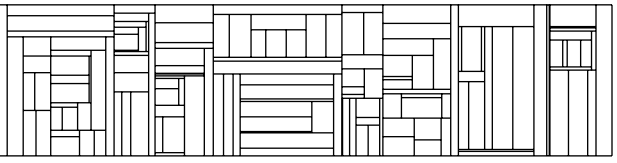
The EGG
A Purely Digital Real-time Sound Synthesizer

Michael J. Manthey

DAIMI PB - 56
March 1976

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF AARHUS
Ny Munkegade, Bldg. 540
DK-8000 Aarhus C, Denmark



The Egg

A Purely Digital Real-time Sound Synthesizer

by

Michael J. Manthey‡

Abstract

The goal of this project was to produce a "state of the art" keyboard instrument, with special attention paid to the requirements of live performance. The article describes the basic characteristics and capabilities of the result.

‡ Author's Address: Dept. of Computer Science, University of Aarhus, Ny Munkegade, 8000 Århus C Denmark

0.0 INTRODUCTION

The purpose of this article is to describe in general the capabilities of the Egg, a real-time digital musical instrument. This instrument distinguishes itself from other synthesizers by its generality and ease of use, capabilities which follow directly from its computer-based technology. In particular, the dominating criteria from the project's inception four years ago has always been that the final result be a musical instrument, in the fullest sense of the word.

It is the author's opinion that other contemporary synthesizers do not live up to this criterion for the following reasons:

a) Difficulty in changing registration - the patch-cord and switch matrix approach to sound specification is ill-suited to performance demands, both because of its complexity and its awkwardness. A performer cannot insert a five-minute (or even five-second) pause into a solo in order to twiddle dials and change connections.

b) Repeatability - it is a well-known problem for synthesizer users to remember the interconnections and settings which yield a particular sound. The level of detail at which the musician is forced to operate, while being a great improvement over first generation equipment, still represents a built-in obstacle for the realization of non-trivial compositions.

c) The mono-voice problem - the first synthesizers only permitted a single tone at a time to be played. In order to allow multiple simultaneous tones, as has been done with newer synthesizers, the circuits involved must be physically duplicated as many times as there are to be multiple simultaneous tones. Clearly economics plays a role in the extent to which such a solution is practicable. With analog technology, all the parameters of a sound are 'computed' by the circuits involved as the sound is produced. This fact has two implications: (1) that all parts of the circuit must be physically present, and (2) that none of the sound parameters can be 'pre-computed'. The former means that the mono-voice problem disappears

only insofar as the money exists to pay for the level of replication desired. The latter means, in essence, that there is no alternative i.e. it will be difficult for analog synthesizers to extract themselves from the replication bind by e.g. mixing digital and analog technologies. Given that synthesizers are viewed as the inheritors of the keyboard traditions of first the organ and then the pianoforte (i.e. symphonic instruments), and not special-effect gadgets, it is critical that the mono-voice problem be solved in a less limiting and more satisfactory manner than appears possible with analog technology.

d) Component precision - the sensitivity of the human ear requires that the components used in analog synthesizers be manufactured to very high levels of precision. Besides being therefore more expensive, they are also more sensitive to the environment e.g. temperature. Digital components, in contrast, are much less influenced by these factors.

It is these drawbacks of analog synthesizers, and especially their crippling effect [except (d)] on live performance, which inspired the Egg's construction. There have been and are a number of attempts to build synthesizers which mix the two technologies, but it is this author's feeling that this approach, while perhaps yielding a quick solution of the many problems in the short run, is inevitably doomed by the limitations inherent in analog technology. Consonantly, the Egg is purely digital except for the final digital-to-analog converter on the output end. It is not however the author's contention that the design presented herein has not replaced these problems with others equally undesirable. As an example, a large amount of analog componentry has been replaced by software, a medium notorious for its unreliability; it also appears that any possibility for feedback is eliminated by the Egg's approach. On the other hand, the following chapters should demonstrate that the general purpose synthesizer's musical horizons have been significantly enlarged.

1.0 A CLOSER LOOK AT THE EGG SYNTHESIZER

This section contains a general overview of the Egg Synthesizer and the strategy it uses to represent and produce sound.

1.1 An Overview of the Instrument

Figure 1-1 is a block diagram of the current synthesizer, which includes

1. Two piano-like keyboards
2. A (Texas Instrument 960A) computer
3. Sound Generator + Digital-to-Analog (D/A) converter
4. Input terminal
5. Action Buttons
6. Paper tape reader and puncher
7. Disk

1.1.1 The Keyboards

The Egg has two touch-sensitive keyboards, which may be played upon simultaneously. Physically, the 'touch' of the keyboard resembles that of an organ, but once the performer becomes accustomed to the keyboard, the nuances of key motion (yielding velocity and acceleration) are available as dynamic parameters to the sound patterns (see 1.3 and 2). Reference [5] describes the mechanical and electronic design considerations for the keyboard, which uses photocells to track Key motion. See Figure 1.1.1.

As far as the two keyboards themselves are concerned, one has the layout of an ordinary piano keyboard, i.e., two black keys signifying $C^{\#}$ and $D^{\#}$, and three more for $F^{\#}$, $G^{\#}$, and $A^{\#}$. The other keyboard, intended for experimentation with microtones, has a black key between <u>every</u> white key, and thus is effectively a whole tone keyboard. In addition, between each black key and white key is a (e.g. gray) key, allowing a further subdivision of the whole tone interval. Lest the reader feel that such a departure from the standard diatonic format impliesunwarranted complication for the performer, we must point out that as soon as one begins to deal with non-diatonic scales, the traditional diatonic keyboard format becomes more and more of an albatross. It is felt that a symmetrically laid-out keyboard in such circumstances

