Perceptual Correspondences of Abstract Animation and Synthetic Sound

by

Adriano Abbado

Diploma of Electronic Music Composition
Conservatory of Milan, Italy, 1977

Submitted to the Media Arts and Science Section
in Partial Fulfillment of the Requirements of the Degree of
Master of Science
at the
Massachusetts Institute of Technology

June 1988

Signature of the Author ______________________
Adriano Abbado
Media Arts and Science Section
May 6, 1988

Certified by [illegible]
Tod Machover
Assistant Professor of Music and Media
Thesis Supervisor

Accepted by [illegible]
Stephen Benton
Chairman
Departmental Committee for Graduate Students

Perceptual Correspondences of Abstract Animation and Synthetic Sound

by

Adriano Abbado

Submitted to the Media Arts and Science Section on May 6, 1988
in partial fulfillment of the requirements of the degree of
Master of Science

Abstract

Composing with synthetic sounds involves an approach that is very different from the usual way of conceiving traditional Western music. In fact, it is usually problematic to organize synthetic sounds according to traditional principles. However, if it is possible to establish links between audio and video dynamic events, then the visual language can help to organize an audiovisual piece, as well as a music composition of synthetic sounds.

In order to propose a specific model for image/sound correspondence, I first analyzed several important abstract films that were related with music. None of them, however, has ever considered a relationship between musical timbres and visual objects.

I then composed an audiovisual piece in which visual objects were modelled after specific sonic objects. The visual events were then used to determine the organization of the sounds. This unusual process gave me the possibility to identify and visualize categories of synthetic sounds, as well as to take advantage of typical features of visual organization such as spatial repetition and localization, and to apply them so as to reinforce auditory perception.

An experimental study was then carried out to better understand objective opinion about correspondences between visual shapes and sonic timbres. The results of this test showed that simple transformation procedures applied simultaneously in the visual and sonic domain can be perceived, and may lead to a unified perception of the two domains.

Thesis Supervisor: Tod Machover
Title: Assistant Professor of Music and Media

Index

1 Introduction

"Certainly I have an aversion to everything that is demonstratively programmatic and illustrative. But this does not mean that I am against music that calls forth associations; on the contrary, sounds and musical coherence always arouse in me ideas of consistency and colour, of visible and recognizable form. And vice versa: I constantly combine colour, form, texture and abstract concepts with musical ideas. This explains the presence of so many non-musical elements in my compositions."

György Ligeti, 1968

I have always been interested in associations between different areas of thought. It is part of my natural way of thinking to see if certain ideas can function when transposed to a different context. I often attempt to abstract concepts from their original context and try to apply them to others. This method often stimulates new ideas, and sometimes I use it as a tool for the creation of art works. Specifically, I often seek to establish parallel behaviors between aural and visual events, the elements of music and visual arts.

This is particularly true in the realm of abstract visual objects and synthetic sounds. I think that it is possible to establish links between perceptual categories of synthetic sounds and abstract visual forms. This conviction has led me to consider different areas of application, mainly through the creation of a new type of audiovisual work, based on the interaction between *sounds* and *visual objects* rather than music and images. Also, the method that I propose in this thesis is based on **visual** language, and can lead to a new way of thinking and creating music compositions based on timbres.

This thesis is divided into three parts. In the first part I briefly analyze the origin of audiovisual art. In the second section I describe a new audiovisual composition of mine in which abstract visual and aural objects are related to each other according to the principles that I propose, and discuss the consequences of such an approach. Finally, I report about an experimental study that I have conducted in order to understand whether or not subjects are sensitive to correlations between abstract visual/aural events, and which such correlations seem to be the most powerful.

2 Music and Abstract Animation: the Origin

2.1 Introduction

In this section, I will briefly review the steps that have led to the creation of a new, truly audiovisual, art form. Although the roots of this art can be found in the XVIII century, it was only at the beginning of this century that artists have begun to create interesting art works, probably because of the invention of the film medium. However, even if several artists have been involved in this new genre, none of them, to my knowledge, has ever related sound as such, considered as a specific and independent entity, to image, but always the music as a whole.

2.2 Music and Image

The interest for the relationships between music and images has been alive for a long time. The first audiovisual machine was in fact the *Clavecin Oculaire* made in the XVIII century by Father Louis-Bertrand Castel, who built an organ that could simultaneously emit music and colors (colored stripes to be precise), so that each note corresponded to a color [1]. In the following century, other machines and esthetic theories were elaborated by various artists (Jameson, Kastner, Bishop) [2]. But it was at the beginning of our century that the audiovisual relationship began to be seriously considered in the art field. In 1911, Alexander Wallace Rimington built a color organ that was used by the Russian composer Alexander Skriabin in his work Prometeo, a musical composition with projection of colors on a screen (fig. 2.1). Skriabin's idea was quite simple: he associated a color to each note of the western scale, creating a range of 12 colors (fig. 2.2). It is interesting to note that most of the *audiovisual* artists used the music as a reference point. Nobody tried to render paintings in music, the process has always been music to image. Wallace also wrote a book called Colour Music. The art of mobile colour. The association color-music has been quite common among artists. In my work Isomorfismi Suono Luce, I translated into animation the theories and the paintings of the Italian artist

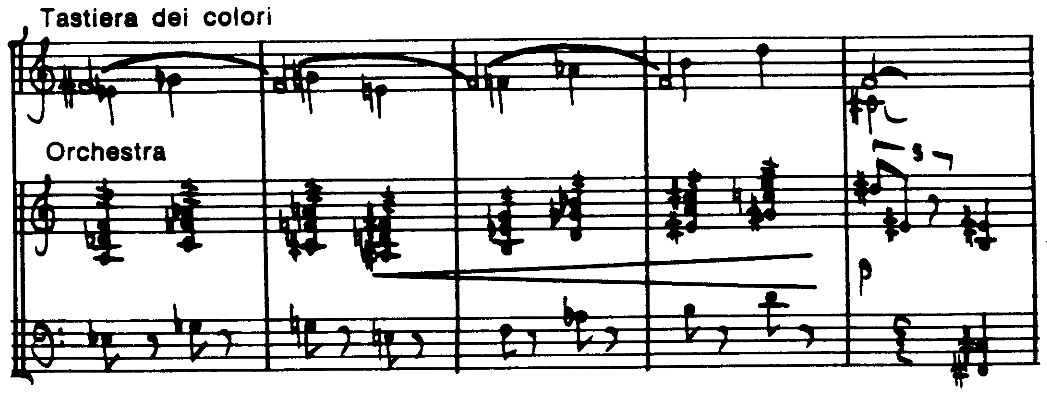

fig. 2.1

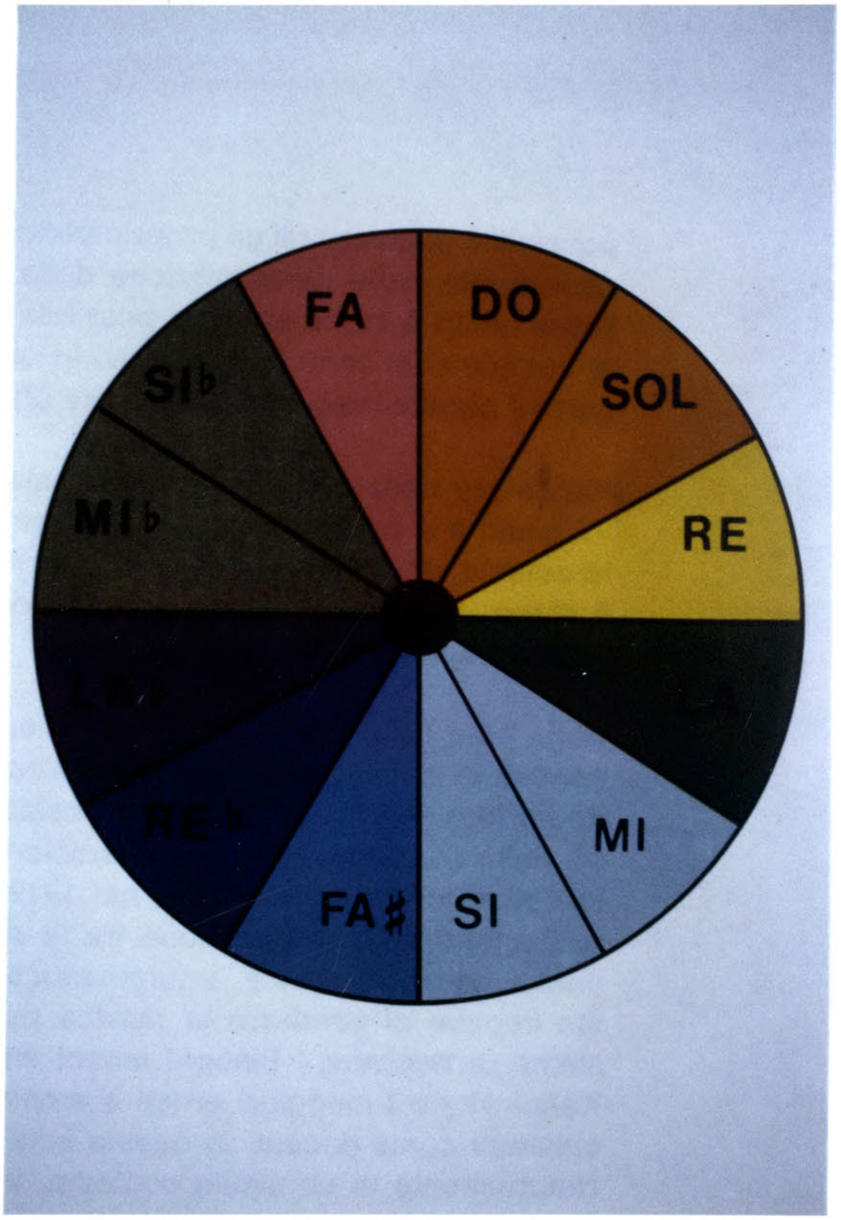

fig. 2.2

Luigi Veronesi, who himself was interested in relationships between color and pitch [video Abbado 1985] [Veronesi]. However, such relationships cannot be understood unless a specific training is undertaken, since we are much more sensitive to pitch changes than color changes. Also, the usage of the western musical scale is completely arbitrary and I believe that a better approach would at least consider the whole range of audio frequencies, rather than a scale.

The first artists that took advantage of the audiovisual medium *per excellence*, film, were the Italian Futurists Arnaldo Ginna and Bruno Corra [3]. In the second decade of the century, they created six films in which the colors followed the music. For example, in the third film they used both the Canto di Primavera by Felix Mendelssohn and a theme taken from a Chopin waltz. Ginna and Corra also invented a technique, direct painting on film, that was later developed by other artists such as Len Lye and Norman McLaren [Rondolino]. Unfortunately, all their films have been lost, and therefore a judgement is this case quite difficult. In the same spirit was the French Cubist Léopold Survage, who wrote an article in which he claims that the *colored rhythm* is not an illustration of music, but an autonomous art form [4]. The concept of visual rhythm, not only within the frame as in the traditional paintings, but above all over time, is very important for Survage and other artists of this period, such as Viking Eggeling and Hans Richter [Rondolino]. I think that this process is particularly interesting because it abstracts the concept of rhythm from music, and succesfully transpose it into the visual domain. For Survage, the fundamental element of this new art is the *visual colored form*, which has an analogous form to the sound in music. The first abstract film that has survived is Diagonal Symphony, made in 1921-23 by the Swedish Dadaist Viking Eggeling (Fig. 2.3). Eggeling was an extremely meticulous artist, and was particularly interested in the parallel of rhythm in music and rhythm in film language.

