (1986), Pipilotti Rist, **7:46 min, color, sound**

1.3Objectives

The objectives of this piece of work were to apply some of the techniques used by video art to live television. I choose to implement:

- Image insertion
- Text insertion through subtitling or text as an image
- Slow/fast motion
- Collage of moving images

Originally I was going to write an application in which the images etc were hard-coded but quickly changed my mind as I realised that there was more potential by allowing it to be configurable thereby transforming it into a tool by which I could create the artwork. This in turn made the application act similar to a more conventional medium such as paint so allowing for those 'happy accidents' where something unintentional happens.

Most modern art, which uses computers there is a divide between the method of creation and the display. For example Adobe Photoshop maybe used to create or manipulate a digital photo or scanned image. The method of showing this work is usually very separate either physically by printing this image onto paper or keeping in the digital realm for instance using Flash to produce an art work but using the flash player to show the work but there is still a divide. I thought it would be interesting to remove this divide and write an application, which is used to both create the art work and also is used to display it.

I decided to divide the application into five discreet entities or display modes which are controlled by a central configuration suite but would generate five very different types of artwork but which all had the television picture as their source. These I entitled:

- **Aleotoric Alterpiece** – the television channel is changed randomly on a regular basis either from a selected number of channels or from the entire channel

range.

- **And they say, there is nothing good on TV (Doodle TV)** – doodles are drawn automatically either all over the screen or on the face of the person on the TV

- **Moving Image as Collage** – up to 4 separate TV pictures are arranged on the screen in the form of a collage allowing the pictures to overlap.

- **Quick, Quick, Slow** – the television is either slowed down or speeded up periodically.

- **Re-contextualising Subtitles** – subtitles are added to the television picture which do not necessarily have any correspondence with the images displayed.

1.4Overview

The use of the TV tuner card I believe holds great potential for making challenging and interesting art. Unlike most conventional art mediums such as painting, sculpture and video where the content remains essential static fixed at the time of its creation by the artist, using TV or radio as input which itself is in constant flux, no one day's schedule is the same as the previous then in turn the art piece is also constantly changing. For the spectator this means that potentially each time they visit the artwork they may be presented with a different experience unlike visiting a video installation or painting that does not change.

As one cannot determine what is likely to appear on TV this introduces an element of chance. This taps into a long tradition of artists who have embraced randomness and chance in their work such as the Abstract painter, Jackson Pollock and Dada artist Jean Arp who made collages by dropping small pieces of paper onto a larger piece, then adhering them to where they landed (see figure 1.10). As the pieces I hope to develop use recognisable imagery from the TV it is also hoped that purely by accident, 'happy accidents' may occur whereby meaning maybe derived by the spectator.

Figure 1.10: Jean (Hans) Arp, Collage Arranged According to the Laws of Chance*, 1916–17*
Torn-and-pasted papers on grey paper, 19 1/8 x 13 5/8" (48.6 x 34.6 cm)

1.5Report Structure

Chapter 2 explains the background and inspiration for the five display modes of the project, including how other artists have dealt with similar subject matter. Chapter 3 outlines the design considerations taken for the project. This is followed by the implementation details in chapter 4. Chapter 5 gives an overall evaluation of the system. We also reflect on the approach we have taken and its limitations. The conclusions are drawn and possible future work for the project is suggested. Appendix A details the development of the 'Media Montage' application which was a spin off project which came from my initial work in the Collage section of the 'Telly Vision' application. Appendix B shows a selection of artworks that have been produced using both of these applications.

2.Background

2.1Aleotoric Alterpiece

The word Aleatory (or aleatoric) means, "pertaining to luck", and derives from the Latin word alea, the rolling die. Aleatoric art is that which exploits the principle of randomness.[11] Most ambitious of these is Raymond Queneau's Cent Mille Milliards de Poèmes. Queneau created the work in 1960, by creating 10 possible options for each line of a sonnet. Each line is selected randomly, creating an almost perpetually novel poem. The idea relates somewhat to the Surrealist game of Exquisite Corpse, in which one person draws the top of a figure, then folds the paper only a small portion is revealed to the next person, who draws the middle portion, and so on[12]. The word 'Alterpiece is play on the words Altarpiece and alter.

When one stares at an off-tune TV one can often begin to make out figures and faces in the static, for me this is very similar to the numerous occasions in history where people have found spiritual and religious significance due to freak occurrences in nature that have produced imagery which they believe is of religious icons like a picture of Jesus. For example in April 2005 hundreds of people flocked to a motorway underpass in the US city of Chicago to view a salt stain on a concrete wall many said was an image of the Virgin Mary (see figure 2.1)[13].

Figure 2.1: A salt stain in US motorway underpass in Chicago said to bear the image of the Virgin Mary

In such cases, the imagery is generated purely by chance and it is the spectator that associates the meaning and significance to it. I thought this would be something interesting to explore. Using TV as the source which is for our purposes a random input as we cannot govern what will appear on the screen, one is just as likely to find something spiritual significant in the work as one would when walking through an motorway underpass.

[11] Taken from http://www.artlex.com/ArtLex/a/aleatory.html
[12] Taken from http://www-personal.umich.edu/~luriea/aboutpoems.html
[13] Taken from http://news.bbc.co.uk/1/hi/world/americas/4468275.stm

2.2And they say, there is nothing good on TV

When I was at school, I often remember opening school textbooks particularly history books and finding that other students had drawn on the illustrations and photos, drawing such childish things as glasses, willies, tits, and Hitler moustaches over these images. I thought it would be an interesting idea to apply this to live TV images. A channel like BBC 24, a 24-hour news channel would be a particularly appropriate TV channel to use as news has a certain gravitas but it is difficult to take something serious when there is pair of comedy glasses or moustache drawn on the newscaster. There is also the opportunity for 'happy accidents' to occur such as if the computer was to draw a Hitler moustache on George Bush then the spectator may read this as a political statement but the computer has no comprehension of who George Bush is so this would happen purely by chance. There is also a dichotomy as the doodle in real-life is such a low-tech thing whereas on the computer it becomes a high tech thing requiring face detection etc.

Although this imagery sounds childish it taps into a long tradition in art of defacing other work. The most well known act of degrading a famous work of art is probably Marcel Duchamp's L.H.O.O.Q., a cheap postcard-sized reproduction of the Mona Lisa upon which in 1919 the artist drew a moustache and a thin goatee beard (see figure 2.2). On one hand L.H.O.O.Q. must be understood as one of Duchamp's "readymade" works of art -- works that he didn't make, but which, by having been placed intellectually within a conceptual framework of "Art," he forces the observer to see ordinary objects from new perspectives. In this way their innate aesthetic contents would make themselves manifest. However, to most observers, instead of elevating the ordinary, Marcel's Mona Lisa works in the opposite direction; it defaces (literally) that which has been cherished, and brings a famous work down to the level of vulgar vandalism and cheap reproduction. The title makes the point, too, but obscurely, since when pronounced in French "L.H.O.O.Q." reports as a pun on the phrase "Elle a chaud au cul," which translates colloquially as "She is hot in the ass."[14]

Figure 2.2: Postcard of Replica of L.H.O.O.Q. (1919), Marcel Duchamp, Collotype, hand-coloured with watercolour.

[14] Taken from http://www.studiolo.org/Mona/MONASV12.htm

A modern example of this can be found in the work of Jake and Dinos Chapman's 'Insult to Injury' (2003) in which a set of Goya's Disasters of War etchings is defaced (see figures 2.3 and 2.4). The artists have systematically gone through the entire 80 etchings and changed all the visible victims' heads to clowns' heads and puppies' heads.

The Chapmans have recognized in the Disasters of War a template for their own nightmares. This work also references the American artist Robert Rauschenberg in 1950's erasing a drawing by Willem de Kooning, the great abstract expressionist painter. However in this case he sought the older artist's permission, of course impossible in this case

Figure 2.3: Etching from 'Disasters of War' series,
Francisco Goya

Figure 2.4: Insult to Injury (2003), Jake & Dinos Chapman

2.3Moving Image as Collage

There is currently a trend for artists to use collage in their work. This can be seen in the works of in-vogue artists such as Christian Marclay (see figure 2.5) and Wangchi Mutu (see figure A.2). Such artists generally use static images as the elements of their collage, in Marclay case the elements are record covers in his Body Mix series and Mutu uses pictures cut from fashion magazines. Therefore I thought it would be interesting concept to extend this to moving images namely television pictures. It also introduces an element of chance that there maybe some correspondence in the images shown on the separate television channels.

Figure 2.5: 'Footstompin', from the Body Mix series, 1999.
Christian Marclay, Record album covers and thread.

2.4Quick, Quick, Slow

Using the TV capture card, I thought it would be interesting to explore peoples' perception of time and memory. The recall of past memories triggered by a Madeleine biscuit is a feature of the book 'In Search of Lost Time' by Marcel Proust. In this piece entitled 'Quick, Quick, Slow' with the name making reference to a rhythm commonly used in dance and life often seen as a dance (see figure 2.6), the spectator is confronted by a TV moving image that is constantly in a state of flux either slowing or speeding up. This will be a recorded clip from TV, for example recorded 30 minutes previously and 30 minutes in duration. This clip is effectively memory being in the past. By playing the clip at different rates, it is similar to our own perception of memory where certain things seem more vivid that others, for instance if we meet someone from our past than it seems the past is catching up with the present. However this obviously cannot happen, and is also the case in the piece that although the clip is speeded up and slowed down overall it is still only 30 minutes hence the past remains the past. It also makes reference to people's perception of time as people often say that 'time flies by when you are having fun' or 'time is dragging'.

Figure 2.6: The Dance of Life by Edvard Munch

2.5Re-contextualising Subtitles

Without audio on the television and with the subtitles on, there is a reliance on the subtitles to aid the understanding, meaning and context of the images being shown on the screen, however if these subtitles have no relation to images on the screen then the images are re-contextualised and consequently their meaning is changed. There is like some of the previous display modes a chance that the spectator may draw some unexpected meaning from associating the images with the text. This would once again be purely random, as one does not know what images are going to be shown on TV.

3. Overall System Design

3.1Hardware Requirements

For the development of this application I used a Hauppauge WinTV Express PCI card (model 756). This is the entry level PCI TV capture card providing one TV signal input stream with mono sound. Although not tested with other cards I have included the functionality for the software to be configured with other cards if installed with the correct drivers on the machine.

3.2Software Tools and Libraries

3.2.1Java Media Framework

It was my initial intention to develop the application using Java and the Java Media Framework Application Programming Interface (JMF). The Java Media Framework is an API for incorporating time-based media into Java applications and applets. This had the advantage of being reasonably well documented; there is a large online-community of Java developers willing to share their experience and that I had had some previous exposure to the Java language already.

Figure 3.1 shows the high-level architecture of the JMF. The JMF 2.0 API provides support for capturing and storing media data, controlling the type of processing that is performed during playback, and performing custom processing on media data streams. In addition, JMF 2.0 defines a plug-in API that enables advanced developers and technology providers to more easily customize and extend JMF functionality.

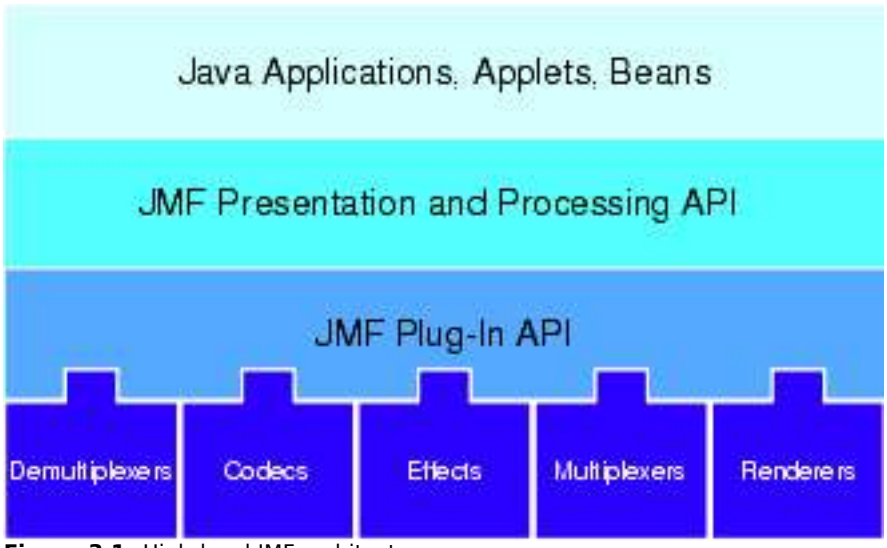


Figure 3.1: High-level JMF architecture

Data sources and players are integral parts of JMF's high-level API for managing the capture, presentation, and processing of time-based media. A data source encapsulates the media stream much like a video tape and a player provides processing and control mechanisms similar to a VCR. Playing and capturing audio and video with JMF requires the appropriate input and output devices such as microphones, cameras, speakers, and monitors. JMF also provides a lower-level API that supports the seamless integration of custom processing components and extensions. This layering provides Java developers with an easy-to-use API for incorporating time-based media into Java programs while maintaining the flexibility and extensibility required to support advanced media applications and future media technologies.

Time-based media can be captured from a live source for processing and playback. Capturing can be thought of as the input phase of the standard media-processing model. To capture time-based media you need specialized hardware. Capturing a TV broadcast requires a TV tuner and an appropriate video capture card. Most systems provide a query mechanism to find out what capture devices are available.

Capture devices can be characterized as either push or pull sources. For example, a still camera is a pull source--the user controls when to capture an image whilst a TV Capture Card is a push source--the live source continuously provides a stream of video and audio.

The format of a captured media stream depends on the processing performed by the capture device. Some devices do very little processing and deliver raw, uncompressed data. Other capture devices might deliver the data in a compressed format.

