This presentation will outline the purposes of the Phthong system, overview the basic command set, and go into some detail about the way in which the commands are being implemented such that they achieve the system's purposes. A detailed user's guide for Phthong is being prepared.

Phthong is a set of general commands and special-purpose routines/processors being developed. It is intended to assist composers in the area of sound selection and organization. Ultimately, the software system will be useful not only to the composers, but sound editors, as well. It will have the capacity to assist in designing and testing a design of the in-time arrangement of virtually any set of sounds for which the user has formulated, in some symbolic system(s), a description of their properties or attributes. These descriptive elements could be derived from or directed towards any number of domains - acoustic, psychoacoustic, emotional - from the quantitative to the sheerly qualitative. The particular property category (variable) and the terminology used to describe each (i.e., the set of possible values) should be completely open to the composer's needs; the constraints on the details being determined by the ways in which the goals, logic and strategies of the composition/arrangement relate to the sounds being organized.

The task of designing a system with this kind of flexibility is not a minor one. One level of the design must enable the user to bring to it a purely
arbitrary set of descriptive terminology which has resulted from an aural analysis of the sound material and be capable of facilitating the composer in exploring some hypotheses about how the sounds in the set might be most effectively arranged. The efficacy of the system is determined by how readily (i.e., economically) the composer/arranger can organize the sounds through the use of Phthong by referring to and creating their orderings using his own terminology. This brings up a second level of difficulty to the design aspect of the system which concerns the connection between the input representation (in this prototype version, a stream of commands entered on a terminal) and the output representation (i.e., the sounds themselves, or as might be necessary in the case of orchestral output, the score). Currently Phthong handles only text input, however interfaces are being built to permit graphic and/or menu-driven input, as well.

Currently the system is not tied to any programmable digital sound synthesis mechanism. It was inspired primarily by dealings with electroacoustic composition in which one rambles off to the studio with numerous reels of tape under the arms and is presented with the problem of remembering just what sounds are on each tape, their location on the tape (by band or time value), and what some of each sounds distinctive features are (in the composer's own words, notations, and scribbles). Anyone who has attempted to organize a set of sounds into a composition (be it either an improvisatory or systematic type) has been confronted with a need to develop or mimic some criteria by which to analyze and describe the sound material. This is the part of composition which involves our listening-to and "grasping" of the sound and the resultant decisions as to what sound(s) go where and why. The criteria we form is a mental mapping of the arbitrary sound into our compositional goals and aesthetic. The result of this process is an (internal) representation of the sounds which the composer can reference (more or less) during the process.
of sound composition or arrangement. This part of composition involves the tuning of the sound, the composer's representation of the sound and his compositional logic (and preferences). It is a highly dynamic problem-solving adventure in which we come up with a relatively unique system of description and reference which guides or triggers the placement or recall (i.e., selection) of the sounds according to our compositional goals.

In our attempts to grasp and mold the sound, we often look for and develop appropriate formal representations that allow us to externally express and describe the often less formal compositional thoughts -- the latter being considered the composer's specific internal representation, the former, an external representation of which there are many possible variations.

Of course, the finished sound-work is the ultimate external representation, however, some form of graphics or notation of the thought and a set of appropriate terminology generally appears on the scene at some point during the compositional process.

Phthong is constructed around the notion that it would be handy to have a programming language that is adaptable to various and dynamic internal and external descriptive representations. Such a system would facilitate a composer in organizing the sound material on hand (i.e., on tape, disk, or in some re-synthesizable form) via commands whose parameters are elements of the composer's chosen or original set of descriptive terminology. It could serve as a "chalkboard", on which the composer can sketch out a system of criteria and goals of the composition in his own terms, and then serve as the mechanism to execute a design and, ultimately, to hear the sounds arranged and rearranged according to these criteria, goals and hypotheses. Hence, Phthong is intended to be a sound arranging as opposed to a sound synthesis language.
Because the system is not yet tied to an automated sound synthesis device for the direct playback of prestored sounds its output has been strictly graphic/notation/text. "Scores" are now being created which are of use for instrumental or for electro-acoustic composition.

