A NEW APPROACH TO COMPUTER ASSISTED INSTRUCTION IN MUSIC THEORY

Richard L. Hamilton, M.M., and Dan W. Scott, Ph.D.

North Texas State University, Denton, Texas, USA

Introduction:

Computer Assisted Instruction (CAI) offers an effective method for presenting drill and practice in music theory. North Texas State University (NTSU) has developed a system which uses a unique group of components. These are a versatile and economic computer sound generation system and a simple yet effective curriculum management and presentation program. The two are independent but combine to form an effective and efficient method of teaching basic ear-training skills.

Every musician needs to acquire the ability to distinguish and identify musical sounds. The study of music is unusual among academic subjects in that the basic skills cannot be taught solely from a book. The particular combination of computer hardware and software described here was developed to provide students at NTSU with a drill and practice facility which would help them to develop their abilities in a structured manner.

North Texas State University has one of the largest music schools in the United States, with an enrollment of over 1700 music students. Nine hundred undergraduates enroll in some part of the two-year core music theory program each semester. All undergraduate music majors are required to pass a barrier examination in sight-singing and ear-training.

The ear-training skills required of music students are best developed through an instructional program designed to fit each student's needs. Such a program must include at least four things: (1) sound--the student needs to hear musical sound combinations; (2) immediate feedback--he needs to know whether he has identified a sound correctly before he forgets that sound; (3) patience--it normally takes many repetitions before a student can accurately and consistently identify sounds; and (4) individualization--each student's weak points need to receive extra attention.

Individual personal instruction would undoubtedly be the ideal solution to this problem, but the economics of such an approach are prohibitive. The CAI program at NTSU provides all four of the above requirements economically.

Two separate but related efforts have been brought together to implement this music CAI system. The first was the design of a general score notation suitable for keyboard entry and its implementation on a microprocessor-based automatic music generation system. The Automatic Music System (AMUS) developed by Dan W. Scott is comprised of specialized hardware and microprocessor programs. In addition to its use in this CAI application, the AMUS can be used as an aid to composition, as a performing instrument or as a means for an amateur to explore and enjoy music. The second of these efforts was the development by Richard L. Hamilton of a special-purpose curriculum driver program for the host, or time-sharing, computer. This program is especially suited to drill and practice through the AMUS.[1]

Emphasis in this application is on drill and practice; the introduction of concepts is left to the instructor. This system provides a resource, musical drill and practice, which most students have not had access to--and does so without overburdening the instructor.

AMUS:

The Automatic Music System plays multiple-part harmonic music of varying amplitude with accurate timing and pitch.[2] Articulation, metronome, and other performance parameters are specified by the user. The user provides a text (a "score") written in ordinary typewriter characters by means of a simple terminal. This score is translated as it is read by a Motorola M6800 microprocessor computer system into pitch, amplitude, and timing control information for the electronic instrument, MUSOR, which is attached to the computer. The score may be typed in each time it is to be defined, it may be stored on a tape cassette, it may be stored or generated as text by a BASIC time-sharing (or other) remote computer facility (the "host computer"), or it may be text in a Computer-Aided-Instruction system which uses the host computer.

Development of AMUS involved (1) the specification of a new and extended scoring/performance notation; (2) the invention of a novel digital-to-analog waveform generator; and (3) the development of an elaborate computer program for score translation and for performance control.

AMUS differs from previous cheaper computer methods used for music generation in its quality of sound, including harmony and articulation, and in the complex software provided. It differs from previous more expensive implementations in the lesser quality of its sound with regard to timbre and amplitude control, and in the simpler score/performance notation, suited to the typewriter keyboard.

The MUSOR instrument contains 1 to 7 voices (5 for CAI use), with waveforms determined for each voice by a set of 16 fixed resistors. The articulation is under computer program control, as determined by the score. Timing is provided through computer program interrupts at the rate of 1760 per second. The MUSOR hardware is especially designed to be inexpensive, trading hardware expense for software complexity to reduce the material cost of the system. The expense has been increased, on the other hand, by a great emphasis on hardware reliability, the ideal being no maintenance.