JMF 2.0 provides the following classes and interfaces that are useful for TV Capture; *`CaptureDevice`*, *`CaptureDeviceInfo`*, *`CaptureDeviceManager`*.[15]

I experimented with JMF but although I was able to see the TV Capture Card using *`CaptureDeviceManager`*, I was not able to change the channel and was only able to see static. In my experience, it would seem that the JMF API does not fully expose all the functionality of the graphical engine of the Operating System it is running on either DirectX in Microsoft or OpenGL.

3.2.2DirectShow

As the TV capture card I was using did not seem to be fully supported on any other platform apart from Windows, there seem to be no advantage in sticking to the JAVA programming language as it's platform independence was irrelevant as the drivers for the TV Capture card determined what Operating system the application would be able to run on. Consequently I decided to experiment with DirectX and more particularly DirectShow, the part of DirectX responsible for streaming media on the Microsoft Windows platform.

DirectShow provides for high-quality capture and playback of multimedia streams. It supports a wide variety of formats, including Advanced Systems Format (ASF), Motion Picture Experts Group (MPEG), Audio-Video Interleaved (AVI), MPEG Audio Layer-3 (MP3), and WAV sound files. It supports capture from digital and analog devices based

[15] http://java.sun.com/products/java-media/jmf/2.1.1/guide/JMFPreface.html#999106

on the Windows Driver Model (WDM) or Video for Windows. DirectShow is integrated with other DirectX technologies. It automatically detects and uses video and audio acceleration hardware when available, but also supports systems without acceleration hardware.

DirectShow simplifies media playback, format conversion, and capture tasks. At the same time, it provides access to the underlying stream control architecture for applications that require custom solutions. You can also create your own DirectShow components to support new formats or custom effects.

To achieve the throughput needed to stream video and audio, DirectShow uses DirectDraw and DirectSound whenever possible. These technologies render data efficiently to the user's sound and graphics cards. DirectShow synchronizes playback by encapsulating media data in time-stamped samples. To handle the variety of sources, formats, and hardware devices that are possible, DirectShow uses a modular architecture, in which the application mixes and matches different software components called filters. DirectShow provides filters that support capture and tuning devices based on the Windows Driver Model (WDM), as well as filters that support legacy Video for Windows (VfW) capture cards, and codecs written for the Audio Compression Manager (ACM) and Video Compression Manager (VCM) interfaces.

Figure 3.2 shows the relationship between an application, the DirectShow components, and some of the hardware and software components that DirectShow supports. As illustrated here, DirectShow filters communicate with, and control, a wide variety of devices, including the local file system, TV tuner and video capture cards, VfW codecs, the video display (through DirectDraw or GDI), and the sound card (through DirectSound). Thus, DirectShow insulates the application from many of the complexities of these devices. DirectShow also provides native compression and decompression filters for certain file formats. [16]

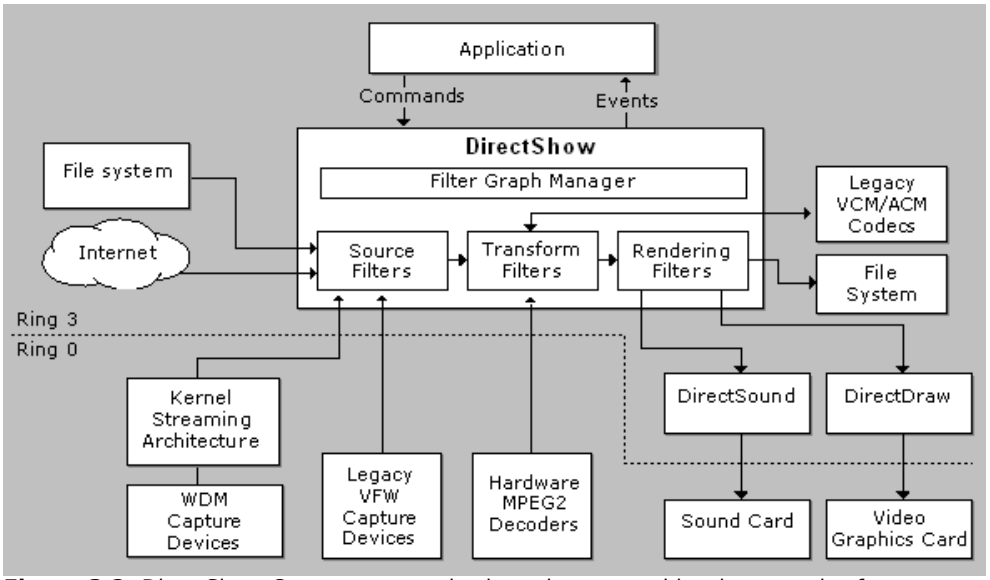


Figure 3.2: DirectShow Components and selected supported hardware and software components

[16] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directshow/htm/generalgraphbuilding.asp

DirectShow is based on the Component Object Model (COM). To write a DirectShow application or component, you must understand COM client programming. For most applications, you do not need to implement your own COM objects as DirectShow provides the components necessary.[17] However the custom interfaces are mainly exposed for C++ whilst there is limited functionality for Visual Basic using DirectShow VB components. Due to the timescales for development I thought it would be unrealistic to be able to learn C++ in the time and Visual Basic would not provide me with sufficient functionality. Fortunately I was able to find the DShowNET project, a library of DirectShow interfaces rewritten in the C# programming language by NETMaster (available online at http://www.codeproject.com/cs/media/directshownet.asp). This process of getting managed (C#) and unmanaged objects (COM) to work together is called interoperability, but it is commonly called interop. I was also very fortunate to find Brian Low's Capture class library for C# (available online at http://www.codeproject.com/cs/media/directxcapture.asp). This class library uses COM Interop to access the full capabilities of DirectShow via the DShowNET libraries. It is designed for capturing audio and video to AVI files, which includes from a TV tuner card. Although this did not exactly do what I wanted, it gave me sufficient pointers to work out how to do the rest.

C# (pronounced C sharp) is a new programming language designed for building a wide range of enterprise applications that run on the .NET Framework. It borrows much of the syntax of Java, a language of which I already had some knowledge.

Microsoft's .NET is an alternative to Java and J2EE. Microsoft's Visual Studio .NET development environment incorporates over 20 different languages including RPG, COBOL and Microsoft's own C# for business programming. With all languages compiling to a common language runtime provides a common, consistent development interface across all languages supported by the .NET Framework.

Figure 3.3: .NET Framework Architecture

3.2.3OpenCV

In the display mode 'Doodle TV' it was my intention to draw silly moustaches etc on the faces of the people on television, obviously to accomplish this would require me to be able to locate the faces on the picture. This is quite a complex problem and one that I

[17] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directshow/htm/introductiontodirectshow.asp

was unlikely to solve by myself therefore I decided to use an existing library of computer vision functions which would allow me to achieve this.

Intel Corporation have made available in source and object form their multi-platform Intel Open Source Computer Vision Library (OpenCV). It is widely used by researchers for advanced image processing. It is actively evolving, presently containing more than 300 C++ functions, including a rich set of matrix operations and linear algebra functions. It has an active user base generating around 450 mailings per month to the subscription list. It provides functionality that ranges from simple image-processing tasks (thresholding, edge detection) to advanced algorithms for face detection, contour extraction, building 3-D models from stereo camera views, correcting 'fish-eye' effects from cameras, feature detection, object tracking, building covariance matrices, extracting eigenvectors and eigenvalues, etc.

As the OpenCV library requires a steep learning curve and one needs significant expertise to get started and is built around a procedural model which is not suited to the object-orientation of the C# programming language, I decided also to use the managed C# OpenCV wrapper, SharperCV developed by the Computing Department at Rhodes University, South Africa (see Resources section for a link).

3.3Architecture

Figure 3.4 illustrates the system architecture of the Telly Vision application. The DShowNET Interface Library is the interop layer so allowing the two layers above to access the DirectShow COM components.

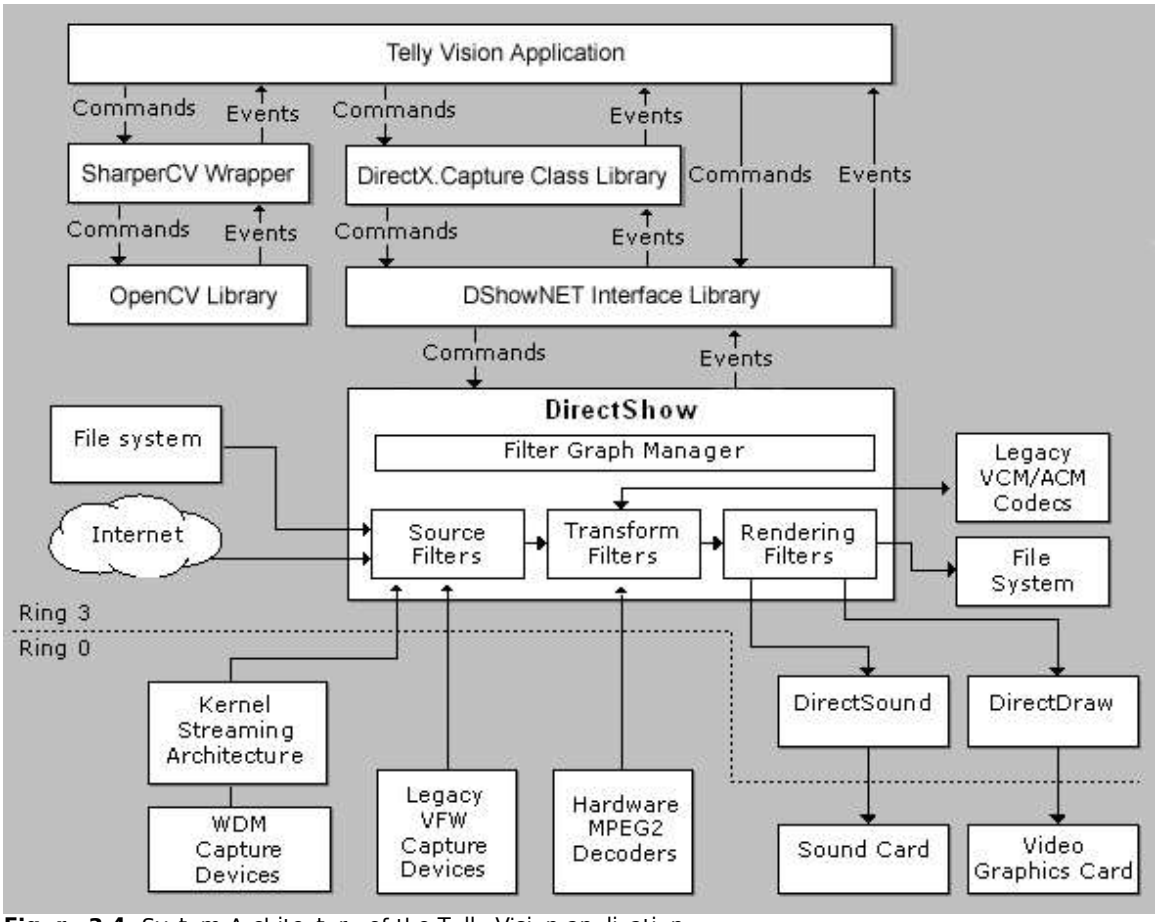


Figure 3.4: System Architecture of the Telly Vision application

4.Implementation

4.1General Overview

When designing programmes with the potential of running on widescreen TV, it is important to take into account the aspect ratio (the width and height proportion of the display). Standard TV and computer monitor displays have an aspect ratio of 4:3 (meaning for every 4 pixels running along the width of the frame buffer there are 3 pixels along the height, sometimes also represented simply as 1.33). Widescreen TV (High definition TV) has an aspect ratio of 16:9. The three most common formats of TV are shown below:

- **480p EDTV** (Enhanced Definition Television) - 480 scan lines presented progressively. This mode can output either 4:3 (with a frame buffer resolution of 640x480) or 16:9 (720x480).
- **720p HDTV** - 720 scan lines presented progressively. This mode is always 16:9 (1280x720).
- **1080i HDTV** - 1080 scan lines presented interlaced. This mode is always 16:9 (1920x1080, or 1920x540 if rendering the interlace fields separately).

It is important that any programme should be able to display images correctly without being stretched or cropped on all of these formats. It is also important to take into account the displayable area on the TV. The entire image frame buffer may not be visible to the end user; often a certain amount of border pixels are covered by the display's front bezel. To ensure that critical UI elements are visible across the spectrum of potential consumer display hardware the "title-safe region" requirements for the target display mode should be observed: For non-HDTV modes, the title-safe region is the inner 85 percent of the frame buffer; for HDTV modes, the title-safe area is the inner 90 percent. Therefore it is best to keep within the inner 85 percent of the frame buffer[18].

Although the TV tuner card that I am currently using only displays the television picture at 4:3 ratio, I have coded all of the various display modes of the project to display correctly at all screen resolutions. Before the television picture and associated image or text is displayed, the television picture size, image size and font sizes are recalculated in relation to the current screen resolution. This has the effect that any output should be indistinguishable from the same output at any other screen resolution.

4.1.1DirectShow and Capture Class Library

In DirectShow, all stream operations are encapsulated as filters, COM objects that have a standard behaviour along with whatever custom capabilities they may be given. Applications are built by connecting these filters together in order to perform a given

[18] Taken from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/directx/TechnicalArticles/MCE.asp

task.

Filters come in three basic types: source filters for input, transform filters for any intermediate processing step, and renderer filters for output.

A source filter introduces data into the stream. This data may originate in a file or in a device such as a TV tuner. Any media device with a correctly implemented Windows Driver Model (WDM) driver is automatically exposed to user-mode applications as a DirectShow source filter, complete with whatever interfaces the hardware vendor has exposed to enable applications to get and set properties on the device.

A transform filter receives input data from some other filter, performs some operation on the data, and then passes the data to another filter downstream. Transformations may be parsing of streams, encoding or decoding, overlay of text, or any type of analysis or manipulation of the audio or video bits.

A renderer filter accepts data from either a source or transform filter and outputs it to the screen, the speakers, a file, a device, or some other location.