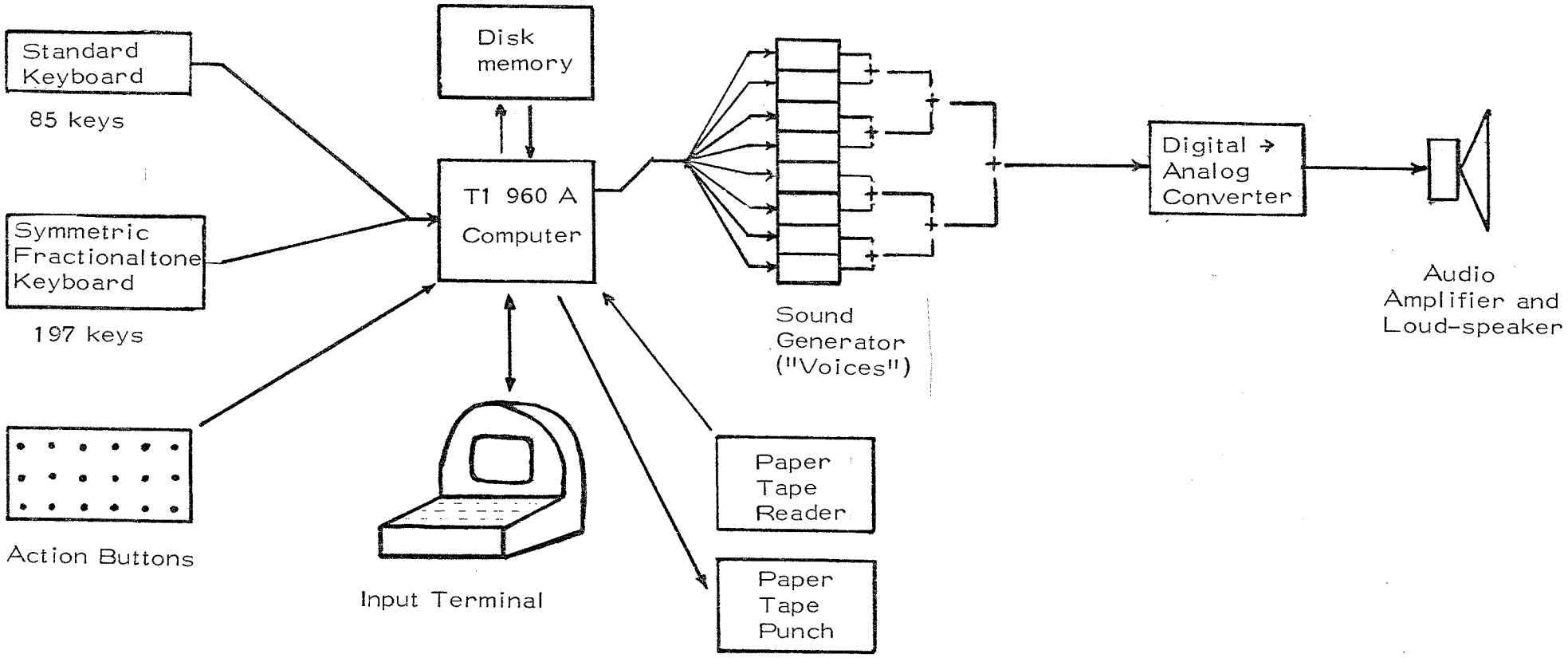
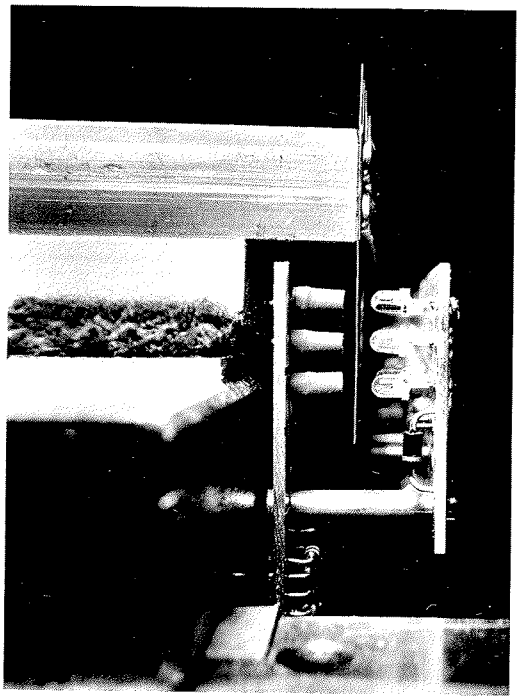


Figure 1-1

The Egg Synthesizer



Above: The EGG Keyboard
Left : Keymotion Sensor (detail)
(see 1.1.1.)

Figure 1.1.1

is in fact easier to play on than a non-symmetric one.

Both keyboards may be "registrated" in two independent ways:

A. frequency (pitch) registration.
B. sound-type registration.

Point A refers to the fact that it is possible to define dynamically the fundamental frequency associated with each key, which facility can be used to change the tempering of the scale or to divide the keyboards up into groups of frequency ranges. Point B refers to the fact that it is possible to define dynamically the 'klangfarbe' (timbre) associated with each key, which facility can be used to divide the keyboards up into regions having different timbres. These two facilities thus allow the user to define arbitrarily what pitch and sound is to be produced by each key.

These definitions are specified to the Egg by the user by typing appropriate commands on the input terminal (described below). For purposes of keyboard registration, the two keyboards are considered one long keyboard, i.e. the symmetric keyboard is an extension of the diatonic keyboard. Further information on how registration is actually accomplished can be found in [2.2].

1.1.2 The Computer

The device which makes all the various capabilities (and limitations) of Egg possible is the mini-computer which sits in the middle of the whole and interprets and controls all the events in the system. It is sufficient for our purposes here to say that the computer's most important component is its memory, which contains both the program which is the synthesizer's "brains", and the necessary data e.g. waveforms, sound patterns, registrations, etc. It is the job of the synthesizer program to interpret what is happening on the two keyboards, translate these happenings to sound patterns, and cause the sound generator to make the appropriate sounds. It is also its job to interpret the commands written by the user on the input terminal and make the commanded modifications to the data structures in the memory.

1.1.3 The Sound Generator and D/A Converter

The sound generator is the means by which the sound patterns, which consist of waveform frequency, amplitude, and time specifications, are ultimately converted to sound.

The process is as follows: a waveform [represented by 64 12-bit numbers] is sent out to the sound generator, together with integers representing frequency and amplitude. The frequency value is counted down at a rate of 20 MH_z, and each time it reaches zero, the next waveform point is fetched, multiplied by the amplitude, and sent out. The digital-to-analog converter is the final step in this conversion, translating the high-speed streams of "finished" numbers from the sound generator into analog voltage signals suitable for driving an ordinary audio amplifier. This process continues until new commands are received from the program e.g. stop, new waveform, etc.

The sound generator consists of a number of channels or "voices", each containing a waveform buffer and frequency and amplitude logic, and at the time of this writing there are sixteen such voices. Reference [6] describes the sound generator in greater detail.

1.1.4 Input Terminal

This device consists of a typewriter-like alpha-numeric keyboard and a TV screen. On the input-terminal's keyboard, the user can specify registrations, define timbres (sound patterns), and ask the program to write on the screen (hopefully) helpful information, to name some of the possibilities.

In the remainder of this article, the "input terminal keyboard" will be referred to as the "alpha-keyboard", to differentiate it from the symmetric or diatonic (music) keyboards, which are referred to simply as "the keyboard" or "the keyboards".

1.1.5 Action Buttons

There are a number of commands to Egg which could well be typed on the alpha-keyboard, but are so common that it is desirable to somehow abbreviate them. Examples are changing registration while playing on the keyboards, and controlling the Pseudo-Tape (see 4.0). This abbreviation is accomplished by the action buttons, which need only be pressed down in order to give the associated command.

1.1.6 Paper Tape Reader and Puncher

These two devices are responsible for the physical reading and punching of paper tape. Paper tape allows the user to save copies of sound patterns and registrations, and later read these copies back into the synthesizer. Recordings made by the Pseudo-Tape can also be saved and read via paper tape. Thus, paper tape gives the user a means, which is independent of what other users do with Egg in the meantime, by which he can save his "work".

