IMG/1
An Incidental Music Generator

Peter S. Langston

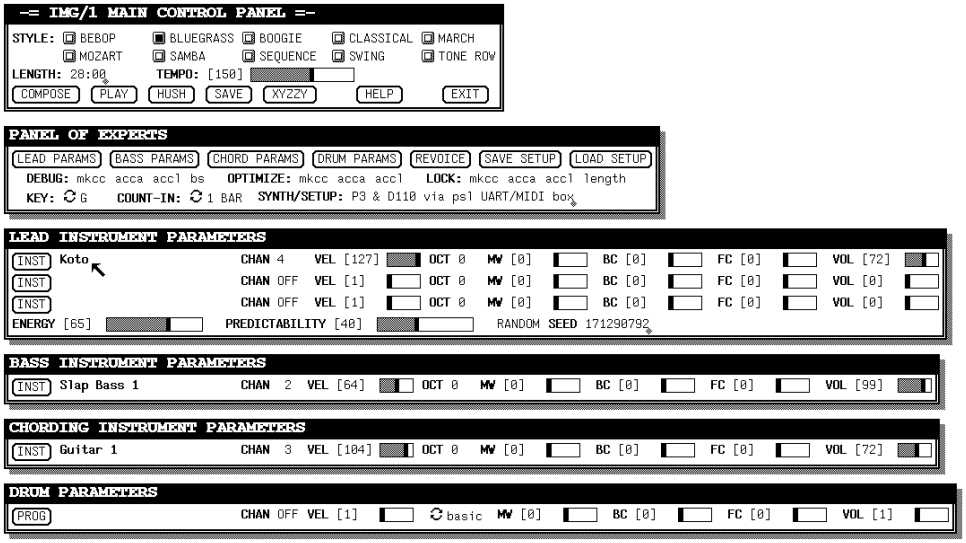
Bellcore
Morristown, New Jersey
April 6, 1990

ABSTRACT

Machines can compose music. Music composition (by machine) requires the solution of a number of difficult problems in the fields of algorithm design, data representation, human interface design, and software engineering in general. Although the automated generation of major symphonic pieces, programme music, or other musical forms that require complex semantic content has yet to be achieved, the automated generation of incidental or background music (e.g. as used in video production soundtracks) is within the realm of current technology.

This report describes ‘‘IMG/1’’, a system of programs designed to produce short, complete pieces of music with an explicitly specified duration and musical style. IMG/1 requires no musical expertise on the part of the user and can generate a virtually infinite number of different pieces in any given style. The user’s task is simply to define the duration and style parameters and then select an acceptable piece. Music generated by IMG/1 has been succesfully used as parts of several Bellcore video productions.

IMG/1 – An Incidental Music Generator



Introduction

Computer programs are available to aid musicians in the composition of music (e.g. ‘‘Music Mouse’’ (SPIEGE85), ‘‘Jam Factory’’, ‘‘M’’ (INTELL88), and others); but there are circumstances in which a person with little or no specific musical expertise (i.e. not a musician) needs to be able to produce simple musical works fitting a simple set of criteria. For example, a research chemist may want to assemble a videotape presentation showing how some new process works. There may be long sequences with no narration – while the solution changes color or the chart recorder runs. Or perhaps someone is compiling a series of interviews that share a common title/credit sequence. In these sequences some background or incidental music would be a perfect accompaniment and would lend continuity. The music required need not be very sophisticated or elaborate; it merely needs to have the right ‘‘style’’ and be the right length. However, unless the project has a very large budget, or the person making the video happens to be a musician with recording facilities available, the resulting sound track is likely to be silence or something pirated from a commercial recording.

Researchers have been exploring algorithmic composition for several years (HILLER70, HILLER81, BOLOGN83, DODGE85, LOY89) but as Kemal Ebcioglu says in a recent article on harmonizing chorales: ‘‘It seems that musical composition is a hard mental task that requires a substantial amount of knowledge, and any serious attempt to simulate ‘noncomputer’ music composition on the computer would have to face the task of constructing a formal model of considerable complexity. We have found that even the algorithmic representation of the knowledge underlying the seemingly simple Bach chorale style is a task that already borders the intractable.’’ (EBCIOG88)

Recent work at Bellcore (LANGST86, LANGST88, LANGST89a, LANGST90) has produced short pieces that, although devoid of semantic content, adequately embody the syntactic structures of music and result in acceptable music. ‘‘IMG/1’’ is a set of programs built on this work that extend it to address a range of musical styles. The central goal in IMG/1 is to allow musically naive users to produce pieces of background music of arbitrary length using algorithmic composition techniques.

Simplicity was an important design goal for IMG/1; the intended users, computer- and music- novices, must be able to compose and play an ‘‘original’’ piece of music with little or no training. As a result, IMG/1 is extremely easy to use; with a few clicks of the

mouse button it can compose and play an ‘‘original’’ piece of music. One mouse click selects a musical style, another mouse click sets the tempo, and typing a length (in seconds) determines the exact duration of the piece.

Most people are more sophisticated in listening than in creating music (‘‘I don’t know anything about making music, but I know what I like’’). Since IMG/1 can generate arbitrarily many variations, all different (and, thanks to the built-in musical knowledge, surprisingly pleasant), the user need only pick from among them. The dozens of parameters that provide detailed control of the composition process default to reasonable values if not set explicitly by the user, but may be manipulated as expertise with IMG/1 and its music increases.

Melodies are generated by a set of algorithms with expert knowledge of various musical styles. The techniques employed range from probabilistic traversal of graphs to simulation of the mechanics of a specific instrument. In all cases the melody generation uses some stochastic information (i.e. none of the algorithms are deterministic). Accompaniments are assembled by transforming a set of stock fragments to fit the progression of the overall harmonic structure. Although this scheme is deterministic, the generation of the harmonic structure is not; thus the accompaniment also changes for each piece (unless the user specifically asks to keep it fixed).