Figure 1, on the next page, is a diagram of the hardware units which comprise the Automatic Music System, including the host computer.

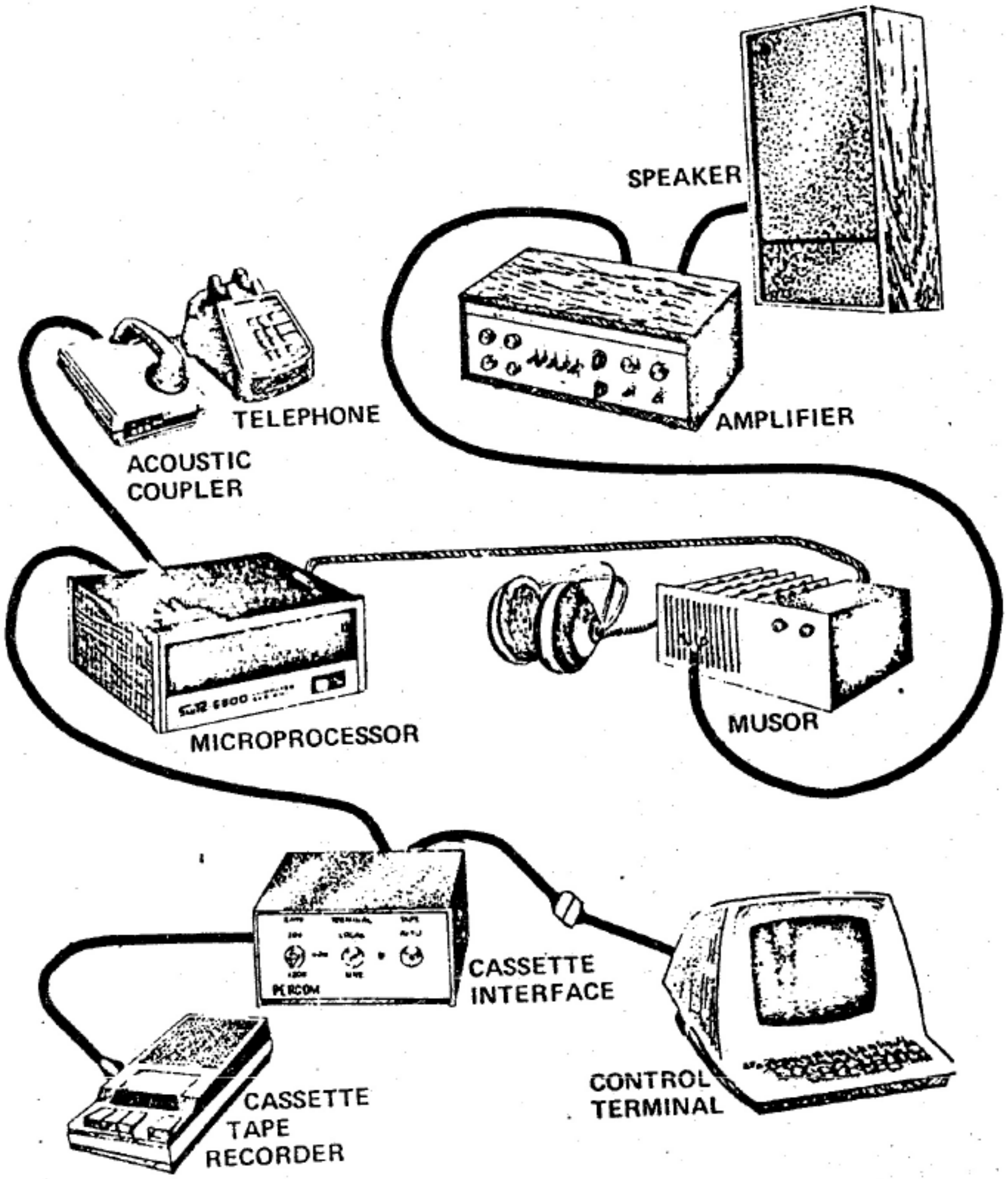


Figure 1. AMUS Hardware

AMUS receives scores and commands from either the local control terminal or from a remote time-sharing host computer. In order to handle a variety of score symbols as well as a wide range of text to be displayed to students, the AMUS program of the microprocessor has three modes of operation--text mode, score mode, and terminal mode. Certain commands change the mode of interpretation of input.

