Algorithmisches Komponieren

Ein Weg zu musikalischer Kreativität und Selbsterfahrung?

von Wolfgang Martin Stroh

Durch den unaufhaltsamen Einzug des Computers in den Musikunterricht sind Inhalte, die die Curriculumdiskussion der 70er und 80er Jahre ausgeschieden hat, reaktiviert worden. Notenlehrgänge, Intervallgehörbildung, Lern- und Bildungsstoffe aus der deutschen Musikgeschichte werden multimedial im Musiksaal wieder salonfähig. Die Rebellion der Schülerinnen und Schüler gegen eine derart lernlastige Auseinandersetzung mit Musik hält sich aufgrund des Fetischcharakters von Computern in Grenzen. „Kreativität", das alte Ideal der Curriculumreform, wird der Microsoftideologie folgend weitgehend auf mausgesteuertes *multiple choice* reduziert. Insgesamt ein musikdidaktisch sehr unbefriedigender Zustand!

Methodisch stehen sich derzeit zwei Herangehensweisen an (Musik-)Computer gegenüber. Die eine benutzt *MIDI-Recordingsysteme*, die sich in der Profi-Szene als brauchbares Werkzeug für Komponieren und Notenschreiben etabliert haben. Die andere ist mit dem Schlagwort *algorithmisches Komponieren* verknüpft und beruht auf der Idee, daß Schülerinnen und Schüler musikproduzierende Programme entwickeln. Musiklehrerinnen und -lehrer haben verständlicherweise in der Regel keinen Zugang zu solcherart Musikmachen. Informatiklehrerinnen und -lehrer, die mit Schülerinnen und Schülern Programme entwickeln, dürften sich jedoch durch das algorithmische Komponieren angesprochen fühlen. An sie richtet sich der vorliegende Beitrag.

MIDI-Recordingsystem („Sequencer"): Ein Computerprogramm, das MIDI-Daten, die von Musikinstrumenten ausgesandt werden, wie ein Tonbandgerät aufnimmt, speichert, als „Partitur" anzeigt, in herkömmliche Notenschrift verwandelt, editierbar macht und entsprechend verändert wieder so abspielen kann, daß MIDI-Musikinstrumente das Abgespielte als Steuerbefehle verstehen. MIDI-Recordingsysteme dienen im einfachsten Fall als digitales Musik-Notizbuch (Aufnehmen/Abspielen), als Kompositionswerkzeug, aber auch als Musikinstrumente.

MIDI-Recordingsysteme erzeugen keine Klänge, sondern senden nur MIDI-Daten als Steuerbefehle an elektronische Instrumente (oder Soundkarten), die dann die Klänge erzeugen.

Wichtige Produkte des deutschen Marktes: „Cubase", „Notator Logic", „Cakewalk".

Algorithmus kennzeichnet die hohe Sphäre der Avantgarde-Computermusik, wie sie auf der jährlichen „International Computer Music Conference" und im „Computer Music Journal" zelebriert wird und Schülerinnen und Schülern musikalisch sehr fern liegt. Diese Musik wird an Forschungsinstituten, am Rundfunk oder an Universitäten produziert. Im Zentrum steht seit den 60er Jahren die Idee, Musik durch Algorithmen zu erzeugen und den herkömmlichen Kompositionsbegriff mit dem des Computerprogramms zu verschmelzen. Es *kann* kein Ziel des algorithmischen Programmierens von Musik sein, Musikstücke, die es auch ohne Computer gibt – von Mozarts Würfelspiel bis zu Schönbergs Zwölftonkanon – nachzuahmen. Algorithmische Musik sollte vielmehr genuine Computermusik sein, die ihre Herkunft nicht verleugnet. Der Computer ist zu einer eigentümlichen *musikalischen Akrobatik* fähig: groteske Tonsprünge, aberwitziges Tempo, extreme Klangwechsel, Erzeugung von Kippfiguren, irreale metrische Genauigkeit, komplexe Polyrhythmik, Formen, die sich zufällig oder chaotisch verwandeln, Microtöne – das ist sein Metier. Der Reiz algorithmischen Komponierens für Schülerinnen und Schüler, die im wesentlichen nur auf Popmusik stehen, liegt neben dieser musikalischen Akrobatik auch in der latenten Unvorsehbarkeit des musikalischen Resultats, dem Aleatorischen und Offenen der Musik. Der Computer avanciert vom bloßen Werkzeug zum (Mit-)Produzenten.