Music generated by IMG/1 has been used as background music in videotaped technical demonstrations, as the complete score for an MTV-like overview of a catalog of technical videotapes, and as a ‘‘live’’ demonstration at a product introduction by a major minicomputer manufacturer.

Structure

IMG/1’s software structure is diagrammed in Figure 1. IMG/1 makes heavy use of the Unix® operating system’s ‘‘pipe’’ facility to interconnect independent programs into a processing pipeline. In Figure 1 the rectangles represent separate processing steps (either individual programs or pipelines of programs) while the ovals represent data files.

The main entry is a program called ‘‘bellscore’’ whose function is to communicate with the user. The initial settings of parameters and the characteristics of the synthesizers attached to the system are read from a ‘‘setup’’ file at startup time. Bellscore displays the current parameter settings, allows the user to change parameters, and invokes the other modules as appropriate. The three primary parameters that bellscore controls are *STYLE*, *LENGTH*, and *TEMPO*. STYLE is a choice from a menu of known musical styles (e.g. bebop, bluegrass, etc.). LENGTH is the duration of the piece in seconds, video frames, or a combination of the two. TEMPO is the beat rate in beats per minute.

The program ‘‘mkcc’’ determines the harmonic structure of the piece. When the user selects the COMPOSE button, bellscore invokes mkcc to determine the length of the piece in bars; this may require adjusting the specified TEMPO parameter so that an even number of bars will take exactly the duration specified by the LENGTH parameter. Mkcc then determines the harmonic progressions in the piece based on the requirements of the selected style, and writes them out in an ascii form closely approximating the chord charts used by studio musicians. Having the harmonic structure expressed in a form easily read by humans not only facilitates debugging, but allows more sophisticated users to specify harmonic progressions directly.

Accompaniments are generated by the program ‘‘acca’’. Once mkcc has created the chord chart file, acca reads that file and produces an accompaniment based on it that is appropriate for the specified style. For some styles the appropriate accompaniment is silence, for others the accompaniment may vary from bass, piano chords, and drums, to a simple Alberti Bass line. Acca uses its ability to look ahead at future harmonic motion to

---

‘‘*Unix*’’ is a registered trademark of AT&T.

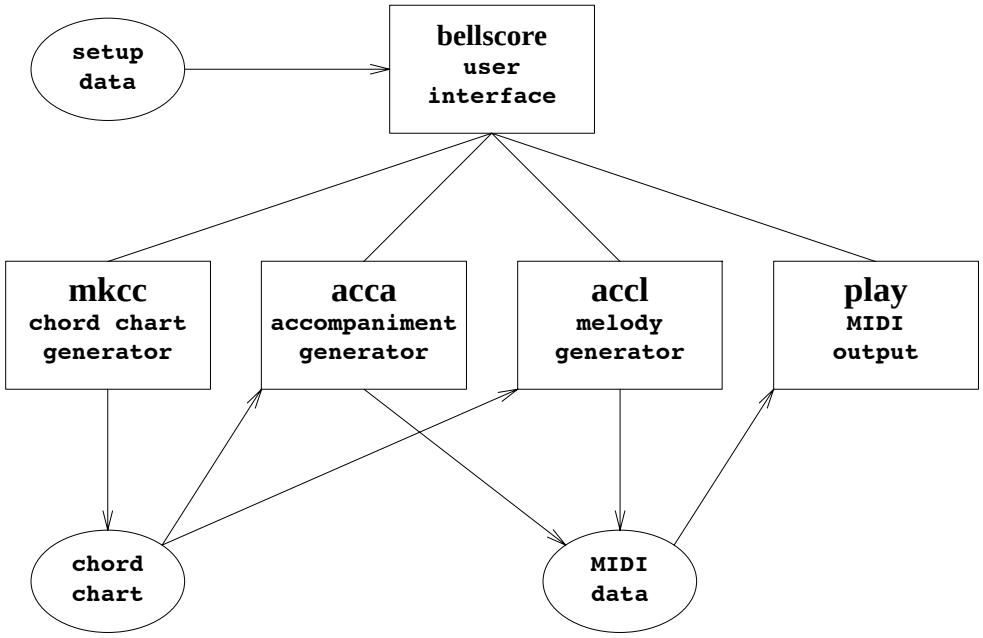


Figure 1 — IMG/1 Software

make effective selections from a small repertoire of precomposed music segments. The selected segments are combined and written out as time-tagged MIDI data (MIDI89)(LANGST89a).

The program ‘‘accl’’ produces melody lines based on the chord chart file. The melody line(s) generated may use up to three different instruments. As with mkcc and acca, accl chooses the composition algorithm based on the current style. The algorithms vary widely in their approaches. Accl produces time-tagged MIDI output.

When the PLAY button is selected bellscore invokes the appropriate program (specified in the startup file) to send the MIDI data to synthesizers. Typically, that program either sends the data to the MPU401 kernel device driver or to the serial line device driver, depending on whether the host machine is equipped with a Roland MPU-401 interface or a serial line / MIDI interface (LANGST89b).

User Interface

IMG/1 runs under the Sun Microsystems windowing system called ‘‘SunView.’’ In its simplest use, IMG/1 displays a small window containing a set of STYLE choices, a LENGTH value, a TEMPO slider, and a set of action buttons. IMG/1 also provides five other control panels to allow control of detailed composition/performance parameters.

Main Control Panel

Figure 2 shows the default display when IMG/1 is first run. This will be the only window needed by the true neophyte.

