Wolfgang Martin Stroh, Saarstr. 22, 26121 Oldenburg, wolfgang.stroh@uni-oldenburg.de, www.iii-twiskenstudiuo.de

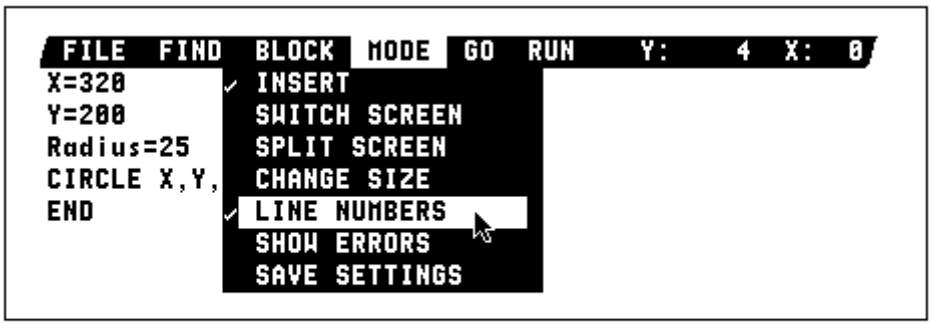
Musikgrafik programmieren mit Basic am Atari ST

Technisches zu „Omikron Basic“

Hier die Checkliste für die Benutzung von Omikron-Basic:

- Programm OMIKRON.PRG" wie üblich starten.
- Wenn das Programm geladen ist, die Taste HELP drücken. Dann ist man in der Arbeitsseite. Dort gibt es eine Menue-Leiste und dort kann man die Programme schreiben.
- Die Programmtexte können wie in jedem Textprogramm geschrieben und verändert werden. Also "Backspace", "Delete", "Insert", Cursor mit Maus verschieben usw.
- Eine geschriebene Textzeile durch Druck auf RETURN in der Arbeitsspeicher geben. Dann zeigt sich, ob der Text in der Sprache "Omikron" richtig gewesen ist. Wenn ja, so wird er bisweilen in der Rechtschreibung etwas korrigiert (z.B. Groß- und Kleinschreibung), wenn nein, dann kommt eine Fehlermeldung. Man muß sich dann den Text nochmals ansehen: man hat irgendetwas falsch gemacht...
- Besonderheit: die Zeile, in der der Cursor ist, wird dadurch gelöscht, daß man Schift + F9 drückt. Komisch!? Aber wahr.
- Ist ein Programm geschrieben und Zeile für Zeile mittels RETURN eingegeben, so wird es mit RUN gestartet. Das geschieht so: Maus auf Wort "RUN" in der Menueleiste, dann klappt ein Menue auf und dort auf das Wort "RUN" klicken. Wenn dann "bad line number" statt eines Startes erscheint:

Ausschnitt der Arbeitsseite von Omikron-Basic:



Wenn bei RUN das Programm nicht läuft, dann ist im Menue MODE die Zeile LINE NUMBER auszuschalten (Haken weg).

- Ist ein Programm abgelaufen, so erscheint "ok" und man gelangt durch Drücken der Taste HELP auf die Arbeitsseite zurück.
- Programme können wie bei Atari gewohnt gespeichert und geladen werden.

Arbeitsschritt 1: Die Maus - unsere Sprühdose

Aufgabe 1

Gib folgende Programmzeile ein:

CIRCLE MOUSEX, MOUSEY, 5

(Zeile schreiben und RETURN drücken...) und starte dies Ein-Zeilen-Programm! Ändere die Zahl "10" ab und schreibe "20", "200", "1", "0" usw.! Formuliere eine Regel!

Erklärung: "CIRCLE MOUSEX, MOUSEY, 5" ist ein Befehl, der den Computer veranlasst, um die Stelle, an der sich der Maus-pfeil befindet, einen Kreis mit dem Radius von 5 Bildpunkten ("Pixeln") zu zeichnen. Da die Maus beim Starten des Programms immer an der "RUN-Stelle" steht (= Mitte oben), ist der Mittelpunkt immer "Mitte oben". HEISSER TIP: die echte Sprydose bekommst Du durch "PCIRCLE " statt "CIRCLE"!

Um die Maus als Pinsel oder Spraydose verwenden zu können, müssen mehrere Kreise nacheinander gezeichnet und muß die Maus dabei bewegt werden können:

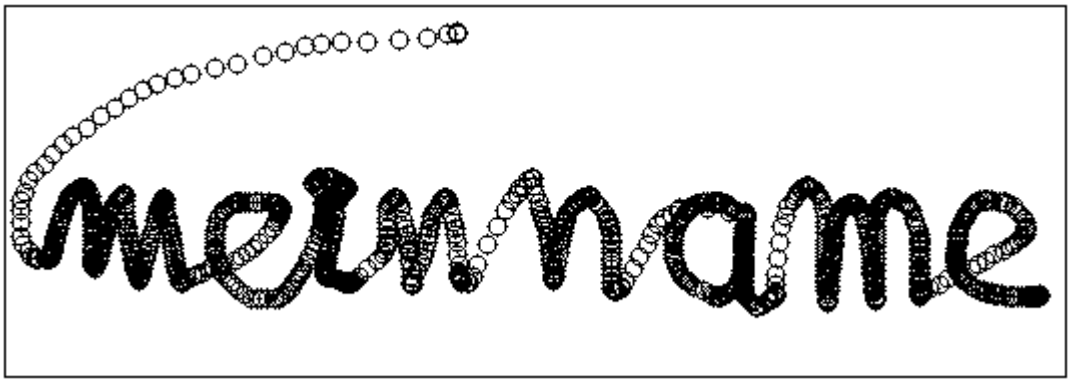
Aufgabe 2

Erweitere die Programm-Zeile von Aufgabe 1 durch zwei Zeilen (eine sog. REPEAT-UNTIL-Schleife)

