Einführung in die Programmierung automatischer CAD-CAM-Prozesse

Gliederung:

Allgemeine Einführung

Einführung zu Kommandos und Stack

- Vorbemerkung
- Erste praktische Schritte mit dem "beep"-Kommando

Vom Geometrie-Kommando zur "Knüttel-Programmierung"

- Das erste Geometrie-Kommando "cyl3d"
- Kopplung von Mathematik und Kommandos - Substitution
- Nutzung von Variablen
- Prinzip der Punktselektion
- Standard-Variable und Systemzustände

Allgemeine Einführung

Heutzutage wird ein Computer im Büroalltag immer noch wesentlich durch Aktivitäten und Initiativen eines Bedieners genutzt. Eine einzelne Person kann natürlich jetzt sehr viel mehr Aufgaben erledigen als zu Zeiten der Schreibmaschine, der Briefpost und des Wählscheibentelefons, da Flexibilität, Geschwindigkeit und Exaktheit enorm zugenommen haben. Zu verdanken ist dies der rasanten Entwicklung von Hardware und Software. Von sich aus macht der Computer im Allgemeinen jedoch nichts. Er braucht immer individuelle Bedienung. Automatische Abläufe wären wünschenswert, sind aber ohne spezielle Programmierung nicht realisierbar. Erschwerend kommt hinzu, dass Software-Spezialisten meistens rar und teuer sind.

Um eine solche Hürde zu umgehen, wird in diesem Kapitel die "do-it-yourself"-Methode angeregt, Handhabungsschritte, die zuvor manuell initiiert wurden, mittels Software selbst zu automatisieren. Das kann natürlich nur gelingen, wenn eine Programmiersprache zur Verfügung steht, die einfach erlernbar ist und die die grundlegenden Funktionen eines Computers steuern kann.

Da bei der Automatisierung im technischen Umfeld aber sehr häufig auch Grafiken, Visualisierung, Geometriedaten, Mathematik u.a. benötigt werden, stoßen viele Programmiersprachen schnell an ihre Grenzen oder werden zu kompliziert. Für **Pictures by PC** gilt das dagegen nicht.

Vor mehr als 35 Jahren schon hat sich **SCHOTT SYSTEME** obigem, anspruchsvollen Themenkomplex gewidmet und in seine CAD-CAM-Software eine leistungsfähige,

interpretative, grafische Programmsprache integriert. Mit ihr lassen sich im objekt-orientierten Windows-Umfeld erstaunliche Automatisierungsaufgaben flexibel und schnell lösen.

Bereits mit einfachsten Mitteln und wenigen Software-Sequenzen lassen sich viele trickreiche Aufgabenstellungen automatisieren.

Praktischen Ideen dazu sollen an dieser Stelle angeregt werden Für erste Automatisierungsschritte werden lediglich grundlegende **Pictures**-Kenntnisse und logisches Denkvermögen benötigt. Bei einem typischen **SCHOTT SYSTEME**-Anwender sind dafür also alle grundlegenden Voraussetzungen erfüllt.

Einführung zu Kommandos und Stack

Vorbemerkung

Die Vermittlung aller Konstruktions-, Kommando- und Programmier-Kenntnisse würde hier natürlich jeden Rahmen sprengen und Jahre an Ausbildung erfordern. Sinn dieser Lektion ist lediglich, Ideen anzuregen und pragmatische Ansätze für eine autodidaktische "Erforschung" aller Möglichkeiten zu schaffen (ein **Pictures**-Programmier-Seminar bleibt empfohlen).

Die folgenden pragmatischen Ansätze werden "echte Softwerker" eher wenig erbauen, gilt hier doch das Motto: "Im selbstdefinierten Umfeld heiligt der Zweck die Mittel". Hauptsache ist, mit solchen "Programm-Knütteln" schnell hilfreiche Lösungen zu schaffen.

Nicht zu vergessen: wenn Sie eine gute Idee verfolgen, kann Ihnen unser Support mit dem einen oder anderen nützlichen Tipp den Weg zur Lösung sicherlich erleichtern.

Eine geraume Zahl von Kunden hat ihre Arbeitsabläufe auf diese Weise schon effektiver gestaltet und automatisiert.

Selbstverständlich lassen sich mit der Programmierung komplette, eigenständige Branchenlösungen (z.B. Orthopädie, Dental, Prozesssteuerung etc.) erstellen. Dazu sind aber entsprechende Programmierkenntnisse vonnöten.

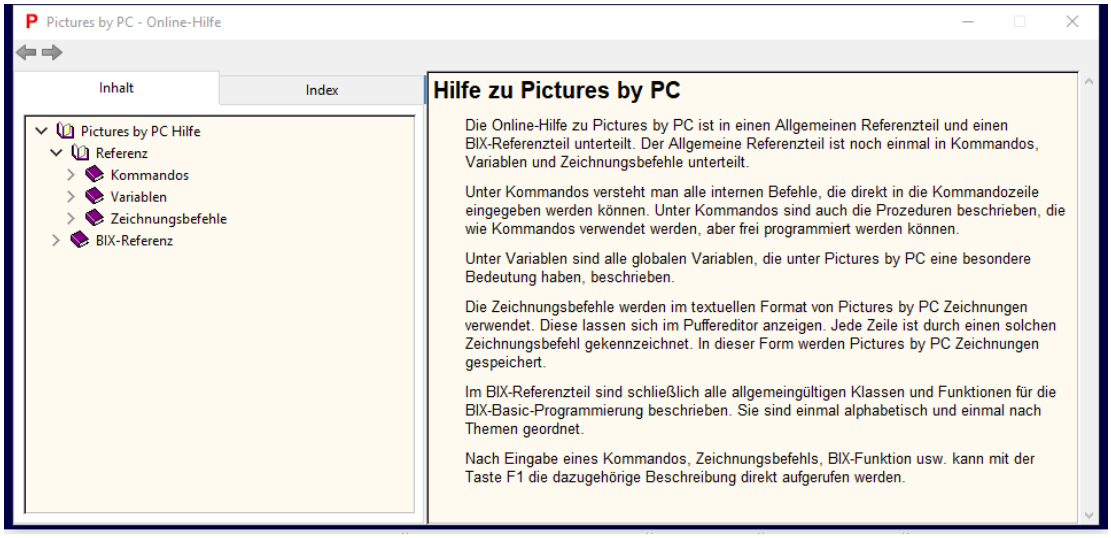
Hier wird folgend mit den allereinfachsten Mitteln und Methoden gestartet.

Leitfaden für die ersten Schritte ist die Datei "NuetzlicheKommandos.stk" im **Pictures**-Unterverzeichnis "procs\samples", die zur aktuellen 3.8-Standard-Auslieferung gehört und die eine Reihe nützlicher Effekte zeigt, die lediglich aus einigen aneinandergereihten Kommandos besteht. Entstanden sind solche Sequenzen durch das praktische Austesten einzelner Kommandos und Sicherung des Kommando-Stapelspeichers (Stack) in eine Datei mit "csave" (vgl. auch "proc", "cload"). Weiter unten in dieser Lektion wird darauf etwas genauer eingegangen.