Algorithmisches Komponieren ist nicht nur mit *Computer*erfahrung, sondern auch mit musikalischer *Selbst*erfahrung verbunden. Algorithmische Musik führt im Sinne experimenteller Musik die Hörerinnen und Hörer aufgrund ihrer musikalischen Akrobatik an Grenzbereiche bisheriger musikalischer Erfahrungsweisen. Mit den folgenden vier Beispielen möchte ich daher zeigen, wie mit einfachen programmiertechnischen Mitteln solche Grenzerfahrungen erarbeitet werden können. Die aktuelle Techno-Szene hat bewiesen, daß es musikalisch nicht anrüchig sein muß, wenn Menschen mit Computern akrobatisch umgehen, solange sie musikalische Grenzerfahrungen von individueller Bedeutung vermitteln.

Die wichtigsten didaktischen Maximen algorithmischen Komponierens lauten also:

Systeme und Literatur zum Thema

Algorithmische Komposition und „Computermusik": Computerprogramme, die mittels Algorithmen Musik erzeugen, oft auch „interaktiv", d.h. indem sie von außen dem Computer zugeführte Daten weiterverarbeiten. Wichtigste Beispiel in Deutschland für Systeme zur flexiblen Erzeugung algorithmischer Musik:

- Lejaren A. Hiller „Illiac Suite" (1956), vorgestellt in Darmstadt 1963 (Illinois).
- Gottfried Michael Koenig „Projekt 1 & 2", seit 1969 laufend weiterentwickelt (Utrecht und Stuttgart).
- Clarence Barlow „Autobusk", seit 1978 laufend weiter entwickelt bis zur Verkaufsreife 1993 (Köln).
- Dirk Reith u. Thomas Neuhaus „PPP" (Parameter Processing Program), seit 1985 ständig weiter entwickelt.
- Wolfgang Martin Stroh „Midi-Planetarium" 1991.

Jedes dieser Systeme hat sich durch Kompostionen und/oder wichtige Konzertvorführungen einen Namen in der Avantgardeszene Deutschlands gemacht.

Literatur zum Thema

Stroh, W. M.: Midi-Exprimente und Algorithmisches Komponieren – Eine Anleitung zum kreativen Programmieren und Komponieren am Computer. Midipädagogische Schriftenreihe, Band 3. Berlin: Musiklabor, 1990.

Sroh, W. M.: Midi-Experimente und Algorithmisches Komponieren – Band 2: Programme und Projekte für den Musikunterricht und die Musikpraxis. Midipädagogische Schriftenreihe, Band 6. Berlin: Musiklabor, 1991.

Schaffrath, H. (Hg.): Computer in der Musik – Über den Einsatz in Wissenschaft, Komposition und Pädagogik. Stuttgart: Metzler, 1991. Darin die Beiträge von Dirk Reith und Walter Schröder-Limmer über „Algorithmisches Komponieren".

Goltermann, F.: Algorithmisches Komponieren – Einsatz Neuer Technologien bei der musikalischen Produktion, durchgeführt in einer 9. Hauptschulklasse. Braunschweig: Staatsexamensarbeit, 1993.

Erläuterungen zu den wiedergegebenen Programmteilen

`Out(X)`: steht im folgenden für ein Unterprogramm oder einen Befehl, der die Zahl „X" an die MIDI-Schnittstelle sendet. Der jeweilige Befehl bzw. das jeweilige Unterprogramm hängt vom Computer, der Programmiersprache und der Art der Schnittstelle ab. Für den *Atari* mit integrierter Schnittstelle ist „Out" ein BIOS-Befehl, der in Omikron-BASIC „BIOS(,3,3,X)" und in GfA-BASIC „Out 3,X" lautet. Für *Mac*s und PCs gibt es je nach Interface zahlreiche Möglichkeiten. Wenn das MIDI-Interface beispielsweise an den Druckerport angeschlossen ist, muß dieser angesprochen werden.

`WAIT X`: steht für einen Befehl oder ein Unterprogramm, der oder das den Programmablauf für den Zeitraum von X Sekunden anhält. Dies „Anhalten" ist eine in der Regel ausreichende Lösung von Timing-Fragen. (Absolut saubere Timing-Lösungen: siehe Stroh, 1991.)