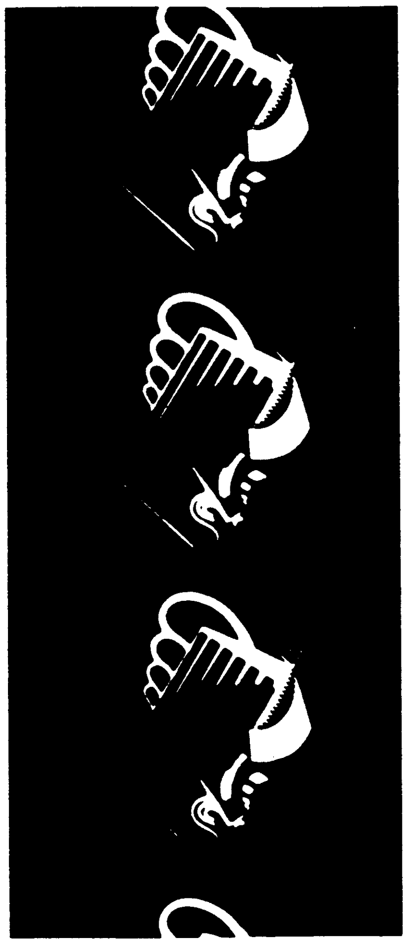

Fig. 2.3

A rigorous analysis of Diagonal Symphonie, made by Louise O'Konor, shows that there are forty elements that are interdependent: an extremely accurate and complex work that recalls the perfect organization of Alexander Nevsky, by the Russian director Sergei Eisenstein, a movie that was based on the contemporary composition of Sergei Prokovief. In fact, in Diagonal Symphonie, only a few basic signs are shown with no movement across the frame in any direction, but with an incredible number of variations on the shapes. The combinations and transformations are always additive: the complex figures are composed of simple elements. Theme and variations is of course a musical concept, but the sequences of Diagonal Symphonie also have their own rhythm, increasing the musical character of the work. The structure of music is present in this silent film. Eggeling's work is traditionally considered as the first art work of abstract animation [Le Grice], [Mitry], [Rondolino].

The first abstract film with a *synthetic soundtrack* (i.e. not recorded) was Tonende Handschrift, by the Swiss Rudolf Pfenninger, composed in 1929. In Tonende Handschrift, Pfenninger painted the area of the film normally used for the soundtrack, therefore controlling the sound with a synthetic tool. This event is extremely important for two reasons. Pfenninger was the first artist to use the film medium as a generator of both visual and musical abstract events. Also, he didn't use old style music, like his predecessors, but created synthetic music that can certainly be considered on the cutting edge, at least for its innovative method if not for the artistic value. In this sense, he was the first truly audiovisual artist.

It is interesting to note that towards the end of the 1920's music visualization using abstract animation was quite popular in Europe, and it was regularly shown in public theatres. Several authors (Hans Richter and Walter Ruttmann, for example) developed special techniques such as the shooting of painting on glass, of drawings or of physical models. The most famous of them, Oskar Fischinger, was even hired by the Disney Studios to direct part of the famous film Fantasia. Fischinger, who produced an enourmous number of abstract films and can be considered as one of the masters, left the Disney Studios when he realized that the purpose of Walt Disney was to vulgarize masterpieces of classical music, rather than creating abstract films based on traditional composers' music. Fishinger's interest was mainly concentrated on the similarities between the structure of music, in terms of rhythm, melody and harmony, and the structure of abstract film. I think that his best film is Motion Painting, realized in America in 1946-49. Motion Painting, which was made by shooting a sheet of glass on which liquid colors were moving, was based on the third Brandenburg Concerto by J.S. Bach, one of the most interesting for its internal structure. In this concerto several musical elements overlap to each other, as often happens in Bach's music. Fischinger represented this structure by using several overlapping sheets of glass. The result is very impressive.

Other artists such as Len Lye, Norman McLaren, Laszlo Moholy-Nagy (Fig. 2.4), and after the war Harry Smith and the Whitney brothers, worked with music and abstract animation [Le Grice], [Mitry], [Rondolino]. They used not only films, but also electronic devices, such as oscilliscopes and cameras, as well as computers, in order to produce their animations. None of them, however, ever created animated abstract shapes in relationship with the timbres of the sounds they were using. Moreover, although many films made by the artists I mentioned were interesting and innovative both from of a technical and a visual point of view,

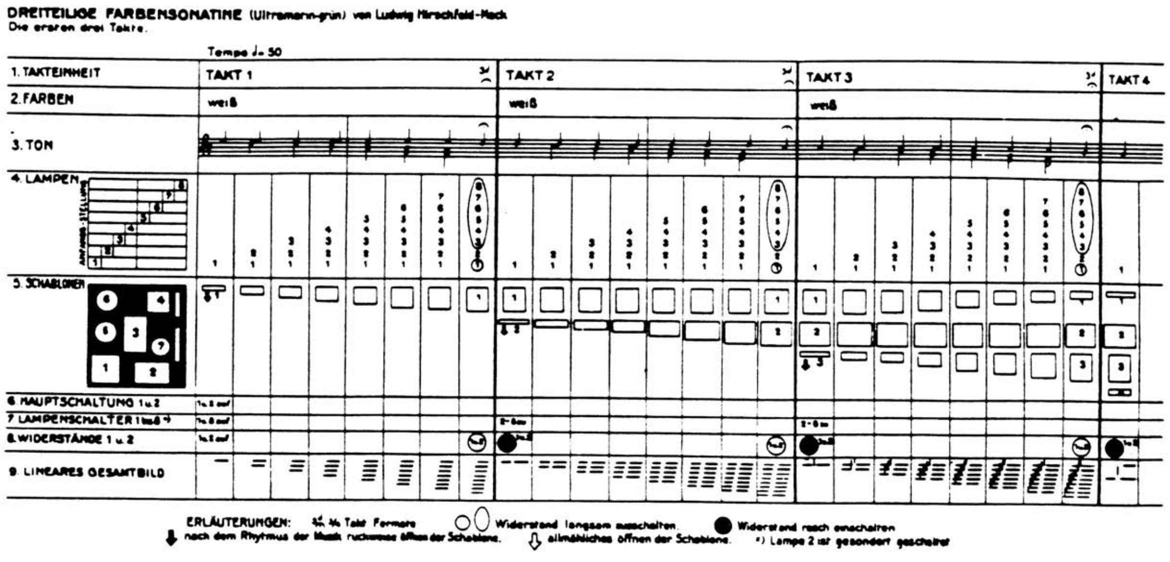
DREITEILIGE FARBENSONATINE (Ultramarin-grün) von Ludwig Hirschfeld-Mack
Die ersten drei Takte.
Tempo ♩= 50
1. TAKTEINHEIT
TAKT 1
TAKT 2
TAKT 3
TAKT 4
2. FARBEN
weiß
3. TON
4. LAMPEN
5. SCHABLONEN
6. HAUPTSCHALTUNG 1 u. 2
7. LAMPENSCHALTER 1 bis 8 *)
8. WIDERSTÄNDE 1 u. 2
9. LINEARES GESAMTBILD
ERLÄUTERUNGEN:
Widerstand langsam ausschalten.
Widerstand rasch einschalten
nach dem Rhythmus der Musik ruckweise öffnen der Schablone.
allmähliches öffnen der Schablone.
*) Lampe 2 ist gesondert geschaltet

fig. 2.4

I think that none of them created a music that was worth considering seriously. Fischinger did some experiment with *musique concrète*, but never created an art work using experimental music. The work I describe in the following chapter is on the contrary based on synthetic sounds, and has an approach that is highly based on computer music, to which I have devoted all my artistic energies.

3 Dynamics

3.1 Introduction

Dynamics is an audiovisual artwork conceived and realized with computers. It lasts 3 minutes and 3 seconds, and has been produced at the MIT Media Lab, using technologies belonging to the Visual Language Workshop, the Animation Group, the Film/Video Group and the Music and Cognition Group. In Dynamics, I have attempted to establish three fundamental correspondences between the audio and visual events: timbre / shape-surface attributes, spatial localization and intensity.

3.2 Timbre / Shape-*Surface Attributes*

Timbre in sound, and shape-*surface attributes* in image, are the most powerful perceptual determinants. Their correspondence should in my opinion form the basis of any new sound-image language.

Traditional western music has always stretched the importance of pitch and rhythm as parameters to be considered in music. Other cultures' music, however, has sometimes considered the timbre of sounds as an equally important element of music. For example, *tabla* players create complex timbres that recall the sound of voice (they even learn a vocabulary of syllabes before learning how to play the instrument) [McAdams and Saariaho]. Similarly, Tibetan monks' chants are based on modulations of their voice's timbre, beyond pitch.

In other words, some music has traditionally payed attention not only to pitch and rhythm, but also to timbre. It is interesting to note that in both cases I mentioned above, there is a strict correlation between timbre in music and speech, since speech is mainly based on timbre rather than pitch. However, musical timbres have become more and more important in Western music compositions during this century. The first composer who clearly claimed a major role for timbres was Arnold Schoenberg who invented the term *Klangfarbenmelodie* (melody of timbres) [1]. In this spirit, one of the original goals of this project was to create a composition using complex timbres. My musical interest is in fact highly concentrated on timbres, which has become one of the areas of major interest in contemporary music [Machover], [Stroppa 1985], together with spatial localization and movement. One of the great innovations that computers have brought to music has been the possibility to synthetize new sounds, immensely increasing the number of instruments a composer can deal with [2]. However, composing with timbres involves an approach that is very different from the usual way of conceiving traditional music. One of the goals of Dynamics was to understand in which way the visual language can complement and clarify the composition of a musical work based on the complex notion of timbre rather than pitch, as in traditional music [3].

In the visual domain, shape is the element that defines an object. However, shapes alone are probably not sufficient to express the perceptual characteristics of timbres. Therefore, I also used other properties, such as the variation of specular reflectance of the surface, color and lightening, to enhance the expressive power of a visual object. Texture mapping would have been very useful. In fact, it would have permitted me to further enrich the visual appearence of the objects I created. However, the program I used, Anima (written by Bob Sabiston and running on an Hewlett-Packard "Renaissance Box" at the MIT Media Lab) although extremely powerful, didn't allow me to use this interesting feature.

The procedure for mapping shapes onto timbre was of primary importance in this project. Although I believe that there is a common agreement about attributes that refer to both sounds and visual objects (for example *metallic*), at least among people of our culture, the relationships I used in this composition were completely subjective. I associated harmonic sounds with smooth shapes and inharmonic sounds with jagged shapes (on this subject it is interesting to know that W. Kandinsky associated keen colors with sharp forms, and soft colors with round forms [4]). This is because partials in harmonic ratio usually sound to me as *non-aggressive*, that in turn I identify with the idea of smooth, while I imagine inharmonic sounds as irregular *aggressive* objects [5]. That was not a general rule, however. It is extremely difficult, in fact, to set such rules and formalize the process I used for the creation of the objects. Sometimes, one rule contradicted another one. For example, white noise was represented as a highly irregular, bumpy and shiny object. Nevertheless, filtered white noise (high frequency filtered out) was represented as a much more regular shape, although still shiny. However, I usually followed my own feeling and intuition to link an aural object to a visual object, rather than following a precise scheme of representation. I think in fact that creating a rigid method of correspondence becomes a limitation that is then reflected is the finished work. I tried this kind of approach in other previous works of mine [video Abbado 1985], without getting any satisfactory result. Therefore, I never looked at the spectral content of the sound, but always created the objects while paying close attention to perceptual sound (but also visual) attributes such as *harsh, wet, cold, metallic, viscous, spongy, granulous, opaque* [Ehreshman and Wessel], [Grey], [Wessel].

3.3 Spatial Localization

Another correspondence I established was between the position of the audio/visual sources in space. Spatial localization has been in my opinion one of the most important and innovative features in contemporary music. Compositions using space placement are already found in the Middle Ages, although with rudimentary techniques. Edgar Varèse and Henry Brant have been important composers in this century to take advantage of the notion of space in a more complex way [6]. Many post-war composers, like Karlheinz Stockhausen and György Ligeti for example, extended this idea to electronic compositions. Computers now permit the simulation not only of position but also of the movement of sound sources between a number of speakers, creating interesting possibilities. In general, I think that nowadays music must be considered as an art form that uses not only the time dimension, but also the space dimension. Or, at least, as an art form that could use the space dimension. In fact, I also think that this possibility hasn't been developed deeply enough, and has been used mostly as an effect rather than as an actual parameter of music. It is true though that the ear is not as sensitive to sound movement as it is to pitch change for example. Nevertheless, I think that it is worth exploring the new possibilities that computers give us in this field. One way to achieve that may be to create a music composition starting just from a visual input, that is from the placement in space of visual objects that represent sounds.