Ein Benutzer, der **Pictures** gerade kennenlernt und ausschließlich die Menüumgebung nutzt, wundert sich vermutlich, warum ständig das Kommandofenster eingeblendet ist. Wie nützlich das ist, soll im Folgenden bewiesen werden.

Der Benutzer kann bei entsprechender Kenntnis Kommandos oder Kommandofolgen direkt in die Kommandozeile eingeben, so dass sie sofort ausgeführt werden können (auch per Button-Menü abgerufene Kommandos werden in dieser Zeile aufgelistet). Alle während des **Pictures**-Programmlaufs ausgeführten Kommandos werden einmalig mit protokolliert, können so per Pfeiltasten wieder abgerufen und in eine Datei (Stack-Datei, *.stk) als Textdaten gespeichert werden.

Im Kommandofenster kann man, z.B. mit der "Pfeil-nach-oben"-Taste, letzte Kommandos wiederholen, ohne Kenntnis der Kommandowelt aber wenig Nutzen daraus ziehen. Zum Automatisieren sind zumindest Grundkenntnisse zu den Kommandos notwendig. Das ist ein komplexes, wenngleich auch logisch geregeltes Umfeld. Die Grundlagen sind jedoch leicht zu erlernen. Öffnet man im Hilfe-Dialog links ((Funktionstaste F1) das 1. Untermenü, erscheinen die Punkte "Referenz" bzw. "BIX-Referenz".



Momentan soll der Blick nur auf Kommandos gerichtet werden. D.h. Sie wählen in der Hilfe "Referenz" und finden unter "Kommandos" diese alphabetisch aufgelistet. aber ohne zusätzliche Hinweise würde es momentan keinen großen Sinn machen, hier weiter zu blättern. Minimieren oder schließen Sie die Hilfe jetzt.

Lassen Sie uns zunächst die ersten Kommandos praktisch erproben.

Erste praktische Schritte mit dem "beep"-Kommando

Begonnen werden soll mit etwas für einen CAD-Konstrukteur besonders Speziellem, das er in seiner praktischen Arbeit vermutlich noch nie gebraucht hat.

So sollen mit **Pictures** ausnahmsweise einmal Töne (ggf. auch Musik und Sprache) erzeugt werden. Natürlich muss unter Windows zuvor der Lautsprecher eingeschaltet und die Lautstärke geregelt worden sein.

Töne werden unter **Pictures** mit dem (sehr einfachen) Kommando "beep" (Piep) erzeugt. Dieses Kommando wurde gerade deshalb ausgewählt, weil es besonders simpel in der Handhabung ist. Geben Sie folgende Buchstabenfolge in die Kommandozeile ein

```
beep
```

und drücken nach dem letzten Buchstaben die F1-Taste. Sie "landen" unmittelbar in der dem Kommando zugeordneten Hilfe. Dort haben Sie nun zu entscheiden, ob Sie die "Anweisung-BEEP" oder das Kommando "BEEP" studieren wollen. Vorerst nutzen Sie immer erst die Kommandos, Anweisungen werden zunächst zurückgestellt, da sie sich auf die BIX-Programmierung beziehen. (Übrigens: BIX ist die **Schott-Systeme**-eigene, objektorientierte BASIC-Interpreter-Programmiersprache, die Bestandteil des **Pictures**-Standard-Auslieferungsstandes ist.)

Wählen Sie also im Hilfe-Dialog "Das Kommando BEEP" und betätigen "Anzeigen" - so erhalten Sie die Kommando-Beschreibung.

In diesem Fall ist das Format sehr überschaubar. Es enthält nur das englisch-sprachige Schlüsselwort "beep" (also ohne weitere Parameter, wie sie allgemein üblich sind (s.u.)).

Minimieren Sie das Hilfe-Fenster, klicken Sie mit der Maus ins **Pictures**-Kommandofenster (alternativ die Tasten <Alt>+<Y> drücken), tippen die Buchstabenfolge "beep" ein und schließen mit der <ENTER>-Taste ab. Dann sollten Sie einen Ton hören. Das ist zwar nicht sehr beeindruckend, nun aber Ihr erstes, direkt eingetipptes Kommando. Bei Kommando-Eingaben ist übrigens sowohl Groß- als auch Kleinschreibung zulässig. Gewöhnlich wird Letztere genutzt.

Aktivieren Sie erneut die Hilfe. Sie sehen drei Optionen: Wunschparameter, die mit einem Minuszeichen, einem Kennbuchstaben und dem entsprechenden Parameter eingegeben werden. Für Sie sind zunächst die Parameter "-f" (für freqency, Frequenz also Tonhöhe) und "-d" (für duration, Dauer) interessant. Also geben Sie

```
beep -f440 -d1
```

ein und schließen mit <ENTER>-Taste ab. Jetzt hören Sie für 1 Sekunde den Kammerton "A" (440 Hz), mit dem Sie beispielsweise auch ihre Musikinstrumente stimmen könnten.

Somit haben Sie also ein einzelnes, "komplettes" Kommando mit Wunschparametern (Optionen) angegeben.

Wichtig zu erwähnen sei, dass Leerzeichen jeweils Kommando, Optionen und Parameter trennen, das Minuszeichen aber unmittelbar vor dem Options-Kennbuchstaben stehen muss. Die Reihenfolge der Optionen (mit den zugeordneten Parametern) ist in der Kommandozeile ansonsten beliebig.

Hinweis: In dieser Lektion werden noch einige komplexere Kommandos folgen, die in die Kommandozeile einzutippen sind. Sofern Ihnen das Dokument im PDF-Format vorliegt, können Sie sich die Arbeit erleichtern, indem Sie im PDF-Dokument das gewünschte Kommando (mit dem Textmarker) markieren, den markierten Text mit <Strg>+<C> in das Clipboard kopieren und dann in die **Pictures**-Kommandozeile mit <Strg>+<V> einfügen. So vermeiden Sie Eingabefehler und sind schneller. Das Kommando schließen Sie wie üblich mit der <Enter>-Taste ab.

Folgend wird anstelle des obigen Einzelkommandos ein Mehrfachkommando generiert, mit dem zwei verschiedene Töne erzeugt werden. Allerdings erübrigt sich das erneute Eintippen des soeben eingegebenen Kommandos.

Mit Abschluss des obigen Kommandos wurde dieses ja automatisch in den Kommando-Stapel-Speicher (Stack) übernommen und kann mit der "Pfeil-nach-oben"-Taste wieder aufgerufen werden.

Mit einmaligem Betätigen der "Pfeil-nach-oben"-Taste holen Sie das letzte Kommando "beep -f 440 -d 1" wieder in die Kommandozeile zurück und ergänzen es am Zeilenende (Semikolon nicht vergessen) zu

```
beep -f440 -d1 ; beep -f590 -d1
```

Wenn Sie mit <ENTER> abschließen, hören sind jetzt zwei Töne, die einem Martinshorns der Polizei ähneln.

Hier wurden also zwei Kommandos hintereinander, getrennt durch Leerzeichen und Semikolon, in die Kommandozeile eingegeben. Das wird als Mehrfach-Kommando bezeichnet. Solche Mehrfach-Kommandos können sehr komplex sein und mehrere hundert Zeichen umfassen (s.u.)