It is disappointing that the goal of linking Phthong to a sound re-synthesis/editing system has not yet been realized and only scores have been produced. Yet, the solution to the general problem posed for Phthong has been found; that which facilitates the translation of (source) compositional-descriptive representations into (target) musical representations.

Phthong is an experiment in developing a context or environment in which to develop paradigms for composition. This environment is expressed by the command set that is being designed. The goal is to enable a composer to use his own source descriptions for the sounds in a set as parameters to the commands provided. The use of these commands results in the generation of data files which, in turn, are translated into some other target statements. The latter statements are generally somewhat more prescriptive than descriptive in that they can be commands belonging to other languages (i.e., graphics or sound synthesis) which, when executed by the appropriate language package(s), result in a target (musical) representation. Used as such, Phthong is a code generator for other compilers/interpreters used especially to generate sound or graphics. The target statements generated, of course, depend not only on the compositional application and strategies, but on the target language selected for producing the target representation. If the composition is for traditional instruments or electro-acoustic sounds stored on analog tape, the target representation might very well be a score, time-sheet or script requiring a translation from the source descriptions in-
to statements sent to a particular graphics or text formatting package. Future implementations will enable the generation of target statements for sound synthesis or editing languages resulting in direct sound output. Figure 1 shows a diagram outlining the general processes for using Phthong.

An Introduction to Using Phthong

Phthong deals with the specification and organization of variable-length blocks of time. These blocks are referred to as "frames" and have certain similarities to measures in traditional music. Besides being a unit of time, a frame like a measure, has certain features which are "global", such as its duration and the collections of instruments or sounds which will participate within its boundaries. Other features pertain more to "local" aspects such as the dynamics for a particular instrument, part or sound.

In the process of using Phthong, one will see that a frame actually takes form through the use of certain commands which generate individual files (i.e., frame components). These files are loosely bound together forming a frame. Frames and their components are identified by a file naming convention which can take a default, or be changed by the composer and made known to the commands via one of several mechanisms. The details of file naming conventions, alterations, etc. is described in the Phthong Logic Manual presently being compiled.

The specific content of these files (i.e., frame components) depends greatly on the Phthong application being implemented, but generally they contain a tag which can be used by other commands to assist in the identification of the file's contents, followed by actual data which pertains to the

574
Figure 1 Generalized Process Flow for Phthong
particular compositional element being specified. The element being specified is also completely open to the application and may be something fairly objective like amplitude (e.g., in dB) or, given the whim of the developer, something relatively subjective like the emotive or figurative quality of the sound(s) being selected from a user-defined sound description data base (e.g., "HUGE_BLACK_CRICKET_SOUND").

These files, or frame components, are created and can later be manipulated or updated with other Phthong commands, which take as some of their parameters the composer's source terminology that was formulated in conjunction with the compositional goals and strategies. Particular forms of some commands require that a reference or property file, or some previously generated frames, be searched for a match to the command parameters. Set-type operations, (e.g., union, intersection, etc.) and inequalities, (e.g., greater-than, less-than, etc.) are recognized and executed by the commands, and may be used if they are appropriate to the application's source terminology. Other forms for some of the commands permit the immediate specification of data which in turn will be tagged and filed within the specified frame component data file. Another form of specifying what information will go into a frame component is by reference to a previously generated frame component of the same type (i.e., variable). In this case, a copy is performed with the new file being appropriately named. Other forms include generating the data for a frame component by specifying a function name which might, for example, generate a sequence of pitches or text for a particular part participating in a frame, or to be treated as the vocabulary for the frame from which any of the parts can select.