`Play(Ch,T,V)`: steht im folgenden für ein Unterprogramm, das einen kompletten „Note-ON"-Befehl enthält. Es lautet explizit <Out(143+Ch):Out(T):Out(V)>.

Discovery – Wir wollen wissen, was wir tun!

Um die Grenzen und Fähigkeiten der verwendeten Systeme – Soundkarte, MIDI-Instrument, Lautsprechersystem und menschliches Gehör – zu erforschen, können alle nur denkbaren Schleifen verwendet werden, die von einem mittleren Aktionsbereich ausgehend allmählich an die Grenzen der Geräte und menschlichen Organe vorstoßen. Bei allem ist Vorsicht geboten: Hochtöner könnten durchbrennen oder das Gehör Schaden erleiden. Zugleich kann sich herausstellen, daß ein MIDI-Instrument gar nicht den vollen Tonumfang von 128 chromatischen Tönen (also 10 ½ Oktaven) besitzt oder die 64 oder 128 Sounds weniger attraktiv sind, als ihre Namen verheißen, oder die Lautstärkeskala unnatürlich gestaucht ist (d. h. die Lautstärken über weite Strecken fast gleich bleiben,

▷ Algorithmisch erzeugte Musik führt in Grenzbereiche musikalischer Erfahrung. Dies Phänomen sollte im Unterricht thematisiert werden.
▷ Algorithmische Musik ist Computermusik, d. h. eine Musik, die musikalisch zu erkennen gibt, daß sie von Computern gemacht ist. Die Simulation von Instrumentalmusik sollte vermieden werden.
▷ Algorithmische Musik stiftet eine Einheit von musikalischer und programmiertechnischer Idee. Programmieren, hörendes Erkunden des Programmierten, Verändern und wieder Hören sind die charakteristischen Arbeitsschritte.

Bild 1: Baukasten von Unterprogrammen.

Pitch_Disc (N1, N2)

Ein Tonhöhenvorrat wird über Kanal 1 abgespielt mit Sound Nr. 1 in rasantem Tempo:

```
Out(192):Out(0)
FOR T=N1 TO N2
Play(1,T,90)
WAIT .1
Play(1,T,0)
NEXT
END
```

Sound_Disc (N1, N2)

Verschiedene Sounds werden auf einer Skala abgespielt:

```
FOR P=N1 TO N2
Out (192):Out(P)
  FOR T=60 TO 72
  Play(1,T,90)
  WAIT.1
  Play(1,T,0)
  NEXT
NEXT
END
```

Bend_Disc (0,127)

Auf allen Sounds wird eine Pitchbend-Aktion („glissando") ausgeführt:

```
FOR P=0 TO 127
Out(192):Out(P)
 Play(1, 60, 90)
  FOR B=N1 TO N2
  Out(224):Out(0):Out(B)
  WAIT.02
  NEXT
  Play(1, 60, 0)
NEXT
END
```

Velocity_Disc(N1,N2)

Verschiedene Lautstärkewerte werden auf eine Tonrepetition über Kanal 1 gespielt.

```
FOR V=N1 TO N2
Play(1,60,V)
WAIT .1
Play(1,60,0)
NEXT
END
```

um dann plötzlich rapide zuzunehmen). Die Qualität der verwendeten Musikinstrumente wird erfahren und das Problem der Geräte- und Gehörschädigung in Verbindung mit der Selbstverantwortung der Schülerinnen und Schüler thematisiert. Bild 1, vorige Seite, zeigt einen Baukasten von Unterprogrammen zu solchen Erkundungszwecken.

Ein systematisches Erkundungsprogramm würde eines dieser Baukasten-Unterprogramme in einer Schleife abspielen, die im Gefahrenfall auch abgebrochen werden kann, zum Beispiel:

```
WHILE Aktion = 0
      Pitch_Disc (63-A , 64+A)
      A = A + 1
 WEND
  END
```

Aktion steht für irgendeine Abbruchsaktion (wie SPACE-Taste oder Maus drücken), die die jeweilige Programmiersprache zur Verfügung stellt. **Aktion <>0** heißt, daß die Aktion stattfindet; **Aktion = 0**, daß sie nicht stattfindet.