In the case of Dynamics, however, a visual object located in a certain position in space was thought of as *emitting* its sound from the very same location. A way to achieve this result was to use a videoprojector, a big screen and four speakers located at corners of the screen. The speakers then created not only the usual stereo image (right-left), but also the top-bottom image, *filling* the area of the screen with a continuous acoustic signal. In order to reach this goal, I digitized the sounds I had previouosly generated with FM machines into Csound files (Csound is a C and UNIX-based language written by Prof. Barry Vercoe, which is presently running on the VAX 11/750, VAXStation II and HP Bobcat workstations at the MIT Media Lab. It is a software package consisting of modules for audio synthesis, analysis, and processing). I then wrote two C programs that permit the input of a sound trajectory with a graphic tablet and stylus, and the use of the x and y tables that define this trajectory as control files for the sampled sounds.

However, the perception of sound localization of sounds is not as precise as in the visual domain, and is mainly a function of the spectral content of the sound [7]. This fact influences the way I imagined the size of the shapes to be matched with the sounds. In other words, the size of a certain shape was a function of the spatial extension of the corresponding sound, which was in turn a function of the spectral content of the sound itself. One problem I had with the construction of the visual objects was that sounds were definitely less defined in space than visual objects. To be clearly perceived as localized in space, sounds have to have some kind of noise component, and also partials above 7000 Hz [8]. This is especially true when vertical localization is considered, a task that our perception system does not accomplish as well as horizontal localization. Therefore, since some of my sounds' spectra did not contain a great deal of high frequencies, some of my sounds were vaguely defined in space. Nevertheless, I had to define them in space as specific visual objects with well defined contours. Sometimes it would have been necessary to have the visual objects smoothly merge with others or with the background, in order to better simulate the sensation that the sounds were producing. A good visual example of shapes that *melt* with others can be found in the computer generated video *Ecology II: Float*, by the Japanese artist Yoichiro Kawaguchi [video Kawaguchi]. It was then necessary to create a scale of sizes to be perceptable and understandable by the listener. The sound that had the perceived highest frequency was set to be very small (precise localization), while the sound with lowest frequency filled the screen, simulating its very vague spatial localization. Consequently, by means of compared and repeated judgement, I was able to determine a scale of sounds (from the lowest to the highest in frequency) that in turn corresponded to a scale of sizes.

3.4 Intensity

The final correspondence, the simplest, was between the perceived intensities of the audio and video events (loudness and brightness). This means that as loudness changed, following an envelope, so did the brightness of the corresponding shape. Eventually, the visual object could fade out while the loudness was becoming zero. However, although the principle was quite simple, it wasn't easy to map one domain to the other, since, in fact, none of them is a linear parameter. Therefore, I used a method similar to the one I used for localization. That is, I created a scale of sounds, from the loudest to the weakest, mapping the loudest to white (very bright), and the weakest to barely visible. I also had to take into account the well known fact that we are not sensitive to colors linearly (for example, we are more sensitive to green than to blue). The entire process was empirical, but the tools I had did not permit me to create a more precise correspondence. A software for the simultaneous generation of sounds and visual objects would be in fact necessary, in order to have an immediate feedback.

3.5 Creating the Correspondences

When generating the correspondences, my approach was:

- I created a sound I was interested in. It is in fact easier for me to model visual objects on sounds than vice versa. This is because sounds have an extremely variable behavior over time, and it is easier to control the change of shapes than the change of sounds (especially FM sounds).
- I sketched an outline of the temporal behavior of the main components of the timbre, indicating the envelope, and the peaks of the sound.
- I imagined a shape that was changing over time and matching the behavior of the sound I was visualizing, and wrote down how the shape should be (round, jagged, shiny and so on).
- When actually creating the 3-d model (with the program 3-dg, running

on a Symbolics 3600 computer at the MIT Media Lab) I listened again to the sound and if necessary modified the previous idea (the sketch) so that it matched the sound. Since often the timbre changed over time, so did the shape. In order to produce this effect, I used two different methods :

1 I created two 3-d models (initial and final), that were then automatically linearly interpolated by the animation program (in one case I used more than two models).

2 I rotated the object along one (or more) axis. Since the shapes were irregular, they revealed other sides that had not been seen yet, behaving like a sound that, while changing, reveals unknown aspects of itself.

3.6 The Composition

While I first created the sounds, and afterwords the visual objects, I preferred to invert the process, and use the visual language to create the composition (the animation), matching the music *a posteriori*. This procedure permitted me to think about the organization of the piece using the visual language. Although the product of the compositional method I chose (mapping the sounds to visual objects, and then using the visual objects to organize the piece), is an audiovisual piece, I think that this method can be useful also when the intention is to compose just a work of music made of abstract sounds. In fact, the musical part of Dynamics is equally interesting even if heard alone, without the visual counterpart. However, I think that if I had used the inverse process (creation of the music composition first, and then matching the visual objects), the final product would have been quite different. In fact, features like spatial location, movement, and speed, traditionally typical of visual events, this way played a major role in the composition, not only in the visual but also in the audio domain. Also, the creation of a correspondence between audio and video events allowed me to better identify categories of timbres, to visualize them, and to associate different sounds only because the corresponding visual objects were associated. Probably, If I had used the inverse process, I would have associated other sounds. In fact, the process that was required in order to associate a visual object to a sound (cf. section 3.2) forced me to think about the sounds themselves and to put them into categories. Although it might be debatable to restrict sounds into categories, this process has proven to be helpful for this kind of composition. In fact, since the notion of timbre is quite loosely defined dformally, I think that timbres should be controlled by means of perceptual rather than acoustic attributes. Two major categories I used were those of *harsh* and *soft* timbres, respectively visualized as jagged and rounded shapes (fig. 3.1, 3.2). Other categories I used were *shiny* and *opaque*, visualized just as shiny and opaque objects. However, these categories were definetely subjective, and were not meant to be valid for everybody. My idea of category is similar to W. Kandinsky's concept of *antithesis* [9]. Categories are for me general families of aural and visual objects that can may be comparable to the instrumental families of the orchestra. Experiments such as those that I will describe later in this thesis are, on the other hand, meant just to understand whether or not people agree

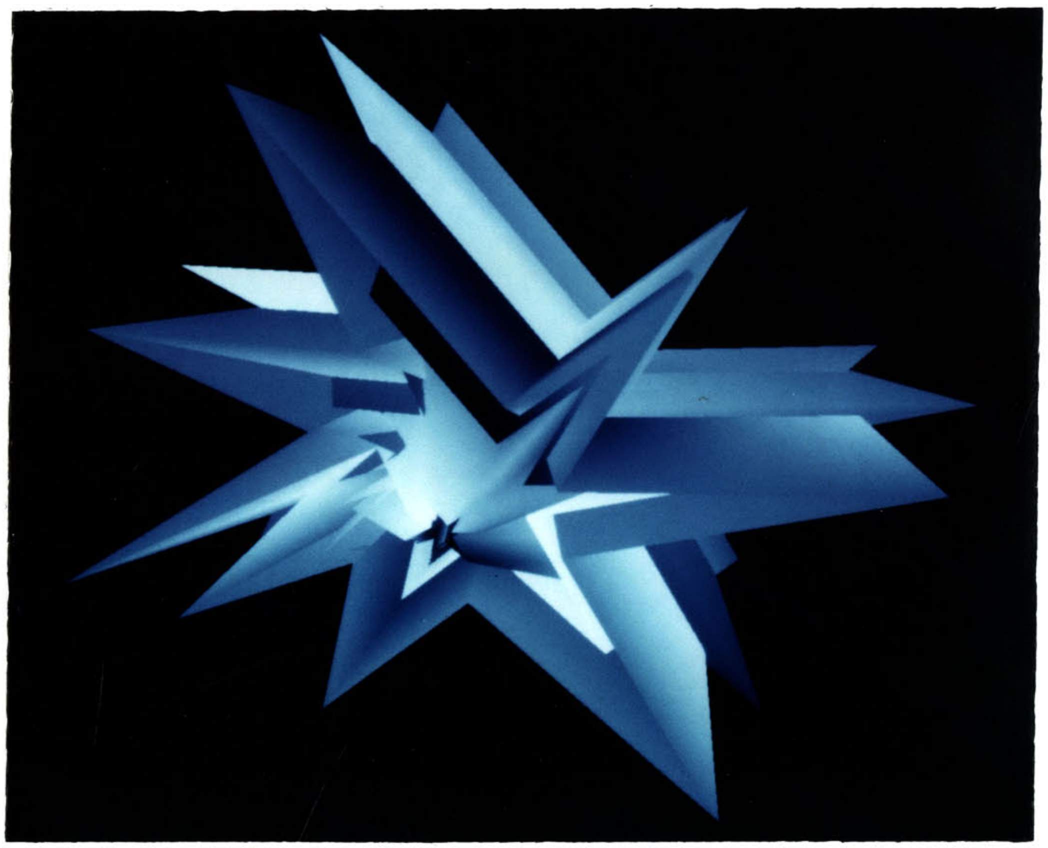

fig. 3.1

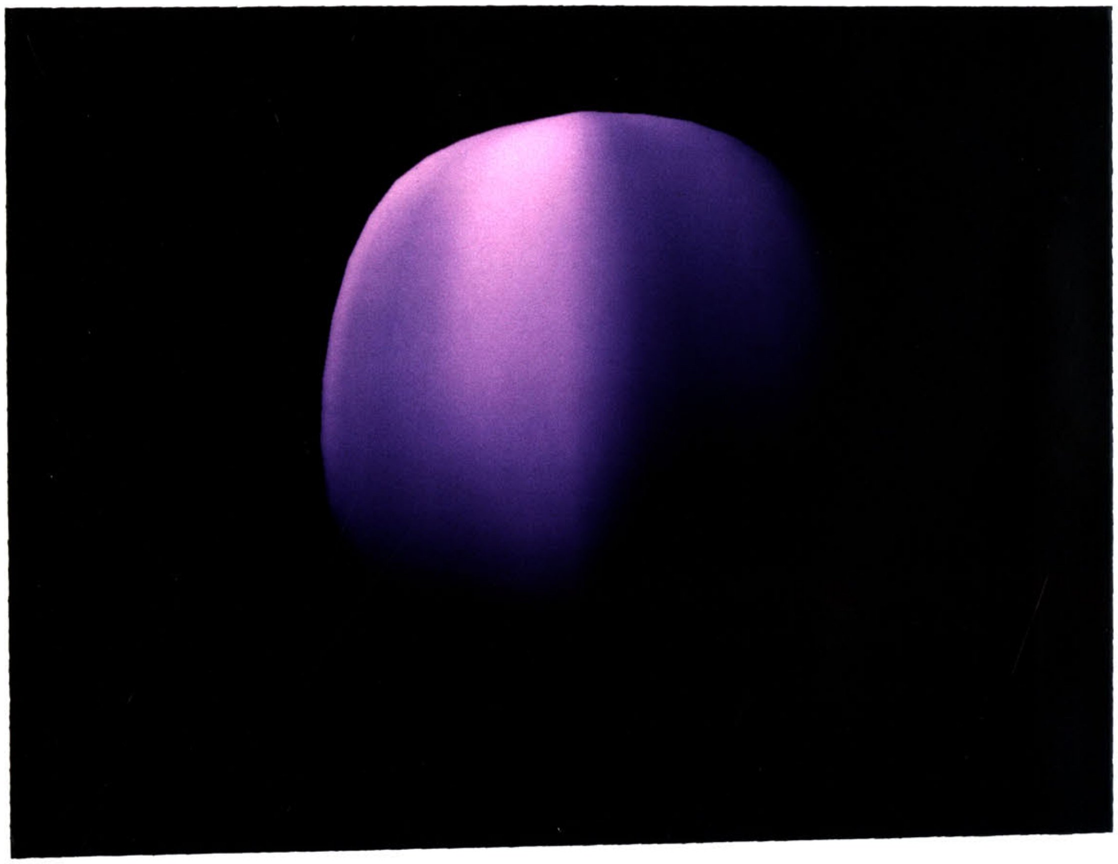

fig. 3.2

on such perceptual attributes of sounds.