In the text mode, anything typed on the control terminal is sent to the remote computer and anything received from the remote computer is displayed on the control terminal. AMUS takes no other action until it receives a question mark (?) or a percent sign (%) from either source. Receipt of a question mark causes the score which was last read in and translated to be performed immediately; students may thereby cause a replay of the music. The percent sign signals AMUS that the next characters may be some kind of directive. If no valid text mode directive is found, the program takes no action and continues to read and display characters, remaining in the text mode.

If either of the directives %S or %SC is detected, AMUS deletes them from the display and moves into the score translating mode. In the score mode the input characters are translated into MUSOR control information for later performance. Score characters which are received from the remote host computer are normally not display to the student.

The score mode directives in general differ from the text mode directives. Table 1, below, gives the valid directives for the AMUS microprocessor in the text and score modes.

Table 1.

Directives--Text and Score Modes

%S	Begin score, go to score mode.
%SC	Continue previous score, go to score mode.
%P	Begin playing, stay in present mode.
%Pi	Begin playing at measure i, stay in the same mode.
?	Begin playing, go to text mode.

Directives--Text Mode

%CS	Copy translated score to cassette tape.
%LS	Load translated score from cassette tape.
%X	Go to terminal mode.
%[	Copy arbitrary text into microprocessor memory.
%]	Transmit text from memory to remote system.

Directives--Score Mode

%E	End score and go to text mode.
%Ii,j	Set attack time to i, steady state time to j units.
%Mi	Set tempo to i units. (%M3904 = mm. Q=72.)
%Kx	Set key signature to x.
%Ti	Transpose following score by i semitones (i = signed integer).
i%Oj	Set the voice i octave to j (middle c octave code is 4).
i%Lj, i%Nj, i%,j	Set articulation types. Voice = i; amplitude code of sustained portion of note = j.
%W	Start displaying score from host computer on the control terminal.
%W-1	Stop displaying score from host computer on the control terminal.
/	Measure marker used by %P.
/: :/	Beginning (/:) and end (:/) of section to be repeated (copied).
( )	Beginning ("(") and end (")") of comment to be ignored.

To leave the score mode a question mark can be typed, which starts playing the score and also gives control to the text mode. $E also transfers control to the text mode, but without playing the score.

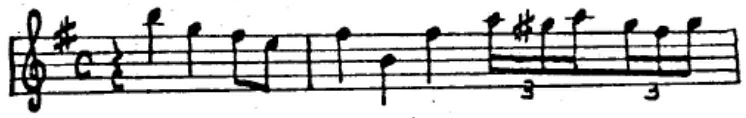
The terminal mode is entered from the text mode by means of the xX directive. In the terminal mode, the system reacts exactly as a usual computer terminal. All text, including the text and score mode directives, is transparently sent from the control terminal to the phone line and vice versa without processing. The terminal mode is exited by pushing a reset key to return to the text mode. The terminal mode is convenient for looking at a stored score without actually translating it.

In addition to these modes, special convenience features are provided for logging on with a remote host computer.

An example of AMUS score notation is shown in Figure 2, below.

Score Example

Allegro moderato Q = 144



```
xS  xK# (OR xKG)  xM2000 (Q=144)
RQ  BUQ  GQ  FE  EE  /  FQ  BDQ  FUQ  A3S  G#3S  A3S  G#3S  F3S  G#3S  /  (...
```

Figure 2.. Excerpt score from the Sinfonia of Handel's Messiah.