REPEAT
CIRCLE MOUSEX, MOUSEY, 5
UNTIL LEN(INKEY$)

und starte das Programm! Bewege nun die Maus! Du kannst das Zeichnen dadurch beenden, daß Du auf die SPACE-Taste (die lange Taste in der Mitte) des Keyboards drückst.

Erklärung: Die Zeile, die zwischen "REPEAT" und "UNTIL" steht, wird solange wiederholt (= "repeat"), bis (= "until") das pas-siert, was hinter "UNTIL" steht. Im vorliegenden Fall steht dort "LEN(INKEY$)", was in der Sprache Omikron-Basic bedeutet "eine Keyboardtaste wird gedrückt".



Aufgabe 3

Ersetze im Programm der Aufgabe 2 die Zeile **"CIRCLE MOUSEX, MOUSEY, 5"** durch die Zeile **"IF MOU-SEBUT = 2 THEN CIRCLE MOUSEX, MOUSEY, 5"!**

Zeichne nun wieder! Bewege die Maus und drücke gelegentlich die linke Maustaste! Bewege die Maus in verschiedenen Geschwindigkeiten! Versuche die Regel zu finden, nach der dies veränderte Prog-ramm arbeitet!

Erklärung: "IF MOUSEBUT = 1 THEN..." heißt "wenn (= "if") die linke Maustaste gedrückt ist (= "mousebut =1"), dann (= "then") zeichne den Kreis und sonst nicht.

Die elektronische Sprühdose ist schon fast perfekt. Es fehlt jetzt nur noch die Möglichkeit, auch zu sehen, wohin man mittels Mausklick sprüht! Mit folgender Programmerweiterung zaubern wir den Mauspfeil auf den Bildschirm, der dann stets angibt, wo gesprüht werden kann:

Aufgabe 4

Erweitere das Programm der Aufgabe 3 um zwei Befehle, die die "Sichtbarkeit" des Mauspfeils betreffen:

```
REPEAT
MOUSEON
IF MOUSEBUT = 1 THEN MOUSEOFF : CIRCLE MOUSEX, MOUSEY, 5
UNTIL LEN( INKEY$ )
```

Zeichne nun Bilder, sprühe den Bildschirm voll! Wähle verschiedene Kreisradien (also statt "5" auch andere Zahlen)! Frag Deine Lehrerin/Euren Lehrer, ob eines Deiner Bilder gespeichert und über ein Zeichenprogramm ausgedruckt werden kann!

Laß mal "MOUSEOFF:" in der dritten Zeile weg und zeichne wieder. Welche Bedeutung hat dieser Teil des Befehls?

Arbeitsschritt 2: Der Synthesizer - unser Musikinstrument

Der Computer sollte nun über ein Midi-Kabel mit einem Synthesizer verbunden sein. Wenn ein Computer auf dem Synthesizer "spielen" soll, so muß er an seinem Midi-Ausgang bestimmte Zahlen aussenden, die der Synthesizer empfangen und "verstehen" kann.

Für alles Mögliche gibt es in Omikron-Basic Befehle, die den Computer veranlassen, etwas Bestimmtes zu tun. Bei "CIRCLE MOUSEX, MOUSEY, 5" zeichnet er beispielsweise an der Mausposition einen kleinen Kreis, bei "REPEAT...UNTIL" wiederholt er einen Programmabschnitt solange, bis irgendeine Bedingung erfüllt ist usw. Der Befehl "Schicke die Zahl X an den Midiausgang des Computers" heißt in Omikron-Basic "BIOS(,3,3,X)".

Aufgabe 5

Wähle am Synthesizer einen percussiven Sound, der von selbst lingen aufhört. Gib die Programmzeilen

```
BIOS(,3,3,144)
BIOS(,3,3,60)
BIOS(,3,3,90)
```

ein und starte dies Drei-Zeilen-Programm! Ersetze die Zahl "60" durch andere Zahlen zwischen 0 und 127. Ersetze die Zahl "90" durch Zahlen zwischen 0 und 127! Was passiert? Formuliere eine Regel!

Erklärung: Die erste der drei Midi-Zahlen "144" bedeutet, daß ein sog. "Note ON"-Befehl (= "Ton anschalten") auf Midikanal 1 an den Synthesizer ergeht. Die zweite Zahl "60" sagt, welche Taste des Synthesizers diesen Befehl erhalten soll; dadurch ist die Tonhöhe bestimmt. Die dritte Zahl "90" sagt, wie schnell die Taste gedrückt werden soll (eine Größe, die "Velocity" heißt); diese Geschwindigkeit reguliert die Lautstärke. - Wenn Du einen nicht von selbst ausklingenden Sound wählst, so wird durch das Programm aus Aufgabe 5 der Ton an- aber nicht mehr abgeschaltet (= Notenhänger!).