Diese beiden Kommandos können noch mehrfach ausgeführt werden, dazu ist die Prozedur "Mult.prc" (Erklärung später) geeignet. Holen Sie Ihr Mehrfach-Kommando wieder aus dem Kommandostack ("Pfeil-nach-oben"-Taste) und ergänzen es wie folgt:

```
mult 5 "beep -f440 -d1 ; beep -f590 -d1"
```

Die zwei beep-Kommandos werden durch Anführungszeichen umschlossen, damit sie trotz der darin enthaltenen Leerzeichen als ein Ausdruck (Kommando) für "mult" interpretiert werden. Die Zahl 5 stellt die Anzahl der Wiederholungen für "mult" dar.

Die Tonfolge erklingt jetzt fünfmal. Wenn Sie die Tondauer etwas kürzer wählen, klingt das gleich dem Martinshorn. Sofern bei der Wiederholzahl (oben 5) versehentlich eine höhere Zahl angegeben wurde, könnte das etwas "nervig" werden. Um dann den "Mult"-Wiederholvorgang vorzeitig abzubrechen, mehrmals gleichzeitig die <Strg>+<Pause>-Tasten drücken. Damit können zeitintensive Kommandos und Programme (sofern sie unterbrechbar sind) gestoppt werden.

Fazit: Sie wissen jetzt also, wie man Kommandos in der Kommandozeile ausführt.

"Spielen" Sie zur Übung selbst einfach noch ein wenig mit den Parametern für Tonhöhe und -dauer herum.

Als Nachtrag zum Kommandostack sei noch erwähnt:

Das Blättern in Richtung älterer Kommandos mit der "Pfeil-nach-oben"-Taste kennen Sie ja schon.

Entsprechend können Sie mit der "Pfeil-nach-unten"-Taste wieder in Richtung neuerer Kommandos blättern.

Das "Suchen" im (ggf. umfangreichen) Kommandostack ist ebenfalls hilfreich. Tippen Sie die gesuchte Buchstabenfolge, hier z.B. "mu", ein und drücken die <TAB>-Taste (Tabulator-Taste), so wird die "mult"-Kommandozeile angezeigt.

Ergänzend sollte noch erwähnt werden, dass der Kommandostack mit "csave dateiname" gesichert und mit "cload dateiname" wieder geladen wird. Praktische Beispiele folgen weiter unten.

Wenngleich "beep" als Kommando sicherlich ein "Exote" bleiben wird, sollten Sie dennoch "erfahren", wie mit "beep" Sound-Dateien abzurufen sind. Unter Windows muss dazu ein Mediaplayer installiert sein (ist Windows Standard), der zum "Abspielen" genutzt wird. Außerdem könnte z.B. Ihre eigene Sprache ausgeben werden, sofern diese zuvor mit dem Sprachrekorder aufgenommen würde.

Zur Demonstration tippen Sie folgendes Kommando ein:

```
beep -s "\\!windows\media\tada.wav"
```

Somit wird der bekannte Windows-Sound "Tada.wav" über den Mediaplayer abgespielt. Bei älteren Windows-Versionen (<Win10) ist ggf. hinterher der Mediaplayer wieder zu schließen, das geschieht nämlich nicht automatisch.

Mit dem obigen Kommando lassen sich alle Sounddateien Ihrer "Hits" abspielen, die auf Ihrem Computer vorhanden sind, indem Sie im obigen Kommando den entsprechenden Titel der Wave-Datei und das richtige Verzeichnis eingeben.

Das "harmlose" beep-Kommando wird jetzt verlassen, nicht ohne allerdings zu erwähnen, dass es auch eine BIX-Anweisung "beep.bix" gibt, bei der sich entsprechend den Basic-Regeln aber Syntax und Parameter unterscheiden z.B.,

```
bix beep 440,1000
```

(mit der Zeit hier in Millisekunden)

oder

```
bix beep "\\!windows\media\tada.wav"
```

Übrigens:

In einer Kommandozeile können Kommandos und Basic-Anweisungen durchaus gemixt werden, z.B. funktioniert auch

```
mult 5 "beep -f440 -d1 ; bix beep 590, 1000"
```

Vom Geometrie-Kommando zur "Knüttel-Programmierung"

Das erste Geometrie-Kommando "cyl3d"

Lassen Sie uns nun ein konkreteres Kommando zur Geometrieerzeugung ausprobieren. Es soll als Beispiel ein Zylinder auf dem Koordinaten-Nullpunkt mit einem Radius von 20 und einer Höhe von 100 erzeugt werden. Ein dazu geeignetes Kommando wäre z.B. "cyl3d". Geben Sie diese Buchstabenfolge in die Kommandozeile ein und drücken die <F1>-Taste. Sie erkennen, dass das Kommando einen Parameter "object" benötigt und eine Reihe von Optionen angeboten wird. Wenn das Kommando gleich mit <ENTER> abgeschlossen wird, erhalten Sie eine Fehlermeldung, dass die Angabe des Parameters "object" nicht optional ist, sondern explizit angegeben werden muss.

Für den Parameter "object", der den Identifikationsnamen des zu erzeugenden Zeichnungsobjekt benennt, wird sehr häufig der Stern "*" angegeben. Wie dem **Pictures** Anwender ja bekannt, vergibt **Pictures** den Namen "obj", gefolgt von einer hochzählenden Zahl, also z.B. "obj13", sofern "obj12" schon existiert. Ist der aktuelle Puffer leer, würde man annehmen, dass mit der Ausführung des Kommandos

```
cyl3d *
```

das Objekt "obj1" erzeugt würde. Machen Sie das wie angegeben, verlangt **Pictures** allerdings zusätzlich, dass Sie mindestens die Option -r für den Radius der Bodenplatte und die Option -h für die Höhe eingeben müssen. Erst dann kann der Mittelpunkt der Bodenplatte interaktiv positioniert werden. Sie müssen also beispielsweise

```
cyl3d * -r20 -h100
```

eingeben und dann den Mittelpunkt der Bodenplatte mit dem Fadenkreuz positionieren und auslösen (klicken).

Sie sollten Ihren Zylinder aber jetzt "vollautomatisch" auf dem Nullpunkt (x=0,y=0,z=0) erzeugen. Dazu geben Sie das Kommando wie folgt ein

```
cyl3d * -c0,0,0 -r20 -h100
```

Geben Sie danach sinnvollerweise

```
setpp -i
```

ein. Dies zeigt Ihre beiden Zylinder in isometrischer Sicht.

Mit

```
shade
```

können Sie die Schattierung einschalten und mit erneut aktiviertem Kommando wieder ausschalten (das ist ein sogenannter Toggle-Modus).

Zurück zu Ihrem Zylinder. Er sollte aber nicht immer senkrecht in Z-Richtung erzeugt werden. Mit der Option -d können Sie einen zweiten Punkt auf der Zylinderachse und so die Richtung angeben, z.B. in X-Minus

```
cyl3d * -c0,0,0 -r20 -h100 -d-10,0,0
```

Der Wert der Koordinate des 2. Punktes ist dabei irrelevant. Die Koordinate muss nur bezogen auf den 1. Punkt (-c Option) in die richtige Richtung zeigen (hier: x= -10, y=0, z=0)

Versuchen Sie selbst verschiedene Varianten und sollte das "Chaos" zu groß werden, können Sie die Zeichnung (den Pufferinhalt) (Achtung!) ohne Warnung löschen :

```
new -p
```

Dieses Kommando ist nicht ungefährlich, da hiermit die gesamte Zeichnung unwiederbringlich gelöscht wird. Wer den sicheren Weg gehen will, lässt in diesem Kommando einfach die Option "-p" (no Prompt) weg. Damit wird der Benutzer gefragt, ob er sicher ist.