Erste algorithmische Kompositionen können aus mehreren Discovery-Programmen zusammengesetzt, die Abfolge einzelner Teile einer Komposition kann explizit definiert oder zufällig gesteuert werden. Zum Beispiel:

Random_Discovery

Zufalls-Sounds auf übermäßiger Dreiklangs-Skala

```
WHILE Aktion = 0
  R = RND(8)
  Out(192):Out(R)
   FOR T=36+4*R TO 39+4*R
           Play (1,T,90)
           WAIT .2
           Play(1,T,0)
  NEXT
WEND
END
```

Aus acht Sounds (bei „General Midi-Sounds" wären es die 8 Klavierklänge) wird einer zufällig ausgewählt und dann eine chromatische Viertonfolge auf einer Tonhöhe gespielt, die vom gleichen Zufallsprozeß bestimmt ist. Der Faktor 4 bei den Schleifengrenzen bewirkt, daß die Zufallstranspositionen sich auf die Tasten 36, 40, 44 usf. beziehen („übermäßiger Dreiklang"), was dem Tonumfang der chromatischen Viertonfolge entspricht. Die Idee dieses Programms kann systematisch variiert werden. Die Schülerinnen und Schüler werden merken, daß gewisse Größen für **R**, für die Schleifengröße, die **WAIT**-Zeit und den Faktor bei den Schleifengrenzen musikalisch mehr Sinn ergeben als andere und daß „gut klingende" Kompositionen solche sind, die sich gezielt einschränken.

Fraktale Musik - Wann passen Ton und Bild zusammen?

Besonderer Beliebtheit in Musikerkreisen erfreut sich der *Hüpfer-Algorithmus*. Er wird bei der Musik-Brainmachine „Illuminator" eingesetzt und ist über diesen Weg bereits in viele Therapiekliniken eingedrungen. Der 1986 im „Spektrum der Wissenschaften" vorgestellte *Hüpfer-Algorithmus* erzeugt recht unterschiedliche fraktale Grafiken, die „einfallsreich" auf kleinste Veränderungen der Anfangswerte reagieren und sich im Laufe der Zeit auch unerwartet weiterentwickeln. Der Algorithmus lautet:

$X = Y - SGN(X)*(B*X - C)$
$Y = A - X$

wobei *A*, *B* und *C* (fast) beliebige Anfangswerte sind. Wegen des Vorzeichen-Wechsels *SGN* „hüpft" die grafische Entfaltung um eine schräg liegende Achse.

Die musikalische Frage, wie eine Grafik in Musik umgesetzt werden kann, ist die Gretchenfrage der „fraktalen Musik". Sie kann in einem Projekt mit Schülerinnen und Schülern erfahrungsbezogen „diskutiert" werden: durch Programmieren, Hören, Verändern, Wiederhören, Reflektieren usw.

Die bisher bekannteste und einfachste Lösung war, daß das zweidimensionale Zeichenbrett, auf dem sich die Grafik Punkt für Punkt entfaltet, als Tonhöhenraum für zweistimmige Musik interpretiert wird: Entlang jeder Achse sind mehr oder weniger Tonhöhen „angebracht". Diese Lösung ist musikalisch sehr unbefriedigend. Die Entfaltung der fraktalen Grafik hat nichts zu tun mit der Entfaltung der zweistimmigen Musik. Die Grafik entwickelt sich in den beliebten chaotischen Sprüngen von einem Mittelpunkt aus in einer gewissen Achsensymmetrie. Die Musik tut dies nicht ... Zudem dauert die Entfaltung der Grafik, wenn Punkt für Punkt als Ton umgesetzt wird, viel zu lange, um ästhetisch wirkungsvoll zu sein.

Da diese Punkt-für-Punkt-Musikalisierung unbefriedigend ist, haben Produzenten multimedialer fraktaler Grafik-Programme (mit Musikbegleitung) in der Regel den Ausweg gewählt, frei assoziativ zur Grafikentfaltung Stimmungsmusik zu komponieren und diese der Grafik beizugeben (z. B. auf der CD-ROM „fractal ecstacy" von *Deep River Publ.*, 1994). Auch dies ist eine Notlösung. Überzeugende fraktale Musik müßte solche algorithmische Musik sein, die sich musikalisch in Wechselwirkung mit der Grafik entfaltet und einen „fraktalen" ästhetischen Wert besitzt.

