2D- und 3D-Visualisierung mit den Graphiksystemen GR, GLI, IDL und AVS - Ein Überblick 45

Dr. H. Schumacher
M.Busch
J.Heinen

20.März.1996

Forschungszentrum Jülich GmbH

Institut für Festkörperforschung

e-mail: J.Heinen@fz-juelich.de

WWW: http://iffwww.iff.kfa-juelich.de/gli/gli.html

ftp: ftp://iffftp.iff.kfa-juelich.de/pub/gli/kurs.ps

Kursinhalt

- *Einführung in die Benutzung des **GLI** (Grundlagen)*
- *Das **G**raphische **K**ern**s**ystem (GLI/**GKS**)*
- ***G**raphics **U**tility **S**ystem (GLI/**GUS**)*
- *Aufruf von **GR**-Unterprogrammen aus GLI*
- *Interaktive Zeichnungserstellung mit GLI/**SIGHT***
- *Bildverarbeitung mit GLI/**IMAGE***
- *Benutzung von **Tcl/Tk** aus GLI*
- *Aufruf von **RPC**-Programmen*
- *Darstellung von **CGM** Dateien*
- *Darstellung von **GKSM** Dateien*

1 Einführung in die Benutzung des GLI (Grundlagen)

Klassifizierung

- ***GLI*** *steht für «**G**raphics **L**anguage **I**nterpreter»*
- ***GLI*** *ist ein interaktiver Kommando-Interpreter zur Visualisierung zwei- oder dreidimensionaler Daten*
- ***GLI*** *ist eine einfache Graphik-Programmiersprache*
- *Im* ***GLI*** *wird die graphische Ausgabe durch Kommandos gesteuert*
- ***GLI*** *ist ein Werkzeug zur Analyse und Darstellung technisch-wissenschaftlicher Daten*

Welche Graphik-Funktionalität bietet GLI?

- *Zwei- und dreidimensionale Liniengraphik*
- *Balkendiagramme*
- *Kreisdiagramme*
- *Oberflächen- und Konturbilder*
- *Histogramme*
- *Interaktiver Zugang zur **GR**-Software*
- *Signalverarbeitung*
- *Bildverarbeitung*
- *Zeichnungseditor*

Benutzerumgebungen des GLI

- *Kommandoebene*

  komfortabler Kommando-Interpreter mit 'command line editing', 'command history', 'on-line help'