Hinweis: Bei komplexen Zeichnungen sollte vor einem Löschvorgang vorsichtshalber immer eine Sicherungs-Datei erzeugt werden (s.u. "save" oder "saveas")

Zurück zum Kommando "cyl3d". Wenn Sie die Optionen genau betrachten, wird auffallen, dass der Kommandoname "cyl3d", der ja suggeriert, dass damit nur 3D-Zylinder erzeugt werden, schon etwas fragwürdig ist. Geben Sie z.B.

```
cyl3d * -c0,0,0 -r20 -h50 -d10,0,0 -s6 -t0
```

ein, erhalten Sie eine sechseckige Pyramide mit den entsprechenden Werten in x-Richtung (das ist wohl kaum ein Zylinder, aber die Mathematik des Kommandos gibt es her). Neben Zylindern lassen sich nämlich Kegel und Pyramiden mit ihren entsprechenden Stümpfen erzeugen. Das ergibt eine unendliche Geometrie-Vielfalt mit nur einem einzigen Kommando. Spielen Sie ggf. mit den Parametern ein wenig herum.

Bizarr wird es, wenn Sie z.B.

```
cyl3d * -c0,0,0 -r50 -h0 -s4
```

eingeben und eine quadratische (doppelseitige) Fläche erhalten. Ein Vieleck würde gewöhnlich nicht mit dem cyl3d-Kommando erzeugt werden. Somit läßt sich schon an einem einzigen Kommando die Varianzbandbreite von **Pictures** erahnen.

Jetzt wissen Sie im Prinzip, wie Kommandos funktionieren und die Hilfe dazu abrufen wird. Das ist schon ein ganz entscheidender Schritt zur "Automatisierung" von Funktionsabläufen.

Kopplung von Mathematik und Kommandos - Substitution

Zur Erzeugung von geometrischen Zeichenobjekten kommt im Hintergrund sehr viel Mathematik zum Einsatz. Da diese auch für viele selbst entworfene Projekte eine große Rolle spielt, soll Mathematik das nächste Thema sein.

Aufgabenstellung sei z.B., eine sinusförmige Kurve zu generieren,

Das wird ohne Mathematik nicht lösbar sein. Da **Pictures** mit Letzterer reichlich ausgestattet ist, ist die Aufgabenstellung bei entsprechender Grundkenntnis leicht zu bewältigen.

Um es auf einen einfachen Nenner zu bringen, heißt das wichtigste Kommando, in dem die Mathematik hinterlegt ist, "cal" (calculate).

Vorausgeschickt werden sollte, dass sich das "cal"-Kommando auf Manipulationen im "Sheet", der Tabellenkalkulation von **Pictures,** bezieht und diese unsichtbar im Hintergrund ständig mitläuft. Viele **Pictures**-Nutzer wissen gar nicht, dass sie es wenngleich unbewusst permanent benutzen. Werfen Sie einmal einen Blick darauf, indem Sie das Kommando

```
sheet
```

eintippen und mit <ENTER> abschließen.

Es erscheint ein Tabellenarbeitsblatt, dessen Fenstergröße Sie ggf. noch anpassen müssen. Dieses Worksheet besteht aus einer Tabelle mit nummerierten Zeilen und mit Buchstaben gekennzeichneten Spalten. Das Sheet dient zur Speicherung und Manipulation von Texten und Zahlen. Das "cal"-Kommando liefert dazu unzählige Operationen, u.a. Mathematik. Dies lässt sich praktisch erschließen. Tippen Sie in die Kommandozeile

```
cal
```

und drücken die F1-Taste für die dazugehörige Hilfe. Das cal-Kommando ist sehr umfangreich und ein gutes Beispiel dafür, dass man ein solch komplexes Kommando nicht komplett beherrschen kann, sondern nur weiß, wie es prinzipiell funktioniert, um die Details in der Hilfe nachzuschlagen.

Die Syntax ist im Prinzip mit

```
cal range,..=formula
```

sehr einfach, sofern man die Bedeutung der Parameter kennt. Mit "range" ist der Sheet-Zellen-Bereich anzugeben, in das der mit "formula" erzeugte Ausdruck letzten Endes geschrieben wird.

Um das schnell am Beispiel der Sinus-Kurve zu demonstrieren, tippen Sie

```
cal a1..a200=0..2*pi
```

in die Kommandozeile ein. In die Zielzellen a1 bis (..) a200 (Spalte A mit 200 Werten) werden somit 200 Zahlen zwischen 0 und 2*pi=6.28318531 eingetragen. Diese 200 Werte stellen später in der Grafik die X-Koordinaten (horizontale Achse) dar.

Nun gilt es die Werte der Y-Koordinaten (vertikale Achse) zu berechnen. Das geschieht mit

```
cal b1..b200=sin(a1..a200)
```

Dies folgert, dass in die Spalte B in die Zellen b1 bis b200 die Sinuswerte aus der zuvor erzeugten Spalte A geschrieben werden. Jede Zahl aus der Spalte A wird mit dem entsprechenden Sinus dieses Wertes in die Spalte B eingetragen. Damit ist der mathematische Teil der Aufgabenstellung bereits gelöst.

Nun gilt es, die Zahlenreihen als Grafik darzustellen. Dazu sollen die jeweils 200 Punkte einfach durch kurze Geraden(stücke) verbunden werden. Das leistet das (variantenreiche) "poly"-Kommando. Ohne momentan die Hilfe aufzurufen, tippen Sie ein

```
poly * -pa1,b1..a200,b200
```

Sofern keine Fehlermeldung auftaucht, wurde jetzt der Kurvenzug (Polygon) im Zeichnungspuffer erzeugt. Vermutlich werden Sie keine Grafik im Zeichenfenster erkennen, selbst wenn Sie das Sheet und das Hilfe-Fenster minimieren. Es liegt daran, dass unsere Grafik sehr klein (etwas mehr als 6 mm breit) ist. Mit dem Kommando

```
zoom -o*
```

können Sie das Objekt in maximaler Größe anzeigen. Als Alternative zu dem obigen "zoom"-Kommando können Sie auch das populärere

```
zall
```