Although the product of the compositional method I chose (mapping the sounds to visual objects, and then using the visual objects to organize the piece) is an audiovisual piece, I think that this method can also be useful when the intention is just to compose a work of music made of abstract sounds. In fact, the musical part of Dynamics is equally interesting even if heard alone, without the visual counterpart.

The composition is divided into 6 major sections. The first one is a presentation of the nine audiovisual events, in succession. One event is shown at a time, with cross fading effects. The subsequent 3 sections constitute the *body* of the piece. Each of those sections is divided into respectively 3, 3, and 2 episodes.

1	2				3				4		▨	5		6
intro	a	b	c	s1	d	e	f	s2	g	h	s3	s2	s1	s1+s2+s3

fig. 3.3

In each episode, one or two audiovisual events perform a specific action, as I will describe later. A final episode, for each of those sections, constitutes the mix of the previous episodes of that section (fig. 3.3). For example, in section 4 the final episode sums up the previous 2 episodes. In section 5, the final episodes are repeated. Notice that episode s3 belongs to both section 4 and section 5 (as indicated by the shaded area in fig. 3.3). The final section is the sum of the 3 sums of each episode, s1, s2, and s3 (fig. 3.4).

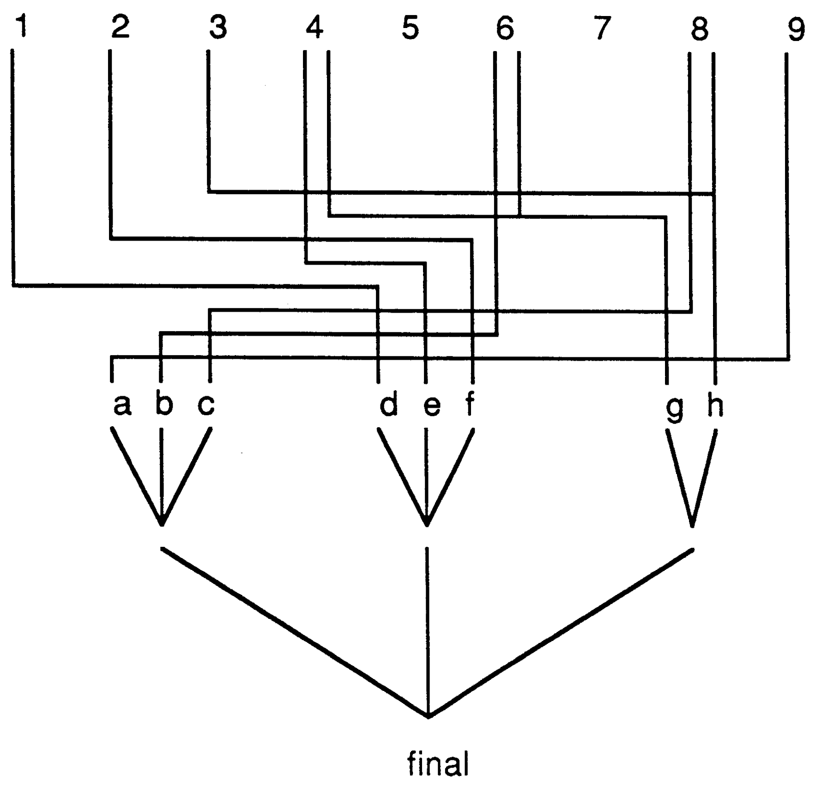

fig. 3.4

I'd like to briefly analyze the a, b, c, d, e, f, g, h sections, since they are at the core of the piece. In these episodes, each object is spatially repeated (i.e. multiple copies of the same object are present). A, b, c are the sections of time, meaning that variations of time are considered, and the spatial positions are not ordered. In a, the object is shown 15 times, very fast, in succession, so that it never overlaps in time. B shows an object (and its copies) that is fades in and out, partially overlapping in time. In c, the copies of the object are perfectly synchronized (fig. 3.5). D, e, f are the episodes of space, meaning that variations of space are made. D presents an object (and again its copies, all in synch) that is moving with a jaggy trajectory. E shows a completely static object (its shape changes, but not its position). In f a regular movement is used (fig. 3.6). Episodes g and h use 2 objects (and copies) at the same time. In these sections, different visual characteristics are opposed. The objects are all synchronized, and have a regular movement. In g, a round object is opposed to a spiny object. In h, an opaque object is opposed to a shiny object. Notice that objects 5 and 7 are used only at the beginning, in the *intro* section.

The organization of the piece was relatively simple. After the presentation of the audiovisual events, I started building episodes that were in different ways coherent (coherence of time, space and opposition of attributes). Each of the final episode combined the previous episodes of its section, creating a more complex and interesting audiovisual texture. The basic idea of the piece is in

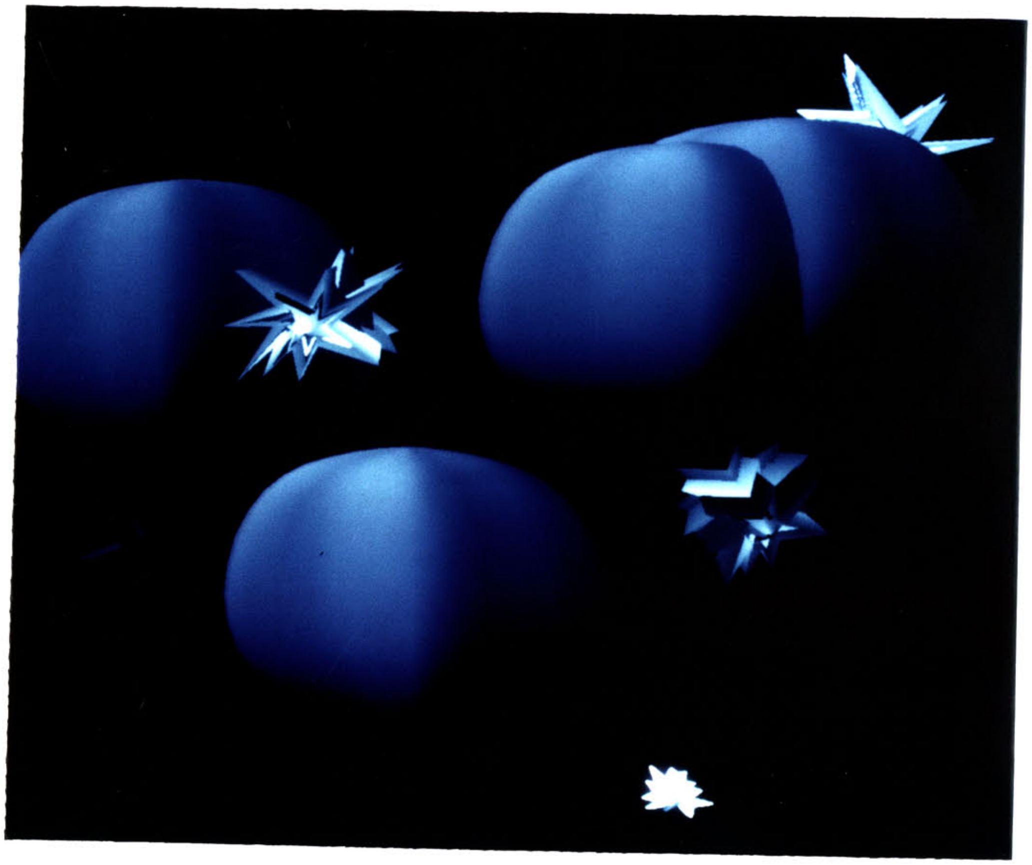

fig. 3.5

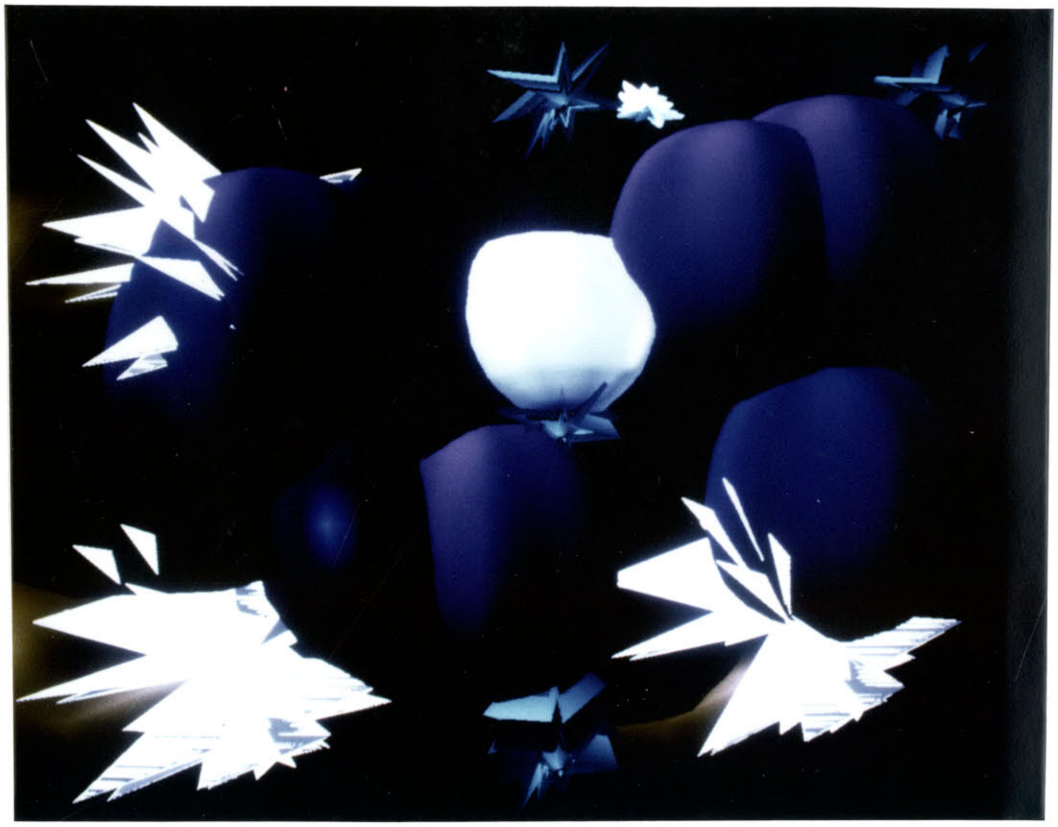

fig. 3.6

fact the construction of complexity, starting from simple elements. Because of this, I repeated again the episodes s2 and s3. This way, after the presentation in which each event is seen and heard alone, and after the second part (sections 2, 3, and 4) in which 2 or 3 audiovisual objects contemporarily play different roles, I created a sequence in which several objects were considered at the same time. In other words, the piece becomes more and more dense. The final section, in which all the previous sums are again added, shows a very dense audiovisual texture, comparable to a musical *tutti.*

4 An Audiovisual Experiment

4.1 Introduction

Since the audiovisual composition Dynamics was fundamentally based on my own subjective correspondences, I decided to create a test that verified other people's reactions to several visual/aural associations. In order to do that, I wrote a C program that generated visual as well as aural events. The purpose of this test was to understand whether or not shapes are objectively related to timbres by asking viewers to indicate preferences to timbre / shapes associations, and to qualify such associations. In order to establish such an experiment, I created 8 visual sequences and 6 sounds, producing a grid of 48 combinations. I first created the sounds, and then the animations (cf. 3.5).

4.2 The Program

The program runs on an Apple Macintosh II (Appendix A). The purpose of the program was to set an environment by which I could both conduct my experiments and create an audiovisual composition to be performed in real-time. For this reason, I chose machines that were going to be easily available, such as the

Apple Macintosh II and the Akai S900 sampler. The computer has an Apple videocard providing a resolution of 640 x 480 pixels at 67 Hz non-interlaced, a 13" Sony monitor, 4 Megabytes of RAM, and a 40 Megabyte hard disc. The program fundamentally generates two 2-d shapes, each constructed by means of 4 Bézier curves, and interpolates all shapes between them, creating simple animation. When the animation begins, the program simultaneously sends a Note_On MIDI signal to the sampler, making it play a sound. A Note_Off MIDI signal is also sent when the animation is done.