The STYLE choices are mutually exclusive; each carries with it a set of detail parameters that may be adjusted independently. (As of this writing, ten STYLEs have been implemented; the current goal is to have thirty-five.) LENGTH may be expressed as decimal seconds (e.g. ‘‘23.5’’ for twenty-three and one-half seconds), as video frames at a rate of 30 frames per second (e.g. ‘‘:705’’ for 23.5 seconds), or as a combination of both

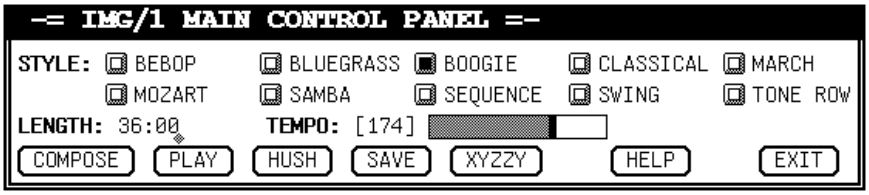


Figure 2 — IMG/1 Main Control Panel

(e.g. "23:15" for 705 video frames). TEMPO is an approximate number of beats per minute. TEMPO may be adjusted by mkcc in order to have an even number of measures within the specified duration.

The COMPOSE button instructs IMG/1 to compose a piece that reflects the parameter choices currently selected and store it as a temporary file. The PLAY button displays a submenu containing all the playable pieces generated during this session, including the last piece generated by hitting the COMPOSE button. The HUSH button stops any piece currently playing. The SAVE button gives the most recently COMPOSEd piece a unique name, insuring that the next COMPOSE operation won't overwrite it (so it can be PLAYed again later). The XYZZY button pops up further parameter windows (see below). The EXIT button terminates the IMG/1 session.

Other Control Panels

The main control panel is the simplest possible interface to IMG/1; some users will never need to go beyond its three parameters. For the more adventurous, however, there are five other control panels available in IMG/1. These panels allow more detailed control of composition and performance parameters.

Figure 3 — "Panel of Experts"

Figure 3 shows the "PANEL OF EXPERTS." This panel contains global parameters and is controlled by the magic XYZZY button in the main control panel. Each of the other detail parameter panels is controlled by a button in the PANEL OF EXPERTS (e.g. "LEAD PARAMS" toggles the "LEAD INSTRUMENT PARAMETERS" panel on and off the screen). A REVOICE button applies parameter changes (e.g. instrument voicings) to existing IMG/1 compositions. Buttons for loading and saving setup files allow information about the current synthesizer setup and all modifyable parameters to be retrieved, edited, and stored. The use of setup files makes IMG/1 both synthesizer-independent and style-independent; the synthesizer complement can be changed simply by editing a setup file; the style repertoire can be changed by adding code to the programs *mkcc*, *acca*, and *accl*, and editing a setup file. A full description of the form of the setup file is beyond the scope of this article, but the appendix contains an example of a typical setup file from which much can be inferred.

The global parameters in the PANEL OF EXPERTS include: the key of the piece, the length of the initial count-in (if any), optimization and debugging controls for the various subprograms, and the synthesizer setup description from the current setup file.

Figure 4 shows the "LEAD INSTRUMENT PARAMETERS" panel that is used to set parameters for the melody generation routines. These parameters include: synthesizer

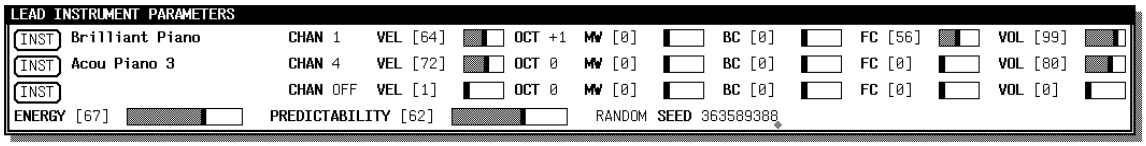


Figure 4 — Melody Parameters Panel

voices to be used (“INST”), MIDI channels (“CHAN”), velocity scaling (“VEL”), octave transpositions (“OCT”), continuous controller settings (“MW,” “BC,” & “FC”), volume scaling (“VOL”), and miscellaneous composition controls (“ENERGY,” “PREDICTABILITY,” & “SEED”).

Selecting the “INST” button pops up a menu containing the list of voices available on the currently selected channel. Once a voice has been selected, a few notes are played to audition the voice. Selecting the “CHAN” field either cycles through the seventeen channel possibilities (1 through 16 and OFF), or pops up a menu containing the list of channels and associated synthesizers. The “VEL” slider sets a multiplicative factor for that channel’s key velocity information. Selecting the “OCT” field either cycles through the possibilities or pops up a menu of transposition choices (-2, -1, 0, +1, or +2 octaves). The “MW,” “BC,” and “FC” sliders set static values for modwheel, breath, and foot controllers. The “VOL” slider sets the channel’s volume controller level (different from VEL which may change the timbre of the sound).

The interpretation of the composition controls is dependent on the style algorithm (see below). Typically, “ENERGY” is used to control rhythmic factors such as the number of notes per measure. “PREDICTABILITY” is usually taken to mean the amount of reuse of earlier themes or motifs. The different styles are free to interpret these parameters in whatever way seems most logical; the goal is to be consistent with the terms “energetic” and “predictable” as commonly understood. The Random number generator “SEED” can either be “Fixed” to start at a given number (to allow repeatability) or “Random” to set the random number seed to a new pseudo-random value every time a melody is generated.