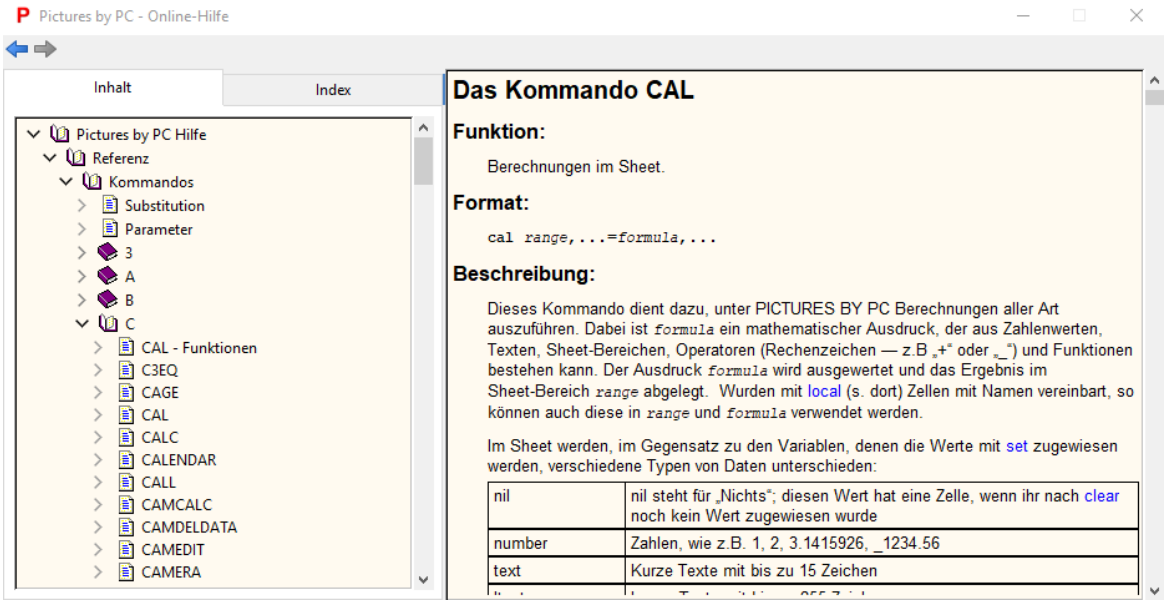
eingeben (zoom all), das alle existierenden Objekte maximal "heran zoomt".

Auf jeden Fall erkennen Sie jetzt Ihre Sinus-Kurve.

Die "Aufgabe" wurde somit erfolgreich gelöst.

Hinweis: Gängige mathematische Funktionen können Sie übrigens mit der sehr simplen Prozedur "sheetpolycurve.prc" grafisch darstellen. Sie sollten den Test aber hinter die nachfolgende Erläuterung des "cal"-Kommandos zurückstellen.

Es bedarf natürlich noch einiger ergänzender Erklärungen. Dazu greifen Sie also wieder auf die Hilfe zum cal-Kommando zurück.



Blättern Sie in dem Hilfe-Text von "cal" langsam vor bis Sie die erste Tabelle komplett sehen. Diese listet alle für das Sheet möglichen Datentypen auf. Wichtig sind u.a. "number" (Zahl) und "text" (Text), die Sie natürlich kennen. Andere sind aber speziell für grafische Anwendungen von großer Bedeutung. Blättern Sie in der cal-Hilfe zur nächsten Tabelle vor. Sie zeigt die Konstanten wie die Euler-Zahl e (Basis des natürlichen Logarithmus) oder die Kreiszahl pi.

Die nächste Tabelle zeigt zahlreiche (teilweise exotische) Operatoren, von denen Sie die Grundrechenarten (Addition (+), Subtraktion (-), Multiplikation (*) und Division (/) natürlich kennen. Aber auch hier gibt es sehr spezielle, für Grafik nützliche Operatoren (u.a. Skalar-, Vektor- und Matrix-Operatoren), über die selbst ein guter, wissenschaftlichen Taschenrechner nicht verfügt.

Blättern Sie in der cal-Hilfe jetzt noch ein bisschen weiter, bis Sie auf das blau markierte Wort "Funktionen" stoßen. Wenn Sie das anklicken, werden zahlreiche Funktionen aufgelistet, unter denen Sie auch die schon oben genutzte Sinus-Funktion (sin (im Bogenmaß)) finden.

Die CAL-Funktionen

Außer den Operatoren stehen bei cal die folgenden Funktionen zur Verfügung:

abs(x)

Absolutbetrag

abs(p)

Abstand des Punktes p vom Ursprung (0,0)

acs(x)

Arcuscosinus (Ergebnis in Bogenmaß)

acsd(x)

Arcuscosinus (Ergebnis in Grad)

ang(x)

Winkel (in Bogenmaß) in Bereich 0...2pi abbilden

angd(x)

Winkel (in Grad) in Bereich 0...360 abbilden

Spätestens mit dieser Auflistung wird deutlich, dass es keinen Sinn macht, ein einzelnes Kommando wie "cal" auswendig lernen zu wollen.

Es genügt der einfache Merksatz:

CAL liefert die Mathematik (für Kommandos)

Ansonsten ist nur zu lernen, die Hilfe zu nutzen. Damit soll an dieser Stelle die Erläuterungen zum "cal"-Kommando abgeschlossen werden.

Allerdings sollten noch einige nützliche Fakten zum SHEET erklärt werden. Grundsätzlich gibt es zur interaktiven Bedienung des Sheets eine Symbolleiste, die Sie unter Tools ("Hammer+Zirkel") finden und selbst testen sollten.

Wichtig ist zu erwähnen, dass jegliche Daten des Sheets (insbesondere jenseits der Spalte EA.. (z.B. ea1 ff.) nicht vor Überschreiben geschützt sind, sondern von **Pictures**-Kommandos und -Routinen ständig dynamisch überschrieben werden.

Sheet-Zellenbereiche können natürlich zuvor in Text-Tabellen-Daten gespeichert ("write") und externe Text-Tabellen-Daten (z.B. Excel-CSV-Dateien) geladen ("read") werden. Der komplette Datenbereich des Sheets kann mit dem Kommando "clear" gelöscht werden.

Um jetzt die Erläuterungen zu Ihrer Sinus-Kurven-Generierung abzuschließen, sollten Sie noch einen Blick auf die Hilfe des "poly"-Kommandos werfen. Tippen Sie in die Kommandozeile ein

`poly` (und F1-Taste)

Das poly-Kommando ist relativ komplex und kann u.a. Geraden, Polygone, Bezier-Kurven, Schraffuren etc. erzeugen. Mit der bei "cyl3d" erlernten Kommando-Logik können Sie das oben genutzte Kommando

```
poly * -pa1,b1..a200,b200
```

sofort verstehen. Hier erzeugt das "poly"-Kommando mit der Option ("-p") eine Kurve aus 200 in den Sheet-Zellen-Bereich a1..b200 hinterlegten Punkt-Paaren (a1,b1 bis

a200,b200). Dabei entstammen die x-Werte der Spalte A und die y-Werte der Spalte B. Das erzeugte Objekt hat den Namen mit der Vorsilbe "obj" und angehängter Nummer (hochzählend). Natürlich können Sie beim "poly"-Kommando mit der Option -p auch explizite x-/y-Koordinaten angeben. Z.B. erzeugt das Kommando

```
poly * -cp100,0,50,86.60254038,0,0
```

bei Eingabe obiger Punkt-Koordinaten ein geschlossenes, gleichseitiges Dreieck mit einer Seitenlänge von 100 und einer Höhe von 86.60254038.

Bedenken Sie übrigens immer, dass unter **Pictures** Dezimalzahlen mit einem Punkt geschrieben werden, während das Komma der Parameter-Trennung (z.B. x,y) dient.

Ergänzend könnte noch erwähnt werden, wie die Höhe des Dreiecks oben berechnet wurde. Die Formel können Sie z.B. einer mathematischen Formelsammlung entnehmen (oder ggf. selbst ableiten). Sie lautet z.B.

