The Communal Groove Machine

Amnon Wolman and Cantor Becker
Center for Music Technology,
School of Music, Northwestern University
amnon@nwu.edu or amnon@vm.tau.ac.il, canton@nwu.edu

Abstract
In order to create the eleven hours of techno dance music needed for the production of "Amnon Wolman's Andy Warhol's Diaries" (presented in June 1994 in Chicago) we developed the "Communal Groove Machine" (CGM) which generated the background dance music used in the piece. The CGM combines the technology of auto-generative music algorithms with that of text based virtual communities. Together, these technologies allow for the interactive generation of a piece of music by a community, moderated by a single person, the composer.

1. General Background

"Amnon Wolman's Andy Warhol's Diaries" is a music/theater piece in which the audience dances to techno music created by the Communal Groove Machine (CGM). This music serves as a continuous backdrop for three live singers/acting performers as Andy Warhols. Throughout the course of the performance, these actors present 31 music/theatre vignettes based on excerpts from the Andy Warhol Diaries, each lasting anywhere from 5 to 20 minutes.

Andy Warhol wrote his diaries in order to keep track of his finances in the early seventies, as he was audited annually by the IRS. Since his personal life was intertwined with his professional life, he kept a diary in which every event in his social life and the amount of money spent on it was recorded. For example: "Got myself together and picked up Peter Wise, Cabbied to Keith Haring's ($8.50)..." Warhol's diary enables us to reconstruct a superficial image of New York's social scene during the seventies and eighties, the period when the AIDS epidemic erupted into the promiscuous "live and let live" scene that had otherwise given up on all ideals.

2. Background: the MOO

The Communal Groove Machine (the CGM) is an auto-generative music algorithm written and functioning within a text based virtual community (a MOO). This environment allows participants to create techno music using a "tangible" program interface, where their actions provide the seeds needed by the generative music algorithm to create the songs.

The virtual environment is which participants interact with the CGM is a MOO, or "Multi-user dimension (Object Oriented)". The MOO, invented by Pavel Curtis of Xerox Parc corporation, was based on previous work by Stephen White, also of Xerox Parc, is a text-based virtual reality (VR), extendible from within the VR using Pavel Curtis' own object-oriented programming language, which will be referred to as "MOOcode". A MOO server is designed to accept simultaneous connections from multiple users anywhere on the Internet who have access to simple text relaying client programs such as "telnet". A user may use the MOO to interact with other users and objects placed in it, and may use the MOOcode to extend the environment itself.

Currently, there are at least 10 extremely popular MOOs on the Internet. Pavel Curtis' original LambdaMOO, available via telnet to lambda.parx.xerox.com, port 8888, typically hosts over 100 simultaneous user connections.

2.1 MOOcode

The Communal Groove Machine itself is entirely written in MOOcode, a language whose fundamental architecture focuses upon textual interaction and communication by members of a MOO community. Every data structure element in MOOcode is either an OBJECT, a VERB, or a PROPERTY. OBJECTS are generally tangible, things such as users' virtual selves (called players),
or the virtual rooms and tools they make. 
.VERBS are programs that are run by \$OBJECTS (such as players who invoke 
command-style verbs). PROPERTIES are 
variables that, among other things, describe 
the interaction and intersection between 
different \OBJECTS. So, for example, every 
player \OBJECT SJECT has programmed it as an 
"inventory". \VERB, which \Alphakes back to the 
player \OBJECT SJECT. A \PROPERTY defined on all 
\OBJECTS, \CONENTS, which lists what 
\OBJECTS are located within the player. 
In other words, \$OBJ.INVENTORY is a \VERB 
that returns \$OBJ.CONTENTS. If 
\$OBJ.LLOCATION equals \$OBJ.B, then 
\$OBJ.CONTENTS contains in its list 
\OBJ.\CONENTS.