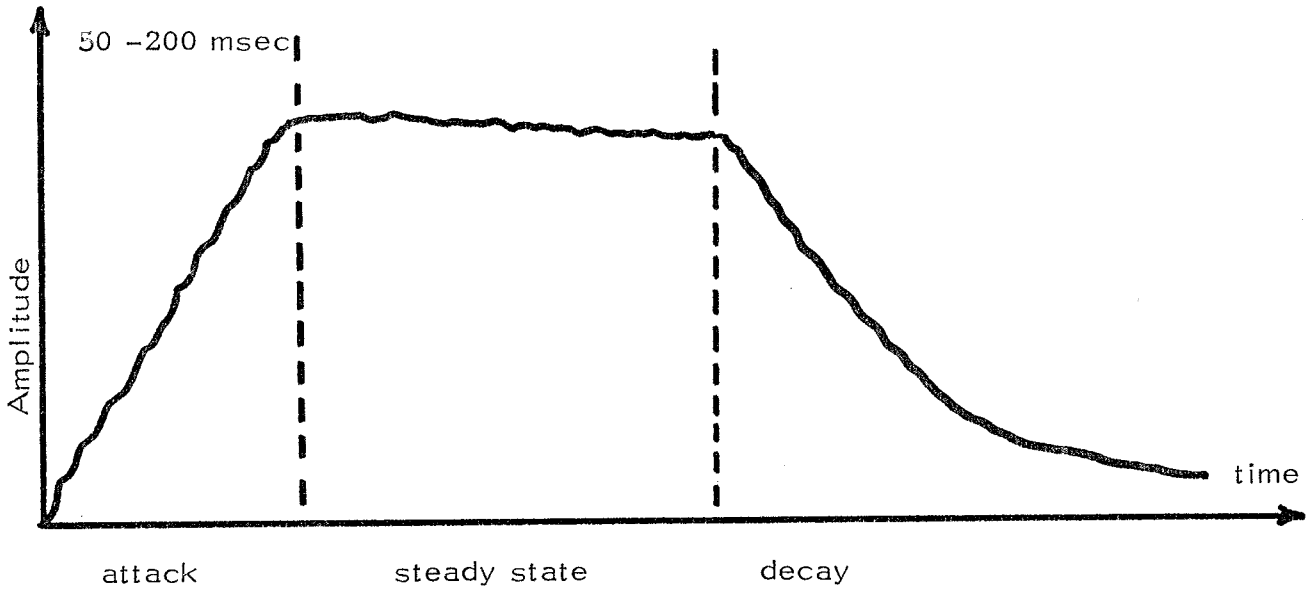
1.1.7 Disk

The disk provides a partial replacement for the paper-tape reader and puncher, and is much more pleasant to use since it is so much faster. It is a "partial replacement" since its capacity is finite, and also because its contents can only be communicated via paper tape to other computers e.g. to produce scores and parts from pseudo-tape recordings.

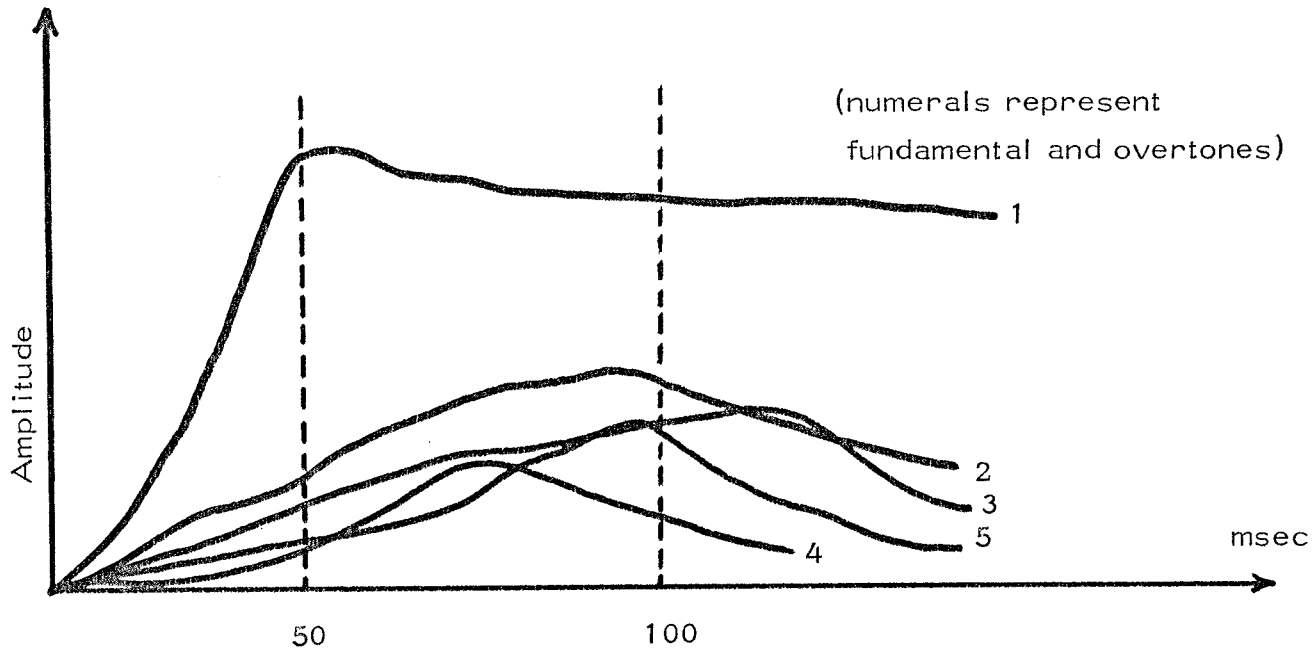
1.2 A Quick Survey of Real Sounds

This paragraph deals with the characteristics of "real" sounds, which in our context we mean real music sounds, such as those made by flutes, violins, etc. An understanding of these characteristics will help in understanding the Egg's strategy for sound production, which we call "sound patterns". It is these sound patterns which the user defines and uses to produce sound via the keyboards.

It has been well established that while most musical sounds are very complex, they can be broken down into three pieces, corresponding to the qualitative changes in the sound as time progresses. The first of these parts is called the attack, which corresponds to the first 50-200 ms of the tone, and is characterized by very complex changes in the frequency, phase, and amplitude spectra; this part, as far as the ear is concerned, is most characteristic of the particular instrument involved. The second part, somewhat misleadingly called the steady state, is characterized by a more or less stable spectrum, and conveys less semantic information to the ear about the identity of the sound (e.g. horn or clarinet) than the attack. The third stage of a sound is called the decay, and is characterized by a falling off of amplitude to a level below the perceptive threshold; it also carries the least differentiative information. Figure 1-2a below shows an idealized diagram of these three stages, and 1-2b an actual sound spectrum for a clarinet.



(a) Typical "Real" Tone



(b) Clarinet (attack)

Figure 1-2. The Three Stages of a Sound

1.3 Egg's Basic Sound Synthesis Strategy

The approach taken in the design of the Egg's sound generation scheme is to imitate this simple model of a sound, while at the same time allowing as much flexibility as conscionable. Thus the user is concerned with defining patterns for a sound, with particular details such as frequency, amplitude, and duration to be filled in at performance time - but a pattern which corresponds to the above attack - steady state - decay model. The user specifies the fundamental waveform and amplitude envelope for each stage of the tone's life, and this "sound pattern" (when activated by a key on the keyboard) is then interpreted by the synthesizer to yield specific numbers and commands to be sent to the sound generating hardware.

To elaborate a little on that last sentence, it is the user's job to build sound patterns, patterns which consist of

1. a sequence of waveforms, which thus determines the timbre of the sound;
2. a sequence of amplitude envelopes, which determines when the sound is loud and when it is soft;
3. some other control information.

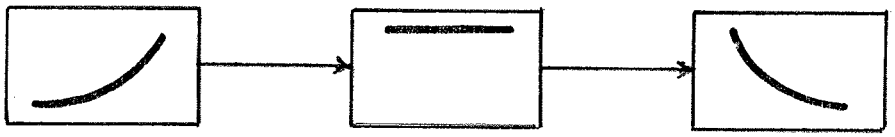
An example of a sound pattern which models Figure 1-2 is shown in Figure 1-3.

When the pattern of Figure 1-3 is activated, the square waveform is sent to the sound generator, along with the frequency this waveform should be cycled (around and around) at. [The frequency was derived from which key on the keyboard was depressed.] The sound generator also receives the amplitude envelope and the fact that all this information is valid for 100 milliseconds. At the end of the 100 msec, the sawtooth waveform, frequency, "constant" envelope, and 3 seconds are sent to the sound generator. Finally after these 3 seconds are elapsed, the sine waveform, frequency, decay envelope, and 2 seconds are sent out. The sound thus dies after 5.10 seconds. (This explanation is a slight oversimplification, but good enough for now.)

a. A sequence of waveforms to model a "typical" sound might consist of a square wave for the attack portion, a sawtooth wave for the steady-state, and a sinewave for the decay. Thus, our waveform sequence is