Nach längeren Experimenten mit Studenten und Besuchern interaktiver Installationen habe ich einen Lösungsvorschlag entwickelt, der mich derzeit am meisten überzeugt:

- ▷ Die Musik ist aus dem Entfaltungsprozeß der Grafik abgeleitet und ist keine Stimmungsmusik zum fertigen Bild.
- ▷ Nicht jeder Bildpunkt wird vertont, sondern nur eine Auswahl davon; das Auswahlraster bestimmt ganz wesentlich die Entfaltungsgeschwindigkeit der Grafik und sollte interaktiv beeinflußbar sein.
- ▷ Die Tonhöhen sollen, wie die grafische Entwicklung selbst, um einen Mittelpunkt herum gruppiert und achsensymmetrisch angeordnet sein.

Fraktale Musik nach dem Hüpfer-Algorithmus

```
A=4000:B=.4:C=-3000              'Anfangswerte
Xo=300:Yo=175                        'Mittelpunkt

WHILE Aktion <>0
   X = Xalt - SNG(Xalt)*((B*Xalt - C)) MOD 64000
```

```
    Y = A - Xalt MOD 40000
    Xalt = X
    Yalt = Y
     Farbe
     DRAW Xo+X/100, Yo + Y/110
     IF Z MOD 35 = 0 THEN Ton
     Z = Z+1
WEND
END
```

In den beiden Zeilen, in denen die Formel für den Hüpfer-Algorithmus steht, wurde „MOD 64000" und „MOD 40000" geschrieben, um beim Überschreiten des 640:400-Bildschirmes die Bildpunkte exakt auf das Ausgangsbild zurückzuprojizieren. Das Programm enthält das Unterprogramm „Ton", das die Tonauswahl trifft und Töne erzeugt, und „Farbe", das gegebenenfalls die Grafikpunkte färbt. Die **IF**-Bedingung vor **Play** besagt, daß jeder 35. Bildpunkt vertont wird. Wie bei jeder fraktalen Grafik müssen die „alten" **X/Y**-Werte festgehalten werden, bis die Berechnung der „neuen" **X/Y**-Werte abgeschlossen ist. Zum Unterprogramm „Ton":

```
Ton:=
Abstand = ((X-Xo)-2 + (Y-Yo)-2)-(1/2)
T = 36 + INT(Abstand/400)
Play(1,Talt, 0)
Play(1,T,90)
Talt=T
WAIT .04
```

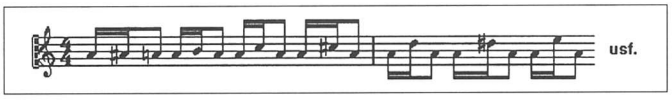
Der euklidische Abstand des Bildpunktes **X/Y** vom Mittelpunkt **Xo/Yo** wird in einem musikalisch sinnvollen Verkleinerungsmaßstab in Tonhöhen umgesetzt. Der tiefste Ton ist der der Taste 36 und liegt am Mittelpunkt. Die **WAIT**-Zeit ist je nach Rechnergeschwindigkeit und nach musikalischen Gesichtspunkten zu wählen.

Die „fraktale Musik", die das vorliegende Programm bei den unterschiedlichen Anfangswerten produziert, ist bestimmt durch wuchernde Entwicklung minimalistischer Motive, die sich sprunghaft ändern können. Weitere an die Grafik angelehnte IF-Bedingungen können verschiedenen Zeichenbereichen unterschiedliche MIDI-Kanäle und damit Klangfarben zuweisen. Hierdurch entsteht unter Umständen virtuelle Zweistimmigkeit nach dem Prinzip der Kippfiguren (Bilder 2 und 3). Rhythmische Strukturen erhält man, indem man gewisse Zeichenbereiche mit „Pause" versieht.

Musikalische Kippfiguren – Unser Gehör wird aktiv

Im Intro der meisten Techno-Hits spielen mehrere Stimmen einfache Tonrepetitionsgestalten, bei denen es schwierig ist, die „Eins" zu finden, und die wie eine grafische „Kippfigur" unterschiedlich gehört und aufgefaßt werden können. Erst nach längerer Zeit der Verunsicherung setzen Bass-Drum und Snare mit dem Grund-

Bild 2: Kippfigur nach van Noorden.



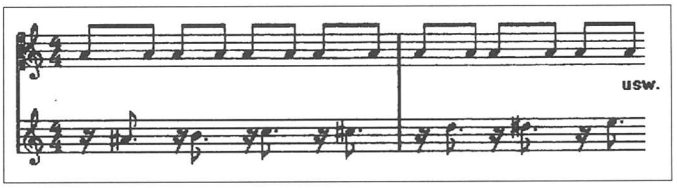


Bild 3: Bei größeren Tonabständen „kippt" die Figur von Bild 2 in zwei melodische Linien um.

