COMPUTER GRAPHIC AIDED MUSIC COMPOSITION

T. Goldberg* and G.F. Schrack†
*Faculty of Education, (Department of Music Education),
University of British Columbia
†Department of Electrical Engineering, the University of British
Columbia, Vancouver

Abstract

Computer-aided design systems need not be restricted to technical tasks. They can be useful in a much larger area including, in particular, the fine arts. Computers as a tool have been used for quite some time to aid artists in producing graphics and to aid composers in producing music.

The project described has been designed to unite both visual and audio compositions, to be presented simultaneously in multi-media performances.

Using a high-level graphics programming language, an applications program was written to the specifications of the composer.

COMPOSITION MUSICALE À L'AIDE DE MÉTHODES GRAPHIQUES INFORMATIQUES

Résumé

Les systèmes d'étude à l'aide de l'ordinateur ne sont pas exclusivement réservés à des travaux techniques. Leur champ d'application est en effet beaucoup plus vaste et s'étend notamment au domaine des beaux-arts. Depuis fort longtemps, les artistes utilisent l'ordinateur pour produire des représentations graphiques et les musiciens s'en servent pour la composition.

Le projet décrit a pour but d'intégrer des compositions visuelles et musicales, de façon qu'elles fassent l'objet de présentations simultanées.

À l'aide d'un langage évolué de programmation graphique, on a rédigé un programme d'application en fonction des besoins du compositeur.

Introduction

It should be mentioned at the outset that we are not to write about computer sound synthesis and composition. This is already a fairly well developed, ample field, with a history of its own. To give an adequate overview of it would take considerable time and expert knowledge. What we will do is describe a very special programming strategy, in which the graphic representation of a piece of music plays a generative role. It will be followed by a description of some of the experiences gained during the project regarding man-machine communication processes.

The Programming Strategy

Since the programs we use are interactive, they make effective tools and instruments which offer immediate access to both research and creative activity in the field which interests us most: the correlation of sound and image. This is still a somewhat grey area suffering from ambiguity and capriciousness in both conceptualization and practitioning. The computer programs offer precision and authentication in one-to-one relationships between sound and image. They do not, however, represent a boundless paradise: one works in a severely restricted medium which requires a great deal of self-denial and discipline.

The final product is a graphic image which is directly related to — and actually generates at the point of its own generation — a piece of music, which may or may not be electronic. There are three stations in the process which are in sequence, although we need not be orthodox about it. The first is the graphics program CAMUS (from computer aided music), the second the operating system command sequence which executes a plot, the third, a program for sound synthesis and composition called POD6[3].

The graphics image — which is really a euphemism for the computer plot — is in the sound synthesis referred to as a frequency-time distribution mask or Poisson mask. It specifies where in pitch and when in time sounds theoretically may occur[4]. The actual calculation where and when sound events take place is determined using program options which define the two-dimensional frequency-time space and calculate within its points which represent sound events. This two-dimensional space with its coordinates is termed frequency-time distribution, since statistical procedures are used to determine the coordinates' location. The function of the frequency-time mask data then is to delimit, for any given point in time, the frequencies between which a sound event may occur. The mask data are produced at the instant the points and lines of the design at the graphics terminal are generated.

The question that is being asked most frequently is whether one writes the music first and makes the design after or vice versa. The answer is that one has to create the concept of the musical composition and the graphic design simultaneously. Of course it would be possible to make the tendency mask of any composition from Palestrina to Stravinsky and obtain the plot with the aid of the computer; but that is contrary to the very nature and the aesthetics of our work. Since any point or line written at the graphics terminal instantly generates information for the sound synthesis, the generative process is truly correlative. The artist-composer cannot afford to proceed to work in one medium at a time and ignore the effect on the other. If for instance one chose to begin with the graphic design and disregard or

defer the sonic cross-check, the musical outcome would be unaccountable. Although we use random or chance selections in this medium, we do apply controls to make results predictable.