The usual format for notes is: voice number, pitch/rest designator, accidentals for pitch modification from the key signature, octave shift specifier, triplet specifier, and articulation specifier. Each of these component specifiers is optional or has a default value. Most of the score specifiers are described briefly in the paragraphs below.

1 2 3 4 5 6 7 *Voice designator.* Must be the first character of a note. If the voice designator is missing, the symbol 1 is understood by default.

A B C D E F G R *Pitch designator.* As the first character of a note (possibly preceded by voice designator), it represents the pitch in a scale or, if "R", a rest.

+ P *Pitch modifier.* Synonyms for one semitone sharp.

X *Pitch modifier.* Two semitones sharp (double sharp).

B - *Pitch modifier.* One semitone flat.

Z *Pitch modifier.* Natural.

Accidentals (as above) normally follow a pitch designator and override the key signature. When more than one is used in a note the effect is cumulative, starting with natural. Note that accidentals do not apply through the measure; they effect a change only in the note that they are attached to. The pitch modifier symbols are also used in specifying key signatures with xK.

U D *Octave shift, up or down, respectively.* Shifts the current and all following notes for the voice up or down one octave. If more than one specifier is used the effect is cumulative.

W H Q E S T 4 *Duration specifier.* These characters stand for whole, half, quarter, eighth, sixteenth, thirty-second, and sixty-fourth durations. They may be combined for additive effect.

. *Dot duration designation.* A dot (period) following a duration in a note adds half of the duration to the left of the dot to the duration of the note.

3 *Triplet designation.* The triplet specifier must precede the duration specifiers to which it applies. Its occurrence in a specification makes the duration specifiers which follow it have 2/3 of their indicated values.

M *Duration subtraction.* The value of the note duration specification which follows M in the note is subtracted from the duration which has preceded the occurrence of M in the note. M resets the dot and triplet designations, so that specifiers which follow M are normally duple.

L N , *Articulation specifiers.* Legato, not-legato (normal), and emphasis articulation, respectively.

Curriculum Driver:

In July 1977 Professor Rosemary Killam began the implementation of a CAI program in music theory at NTSU. At that time two packages were available on the Hewlett-Packard 2000 time-sharing computer used at NTSU for CAI. These were: the Instructional Dialogue Facility (IDF) and PILOT.

IDF is aimed toward the user who has no computer expertise. It functions in an interactive manner both with the student and with the instructor who is developing a curriculum. It prompts the instructor for texts, questions, answers, and responses. It provides the student with texts, questions, and hints. Student answers are evaluated according to the patterns supplied by the instructor, and a limited form of branching is permitted. IDF also stores student responses and statistics.

Entering a lesson into IDF is simple for the instructor, but any variation from the set pattern can cause problems, and the random presentation of questions to students can only be accomplished with great difficulty. While IDF is useful for many applications (it was used for the music theory course at NTSU before the new curriculum driver was ready), it is not well-suited for the music theory curriculum, especially when the requirements described in this paper's introduction are considered.

The alternative available on the Hewlett-Packard 2000, PILOT, has a number of desirable features for the instructor: more flexible formats and more compact means to enter instructional material. The instructor types in simple commands which are interpreted and executed when the lesson is presented to the student. The commands are easily learned and easy to use. The main drawback of PILOT for the evaluation and development of curriculum is that the Hewlett-Packard version does not keep records of student performance data. Also, there is no concise way to implement repetitive drills.

The NTSU package developed by Richard L. Hamilton provides for data collection. It uses a format similar to PILOT, but it includes additional features necessary for drill and practice: concise entry, random presentation order, and three methods to judge student success. The package is similar to IDF in that it includes an enrollment procedure and complete data collection capabilities. All student responses are recorded for analysis, and a condensed file of statistical information is maintained for immediate reference.[1]

Students are given a suggested progression of courses and lessons but they can easily change this order to fit their own needs. A weekly report is generated which allows teachers to monitor their students' progress. Teachers can easily add lessons for special purposes.