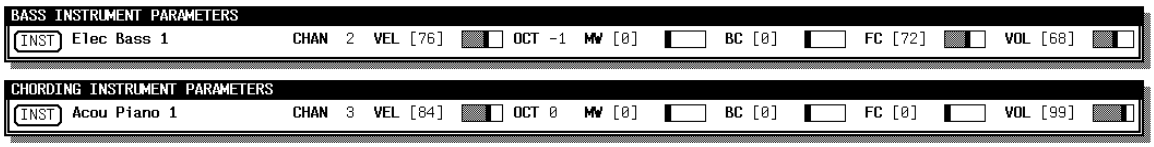


Figure 5 — Bass and Chording Parameters Panels

Figure 5 shows the “BASS INSTRUMENT PARAMETERS” and “CHORDING INSTRUMENT PARAMETERS” panels. These windows have all the parameters described for the Melody Parameters Panel except those dealing with the random number generator (“ENERGY,” “PREDICTABILITY,” and “SEED”). At the moment, the assembly of the accompaniment is a deterministic process based only on the harmonic structure of the piece (see “acca” below); unless (until?) the accompaniment generation is changed to be stochastic, random number generator parameters are not needed.

Figure 6 — Drum Parameters Panel

Figure 6 shows the “DRUM PARAMETERS” panel. It differs from the Bass and Chording windows in that it has no voice selection (“INST”) per se, nor does it provide

octave transposition options. Since some drum machines offer different ‘‘drum kits’’ that can be selected by sending program change commands, this panel has a ‘‘PROG’’ button that pops up a list of the drum kits (if appropriate). Some styles offer different drum parts; for example, the SAMBA style provides three drum parts – a ‘‘basic’’ part that consists of a small latin rhythm section, an alternative ‘‘alt 1’’ part that consists of a full Desi Arnaz latin rhythm section, and a second alternative ‘‘alt 2’’ part that consists of a single trap-set drummer doing the best he can. The other parameters are as described for the Melody Parameters Panel.

Style Algorithms

Each style in IMG/1 is represented by three algorithms: one to generate harmonic structure, one to generate the accompaniment, and one to generate a melody.

mkcc

As mentioned earlier, the harmonic structure of a IMG/1 composition (the output of mkcc) is specified as a chord chart file. The chord chart file is an ascii file consisting of control lines and data lines with (typically) four chord symbols representing the harmonic

```
# Title 617812721
#STYLE  swing
#INCLUDE        "/u/psl/midi/etc/accagc.cc"
#QUANTUM        quarter
#PART Lbgv
Bb  /   /   /   Eb7 /   /   /   Bb  /   /   /   Bb  /   Bb  /
C7  /   /   /   F7  /   /   /   Bb  /   F7  /   Cm  /   F7  /
#PART Lbgv
Bb  /   /   /   Eb7 /   /   /   Bb  /   /   /   Bb  /   Bb  /
C7  /   /   /   F7  /   /   /   Bb  /   Cm  F7  Bb  /   /   /
#PART Lbgb
Eb  /   /   /   Eo  /   /   /   Bb  /   /   /   Bb  /   D7  /
Gm  /   /   /   Gb7 /   /   /   F7  /   /   /   F7  /   /   /
#PART Lbgv
Bb  /   /   /   Eb7 /   /   /   Bb  /   /   /   Bb  /   Bb  /
Cm  /   /   /   F7  /   /   /   Bb  /   Bb  /   Bb  /   /   /
```

Figure 7 — Typical IMG/1 Chord Chart

structure of each measure.

The first line in Figure 7 is a comment (because it begins with ‘‘#’’ followed by a space) that indicates the random number kernel used to generate this particular chord chart, coincidentally the number of seconds since midnight on January first, 1970 (GMT). The #STYLE line defines the musical style intended; this is used by acca and accl. The #INCLUDE line specifies a file that defines all the chord symbols used. The chord symbols themselves are arbitrary sequences of ascii characters separated by whitespace. The format of the chord definition file is described under ‘‘Chord Symbol Definition’’ below. The #QUANTUM line in Figure 7 specifies that each chord symbol has the time value of a quarter note. The #PART lines are essentially comment, although accl uses them to help recognize the overall structure; (although the chords differ in the first, second, and fourth pairs of chord lines, they are based on the same structure).

Mkcc's first task is to determine the length of the piece to be generated. Associated with each STYLE is a duration quantum (not related to the #QUANTUM line in the chord chart); any piece must be an integral multiple of this duration. In order to force an integer multiple, mkcc will adjust the TEMPO specification since the LENGTH specification must be met exactly. The most common duration quantum is two bars; the more free-form styles, e.g. "sequence" have shorter values. Longer duration quantum values assure more graceful overall harmonic structures, but require greater lattitude in adjusting the TEMPO specification.

Once mkcc has determined the length of the piece it then divides the piece into sections as is appropriate for the specified style. As an example, if the piece is in the "bluegrass" style, mkcc first checks whether the length (in bars) can be expressed as 16n+9 (where n ≥ 1); if so, it chooses a general 16 bar pattern that has an accompanying 8 bar reprise, repeats the 16 bar part n times, appends the 8 bar reprise, and finally, appends a single bar ending. The chord patterns include choices of alternative chords or chord sequences, thereby providing variety in the sequences generated. If the length doesn't fit 16n+9, then 16n+1 is tried. Failing that, the following are tried in order: 12n+1, 12n, 8n+2, 8n, 4n+1, 4n, 5n+1, and 5n. These patterns characterize a large fraction of bluegrass chord structures; (perhaps the only surprise here being the 5n, which turns out to be a stylistically stretched 4n). If a match still has not been found, a single bar introduction is generated, the length is reduced by one bar, and the matches are tried again against the new length.