The main task of an application is to get these filters working together. For that task, applications use a high-level object called the Filter Graph Manager (FGM) to connect filters together. The collection of connected filters is called a filter graph. Data in a filter graph always moves downstream from the source filter to the renderer.

The modular design of DirectShow requires that all filters follow certain rules for communicating with other filters. These rules make up the connection and streaming protocols. But before filters can communicate with these protocols, they must speak a common language. This language is the media type. A media type is the shorthand way of referring to a group of data structures that collectively describe the data in a stream. Filters communicate by passing media types back and forth to indicate what types of data they can process.

Before two filters can stream data, they must use the language of the media type to agree on what that data is. Filters accomplish this in the connection protocol. To be precise, it is not the filter that performs the connection protocol; rather, it is a separate object that is created and owned by the filter, called a pin. Pins come in two varieties, input and output. Whenever a filter needs to connect with another filter, it does so through a pin.

The FGM or the application is responsible for adding filters to the graph and specifying which two pins should attempt to connect. But the pins themselves perform all the actual work of connecting. When two pins connect, they must agree on three things: a media type, which describes the format of the data, a transport, which defines how the pins will exchange data, and an allocator, a new object owned by one of the pins, which will create and manage the buffers that the two pins will use to exchange data.

Data in the filter graph is pushed through the graph by a source filter. Pushing means that the filter fills up the downstream filter's buffers with data as fast as it can. For most video processing operations on modern PCs, the processing power is more than enough to render 30 frames per second, so the problem is how to slow down the graph so that it runs at the correct speed. The renderer filters look at the time stamps on the media

samples and wait until the current time matches the time stamp before rendering a frame or an audio sample. While they are waiting, the entire graph is blocked once all the buffers between each pair of filters are filled

The Capture Class library simplifies this process. By creating a new instance of the *Capture* class, the correct filters, media types, and pins are created to allow the television media stream to be displayed in a chosen panel on a form set by the *PreviewWindow* property of the *Capture* class.

4.1.2 General Interface Design

Many computer applications interfaces are very self-important with dialog boxes that always state 'Are you sure you want to…' as if it is life or death decisions. I want my design to be less self-important a little bit tongue-in-cheek, hence being able to choose such options as "teacher's pet". I had originally considered adding doodles and graffiti to the configuration form to continue this informal feel but so far have not done so.

Although it has always been the plan that this application is purely a tool for myself to generate artwork, I decided that would make the interface as user-friendly and self-explanatory as possible as there is always the possibility that if I was exhibiting a work somewhere then someone else maybe called to configure the application to my instructions hence the need for it being user-friendly.

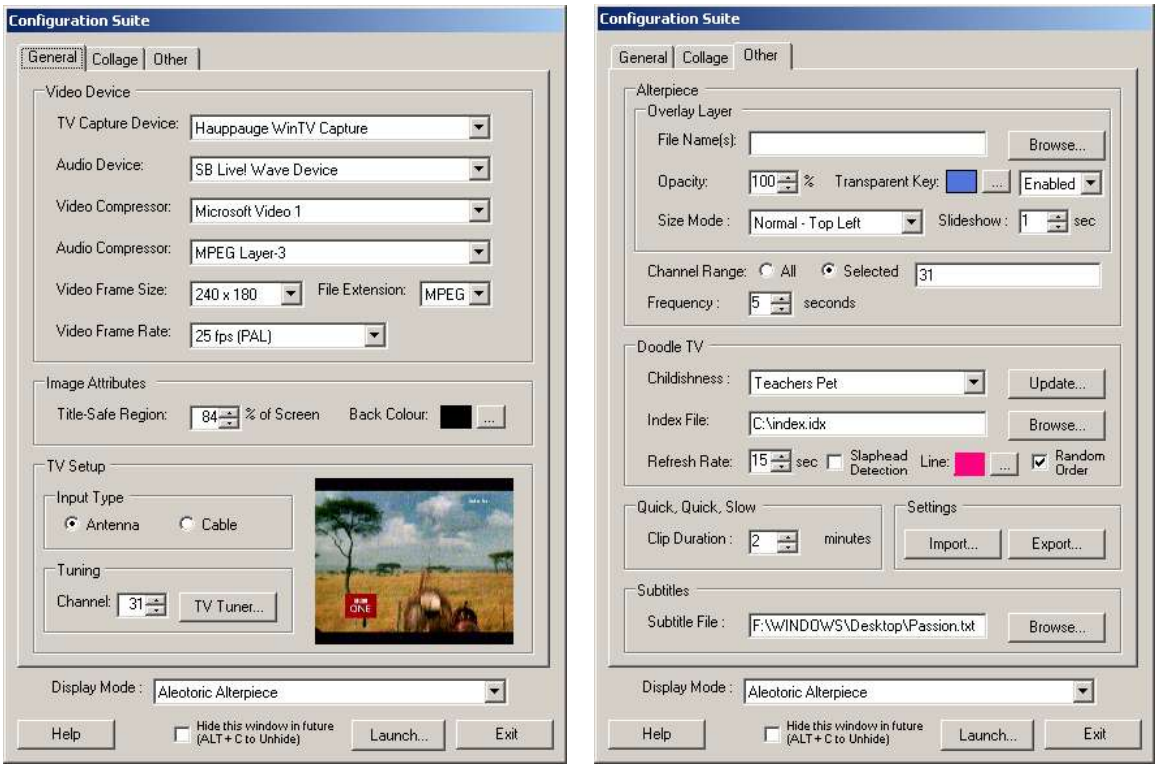


Figure 4.1: Two of the three tabbed panes from the Configuration Suite

The interface for the configuration suite in which the five display modes are controlled is designed to fit on screen resolutions 800 x 600 or higher. The Interface is designed using a tabbed layout so only a limited number of configurable options are on view at any one time. The fields of the configuration form are populated by registry values if the application has been previously run or by default values. There is validation behind all of the configurable fields that do not have restricted options i.e. text fields (see the relevant section for specific validation). Validation occurs when the display mode is launched but only on the fields for the particular display mode that is being launched. If the validation is successful then all of the fields are written to the registry and the display mode is launched.

4.2Aleotoric Alterpiece

4.2.1Overview

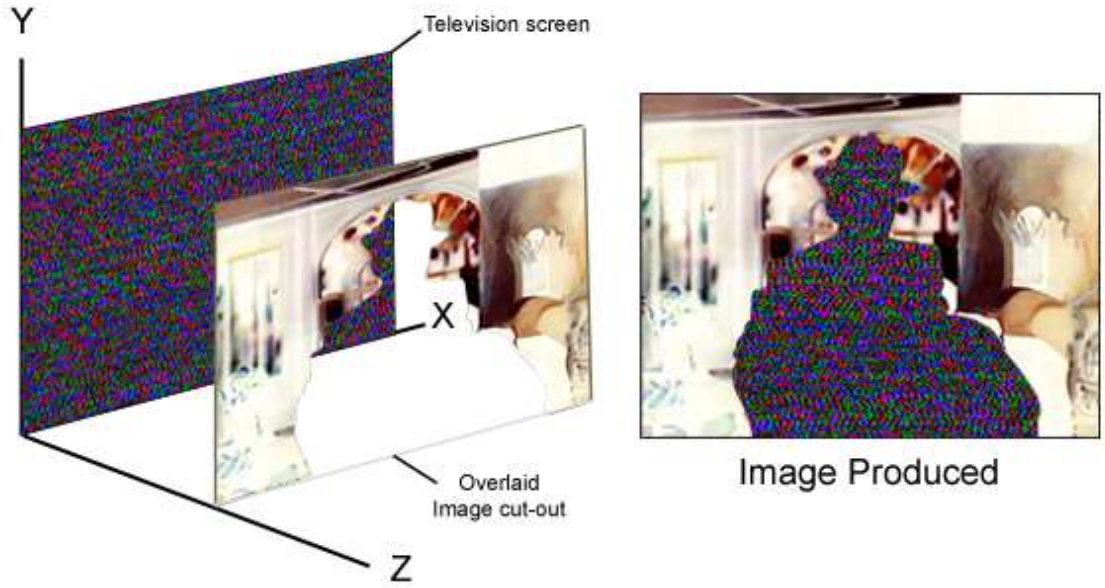


Figure 4.2: Diagram illustrating how the 'Aleotoric Alterpiece' display mode works

The 'Aleotoric Alterpiece' display mode consists of two elements; the television picture, which the channel is periodical changed and the optional image overlay. The figure above illustrates how these two elements are combined to produce a single on-screen collage.

By utilising Brain Low's Capture Class library, it is relatively easy to change the channel of the television. The `Capture` class library includes a *`Tuner`* class by which it is possible to set the *`Channel`* property with an integer value. This property is set by the *`Tick`* event of a *`Timer`* control class within the *`frmBaseLayer`* form class within Telly-Vision project. The *`frmBaseLayer`* form class is the main form responsible for showing the television picture once the selected display mode is launched. The frequency of the *`Timer`*'s *`Tick`* event being fired is set by the *`Interval`* property, which is set by the multiplication of one thousand and the *`Frequency`* property of *`frmBaseLayer`* form

class. This in turn has been set by the *frmConfig* from class. Consequently this will cause the television channel to be changed periodically by the desired amount of seconds as set on the Configuration form (an instance of the *frmConfig* class).

If an overlaid image is present, an instance of the *frmOverlay* form class is created to contain the image or images. This exploits the property within the .NET framework that it is possible to have transparent forms by setting the *TransparentKey* property to the same colour as the background colour of the form; this allows the images to appear as if they are cut-outs placed over the television picture. The class also exploits the property within the .NET framework that it is possible to change the opacity of the form by changing the *Opacity* property value of the form. The file names with their file directories are passed to the *frmOverlay* in the property *OverlayImage* which accepts an array. These images are each passed to the method *ImageAnimator.CanAnimate* to determine whether they are animated GIFs. If they are GIFs then pictureBox controls are used to display them as these can be resized without affecting the properties of the GIF. If they are standard image files then they are set to the background of a panel control. These can then be resized and positioned on screen according to the property *OverlaySizeMode,* which accepts an enum of the type *ESizeMode*. This has seven options; keep the existing image size and show in the top right corner of the viewable screen (*NormalTopRight*); keep the existing image size and show in the top left corner of the viewable screen (*NormalTopLeft*); keep the existing image size and show in the bottom right corner of the viewable screen (*NormalBottomRight*); keep the existing image size and show in the bottom left corner of the viewable screen (*NormalBottomLeft*); keep the existing image size and show in the centre of the viewable screen (*CentreImage*); stretch the image to the size of the screen but keeping proportions (*StretchConstraint*); stretch the image to the size of the screen regardless of keeping proportions (*StretchNoConstraint*). If there is more than one image passed to the form then each image will be displayed for a set length of time determined by the property OverlaySlideShow. In this scenario whilst one image is being displayed, another image's size and screen location is being calculated to make the transition from one image to another as smoothly and quickly as possible.

4.2.2Interface Design

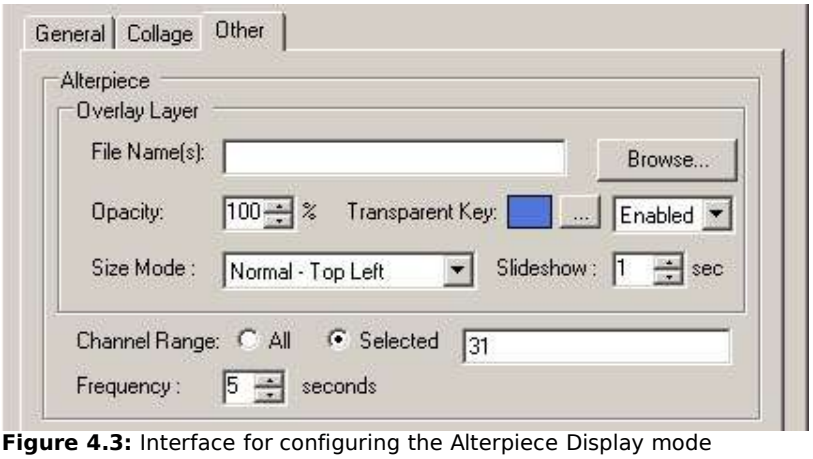


Figure 4.3: Interface for configuring the Alterpiece Display mode

The interface for the ‘Aleotoric Alterpiece’ section consists of two elements; the Alterpiece configuration and the Overlay Layer. The Alterpiece configuration consists of the option to reduce the channel range to a selected number of channels each separated by a comma, or semi-colon else the channel range will be the full range of channels from 1 to either 69 if using an Antenna or 107 if the cable option has been selected. Frequency by which the TV channel is changed is controlled by the Frequency control. There is also validation on the ‘Selected’ Channel range if this option has been selected.

4.2.2.1Image Overlay

Figure 4.4a: Standard Colour picker

Figure 4.4b: Enhanced Colour picker allowing you to picker colour from image

The Overlay function refers to the option of having an image placed over the top TV screen. This can be a series of images separated by a semi-colon in the File Name text field. The overlay function exploits the inbuilt functionality of the .NET forms by which it is possible to alter their opacity. The transparent key is the colour that becomes transparent if the enabled option is selected when the image or images are displayed. It is possible to select the transparent key specifically from the image that is selected in the ‘File Name’ field if there is more than one (see figure 4.4b). If the image is not found or the ‘File Name’ field is blank then the standard colour picker is shown instead (see figure 4.4a).

The overlay function can show the entire standard image file formats including JPEG and PNG. It can also show animated GIF although the transparent key will not be enabled for these types of images. When more than one image is present then the frequency field controls the frequency by which these images are shown in sequence. The Size Mode field controls the position and size of the image that is displayed over the television screen.

There is validation on the ‘File Name’ field if not blank to see if the file or files exist and they have valid file extensions.