Commands in the entry language used by the instructor are provided to control the presentation of the lesson and to evaluate immediately the student's progress. The driver can repeat material depending upon this evaluation. Repetitive structures are simple for the instructor to use. Randomization of the sequence of questions enables repetition to be concealed from the student; in addition, provision is made for random transpositions of scores ($R). The extent of repetition is controlled by evaluation of student success as the lesson progresses. A summary of the commands of the curriculum driver is given below. Each line begins with a one- to three-digit number and ends with a return.

T: The lesson is to type out the text which follows the colon (:).

A: Wait for an answer (reply) from the student.

M: Match the student's latest reply with the text following the colon. Remove blanks and question marks (this allows the student to request repetitions of the music performance without being penalized).

J: Jump in the lesson to the statement whose number follows the colon.

JN: Jump only if the student's last response was incorrect.

JY: Jump only if the student's last response was correct.

TN: Type the text following the colon if the last response was incorrect.

TY: Type the text following the colon if the last response was correct.

P: Pause for the number of seconds given after the colon.

E: End of the lesson; causes termination of the lesson. This must be the last statement in the lesson.

Q: *n<p\a\c\t>* This is the format for questions. *n* specifies the number of questions which follow in the form *<question\correct answer>*. The instructor uses as many lines as are necessary for his convenience. The parameters of the Q command are summarized as follows:

n gives the number of questions in this group of questions.

p specifies the presentation method. R causes the questions of the group to be presented in random order, while D steps through the examples in the order given by the instructor.

a specifies the method of evaluation of the student's answer. D means that the student must match the answer provided by the instructor exactly; I means that no evaluation will be done.

c specifies the criterion for termination of presentation of the group of questions. A*i* means to continue presenting questions from the group until the student gets *i* questions right on his first answer to each question. P*j/k* means to present questions until the student gets *j* right on the first trial. If he has not gotten *j* right before seeing *k* questions from the group, then the counts are restarted for the presentation of more questions from the group, until the criteron is satisfied. After passing the criterion the student continues with whatever follows the group of questions. A third option, N, steps the student through the group of questions once.

t specifies the number of trials, 1 to 9, that a student gets to answer a question when it is presented each time. Note that this feature has no effect on the *c* = A or P criteria, which count only first-trial success.

RY: Right-answer reply for all questions of the group. A default reply exists, provided by the curriculum driver, which can be restored by using RY: with no text.

RN: Wrong-answer reply for all questions of the group. A default reply exists, provided by the curriculum driver, which can be restored by using RN: with no text.

F: Failure reply for the question group. The text which follows the colon specifies the reply for an incorrect answer on the last trial allowed for the question. A default reply exists, provided by the curriculum driver, which can be restored by using F: with no text.

H: Specifies the text for a hint. Any number of consecutive H: statements are saved and will be presented, as a group, each time the student types //HINT. If no text follows the colon, then all previous hints of the group are eliminated.

In addition, there are abbreviations which manipulate strings of characters. A dollar sign ($) in any output string (whether question, response, or other text) signals one of several operations, depending upon the character which follows the dollar sign. $N is replaced in the string by the student's first name, which is obtained at the first of the lesson when it is presented. $A is replaced by the correct answer to the most recent question. $R generates xS xT*i*, where the generator automatically replaces *i* with a random number from -7 to 7 in value. This causes the score which follows to reflect a transposition by a random number of semitones; the value of *i* is recorded with the student's response.

The criterion feature of the Q command allows the teacher to control quantitatively the conditions of presentation. This feature is particularly useful when the teacher needs to present a limited number of examples, perhaps 3 or 4, but wants to insure that the student can hear each example as many times as he needs for comprehension. For instance, if the criterion provided by the instructor specifies ten correct answers, the student cycles through the statements of the Q group until he achieves the necessary ten correct answers.