2.1 MOOcode as used for the CGM.

The Communal Groove Machine consists 
of four \OBJECTS. The first is \LISTENER, 
a dummy player \OBJECT SJECT whose job it is to 
relay the MIDI data generated by the CGM 
to a disk file. The other three \OBJECTS 
appear to other players in the MOO as three 
rooms connected by conventional exits. 
Two of the rooms are "libraries" where 
loopable rhythm, bass, and chord scores are 
stored. These two rooms also contain 
.VERBS that quantity exemplar MIDI data so 
that the player can create new elemental rhythm 
sequences that the CGM's algorithm can use. 
The rooms have the potential for including 
.VERBS that will manage and combine 
existing rhythm elements so as to create new 
ones. As an alternative, players can 
contribute to the library, adding their own 
baselines and chordlines.

The fourth \OBJECT SJECT, the CGM room, 
comprises the music generation algorithm 
and its interface that allow users to 
set state variables that guide the algorithm. 
These state variables help to prevent 
situations in which players' contributions of 
particular baselines and chordlines produce 
songs containing those exact baselines and 
chordlines. Instead, the CGM may mix, 
mash, or ignore progressions contributed by 
players.

3. Background: Techno music

Techno is a style of popular music which 
celebrates dance, technology, and rhythm, 
placing little weight on how "human" or 
natural its rhythms and samples sound. Until 
recently, Techno music was mostly heard in 
"raves", underground dance parties featuring 
hallucinogens, lasers, and very loud and fast 
techno music. Around 1991, a slower, 
gentler form of techno music became the 
mainstay of Top-40, still heard in дискотеках 
dance clubs today.

In order to represent the techno idiom we 
have refined an algorithm which accurately 
describes the music. The algorithm's 
knowledge-base is made up of libraries of 
cliches, stored as elemental rhythmic 
patterns, chord progressions, bass melodic 
lines, and part sequences. Elements are 
chosen either by random or weighted 
selection, and are combined to form patterns 
that correspond to clichéd introductory, 
verse, bridge, outro, and ending sequences. 
Characteristics most definitive of techno 
music, such as the A-B-A-B form and the 
use of long samples, are coded rigidly into 
the CGM algorithm. However, since the 
CGM can't generate samples itself, it 
provides MIDI cues to trigger samples which 
must be selected by the composers. 
The composers' choice of samples are influenced by 
guidelines given by players who 
interacted with a message-writing interface 
within the CGM.

4. The implementation

When "walking" around the MOO the 
CGM room is described to players as follows:

The Communal Groove Machine: 
You are simultaneously at 
the vertex, zenith, and 
center of an enormous and 
terribly impressive 
glittering machine. Busy 
neon green LEDs flash while 
synchronized relays click on 
and off to ghost rhythms and 
grooves. All around you, 
knobs, buttons, switches, 
sliders, levers, and various 
visual feedback devices 
generate an awesome magnetic 
force that inspires you to 
twist, push, flip, slide, and 
pull everything you see. You 
see the Tempo Knob, the Mood 
Modulator, the Complexity
Lever, the Author Hot Pad, the Length Dial, the Title, the INSTRUCTION PAMPALET, a big button labeled DOIT, the Song Log, the Style Selection Slider, and the Cliché Switch here.

After entering the CGM, players have the opportunity to interact with and examine different parts of the program itself, so it is difficult to separate the discussion of the VR interface from the algorithm (A player can type "smell the generative algorithm" and expect to receive an interesting response, even though the smell of the algorithm isn't essential to its function.) Most of the levers, dials, and buttons are SUBJECTS with defined VERBS that set state variables which constrain parameters of the music algorithm. Some of the controls allow for more qualitative input, so that the player can specify how she would like to see the final product mastered, while others require numeric or binary settings. If a player types "read song log" she can see what others have done before her. For example typing "look mood modifier" returns:

A transkingneuromodulation interface that most people were taught to use is a global dream propagated by the Mushroom Machine Elves during the Harmonic Convergence. You type 'set mood to <mood>', where <mood> is either Major, Minor, or Ambiguous.