4.3And they say, there is nothing good on TV (Doodle TV)

4.3.1Overview

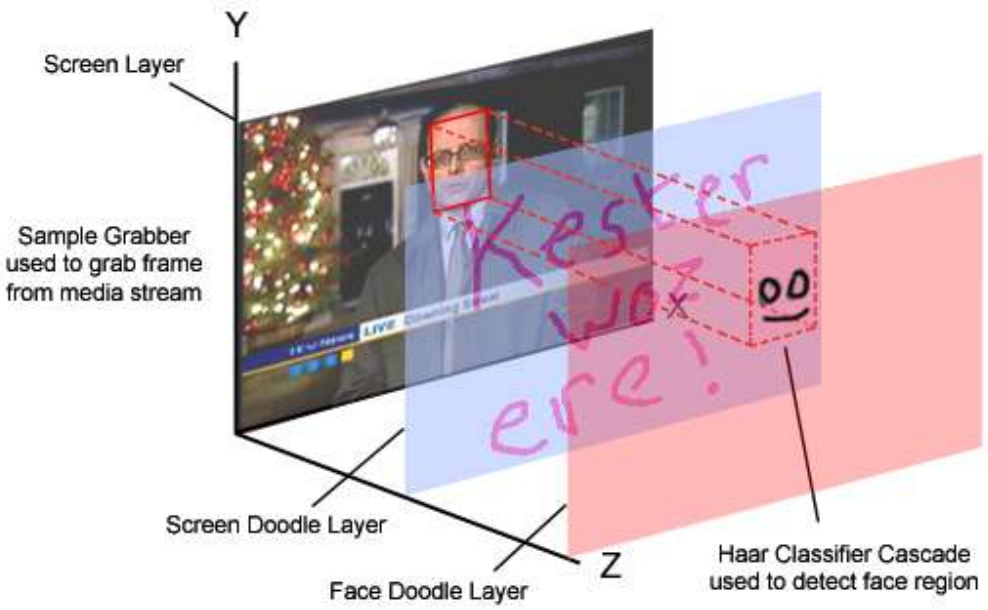


Figure 4.5: Diagram showing overview of how Doodle TV display mode works

The figure above shows the four elements of the 'Doodle TV' display mode. The first and bottom layer is the television layer. This is an instance of the class form *frmDoodleBaseLayer*. This is the only one of the five display modes not to use the class *frmBaseLayer* to display the television screen as it contains a Sample Grabber filter. The Sample Grabber filter is a transform filter that can be used to grab media samples from the stream as they pass through the filter graph. This is used in the analysis of the television picture in the face detection process for the face doodle, which is covered in more detail later on. The other two elements are contained in separate forms overlaid over each other and utilising the built in functionality of the .NET framework of being able to make forms transparent. The *frmDoodleOverlay* form class is responsible for the 'Screen Doodle', whilst the *frmDoodleFaceLayer* form class is responsible for the 'Face Doodle' and is the top-level form.

The C# sample program ScribbleDoc from Microsoft's MSDN website has been adapted to draw the doodles at both design and runtime. The functionality of the *ScribbleDoc* class has been extended to allow greater manipulation of the arraylist *StrokeList,* which contains the bounding rectangles of two points. To keep the proportions of the doodle the same when creating it using the *frmDoodleConfig* form class as when

drawing the doodle in display mode using the *frmDoodleOverlay* form class and *frmDoodleFaceLayer* form class, the two points in each bounding rectangle and line width are multiplied by the same factor as the panel's width in *frmDoodleConfig* in relation to the panel's width in *frmDoodleOverlay* or *frmDoodleFaceLayer*.

To allow for multiple ScribbelDoc files to be displayed in succession on the screen at run time, an index text file is used. This file contains the file locations of the ScribbleDoc files organised by the level of Childishness (Teacher's Pet, Slightly Odd, Very Silly and Nutter) and the Type (either face or screen doodle). Each doodle in its category is given a unique name so can be picked from a list of doodles at design time if so desired. There is an additional field for the screen doodles within this file, which states how long in minutes the doodle will be displayed on the screen at run time. The four level of Childishness allow for four different settings to be held within one index file. By merely changing the level of Childishness will there by change the doodles that are available to be either placed on the screen or face at run-time. It is possible to choose to have the doodles displayed in the order they are read from the index file or randomly. The latter uses the Random class seeded by the system time.

The *Paint* method of the *frmDoodleOverlay* class form has been overloaded to allow the doodle to be drawn more realistically on the screen as if by an invisible pen. This is achieved by adding a sleep to the main thread responsible for the paint method after each element in the StrokeList arraylist has been drawn on the screen. This does however mean that it is not possible to close this application if the user presses a key during this short period whilst the doodle is being drawn on the screen, as it is not possible to interrupt it.

A timer is used to grab a frame from the media stream using the Sample Grabber at configurable intervals in the class form *frmDoodleBaseLayer*. This image is saved and the filename is passed to *DoodleRefresh* method in the instance of the *frmDoodleFaceLayer* class form. The image can then be opened as a *CvImage* object. This is the image class used by the OpenCV library. It is possibly not the most efficient way of doing this but there didn't seem to be any means of casting a C# *Bitmap* or *Image* object to a *CvImage* other than opening it up from a file. A new background thread is created and a Haar Classifier Cascade is carried out on the screen image to find likely rectangles in which faces are located. In the first of these rectangles a face doodle is drawn. If no likely rectangles are found then any existing face doodle is removed.

4.3.1.1Haar Classifier Cascade

The Haar Classifier Cascade is an object detector that was initially proposed by Paul Viola and improved by Rainer Lienhart. The method uses simple Haar-like features (so called because they are computed similar to the coefficients in Haar wavelet transforms) and a cascade of boosted tree classifiers as a statistical model. Initially, the classifiers are trained on images of fixed size, and detection is done by sliding a search window of that size through the image and checking whether an image region at a certain location "looks like a face" or not. To detect faces of different size it is possible to scale the image, but the classifier has the ability to "scale" as well. Fundamental to the whole approach are Haar-like features and a large set of very simple "weak" classifiers that use a single feature to classify the image region as face or non-face. Each feature is described by the template (a shape of the feature), its coordinate

relative to the search window origin and the size (scale factor) of the feature[19]. Figure 4.6 shows a set of 14 templates.

After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a "1" if the region is likely to show the object (i.e., face/car), and "0" otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily "resized" in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales[20].

The word "cascade" in the classifier name means that the resultant classifier consists of several simpler classifiers (stages) that are applied subsequently to a region of interest until at some stage the candidate is rejected or all the stages are passed.

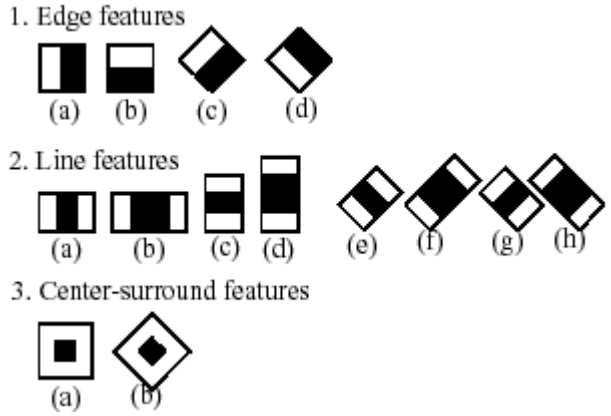


Figure 4.6: Extended set of Haar-like features

4.3.2Interface Design

The interface for the Doodle TV display mode has two elements. The first is contained in the main ‘Configuration Suite’ form and controls the general settings for the mode. It is possible to change level of childishness, random playback, the colour of the lines for both screen and face doodles, the index file and refresh rate whereby the frame is grabbed from the media stream for face recognition.

[19] Taken from http://www.intel.com/technology/itj/2005/volume09issue02/art03_learning_vision/p04_face_detection.htm

[20] Taken from http://www.cs.bham.ac.uk/resources/courses/robotics/doc/opencvdocs/ref/OpenCVRef_Experimental.htm

Figure 4.7: Interface for the high-level configuration of the Doodle TV display mode

By pressing the 'Update' button on this frame, the index file will be opened if present or will be created. The user is then presented with the Doodle Drawer screen. This allows the user to either update an existing doodle entry in the index file this maybe either the doodle itself or if it is a screen doodle the duration it is visible on the screen for at runtime, or create a new doodle.

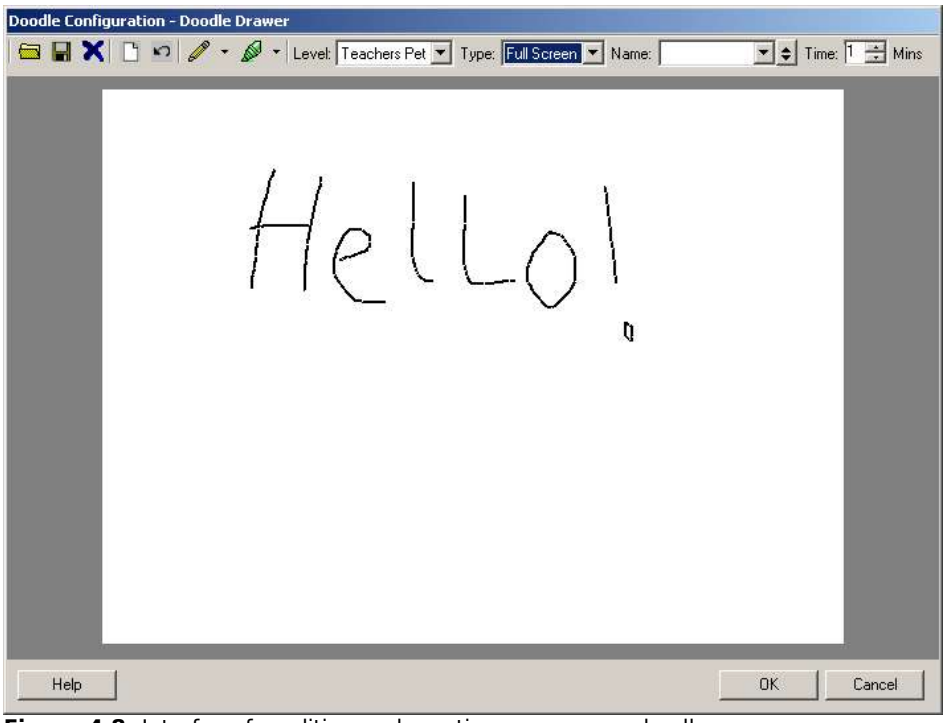


Figure 4.8: Interface for editing and creating new screen doodles

It is possible to view the current list of doodles for a particularly level and type by selecting the appropriate drop-down lists at the top of the screen. Clicking the desired doodle name from the drop-down list opens the doodle, which in turn can be altered or deleted. By clicking on the fourth icon at the top left of the screen a new blank doodle is generated. A doodle can be drawn using two type of pen; either thin or thick which in turn can be altered by the associated drop-down lists. Dependant on the type of doodle selected, the drawing area is either a larger 4:3 ratio area for screen sized doodles or the smaller area for face doodles with a face in the background as a guide.

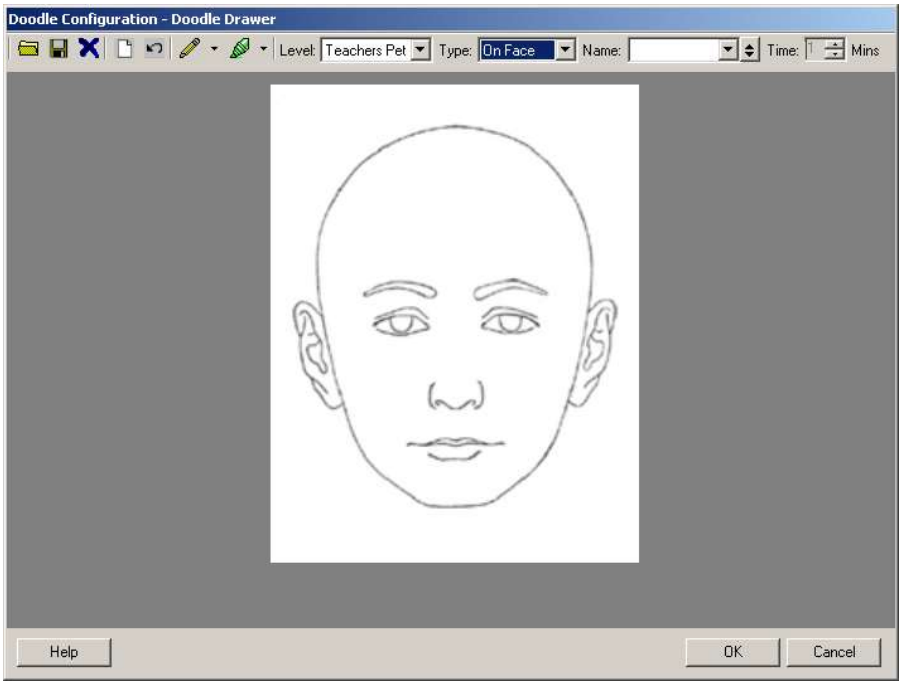


Figure 4.9: Interface for editing and creating new face doodles

The order of the doodles as they are written to the index file can be changed by selecting a doodle name from the drop-down list as below and using the arrows to the right to either move the doodle up or down the order.