The random presentation order and random transposition options allow the patterns of question repetition to be concealed. Thus the teacher can specify a large amount of curriculum with a minimum of text entry. A text editor is provided with the curriculum driver which allows duplication of lines and permits easy editing within lines. These features expedite curriculum entry.

The Q command and the general economy of the system have greatly reduced file storage space as well as entry time. Lessons originally created in IDF which have subsequently been converted to the new system are now stored in approximately one-fifth the file space of the original and were entered in much less time. The sample lesson below shows the basic format and illustrates the use of the Q command and the format for questions and answers.

```
10  T:  HERE'S A LESSON TO HELP YOU LEARN INTERVALS IN MAJOR TRIADS.
20  T:  YOU'LL HEAR A TRIAD, THEN AN INTERVAL.
30  T:
40  T: YOU TYPE THE INTERVAL'S NOTES BY TRIAD MEMBER.
50  T:
60  T:      TYPE U (FOR UP) BETWEEN NUMBERS IF THE INTERVALS ASCENDS,
70  T:      TYPE D (FOR DOWN) BETWEEN NUMBERS IS THE INTERVAL DESCENDS.
80  T:
90. H: TRIAD MEMBERS ARE 1 OR 3 OR 5.
100 Q:  3<R\D\P4/5\3>
110  <$R 4CD 3GD 2C EH,  1R 2R 3R 4RQ  2CH,  EH,  ? \ 1U3 >
```

(continued...)

(Lesson example, continued from the preceding page.)

```
120 <$R  4CD 3GD 2C EH,  1R 2R 3R 4RQ  2EH,  CH ?   3D1>
130 <$R  4CD 3C  2E GH,  1R 2R 3R 4RQ  2CH,  GH ?   1U5>
9999 E:
```

Evaluation:

After a semester of operation with a single AMUS terminal, the results are encouraging. Subjectively, the student response to the use of CAI is favorable. Objectively, the first generation of the system was used in an experiment with two sections of the twelve freshman music theory classes at NTSU in the fall of 1977. These two sections had the lowest placement scores on a standard examination. Both were taught by the same instructor. Students in one class used the computer drill-and-practice facility while the others did not. On the mid-term examination the students who used the facility had a median score of 20 percentage points higher in ear-training skills than those who did not; at the end of the semester, a significantly larger proportion passed, and with higher grades, than the other group.[3]

Prospects:

Future plans include: display of a limited amount of music notation on control terminals, addition of question generation capabilities to the driver, an increased number of AMUS terminals available for CAI and experimentation, and modifications of MUSOR to widen the range of timbral possibilities and articulation. Computer identification of the pitches of notes hummed by the student is being considered. Longer-range ideas include the development of score entry via piano keyboard and the eventual production of a stand-alone version of the entire system using microprocessors to eliminate the time-sharing host computer. Such a system might offer four AMUS terminals, each processing the student interactions, with lessons and statistics kept on a fifth processor with disk memory.

The situation at NTSU also provides an opportunity to study a large population of musicians and to observe how they can best learn the special skills they must obtain to become proficient in their art.

REFERENCES

1. Most of the statistical routines are the work of Jack Chao of the North Texas State University Computer Sciences Department.

2. Scott, Dan W. "AMUS, The Automatic Music System". Presented at the meeting of the Fort Worth Chapter of the Association for Computing Machinery, September 13, 1977.

3. Professor Rosemary Killam of NTSU is preparing a more detailed study of the results of using CAI drill-and-practice in the music theory program at NTSU.

AMUS: THE COMPUTER IN MUSIC INSTRUCTION

Rosemary N. Killam
Assistant Professor, Music Theory

W. Kenton Bales
Teaching Fellow, Music Theory

Richard L. Hamilton
Teaching Fellow, Computer Sciences

Dan W. Scott
Chairman, Computer Sciences Department

North Texas State University
Denton, Texas

TEXAS MUSIC EDUCATORS' ASSOCIATION CONFERENCE

February 8, 1979