Die drei untereinander stehenden Zeilen von Aufgabe 5 können auch nebeneinander stehen, wenn sie durch einen Doppelpunkt getrennt sind.

Aufgabe 6

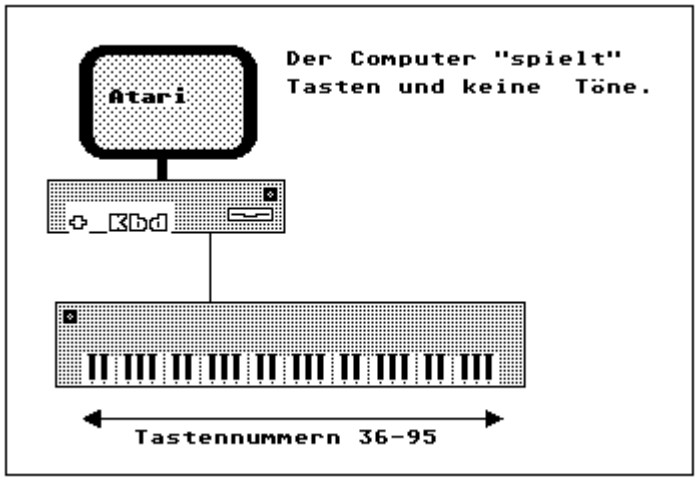
Stelle am Synthesizer einen Sound ein, der nicht von selbst ausklingt. Gib folgende Programmzeilen ein:

```
BIOS(,3,3,144):BIOS(,3,3,60):BIOS(,3,3,90)
WAIT  2.5
BIOS(,3,3,144):BIOS(,3,3,60):BIOS(,3,3,0)
```

und starte dies Programm. Ersetze die Zahl "2.5" hinter "WAIT" durch andere Zahl zum Beispiel "0.2" oder "4"! (Dezimalzahlen mit Punkt und nicht mit Komma schreiben!) Was bedeutet "WAIT" und was passiert in der 3. Zeile? Formuliere eine Regel!

Erklärung: Wenn der Computer an die Zeile mit "WAIT" kommt, so wartet (= "wait") er so viele Sekunden, wie die Zahl hinter "WAIT" besagt. "BIOS(,3,3,0)" setzt die Lautstärke auf Null, schaltet also den Ton ab.

Die mittlere Zahl der drei BIOS-Befehle gibt die Taste des Synthesizers an, die gespielt werden soll. Bei Synthesizern mit 5 Okaven hat die tiefste Taste die Nummer 36, das mittlere C die Nummer 60 und die höchste Taste die Nummer 96. Der Kammerton (das eingestrichene A) hat also die Nummer 69 (9 Halbtöne über dem eingestrichenen C).



Um mehr als einen einzigen Ton zu spielen, muß der Computer die drei Zeilen von Aufgabe 5 mehrfach durchlaufen. Der Programmbefehl, der eine derartige Wiederholung bewirkt, ist mit "REPEAT...UNTIL" aus Aufgabe 2 von Arbeitsschritt 1 bereits bekannt.

Aufgabe 7

Schreibe, indem Du Aufgabe 2 und 6 kombinierst, ein Programm, in der ein Ton von der Dauer einer Zehntel Sekunde solange wiederholt wird, bis die SPACE-Taste gedrückt wird!

Schreibe nochmals das Programm von Aufgabe 4 (Arbeitsschritt 1)! Gib statt der 3. Zeile die drei Zeilen

```
IF MOUSEBUT = 1
THEN MOUSEOFF : CIRCLE MOUSEX, MOUSEY, 5
ENDIF
```

ein und starte das Programm! Der Effekt dieser beiden Schreibweisen ist identisch. Der Vorteil der zunächst umständlicheren Schreibweise ist der, daß nun hinter die IF-Bedingung noch mehr Befehle geschrieben werden können. Zum Beispiel:

Aufgabe 8

Gib im Programm von Aufgabe 4 statt der 3. Zeile die Zeilen

```
IF MOUSEBUT=1
THEN MOUSEOFF: CIRCLE MOUSEX, MOUSEY,5
BIOS(,3,3,144):BIOS(,3,3,60):BIOS(,3,3,90)
WAIT .1
BIOS(,3,3,144):BIOS(,3,3,60):BIOS(,3,3,0)
ENDIF
```

ein und starte das Programm! Bediene die Sprühdose (Mausbewegung, Mausklick)! Was passiert? Wähle auch unterschiedliche "WAIT"-Zeiten und verschiedene Tastennummern.

Erklärung: Immer wenn die linke Maustaste gedrückt wird (MOUSEBUT = 1) wird ein Kreis gezeichnet und sodann wie in Aufgabe 7 ein Ton am Synthesizer gespielt. Leider aber immer derselbe... Dieser Mangel soll demnächst behoben werden!