Players can easily control the tempo, length, "mood" (major, minor, or ambiguous), and "style" of the song. (Hardcore, Funky, Random, or Thrashing.) Two additional controls, "complexity" and "riches", may also be manipulated by the user. (Their effects on the algorithm are discussed later.)

4.1 The Communal Groove Algorithm
The algorithm is a four-stage process:

4.1.1 Stage 1: GET-CONSTRAINTS
In this stage various PROPERTIES are set by players interacting with the CGM's user interface. After the user configures various CGM properties, pressing the DOIT button, invokes stages 2 through 4.

4.1.2 Stage 2: GENERATE-BLOCKS
In this stage, a series of VERBS generate "music blocks". The complexity PROPERTY set by players in the GET-CONSTRAINTS stage determines how many blocks are to be created. A "music block" is a loopable score, usually 16 beats long, consisting of MIDI data for 10 individual drums, a bass line, and a chord line. Each drum's rhythm is selected from a set of elemental rhythm sequences stored in a rhythm library. The choice of elements is made by a weighted random choice in which weights are determined by state variables such as "style" and "mood".

Bass and chord progressions are chosen from libraries organized by tonality ("mood"). If the cliché PROPERTY is set to TRUE, then "best match" paired bass and chord progressions are used to create a clichéd song. On the other hand, if the cliché PROPERTY is set to FALSE, then any bass progression will be matched with any chord progression in the same tonality, creating a less predictable song.

4.1.3 Stage 3: ASSIGN-BLOCKS
After the initial blocks have been generated, an auxiliary VERB is triggered that assigns music blocks to each of the following: the verse, the chorus, and the bridge parts. Multiple verses, choruses, and bridges can exist if the complexity PROPERTY is sufficiently great.

4.1.4 Stage 4: ASSEMBLE-SONG
The ASSEMBLE-SONG stage consists of a random introduction sequence and random ending sequence that brackets a loop in which verse, chorus, and bridge parts are appended until the song's LENGTH property (set in GET-CONSTRAINTS) equals the length of the song being generated. Select instruments (such as drums, bass, or chorus) will be parsed out of the music blocks as they are concatenated to form the song. For example, the following two VERBS output an introduction part and verse part to the SLEETNER.

SGM: INTRODUCTION 1
4.2. Playing the Songs

After a song has been generated by the $CGM$ and recorded by the $LISTENER$, it must pass through two more programs before it can be mastered. The first is a UNIX resident C program that performs two functions; the first function is to generate a random melody based on the chord progressions and the random number seed provided by the $CGM$, the second function is to parse each music event into an event table. The event tables are subsequently downloaded onto a MacIntosh, where a MAX patch drives a set of samples, synthesizers, and effects processors to produce the final mix. Samples for each song are chosen based upon the choices the song author leave in the $AUTHOR-NOTE-PAD$. For example,

```
SONG # 29 : Sat May 7
13:65 & l 1994 CDT
AUTHOR'S NOTES:
This song is dominated by an ultra deep, paper-fat bass groove. Lots of liquidy-slow bubbly sounds punctuate the heavy percussion. No V O C A L S! Except for some rastaman saying '6 processors'. Peace. -jco@columbia.edu
```

5. Conclusions and Future Plans

As we write this paper, we are in the process of mastering the songs for the production of the piece. The success of the algorithm will be evaluated during the performance, as we observe the enthusiasm of the audience. Results will be reported during the oral presentation of this paper in September, 1994.

For lack of space, two components of the $CGM$, the melody generator and the MAX mixer patch, are not discussed in this paper. We plan to describe both in detail in another paper in the near future.

Already in production is a World Wide Web interface to the Communal Groove Machine, available via: http://cctrnet.acm.org/medbecker/techno/techno.html The WWW interface provides users with information on the $CGM$, and allows them to listen to examples of final mixes. Future plans include making it possible to manipulate GET-CONSTRAINTS properties from within WWW clients, eventually allowing users with MIDI systems to listen to live output from the $CGM$.

References