Höhe=Kantenlänge/2*sqr(3)

Um es zu "toppen", sollte erwähnt werden, dass alle unter "cal" hinterlegten Formeln und Funktionen unmittelbar auch mit dem Kommando "val" zur schnellen Berechnung von Ausdrücken direkt in der Kommandozeile genutzt werden können (also "val" statt "cal"). So ist die y-Koordinate des 2. Punktes im obigen "poly"-Kommando mit dem Ausdruck

```
val 100/2*sqr(3)
```

=86.60254038

berechnet worden. Sqr() ist die Quadratwurzel-Funktion (square root, s. cal-Kommando)

Übrigens könnte auch eine andere Formel zur Berechnung der Dreiecks-Höhe verwendet werden, die natürlich zum gleichen Ergebnis führen, z.B.

`val 100*cosd(30)` (cosd() ist die Kosinus-Funktion im Gradmaß)

oder auch

`val 100/(2*tand(30))` (tand() ist die Tangens-Funktion im Gradmaß)

Jetzt wäre noch die Nutzung der Mathematik mittels Substitution in Kommandos zu erläutern. Das klingt weit komplizierter als es ist. Soll z.B. in einem Kommando eine explizite Zahl durch einen zu berechnenden Ausdruck (Formel) ersetzt werden, kann unter **Pictures** die sogenannte Substitution (nachlesbar in der Hilfe-Referenz, Kommandos, Referenz- Substitution (=Ersetzung)) angewandt werden.

Prinzipiell erlaubt z.B. das Kommando "echo" die einfache Ausgabe einer Meldung in der Infozeile (Zeile über der Kommandozeile). Folgendes Kommando

```
echo 100/2*sqr(3)
```

zeigt, wie zu erwarten war, als Ergebnis nur den Meldungstext (natürlich ohne echo) in der Infozeile an. Sie wollen aber, dass das echo-Kommando die mathematische Formel auswertet. Das geschieht mit Hilfe der Substitution ($-Zeichen), die einem einfachen Formalismus unterliegt. Geben Sie z.B.

```
echo $[?100/2*sqr(3)]
```

ein, erhalten Sie den gewünschten mathematischen Wert. Die Syntax ist einfach erklärt. Der mathematischen Formel "100/2*sqr(3)" stellt man ein Fragezeichen "?" als Zeichen für "berechne" voran, umklammert den Ausdruck mit eckigen Klammern "[ ]" und setzt davor das Dollar-Zeichen "$" (als Zeichen für die Substitution). Dann besagt das, dass der dem $-Zeichen folgende, eckig umklammerte Ausdruck als Formel betrachtet wird. Der Text wird also durch die Formel ersetzt (substituiert). Das Ergebnis ist die gesuchte Zahl (hier 86.60254038). Die Syntax wirkt schon etwas eigentümlich. Das kommt daher, dass die Regeln noch aus den sehr eingeschränkten DOS-Betriebssystem-Zeiten (vor mehr als 35 Jahren) stammen.

Den entsprechenden Ausdruck können Sie jetzt auch direkt in Ihr "poly"-Kommando für den 2. y-Wert einsetzen und ersparen sich so die Eingabe der mehrstelligen Zahl.

```
poly * -cp100,0,50,$[?100/2*sqr(3)],0,0
```

Nun haben Sie ein typisches Zeichen-Kommando wie "poly" kennen gelernt und darin auch eine mathematische Formel verwendet.

Wenn Sie das verstanden haben, ist der nächste Schritt natürlich naheliegend.

Nutzung von Variablen

Werden in einer Zeichnung Geometrien erstellt, ist eine individuelle Abfrage der wichtigsten Werte der Erzeugungs-Kommandos vorteilhaft.. Das kann mit "Variablen" realisiert werden. Variablen können Werte zugewiesen werden, auf die später zugegriffen werden kann. Eine solche Zuweisung kann z.B. lauten:

```
slength=100
```

oder auch

```
set slength=100
```
(in Prozeduren immer nur mit vorangestelltem "set")

Hierbei ist "slength" der willkürlich gewählte Name einer Variablen (eine eindeutige Buchstaben- und Zahlen-Kombination, die weder mit Standard-Variablen-Namen (s.u), Sheet-Zellenbezeichnungen oder Kommando-Namen übereinstimmen sollte) und "100" ist der durch das "=" (Gleichheitszeichen) zugeordnete Wert. Eine solche Variable wird "global" genannt, da sie während des aktuellen **Pictures**-Programmlauf dauerhaft gültig ist, es sei denn, sie wird durch Leerzuweisung (z.B. slength=) gelöscht.

Auf den Inhalt einer solchen Variablen kann jederzeit durch Substitution mit einem vorangestellten "$" (Dollar-Zeichen) zugegriffen werden. Z.B. gibt das Kommando

```
echo $slength
```

den Wert 100 in der Infozeile aus.

Angewandt auf das Kommando oben, lautet dies

```
poly * -cp$slength,0,$[?$slength/2],$[?$slength/2*sqr(3)],0,0
```

Der ursprüngliche x-Wert "100" wurde also jetzt durch "$slength" ersetzt und konsequenterweise natürlich auch der 1. y-Wert (ursprünglich 50) durch den Ausdruck "$[?$slength/2]".

Anstelle eines fest vorgegebenen Wertes wäre es natürlich wünschenswert, den Variablen-Wert individuell abfragen zu können. Dazu ist das Kommando "dinput" (genau genommen eine Basic-Routine "dinput", die wie ein Kommando wirkt) sehr gut geeignet. In der Hilfe lesen Sie nach Eingabe von

`dinput` (und F1-Taste)

die Format-Zeile

```
dInput varcell [prompt] [caption]
```

und eine Reihe von Optionen.

Zweifellos haben Sie es hier mit einem schon deutlich komplexeren "Kommando" zu tun als bisher. Aber auch das folgt der gleichen Logik.

Das sollen Sie gleich auf Ihr obiges, gleichseitiges Dreieck anwenden. Tippen Sie in die Kommandozeile Folgendes ein

```
dinput slength "Kantenlänge eingeben: <100> " -d100 -
e100 "Gleichseitiges Dreieck"
```

Mit der Abfrage (prompt) "Kantenlänge eingeben: <100> " und dem Vorgabe-Wert (default) "-d100", der mit "-e100" auch editiert (edit) werden kann, sowie der Dialog-Überschrift (caption) "Gleichseitiges Dreieck" wird der Wert der Variablen "slength" abgefragt und nach der Eingabe mit <ENTER> zugewiesen.

Die Variable "slength" hat jetzt den von Ihnen gesetzten Wert. Das kann mit

```
echo $slength
```

überprüft werden.

Wählen Sie nun aus dem Kommandostack das letzte "poly"-Kommando und schließen es mit <ENTER> ab, wird das gleichseitige Dreieck mit der individuellen Kantenlänge erzeugt.

Setzen Sie anschließend das "dinput"- und das "poly"-Kommando in einer Zeile (getrennt durch ein ";" (Semikolon)) zusammen, haben Sie schon fast ein kleines, einzeiliges "Programm" für ein gleichseitiges Dreieck "geschrieben".