This real-time capability is important for two reasons: first, the experience of producing Dynamics clearly demonstrated that when working in an audiovisual environment it is very important to have immediate feedback of what one is creating. In fact, it is often necessary to compare the images with the music. In a real-time environment, this is instantaneous. Also, the quality of both image and sound can be degraded once put on tape or film, and as an artist I want to preserve the quality of my products as much as I can. Unfortunately, in most computer art festivals and competitions tapes are still the only available medium.

4.3 Music

The software I wrote used the MIDI standard in order to communicate with musical instruments. Although a number of commands are available, partly depending on which musical device is controlled, this version of my program required only two basic functions: Note_On and Note_Off. Note_On and Note_Off are the names of two C functions that in turn call a MIDI_Write function that is present in another package, the MIDI driver that Lee Boynton wrote at the MIT Media Lab. Further versions of my program will include other MIDI capabilities, such as the panning function and program change. The panning function allows the creation of stereo movement of the sound source. The program change permits software-controlled change of the *instrument* used.

4.4 Images

Each Bézier curve is defined by 4 control points and approximated by 10 segments. The final control point of each curve is forced to be coincident with the first of the next one, so that the 4 curves are connected. The program takes as input, through a mouse, the 12 control points that define the initial shape. Similarly, 12 more points are used as input for the final shape. Each point is re-definable, so that the shape can be interactively changed by the user. Once the two shapes are set, the program computes the interpolated control points, using linear interpolation, and consequently computes the interpolated shapes. Both the control points and the segments that approximate the interpolated curves are stored in 1-d arrays, so that the contents of the arrays are accessed very quickly. Then the pre-computed shapes are displayed, using these Macintosh QuickDraw routines: OpenPoly, PolyLine, FillPoly, ClosePoly. The colors of the initial and final shapes are also user-definable, by means of the Color Picker Package, and are linearly interpolated in the RGB space. However, in the version of the program used to conduct experiments, the color of the first shape, when defined, is grey, while the color of the second is white. The background is black. When animated, the saturation of the color of the shapes is fixed at zero, so that the viewer is not influenced by the hue. Once the animation is done, the segments that approximate the interpolated shapes can be saved on disc and re-loaded.

Unfortunately, this method of animating shapes did not permit complex images, because features like anti-aliasing or texture mapping, that could have greatly enhanced the quality of the images, were not available. In fact, both anti-aliasing and texture mapping are time expensive, and real-time animation would have been impossible. A solution to this problem would have been to create all the shapes in advance, and then to store (and successively display) in RAM the bit maps instead of the approximated curves. There are two advantages to this method: the images can be as complex as necessary, and the speed of display is very high. The cost is that a huge amount of memory is needed. For example, an image of size 1" x 1", relatively small, is made up of 72 x 72 pixels. This is equivalent to 648 bytes. If 8 color planes are used, then 5184 bytes are needed to store one shape. One second of animation, at the speed of one frame per video refresh (67 Hz), is therefore 347328 bytes, a considerable amount if an animation lasting 2 minutes and involving 10 1" x 1" objects is considered. Such a configuration would in fact require more than 416 Megabytes of memory. The situation would be even worse if the objects were larger. Of course, it would be possible to have most of the images stored on a huge hard disc, with only the necessary images kept in RAM. Probably a Macintosh II can deal with the transfer speed between hard disc and RAM. However, the 40 Megabyte hard disc available for the experiment did not have sufficient memory.

4.5 The Sounds

The sounds were generated by using a commercial package called Softsynth, by Digidesign, Inc., running on Apple Macintoshes. Both additive and FM synthesis are available, in combination if necessary. It is particularly suited for my goal, because I was able to use additive synthesis to create simple and precise sounds. In fact, additive synthesis gave me control of each partial of the sound. Three sounds were generated by using FM synthesis, which permits quite easily to create complex sounds, with just 2 oscillators and simple modulations.

When I chose the sounds, I knew which matching I was looking for (cf. 4.6). I therefore created sounds with variable smoothness. Sounds 1 and 2 can be considered as *medium smooth*, sounds 3 and 4 as *very smooth*, and sounds 5 and 6 as *not smooth*. Also, I considered changes in frequency and amplitude envelope, as is underlined in the following brief description of the sounds.

The sounds all lasted 5 seconds and were sampled at a frequency of 22 KHz. Sound 1 was an FM sound, using two oscillators. The carrier was set at 220 Hz, and its attack lasted 0.5 seconds, and the decay 4.5 seconds. The modulator was set at an amplitude of 1/3 the amplitude of the carrier, the frequency was again 220 Hz, and attack and decay were respectively of 2 and 3 seconds. Sound 2 had an attack of 1.25 seconds and a decay of 3.75. The fundamental frequency was set at 440 Hz, but in the case of this sound only the 11th through 20th partials were used, producing a high frequency sound. Also, the partials were not in perfect harmonic ratio, so the sound was a bit *dirty*. Sound 3 and sound 4 were similar, both FM, and the difference was only in the frequency of the two oscillators. In the first case, in fact, the carrier was 110 Hz and the modulator 154 Hz (ratio 1 : 1.4). In the second case, the carrier was 440 Hz and the modulator 616 Hz (same ratio). This difference in turn affected the size of shapes (cf. 3.3). The envelope, as well as the amplitude of the two oscillators were identical. The attack of the carrier was 0.05 seconds and the decay was 4.95 seconds. The attack of the modulator was, inversely, 4.95 seconds, and the decay 0.05 seconds. Also, the last two sounds had a similar structure. Both had a fundamental frequency of 220 Hz, with 32 harmonics set to their maximum level of amplitude. Therefore, the only difference was in their envelope. Sound 5 had an attack of 4.975 seconds, and a decay of 0.025. Sound 6 had the opposite characteristics.

4.6 The Shapes

To create the images, I followed three simple principles of mapping: frequency/size, envelope/brightness, smoothness of the sound/smoothness of the shape. Three characteristics of sound spectra seem to determine their *smoothness.* A slow attack, or an integer ratio between the partials, or partials steady over time, seem to determine a sound's smoothness.

The first two mappings were easy to visualize. In the first mapping, since the sounds had just 4 levels of frequency (110, 220, 400, and 4840 Hz), I created 4 different sizes. In the second case, I followed the simple amplitude envelope of the sounds. In order to do that, I mapped audio level zero to black (the background) and peak of the sound to white.

The third mapping (smoothness of the sound/smoothness of the shape) was the most difficult. The first sound was visually represented twice, with different sizes, as a slightly changing round shape. Sound number two was visualized as a non-regular, very small changing shape. It was visualized twice, changing the size. In the third and fourth sound, emphasis was put on the transformation of the shape. In fact, in these FM sounds a transforming spectrum was clearly detectable, and therefore I created very plastic non regular shapes. Otherwise, as before, the two sounds were differentiated only by their sizes. The final two sounds, being steady over time, were represented with steady triangular shapes. In the case of these sounds, the envelope was the only distinguishing parameter.

4.7 Testing the Subjects

The experiment was conducted in the Audio Studio of the MIT Media Lab's Music and Cognition Group. Ten subjects, one at a time, watched the screen and listened to the sounds through 2 speakers. Since all images were centered on the screen, the sounds arrived from the center of the stereo image created by the two speakers. Each subject was given a sheet like the one shown in fig. 4.1. The three symbols (high, medium and low) indicated the degree of correspondence between the image and the sound. When I successively analyzed the results, I assigned 2, 1 and 0 points, respectively, to those symbols. The shapes are placed along the horizontal axis, with the sounds along the vertical axis. The subjects listening to the different sounds, keeping fixed the shape, proceeded vertically from top to bottom for each column, and left to right. They were free to assign whatever symbol they wanted to whatever correspondence, even repeating the high or low symbol several times. Moreover, the subjects were also free to experience the sounds/shapes before actually judging the correspondences. Even when judging they could replay the animation several times. The average results, normalized between 0 and 1, are shown in fig. 4.2. The original matchings are circled, and indicate the combination I originally created.

	sh1	sh2	sh3	sh4	sh5	sh6	sh7	sh8
S1								
S2								
S3								
S4								
S5								
S6								

for each combination please choose

one of the following symbols:

○	△	□
high	medium	low

fig. 4.1

	sh1	sh2	sh3	sh4	sh5	sh6	sh7	sh8
S1	○ .8	○ .7	.25	.6	.4	.45	.6	.4
S2	.2	.55	○ .6	○ .7	.35	.2	.55	.5
S3	.55	.6	.5	.6	○ .8	.65	.35	.35
S4	.65	.65	.6	.6	.75	○ .8	.55	.45
S5	.35	.25	.6	.45	.2	.2	○ .65	.15
S6	.05	.0	.1	.05	.0	.2	.2	○ .55

fig. 4.2

4.8 Interpreting the Results

The interpretation of the results, done with the help of a statistical program, is relatively complex (Appendix B). Two main elements can be extrapolated from this experiment: the subjects were significantly affected by the changes of sounds ($F(5,45) = 11.82$, $p < .001$), and, even more importantly, they were very affected by the interaction of sounds and shapes ($F(35,315) = 2.466$, $p < .001$).

However, the following analysis seems plausible. Shape 1 was quite successful, in the sense that the subjects mostly chose S1 as the best sound. Notice that the best sounds, other than S1, were S3 and S4, probably because their shape is rounded at the beginning, like S1. Sounds S3 and S4 always got good judgement, except for with shapes 7 and 8, that were in fact steady (representing the steady sounds 5 and 6). Also, notice the low values of S6 through all the sounds, except for *its* shape, sh8. This is explainable considering that S6 had a very sudden attack (0.025 seconds), while most of the shapes (except sh8 of course) had a smoother attack. Therefore, a first conclusion is that the envelope of the sound can be a factor in matching sounds with images. Shape 2 was the same as shape 1 but smaller. The results are similar to those of shape 1, although less convincing. Shape 3 and 4 have also similar results. In both cases the matching between shape and sound doesn't seem to be successful enough. Shapes 4 and 5 are vice versa extremely successful. In fact, without any doubt the subjects chose the FM sounds associated with those shapes, giving a slight preference to their own sounds, respectively S3 and S4. This result is particularly interesting, because it clearly demonstrates that the changing spectrum of those FM sounds was well represented by the changing shape of sh5 and sh6. Shape 7 was best represented by S5, although other sounds got pretty good judgements. Similarly, shape 8 was best represented by S6. Notice that also S1 and S2 got decent judgements, definitely better than S3 and S4. This is may be explainable by the fact that S1 and S2 are more steady sounds than S3 and S4.

4.9 Conclusions

A few considerations can be made after looking at the results. The subjects preferred some sound/shape associations rather than others. The loudness/brightness envelope is certainly perceived as very important. In fact, if the envelope of the sound does not correspond to the brightness of the shape, then a mismatch is easily detected. Finally, more than the shape itself, which is however recognizable as an element to be considered, the transformation of the shape is clearly perceived as a change in the spectrum of the sound. These two results are in my opinion particularly interesting, since they indicate that animated abstract objects are well suited for the representation of changing timbres.

Notes

Chapter 2

[1] For example, Father Castel associated the notes C, E, G with the colors blue, yellow, red.

[2] [Rondolino, pag. 101].

[3] Ginna stated "Since 1907 I had understood the pictorial possibilities of cinema ... I thought to directly paint on film" [Rondolino, pag. 99].

[4] Survage wrote "Colored rhythm is not an illustration or an interpretation of a musical work. It is an autonomous art, even if it is based on the same psychological data on which music is based" [Rondolino, pag. 102].

Chapter 3

[1] Schoenberg more precisely stated, "I think that sound reveals itself by means of the timbre and that pitch is a dimension of the timbre. The timbre is therefore the whole, the pitch is part of this whole, or better, pitch is nothing but timbre measured in just one dimension" [Schoenberg, pag. 528-529].

[2] Jean Claude Risset and David Wessel also wrote, "With the control of timbre now made possible through analysis and synthesis, composers ... can articulate musical compositions on the basis of timbral rather than pitch variations ... It is conceivable that proper timbral control might lead to quite new musical architectures" [Risset and Wessel, pag. 28-29].