Beat „four-to-the-floor" ein, was von den Ravern mit ekstatischem Jubel begrüßt wird. Das musikalische Prinzip solcher „Kippfiguren" ist in der einstimmigen Barockmusik gerne verwendet worden, um virtuelle Zweistimmigkeit zu suggerieren. Bei schnellen Tonfolgen, deren Töne weit auseinanderliegen, ist es nicht mehr möglich, die Töne linear-melodisch zu hören. Vielmehr hört man zwei getrennte Stimmen. Leo Pauls A. S. van Noorden hat 1975 dies gestalttheoretisch verwunderliche Phänomen mit folgender Kippfigur demonstriert und erforscht:

Während bei geringen Tonhöhenabständen der Rhythmus von Bild 2 zu hören ist, hört man bei größeren Tonabständen zwei voneinander unabhängige rhythmische Linien.

Mittels Pitchbend ist es möglich, die Tonhöhe des mittleren Tons der Kippfigur kontinuierlich zu verändern. Es ist dabei erstaunlich, daß bei allen Zuhörern an einer ganz bestimmten Stelle der Höreindruck von Bild 2 in den von Bild 3 übergeht, „umkippt". In vielen empirischen Experimenten mit Studentinnen und Studenten habe ich feststellen müssen, daß es nur in engen Grenzen möglich ist, willentlich den Zeitpunkt, an dem der Kippeffekt eintritt, zu verschieben.

Noorden_Kippfiguren:

Möglichst den „Pitchbend Range" am Synthesizer auf 12 einstellen!

```
FOR B= 0 TO 127
  Bend(1,63)
Play(1,60,90):WAIT .1:Out(1,60,0)
  Bend (2,B)
Play (2,60,90):WAIT.1:Out(2,60,0)
  Bend (1,63)
Play(1,60,90):WAIT .1:Out(1,60,0)
  WAIT.1
NEXT
END
```

Unterprogramm **Bend**

```
Bend(Ch,B) :=
<Out(223+Ch):Out(0):Out(B)>
```

Innerhalb der Schleife befinden sich drei gleichlange Töne und eine Pause. Der mittlere der drei Töne wird mittels Pitchbend allmählich verstimmt. Der unverstimmte Pitchbend-Wert ist 63. Beim Pitchbend-Range 12 liegt der kleinste Wert (B=0) 1 Oktave unter und der größte Wert (B=127) 1 Oktave über diesem Mittelwert – das Programm arbeitet mit zwei MIDI-Kanälen, kann daher für den mittleren Ton auch eine andere Klangfarbe verwenden. Besonders raffiniert und verwirrend ist die ganze Angelegenheit, wenn MIDI-Kanal 1 auf den linken, MIDI-Kanal 2 auf den rechten Stereoausgang gelegt und das Ganze über Kopfhörer abgehört wird. Nun ist die Integrationsfähigkeit der beiden Gehirnhälften gefragt ...

Entwickelnde Variation eines Bass-Riffs nach Arnold Schönberg

```
DIM Ch(12): DIM T(12): DIM V(12)
FOR N=1 TO 12
READ T(N),V(N)
NEXT
DATA 36,100,41,60,43,65,36,0
DATA 42,75,45,90,47,70,38,60
DATA 36,95,43,0,47,75,38,80

WHILE Aktion =0
  FOR N=1 TO 12
    Play(1,T(N),V(N)):WAIT .1:Play(1,T(N),0)
  NEXT
  Variation
  Zeichnen
  X=X+1
WEND
END
```

Unterprogramm **Variation**:

```
Variation:=
R= RND(12)+1
IF R<12
     THEN Ch=Ch(R+1):Ch(R+1)=Ch(R):Ch(R)=Ch
    T=T(R+1):T(R+1)=T(R):T(R)=T
    V=V(R+1):V(R+1)=V(R):V(R)=V
ENDIF
IF R=12
    THEN Ch=Ch(1):Ch(1)=Ch(12):Ch(12)=Ch
    T=T(1):T(1)=T(12):T(12)=T
    V=V(1):V(1)=V(12):V(12)=V
ENDIF
```

Unterprogramm **Zeichnen**

```
Zeichnen: =
PBOX 10+3*X,200-N*V(N)/20+T(N)/10,3,3
```

Zu derartigen algorithmischen Programmen lassen sich beliebig viele grafische Darstellungen entwickeln, die die eigentümliche Schönheit des musikalischen Prozesses verdeutlichen. Das angeführte – zugegebenermaßen durch bloßes Probieren „erfundene" Unterprogramm **Zeichnen** – hat das Strickmuster in Bild 6 erzeugt.