Arbeitsschritt 3: Der Bildschirm - unser Musizierzeichenbrett

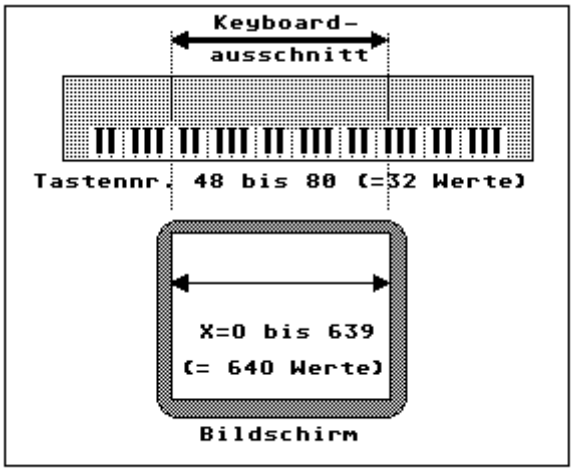
Überlegt Euch mal, wie man das Zeichnen von Arbeitsschritt 1 in Musik umsetzen könnte! Die Lösung von Aufgabe 8 ist etwas monoton: zu jedem gezeichneten Kreis erklingt stets derselbe kurze Ton. Besser ist es schon, wenn zum Beispiel wie beim Keyboard links die tiefen und rechts die hohen Töne wären. Und vielleicht oben die leisen und unten die lauten...

Der Bildschirm hat von links nach rechts 640 Pixel (Bildpunkte) und von oben nach unten 400 Pixel. Diese Tatsache kannst Du konkret überprüfen, indem Du im Programm von Aufgabe 1 (Arbeitsschritt 1) konkrete Zahlen eingibst:

Aufgabe 9

Gib die Programmzeile **"CIRCLE 320, 200, 5 "** ein und starte das Ein-Zeilen-Programm! Ersetze 320 und 200 durch andere Zahlen, zum Beispiel 640, 0, 1000, 10 usw. ! Programmiere einen Kreis um den Punkt (50/50) mit Radius 20! Programmiere den größten vollständigen Kreis! Programmiere den größtmöglichen Halbkreis!

Wenn im Musik-Zeichenprogramm links tiefe und rechts hohe Töne liegen sollen, so müssen den weiter links liegenden Bildpunkte niedrige, den weiter rechts liegenden höhere Tastennummern zugeordnet werden. Da es höchstens 128 verschiedene Tastennummern gibt, kann aber nicht jedem Pixel eine eigne Taste zugeordnet werden. Mehrere Pixel werden zusammengefaßt und jeweils einer Tastennummer zugeordnet:



Es werden der Taste Nr. 48 die X-Werte 0 bis 19, der Taste 49 die X-Werte 20 bis 39, der Taste 50 die X-Werte 40 bis 59 usw. zugeordnet. In Omikron-Basic lautet die entsprechende Zuordnungsformel einfach

Taste = 48 + X/20

(weil X/20 auf die nächstkleine ganze Zahl abgerundet wird).

Zuordnungsformeln wie "Taste = 48 + X/20" kann man in ein Programm der Art von Aufgabe 1 bis 9 so einfügen, daß man anstelle von Zahlen "Variablennamen" schreibt und in einer extra Zeile den Zahlenwert solcher Variablen definiert, z.B.
Taste = 48 + X/20 : Lautst = 90
BIOS(,3,3,144):BIOS(,3,3,Taste):BIOS(,3,3,Lautst)
anstelle von "BIOS(,3,3,144):BIOS(,3,3,48+X/20):BIOS(,3,3,90)".

Aufgabe 10

Gib folgendes Programm (Kombination von Aufgabe 8 mit der "Zurodnungsformel")

```
REPEAT
MOUSEON
IF MOUSEBUT = 1
THEN MOUSEOFF : CIRCLE MOUSEX, MOUSEY, 5
Taste = 48 + MOUSEX/20
BIOS(,3,3,144) : BIOS(,3,3,Taste) : BIOS(,3,3,90)
WAIT .1
BIOS(,3,3,144) : BIOS(,3,3,Taste) : BIOS(,3,3,0)
ENDIF
UNTIL LEN( INKEY$ )
```

ein und starte es! Zeichne mit der Maus und achte auf die Tonhöhenveränderungen! Überlege: Warum steht "Taste = 48 + MOUSEX/20"? Ersetze diese Zeile durch "Taste = MOUSEY/20", durch "60+MOUSEX/20", durch "36 + MOUSEX/12"! Höre auf die Veränderungen der Umsetzung von Grafik in Tonhöhen und erkläre das Ergebnis!

Erklärung: Durch die Veränderung der ersten Zahl in der Zuordnungsformel werden die Tohöhen transponiert. Wird MOUSEY verwendet, so ändert sich die Tonhöhe entlang der Vertikalen anstatt entlang der Horizontalen. Ein anderer Nenner erweitert oder verengt den Tonhöhenumfang. Beachte: es dürfen nur Tonhöhenwerte zwischen 0 und 127 ausgegeben werden.