[3] Recently, Fred Lerdhal has proposed hierarchic structures and concepts taken from the context of traditional music as methods to organize new timbres [Lerdhal]. Although both methods are powerful tools for dealing with complex compositional structures, it seems to me that hierarchies are a too rigid method for the creation of music compositions, and that new timbres require a substantially different approach from that of traditional compostition. In this sense, since comprehension takes place whenever the perceiver is able to assign a

precise mental representation to what is perceived, I think that visual representation of timbres can help the process of understanding timbres, and take the place of a specific hierarchic organization of the timbral elements themselves. Therefore, visual representation of timbres can be an useful method for the organization of musical composition.

[4] In his Concerning the Spiritual in Art, Kandinsky stated "On the whole, keen colors are well suited by sharp forms (e.g., a yellow triangle), and soft, deep colours by round forms (e.g., a blue circle)" [Kandinsky, pag. 29].

[5] Kandinsky again wrote "Yellow, if steadily gazed at in any geometrical form, has disturbing influence, and reveals in the colour an insistent, aggressive character" [Kandinsky, pag. 37].

[6] [Erickson, pag. 141].

[7] [Dodge and Jerse, pag. 240-247].

[8] [Erickson, pag. 143].

[9] [Kandinsky, "The Language of Form and Colour", fig. 1-2].

Bibliography

Abbado, Adriano, Morda', Claudio, and Rocca, Gianluigi, *Immagini con il Computer*, Mondadori, Milano 1985.

Alfio Bastiancich, *L'opera di Norman McLaren* , Giappichelli, Torino 1981.

Blauert, Hans, *Spatial Hearing*, MIT Press, 1983.

Buxton, William et al., *Objed and the Design of Timbral Resources,* Computer Music Journal, Vol. VI No. 2, 1982.

Dodge, Charles, and Jerse, Thomas A., *Computer Music*, Schirmer Books, New York, 1985.

Ehresman, David, and Wessel, David, *Perception of Timbral Analogies,* IRCAM, Paris 1978.

Eisenstein, Sergei M., *La natura non indifferente*, Marsilio Editori, Venezia 1981.

Erickson, Robert, *Sound Structure in Music*, University of California Press, Berkeley CA 1975.

Foley, James D., and Van Dam, Andries, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading MA 1984.

Franke, Herbert W., *Computer Graphics - Computer Art*, Springer-Verlag, Berlin 1985.

Gardner, Howard, *Frames of Mind*, Basic Books, New York 1983.

Goldberg, Theo, "The Prefiguration of a Musical Composition: Model of a Computer Graphics Program", *Proceedings of the 1985 International Computer Music Conference*, The Computer Music Association, San Francisco.

Grey, John M., "An exploration of Musical Timbre", Doctoral dissertation, Stanford University 1975.

Kandinsky, Wassily, *Concerning the Spiritual in Art*, Dover Publications, Inc., New York 1977.

Karkoschka, Erhard, *Notation in New Music*, Praeger Publishers, New York 1972.

Le Grice, Malcom, *Abstract Film and beyond*, The MIT Press, Cambridge MA 1977.

Lerdahl, Fred, "Timbral Hierarchies", *Contemporary Music Review* , Vol. II No. 1, Stephen McAdams editor, Harwood Press, London 1987.

Ligeti, György, and Wehinger, Rainer, *Artikulation*, B. Schott's Söhne, Mainz 1970.

Machover, Tod, "The Extended Orchestra", *The Orchestra*, Joan Peyser editor, Charles Scribner's Sons, New York 1986.

Malouf, L. Frederick, "A System for Interactive Music Composition through Computer Graphics", *Proceedings of the 1985 International Computer Music Conference*, The Computer Music Association, San Francisco.

McAdams, Stephen, and Saariaho, Kaija, "Qualities and Functions of Musical Timbre", *Proceedings of the 1985 International Computer Music Conference*, The Computer Music Association, San Francisco.

Mitry, Jean, *Storia del cinema sperimentale*, Mazzotta, Milano 1971.

Risset, Jean-Claude, *Musical Acoustics*, IRCAM, Paris 1978.

Risset, Jean-Claude, *Hauteur et timbre des sons*, IRCAM, Paris 1978.

Risset, Jean-Claude, and Wessel, David, *Indagine sul timbro mediante analisi e sintesi*, Quaderno 2, Limb / La Biennale, Venezia 1982.

Roads, Curtis, *Composers and the computer*, William Kaufmann, Los Altos CA 1985 .

Rogers, David F., *Procedural Elements for Computer Graphics*, McGraw-Hill, New York 1985.

Russolo, Luigi , *The Art of Noises*, Pendragon Press, New York, 1986.

Rondolino, Gianni, *Storia del cinema d'animazione*, Einaudi, Torino 1974.

Schoenberg, Arnold, *Manuale di armonia*, Il Saggiatore, Milano 1963.

Stautner, John P., "A Flexible Acoustic Ambience Simulator", *Proceedings of the 1982 International Computer Music Conference*, The Computer Music Association, San Francisco.

Stroppa, Marco, *Sull'analisi della musica elettronica*, Quaderno 4, Limb/La Biennale, Venezia 1984.

Stroppa, Marco, *L'esplorazione e la manipolazione del timbro*, Quaderno 5, Limb/La Biennale, Venezia 1985.

Veronesi, Luigi, *Proposta per una ricerca sui rapporti suono / colore*, Siemens Data, Milano 1977.

Wessel, David, *Low Dimensional Control of Musical Timbre*, IRCAM, Paris 1978.

Whitney, John, *Digital Harmony*, McGraw-Hill, New York 1980.

Xenakis, Iannis, *Formalized Music*, Indiana University Press, Bloomington 1971.

Videography

Abbado, Adriano, *Isomorfismi Suono Luce*, Commodore Italiana, Milano 1986.

Abbado, Adriano, *Dynamics*, MIT Media Lab, Cambridge MA 1988.

Eggeling, Viking, *Diagonal Symphony*, Berlin 1925.

Eisenstein, Sergej M., *Alexander Nevsky*, Moscow 1938.

Fischinger, Oskar, *Motion painting*, Hollywood 1949.

Jenny, Hans, *The Healing Nature of Sound*, MACROMedia, Brookline MA 1986.

Kawaguchi, Yoichiro, *Ecology II: Float*, Nippon Electronics College, Shinjuku-ku, Tokyo 1987.

Lye, Len, *Colour Box*, GPO Film Unit, Glasgow 1934.

Mitry, Jean, *Images pour Debussy*, 1952.

McLaren, Norman, *Colour Cocktail*, Glasgow 1935.

McLaren, Norman, *Begone Dull Care*, National Film Board of Canada, Ottawa 1948.

McLaren, Norman, *Workshop Experiment in Animated Sound*, National Film Board of Canada, Ottawa 1948.

Pfenninger, Rudolf, *Tonende Handschrift*, Geiselgasteig 1929.

Richter, Hans, *Rhytmus 21*, Berlin 1921.

Ruttmann, Walter, *Lichtspiel Opus III*, Berlin 1924.

Schwartz, Lillian, *Pixillation*, 1970.

Smith, Harry, *Film Number 1-7*, before 1950.

Whitney, John Sr., *Arabesque*, Pyramid, Santa Monica CA 1975.

Whitney, John Sr., *Victory Sausage*, Pacific Palisades CA 1987.