Figure 4.10: Drop-down list of screen doodles with arrow button to the right to reorder the doodle sequence

4.4Moving Image as Collage

4.4.1Overview

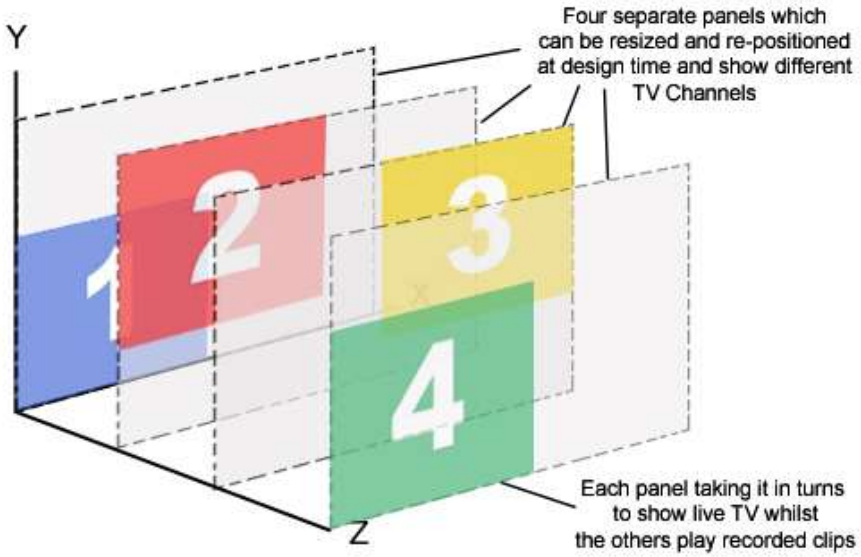


Figure 4.11: Diagram showing overview of how Collage display mode works

This display mode utilises the .NET Framework in-built function to allow you to change the opacity of a form and the transparent key of a form. The original design was to use a separate form for each Collage screen therefore allowing the user to configure these two elements independently for each form and hence each collage element. As the TV capture card that I am currently using allows only one mono output of the TV, it would require three additional cards to be able to show four live inputs of TV. As at this stage it was more a proof of concept rather than a finished piece and also not necessarily all computers are equipped with sufficient PCI slots for all four TV capture cards and graphics and sound cards as well, I decided against using four cards instead opting for the idea of recording the TV output and playing it back on three of the collage screens whilst the remaining screen was showing live TV. I would then alternate the screen showing the live feed so that each screen got the same time showing the live feed whilst the others showed a recording. This would still give the impression that all four feeds were live.

The recording of the television to a file is relatively straightforward using Brain Low's Capture Class library. The capture class has three methods responsible for recording the media stream to a file; these are *cue*, *start* and *stop*. When called in this order the TV is recorded to the file, which is set by the property *Filename*. I use an array of four file names, one for each of the panels so allowing each panel to have its own channel independent of the other three. Thereby at any one time one panel is showing live TV and recording it to file, whilst the other three are playing back the movie files that were recorded when they were showing live TV. I wrote a *playVideo* class, which I added to the *DShowNET* library of classes to make playback easier.

However technically it proved to be difficult to swap the television feed from one collage form to another. It is not possible to have more than one instance of the capture class at any one time and to destroy and create a new instance of the capture class for each collage form would mean there would be periods when the screen would go blank as it takes a few seconds for the picture to appear once the capture object is created and equally would create a lot of overheads creating and destroying objects on a regular basis. My original solution to this problem was to merely change the capture class of my application responsible for showing the live TV image stream to point to another panel control on each of the separate forms in turn. However maybe due to the Operating system I was working with (Windows 2000) this only intermittently worked sometimes triggering a Fatal Access Violation error in Win32k.sys of the operating system. As Microsoft does not provide meaningful error messages for their operating system merely hex addresses, I was at a loss as to how to rectify this problem so instead decide to seek an alternative method.

I decided keep the *capture* class pointing to the panel control to output the image stream but move the position of this panel to give the impression that this was a different collage screen with the other three collage screen swapping places also. However this also posed problems, as although I was able to change the position of the forms on the X Y coordinates I was not able to change the Z-order of the forms in relation to each other in terms of their overlaps. I explored using the *System.Windows.Forms.Control.ControlCollection,* which is the Collection of controls whose order determines the Z-order of the controls on the screen. However the *Form Control* is the top level of this collection and so is not able to help in the current context. Therefore I decided to put all four panels on one form so allowing me to use the *ControlCollection* as mentioned above. This did however mean that I was then not able to change the opacity and transparency independently for each Collage panel, as these are form properties so would be applied to all of them. The *ControlCollection* did not seem to have method by which one could move a Control up or down the collection order and although the *Add* and *Remove* methods of the *ControlCollection* may have worked in conjunction with each other to achieve the same result I decided that this could lead to problems in itself. Consequently I decided to use the existing methods *BringToFront* and *SendToBack,* which are inherited by all the controls on the form and as their names imply these move the control to the top and bottom of the *ControlCollection* respectively. By using either method on the four controls each time the live television panel swapped places I was able to preserve the Z-order if any of the panels were overlapping each other so to the appearance of the viewer nothing had happened.

4.4.2Interface Design

Figure 4.12 shows the interface for configuring the Collage display mode. The collage function provides the possibility of showing four independent television pictures at the same time on the screen. The position of each of these layers is controlled either by dragging the corresponding numbered box in the frame on the right or by altering the corresponding x and y fields on the left of the form. The corresponding size and Channel are also altered on the left. It is possible to reduce the size of each layer to zero so that this layer will not show when the display mode is launched provided there is at least one layer with a size greater than zero. At the bottom of this tabbed page is an attributes section this refers to all four of the layers by which it is possible to control

their opacity and the frequency by which the live TV feed is swapped between these four layers to give the impression that there are four live feeds with only one TV capture card.

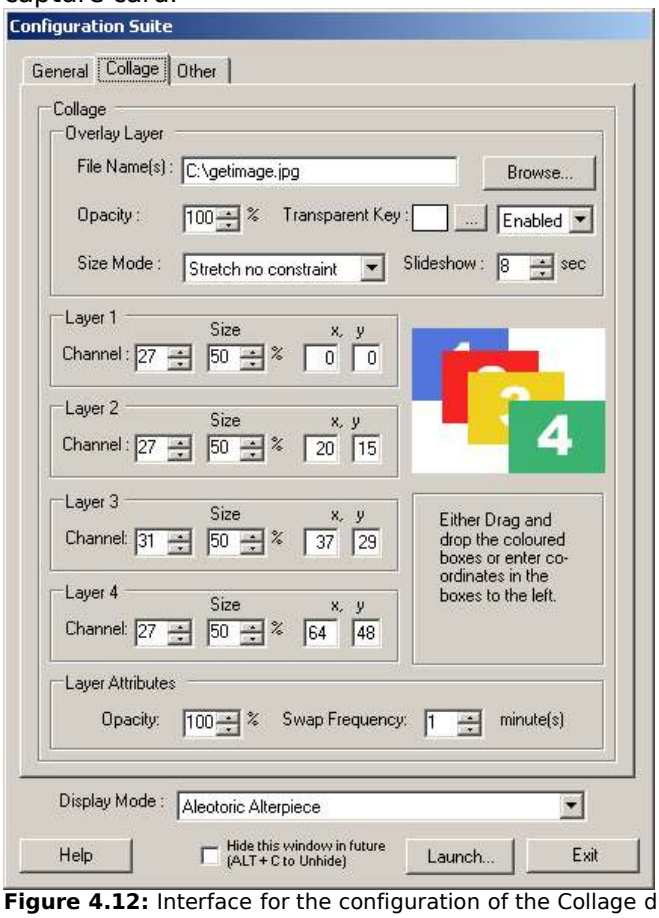


Figure 4.12: Interface for the configuration of the Collage display mode

4.4.2.1Image Overlay

The Overlay function provides the same functionality of overlaying an image or images over the collage of the television pictures as the Alterpiece overlay function does as detailed in section 4.2.

4.5Quick, Quick, Slow

4.5.1Overview

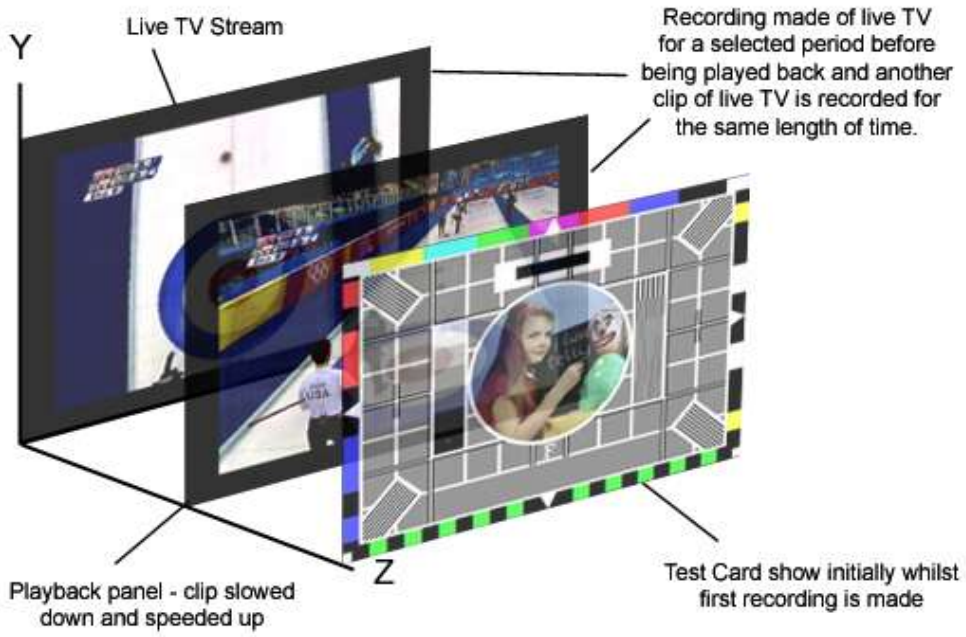


Figure 4.13: Diagram showing overview of how 'Quick, Quick, Slow' display mode works

Figure 4.13 above shows an overview of how the 'Quick, Quick, Slow' display mode works. There are essentially two phases to this display mode; the recording of the television feed and the playing back of the files recorded at varying speeds. Similar to the Collage display mode the recording of the television to a file utilises Brain Low's Capture Class library. The capture class has three methods responsible for recording the media stream to a file; these are *cue*, *start* and *stop*. When called in this order the TV is recorded to the file, which is set by the property *Filename*. In this case I use an array of two file names, with one being played back whilst the other is being recorded and vice versa. The playback uses the *playVideo* class I added to the DShowNET library. It contains two methods *SetRate* and *ModifyRate*, which both accept doubles and determine the speed that the recorded movie clip plays back at (with the value 1.0 being normal speed).

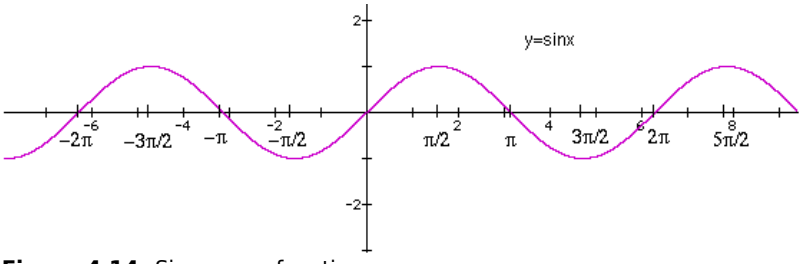


Figure 4.14: Sine curve function

I had originally thought of using a sine function such as the one in figure 4.14 to smoothly change the rate of the playback whilst the clip is playing. The peak and trough of the curve would effectively cancel each other out in terms of the speed of the playback so that it would play for the same length of time as if it had been set to normal speed throughout. However in practise on the computer that I was testing the application on, the regular modifying of rate of playback seemed to be too much for the computer at the same time as recording live television with the result that frames were missed in the playback. I therefore decided to simplify this so that half the duration of the clip is played at half speed and the rest is played at double the speed so once again cancelling each other out so that there is a seamless gap between the playback of one clip to the next. The clip duration is set by the property *`Duration`* in the *`frmBaseLayer`* form class.

4.5.2Interface Design

Figure 4.15: Interface for configuring the 'Quick, Quick, Slow' display mode

The 'Clip Duration' controls the length of each video clip that is recorded before being played back at varying speeds (see figure 4.15).

4.6Recontexualising Subtitles

4.6.1Overview

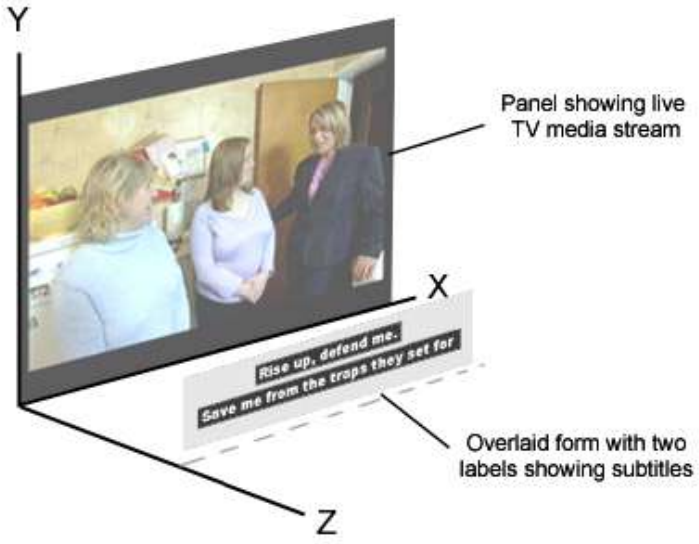


Figure 4.16: Diagram showing overview of how the 'Recontextualising Subtitle' display mode works

This display mode consists of two elements. The first and bottom layer is the live TV image, an instance of the *frmBaseLayer* form class. The second is separate form laid over the top, an instance of the *SubOverlay* form class. This exploits the .NET framework property of being able to have transparent forms with only two label controls being visible positioned over the bottom portion of the screen similar to TV subtitles. There are in fact three label controls but one of them is not visible as the *Visible* property is set to false. This has the *AutoSize* property set to true; thereby it automatically resizes itself given what is set by the *Text* property to be displayed. The subtitles are contained within a text file, which is read in line-by-line and word-by-word into this first invisible label. This automatically resizes itself and the dimensions of this label can be used to calculate the position of the upper visible label so that it appears to grow from the centre of the screen as more words are added. Once all of the words are used from the particular line read from the file then the contents of this label are moved to the lower visible label and the process begins again. There is a pause between each of these processes to mimic how television subtitles are put on the screen.