Aufgabe 11

Schreibe zwischen die 5. und 6. Zeile im Programm der Aufgabe 10 die Zeile

Lautst =MOUSEY/4

und ersetze in der nächsten Zeile "BIOS(,3,3,90)" durch "BIOS(,3,3,Lautst)"! Starte das Programm und höre auf Deine Zeichnung. Erkläre den Effekt!

Was müßte am Musik-Zeichenprogramm von Aufgabe 11 noch verbessert werden? Vorschläge:

1._____

2._____

3._____

Arbeitsschritt 4: Die Zeichenpalette

Die Zeichenpalette von Omikron-Basic sieht folgendermaßen aus:

CIRCLE X,Y,R	Kreis um (X/Y) mit Radius R
ELLIPSE X,Y,A,B	Ellipse um (X/Y) mit Hauptachsen A und B
BOX X,Y TO X',Y'	Rechteck, Ecken li oben (X/Y) und re unten (X'/Y')
PCIRCLE, PELLIPSE, PBOX	jeweils schwarz ausgefülltes Zeichen

Diese Zeichen sollen durch die verschiedenen Möglichkeiten, die Maustaste zu drücken, hervorgebracht werden. Die Bezeichnungen für den Zustand der Maus lauten:

MOUSEBUT = 0	keine Maustaste gedrückt
MOUSEBUT = 1	linke Maustaste gedrückt
MOUSEBUT = 2	rechte Maustaste gedrückt
MOUSEBUT = 3	beide Maustasten gedrückt
MOUSEBUT	linke, rechte oder beide Maustasten gedrückt

Aufgabe 12

Gib das Programm

```
REPEAT
MOUSEON
X = MOUSEX : Y = MOUSEY
IF MOUSEBUT  = 1 THEN  MOUSEOFF :  PCIRCLE X, Y, 5
IF MOUSEBUT  = 2 THEN MOUSEOFF: PBOX X-2, Y-2 TO X+2,Y+2
IF MOUSEBUT  = 3 THEN MOUSEOFF: ELLIPSE X,Y, 24, 16
IF MOUSEBUT
THEN  Taste = 48+ X/20 : Lautst= Y/4
BIOS (,3,3,144):BIOS(,3,3,Taste):BIOS(,3,3,Lautst)
WAIT .1
BIOS(,3,3,144):BIOS(,3,3,Taste):BIOS(,3,3,0)
ENDIF
UNTIL LEN( INKEY$ )
```

ein und starte es. Zeichne, indem Du die linke, rechte und beide Maustasten drückst! Verändere die Größe und Art der Zeichen in den Programmzeilen 4 bis 6, indem Du die verschiedenen Befehle für die

Zeichen der Palette verwendest! Verändere auch die Zahl bei "WAIT" (vor allem auch kürzere Werte, zum Beispiel 0.05). - Mache Dir den Programmablauf klar!

Erklärung: Die Zeilen 4 bis 6 bewirken, daß, je nach Maustaste, unterschiedliche Zeichen ausgegeben werden. Immer, wenn irgendein Zeichen ausgegeben und damit eine Maustaste gedrückt wurde, ist die Bedingung "IF MOUSEBUT" erfüllt. In jedem dieser Fälle wird ein Ton gespielt.

Aufgabe 13

Wähle im folgenden einen percussiven Klang oder - am besten - einen "Drum-Sound". Ersetze die Zeilen 9-12 im Programm von Aufgabe 12 durch folgende Zeilen:

```
IF MOUSEBUT
THEN
Taste1 = 60 + X/20
Taste2 = 36 + (400-Y)/10
BIOS(,3,3,144):BIOS(,3,3,Taste1):BIOS(,3,3,90)
BIOS(,3,3,144):BIOS(,3,3,Taste2):BIOS(,3,3,90)
ENDIF
```

Welchen Effekt hat diese Änderung? Warum ist die X-Richtung für die Ober-, die Y-Richtung für die Unterstimme zuständig?

Erklärung: Jetzt werden die X- und Y-Werte als Tonhöhen interpretiert. Die Ausgabe von zwei BIOS-Zeilen bedeutet Zweistimmigkeit. Die besondere Klammer "(400-Y)" für "Taste2" bewirkt, daß die tieferen Töne unten statt oben im Bildschirm liegen.

Das optimale Zeichenprogramm würde so funktionieren, daß Töne immer solange erklingen, bis entweder die Maustaste losgelassen oder die Maus bewegt wird. Leider ist ein gewisser Programmieraufwand erforderlich, um diese Bedingung zu erfüllen.

Aufgabe 14

Ladet das Omikron-Basic-Programm mit der Bezeichnung "AUFG_14.BAS" von Diskette und startet dies Programm. Es hat die gewünschten Eigenschaften: aufgrund der Zeile mit "IF MOUSEBUT = 0" wird der Ton abgeschaltet, wenn die Maustaste losgelassen wird. Wird die gedrückte Maus bewegt, so tritt ein komplizierter Mechanismus in Kraft: Im Unterprogramm "Note" wird untersucht, ob der neue X-Wert zu einer neuen Taste führen würde oder nicht. Nur wenn eine neue Taste notwendig wäre, so wird die alte Taste ab- und die neue angeschaltet (Unterprogramme "Note_Off" und "Note_On").