Figure 1 is an example of how one medium imposes strictures upon the other. This is a composition of a Latin text by Ovid to be sung by

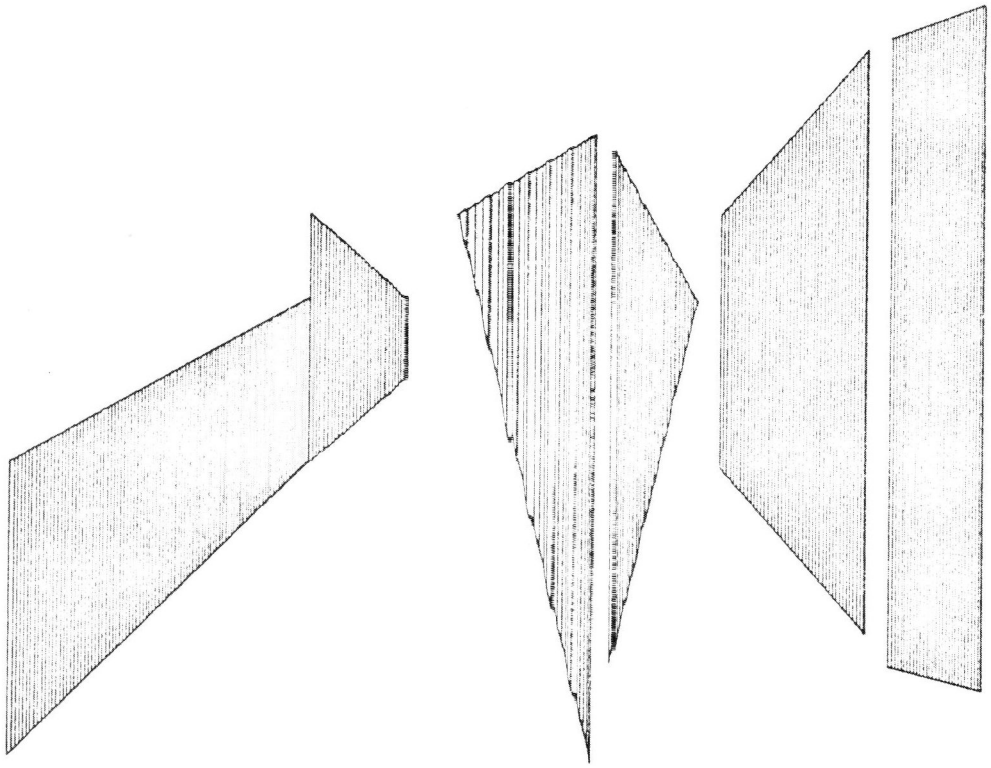

Figure 1

a mezzo-soprano. The horizontal expanse of the design represents the duration of that piece. It happens to be four minutes, but could be four seconds or four hours, depending upon the basic temporal unit one specifies. Since the text has distinct paragraphs, there are separate sections in the design. The sections have to be in correct proportions to each other according to the number of syllables contained. The total number of syllables was counted to provide the computer with the total number of events to be calculated. The poem was read to arrive at a reasonable diction which would provide the computer with the information to create the correct density of sound, represented here by the vertical lines filling the individual shapes. Because we are dealing with text, the increase or decrease of the densities is negligible. The vertical axis represents the distribution of pitches. The range of voice of the soloist determined the scale chosen to allow the vertical expanse of the shapes to have harmonic proportions. In all that, the general appearance of the design, the interaction of the shapes with each other, their perspective and their sequence had to be considered, because there was a supposition of an inherent symbolism which may not be apparent to this audience because it does not see the images preceding and following this original plot.

After the design has been made at the graphics terminal, the data is now transferred via paper tape to the sound synthesis program. This information provides the computer with the macro-form of the work, that

is, its duration, the distribution of pitches, and the density of sounds. What remains to be done is the micro-form, that is to program the individual sound objects which are distributed over the composition. Since the program is interactive, all paramenters of sound can be carefully specified and immediately tested. The paramenters are

(1) Amplitude envelope, specifying the duration of attack, sustain, and decay of the sound.

(2) The ratio between the carrier frequency and the modulation frequency. This is the most important factor in shaping the timbre of the object. On a continuing scale expressed in integers from 1 to 500, this option provides an inexhaustible reserve of timbres.

(3) The maximum modulation index, that is the strength of the spectrum; the index determines the largest number of partials significantly present in the spectrum.

(4) The modulation index envelope, that is the temporal behaviour of the spectrum determined by the change of the modulation index during the duration of the sound.