- *Prozedurebene*

  Aufruf von Kommando-Prozeduren und/oder Systembefehlen

- *WYSIWYG Ebene*

  *vorgefertigte oder benutzereigene **P**oint**&C**lick Module*

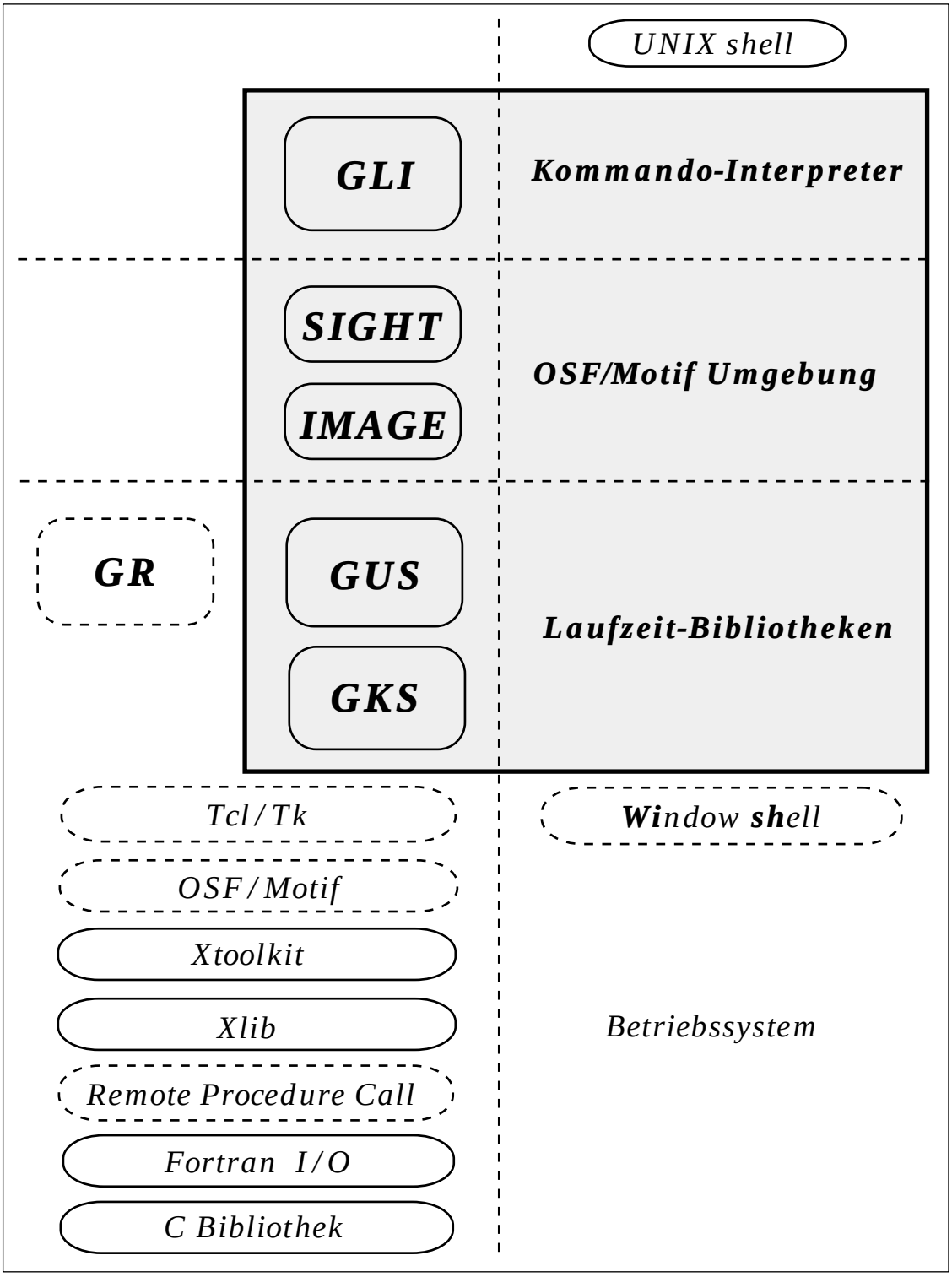
- *Anwendungsebene*

  Benutzung der GLI-Bibliotheken als graphische Entwicklungsumgebung

Ausgabe

- *Unterstützung verschiedener Graphik-Terminals und Workstations*
- *Einfache Ausgabe auf verschiedene Drucker und Plotter (***PostScript, Sixel, HP-GL***)*
- *Integration in verschiedene Textverarbeitungs-Systeme möglich (***T_EX, FrameMaker, DECwrite, Adobe Illustrator***, u.a.)*
- *Erstellung von Bilddateien (***CGM, GKSM***)*

Aufbau


UNIX shell
GLI
Kommando-Interpreter
SIGHT
IMAGE
OSF/Motif Umgebung
GR
GUS
GKS
Laufzeit-Bibliotheken
Tcl/Tk
Window shell
OSF/Motif
Xtoolkit
Xlib
Betriebssystem
Remote Procedure Call
Fortran I/O
C Bibliothek