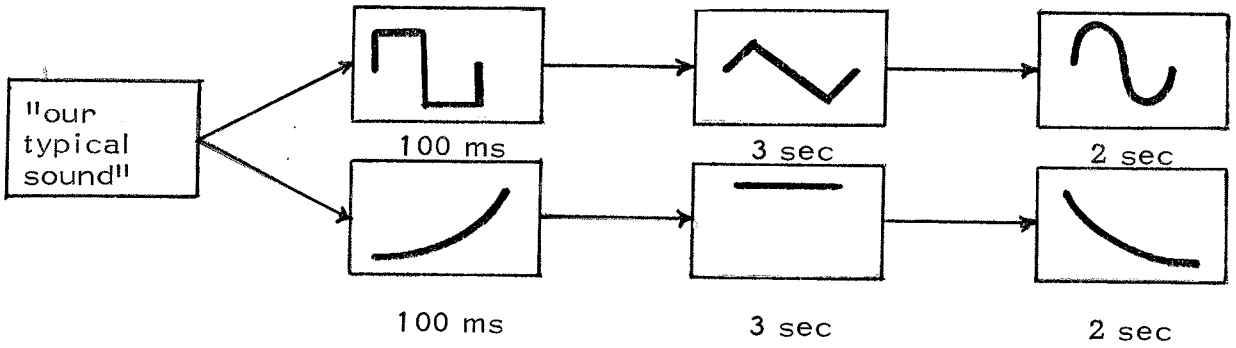
attack steady state decay

b. A sequence of amplitude envelopes to model the "typical" sound might consist of a rapidly increasing amplitude in the attack, constant amplitude during the steady state, and decreasing amplitude in the decay. Thus



attack steady state decay

c. The complete pattern, with some control information added, is



It should be emphasized that, while this pattern would make an acceptable sound, it is very primitive in comparison to the patterns that could have been devised for the same "typical" sound.

Figure 1-3.

A Primitive Sound Pattern.

2.0 THE STRUCTURE OF SOUND PATTERNS

This chapter contains a complete description of the elements which make up a sound pattern, and of how they can be used to build complex sounds.

2.1 Elements of a Sound Pattern

The sound pattern of Figure 1-3c illustrates the three major components of a sound pattern: the Name Head of the sound, the list of waveforms (the Tone List), and the list of envelopes (the Envelope List). We now examine each of these.

2.1.1 The Name Head

The Name Head (henceforth spelled "namehead") is the component of the sound pattern which contains, unsurprisingly, the name of the sound pattern. Examples of valid names are: OBOE, VIOLIN, BASS, DBL, A123. Besides the name of the sound pattern, the namehead contains two other items, called "preparation procedures". These are used to define those items in the pattern which are initially undefined (e.g. frequency), and are explained in [2.2].

The name in the namehead of the sound pattern is the means by which the user refers to the pattern using the command language.

2.1.2 The Tone List

The Tone List (henceforth spelled "tonelist") is the component of the sound pattern which lists the sequence of waveforms, frequencies, and durations for the sound. Each block in the tone list contains the following items:

a. the name of a waveform;
b. the length of time this waveform is to be played;
c. a timer value [explained in 2.3];
d. a "pointer" to another block in the tonelist [see 2.3];
e. "Pointers" to three sequence-control procedures [see 2.3];
f. the frequency at which the waveform is to be played [see 3.3].

It should be clear that items a, b, and f are those needed to actually define a physical sound. The others (c, d, e) are used to control more dynamic as-

pects of the pattern e.g. what action to take (in the sound) when a "key-off" signal is received from the keyboard, vibrato control,etc. These are explained in [2.3].

It is appropriate at this time to define the exact Egg values for these items:

1. A waveform consists of 64 points, which may range from 0 to 4095. The only restriction is that the 64'th point must have the value 2048. By way of example, a sine wave is usually defined between -1 and +1. To enter a sine wave into the Egg, each of the 64 points (making up the approximation to the sine wave) must be converted to the range 0 to 4095, with the original zero of the sine being mapped onto 2048. It is not, however, necessary for the user to concern himself with these matters, as he may use the predefined waveforms.
2. Frequency may range from 10 Hz to 20,000 Hz.
3. The duration of a given block may be from 3 msec to 32 seconds.
4. There is no theoretical limitation on the number of blocks in a tonelist, although at the present there is an implementation restriction (easily changed) of 16.

2.1.3 The Envelope List

The Envelope List (henceforth spelled "envlist") is the component of the sound pattern which lists the sequence of amplitude envelopes to be applied to the sound. Each block in the envelope list contains the following items:

a. the name of an envelope;
b. time between envelope points;
c. the Timer [see 2.3];
d. a "pointer" to another block in the envlist [see 2.3];
e. "pointers" to three sequence control procedures [see 2.3].

An envelope in the Egg is a set of e.g. 10 points which are sequentially multiplied with the waveform points to effect the final amplitude of the sound. If we refer to the first (i.e. attack) envelope in Figure 1-3c, we see that the

envelope should take 100 msec. to complete, which is to say 100/10 = 10 msec. for each envelope point's duration. It is this time (i.e. the 10 msec.) which is referred to in point b in the above list.

The exact values for these items follow..

1. An envelope consists of any number of points, with values between 1 and 15 The value 1 is equivalent to -42 db, while the value 15 is equivalent to 0 db. The difference between each point (e.g. 13 and 14) is 3 dB. [We expect to decrease this to 1.5 dB.]
2. The time between envelope points is counted in clock ticks of 1 msec.

For the user who is not interested in defining new envelopes, there are a number of predefined envelopes available.

2.2 Preparing the Patterns.

Up until this point, we have discussed the sound patterns as rather static structures i.e. from the point of view of their definition, rather than their use. In this and the following sections, we examine what happens with the patterns when they are activated (by a key-stroke) and used to produce sound, We therefore begin our discussion by seeing what happens when a key on the keybord is depressed deeply enough to cause the key to be considered "ON". At this point, the sound pattern associated with that key [see 3.2] must be particularized with the data it has heretofore lacked: frequency, and perhaps tone duration and attack-envelope speed (based on the key's velocity).

It is this particularization of the sound pattern for each individual activation of it that is performed by the "preparation procedures" first mentioned in [2.1.1] in our discussion of what is found in the namehead. There are two preparation procedures, one for the tonelist and one for the envlist. The tonelist preparation procedure is a subprogram of the synthesizer which operates on each block of the tonelist performing the same action in each to prepare it for use in sound generation.

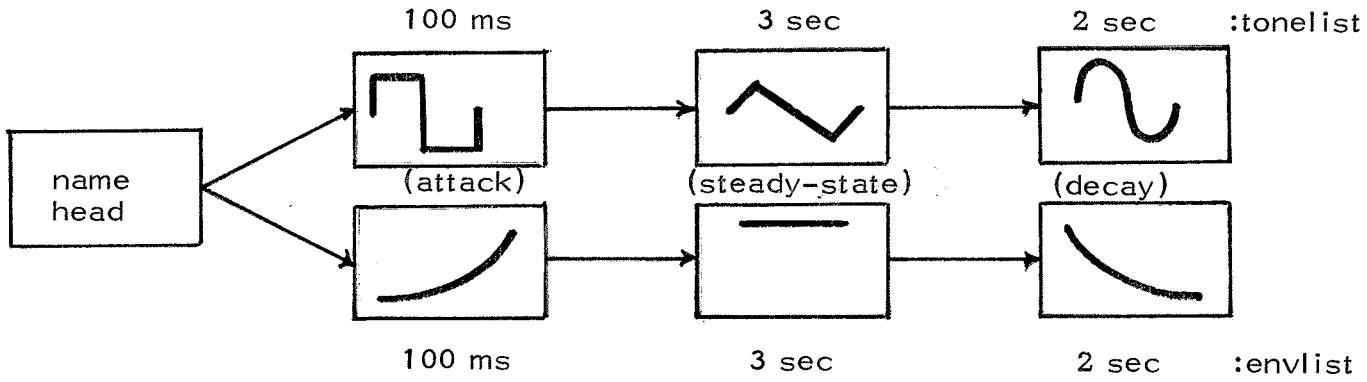
Envlist preparation procedures operate in manner completely analogous to the tonelist preparers, except of course they are concerned with envelopes instead of frequencies and waveforms.