Having already specified the frequency and density of masks with the graphic design, one moves now to amplitude selection, the choice of methods for controlling the loudness of each event. Then follows the sound object selection. This is the assignment of the specified sound objects to a given range of events. The options are simple aleatoric, weighted aleatoric, time-varying aleatoric which is again specified with a tendency mask, and sequential as well as permuted sequential choice. Further options take care of performance variables, something akin to the various interpretations possible in performances of a piece of classical music.

Finally, the POD6 program may serve as input to related programs which process the piece further. One such program facilitates transcription into conventional notation. Most recent developments produce overlap of events and digitally calculated reverberation[4].

Specifications

A number of reasons were instrumental from the outset for the desire to assist the composer and to be involved in such a project. Firstly, it implied playing a part, albeit a small one, in an exciting artistic project. Secondly, the high-level graphics language LIG and program system [2], developed in house, was available and is obviously well suited for it. The system had been tested and used on numerous occasions by undergraduates for many projects but had never been used for a "real life" project. Thirdly, the composer had had no previous experience with computer systems, hence a rare opportunity presented itself to study man-machine communication processes. In retrospect, it was indeed worthwhile to take on the project and, as hoped, a number of insights, some unexpected, resulted.

The initial specifications agreed upon were the following.

(1) Write an application program (CAMUS) which allows the artist to create interactively sequences of graphical objects called segments (sets of which make up a tendency mask) within a frequency/time coordinate system. The segments observe a number of characteristics:

(i) The parallel sides are always parallel to the frequency axis.

(ii) A trapezoid may degenerate into a triangle or a line.

(iii) The segments are ordered in contiguous sequences, i.e. no time gap may appear between them. (A pause is represented by a degenerate trapezoid of very low frequency.)