Funktionsblöcke

- *Der **GLI** Kommando-Interpreter ist die eigentliche Verarbeitungszentrale*
- *GLI/**GKS** ist das Basissystem für die graphische Ein-/Ausgabe*
- *GLI/**GUS** (**G**raphics **U**tility **S**ystem) ist eine Anwendungsbibliothek mit komplexen graphischen Funktionen*
- *GLI/**SIGHT** (**S**imple **I**nteractive **G**raphics **H**andling **T**ool) ist ein Werkzeug zur interaktiven Zeichnungserstellung und -bearbeitung*
- *GLI/**IMAGE** ist ein Bildverarbeitungssystem*

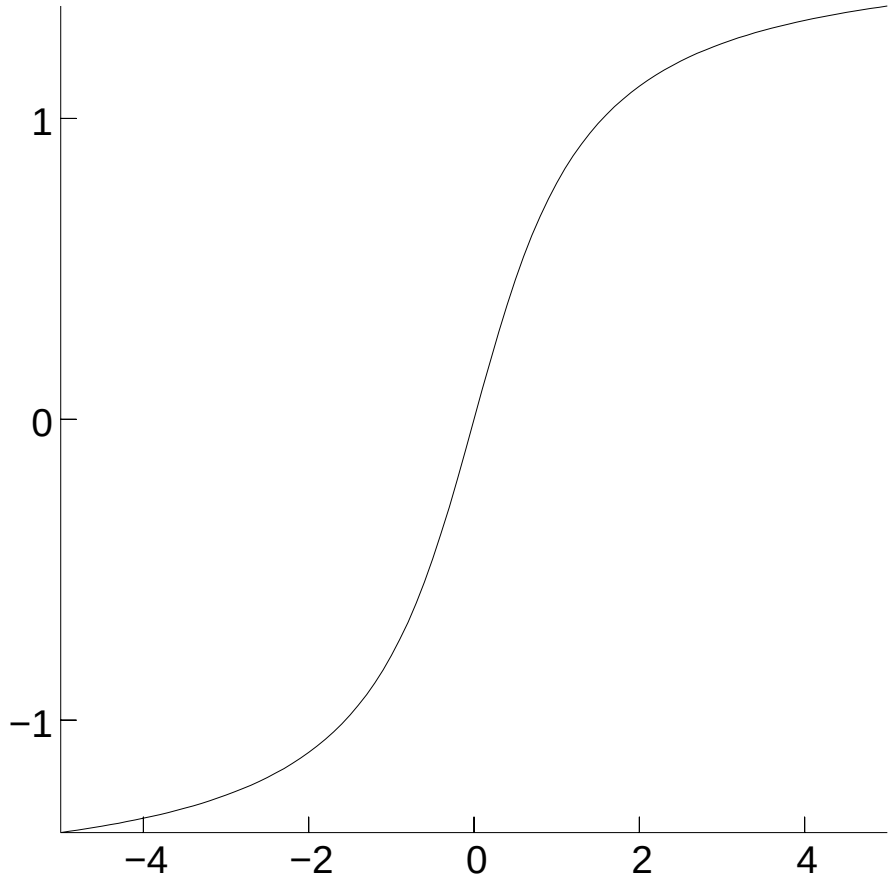
Arbeitsweise

- *GLI bietet verschiedene Bedienschnittstellen:*
  - *tastenorientiert*
  - *Kommando-gesteuert*
  - *Menü-gesteuert*
  - ***P**oint**&C**lick*
- *Benutzereigene Funktionen können über die **R**emote **P**rocedure **C**all-Schnittstelle (**RPC**) integriert werden*
- *Die Bedienschnittstelle kann durch benutzereigene **Tcl/Tk-**Scripts erweitert werden*

Ein erstes Beispiel

```
jheinen@iff201% gli

gli> x := -5(0.1)5
gli> y = arctan(x)
gli> plot x y
```



Eingabe von Kommandos

- *GLI Kommandos bestehen aus einem Namen und eventuell mehreren Schlüsselworten*

- *Kommandonamen und Schlüsselworte können abgekürzt werden*

```
gli> h
gli> he
gli> hel
gli> help
```

- *Der Benutzer kann symbolische Namen (aliases) für Kommandos definieren*

```
gli> exp*lain = "help introduction"
gli> cd = "set default"
```

Der Asterisk (*****) dient dabei als Abkürzungszeichen.