Several different preparation procedures are available for each list, depending on the effects desired.

2.3 Events and Looping

2.3.1 Key-on and Key-off Events

The initial important event in the life of a sound is the "key-on" signal from the keyboard. Upon its receipt, the pattern is "prepared" and sound begins. The next important event which can occur is "key-off" i.e. the key is released. On a piano, this event causes the string dampers to be re-engaged, thereby beginning the decay phase of the tone. If we re-examine Figure 1-3's sound pattern, reproduced below,



what we would like to model is the fact that no matter where in the pattern (i.e. when) key-off occurs, we are guaranteed that the decay phase will immediately begin. This is accomplished by using the "pointer to another block in the list" [see 2.1.2, 2.1.3] found in each tone- and envlist block. Thus, we can model this damping phenomenon by causing this pointer, in each attack and steady-state block, to point to the corresponding decay block. In order to make use of this pointer, finally, we define the "key-off event procedure" as being the one which causes control of the pattern to shift to the block defined in the "another block" field. The completed pattern is shown in figure 2-1.

The pattern of Figure 2-1 illustrates the use of a block-to-block sequencing procedure which is activated by a key-off signal. It should also be noted that the "key-off" sequencing procedure is activated only by a key-off event. There are two other events which each activate their corresponding sequencing procedure, in a manner exactly analogous to that of key-off: these are the End-of-Points and Timer events.

2.3.2 The End-of Points (EOP) Event

The phrase "end-of-points" refers to the exhaustion of the data points which comprise the envelope or waveform. In the case of the envelope, after the last point in the envelope has been consumed and the pattern needs another one, the EOP sequencing procedure in the current block is activated, with the result that control of the pattern passes to a new block (either NEXT orJUMP), with a new envelope. The "new" block may be the "old" one, but is considered "new" by the pattern and begun all over again.

In the case of a waveform, which is repeated many times to make the sound, "end-of-points" refers to the consumption of the stated duration of that block i.e. point b in 2.1.2's list. As with envelopes, when EOP occurs, the EOP sequencing procedure is activated, and depending upon which sequencing procedure has been specified, control of the pattern passes to either the NEXT or the JUMP block.

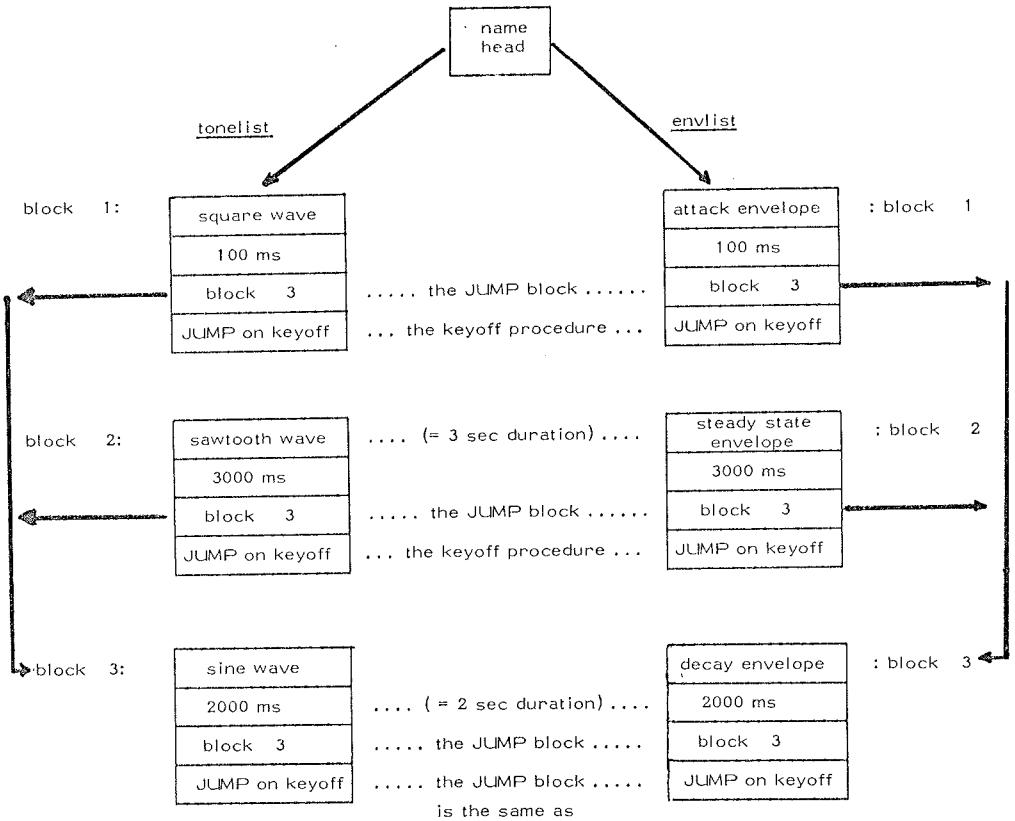


Figure 2-1

A sound pattern which jumps to its decay phase
on the occurrence of a keyoff signal from the keyboard.

Figure 2-2 illustrates how the EOP sequencing procedure can be used to create what is called a "loop", which means that the same sequence of blocks is repeated over and over again i.e. "looped through". The example illustrates a vibrato in both amplitude and frequency. It should also be pointed out whenever the "target" block of a sequencing procedure is either zero or nonexistent, the pattern is terminated and the sound stopped; this fact is used by the key-off specifications in Figure 2-2 to cause the tone to stop.

2.3.3 The Timer Event

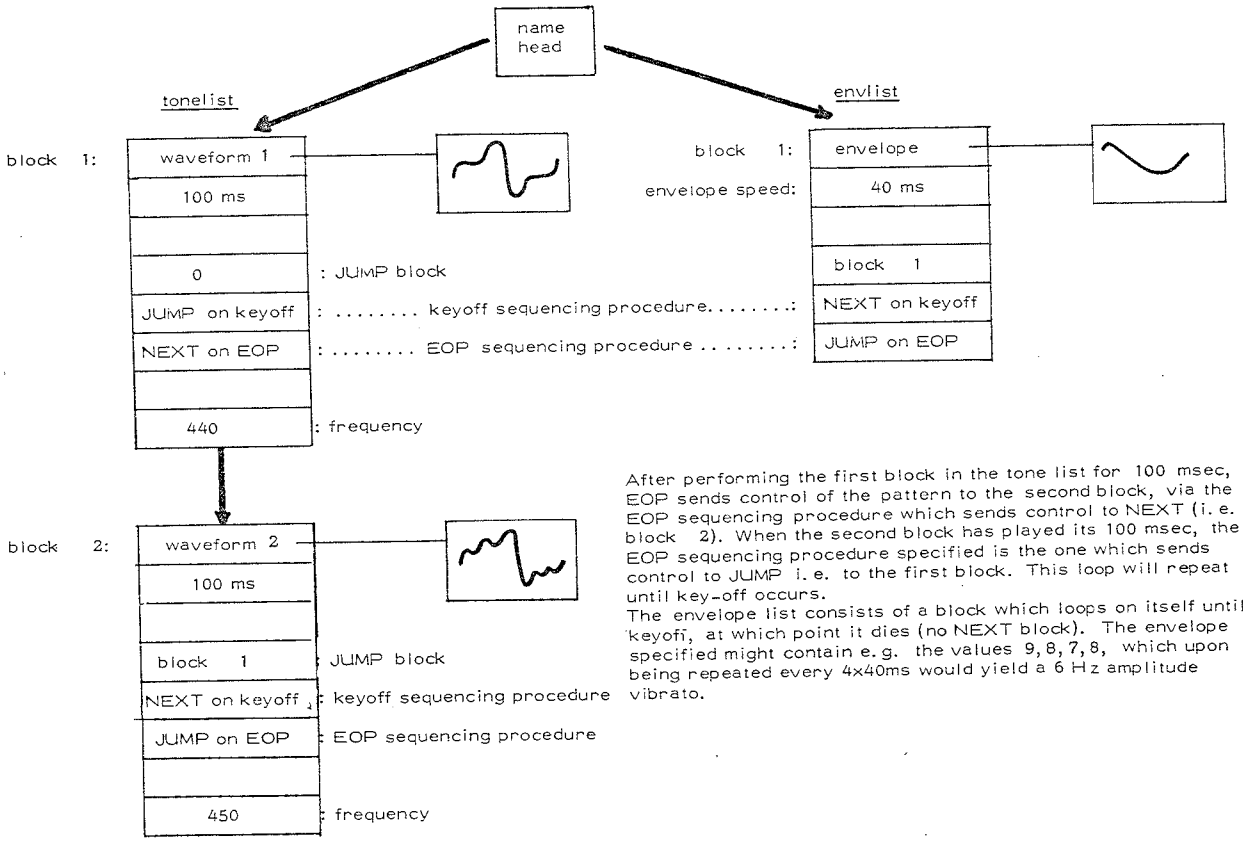
It is possible that one would like to define a pattern like that of Figure 2-2, but which, besides stopping on key-off, would also stop after a certain amount of time has passed.