Wenn Sie das nahezu "perfekt" machen wollen, nutzen Sie zusätzlich noch eine dynamische Verschiebungs- und Dreh-Routine namens "mvrt".

Dazu analysieren Sie die Hilfe-Beschreibung eigeninitiativ für sich selbst. Als Ergebnis sollten Sie das obige Mehrfach-Kommando durch folgende Zeichenfolge

```
; mvrt * -b5 ; set slength=
```

ergänzen. Dadurch können Sie mit "mvrt * -b5" das Dreieck interaktiv (mit dem linken, unteren Eckpunkt) frei positionieren und zusätzlich durch "set slength=" die (nicht mehr benötigte) Variable "slength" aus dem Speicher löschen.

Am Ende sollte das Mehrfach-Kommando nunmehr wie folgt lauten:

```
dinput slength "Kantenlänge eingeben: <100> " -d100
-e100 "Gleichseitiges Dreieck" ; poly *
-cp$slength,0,$[?$slength/2],$[?$slength/2*sqr(3)],0,0
; mvrt * -b5 ; set slength=
```

Übrigens kann (als "Nebeneffekt") mit "mvrt" das Objekt, solange es am Fadenkreuz "hängt", mit Hilfe der "Bild-nach-unten"- bzw. "Bild-nach-oben"- Taste auch noch gedreht werden.

Obwohl das Kommando schon recht komplex aussieht, sollte es mit den formalen Erklärungen von oben leicht verständlich sein.

Sicherlich lässt sich jetzt schon erahnen, dass derartige Kommandofolgen für die tägliche Arbeit sehr nützlich sein können und individuell gestaltbar sind.

Zum Selbststudium wird aus diesem Grund mit dem **Pictures**-Standard (Rev. 3.8) die Datei "NuetzlicheKommados.stk", die zahlreiche, solch direkt ausführbare Mehrzeilen-Kommandos enthält, ausgeliefert (Details s.u.).

Natürlich können solche Sequenzen auch eigenen Kommando-Buttons zugeordnet werden. Eine Beschreibung dazu finden Sie ggf. in der Hilfe-Datei "p32_IndividuelleSymbolleistenErstellen.pdf" im Verzeichnis "C:\Program Files\SchottSysteme\CommonFiles\Help\PDF\DE" oder in der PDF-Themenübersicht (unter dem gleichen Namen).

Prinzip der Punktselektion

Oben bei der Erzeugung der Sinus-Kurve wurde gezeigt, wie aus dem Sheet (der Sheet-Mathematik) eine Grafik erzeugt wurde. Der umgekehrte Vorgang ist ebenfalls notwendig. D.h., eine interaktiv selektierte Punkt-Position soll ins Sheet übernommen werden, um sie ggf. mathematisch auszuwerten.

Das leistet die recht komplexe Routine "pselect" perfekt. Natürlich sollten Sie zum Verständnis wieder die Hilfe aufrufen. Um die "Forschung" zu "pselect" aber nicht ausufern zu lassen, soll das Kommando für unseren einfachen Test schrittweise erklärt werden.

Die Aufgabenstellung lautet: Mit dem Fadenkreuz soll interaktiv in einer Zeichnung ein Punkt abgegriffen und die zugehörigen Punktkoordinaten in eine Sheet-Zelle (z.B. in die Zelle C1) übernommen werden. Folgende Kommando-Variante würde das z.B. leisten:

```
pselect c1, "Bitte 2D-Punkt abgreifen" -mio all -2?
```

Die Parameter sind leicht erklärt. "c1" ist die Sheet-Zelle, in die die 2D-Punktposition nach dem Anklicken übergeben wird (das nachfolgende Komma besagt, dass "c1" keine "normale" Variable ist, sondern eine Sheet-Zelle). In diesem Fall werden die x- und die y-Koordinate gemeinsam (durch Komma separiert) in die Zelle "C1" eingetragen. "Bitte 2D-Punkt abgreifen" ist die individuelle Meldung (prompt) zur Anweisung an den Benutzer. Die Optionen "-m" (marker) markiert den abgegriffenen Punkt auf dem Bildschirm, "-i" (ignore) ignoriert eine Fehlermeldung, sofern keine Objektrastung erfolgt und "-o all" (object) besagt, dass auf alle bestehenden Objekte im Zeichnungspuffer mit (nicht spezifiziertem) Objektfang gerastet werden kann. Die Optionen "-m" und "-i" sind im obigen Fall, da sie keine Parameter haben, zu "-mio all" zusammengefasst worden. Sie hätten aber auch einzeln angegeben und getauscht werden können, z.B. "-i -m -o all". Die Option "-2?" bewirkt, dass sowohl 2D-Koordinaten als auch, bei eingeschalteter 2.5D-Ebene, 2.5D-Koordinaten ausgewertet werden.

Sofern Zeichnungsobjekte im Puffer vorliegen, ist die Objekt-Rastung dann zu beobachten, wenn das Fadenkreuz in die Nähe (< 2% des Bildausschnitts) eines Objekt-Punktes gebracht wird. Nach dem Anklicken verschwindet das Fadenkreuz und die selektierten Punktkoordinaten sind in die Sheet-Zelle C1 übergeben worden.

Sie überprüfen das mit

```
echo $c1
```

Eine Sheet-Zelle wirkt also genauso wie eine Variable. Selbstverständlich können Sie sich auch den Inhalt der Zelle C1 im Sheet ansehen (ggf. Spalte etwas verbreitern). Ebenso kann mit dem "cal"-Kommando auf diese Zelle bzw. auf deren Inhalt zugegriffen werden.

Da oben das "poly"-Kommando bereits genutzt wurde, seien hier ergänzend zwei weitere, elementare Zeichenkommandos ("arc" und "label") unter Nutzung der Koordinaten aus der Sheet-Zelle C1 erwähnt, z.B.

```
label * "Pictures by PC" -bp$c1
```

oder

```
arc kreis* -tc$c1 -r20
```

Die Details zu diesen Kommandos lesen Sie bitte selbst nach. Sie finden Sie wie üblich in der F1-Hilfe.

Standard-Variable und Systemzustände

Um jetzt langsam eigene, einfache "Projekte" auf der Basis von Mehrfachkommandos in Angriff zu nehmen können, bedarf es grundsätzlich noch der Kenntnis der **Pictures**-Standard-Variablen. Das sind Variable, die System-relevante Zustände und Eigenschaften in jeweils speziellen Formaten speichern. In der F1-Hilfe findet man sie unter

Pictures by PC Hilfe - Referenz - Variablen (alphabetisches Untermenü)

Es gibt eine sehr große Anzahl solcher Standard-Variablen. Häufig genutzt sind z.B.

COLOR	(Farbzahl (Farbe))
TXTSPEC	(Schriftart)
HATCH1	(Schraffurart)
COLSNAP	(Farbrastung)
COLPLOT	(Farbfilter)
LINETYPE	(Linientyp)
ZOOM	(Zoom-Bereich)

... und viele mehr.

Abfragen können Sie solche Variablen wie üblich, z.B. durch

```
set txtspec
```

und erhalten dann z.B. eine Meldung, wie diese

TXTSPEC=txt/3/2

Das besagt, dass der Schriftsatz "TXT" benutzt wird, die Schriftgröße 3% der Höhe des Standard-Bildschirmfensters beträgt und die Schriftfarbe Grün ist (2=grün).