- *Symbole können an beliebiger Stelle in Kommandos eingesetzt werden*

```
gli> arch = "/u/jheinen/archive"
gli> cd 'arch'

gli> cd 'HOME'
```

- *Bei der Substitution werden GLI Symbole, Environment-Variablen, GLI-Variablen bzw. Funktionen und Tcl-Variablen berücksichtigt*

- *Fehlende Parameter werden automatisch vom GLI erfragt*

```
gli> set
_What: default
_Directory: data
gli>
```

- *Information zu einzelnen Kommandos kann über den* **HELP** *Befehl angezeigt werden*

```
gli> help
gli> help gks
gli> help gks set
```

- *Auch während der Eingabe kann durch Drücken der* **HELP**-*Taste (oder eines Fragezeichens) eine Liste der möglichen Schlüsselworte angezeigt werden*

```
gli> gks set pmark ?
_Attribute:
  TYPE    SIZE       COLOR_INDEX
_Attribute:
```

- *Die GLI Kommandozeile kann editiert werden*

Taste	Funktion
DELETE	*letztes Zeichen löschen*
CTRL/A (F14)	*Umschalten Einfügen/Überschreiben*
CTRL/B (Up Arrow)	*letzten Befehl anzeigen*
CTRL/C (Cancel)	*Befehl abbrechen*
Left Arrow	*bewegt den Cursor ein Zeichen zurück*
CYTL/E	*Cursor an das Zeileende*
CTRL/F (Right Arrow)	*Bewegt den Cursor ein Zeichen vor*
CTRL/J (LINEFEED, F13)	*letztes Wort löschen*
CTRL/R	*Befehl neu anzeigen*
CTRL/U	*Zeichen vor dem Cursor löschen*
CTRL/Z (CTRL/D, F10)	*Datei-Ende (EOF)*
Down Arrow	*nächsten Befehl anzeigen*

- *Der GLI speichert bis zu 40 Kommandos*
- *Alle Kommandos werden in einer Journal-Datei (**gli.jou**) mitprotokolliert*
- *Kommandos, die mit einem Ausrufungszeichen (**!**) beginnen, werden als Kommentar interpretiert*
- *Kommandos, die mit dem Dollarzeichen beginnen (**$**), werden als shell Kommando ausgeführt*

Definition von Daten

- *Allgemeines Format:*

```
identifier := data-item
```

oder

```
identifier = data-item
```

- *Datentypen sind* ***Symbole***, ***Variablen*** *und* ***Funktionen***

- *Definition eines* ***Symbols***

```
symbol := "equ_symbol"
```

oder

```
symbol = "equ_symbol"
```

- *Definition einer* ***Variablen*** *(Zuweisung)*

```
variable = subrange ...
variable = range ...
```

oder

```
variable := constant ...
variable := subrange ...
variable := range ...
variable := identifier ...
variable := expression ...
```

- *Deklaration von* ***Funktionen***

```
function = constant ...
function = identifier ...
function = expression ...
```

- *Datenelemente*

Datenelement	***Bedeutung***	***Beispiel***
constant	*konstanter Ausdruck*	`3.1415`
subrange	*lower-bound..upper_bound*	`1..100`
range	*initial-value(increment)final-value*	`-1(0.1)2`
identifier	*Variablen- oder Funktionsname*	`x`
expression	*arithmetischer Ausdruck*	`sqrt(x**2+y**2)`

- *Arithmetische Operatoren* **+, -, *, /, ****
- *Relationale Operatoren*
  =, <>, <, <=, >=, >
- *Logische Operatoren* **or, and**
- *Klammeroperationen*

```
8*5/2-4     ergibt 16
8*5/(2-4)   ergibt -20
```