Of the basic command set, those which result in the "construction" of frames include FRAMEDEF, PARTDEF, and GENFRAMS. Other commands such as
LSFRAME display the data of an existing frame(s). Information, such as the number of the frame last constructed or the sounds which were selected for a particular frame can be displayed by the QFRAME command. Frames can be analyzed by scanning, searched, and grouped by specified attributes (i.e., composer-defined descriptions) or combinations of attributes using the AN FRAMES command. When composing for any pre-generated sound set (e.g., concrete sounds on tape) the initial (or source) data base is constructed by building a table associating the composition's sound set with these user-defined attributes or properties. Frames can be reordered, if desired, using the REORDER command. This is a partial list of the commands which are being implemented to deal with the source data base or generate and analyze source representation files.

Two commands being developed to perform translations into target statements are SCORFRAM and TREEFRAM, both of which are intended to be used for producing printed output of the source representation files (i.e., frames and their components): SCORFRAM is used in producing graphic scores while TREEFRAM generates tree diagrams. Ultimately, a member of this subset, PLAYFRAM will be implemented. Its function will be to translate the source representation files into the appropriate commands to drive a programmable sound (re-)synthesis or editing system.

The commands with which frames and frame components are constructed or modified might be likened to a painter's set of brushes: Two of the commands (FRAMEDEF and GENFRAMS) are analogous to broad brushes which might be best suited for applying the undercoating, as they are used to define the global aspects of a frame (or string of frames in the case of GENFRAMS). When some detail work needs to be done to fill in the local aspects of a frame, the PARTDEF command serves as a narrow-tipped brush.
Some of the other commands (ANFRAMES, LSFRAME, and QFRAME) are useful for shifting one's perspective on the work-in-progress; somewhat like a painter who occasionally stands back from the canvas as the painting is developing so that the results can be reassessed and a determination can be made as to what structure on the canvas requires modification or what strategy to adopt next for the (re)development of an area.

Many of the Phthong commands can be thought of as editing routines which can be used to create or generate, modify, search or locate, and move Phthong frames. At any time in the frame construction process, the ANFRAMES command can be used to search through the existing frames for specified properties and report the frame(s) (by number) which contain these properties. This is useful if one wants to check an intended progression (e.g., gradual decrease in specified amplitudes, densities, etc. across frames). Modification of existing frames, if desired, is achieved via the frame generation commands.

The ANFRAMES command is a good representative of the Phthong command set because it shows the general concept of the language. For example,

ANFRAMES DEN (DEC 5) 1 40

is interpreted as follows: Search through frames one through forty in groups of 5 and report only those groups in which a monotonic DECREASE in DENSITY is found.

This is a fairly complex use of ANFRAMES that might require look-up tables that rank the possible values (from the source terminology) assigned to the density variable. Simple forms of ANFRAMES do not require such tables.
For example,

\[ \text{ANFRAMES AMP(VERY-LOUD) X TIMB(LOW-RUMBLE)} \]

which will search all frames generated thus far, or the sound-to-source de-
scription data base, if applicable, and report any frame (by number) in which
the amplitude value is "very-loud" and the timbre is described as "low-rumble". In other words, it takes the intersection (denoted by the \( X \)
operation) of the two (or more) search arguments.

REORDER permits the user to "pick up" a whole set of files associated with
one frame number and move it to a new position in the sequence of frames.

Once the frame(s) for a piece are built (or rebuilt), a score or script may be
generated. The tree diagram shown in figure 2 resembles those produced by
the TREEFRAM routine. The example shows a sample from the first frame of the
piece generated for the prototype application. An IBM text processing pack-
age, Document Composition Facility (DCF), was the target language in this
case.

In general, the software selected as the target language is a major deter-
mining factor in the design of the translation programs. During the develop-
ment of the Phthong prototype application, which was done on an IBM 4341
running VM/CMS, the choices of graphics packages available were
constrictive. SAS/GRAPH was used, but was not particularly well suited to the
task. It was, however, the most flexible package on the system at the time.
Later implementations done on the IBM have used the Graphical Data Display
Manager (GDDM) which permits the user to create symbol sets (type faces,
scoring symbols, etc.) that are quite useful for designing graphic scores.
Figure 2 A tree diagram similar to that produced by the TREETRAM command when fed the parameter for frame number, here 1. Note that the elements shown pertain to the first frame for the prototype composition for 10 voices. The Global Frame Files diagrams were originally filled using the following commands:

FRAMEDEF 1 {DUR(IMMED 15)}
FRAMEDEF 1 {NUMETS(IMMED 3)}
FRAMEDEF 1 {TEXT(BYPROP<VO1})

A sample of the commands used to build the Local Frame Files for Event 1 are shown below:

PARTDEF 1 {VOICE((IMMED(SOFT))}
PARTDEF 3 {ENTEXT(IMMED(0 15))}
PARTDEF 1 {ENV(IMMED(pp > ff))}
PARTDEF 1 {DEN(IMMED(spar = den))}
The programming technique adopted for the task of translating the source to target data (presumably statements in the target software language) involved the design of a set of "format templates" which contain generic translations for each basic portion (e.g., the borders and syntax) of the "scored" (or diagrammed) frame. The templates used by the translation routines essentially draw the basic forms and outlines which make up a frame in its scored representation. Different score formats and templates can be developed (or reused) fairly easily for each new application. The translators essentially edit the format templates, filling in the detail data which they pick up from the global and local frame files.

Although the prototype application was aimed at being able to produce graphic representations, the technique is certainly applicable to translators which generate code for a suitable sound synthesis or editing system; the task at hand being to translate/transform an arbitrary descriptive data set into a set of prescriptions interpreted by a target language. Figure 3 gives a functional description of the format templates used to produce the prototype score. Figure 4 shows the result when the data for the first frame of the piece discussed, above (refer to the tree diagram in figure 2) was passed through the SCORFRAM translator and subsequently processed using SAS/GRAF and a Hewlett-Packard plotter. One of the final functions of the translation program (in this example SCORFRAM) is to merge the target statements produced for all (or specific) frames and pass these to the appropriate target language processor (here, SAS/GRAF).

Some Details on Phtong's Implementation

One of the major goals and requirements of this system is that the com-
Global Frame Variables

<table>
<thead>
<tr>
<th>Source Variable</th>
<th>Format Template Name</th>
<th>Description of Template Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DURATION</td>
<td>BORDUR</td>
<td>Draws borders - The coordinate data is computed based on the duration values in the global frame file for frame duration.</td>
</tr>
<tr>
<td>VOCABULARY</td>
<td>WRTVOCAB</td>
<td>Fills in the vocabulary found in the global frame file for frame vocabulary.</td>
</tr>
</tbody>
</table>

Local Frame Variables

<table>
<thead>
<tr>
<th>Variables</th>
<th>Format Template Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOICES</td>
<td>WRTVOICE</td>
<td>Writes the names of the voice parts found in the local frame files associated with the VOICES frame component using the voice to derive the position (line number) of placement. For example SDPI is placed on the seventh line of the scored frame.</td>
</tr>
<tr>
<td>ENVELOPE</td>
<td>DRAENVEL</td>
<td>Draws the envelope form deriving the coordinates to the &quot;plugged&quot; into the format template statements from data found in the local frame files associated with envelope and entry-exit files</td>
</tr>
</tbody>
</table>

Figure 3 SCONE CONTROL is a look-up table used by SCORFRAM which associates source variables and data found in the global and local frame files with the format templates coded in target language statements.
mands be flexible and reconfigurable -- that they preserve their general functions regardless of the input terminology (i.e., command parameters) used by the composer to describe the sounds which constitute the composition's sound set.

The first attempts at implementing the basic command set were abandoned were scrapped when it was realised that their original design simply did not support the goal of maximum flexibility and reconfigurability. This goal is actually a requirement if the system is to be adaptable to broad variations in source terminology, format, target output, and compositional strategies.

The alternate programming technique adopted is one which opts, at the command level, for a fairly fixed syntax and semantics as is appropriate to the function of each command. This design moves reference to the processor routines necessary to handle the particular descriptive system being used to the file(s) which either contain the description itself, or provide execution-time information to the command modules.