Just as in proper television subtitles there is a different coloured subtitled for each person speaking to make it easier to understand, this is also mimicked during this process. The text file can contain instructions in first two characters of the line on the colour of the text to be displayed and is set by changing the *ForeColor* property of the label control. A comment in the text file is the escape sequence *//* then the rest of the line is treated as a comment and ignored. If the first two characters are the escape sequence */c* then the line colour is changed to cyan; */b* also to cyan; */m* to magenta; */l* to lime; */y* to yellow; */w* to white; */g* to lime green.

4.6.2Interface Design

Figure 4.17: Interface for configuring Subtitle display mode

The Subtitles section of the Configuration suite allows for the user to select a new subtitle file. There is validation to check where this file exists before the display mode is launched.

5.Conclusion

I have adopted a top-down approach starting with the idea and then working out the method to realise this concept. Lots of other computer art seems to use a bottom-up approach with the artist first coming up with a method or adopting an existing method or concept in computing such as evolution, aesthetics and developing an idea that comes from this concept. To fully appreciate such work sometimes one must understand the procedure involved whereas I was trying to make an art piece that would stand-alone. I hope that I have been successful in this but the only true test is for the work to be seen by a wider audience.

5.1Evaluation

Dividing the application into display modes may in the future prove to be restrictive but at the time of coding seemed to be logical. All of the display modes work as I intended them to with the exception of the Collage display mode. At the time of coding I was not aware that if I change the opacity of the collage to less than 100% or apply an image overlay over the collage this would produce the recorded clips to flash when played back. This I think is due to the fact that at least on the machine I tested it on, it was able to calculate the opacity for each frame or the transparency of the overlaid image and still keep up with the normal frame rate of playing the clip. This may prove to be only a localised problem on this particular machine and with a faster machine this problem may not exist. This may also be true for the other slight performance issues with the ‘DoodleTV’ and ‘Quick, Quick, Slow’ display modes as they also show some degree of missing frames due to the high level of background processes taking place, i.e. recording live TV and face detection.

5.2Final Appraisal

The main objective of the project was to produce a flexible toolset to allow me to create artworks with television as the backdrop. I choose to implement:

- Image insertion
- Text insertion through subtitling or text as an image
- Slow/fast motion
- Collage of moving images

This has been successful as I was able to implement all of these features which as given me enough flexibility in the tools available to produce a number of different artworks (see Appendix B) without having to limit my imagination because of a too restrictive toolset. I may however need to extend this toolset but I have given myself enough flexible in the way in which this application is configured to allow for ‘future scoping’.

5.3Future Scope

It may in the future be interesting to explore the audio of the television output. This could possibly be in the form of making the picture either smaller or larger dependent on the volume of the noise from the television at any one time.

All the projects so far leave the media stream intact, it may in the future be interesting to try and change the media stream. By using a transform filter it may be possible to swap brown skin colour for white and vice-versa on the screen as it is being shown. This could then be developed as an artwork, exploring race on the television.

6.Bibliography and Resources

6.1Bibliography

‘Digital and Video Art’, Chambers Arts Library, by Florence de Mèredieu, English Language Edition Pub. Chambers Harrap Publishers Ltd 2005 ISBN 0-550-10170-5

‘Internet Art – The Online Clash of Culture and Commerce’, Julian Stallabrass, Pub Tate Publishing 2003, ISBN 1-85437-345-5

‘Internet Art’, Rachel Greene, Pub Thames & Hudson 2004 ISBN 0-500-20376-8

‘Digital Art’, Christiane Paul, Pub Thames & Hudson 2003 ISBN 0-500-20367-9

'History of Television', Grolier Encyclopedia, Article by Mitchell Stephens (http://www.nyu.edu/classes/stephens/History%20of%20Television%20page.htm)

'TV conquers remote Bhutan', Geeta Pandey BBC News, Thimphu, 10 March 2005 (http://news.bbc.co.uk/1/hi/world/south_asia/4332357.stm)

'Television', Wikipedia - the free encyclopedia (http://en.wikipedia.org/wiki/Television#Geographical_usage)

'Pipilotti Rist Biography', Electronic Arts Intermix Online Catalogue (http://www.eai.org/eai/biography.jsp?artistID=8817)

6.2Resources

Java Media Framework Resources

'The Java Media Framework Tutorial', The Interactive TV Web (http://www.interactivetvweb.org/tutorial/mhp/javamedia.shtml)

'JMF In Action - Code Samples and Apps', Sun Microsystems (http://java.sun.com/products/java-media/jmf/2.1.1/samples/)

DirectShow .NET Resources

'DirectShow MediaPlayer in C#', By Daniel Strigl, The Code Project (http://www.codeproject.com/cs/media/directshowmediaplayer.asp)

'Programming DirectShow applications in C#: A set of tutorials', The Code Project (http://www.codeproject.com/cs/media/PrgmngDirectShowappsCS.asp)

'DirectShow.NET', by NETMaster, The Code Project (http://www.codeproject.com/cs/media/directshownet.asp).

DirectShow tutorial in C++
(http://www.flipcode.com/articles/article_directshow01.shtml)

'DirectShow', by Michael Blome and Mike Wasson, MSDN
(http://msdn.microsoft.com/msdnmag/issues/02/07/DirectShow/default.aspx)

'Building the Filtergraph', MSDN Library
(http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directshow/htm/buildingthefiltergraph.asp)

P5 Glove Resources

P5 Yahoo Group - (http://groups.yahoo.com/group/p5glove/)

P5 Community Page -
(http://www.zzz.com.ru/index.php?area=pages&action=view_page&page_id=11#P5DLL
)

#P5Glove Wrapper for C# for use with Dual-mode driver -
(http://www.robotics4.net/Software/P5GloveSharp.aspx)

Face Recognition Resources

Computer Vision Research Code - (http://www.cs.cmu.edu/afs/cs/project/cil/ftp/html/v-source.html)

The Human Body Project –
(http://www.fuzzgun.btinternet.co.uk/rodney/humanbody.htm#Source%20Code)

'Face Recognition Demo Page', MIT Media Laboratory Vision and Modelling Group
(http://vismod.media.mit.edu/vismod/demos/facerec/)

Eigen-Faces and Codification -
(http://sirio.psi.ucm.es/PROYECTOS/EIGENCAR/eiweb2_e.html)

Face Recognition Using Eigenfaces –
(http://www.cim.mcgill.ca/~wsun/sa/project/node9.html)

SharperCV wrapper for Intel OpenCV –
(http://www.cs.ru.ac.za/research/groups/SharperCV/)

'The OpenCV Library', By Gary Bradski, Dr. Dobb's Journal November 2000 –
(http://www.ddj.com/documents/s=879/ddj0011k/0011k.htm)

Intel Open CV - (http://www.intel.com/technology/computing/opencv/index.htm)

Miscellaneous

'Transparent Desktop Video', By Sean McLeod, The Code Project -
(http://www.codeproject.com/cs/media/TransparentDesktopVideo.asp)

Gestural based interface – (http://users.rsise.anu.edu.au/~rsl/rsl_vr.html)

Appendix A -Spin-off Project

Introduction

Inspired by the work I had done on the Collage section of the 'Telly Vision' project (section 3.2), I decided to explore the concept of collage further in a separate piece entitled 'Media Montage'. The term collage was originally defined as the technique of incorporating fragments of commercially printed paper into artistic compositions but today any material fixed to a surface may be termed collage[21]. The word Montage is a synonym of collage but is invariable used to refer to the technique when applied to photography.

The development of collage can be traced through the development of Modern Art from the turn of the 20th Century. The Cubist artists Georges Braque and Pablo Picasso circa 1909 introduced the technique of collage into their work. In Picasso's 'Man With A Hat', 1912 (see figure A.1) the image is created with a piece of newspaper pasted onto the canvas with some additional colour and a few lines, so creating a radical statement as to what constituted Fine Art. The technique of Collage was later developed by artists of the Dada and Surrealist movements to include found objects. These techniques although developed nearly a hundred years ago are still relevant today as can be seen in the work of the contemporary artist, Wangechi Mutu. In figure A.2 one can see echoes of Picasso's "Man with A Hat" with the newspaper print now replaced by glossy fashion magazines. The longevity of the collage technique can perhaps be explained as it "recognizes multiplicity of form and content, duality, contradiction, juxtaposition, complexity ...and expansion as the essential qualities of modern life and modern art.[22]" Collage, as the art of change, is the newest and ideal exponent of transformation, transfiguration and metamorphosis. As such Collage may well reflect the very nature the world is made of.

[21] Glossary of the Online Catalogue Raisonné, National Gallery of Art, Washington, DC (www.nga.gov/gemini/glossary.htm)
[22] 'The Art of Collage' (http://www.silewen.com/collage/), quote taken from "The Collage Esthetic" in Art Forum, April Kingsley

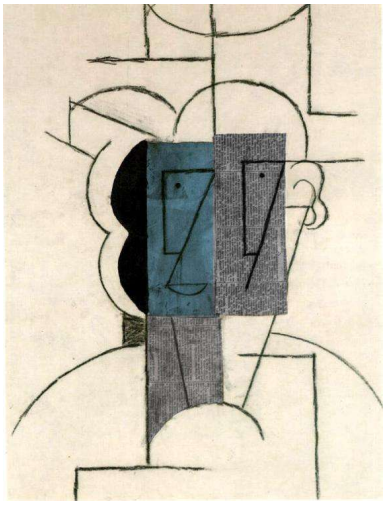


Figure A.1: 'Man With A Hat' (1912), Pablo Picasso Newspaper and Oil on Canvas

Figure A.2: Untitled (Classic Profile Series) 2003, **Wangechi Mutu,** Ink, collage on mylar

I have for some time admired the photographic montages that David Hockney made in the mid 1980's (see figure A.3). I thought it was an interesting concept to take the ideas developed by Pablo Picasso and George Braque in their Cubist paintings of showing objects from various angles but in the same image and apply them to a different medium, namely photography. With this idea in mind, I thought it would be interesting to extend this idea of cubist photography to moving images. This would allow me to create portraits and images similar to those created by David Hockney but with the added notion of time.

Figure A.3: 'Mother I, Yorkshire Moors, August 1985 #1' (1985), David Hockney
Photographic collage, 18 1/2 x 13 in (47 x 33 cm)

With this element of time it would be possible to add a narrative. With the juxtaposition of two or more moving images on the screen it is only natural to try and read these images as a whole. This concept has been used as a narrative device using split screen in such films as 'The Andromedia Strain' (1971) (see figure A.4) or more recently in the television series '24' (see figure A.5). Almost every episode of the '24' television series contains several multi-frame scenes. The majority are of people talking on phones or intercoms across great distances. At other times, one of the multiple sub-frames is used to show the point of view of a character in another sub-frame. Several contemporary artists such as the Turner Price nominated artist, Isaac Julien, have adopted this spilt screen device. In his piece, 'Vagabondia' (see figure A.6) each of the two sub-frames shows the mirror image of the other so the spectator becomes disorientated.

Figure A.4: One of the split screen scenes from 'The Andromeda Strain' (1971) Directed by Robert Wise, Written by Michael Crichton

Figure A.5: Split screen scene from television series '24', (2001 – 2004)

Figure A.6: 'Vagabondia' (2000) Isaac Julien, Double DVD Projection

In the previous examples of the use of split screen, there is generally a single soundtrack when the split screen is employed else the combined sound of all of the sub-frames would become confusing. This effect is however explored in the artist, Cory Arcangel's 'Geto Boys/Beach Boys' (2004) in which two videos by the two eponymous bands are played side by side creating an oddly harmonic synchronicity (see figure A.7). This idea of playing multiple images with audio simultaneously is taken to its logical conclusion in the work, 'Video Quartet' by Christian Marclay (see figure A.8).

Figure A.7: 'Geto Boys/Beach Boys' (2004), Cory Arcangel, DVD Projection

Figure A.8: Scenes from 'Video Quartet' (2002), Christian Marclay

'Video Quartet' is a 13-minute musical composition of nearly 600 separate film clips, on four simultaneous channels, projected onto a 40ft long screen. Video Quartet owes its existence to the recent emergence of real desktop editing software, and the artist's highly unconventional use of it. Marclay learned and used Final Cut Pro: "I sat in front of a computer for almost a full year," he said. With the concept and an abstracted

narrative structure in mind and starting with the films he knew, Marclay gathered scenes with music, performance, or sounds. He made bins for various categories (e.g., piano playing, singing, gongs, violins, tap-dancing), hand-building a database of clips to work from. Then he started constructing passages or scenes and built "bridges" between them. Along the way, Marclay would search out additional films and pull from them "the right combination of music and image." The result is exquisitely composed sound throughout, with absorbing images choreographed across four screens, flecked with just a touch of visual chance[23].

I recall back in 2002 attending a concert by the musician, DJ Shadow. DJ Shadow live stage show involves him mixing live tracks using a record turntable and DJ mixing desk. On the side of the stage there was also someone with a laptop mixing the live visuals in time to the music (see figure A.9). As part of his stage show at the time, DJ Shadow used an electronic drum pad with each individual drum programmed to play a section of an instructional video entitled 'Teach yourself to play the drums' on the screen behind him when hit.

Figure A.9: Live visuals of DJ Shadow's live set at Rock City, Nottingham November 2004

This idea of combining the visuals and the audio in a live performance intrigued me. I thought it would be an interesting idea to be able to produce something as sophisticated as Christian Marclay's 'Video Quartet' but in a live environment. The drum kit DJ Shadow used probably only consisted of around five drum pads so would be limited to only playing five clips whereas Marclay's piece involved 600 clips. A standard computer keyboard consists of over 100 keys but to remember what each key represented would be difficult and would still require some clips to use multiple key combinations so would be even harder to remember. Therefore I thought it would be appropriate to use a gestural interface whereby a particular gesture of the user's hand is mapped to a particular clip. This has the benefit of being more intuitive than using the keyboard and there are a great many more possibilities in terms of possible

[23] Taken from Review of Video Quartet at Grey.org, 24th January 2003 (http://greg.org/archive/2003/01/24/see_christian_marclays_video_quartet_at_paula_cooper_by_saturday.html

gestures than key combinations.