Other styles use different algorithms. The algorithm for "swing" is similar to that for "bluegrass", but the pattern lengths tend to be longer and a two bar "turnaround" sequence is the preferred filler used to adjust lengths. In some cases no harmonic structure is generated by mkcc (e.g. the so-called "tone row" style). In these cases the chord chart simply contains tonic chord symbols to indicate the planned duration of the piece.

Chord Symbol Definition

```
#      name     group     trans
#CHORD  Cm       tri       0,0,0,0,-1,0,0,0,0,-1,0,0
#CHORD  C7       dom7      0,0,0,0,0,0,0,0,0,0,0,0
#CHORD  D7       dom7      2,2,2,2,2,2,2,2,2,2,2,2
#CHORD  Eb       tri       3,3,3,3,3,3,3,3,3,3,3,3
#CHORD  Eb7      dom7      3,3,3,3,3,3,3,3,3,3,3,3
#CHORD  Eo       dom7      4,4,4,4,3,4,4,3,4,4,3,4
#CHORD  F7       dom7      5,5,5,5,5,5,5,5,5,5,5,5
#CHORD  Gb7      dom7      6,6,6,6,6,6,6,6,6,6,6,6
#CHORD  Gm       tri       7,7,7,7,6,7,7,7,7,6,7,7
#CHORD  Bb       tri       -2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2,-2
```

Figure 8 — Excerpts from "/u/psl/midi/etc/accagc.cc"

Figure 8 shows the definitions of the chord symbols used in the preceding chord chart. Each #CHORD line contains 3 entries: the symbol itself, the chord group (the chord upon which this chord is based, tri ≡ major triad, dom7 ≡ dominant seventh, aug5 ≡ augmented fifth, dim5 ≡ diminished fifth), and the transpositions for each of the twelve pitch classes that may appear in the basis chord or in an accompaniment. These chord definitions are used by acca to transform accompaniment fragments in the key of C into the appropriate fragments for each chord symbol. The chord definitions are also used during melody generation to define chord and scale tones at each point in the piece. In this case, all the basis chords are C chords. Thus the definition of "Cm" flats the third

and sixth of a C major triad to get the harmonic structure of C minor. Similarly, the definition of "C7" uses the C dominant seventh unchanged. Note that the E diminished specification ("Eo") is really a diminished seventh chord, so it simply flats the third, fifth, and seventh of a C dominant seventh chord that has been transposed up by four half-steps.

acca

The program acca assembles accompaniments according to the harmonic information contained in the chord chart file. It does so by cutting and pasting precomposed

Root Motion	Chord Basis			
	Maj. Triad	Dom. 7th	Aug. 5th	Dim. 5th
none	2	2	2	2
up a step	1	1	1	1
up a fourth	1	1	1	1
up a fifth	1	1	1	1
up a sixth	1	1	1	1
ending	2	2	2	2
misc	1	1	1	1

Figure 9 — Stored Accompaniment Bars

accompaniments. For each style thirty-six bars of accompaniment are stored. Figure 9 shows the breakdown of stored accompaniment bars. When the harmonic structure isn't changing (root motion = "none" in the figure), a two-bar accompaniment fragment is used; when the root of the harmonic structure is about to move by a recognized interval (up by 2, 5, 7, or 9 half steps) an appropriate fragment is used, adjusted so that rhythmic boundaries are maintained. When the harmonic structure is about to disappear (i.e. at the end of the piece) a special ending fragment is used, aligned so it falls on an even (multiple of 2) bar boundary. When the chord root is changing quickly (duration of a quarter note or less) a relatively static "misc" fragment is used (to avoid fragmentation of motion). With these few special cases a fairly convincing accompaniment can be assembled from very few bars of precomposed material. It is worth noting that the augmented fifth and diminished fifth chord bases are almost never used and could probably be replaced by transformations of the dominant seventh material, thereby reducing the stored accompaniment to a mere eighteen bars.

accl

The program accl generates melodies that follow the harmonic structure specified in the chord chart file. The techniques used depend on the specified style and vary widely. For example, the STYLE button labeled "classical" (which uses a simple Alberti Bass accompaniment) composes a melody by first generating a rhythmic plan from a transition network like that shown in Figure 10.

In Figure 10, the letters represent rhythmic motives, each one measure in duration; for example, "G" is a single whole-note, while "C" is four sixteenths, two eighths, a quarter, and two eighths. Associated with each arc in the network is a probability used to make a weighted random choice. Once the rhythmic plan is known, accl selects melodic motion figures that fit both the rhythmic motive and the harmonic structure at each point.

As a contrasting example, the algorithm used by accl for the "bluegrass" style expends much less effort on rhythmic diversity. It composes solo parts that are particularly suited to the five-string banjo played in the "Scruggs" style. This style typically consists of complicated, rolling sixteenth-note arpeggiations in which the lower or upper (or sometimes middle) voices are artfully arranged to carry the melody while the rest of the arpeggiation provides a harmonic background.