In general, the EOP and key-off events are sufficient to specify the desired sound, but occasionally some other means of causing things to happen is required. Each block can therefore contain a time interval at which the effect can be caused (e.g. stop, frequency vibrato or glissando).

Figure 2-3 is a copy of Figure 2-2 with a Timer of 2 seconds i.e. the pattern will terminate either on key-off or when 2 seconds have elapsed, whichever occurs first. It should be noted that this is the first pattern we have seen which is complete i.e. which could be defined and used on the Egg. All the other patterns we have seen have lacked the specification of one or another of the necessary fields in the blocks.

2.4 Dynamic Modification of the Pattern

Up until this point, we have discussed sound patterns which, with the exception of frequency, are completely specified in the defined pattern. As examples we can point to the duration of the tone, the values of the Timer, and the speed of the envelope's application on attack. It is possible, however, to modify these specifications as well, and such modifications happen when a key-on event causes the sound pattern to be activated.



After performing the first block in the tone list for 100 msec, EOP sends control of the pattern to the second block, via the EOP sequencing procedure which sends control to NEXT (i. e. block 2). When the second block has played its 100 msec, the EOP sequencing procedure specified is the one which sends control to JUMP i. e. to the first block. This loop will repeat until key-off occurs.
The envelope list consists of a block which loops on itself until 'keyoff', at which point it dies (no NEXT block). The envelope specified might contain e. g. the values 9, 8, 7, 8, which upon being repeated every 4x40ms would yield a 6 Hz amplitude vibrato.

Figure 2-2

A Pattern with Frequency Alternation and Amplitude Vibrato

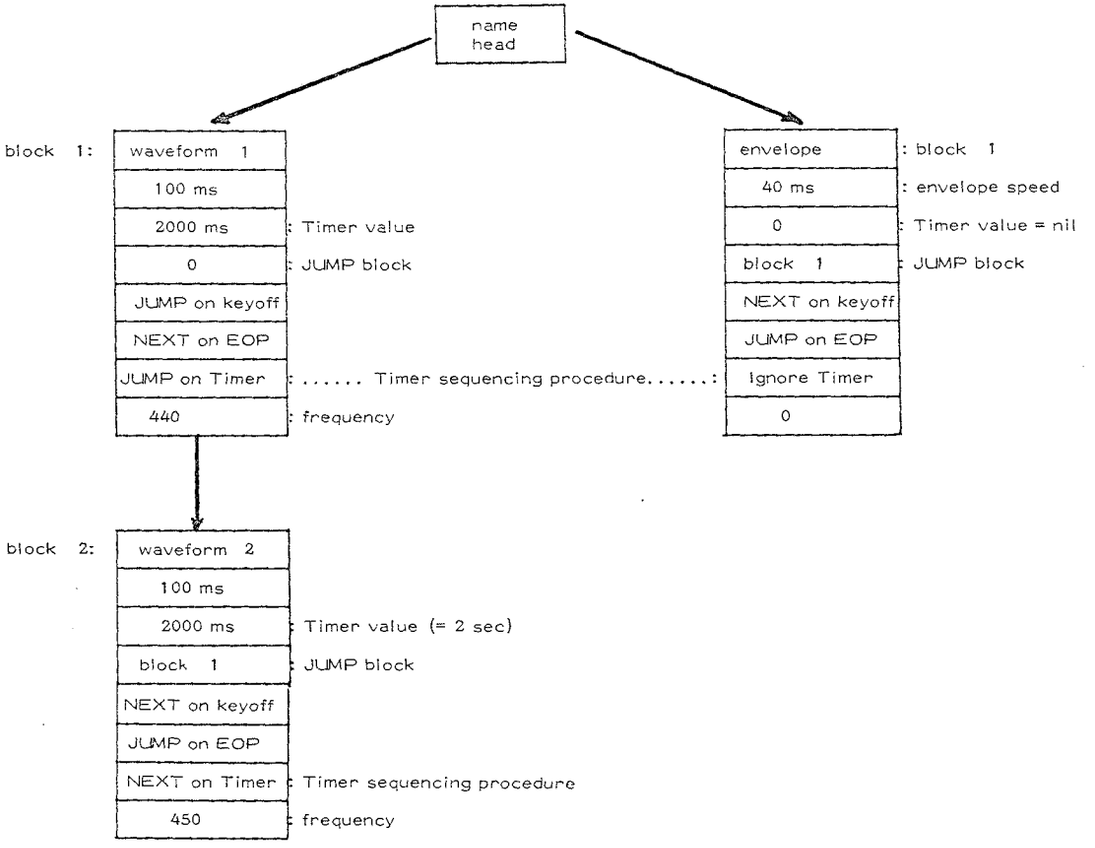


Figure 2-3

The Pattern of Figure 2-2 with Time-out Using the Timer

There are basically two parameters available by which to accomplish such dynamic modification: the velocity and acceleration (rate of change of velocity) of the key. These parameters can be used to affect the duration of the tone, the envelope speed, or Timer, to name the most obvious possibilities. As an example, the velocity can be used to control envelope speed (attack rate) and acceleration to control duration of the entire tone. These controls are specified by preparation procedures as mentioned in [2.2].

It is also possible to include sequencing procedures which modify the pattern while it is producing sound, an example being (frequency) glissando. Such an effect is specified by specifying the appropriate sequencing procedure.

2.5 Gemisches

A gemisch in the Egg is a collection of ordinary sound patterns and/or other gemisches which can be activated by a single keystroke. However, before saying more about gemisches themselves, it is necessary to motivate their existence as a necessity, and not just a fancy facility.

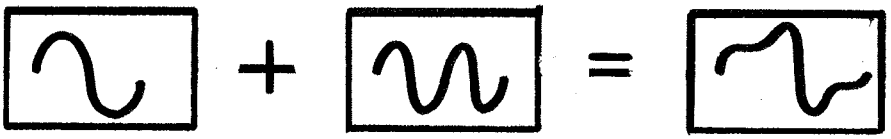
We begin the discussion by defining two types of overtones:

> a harmonic overtone is a frequency which is an integer (i.e. 2, 3, 4. . . .) multiple of the fundamental frequency of the tone;
>
> an inharmonic overtone is a frequency which is a non-integer (e.g. 1.3, 2.1 etc.) multiple of the fundamental frequency of the tone.