There are principally two types of gestural Interface. The first is a stereo vision-based system which uses cameras to track the hand in relative space and uses shape detection to determine the gesture of the hand i.e. is the fist clenched or the hand pointing, such a system has been developed by Robotic Systems Lab at the Australian National University[24]. The second is to use a data glove, which records the x, y, z, and pitch, roll and yaw of the hand as well as the finger bends.

I decided to use a data glove. The P5 glove is a relatively inexpensive glove-like peripheral device, based upon proprietary bend sensor and remote tracking technologies developed for the games market. Figure A.11 shows the location of the LEDs on the glove portion of the device, the receptor, which can be seen in the background of figure A.10, detects the location of these LEDs in space so that the x, y and z, and the yaw, pitch and roll can be calculated. The P5 glove has 6 degrees of tracking (X, Y, Z, Yaw, Pitch and Roll); 5 independent finger measurements; 0.5 degrees resolution (0-90 degree range); a 3-4 foot range from the receptor and 60Hz refresh rate[25].

Figure A.10: P5 Glove with Receptor

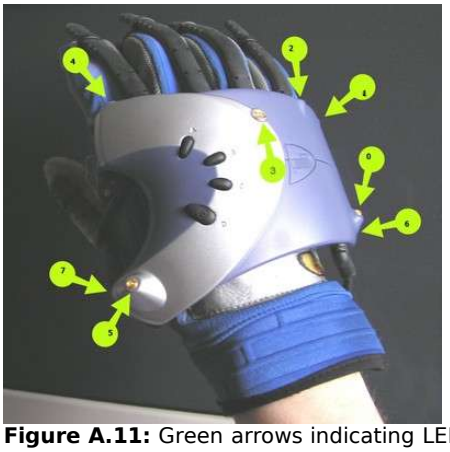


Figure A.11: Green arrows indicating LEDs on P5 glove

Due to the P5 glove being relatively inexpensive, a large online community of amateur programmers and computer enthusiasts has developed with their own yahoo group and community web page (see section 6 for more details).

This community includes Simulus, a collaborative project between Melbourne based composers Steve Adam, Ross Bencina and Tim Kreger. The ensemble combines studio aesthetics with live improvisation, performing on instruments created through a diverse range of electroacoustic software running on Macintosh and Windows computers. Simulus is developing interface systems for musical performance using the P5 virtual reality glove, interfacing with existing computer music software such as SuperCollider,

[24] Taken from 'A Gestural Interface to Virtual Environments' Robotic Systems Lab, July 2001 (http://users.rsise.anu.edu.au/~rsl/rsl_vr.html)
[25] Specifications taken from official reseller of P5 Glove (http://www.vrealities.com/P5.html)

Max/MSP, Reaktor, and AudioMulch[26].

Objectives

As had been the ethos of the 'Telly Vision' project, I did not want to tie myself down to one particular idea when developing an application. I decided to try and make this application as configurable as possible and thereby flexible as possible. This would later give me as much scope as possible when creating artwork using these tools. The principle objectives of the application were to:

- Play either movie clips (mpeg), or sound clips, or show image file (jpeg)
- Change their relative size and position on the screen
- Change the playback speed of the sound and movie clips
- Play movie or sound clips once, or repeatedly in a loop, or triggered by the Gestural interface using the P5 Glove
- Record the position of the P5 glove in space (x, y and z) and pitch, roll and yaw

System Design

Software Tools and Libraries

The system design borrows heavily from the work I have already done during the development of the 'Telly Vision' project. Once again I am using the C# programming language and the DShowNET project, a library of DirectShow interfaces rewritten in the C# programming language as the interop layer.

I do not use the official driver for the P5 Glove instead opting for a dual mode driver (Absolute Mode and Relative Mode) developed by Carl Kenner (see Resource section for link). This driver provides enhanced functionality compared to the factory driver. The new features include compatible with existing relative or absolute mode P5 software, new APIs which can be called from Delphi, Visual Basic 6, Java, C, or Visual C++ 6, Relative or Absolute mode, more accurate position and rotations, and absolute finger bend values. I use this in conjunction with P5Glove#, a managed library that exposes the features of P5 to .NET applications. The library is a managed C# wrapper to the C++ SDK of Carl Kenner's driver (available at http://www.robotics4.net/Software/P5GloveSharp.aspx).

[26] 'Simulus - P5 Glove Developments ' May 2004 (http://www.audiomulch.com/simulus/p5glove/)

Architecture

Figure A.12 illustrates the system architecture of the Media Montage application. The DShowNET Interface Library is the interop layer so allowing the Media Monatge application above to access the DirectShow COM components. The P5Glove# wrapper allows the application to access the functionality of the P5 Dual Mode Driver.

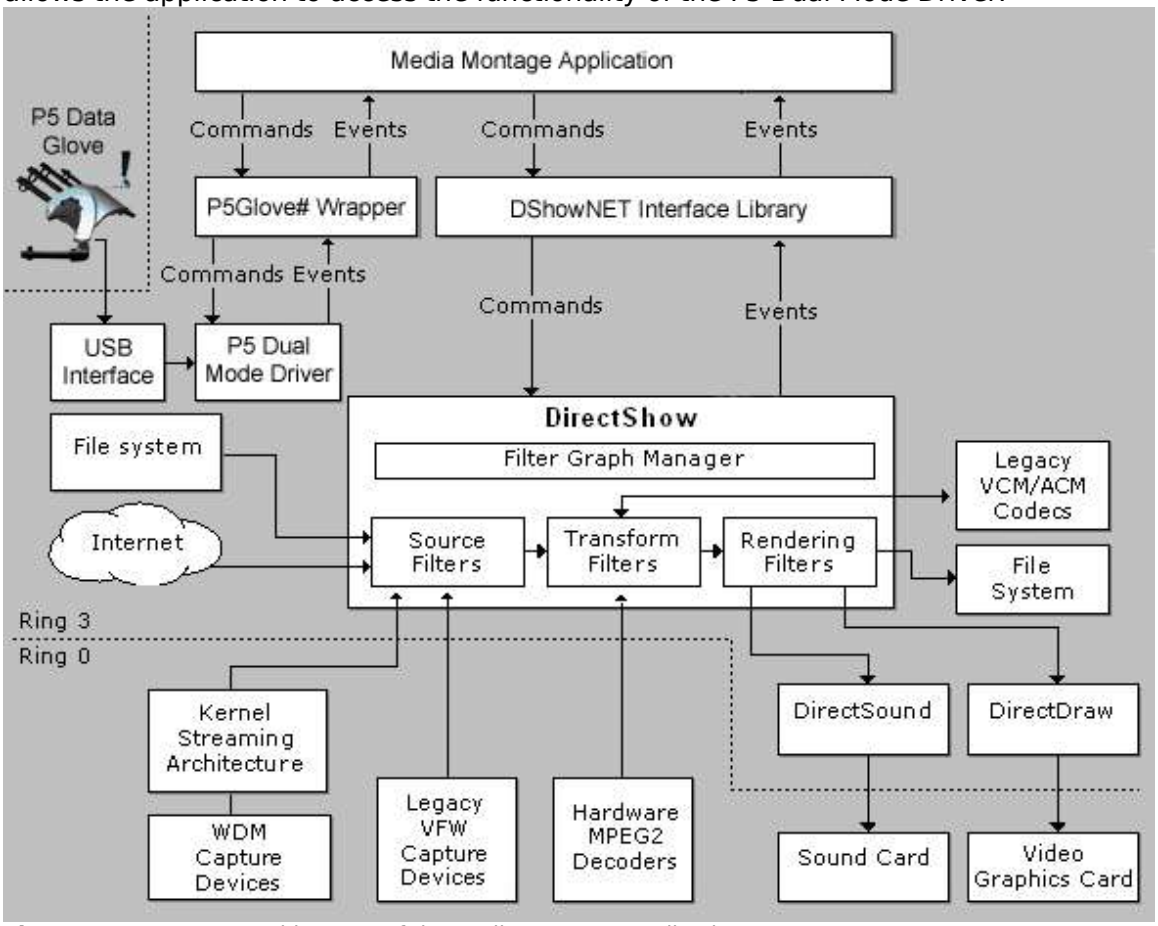


Figure A.12: System Architecture of the Media Montage application

Implementation

Overview

There are two main form classes within the application; the first is *`frmConfig`* class, which is responsible for the configuration of the montage, and the second is *`frmMontage`* class, which is responsible for displaying the montage. In the *`frmConfig`* class, a *`videoPanelCollection`* Collection class is used to hold a number of objects of the *videoPanel* class, which inherits from the pictureBox control. Each panel in the final montage has its own *`videoPanel`* object, this stores in a series of properties all of the information which is gathered during the configuration stage for each panel, including the file name (*`FileName`*), panel size (*`PanelSize`*), the playback speed

(*PlaybackSpeed*), as to whether it is played once, looped repeatedly or triggered by the P5 Glove (*PlaybackAction*) and the P5 Glove settings that will trigger the clip to play. Once the configuration is complete, the instance of the *videoPanelCollection* is passed to an object of the *frmMontage* class. The *PanelResize* method for each *videoPanel* object contained within this collection is called which resizes the panel in relation to the full screen. The *Activate* method is also called for each *videoPanel* object. This method creates a *playVideo* object for each *videoPanel* object, which is responsible for playing the clip be it either image file, sound file, or movie file. If the *PlayBackAction* property of the *videoPanel* object is set to P5 trigger then a separate thread is created to listener for the P5 Glove finger bends, X, Y, Z, Pitch, Roll and Yaw measurements. If these measurements match the property settings for that particular *videoPanel* object then the specific clip is played once and the thread begins again once the clip is finished.

Forms usually only have one movie clip per form so similar to Windows Media player, but as I needed to have multiple clips per form this posed a problem particular when trying to make clips play repeatedly. Usually when playing media clips, the *WndProc* of the form is overridden. The *WndProc* is used to process Windows messages. All messages are sent to the *WndProc* method after getting filtered through the *PreProcessMessage* method. The WndProc method corresponds exactly to the Windows *WindowProc* function. Usually when a clip has finished a message is sent to the *WndProc* and the form handles it appropriately namely in this case replaying the clip. However as one can only have one *WndProc* per form, it would be difficult to know which clip had finished playing. Consequently I decided to alter the *playVideo* class, which I had added to the DirectShow project as part of the Telly Vision application. I added a thread *stateChecker* which checks every second to see if the media clip's duration (*IMediaPosition.get_Duration*) and the current position in the clip (*IMediaPosition.get_CurrentPosition*) are the same, if so it will raise the event, *stopped* within the *playVideo* class. This can then be handled by the object, which created the instance of the class, the *videoPanel* object. This means that I can have multiple clips playing on the same form with each object responsible for repeating its own media clip.

Interface design

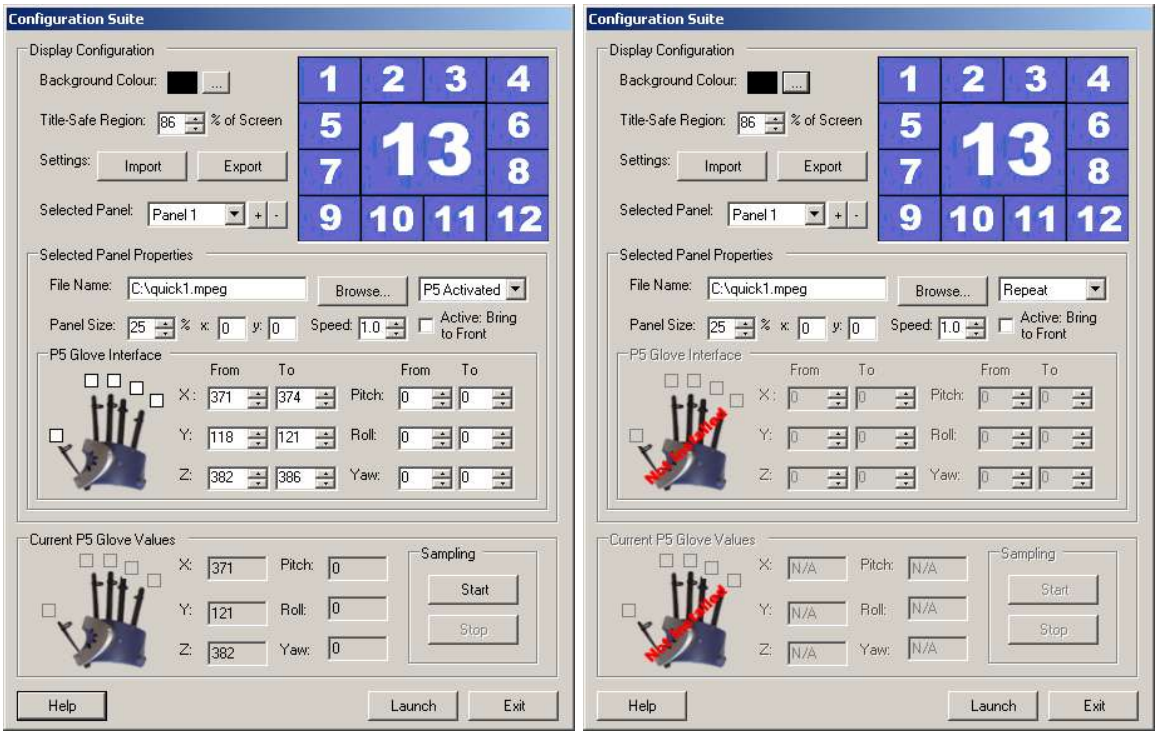


Figure A.13: Configuration Suite interface – when P5 Glove present (left), without (right)

Figure A.13 shows the interface for configuring the Media Montage. The interface is divided into three sections. The first and top section controls the general configuration of the montage. The background colour can be changed just as the title-safe region – the percentage of the screen, which is deemed to be viewable. It is possible to export all of the settings to a settings file or import the settings from a settings file. The layout of the montage is shown on the top right. The elements of the montage are described as panels within the configuration form.