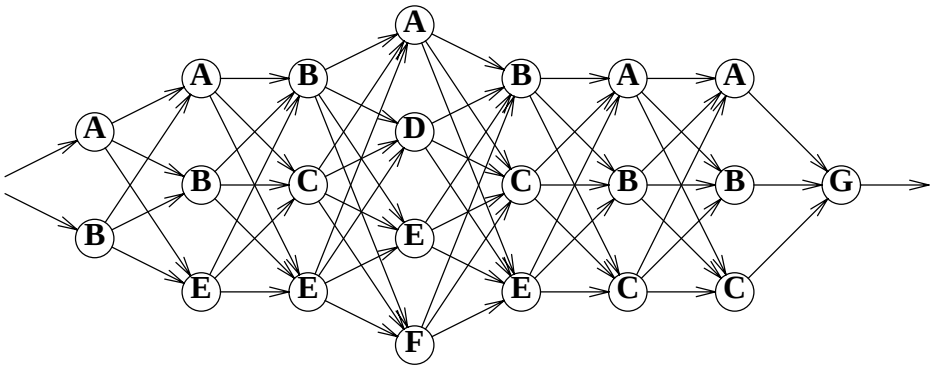


Figure 10 — Alberti Bass Melody Rhythmic Network

Banjo beginnings and endings are chosen from a fixed repertoire (guided by length and other fairly mechanical considerations). The melody notes in the body of the piece are chosen based on the harmonic structure of the piece, a knowledge of right-hand banjo picking patterns (more about these later), left-hand ''fretting'' possibilities, and the mechanics of the five-string banjo itself. For each possible key a particular tuning of the banjo is assumed; it is common for banjo players to retune their instruments such that the notes produced by playing the ''open'' (i.e. unfretted) strings will be the tonic chord of the piece (e.g. A E A C# E in the key of A) or the dominant (e.g. G D G B D in the key of C). As a result, a banjo part for a piece in A will be quite different structurally from a part for the same piece transposed to C.

Accl disallows any note or sequence of notes that violates the mechanical restrictions of the banjo. There are quite a few such constraints. Depending on the tuning, it may be impossible to play a specific pair of notes together (in almost any tuning, D and E below middle C cannot be played together). Other pairs of notes may require that the player's left hand stretch too far. At common (fast) bluegrass tempi it would not be reasonable to expect to be able to use the same finger on the right hand to pluck two notes in succession. Accl limits itself to using right-hand picking patterns chosen from a small set of common patterns. Each of these patterns is an eight note permutation of right-hand fingers (e.g. the ''forward roll'' is: thumb, index, middle, thumb, index, middle, thumb, middle). Although there are 384 permutations of three fingers that involve no consecutive repeats, most banjo playing only uses three or four standard patterns with an occasional exception to accomodate a tricky melodic passage.

Each musical style has specialized code in accl to express the characteristics and peculiarities of the style. Although several styles share code to implement common characteristics, (swing and bebop, march and classical), adding a new style typically means adding a new routine or generalizing an existing routine. As more styles are added, it is expected that generalizing an existing routine or using an already generalized routine will eventually prevail. For mkcc and acca, this is already the case; adding a new style to mkcc means adding new tables of harmonic progression, ending, and overall structure information, while adding a new style to acca means adding thirty-six (carefully chosen) measures of canned accompaniment.

Conclusion

We have described the system known as ''IMG/1'' and some of the programs that it uses to compose music to fit a set of user-supplied parameters. The music generated, while unlikely to appear on the concert stage, is of sufficient quality to be used as incidental or background music and has been used quite successfully as such (despite a few snide suggestions that it be dubbed ''ElevatorScore''). Further expansion is planned, both

in the style repertoire and in the number of user-modifiable composition parameters.

Acknowledgements

Gary Haberman, by dint of his musical expertise and enthusiasm has been a great help in this project. He helped design melody generation algorithms and composed several of the sets of stored accompaniment material. He also acted as a sounding board and reality check, thereby filtering out some of the more ridiculous ideas before much time was wasted on them. Gareth Loy and Michael Hawley provided much of the initial software that allowed us to control synthesizers with computer programs. Stu Feldman provided many useful suggestions and ran interference at the administrative level.

References

BOLOGN83 Bolognesi, T. 1983. "Automatic Composition: Experiments with Self-similar Music.", *Computer Music Journal*, 7(1): 25–36.

DODGE85 Dodge, C. & Jerse, T.A. 1985. *Computer Music: Synthesis, Composition, and Performance*. Schirmer Books.

EBCIOG88 Ebcioglu, K. 1988. "An Expert System for Harmonizing Four-part Chorales." *Computer Music Journal*, 12(3):3–51.

HILLER70 Hiller, L. 1970. "Music Composed with Computers – A Historical Survey." *The Computer and Music*, pp. 42–96. Cornell University Press.

HILLER81 Hiller, L. 1981. "Composing with Computers: A Progress Report." *Computer Music Journal*, 5(4):7–21.

INTELL88 Intelligent Music 1988, "M", "Jam Factory." P.O. Box 8748, Albany, New York.

LANGST86 Langston, P.S. 1986. "(201) 644-2332 • Eedie & Eddie on the Wire, An Experiment in Music Generation." *Proceedings of the Usenix Summer '86 Conference*.

LANGST88 Langston, P.S. 1988. "Six Techniques for Algorithmic Composition." Bellcore Technical Memorandum #ARH-013020 (presented at the 1989 Internation Computer Music Conference).

LANGST89a Langston, P.S. 1989. "Little Languages for Music." Bellcore Technical Memorandum (to appear in *Computing Systems* journal 3(1)).

LANGST89b Langston, P.S. 1989. "Getting MIDI from a Sun." Bellcore Technical Memorandum #ARH-016282.

LANGST90 Langston, P.S. 1990. "Unix MIDI Tools." Bellcore Technical Memorandum (to appear in *Software – Practice & Experience*).

LOY89 Loy, D. Gareth 1989. "Composing with computers--a survey of some compositional formalisms and programming languages for music." In *Current Directions in Computer Music*, ed. Max Mathews. MIT Press.

MIDI89 The International MIDI Association, 1989. *MIDI 1.0 Detailed Specification, Document version 4.1*. I.M. A., 5316 W. 57th St., Los Angeles, CA 90056.

SPIEGE85 Spiegel, Laurie 1985. "Music Mouse" Aesthetic Engineering 175 Duane Street, N.Y.C., N.Y.