Appendix A

```
/*************************************

          clouds.c
         (c) Adriano Abbado, MIT Media Lab, 1988

**************************************/

#include   <stdio.h
#include   "QuickDraw.h"
#include   "Color.h"
#include   "MacTypes.h"
#include   "FontMgr.h"
#include   "WindowMgr.h"
#include   "MenuMgr.h"
#include   "TextEdit.h"
#include   "DialogMgr.h"
#include   "EventMgr.h"
#include   "DeskMgr.h"
#include   "FileMgr.h"
#include   "ColorToolbox.h"
#include   "ToolboxUtil.h"
#include   "ControlMgr.h"
#include   "File.h"

#define    NPOINTS 10L
#define    CONTROLP 16L
#define    XMAX 629L
#define    YMAX 432L
#define    SUP 418
#define    STEP 4L
#define    ITEMS1 4
#define    ITEMS2 9
#define    MNINCR1 XMAX/ITEMS1
```

```
#define     MNINCR2 XMAX/ITEMS2
#define     NFRAMES 60L
#define     TRUE 1
#define     FALSE 0
#define     NIL 0L
#define     TASKTABLESIZE 1000

typedef     float *Ptr1;
typedef     short *Ptr2;

Rect        rect;
Rect        rmenu;
RGBColor cf1={32767,32767,32767};

/*********************************************************/

main()
{
            Rect r1;
            int window_width, window_height;
            WindowPtr w1;
            Point p2;

            printf("\n");

            r1.top = screenBits.bounds.top + 42;
            r1.left = screenBits.bounds.left + 5;
            r1.bottom = screenBits.bounds.bottom - 5;
            r1.right = screenBits.bounds.right - 5;


            w1 = NewCWindow(NIL, &r1, "\pclouds.c", TRUE, documentProc,
(WindowPtr)(-1), FALSE, NIL);

            window_width = r1.right - r1.left;
            window_height = r1.bottom - r1.top;

            ShowWindow(w1);
            SetPort(w1);
            SetOrigin(0,0);

            rect.top = 0;
```

```
          rect.left = 0;
          rect.bottom = YMAX;
          rect.right = XMAX;
          FillRect(&rect, &black);

          rmenu.top = 0;
          rmenu.left = 0;
          rmenu.bottom = SUP;
          rmenu.right = XMAX;

          p2.h = 0;
          p2.v = 0;
          GetColor(p2,"\pPick a color:",&cf1,&cf1);
          FillRect(&rect, &black);

          MIDI_Open(1);     /* modem port */

          main_menu();
          main_loop(p2, rect, rmenu);
          MIDI_Close();
          CloseWindow(w1);
          SetEventMask(everyEvent);
          ExitToShell();
}

/********************************************************/

main_loop(p2, rect, rmenu)
Point     p2;
Rect      rect, rmenu;
{
          Ptr1 newx=(Ptr1)NewPtr(CONTROLP*NFRAMES*10);     /* intp controlp */
          Ptr1 newy=(Ptr1)NewPtr(CONTROLP*NFRAMES*10);
          Ptr2 px=(Ptr2)NewPtr(CONTROLP*2*2);    /* initial & final controlp */
          Ptr2 py=(Ptr2)NewPtr(CONTROLP*2*2);
          Ptr2 x0=(Ptr2)NewPtr((NPOINTS+1)*NFRAMES*4*2);    /* the curve */
          Ptr2 y0=(Ptr2)NewPtr((NPOINTS+1)*NFRAMES*4*2);
          short x, y;
          short count=0, start, curve, j=-1;
          short p[2];
          RGBColor black;
          RGBColor c0,c1;
```

```
FILE *fopen(), *fp;

Point pp;

pp.h = 0;
pp.v = 0;

black.red = 0;
black.green = 0;
black.blue = 0;

while (1) {
  while(!Button());
  GetMouse(&p);
  while(Button());
  x = p[1];
  y = p[0];
  if (y >= SUP) {
    if (x >= 0 && x < MNINCR1) {     /* animate */
      RGBForeColor(&black);
      PaintRect(&rect);
      shapes_menu();
      count = shapes_loop(p2, count, rmenu, newx, newy, x0, y0, px, py);
      RGBForeColor(&black);
      PaintRect(&rect);
      main_menu();
    }
    if (x >= MNINCR1 && x < MNINCR1*2) {     /* load */
      HideCursor();
      fp = fopen("test", "r");
      fscanf(fp, "%d\n", &count);
      fscanf(fp, "%d\n", &cf1.red);
      fscanf(fp, "%d\n", &cf1.green);
      fscanf(fp, "%d\n", &cf1.blue);
      for (j=0; j < (NPOINTS)*NFRAMES*4; j++) {
        fscanf(fp, "%d\n", &x0[j]);
        fscanf(fp, "%d\n", &y0[j]);
      }
      fclose(fp);
      ShowCursor();
    }
    if (x >= MNINCR1*2 && x < MNINCR1*3) {     /* save */
```

```
                HideCursor();
                fp = fopen("test", "w");
                fprintf(fp, "%d\n", count);
                fprintf(fp, "%d\n", cf1.red);
                fprintf(fp, "%d\n", cf1.green);
                fprintf(fp, "%d\n", cf1.blue);
                for (j=0; j < (NPOINTS)*NFRAMES*4; j++) {
                  fprintf(fp, "%d\n", x0[j]);
                  fprintf(fp, "%d\n", y0[j]);
                }
                fclose(fp);
                ShowCursor();
              }
              if (x >= MNINCR1*3 && x < MNINCR1*4)     /* quit */
                break;
            }
          }
}

/***********************************************************/

shapes_loop(p2, count, rmenu, newx, newy, x0, y0, px, py)
Point     p2;
short     count;
Rect      rmenu;
Ptr1      newx;
Ptr1      newy;
Ptr2      x0;
Ptr2      y0;
Ptr2      px;
Ptr2      py;
{
          short x, y;
          short start, curve, j;
          short p[2];
          RGBColor black;

          black.red = 0;
          black.green = 0;
          black.blue = 0;

          while (1) {
```

```
while(!Button());
GetMouse(&p);
while(Button());
x = p[1];
y = p[0];
if (y >= SUP) {
  if (x >= 0 && x < MNINCR2) {     /* make 1 */
     start = 0;
    curve = 0;
    make(px, py, newx, newy, x0, y0, start, curve);
  }
  if (x >= MNINCR2 && x < MNINCR2*2) {     /* modify 1 */
    start = 0;
    curve = 0;
    modify(px, py, newx, newy, x0, y0, start, curve);
  }
  if (x >= MNINCR2*2 && x < MNINCR2*3) {     /* make 2 */
    start = CONTROLP;
    curve = 1;
    make(px, py, newx, newy, x0, y0, start, curve);
  }
  if (x >= MNINCR2*3 && x <= MNINCR2*4) {     /* modify 2 */
    start = CONTROLP;
   curve = 1;
    modify(px, py, newx, newy, x0, y0, start, curve);
  }
  if (x >= MNINCR2*4 && x <= MNINCR2*5) {     /* color */
    GetColor(p2,"\pPick a color:",&cf1,&cf1);
    RGBForeColor(&cf1);
    RGBForeColor(&black);
    PaintRect(&rmenu);
    shapes_menu();
  }
  if (x >= MNINCR2*5 && x < MNINCR2*6) {   /* compute */
    HideCursor();
    count = intp_controlp(px, py, newx, newy);
    curve = 0;
    intp_curve(newx, newy, x0, y0, count, curve);
    ShowCursor();
  }
  if (x >= MNINCR2*6 && x < MNINCR2*7) {     /* play */
    HideCursor();
```

```
                RGBForeColor(&black);
                PaintRect(&rmenu);
                draw_erase(rmenu, x0, y0, count-1);
                RGBForeColor(&black);
                PaintRect(&bbox[j]);
                shapes_menu();
                ShowCursor();
              }
              if (x >= MNINCR2*7 && x <= MNINCR2*8) {     /* clear screen */
                RGBForeColor(&black);
                PaintRect(&rmenu);
                shapes_menu();
              }
              if (x >= MNINCR2*8 && x <= XMAX)     /* main_loop */
                return(count);
            }
          }
}

/*******************************************************/

main_menu()
{
          short x;
          short p[2];
          char foo[255];
          Rect r;
          RGBColor white, black;

          white.red = 65535;
          white.green = 65535;
          white.blue = 65535;
          black.red = 0;
          black.green = 0;
          black.blue = 0;
          RGBForeColor(&white);

          r.top = SUP;
          r.bottom = YMAX;

          for (x = 0; x <= (short)XMAX-(MNINCR1*2); x += MNINCR1) {
            r.left = x+1;
```

```
		r.right = x+MNINCR1-1;
		PaintRect(&r);
	}
	r.left = x+1;
	r.right = (short)XMAX;
	PaintRect(&r);

	RGBForeColor(&black);
	p[0] = SUP+11;
	p[1] = 10;
	sprintf(foo, "animate");
	CtoPstr(foo);
	MoveTo(p[1],p[0]);
	DrawString(foo);

	p[0] = SUP+11;
	p[1] += MNINCR1;
	sprintf(foo, "load");
	CtoPstr(foo);
	MoveTo(p[1],p[0]);
	DrawString(foo);

	p[0] = SUP+11;
	p[1] += MNINCR1;
	sprintf(foo, "save");
	CtoPstr(foo);
	MoveTo(p[1],p[0]);
	DrawString(foo);

	p[0] = SUP+11;
	p[1] += MNINCR1;
	sprintf(foo, "QUIT");
	CtoPstr(foo);
	MoveTo(p[1],p[0]);
	DrawString(foo);
}

/*********************************************************/

shapes_menu()
{
	Rect r;
```

```
short x;
short p[2];
char foo[255];
RGBColor white, black;

white.red = 65535;
white.green = 65535;
white.blue = 65535;
black.red = 0;
black.green = 0;
black.blue = 0;
RGBForeColor(&white);

r.top = SUP;
r.bottom = YMAX;

for (x = 0; x <= (short)XMAX-(MNINCR2*2); x += MNINCR2) {
  r.left = x+1;
  r.right = x+MNINCR2-1;
  PaintRect(&r);
}
r.left = x+1;
r.right = (short)XMAX;
PaintRect(&r);

RGBForeColor(&black);
p[0] = SUP+11;
p[1] = 5;
sprintf(foo, "1");
CtoPstr(foo);
MoveTo(p[1],p[0]);
DrawString(foo);

p[0] = SUP+11;
p[1] += MNINCR2;
sprintf(foo, "modify1");
CtoPstr(foo);
MoveTo(p[1],p[0]);
DrawString(foo);

p[0] = SUP+11;
p[1] += MNINCR2;
```

```
sprintf(foo, "2");
CtoPstr(foo);
MoveTo(p[1],p[0]);
DrawString(foo);

p[0] = SUP+11;
p[1] += MNINCR2;
sprintf(foo, "modify2");
CtoPstr(foo);
MoveTo(p[1],p[0]);
DrawString(foo);

p[0] = SUP+11;
p[1] += MNINCR2;
sprintf(foo, "color");
CtoPstr(foo);
MoveTo(p[1],p[0]);
DrawString(foo);

p[0] = SUP+11;
p[1] += MNINCR2;
sprintf(foo, "compute");
CtoPstr(foo);
MoveTo(p[1],p[0]);
DrawString(foo);

p[0] = SUP+11;
p[1] += MNINCR2;
sprintf(foo, "play");
CtoPstr(foo);
MoveTo(p[1],p[0]);
DrawString(foo);

p[0] = SUP+11;
p[1] += MNINCR2;
sprintf(foo, "clear");
CtoPstr(foo);
MoveTo(p[1],p[0]);
DrawString(foo);

p[0] = SUP+11;
p[1] += MNINCR2;
```

```
		sprintf(foo, "go back");
		CtoPstr(foo);
		MoveTo(p[1],p[0]);
		DrawString(foo);
}

/*******************************************************/

make(px, py, x, y, x0, y0, start, curve)
short		px[], py[];
short		x0[], y0[];
float		x[], y[];
short		start, curve;
{
		short count, erase = curve +1;

		get_controlp(px, py, start);
		for (count = start; count < start + CONTROLP; count++) {
		  x[count] = px[count];
		  y[count] = py[count];
		}
		count = curve + 1;
		intp_curve(x, y, x0, y0, count, curve);
		draw(x0, y0, count, curve, erase);
}

/*******************************************************/

get_controlp(px, py, start)
short		px[], py[];
short		start;
{
		short count;
		short p[2];
		short x;
		short y;
		RGBColor green, magenta;

		green.red = 0;
		green.green = 65535;
		green.blue = 0;
		magenta.red = 65535;
magenta.green = 0;
magenta.blue = 65535;

if (start == 0)
  RGBForeColor(&green);
else
  RGBForeColor(&magenta);

for (count = start; count < start+4; count++) {     /* 1st curve */
  while(!Button());
  GetMouse(&p);
  while(Button());
  x = p[1];
  y = p[0];
  px[count] = x;
  py[count] = y;
  plot (x, y);
}

px[start+4] = px[start+3];     /* 2nd curve */
py[start+4] = py[start+3];
for (count = start+5; count < start+8; count++) {
  while(!Button());
  GetMouse(&p);
  while(Button());
  x = p[1];
  y = p[0];
  px[count] = x;
  py[count] = y;
  plot (x, y);
}

px[start+8] = px[start+7];     /* 3rd curve */
py[start+8] = py[start+7];
for (count = start+9; count < start+12; count++) {
  while(!Button());
  GetMouse(&p);
  while(Button());
  x = p[1];
  y = p[0];
  px[count] = x;
  py[count] = y;
```

```
            plot (x, y);
          }

          px[start+12] = px[start+11];    /* 4th curve */
          py[start+12] = py[start+11];
          for (count = start+13; count < start+CONTROLP-1; count++) {
            while(!Button());
            GetMouse(&p);
            while(Button());
            x = p[1];
            y = p[0];
            px[count] = x;
            py[count] = y;
            plot (x, y);
          }

          px[start+CONTROLP-1] = px[start];
          py[start+CONTROLP-1] = py[start];
}

/*******************************************************/

modify(px, py, x, y, x0, y0, start, curve)
short     px[], py[];
short     x0[], y0[];
float     x[], y[];
short     start, curve;
{
          short j, erase, count, pos;
          short p[2];
          short xx;
          short yy;

          while(!Button());
          GetMouse(&p);
          while(Button());
          xx = p[1];
          yy = p[0];

          for (j = start; j < start + CONTROLP; j++) {
            if ((xx> (px[j]-3) && xx < (px[j]+3)) && (yy > (py[j]-3) &&yy(py[j]+3))){
              while(!Button());
```

```
                GetMouse(&p);
                while(Button());
                xx = p[1];
                yy = p[0];
                px[j] = xx;
                py[j] = yy;
                if (j == start) {
                  px[start+CONTROLP-1] = px[j];
                  py[start+CONTROLP-1] = py[j];
                }
                if (j == start+3 || j == start+7 || j == start+11) {
                  px[j+1] = px[j];
                  py[j+1] = py[j];
                }
                plot(xx, yy);
                count = curve + 1;
                erase = 0;
                draw(x0, y0, count, curve, erase);    /* erase old curve */
                for (count = start; count < start + CONTROLP; count++) {
                  x[count] = px[count];
                  y[count] = py[count];
                  pos++;
                }
                erase = curve + 1;
                count = curve + 1;
                intp_curve(x, y, x0, y0, count, curve);
                draw(x0, y0, count, curve, erase);    /* new curve */
                break;
              }
            }
}

/*******************************************************/

intp_controlp(x, y, newx, newy)
short       x[], y[];
float       newx[], newy[];
{
            short absx[CONTROLP], absy[CONTROLP], distx[CONTROLP], disty[CONTROLP];
            float incrx[CONTROLP];
            float incry[CONTROLP];
            short supmax, supmay, maxdist;
```

```
          short i, count, start = 0, end = CONTROLP, times = 0;
          float incr, initvalue, endvalue, temp;

          for (count = 0; count < CONTROLP; count++) {
            distx[count] = x[count+CONTROLP] - x[count];
            disty[count] = y[count+CONTROLP] - y[count];
          }

          for (count = 0; count < CONTROLP; count++) {     /* abs distances between
vertices */
            if (distx[count] < 0)
              absx[count] = -distx[count];
            else
              absx[count] = distx[count];
            if (disty[count] < 0)
              absy[count] = -disty[count];
            else
              absy[count] = disty[count];
          }

          supmax = maxi(absx);     /* find max distance between 2 homogeneous points
(like a and a') */
          supmay = maxi(absy);
          if (supmax > supmay) {
            if (supmax < 0)
              maxdist = -supmax;
            else
              maxdist = supmax;
          }
          else {
            if (supmay < 0)
              maxdist = -supmay;
            else
              maxdist = supmay;
          }


          for (count = 0; count < CONTROLP; count++) {     /* find incrx, incry for all
points */
            incrx[count] = (float)distx[count]/(NFRAMES);
            incry[count] = (float)disty[count]/(NFRAMES);
          }
```

```
            for (count = 0; count < CONTROLP; count++) {     /* assign x, y to newx[0],
newy[0] */
              newx[count] = (float)x[count];
              newy[count] = (float)y[count];
            }

            initvalue = y[0];     /* assumes 1st shape is higher than 2nd */
            endvalue = y[CONTROLP];
            if (initvalue > endvalue) {     /* if not, swap */
              temp = initvalue;
              initvalue = endvalue;
              endvalue = temp;
            }

            if (incry[0] < 0)
              incr = -incry[0];
            else
              incr = incry[0];

            start = CONTROLP;
            end = CONTROLP*2;
            while (initvalue < endvalue) {     /* compute points */
              i = 0;
              for (count = start; count < end; count++) {
                newx[count] = newx[count-CONTROLP] + incrx[i];
                newy[count] = newy[count-CONTROLP] + incry[i];
                i++;
              }
              start += CONTROLP;
              end += CONTROLP;
              initvalue += incr;
              times++;
            }
            return(times);
}

/*********************************************************/

maxi(x)
short        x[];
{
        short i;
        short temp;

        temp = x[0];
        for (i = 1; i < CONTROLP; i++) {
          if (x[i] > temp)
            temp = x[i];
        }
        return(temp);
}

/********************************************************/

intp_curve(x, y, x0, y0, i, curve)
short     x0[], y0[];
float     x[], y[];
short     i, curve;
{
        short count, count2, start, end, j, k;
        short miny = 480, minx = 640, maxy = 0, maxx = 0;
        float temp,temp2,temp3,temp4,temp5, temp6, temp7, temp8, temp9, temp10;
        float t, incr;

        incr = 1.0/NPOINTS;

        if (curve == 0) {
          count = 0;
          k = 0;
        }
        else {
          count = NPOINTS*4;
          k = CONTROLP;
        }

        for (j = curve; j < i; j++) {
          for (t = 0.0; t <= 1.0+incr; t += incr) {
            temp1 = 1 - t;
            temp2 = t - 1 ;
            temp3 = t * t;
            temp4 = temp1 * temp1 * temp1;
            temp5 = temp2 * temp2;
            temp6 = 3 * t;
            temp7 = 3 * temp3;
            temp8 = t * temp3;
            temp9 = temp6 * temp5;
            temp10 = temp7 * temp1;
            x0[count] = temp4 * x[k] + temp9 * x[k+1] + temp10 * x[k+2] + temp8 *
x[k+3];
            y0[count] = temp4 * y[k] + temp9 * y[k+1] + temp10 * y[k+2] + temp8 *
y[k+3];
            x0[count+NPOINTS] = temp4 * x[k+4] + temp9 * x[k+5] + temp10 *
x[k+6] + temp8 * x[k+7];
            y0[count+NPOINTS] = temp4 * y[k+4] + temp9 * y[k+5] + temp10 *
y[k+6] + temp8 * y[k+7];
            x0[count+2*NPOINTS] = temp4 * x[k+8] + temp9 * x[k+9] + temp10 *
x[k+10] + temp8 * x[k+11];
            y0[count+2*NPOINTS] = temp4 * y[k+8] + temp9 * y[k+9] + temp10 *
y[k+10] + temp8 * y[k+11];
            x0[count+3*NPOINTS] = temp4 * x[k+12] + temp9 * x[k+13] + temp10 *
x[k+14] + temp8 * x[k+15];
            y0[count+3*NPOINTS] = temp4 * y[k+12] + temp9 * y[k+13] + temp10 *
y[k+14] + temp8 * y[k+15];
            count++;
          }
          start = count-NPOINTS-1;
          end = count + NPOINTS*3;
          for (count2 = start; count2 < end; count2++) {
            if (x0[count2] < minx)
              minx = x0[count2];
            if (x0[count2] > maxx)
              maxx = x0[count2];
            if (y0[count2] < miny)
              miny = y0[count2];
            if (y0[count2] > maxy)
              maxy = y0[count2];
          }
          bbox[j].top = miny;
          bbox[j].left = minx;
          bbox[j].bottom = maxy;
          bbox[j].right = maxx;
          miny = 480;
          minx = 640;
          maxy = 0;
          maxx = 0;
```

```
                k += CONTROLP;
                count += NPOINTS*4-NPOINTS-1;
            }
}

/*****************************************************/

draw_erase(rmenu, x0, y0, i)
Rect        rmenu;
short       x0[], y0[];
short       i;
{
            RGBColor  black, col;
            PolyHandle poly[NFRAMES+1];
            short j, count, curve, color, start, end;
            short p[2];
            int chn, pitch, vel;
            short rincr, gincr, bincr;

            start = 0;
            end = NPOINTS*4;

            col = cf1;
            black.red = 0;
            black.green = 0;
            black.blue = 0;
            rincr = col.red/(NFRAMES-1);
            gincr = col.green/(NFRAMES-1);
            bincr = col.blue/(NFRAMES-1);

            chn = 1;
            pitch = 60;
            vel = 100;

            RGBForeColor(&black);
            PaintRect(&rmenu);

            MIDI_NoteOn(chn+143, pitch, vel);
            for (j = 0; j < i; j++) {
              count = start;
              RGBForeColor(&col);
              poly[j] = OpenPoly();
            MoveTo(x0[count],y0[count]);
            for (count = start; count < end; count++)
              LineTo(x0[count],y0[count]);
            LineTo(x0[start],y0[start]);
            ClosePoly();
            PaintPoly(poly[j]);
            RGBForeColor(&black);
            PaintPoly(poly[j]);
            col.red -= rincr;
            col.green -= gincr;
            col.blue -= bincr;
            start += NPOINTS * 4;
            end += NPOINTS * 4;
          }
          vel = 0;
          MIDI_NoteOff(chn+143, pitch, vel);
}

/*********************************************************

draw(x0, y0, i, curve, erase)
short       x0[], y0[];
short       i, curve, erase;
{
            short count, j, start, end;
            PolyHandle poly;
            RGBColor black, grey, white;

            black.red = 0;
            black.green = 0;
            black.blue = 0;

            grey.red = 32767;
            grey.green = 32767;
            grey.blue = 32767;

            white.red = 65535;
            white.green = 65535;
            white.blue = 65535;

            if (!curve) {
              start = 0;
            end = NPOINTS*4;
            RGBForeColor(&grey);
          }
          else {
            start = NPOINTS*4;
            end = start + NPOINTS*4;
            RGBForeColor(&white);
          }

          if (!erase)
            RGBForeColor(&black);
          for (j = curve; j < i; j++) {    /* just 1 curve */
            poly = OpenPoly();
            count = start;
            MoveTo(x0[count],y0[count]);
            for (count = start; count <= end; count++)
              LineTo(x0[count],y0[count]);
            ClosePoly();
            PaintPoly(poly);
          }
}

/*******************************************************/

MIDI_NoteOn(midichn, note, vel)
int         midichn, note, vel;
{
            MIDI_Write (midichn);
            MIDI_Write (note);
            MIDI_Write (vel);
}

/*******************************************************/

MIDI_NoteOff(midichn, note, vel)
int         midichn, note, vel;
{
            MIDI_Write (midichn);
            MIDI_Write (note);
            MIDI_Write (0);
}
```

```
/**********************************************************/

int plot(x0, y0)
		short
		x0, y0;
{
		MoveTo(x0, y0);
		LineTo(x0, y0);
}
```