In diesem Programm stehen an einigen Stellen Worte wie "Note", "Note_On" und "Note_Off". Immer, wenn der Computer auf eines dieser Worte trifft, sieht er weiter unten nach, ob es eine Zeile mit "DEF PROC" gibt, in der dies Wort vorkommt. Er schiebt dann alles, was zwischen "DEF PROC" und "RETURN" steht dort ins Programm ein, wo er das Wort zuerst angetroffen hat. - Man nennt die eingeschobenen Programmteilke "Propceduren" oder einfach "Unterprogramme". Vorteil: dasselbe Unterprogramm kann von mehreren Stellen des Hauptprogramms aus "angewählt" werden.

Erweitere dies Programm durch zusätzliche Zeichen, die mit der rechten Maustaste bzw. mit beiden Maustasten gezeichnet werden können wie in Aufgabe 12!

Arbeitsschritt 5: Fraktale Musik und Grafik der Molekularbewegung

Ein Staubmolekül in der Luft vollführt die "Brown'sche Molekularbewegung". Es wir in einer zufallsbedingten Reihenfolge mal in die eine, mal in die andre Richtung gestoßen. Wenn man den Weg eines Moleküls aufzeichnen könnte, so würde man eine wundersame "fraktale" Grafik erhalten. Dieser Vorgang soll mit den folgenden Aufgaben nachgeahmt werden. Im Unterschied zu allen bisherigen Aufgaben zeichnet diesmal der Computer selbst und nicht die Maus bzw. die die Maus führende Hand.

Wenn Du in einem Programm die Zeile R = RND(7) schreibst, so wird der Computer, sobald er diese Zeile liest, eine Zufallszahl zwischen 0 und 6 auswählen. Möchte man also beispielsweise Zufallszahlen zwischen -2 und +2, so muß man R = RND(5)-2 schreiben. - Im folgenden wird noch ein neuer Grafikbefehl verwendet: "DRAW X,Y". Mit diesem Befehl wird an der Stelle (X/Y) ein Punkt (Pixel) gezeichnet.

Aufgabe 15

Gib das Programm

```
X = 320 : Y = 200
REPEAT
X= X + RND(5) - 2
Y= Y + RND(5) - 2
DRAW  X, Y
UNTIL LEN ( INKEY$ )
```

ein und starte es! Starte es mehrfach! Ändere die Werte "320" und "200" in der ersten Zeile! Ändere die Zahl "5" im Argument von "RND( )"! Versuche das Ergebnis zu erklären!

Erklärung: In der 1. Zeile werden die sog. "Anfangswerte" gesetzt, d.h. die Stelle, an der das Programm zu zeichnen beginnt. In der 3. Zeile passieren drei Dinge: zunächst wählt der Computer eine Zufallszahl zwischen 0 und 4 aus und zieht von dieser Zahl 2 ab (sodaß eine Zufallszahl zwischen -2 und +2 resultiert), dann zählt er diese Zahl zu der hinzu, die auf dem Speicherplatz, der den Namen "X" trägt, gespeichert ist, und schließlich legt er das Ergebnis dieser Addition wieder auf dem Speicherplatz mit dem Namen "X" ab. [Bemerke: als mathematische Gleichung ist X = X+2 oder dgl. absolut unsinnig, nicht jedoch als Befehl!] Im Endeffekt hat der Computer zufallsbedingt einen X-Wert "ganz in der Nähe" des alten X-Werts ermittelt. In der übernächsten Zeile wird dann ein entsprechender Zufallspunkt gezeichnet. Mit dem zufälligen Y zusammen ergibt sich eine "Zufallsbahnkurve". - Der mögliche Wertebereich in der 3. und/oder 4. Zeile bestimmt, wie dicht oder locker das fraktale Gesamtbild ist, das sich aus der Brown'schen Molekularbewegung ergibt.

Laß Dir Zeit! Beobachte - und warte auf besonders schöne Bilder, skurille Landschaften, wolkige Traumwelten und verknüllte Urwälder! Um eine gewisse "Perspektive" zu erhalten, solltest Du die folgenden Werte ausprobieren:

X = X + RND(7) - 3 : Y = Y + RND(3) - 1
X = X + RND(9) - 5 : Y = Y + RND(5) - 3

Woher kommt die "Perspektive"? Wie entsteht "Dichte" oder "Luftigkeit"?

Das Bild links ist mit den Werten X= X + RND(3) - 1 und Y = Y+ RND(3) - 1 entstanden, d.h. durch eine dichte Zufallsbewegung, bei der der jeweils nächste Bildpunkt maximal 1 Pixel vom alten entfernt ist. Im Bild links unten ist der Zufallsspielraum größer (+/- 2 Pixel Abstand). Das Bild wird lockerer. Im Bild rechts unten ist für die X-Richtung ein größerer, für Y ein kleinerer Spielraum gewählt: X = X + RND(7) - 3 und Y = Y + RND(3) - 1. Dadurch entsteht eine Art Tiefendimension.