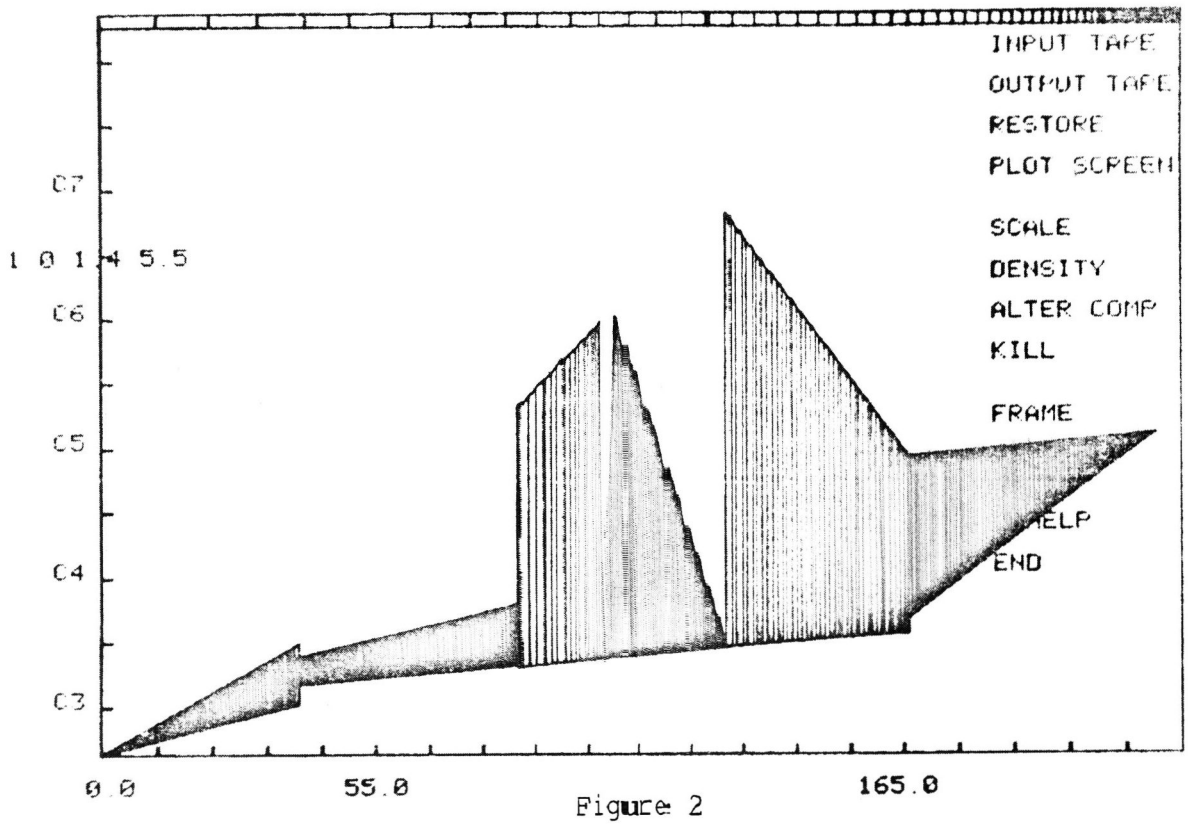

Figure 2

(iv) Attached to the start and end time of a segment are two density values. They are represented graphically by vertical density bars "filling" the trapezoid. Thus, a single segment is defined completely by the set

$$S_i = \{\Delta t_i,\ f_{ij},\ d_{ik} \mid j=1,2,3,4;\ k=1,2\}$$

where $\Delta t_i = t_{i2} - t_{i1}$, the duration,
f_{ij} = four frequencies defining the corners of the segment,
d_{ik} = the two density values.

A tendency mask is given by a sequence of segments S

$$T = \{ S_i \mid i=1,2,\ldots,N \}.$$

(2) The input of the segments will be achieved by direct input at the screen. Soon after the first trials of CAMUS it became obvious that input should also be possible from a predesigned sketch, to be followed, if desired, by alterations, i.e. graphical editing, from a previous session via disk file or a paper tape, or to match a plot representing a finished tendency mask to allow composition of several concurrent channels.

(3) It should be possible to direct the output to be graphical on screen, graphical on plotter, numeric on paper (a listing), numeric on paper tape (as input to the POD6 program) and on disk file to allow continuation of a prematurely terminated session.

(4) The applicaction program should allow interactive editing capabilities, specifically additions to a previous sequence (which may be the null sequence), changes to a previous sequence (alterations, insertions, deletions), changes of scale of axes e.g. dilation of the frequency or time scale to allow more detail or more precision, or selection of subintervals to allow the artist to move back and forth over the time scale at will or to concentrate on a specific range of fre-

quencies, and compression of the time scale to allow seeing the "macro-structure of a composition".

Implementation

Since the language for interactive graphics (LIG) had been implemented for some time on the departmental computer, there were no difficulties anticipated nor encountered for this application program.

However, a program of the specified complexity requires considerable planning, effort and attention. LIG was specifically designed for writing interactive application programs. Its characteristics proved the usefulness of the incorporated concepts in a number of instances:

(i) LIG allows to segment with ease programs which are too large for the mini-computer environment used.

(ii) With LIG it is easy to create screen menus and their associated decision structures.

(iii) The handling of graphical input and output presents no difficulties.

Also, LIG was specifically designed for drawing highly structured technical schematics and plans which are composed of many topologically equivalent symbols. The graphical objects for CAMUS (the segments) do not possess this invariance; they can be represented only by graphical functions. Thus, a major feature of the LIG system was used only sparingly.

Man-machine communications

As has been mentioned, one major interest in the project were the man-machine communication aspects. Unless one is forced into a genuine design situation, all such considerations remain theoretical and hence unproven.

Fortunately, the programmer agreed at the outset on the major principle guiding the design, and on the philosophy on how to implement it. This principle is the following.

> MACHINES MUST BE ADAPTED TO THE HUMAN USER AS MUCH AS POSSIBLE

Its corollary,

> NEVER EXPECT A HUMAN TO ADAPT TO A MACHINE IF THIS CAN BE AVOIDED

is equally as important. (In this context, machine refers to both the hardware and the software.)

Here are some examples where we tried to observe these principles. Their order is not meant to signifiy their importance.

(1) Freedom of sequence selection. Any symbol on the menu can be selected at any time during the design without interrupting or confusing the program. The definition of a segment by its four points must be possible by any of the 24 permutations of the four frequencies.

(2) Feasibility of input. After each single input operation, the datum is checked as to possible errors or inconsistencies. If one is detected, the condition is signalled by a message and renewed input is asked for without aborting the design sequence.

(3) Provision for redundancy on input. All string menu buttons begin with a different letter. The selection of a menu button can be

made either with the cross hairs or by striking a key with that letter. Densities can be specified by numeric key input or by cursor input.