APPENDIX

```
# BS setup file - created: Fri Apr 27 14:38:07 1990
#
# Synth-specific information
#SYNTH   'P3 & D110 via psl UART/MIDI box'
#SMISC   'Setup in 2D-396 on host Puddle'
#SINIT   'inst 10=64 | ump'
#SFINI   ' '
#SPLAY   'ump'
#SCLICK  chan=10 key=75
#SVNAM   chan=1 list 8    ' 1 Korg P3'
    'Brilliant Piano' 'Mellow Piano' 'Saxophone' 'Organ'
    'Fretless Bass' 'Drums' 'Drums' 'Drums'
#SVNAM   chan=2 list 128  ' 2 D110 Part1'
    'Acou Piano 1' 'Acou Piano 2' 'Acou Piano 3' 'Honky-Tonk'
    'Elec Piano 1' 'Elec Piano 2' 'Elec Piano 3' 'Elec Piano 4'
    'Elec Organ 1' 'Elec Organ 2' 'Elec Organ 3' 'Elec Organ 4'
    'Pipe Organ 1' 'Pipe Organ 2' 'Pipe Organ 3' 'Accordion'
    'Harpsi 1' 'Harpsi 2' 'Harpsi 3' 'Clav 1'
    'Clav 2' 'Clav 3' 'Celesta 1' 'Celesta 2'
    'Violin 1' 'Violin 2' 'Cello 1' 'Cello 2'
    'Contrabass' 'Pizzicato' 'Harp 1' 'Harp 2'
    'Strings 1' 'Strings 2' 'Strings 3' 'Strings 4'
    'Brass 1' 'Brass 2' 'Brass 3' 'Brass 4'
    'Trumpet 1' 'Trumpet 2' 'Trombone 1' 'Trombone 2'
    'Horn' 'Fr Horn' 'Engl Horn' 'Tuba'
    'Flute 1' 'Flute 2' 'Piccolo' 'Recorder'
    'Pan Pipes' 'Bottleblow' 'Breathpipe' 'Whistle'
    'Sax 1' 'Sax 2' 'Sax 3' 'Clarinet 1'
    'Clarinet 2' 'Oboe' 'Bassoon' 'Harmonica'
    'Fantasy' 'Harmo Pan' 'Chorale' 'Glasses'
    'Soundtrack' 'Atmosphere' 'Warm Bell' 'Space Horn'
    'Echo Bell' 'Ice Rains' 'Oboe 2002' 'Echo Pan'
    'Bell Swing' 'Reso Synth' 'Steam Pad' 'Vibe String'
    'Syn Lead 1' 'Syn Lead 2' 'Syn Lead 3' 'Syn Lead 4'
    'Syn Bass 1' 'Syn Bass 2' 'Syn Bass 3' 'Syn Bass 4'
    'Acou Bass 1' 'Acou Bass 2' 'Elec Bass 1' 'Elec Bass 2'
    'Slap Bass 1' 'Slap Bass 2' 'Fretless 1' 'Fretless 2'
    'Vibe' 'Glock' 'Marimba' 'Xylophone'
    'Guitar 1' 'Guitar 2' 'Elec Gtr 1' 'Elec Gtr 2'
    'Koto' 'Shamisen' 'Jamisen' 'Sho'
    'Shakuhachi' 'Wadaiko Set' 'Sitar' 'Steel Drum'
    'Tech Snare' 'Elec Tom' 'Revrse Cym' 'Ethno Hit'
    'Timpani' 'Triangle' 'Wind Bell' 'Tube Bell'
    'Orche Hit' 'Bird Tweet' 'One Note Jam' 'Telephone'
    'Typewriter' 'Insect' 'Water Bells' 'Jungle Tune'
#SVNAM   chan=3 like 2    ' 3 D110 Part2'
#SVNAM   chan=4 like 2    ' 4 D110 Part3'
#SVNAM   chan=5 like 2    ' 5 D110 Part4'
#SVNAM   chan=6 like 2    ' 6 D110 Part5'
#SVNAM   chan=7 like 2    ' 7 D110 Part6'
#SVNAM   chan=8 like 2    ' 8 D110 Part7'
#SVNAM   chan=9 like 2    ' 9 D110 Part8'
#SVNAM   chan=10 list 4   '10 D110 Rhythm'
```

```
    'Standard Trap Set' 'Weird Trap Set' 'Weird Trap Set' 'Weird Trap Set'
#SKMAP   chan=10 ' 38=40 40=38'
#SVNAM   chan=11 list 0   '11 UNUSED'
#SVNAM   chan=12 list 0   '12 UNUSED'
#SVNAM   chan=13 list 0   '13 UNUSED'
#SVNAM   chan=14 list 0   '14 UNUSED'
#SVNAM   chan=15 list 0   '15 UNUSED'
#SVNAM   chan=16 list 0   '16 UNUSED'
#
# Style-specific information
#STYLE   bebop    'BEBOP' 480 960, 10,0,0,0 'Bebop Jazz'
#PARMS   bebop    key=10 cnti=0 len='21:00' MM=183 ener=60 pred=67 seed='0'
#LEAD    bebop    42,4,40,2,0,0,0,99 0,0,0,2,0,0,0,1 0,0,0,2,0,0,0,1
#BASS    bebop    inst=89 chan=2 vel=99 oct=2 ctl=0,0,0,99
#CHORD   bebop    inst=2 chan=3 vel=64 oct=2 ctl=0,0,0,99
#DRUM    bebop    inst=0 chan=10 vel=64 part=0 ctl=0,0,0,99
#STYLE   grass    'BLUEGRASS' 480 240, 0,0,0,0 'Bluegrass Banjo'
#PARMS   grass    key=7 cnti=1 len='28:00' MM=150 ener=65 pred=40 seed='0'