„248" – Der Schlüssel zur Welt der Popmusik!

Der im vorliegenden Aufsatz herausgearbeitete Gegensatz zwischen Komponieren von Musik mit MIDI-Recordingsystemen und algorithmischem Komponieren von BASIC-Programmen ist durch einen kleinen MIDI-Trick versöhnbar: Sendet ein algorithmisches Programm in regelmäßiger Folge die Zahl „248", so kann es mit einem MIDI-Recordingsystem synchronisiert werden. Je nach Voreinstellung „erwartet" ein MIDI-Recordingsystem 48, 96 oder 192 MIDI-Clocks pro Takt. Es rückt also bei 48 etc. MIDI-Clocks genau einen Takt weiter. Das Startsignal für diese Art zeitlicher Synchronisation von algorithmischem Programm und MIDI-Recodingsystem gibt die einmalig gesendet MIDI-Zahl „250". Bei Tempo **MM** (= Schläge pro Minute) ist die Periode der MIDI-Clocks T = 240/(C*MM),

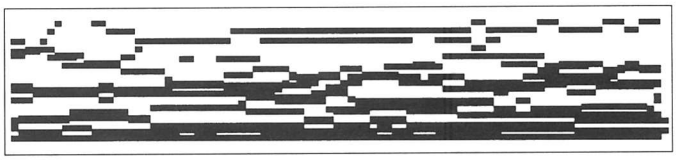


Bild 6: Eine grafische Darstellung der entwickelnden Variation.

wobei C = 48, 96 oder 192 ist. Im Programm muß also alle **T sec** ein „Interrupt" stattfinden, der **Out(248)** aufruft.

Es müssen bei der Synchronisation von algorithmischem Programm und MIDI-Recordingsystem in der Regel zwei Computer oder aber ein Computer (für den Algorithmus) und ein MIDI-File-Player (für die zu synchronisierende Musik) vorhanden sein. Mit dieser Synchronisation können Songs, die als MIDI-File gekauft wurden, oder irgendwelche Kompositionen oder Stimmen, die in MIDI-Recordingsystemen erstellt worden sind, zum algorithmischen Programm mitlaufen. „African Drumming" oder die „Entwickelnde Variation" kann beispielsweise hervorragend in einen einfachen Rock-Groove eingebaut werden. (Näheres in Stroh, 1990, S. 96 ff.)

Prof. Wolfgang Martin Stroh
Universität Oldenburg
FB 2 - Musik/AK
Ammerländer Heerstraße 114-118
26111 Oldenburg

Literatur

Stroh, W. M.: Midi-Exprimente und Algorithmisches Komponieren – Eine Anleitung zum kreativen Programmieren und Komponieren am Computer. Midipädagogische Schriftenreihe, Band 3. Berlin: Musiklabor, 1990.

Stroh, W. M.: Midi-Experimente und Algorithmisches Komponieren – Band 2: Programme und Projekte für den Musikunterricht und die Musikpraxis. Midipädagogische Schriftenreihe, Band 6. Berlin: Musiklabor, 1991.

Stroh, W. M.: Harmonie, Chaos und Computer – Neue Technologien im New Age. In: Kongreßbericht KlangArt '91, hg. von Bernd Enders und Stefan Hanheide. Mainz: Schott, 1993, S. 94-108. Hier werden „Grenzerfahrungen" mit Computermusik diskutiert, insbesondere auch das Noorden-Experiment auf historischem Hintergrund.

Van Noorden, Leo P. A. S.: Temporal Coherence in the Perception of Tone Sequences, Dissertation/Eindhoven 1975. Komplettes Material zu den Kippfiguren-Experimenten.

LOG IN-Service

Die hier besprochenen Programme liegen in Omikron-BASIC für den *Atari ST* und als ASCII-Dateien vor und sind im LOG IN-Service erhältlich (vgl. S. 79).