Die Verbindung dieser fraktalen Grafik mit Musik ist insofern etwas schwierig, als einigermaßen gute Grafiken sehr viele Bildpunkte (z.B. 50 000 im Fall der drei abgebildeten Grafiken) erfordern und daher

bei einem Musikstück von nur wenigen Stunden Dauer nicht jedem Bildpunkt ein Ton zugeordnet werden kann. Im folgenden Programm, das sich an die Idee des Programms von Aufgabe 14 anlehnt, sind Tasten-Werte zwischen 36 und 103 in Y-Richtung ausgebreitet und es wird immer nur bei solchen Bildpunkten ein Ton gespielt, bei denen sich dieser Ton vom vorigen unterscheidet.

Aufgabe 16

Wähle (zunächst) einen percussiven Sound. Gib im Programm von Aufgabe 15 zwischen der vorletzten und letzten Zeile (d.h. unmittelbar nach "DRAW X,Y") die Zeilen

```
Taste = 36 + (400 - Y)/6
IF Taste_Alt <> Taste
THEN  BIOS(,3,3,144):BIOS(,3,3,Taste):BIOS(,3,3,90)
Taste_Alt = Taste
ENDIF
```

ein und starte das Programm. Ersetze die erste Zeile durch "Taste = 36 + X/10"! Was ändert sich? Welche Töne werden wiederholt? Ersetz den Nenner "6" in der 1. Zeile durch einen größeren Wert! Was ändert sich?

Erklärung: Der "Trick" dieses Programms ist, daß sich der Computer die zuletzt gespielte Taste unter der Bezeichnung "Taste_Alt" merkt (vorletzte Zeile!). Wenn dann das (schnelle) Zeichenprogramme stets neue Tasten-Werte ermittelt (1. Zeile), so wird nur dann ein Ton gespielt, wenn "Taste_Alt <> Taste" ist, was bedeutet, daß die neu ermittelte Taste NICHT gleich (Bezeichnung "<>") der alten Tasten ist. Eine Änderung des Nenners in der 1. Zeile verändert die Anzahl der Tasten, die gespielt werden und somit auch das Tempo, weil bei weniger Tasten mehr Bildpunkte gezeichnet werden, bis eine neue Taste dran kommt.-- Wenn nicht-percussive Sounds gewählt werden, so muß die alte Taste auch noch ausgeschaltet werden. Dies ist im Programm "AUFG_16.BAS" der Fall, das sich auf Diskette befindet. Eine 2stimmige Version als "AUFG_16A.BAS".

Arbeitsschritt 6: Ornamente und Muster zeichnen und hören

Computergrafiken werden durch Programme und "Algorithmen" erzeugt. Ein Algorithmus ist eine Rechenvorschrift (zum Beispiel X = Y + 3*X). 1986 hat ein australischer Tüftler einen Algorithmus gefunden, der sehr schöne Ornamente und Muster zeichnet. In der Zeitschrift "Spektrum der Wissenschaft" wurde er von A. K. Dewdney 1986 "Hüpfer-Algorithmus" genannt. Die Rechenvorschrift selbst kann man leicht abschreiben, "verstehen" tut sie bis heute niemand vollständig...

Aufgabe 17

Gib das Programm

```
DEFSNG "A-Z"
A = 3000 : B = 0.31 : C = -999
Xo = 300 : Yo = 175
REPEAT
X = Yy - SGN(Xx)*((B*Xx - C)) MOD 64000
Y = A - Xx MOD 40000
Xx = X : Yy = Y
DRAW Xo + X/100, Yo + Y/110
UNTIL LEN( INKEY$ )
```

ein und starte es! Verändere die Anfangswerte "3000", "0.31" und "-999" in der 1. Zeile!Laß Dir viel Zeit und experimentiere mit allen möglichen Werten.

Interessante Anfangswerte für das Programm von Aufgabe 17 (und 18):

A = 5000, B = 0.1, C = 5025
A = 5000, B = 0.2, C = 5025
A = 10000, B = - 0.01, C = - 10000
A = 123, B = 0.456, C = 7890
A = 6000, B = 0.02, C = 6000
A = 7010, B = - 0.3, C = 4000

Erläuterung: Das Programm errechnet in Zeile 5 und 6 und zeichnet in Zeile 8 in rascher Abfolge Bildpunkte. Die Koordinaten X und Y des jeweils neuen Bildpunkts werden aus den Koordinaten Xx und Yy des vorangegangenen Bildpunktes mit einer Formel errechnet, in der noch drei weitere Werte, A, B und C, eine Rolle spielen. "SGN(Xx)" heißt soviel wie "Vorzeichen von Xx". "MOD 64000" bewirkt, daß die Grenzen des Bildschirms eingehalten werden. Der Ausdruck in der ersten Zeile bewirkt in Omikron-Basic, daß alle vorkommenden Zahlen ca. 9 Stellen hinterm Komma genau berechnet werden.

Würde man jeden Bildpunkt vertonen, so würde das entstehende Musikstück bei einer Tondauer von einer Zehntel Sekunde 142 Minuten dauern. Um die Vertonung abzukürzen, ist in Aufgabe 18 "nur" jeder 30. Bildpunkt vertont worden.