Viele der Standard-Variablen können mit Hilfe des Pictures-Parameter-Dialogs gesetzt und überprüft bzw. durch spezielle Buttons ein- und ausschaltet werden. Besonders "gefährliche" Zustände (Variablen) werden durch das "Einschalten" im Button-Menü in der Farbe Rot gekennzeichnet. Den zugeordneten Variablen sollte unbedingt Aufmerksamkeit geschenkt werden.

Auf jeden Fall sollten Sie vor Ihrer Verwendung genau wissen, was sie bewirken und welchem Format sie unterliegen. Falsch angewandte Variablen können erhebliche "Überraschungen" bereiten oder Fehlfunktionen bedingen.

Wenn also in Kommando-Sequenzen Standard-Variablen vorkommen, prüfen Sie zuvor unbedingt deren Funktion und Syntax.

Bevor Sie selbst wirklich loslegen, lassen Sie uns einen kurzen Blick in die Datei "NützlicheKommandos.stk" werfen und anhand einiger Kommandos die Handhabung praktisch erklären. Achtung! Es gibt zahlreiche unterschiedliche Revision-Stände dieser Datei. Liegt also ein speziell erwähntes Kommando nicht vor, lassen Sie es einfach aus und probieren ein anderes.

Um die Datei zu laden, nutzen Sie am besten den vielseitigen Programmier-Editor "proc" und geben Folgendes ein

```
proc "procs\samples\NuetzlicheKommandos.stk"
```

Damit ist die Datei (die keinesfalls unabsichtlich geändert werden darf) im Pictures-Editor geladen. Die Taste-F2 führt an das Ende der Datei und die F1-Taste entsprechend an deren Anfang (Achtung! F1 ist im Editor nicht die Hilfe-Funktion).

In den ersten Zeilen der Datei wird kurz die Bedienung der vorliegenden Kommando-Sequenzen erklärt. Zu Beginn einer jeden Kommandozeile gibt es ein "echo"-Kommando, das die Bedienung und die Wirkung des darauf folgenden Mehrfach-Kommandos erklären soll und bei Ausführung kurz (pause ..) angezeigt wird.

Starten Sie zunächst einmal mit einem Ihrem obigen Dreiecks-Kommando ähnlichen, indem Sie die Suche im Editor mit "Strg+F" einleiten und nach "Dreieck" suchen. Die gefundene Zeile (vermutlich Zeile 83) sollen Sie jetzt markieren. Das geschieht am einfachsten, indem Sie im Editor direkt auf die zugehörige Zeilennummer klicken. Dann wird die ganze Zeile blau markiert. Mit "Strg+C" kopieren Sie jetzt den kompletten Zeileninhalt ins Clipboard (Zwischenablage), wechseln in das **Pictures**-Kommandofenster, fügen mit "Strg+V" die Kommando-Sequenz aus der Zwischenablage (Clipboard) in die Kommandozeile ein und schließen die Eingabe mit der <ENTER>-Taste ab. Danach geben Sie den gewünschten Radius-Wert (für den Umkreis) ein und platzieren das Dreieck an die gewünschte Position in der Zeichenfläche. Wie Sie sehen, ist das Ergebnis ein gleichseitiges Dreieck mit Umkreis. Bei dieser Variante wird mit dem "poly"-Kommando übrigens das Dreieck aus den Winkel-Funktionen im Grad-Maß ("sind" und "tand") berechnet.

Testen Sie jeweils auch die nächsten 3 bis 4 Kommando-Sequenz-Zeilen aus und versuchen Sie sie schrittweise zu analysieren, indem Sie die Hilfe-Funktion für jedes einzelne Kommando nachlesen.

Ggf. sollte vor der Ausführung eines neuen Kommandos jeweils der Puffer mit

```
new -p
```

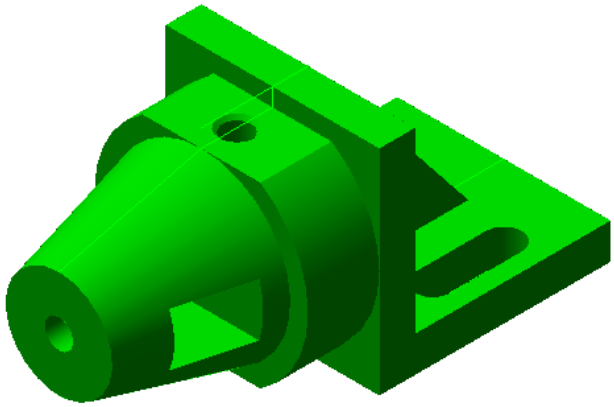
gelöscht werden.

Es macht an dieser Stelle sicherlich keinen Sinn, jede einzelne Zeile dieser Datei austesten zu wollen. Man sollte sie einfach als Fundus und Anregung für eigene Versuche betrachten. Es sollte dabei auch immer daran gedacht werden, dass es sich um eine erste "Knüttel-Programmierung" handelt, die alles andere als perfekt oder

vollständig ist, aber ein erster Anfang. Auch wenn einige der Kommando-Sequenzen in dieser Datei durch neuere Funktionen schon überlebt sind oder schlimmstenfalls auch zu Fehlermeldungen führen, die ersten Fertigkeiten sind aber jetzt vorhanden.

Ein paar Tests sollten Sie noch zusätzlich machen, indem Sie ein paar Zeilen mit "xscale" suchen und die Sequenzen ausführen lassen. "xscale" ist eine sehr trickreiche Skalierungs-Funktion, mit der bestehende Zeichnungsobjekte mittels mathematischer Funktionen verformt werden können.

Suchen Sie z.B. auch noch nach "Konus" und führen Sie die zugehörige Kommandosequenz aus, dann wird ein einfacher 3D-Konus erzeugt, den Sie ggf. noch aus dem Pictures-Basis-Seminar kennen.



Hinweis: Ein typischer Nachteil bei solchen durch Mehrfach-Kommandos entstandenen "Programmier-Knütteln" ist z.B., dass jedes Kommando einen "undo"-Schritt erzeugt. Mit "undo" wird also nicht das "ganze Gebilde" sondern nur der jeweils letzte Schritt zurückgenommen. Zur schnellen Löschung empfiehlt sich also "mdel" für Objekte oder "new -p" für den gesamten Puffer.

Wenn Sie zuvor die mathematische Kurvengenerierung noch nicht getestet haben, führen Sie einfach die Prozedur wie ein Kommando

```
sheetpolycurve
```

aus und quittieren alle Vorgabewerte mit <ENTER>. Sie erhalten zu dieser Funktion "f(x)=3*e^(-2*x/pi)*sin(4*x)" dann die grafische Darstellung. Andere Funktionen können Sie selbst ausprobieren. Nebenbei bemerkt besteht "sheetpolycurve", sieht man einmal von der Dialog-Eingabe und den Achsenmarkierungen bzw. -beschriftungen ab, im Kern auch nur aus den beiden Kommandos "cal" und "poly", wie wir sie oben bei unserer Sinus-Kurve benutzt haben.

Nun steht Ihrem weiteren "Entdeckerdrang" also nicht mehr viel im Wege.