As an example, if the fundamental frequency of a tone is 1000 Hz (cycles per second), then harmonic overtones would be 2000 Hz, 3000 Hz, 4000 Hz etc. Examples of inharmonic overtones would be 1001 Hz, 2500 Hz, etc. The most dominant overtone is usually the octave, which corresponds to the frequency 2 x F, where F is the fundamental frequency; 3 x F is the fifth above the octave; 4 x F is two octaves above F. One can clearly hear the effect of the overtones on a pipe organ by selecting one stop and then, while holding a key down, successively adding more stops. It is the overtones of a sound, along with the attack, which give

it its unique characteristics.

Harmonic overtones can be built by combining several waveforms into one, for example:



The resulting tone now contains both the fundamental and its octave (first overtone). Other harmonic overtones can be formed in a similar fashion, and all such waveforms can be used in ordinary sound patterns [the generate command mentionel in [5.0] is used for this purpose].

Inharmonic overtones present a more difficult situation because a multiple such as 2.1 will not "fit into the box" in the above diagram as does the multiplier 2. This is a way of saving that one cannot construct a single Egg waveform which contains several partial tones having a non-harmonic relationship to each other. Thus the only means by which inharmonic overtones can be realized is to activate an independent sound pattern whose frequency is nonharmonic with the fundamental's pattern's frequency. This approach means that in order to define sounds which contain inharmonic overtones and can be activated by a single key-stroke, multiple sound patterns must be associated with each key. Such a multiple-association is called a "gemisch".

Since gemisches allow sound patterns which have been defined "stand alone" to be used as (harmonic or non-harmonic) overtones (and also undertones), some problems arise, because an ordinary sound pattern expects to receive its (missing) velocity, acceleration and keynumber (for defining frequency) from the keyboard. If we therefore wish to activate some "extra" sound patterns, we must supply them with synthetic versions of these values. These synthetic values in most cases will not be same as the " originals" from the keyboard, since this would result in identical attacks (and worse yet) fundamental frequencies for all the partial tones.

Therefore, when one defines a gemisch, one first specifies which sound patterns are to be activated, and then for each of these, the following

1. A ratio to be multiplied by the frequency which is associated with the key. This new frequency thus defines the "fundamental" frequency of the over - or undertone.
2. A + or - value to be added to the original key's velocity. This causes the attack (or whatever velocity is being used for in the pattern) to be modified according to the pattern's subsidiary role in the total sound.
3. A + or - value to be added to the original key's acceleration. This is philosophically the same as (2) above.
4. The number of milliseconds of delay after the key-on and key-off events before this pattern is to be affected. This allows overtones to be "out-of-phase" with respect to the fundamental.

Note that no one of the sound patterns in a gemisch has been singled out as the "ground tone" or "most important". It is up to the user to decide if he wishes one of the patterns to dominate, which is done by specifying 1/1 for item one, and zero for items 2, 3, 4 above. A sound which is an equal blending of several patterns would either have such values for all the items, or in some other way adjust 1, 2, 3, 4 above to achieve the desired characteristics.

3.0 Keyboard Registration

This section discusses the manner in which particular frequencies and sound patterns are associated with each key on the keyboards.

3.1. Scales

A scale is a list of frequencies having a particular relationship to each other.

Traditionally, these relationships have been

(1) increasing pitch coupled with

(2) a formula for calculating the next member of the scale.

In addition, there is the more subtle relationship of

(3) each successive element lying physically to the right of its predecessor on the keyboard.

There is no reason to believe that all users of the Egg would be willing to be bound by these strictures, and given the possibility of scales with 22 or 43 or 72 tones per octave, there are good performance reasons as well for a more flexible approach.

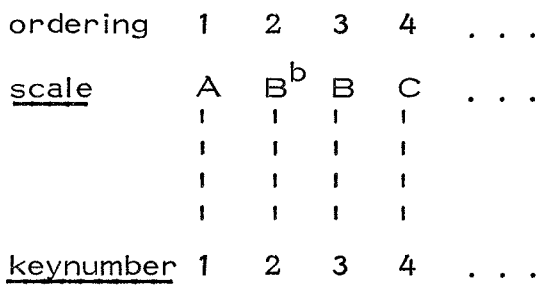
Therefore, when a user defines a scale, it is defined merely as a list of frequencies, this list having as yet no physical relationship with the keys on the keyboard. The only inherent structure of such a list of frequencies is that it is ordered i.e. the first frequency on the list, the second, the third, etc. As to whether "first" means "leftmost" and "last" means "rightmost" or something entirely different is determined by the concept of a "keyboard section", which we now explain.

3.2 Keyboard Sections

The keys on the keyboards are numbered from left to right, with the leftmost key of the upper keyboard being viewed as the successor of the rightmost key on the lower keyboard. Thus, each key has a unique number which differentiates it from all the others; this number is called the physical key number.

A Keyboard Section is a list of physical key numbers. The only inherent structure of such a list of key numbers is that it is ordered i.e. the first key in the list, the second, the third, etc.

It is now possible to see how to construct a "keyboard" which resembles that of a piano. We define a well-tempered diatonic scale, i.e. a list of frequencies, with the lowest first and the highest last. We define a keyboard section listing the leftmost key on the lower keyboard first through to the rightmost key on the lower keyboard last. The only relationship between these two lists is the ordering of their respective elements i.e.



The fact that the keynumbers are identical to the ordering is a coindidence due to the fact that the physical layout of the lower keyboard corresponds directly to both the scale and the numbering - and this is not generally true.

3.3 Sound Distributions

Up to this point, we have seen how a keyboard section and a scale can be mapped together to allow any scale to be located anywhere on the keyboard. A similar strategy is used to associate sounds (i.e. sound patterns or gemisches) with a key.

A sound distribution is a list of sound patterns or gemisches. As with scales and keyboard sections, the only inherent structure of such a list is that it is ordered i.e. the first sound on the list, the second, the third, etc. As with scales and sections, it is precisely this ordering which allows them to be associated with each other to achieve the final goal: the ability to associate any key with any frequency with any sound. Notice that this has been accomplished while still allowing one to deal with useful grouping concepts such as a "scale". Furthermore, these groupings are independent of each other, and therefore need only be defined once e.g. a

well-tempered diatonic scale is defined once, and may henceforth be used with any keyboard section and/or any sound distribution.

3.4 Action Buttons

We have thus far seen how a keyboard section, a scale, and a sound distribution can together specify an arbitrary partitioning of the keyboards. This discussion however said nothing about how one causes said partitioning to actually happen, and we now take this up.

The Egg is equipped with a numbered array of buttons which we call action buttons. The user of the synthesizer can define any number of abstract (named) buttons, and cause these abstract buttons to be connected to the physical buttons. There are no restrictions on this connection i.e. an action button may have several abstract buttons attached to it to achieve multiple simultaneous activation of the abstract buttons, and/or the same abstract button may be connected to several physical buttons. The reason for introducing the concept of abstract buttons instead of connecting scales, keyboard sections, and sound distributions directly is that it is entirely likely that a composition will need more than the approximately 30 physical buttons (i.e. keyboard registrations) available and therefore it is necessary to allow a much larger number of button definitions, and then just adjust their association with a physical button.

In practice, therefore, the user of the Egg defines an abstract button to contain some collection of scales, sound distributions, and keyboard sections, and even other abstract buttons (either of the first two items may be omitted, but there must be at least one keyboard section to drive the mapping). The abstract button is then (by command to the computer) connected to the specified physical button. Pressing that physical button will cause the associated keyboard registration to takeplace.