Any command which must operate on the descriptions looks inside the file and picks up the name of the routine(s) which specialize in its handling, and then invokes the named routines, passing them the arguments found on the command line. When control returns to the command module (i.e., calling module), it goes on to its next step for which it may be able to take a default or may have to look inside another file (referred to as an "info" or "agenda" file) to get further control-flow information. For lack of a better name, I have referred this technique "data-directed" programming since the calling sequence for the processors is directed by keywords which are actually embedded within the input data.
The appendix gives the FRAMEDEF command syntax, a BNF-style description, and
the "agenda file" as configured for a prototype Phthong application.

This technique seems worth mentioning here as it acts as a solution to the
genral problem of creating a system which works without depending on pre-set
data formats. It permits the developer to mold the system to the desired in-
put data structure (i.e., the source terminology). Of course, it requires a
bit more work in the initial formalization of an applications, input and out-
put data structures in that the developer must implant the names of the spe-
cial processing routines which will be necessary in handling its particular
formats. It also means that the specialists (usually very short routines)
may themselves have to be created for each application, but it preserves the
command level functions, and reconfiguration will tend to become easier as
more and more applications are developed since specialist routines have the
potential to be reused or modified within and across applications.
Appendix

Below is a syntactic and BNF description of the FRAMDEF command. The BNF and FRAMDEF AGENDA descriptions document the command as it was configured especially for the second prototype application which was a composition for 10 voices and various percussion instruments.

A similar set of descriptions exists for each of the Phthong commands e.g., PARTDEF, GPHNUME, ANFRAMS, etc.) and will appear in the Phthong manuals being prepared.

FRAMEDEF Command Syntax

FRAMEDEF frame-no (var-name(slct-spec(args)))
 conj(slct-spec(args))...

Where,

frame-no is the integer number representing the frame to be generated.
var-name is the variable name being described.
slct-spec is the "selection specifier" (i.e., the manner in which the arguments are to be treated when generating the frame.
args is/are the argument(s) to the FRAMDEF processor routines.
conj specifies a conjunction if other slct-spec,args pairs follow otherwise it is ignored.
Partial BNF Description of FRAMEDEF Command Syntax

[frame-no] ::= [integer]
[integer] ::= 0,1,2,3,...,99

[variable-name] ::= TEXT specifies frame text/vocabulary from which vocalists selects
PERSSEL specifies frame percussion
DURATION specifies frame duration
NUMEVENTS specifies number of events (i.e., instruments) in frame

[select-spec] ::= IMMED following argument(s) specifies the value (as is) to be placed in frame
BYPROP select elements for frame according to the properties as specified in the argument(s) and the input file (refer to FRAMEDEF AGENDA description, below)
BYREL select elements for frame according to the relationship between the elements as specified in the input file (refer to FRAMEDEF AGENDA)
BYPROC select elements for frame according to the procedure named (refer to FRAMEDEF AGENDA)
BYREF select elements for frame by copying the values of like-variable elements from a previously generated frame as specified by the argument(s)

[argument] ::= determined solely by the application, in general, they can be words and/or numbers which have well-defined meanings as interpreted by the selection specifier (i.e., select-spec) parameters.

FRAMEDEF AGENDA Description

<table>
<thead>
<tr>
<th>variable-name</th>
<th>select-spec</th>
<th>default-input-file</th>
<th>default-procedure</th>
<th>default-output-file</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEXT</td>
<td>BYPROP</td>
<td>TEXTPROP TABLE</td>
<td>FINDTEXT</td>
<td>Fframe-noTEXT</td>
</tr>
<tr>
<td>PERSSEL</td>
<td>BYPROP</td>
<td>PERCPROP TABLE</td>
<td>FINDPerc</td>
<td>Fframe-noPerc</td>
</tr>
<tr>
<td>DURATION</td>
<td>BYPROC</td>
<td>n/a</td>
<td>DURPROC</td>
<td>Fframe-noDUR</td>
</tr>
</tbody>
</table>