(4) Interruption of an action sequence. The user is not forced to complete a sequence of actions (usually input actions) which normally are carried out as a set. The sequence may be interrupted for any reason at any time by selection of another action.

(5) Cancellation of the immediately preceding action. The user may change his maind regarding the last sequence of actions by the cancellation command "undo".

(6) Data back-up. All pertinent data are immediately saved on disk in a back-up file such that any forced interruption or termination does not imply a loss of the design session.

(7) Action echo. All actions are acknowledged by the machine either visibly or audibly. For example, after selection of a menu button, the menu button is echoed.

(8) Prompting of actions. The machine must never get into a wait state without signifying so. First output a message describing what is expected, then expect the input action.

(9) Assurance of ongoing operation. If lengthy programming actions are necessary, periodic output of progress report messages should appear on the screen in order to assure the user of the ongoing action. Users should be conditioned never to wait patiently for the machine unless it urges him to do so.

(10) Prompting and error messages must be explicit, precise, specific, reasonably short and courteous.

(11) Use of language. The communication of the machine with the user must employ the language of the user, not that of the system programmer. In particular, computerese, acronyms and colloquial expressions must be regarded as strictly illegal, no matter how obvious or acceptable they may seem to the programmer.

(12) Help states. Detailed explanations of any program action can be solicited by the user by activating a help request button.

(13) Uniformity of required response action. A program often requires from the user certain character strings, e.g. numbers, file names, etc. Such character sequences obviously must be terminated by a special character, the "RETURN" key. However, at times the choice of a single character suffices as input action (e.g. striking a key in lieu of a menu button). Nevertheless, that single character should also require the string terminating character to insure uniformity of all input actions.

(14) Simplification of command sequences. Usually, an application program is imbedded in a formidable hardware and software system environment. In the case on hand, a Nova 840 with graphics terminal, tablet, disk, and paper tape can communicate on request via a data link with a 370/168 and a Calcomp plotter, while the synthesis program resides in a HP-computer with its associated peripherals. Besides CAMUS and the POD6 programs there are three different operating systems to contend with. For a non-technical person, this can lead to untold confusions, the possibility of which must be reduced as much as possible. The generation of a plot, for example, requires sign-on on the 370, start-up of the data link and issuing complicated sequences of system commands on the two machines. Such command sequences should be stored in files or programs which are to be activated as a single unit.

(15) Documentation. In fact, considerations such as above point to the importance of the documentation of an application program. It

must suggest a number of actions for possible (and "impossible") situations. Naturally, these can only be collected by actual observation of the user in action. One very important command sequence is the one which allows an orderly abortion of the design session and shut-down of the machine under all circumstances. The documentation should be internal (i.e. the help state of the program istelf) as well as external in a users´ manual.

(16) Man-man communication. Although not unexpected, it was interesting to experience the degrees of difficulty in communicating with a colleague from another faculty. A statement from one, when repeated by the other in his own language, sometimes does not verify that complete understanding between the two has been reached. Such man-man communication problems are but a small indication of the importance to be attached to man-machine communication mechanisms. Also, suggestions for improvements must come from the implementors, not the user alone, since it is difficult for the user to assess what can be and what cannot be accomplished by the machine. A good application program therefore develops and evolves during a certain time span.

As is well known, above suggestions and insights are not all new, see, e.g. Martin[1]. Yet only a situation such as the one described brings to light their relevancy.

Conclusions

Without the use of the computers, much of the work would have been difficult or impossible to do. Involvement in a project of continued importance and interest for someone else generates needed motivation and enthusiasm for the computer scientist.

Acknowledgement

D.W.L. Teeple has accomplished the design and implementation of CAMUS with a great deal of sensitivity and grasp of musical concepts.

References

[1] James Martin, Design for Man-Computer Dialogues, Prentice-Hall, Englewood Cliffs, 1973.

[2] Guenther Schrack, "Design, Implementation and Experiences with a High-Level Graphics Language for Interactive Computer-aided Design Purposes", Computer Graphics (ACM-SIGGRAPH), vol. 10, no. 1 (1976), pp.10-17.

[3] Barry Truax, "The POD Programs for Sound Synthesis and Composition at SFU", unpublished manuscript, Vancouver, 1975.

[4] Barry Truax, "A Communicational Approach to Computer Sound Programs", Journal of Music Theory, vol. 20, no. 2, Fall 1976.