4.0 THE PSEUDO-TAPE

The Pseudo-Tape is a model of a 63 track tape deck. As such, it can record and playback what has happened on the keyboards, and also has controls such as rewind, etc. It is called a pseudo tape because it is not really a tape recorder, but rather a program simulation of one which plays through the sound generator. Thus, what is recorded is not the actual sound, but rather the key-on and key-off actions which caused the sound. Because it is simulated, it has a number of characteristics which are different from a normal tape recorder:

1. The recording is exact i.e. the time, velocity, and acceleration of the struck key, and the time of its release (both accurate to $\pm$1.5 msec.).
2. Due to 1 above and the fact that there are of course no play/record heads, the recording cannot deteriorate with repeated playing, copying, editing, or overdubbing.
3. Each of the 63 tracks can be spaced forward or backward independently of the others. This is due to the fact that, unlike multi-track analog tape, the individual tracks (because they are only simulated) are not physically connected to each other.
4. Due to 3, insertions and deletions on one track have no effect on other tracks, except insofar as the edited track takes a longer or shorter time to play.
5. Either single or arbitrary combinations of tracks may be played back simultaneously, although of course the total number of physical tones is limited by the number of hardware voices in the sound generator.

Besides recording key-on and key-off events, action button definitions and actions are also recorded. The net effect is that an exact recording can be made of a composition, including registration changes.

The commands available for manipulating the tape include.

1. Copy track-a to track-b;

2. Move track forward/backward n sec/msec;
3. Shrink/Expand track by timefactor;
4. Make track passive/active;
5. Write/erase audible/inaudible mark;
6. Wind tape forward/backward;
7. Single-step forward/backward;
8. Erase to mark/end-of-tape;
9. Play/stop;
10. Record/insert on track;
11. Align track-a with track-b (in time).

The pseudo-tape existed on an earlier version of the Egg, but due to other pressures, its interface to the rest of the system has not been updated. While it is therefore currently unavailable, its usefulness was undeniable and we look forward to its reincarnation.

5.0 The Command Language

We have deseribed in the foregoing sections the hardware configuration, the basic idea underlying our scheme for generating and controlling sound, and the abstract structures the user can build (waveforms, envelopes, sound patterns, keyboard sections etc) to utilize the facilities at hand. In this section, we will briefly discuss the concrete means by which these structures are built and connected: the command language.

The command language consists of a number of imperative verbs which form the start of a dialogue between the musician and the Egg. This dialogue is carried out via the alpha-keyboard and screen. The latter's speed and silence is exploited by exhibiting generous amounts of explanatory text automatically. This excess verbiage is the normal mode of conversation and cannot be disabled - novice users are grateful and experienced users (victims of habituation) stop seeing it.

Table 5-1 summarizes the commands available and what they do. Figure 5-1 shows a sample dialoque, which takes an experienced user about 20 seconds to complete.

The construction of a pleasant interactive interface, which we feel the above is an example of, was an important aspect of the instrument's development. It was always our philosophy that no more understanding would be expected of our users than is contained in this article (though a more digestible form is available in [4]), and all other details must be readily apparent from the dialoque. In order to meet these requirements, a special parser for interactive conservations was designed and programmed [1], and with this tool, conversations like the example are trivial to express. We are thus able to supply our users with dialogues which are much more supportive than the usual computer user
The point we wish to make is not that this parser or dialogue are any kind of acme, but rather that these techniques are standard tools in computer science, and that it is neither necessary nor humane to supply the user with a less congenial environment.

CREATE -	used to create all the aforementioned entities - WAVEFORMs, ENVELOPEs, BUTTONs, etc.
RESTORE -	used to restore WAVEFORMs, ENVELOPEs, etc. to the Egg's memory from the library.
STATUS -	writes the status of the objects in the Egg's memory on the terminal.
SAVE -	saves the given objects for entry in the library.
INSERT -	used to insert additional (i.e. new) blocks in sound patterns and gemisches (only).
MODIFY -	used to modify existing blocks in sound patterns and gemisches (only).
CONNECT -	used to connect a (created) BUTTON to one of the physical buttons.
KILL -	Kills all sound currently playing and reinitializes the keyboard.
GENERATE -	generate a waveform with a given overtone spectrum for eventual inclusion in a sound pattern.

Table 5-1 User Commands

6.0 Conclusions

The preceding pages have described the Egg and how it functions. We would now like to say a few words about how it is to use.

From the point of view of a new user, learning to use the instrument constitutes two phases - defining sounds (sound patterns and gemisches) and defining registrations. One "guided" session of several hours on each topic appears to be sufficient to equip the new user to use the instrument alone. Nevertheless, the effect of these sessions is some what overwhelming for most, but with hindsight the experience can be compared to encountering a large analog synthesizer for the first time. Presumably when our library of sounds and registrations is more complete, it will contribute to a "softer landing" on this point. In any event, both composers and students have used the system successfully.

One somewhat unexpected dividend of the Egg's emphasis on performance and ease of use has been that hitherto abstruse topics such as tuning and temperament can be presented to students very successfully. By actually playing in various tunings (and perhaps hearing a pure trichord for the first time in their lives), students can attain an intuitive grasp of the subject that no amount of theory can give.

The ability to generate arbitrarily complex waveforms (with a static overtone spectrum) and then combine these in independently developing sounds using gemisches has meant that it has been very easy to create lush and dynamically varying tones, and such niceties as organ pipe "pift" are trivial to arrange. The general amplitude and timing facilities available for enveloping, together with looping, allow a wide variety of complex rythmic passages to be performed.

From the point of view of the facilities available, it does not appear that there exist any major superfluities or omissions. We have however noted that especially in the case of envelopes, it seems more natural to say "from ppp to mf in N milliseconds" than to specify to exact series of points and their Δ- time. We are therefore considering changing the Command Language so that such "vector thinking" can be accommodated (without changing the underlying software structure). The software

is forgiving of attempts to play using incompletely specified objects e. g. sound patterns, and is in general so stable that one can concentrate totally on the musical aspects of one's work.

Reference [2] contains some details on the pseudo-tape which do not appear in the present paper, but is essentially otherwise identical. Reference [3] describes the software design.

7.0 ACKNOWLEDGEMENTS

The original idea and early development for the Egg was done by the author beginning in the spring of 1970, and its subsequent development proceeded with the support of Finn Egeland Hansen of the Institute of Musicology in Aarhus and the Danish State Research Council. This subsequent development saw the refinement of the original ideas into a concrete specification, which was then implemented on a Texas Instruments 960 A computer by the author with the help of Finn Haahr Kristiansen and Peter Houmann; the necessary special purpose hardware was designed and built under the direction of Kurt H. Andersen of the Datalogisk Institut in Aarhus by Finn Søberg Sørensen. Special mention should also be made of the supporting contributions of Ole Bromose Møller, Erik Bak Kristensen, Thorkild Laursen, Jens Otto Michelsen, Erik Thomsen and Walter Rasmussen.

The Egg achieved its first squawks in the fall of 1973, became a usable tool in January of 1976, and a public debut concert is planned for the near future. This article is dedicated to all those without whose help it would never have existed.

Incidentally, the instrument is called the Egg because of our hopes for it in the future of music, and equally hopefully has nothing to do with the laying of same.

References

1. Manthey, M. J. : "A Technique for Implementing Interactive Conversations", CS Report PB-57, 1976.

2. Manthey, M. J.: "A Non-Technical Description of the Egg Real-Time Sound Synthesizer", CS Report PB-56, 1976. Also in Electronic Music and Musical Acoustics No. 1, 1975. See [5].

3. Manthey, M. J. & F. E. Hansen: "The Design of the Egg Real-Time Sound Synthesizer", CS Report PB-72, 1977.

4. Manthey, M. J.: "Egg Users Manual", CS Report MD-27 1977.

5. Andersen, K. H.: "The Design of the Egg Synthesizer's Key-board", Electronic Music and Musical Acoustics No. 2, 1976. Dept. of Musical Acoustics, Institute of Musicology, Universitetsparken, University of Århus DK.

6. Andersen, K. H. : "The Design of the Egg Synthesizer's Sound Generator", Electronic Music and Musical Acoustics No. 2, 1976. See [5].