#LEAD    grass    105,4,127,2,0,0,0,72 0,0,0,2,0,0,0,0 0,0,0,2,0,0,0,0
#BASS    grass    inst=93 chan=2 vel=64 oct=2 ctl=0,0,0,99
#CHORD   grass    inst=101 chan=3 vel=104 oct=2 ctl=0,0,0,72
#DRUM    grass    inst=0 chan=0 vel=1 part=0 ctl=0,0,0,1
#STYLE   boogi    'BOOGIE' 480 960, 10,0,0,0 'Boogie-Woogie'
#PARMS   boogi    key=0 cnti=0 len='36:00' MM=174 ener=66 pred=62 seed='0'
#LEAD    boogi    1,1,64,2,0,0,56,99 3,4,72,2,0,0,0,80 0,0,0,2,0,0,0,0
#BASS    boogi    inst=91 chan=2 vel=76 oct=1 ctl=0,0,72,64
#CHORD   boogi    inst=1 chan=3 vel=88 oct=2 ctl=0,0,0,99
#DRUM    boogi    inst=1 chan=10 vel=32 part=0 ctl=0,0,0,64
#STYLE   class    'CLASSICAL' 480 960, 0,0,0,0 'Alberti Bass'
#PARMS   class    key=0 cnti=0 len='28:00' MM=120 ener=50 pred=50 seed='0'
#LEAD    class    17,4,80,2,0,0,0,99 0,0,0,2,0,0,0,0 0,0,0,2,0,0,0,0
#BASS    class    inst=17 chan=2 vel=64 oct=2 ctl=0,0,0,99
#CHORD   class    inst=33 chan=0 vel=1 oct=2 ctl=0,0,0,1
#DRUM    class    inst=0 chan=0 vel=1 part=0 ctl=0,0,0,1
#STYLE   march    'MARCH' 480 960, 10,0,0,0 'Strident March'
#PARMS   march    key=8 cnti=0 len='24:00' MM=120 ener=50 pred=50 seed='0'
#LEAD    march    41,4,80,2,0,0,0,99 0,0,0,2,0,0,0,0 0,0,0,2,0,0,0,0
#BASS    march    inst=48 chan=2 vel=64 oct=2 ctl=0,0,0,99
#CHORD   march    inst=38 chan=3 vel=64 oct=2 ctl=0,0,0,99
#DRUM    march    inst=0 chan=10 vel=64 part=0 ctl=0,0,0,99
#STYLE   mozar    'MOZART' 360 360, 0,0,0,0 'Mozart Waltz'
#PARMS   mozar    key=0 cnti=0 len='30:00' MM=96 ener=50 pred=50 seed='0'
#LEAD    mozar    17,4,80,1,0,0,0,99 0,0,0,2,0,0,0,0 0,0,0,2,0,0,0,0
#BASS    mozar    inst=17 chan=0 vel=1 oct=2 ctl=0,0,0,1
#CHORD   mozar    inst=33 chan=0 vel=1 oct=2 ctl=0,0,0,1
#DRUM    mozar    inst=0 chan=0 vel=1 part=0 ctl=0,0,0,1
#STYLE   samba    'SAMBA' 480 960, 10,11,12,0 'Latin Samba'
#PARMS   samba    key=4 cnti=0 len='48:00' MM=120 ener=50 pred=67 seed='0'
#LEAD    samba    70,4,99,2,0,0,0,99 0,0,0,2,0,0,0,0 0,0,0,2,0,0,0,0
#BASS    samba    inst=87 chan=2 vel=64 oct=2 ctl=0,0,0,99
#CHORD   samba    inst=7 chan=3 vel=64 oct=2 ctl=0,0,0,99
#DRUM    samba    inst=0 chan=10 vel=64 part=0 ctl=0,0,0,99
#STYLE   seque    'SEQUENCE' 480 120, 0,0,0,0 'Pentatonic Wallpaper'
#PARMS   seque    key=0 cnti=0 len='10:00' MM=168 ener=50 pred=50 seed='0'
```

```
#LEAD    seque    72,4,72,2,0,0,0,99 79,2,80,2,0,0,0,99 0,0,0,2,0,0,0,0
#BASS    seque    inst=87 chan=0 vel=1 oct=2 ctl=0,0,0,1
#CHORD   seque    inst=69 chan=0 vel=1 oct=2 ctl=0,0,0,1
#DRUM    seque    inst=0 chan=0 vel=1 part=0 ctl=0,0,0,1
#STYLE   swing    'SWING' 480 960, 10,0,0,0 'Swing Combo Jazz'
#PARMS   swing    key=10 cnti=0 len='60:00' MM=128 ener=60 pred=67 seed='0'
#LEAD    swing    60,4,80,2,0,0,0,99 0,0,0,2,0,0,0,0 0,0,0,2,0,0,0,0
#BASS    swing    inst=89 chan=2 vel=64 oct=2 ctl=0,0,0,99
#CHORD   swing    inst=2 chan=3 vel=64 oct=2 ctl=0,0,0,99
#DRUM    swing    inst=0 chan=10 vel=64 part=0 ctl=0,0,0,99
#STYLE   toner    'TONE ROW' 360 360, 0,0,0,0 '12-tone Sequences'
#PARMS   toner    key=0 cnti=0 len='21:00' MM=120 ener=74 pred=67 seed='0'
#LEAD    toner    30,2,96,2,0,0,0,99 70,3,80,2,0,0,0,99 1,4,80,2,0,0,0,99
#BASS    toner    inst=89 chan=0 vel=1 oct=2 ctl=0,0,0,1
#CHORD   toner    inst=1 chan=0 vel=1 oct=2 ctl=0,0,0,1
#DRUM    toner    inst=0 chan=0 vel=1 part=0 ctl=0,0,0,1
```