Aufgabe 18

Füge im Programm der Aufgabe 17 hinter die vorletzte "DRAW-Zeile" die Zeilen

```
IF N MOD 30 = 0
THEN Taste = 36 + X/10 BIOS(,3,3,144):BIOS(,3,3,Taste):BIOS(,3,3,90)
ENDIF
N=N+1
```

ein und wähle einen percussiven Sound. Dies Programm "zählt" in der Zeile "N = N+1" die Anzahl der Bildpunkte und die Bedingung "IF N MOD 30 = 0" ist immer dann erfüllt, wenn N durch 30 teilbar, d.h. ein Vielfaches von 30 ist. Endeffekt: Jeder 30. Bildpunkt wird "vertont". Ändere die Werte hinter "Taste"! Ändere die Zahl bei "MOD" und beobachte Musik- und Grafiktempo.

Aufgabe 19

Schreibe an den Anfang des Programms die Zeile

```
ON HELP GOSUB Loeschen
```

und ans Ende des Programms die Zeilen

```
- Loeschen
CLS
RETURN
```

und starte das Programm. Drücke nun bei laufendem Programm gelegentlich auf die HELP-Taste! Beobachte und Beschreibe, was passiert!

Erklärung: Beim Drücken der HELP-Taste wird der Bildschirm gelöscht. Dadurch sieht man, wo das Programm gerade zeichnet. (Oft fallen ja neue Punkte auf alte, bereits gezeichnete, sodaß man nichts Neues sieht.) - Der Befehl "CLS" löscht den Bildschirm. "ON HELP GOSUB Loeschen" bewirkt, daß bei Drücken der HELP-Taste das Programm unterbrochen wird und der Computer in die Zeile "-Loeschen" springt und aus der Zeile "RETURN" dann wieder zurück an eben die Stelle des Programms, wo er es verlassen hat.

Arbeitsschritt 7: Mauszeichnen von Computergrafiken

Auf den Arbeitsblättern 1 bis 4 hast Du mit der Spraydose "Computermaus" Grafiken und Musik gezeichnet. Die Programme von Arbeitsschritt 5 und 6 waren Programme für Computergrafik, die "automatisch" abliefen und nicht mehr von Hand beeinflußt werden konnten. Reizvoll sind natürlich Kombinationen, die allerdings höheren Proammieraufwand erfordern als die bisherigen Programme. Drei Beispiele sollen aber doch "das Prinzip" zeigen.

Die Idee des Programms der folgenden Aufgabe ist es, daß Du zwar wie auf Arbeitsschritt 4 Musik und Grafik mit der Maus zeichnest, der Pinsel aber eine gewisse computergesteuerte ("algorithmische") Eigenaktivität hat. Immer dann, wenn Du mit einem Mausklick ein Zeichen setzen willst - wie auf den Arbeitsblättern 1 bis 4 -, dann zeichnet ein Mini-Computerprogramm mehr als nur ein einziges Zeichen:

Aufgabe 20

Gib das folgende Programm ein

```
REPEAT
MOUSEON
IF MOUSEBUT = 1
THEN REPEAT
Xo = MOUSEX : Yo = MOUSEY
A=RND(20) - 10 : B = RND(20) - 10 : R = RND(20)
MOUSEOFF: CIRCLE  Xo + 3*A, Yo + 3*B, R+5
UNTIL MOUSEBUT = 0
ENDIF
UNTIL LEN( INKEY$ )
```

und starte es. Drücke die linke Maustaste und bewege die Maus. Was passiert? Verändere die Zahl 20 im Argument von "RND( )"! Wähle hinter "CIRCLE" statt "3" auch andere Faktoren vor A und B! Beschreibe und erkläre den Effekt. Verwende auch andere Zeichen (PCIRCLE, BOX, DRAW, ELLIPSE)! Erweitere das Programm im Sinne von Aufgabe 12 durch Zeichen-Ideen, die mit der rechten etc. Maus hervorgerufen werden!

Erklärung: Anstelle eines Zeichens (wie in den Programmen bis zu Aufgabe 12), werden hier nun viele Zeichen gezeichnet. Diese Verfielfachung geschieht in einer zweiten REPEAT - UNTIL-Schleife, die dann verlassen wird, wenn die Maus losgelassen wird (MOUSEBUT = 0). Immer wenn "MOUSEBUT = 1" gilt, beginnt diese "innere Schleife" zu laufen und zu zeichnen.

Aufgabe 21

Füge in das Programm von Aufgabe 20 direkt vor "MOUSEBUT = 0" die Zeilen

```
Taste = 36 + (Xo + 3*A)/10
BIOS (,3,3,144):BIOS(,3,3,Taste):BIOS(,3,3,90)
WAIT 0.02
```

ein, wähle percussive Sounds und starte das Programm. Wie funktioniert die Vertonung? Ersetze hinter "WAIT" die Zahl "0.02" durch "R*0.1". Was passiert?

Erklärung zu Aufgabe 21: Die Mittelpunkte der vielen Kreise werden wieder in das Rasterc der Tastennumern entlang der X-Achse gedrängt. Wenn hinter "WAIT" die Zeitdauer "R*0.1" steht, so dauern die von den großen Kreisen erzeugten Töne länger als die von den kleinen erzeugten. - Das Programm "AUFG_21.BAS" enthält eine Erweiterung, die es ermöglicht, auch nicht-percussive Sounds zu spielen.