Appendix B

```
SOURCE: grand mean
sound  shape    N    MEAN      SD       SE
               480   0.8896   0.8163   0.0373

SOURCE: sound
sound  shape    N    MEAN      SD       SE
1               80   1.0500   0.7779   0.0870
2               80   0.9125   0.8597   0.0961
3               80   1.1000   0.7730   0.0864
4               80   1.2625   0.7247   0.0810
5               80   0.7125   0.7826   0.0875
6               80   0.3000   0.6038   0.0675

SOURCE: shape
sound  shape    N    MEAN      SD       SE
       sh1     60   0.8667   0.7695   0.0993
       sh2     60   0.9167   0.8087   0.1044
       sh3     60   0.8833   0.8654   0.1117
       sh4     60   1.0000   0.8636   0.1115
       sh5     60   0.8333   0.8061   0.1041
       sh6     60   0.8500   0.7988   0.1031
       sh7     60   0.9667   0.8431   0.1088
       sh8     60   0.8000   0.7983   0.1031

SOURCE: sound shape
sound  shape    N    MEAN      SD       SE
1      sh1     10   1.6000   0.5164   0.1633
1      sh2     10   1.4000   0.6992   0.2211
1      sh3     10   0.5000   0.7071   0.2236
1      sh4     10   1.2000   0.7888   0.2494
1      sh5     10   0.9000   0.7379   0.2333
1      sh6     10   0.8000   0.6325   0.2000
1      sh7     10   1.2000   0.9189   0.2906
1      sh8     10   0.8000   0.7888   0.2494
2      sh1     10   0.4000   0.6992   0.2211
```

2	sh2	10	1.1000	0.8756	0.2769
2	sh3	10	1.2000	0.7888	0.2494
2	sh4	10	1.4000	0.9661	0.3055
2	sh5	10	0.4000	0.6992	0.2211
2	sh6	10	0.7000	0.8233	0.2603
2	sh7	10	1.1000	0.8756	0.2769
2	sh8	10	1.0000	0.8165	0.2582
3	sh1	10	1.1000	0.5676	0.1795
3	sh2	10	1.2000	0.6325	0.2000
3	sh3	10	1.0000	0.9428	0.2981
3	sh4	10	1.2000	0.7888	0.2494
3	sh5	10	1.3000	0.8233	0.2603
3	sh6	10	1.6000	0.6992	0.2211
3	sh7	10	0.7000	0.6749	0.2134
3	sh8	10	0.7000	0.8233	0.2603
4	sh1	10	1.3000	0.4830	0.1528
4	sh2	10	1.3000	0.6749	0.2134
4	sh3	10	1.2000	0.9189	0.2906
4	sh4	10	1.2000	0.9189	0.2906
4	sh5	10	1.6000	0.5164	0.1633
4	sh6	10	1.5000	0.5270	0.1667
4	sh7	10	1.1000	0.7379	0.2333
4	sh8	10	0.9000	0.8756	0.2769
5	sh1	10	0.7000	0.8233	0.2603
5	sh2	10	0.5000	0.7071	0.2236
5	sh3	10	1.2000	0.7888	0.2494
5	sh4	10	0.9000	0.7379	0.2333
5	sh5	10	0.4000	0.6992	0.2211
5	sh6	10	0.4000	0.5164	0.1633
5	sh7	10	1.3000	0.9487	0.3000
5	sh8	10	0.3000	0.4830	0.1528
6	sh1	10	0.1000	0.3162	0.1000
6	sh2	10	0.0000	0.0000	0.0000
6	sh3	10	0.2000	0.6325	0.2000
6	sh4	10	0.1000	0.3162	0.1000
6	sh5	10	0.4000	0.5164	0.1633
6	sh6	10	0.1000	0.3162	0.1000
6	sh7	10	0.4000	0.6992	0.2211
6	sh8	10	1.1000	0.8756	0.2769

FACTOR :	subject	sound	shape	rating
LEVELS :	10	6	8	480
TYPE :	RANDOM	WITHIN	WITHIN	DATA

```
SOURCE          SS        df        MS          F          p
==============================================================
mean        379.8521      1    379.8521   194.404    0.000 ***
s/           17.5854      9      1.9539

sound        47.0854      5      9.4171    11.820    0.000 ***
ss/          35.8521     45      0.7967

shape         1.9313      7      0.2759     0.547    0.796
ss/          31.7979     63      0.5047

ss           45.8312     35      1.3095     2.966    0.000 ***
sss/        139.0646    315      0.4415
```

Aknowledgements

I would like to thank my advisor Tod Machover for the great help and suggestions he gave me through the two year period that led to this thesis.

I would also like to thank all the people that have contributed to the realization of this thesis and of the works I describe in it. Specifically, I cite Lee Boynton, Barry Vercoe, Phil Sohn, Ron MacNeil, Greg Tucker, Stuart Cody, David Zeltzer, Bob Sabiston, Dave Chen, Tomoyuki Sugiyama, Tom Sullivan, Caroline Palmer, Wendy Plesniak, Alan Ruttenberg, Rainer Boesch, Carlo Serafini, Marina Martini, Dave Cumming, Dave Fleuchaus, Suguru Ishizaki.