The middle section shows the specific configuration for the selected panel. The panel can be selected by either selecting the panel name from the drop-down list, which lists the available panels or by clicking on the relevant panel in the montage area on the top right of the configuration form. By pressing the '+' button to the right of the selected panel drop-down list will create a new panel, which will be added to the montage layout. By pressing '-' button will delete the selected panel. It is not possible to delete the panel named 'panel1'. Once a new or existing panel is selected, the middle fields are populated with the configuration for this particular panel. It is possible to change the media file that the panel will play, its size, playback speed, location and whether the clip will be played once, repeated or activated by the P5 Glove if it is present else this option will not be available and the user will be presented with a configuration form similar to the one in figure A.13 on the right. If the P5 Glove is present and the option is selected than the lower portion of this middle section will become enabled. In this section it is possible to configure the particular gesture that will be used to trigger this particular media clip to play. The user can input values into the corresponding fields for finger bends, X, Y, Z, Pitch, Yaw and Roll, or they can choose to sample the existing P5

Glove movement by pressing the Start button at bottom right of the form to begin and the Stop button to end sampling. The corresponding values of the P5 Glove made during this sampling period which also can be seen in the lower section of the from will then be added to the middle P5 Glove Interface section.

Conclusion

I think in some ways this project has been more successful than the Telly Vision application as it offers more flexibility in the toolset allowing for all the media formats to be used; images, movies, and sounds whereas I restricted the Telly Vision application to discreet display modes. Due to its flexible nature I have yet to fully explore the potential of this application however I am aware that that there maybe restrictions imposed by the hardware due to the memory and processor intensive nature of such multimedia environment.

Appendix B -Selected Artworks produced

Self-portrait

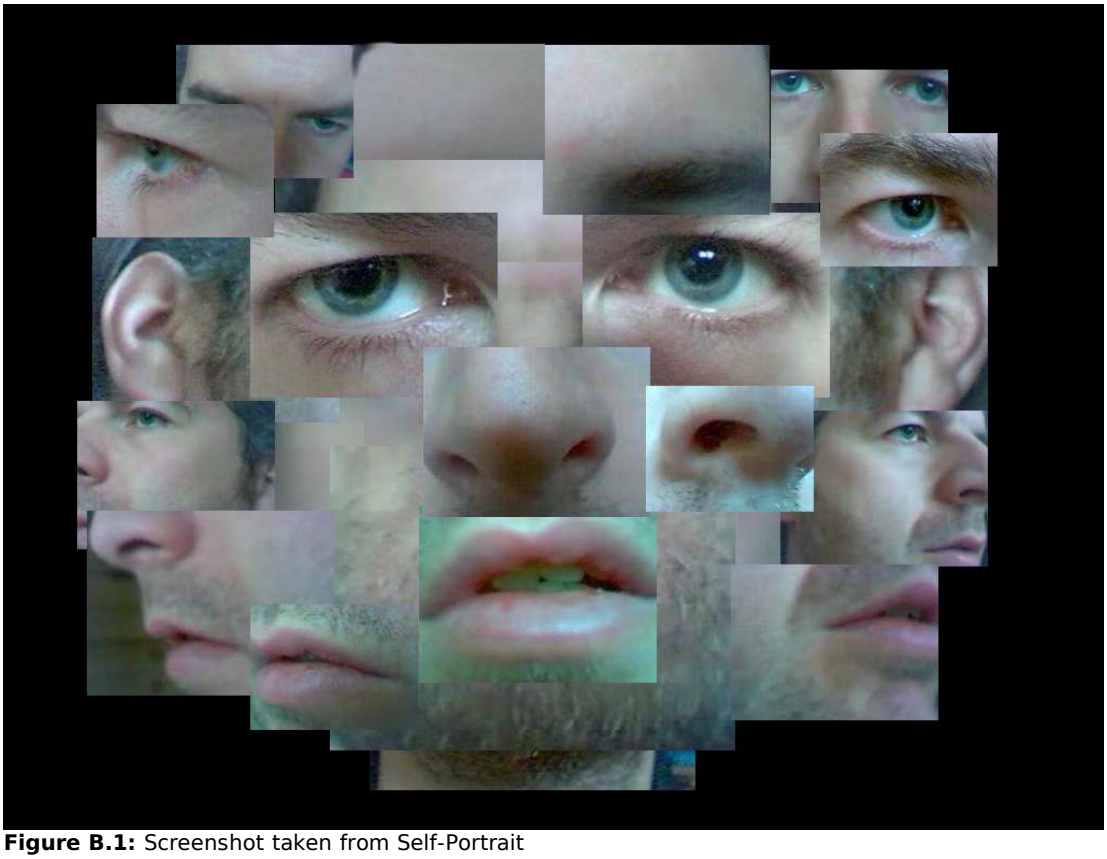


Figure B.1: Screenshot taken from Self-Portrait

Using the Media Montage toolset, I produced the work above. I filmed my face from various angles and degrees of close-up to produce a series of movie files. Using the Media Montage application I have collaged these various movie clips into one image. It consists of 20 separate panels with the Media Montage application; each one has the clip set to repeat so the entire image is constant moving; the eyes blink etc so giving it more a sense of being alive.

Musical Montage

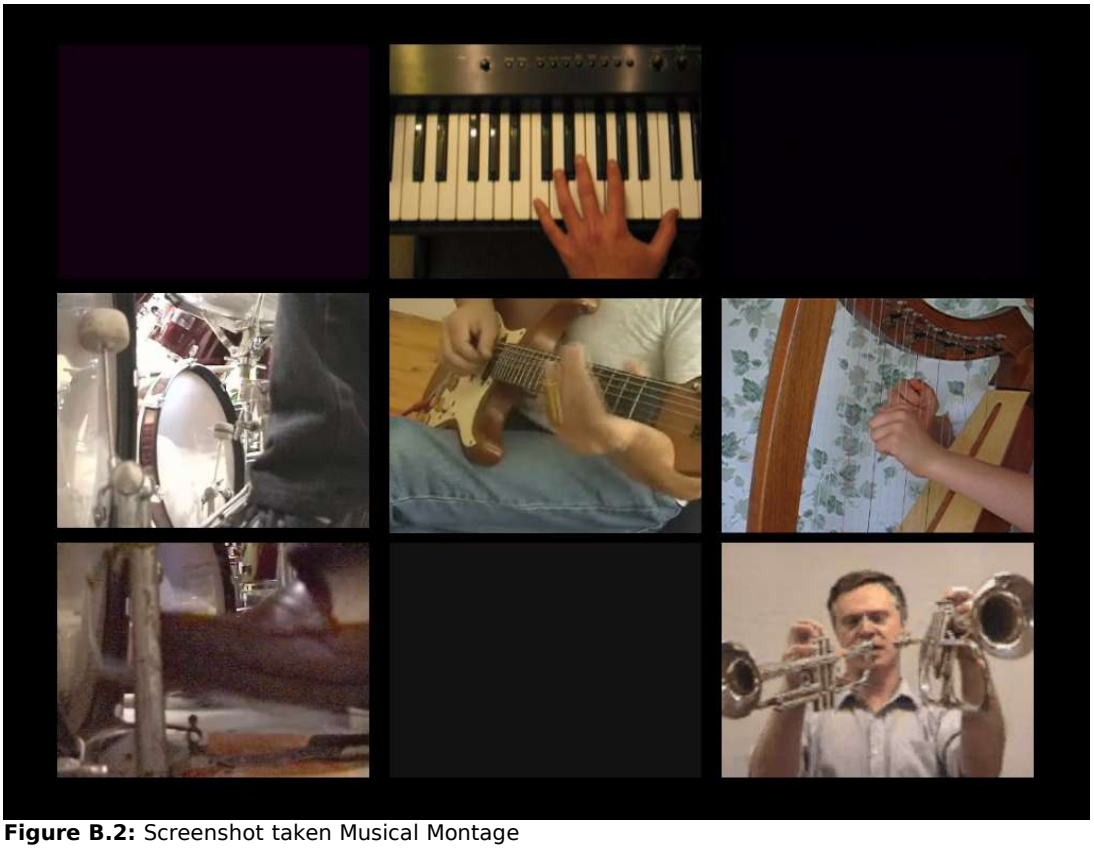


Figure B.2: Screenshot taken Musical Montage

This piece also uses the Media Montage application. It consists of nine separate movie clips although only six are visible in the screenshot above. Each movie clip is of a musical instrument being played; base drum, snare drum, drum-pad, keyboard, electric guitar, acoustic guitar, violin, harp and trumpet. The Media Montage application has been configured in such a way that if a particular gesture is made whilst wearing the P5 Glove then this will trigger the specific clip to play. As all of the clips have audio as well as the visual element, it effectively makes it into a musical instrument.

Ode to Mary Whitehouse

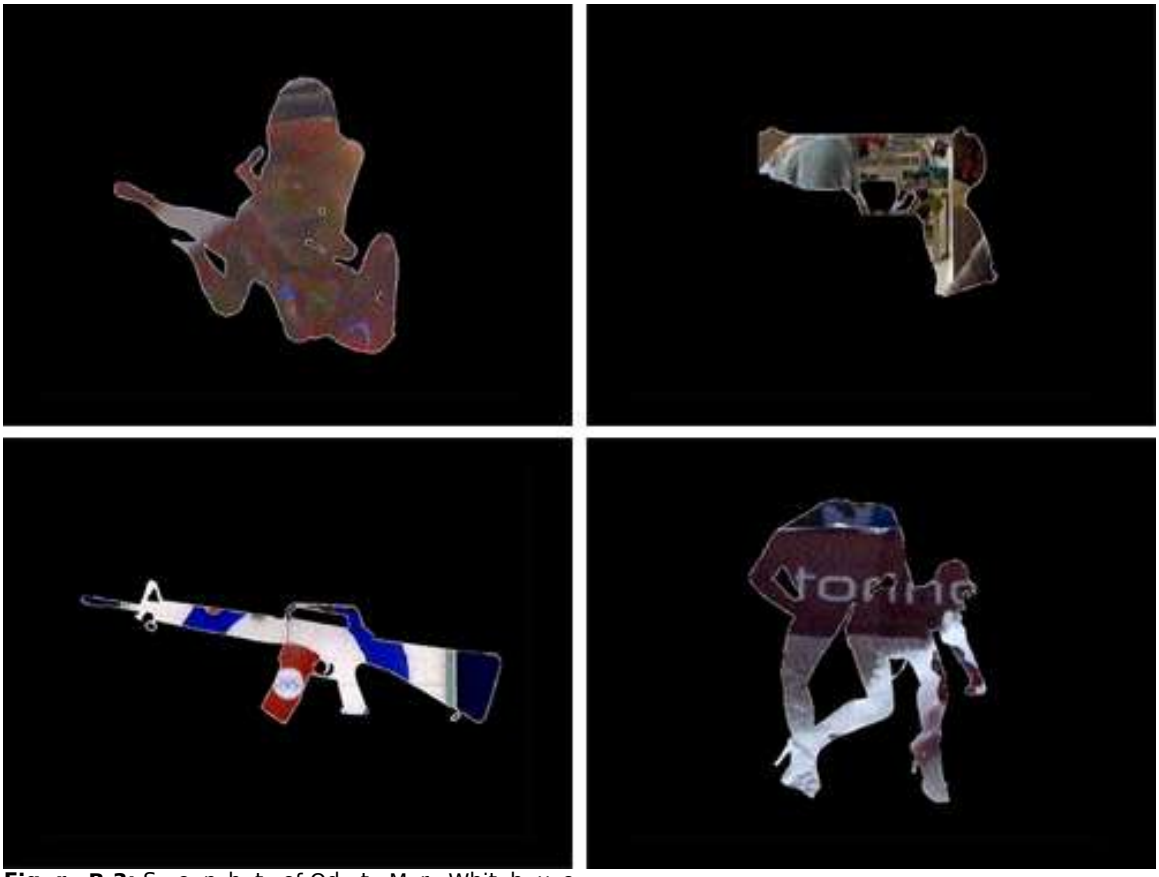


Figure B.3: Screenshots of Ode to Mary Whitehouse

During the 1980's Mary Whitehouse campaigned against sex and violence being show on television amongst other things. I therefore thought it would be interesting to replace the actual sex and violence with the picture from the television. This piece is produced using the Alterpiece display mode with the Image overlay set to show a series of cutout images similar to the ones above.

Subtitled

Figure B.4: Screenshot taken from Subtitled

This piece uses the Subtitle display mode of the Telly Vision application. The subtitles are taken from the film, The Passion of Christ, which was a highly emotive movie directed by Mel Gibson (see figure B.5). It is claimed to be the most accurate and realistic adaptation of the story of Jesus Christ. The actors speak Aramaic throughout so English subtitles have been added although Mel Gibson originally wanted the film to be released without them but changed his mind at the last stage. It is now considered by many that there are now too many subtitles in the film[27]. With this as background I thought it would be interesting to use these subtitles by placing them over live TV. Without the associated images much of the power of the film is lost if it is told only through the subtitles.

Figure B.5: Scene from The Passion of the Christ (2004) directed by Mel Gibson

[27] Taken from http://www.killermovies.com/p/passion/articles/3226.html

I’m also Dreaming of a White Christmas

Figure B.6: Screenshot from “I’m also dreaming of a White Christmas”

This piece uses the Alterpiece display mode in Telly Vision application. It references Richard Hamilton’s "I’m dreaming of a white Christmas" (see below). In my piece, Bing Crosby has been cut out, revealing an off-tune television picture below. The static of the off-tune television picture is often described as ‘snow’ hence it might be a white Christmas as it is snowing.

Figure B.7: I'm dreaming of a white Christmas, Richard Hamilton (1967-68) Oil on canvas

And they say there is nothing good on TV

Figure B.8: Screenshot from 'And they say there is nothing good on TV'

This piece uses the Doodle TV display mode in the Telly Vision application. It draws a series of silly glasses and moustaches over the faces on the screen.