- *Intrinsic Funktionen*

Name	Funktion
ABS	*Betrag*
ARCCOS	*Arcus Cosinus*
ARCOSH	*Arcus Cosinus Hyperbolicus*
ARCSIN	*Arcus Sinus*
ARCTAN	*Arcus Tangens*
ARSINH	*Arcus Sinus Hyperbolicus*
ARTANH	*Arcus Tangens Hyperbolicus*
COS	*Cosinus*
COSH	*Cosinus Hyperbolicus*
DEG	*Grad*
E	*e*
ERF	*Fehlerfunktion*
ERFC	*komplementäre Fehlerfunktion*
EXP	*Exponentialfunktion*
FRAC	*Fraktion*
GAMMA	*Gammafunktion*
INT	*Integer*
LN	*natürlicher Logarithmus*
LOG	*Logarithmus*
MAX	*Maximum*
MEAN	*Mittelwert*
MIN	*Minimum*
PI	*pi*
RAD	*Rad*

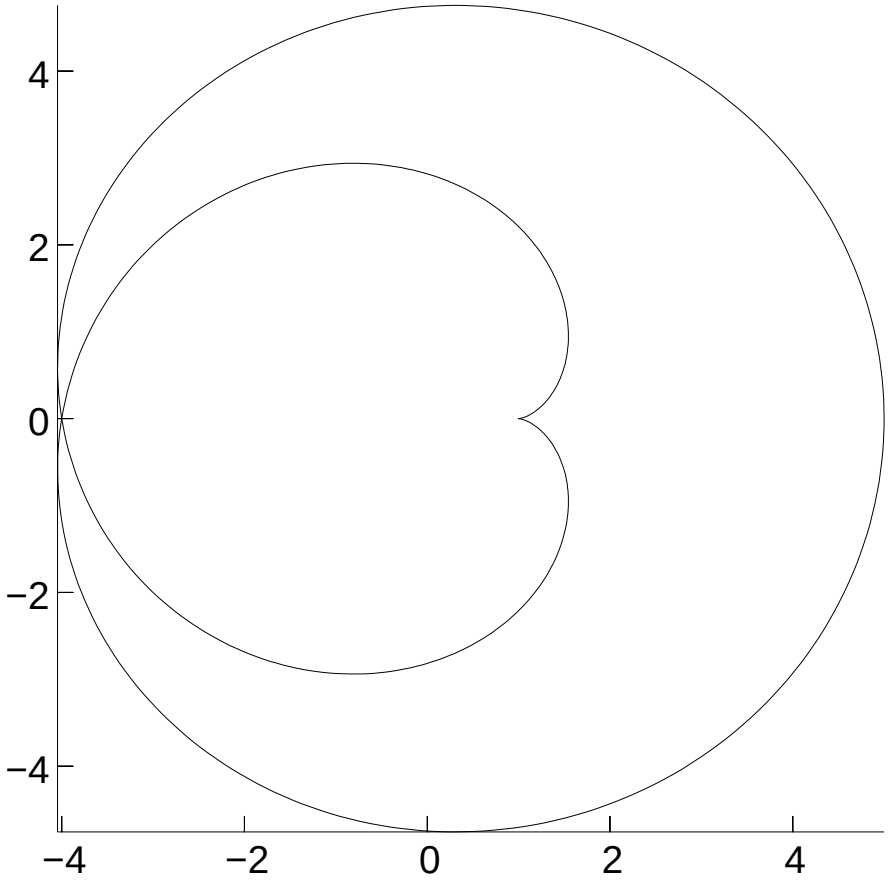
Intrinsic Funktionen (Forts.)

Name	***Funktion***
RAN	*gleichverteilte Zufallszahl*
RAND	*normalverteilte Zufallszahl*
SIGN	*Vorzeichen*
SIN	*Sinus*
SINH	*Sinus Hyperbolicus*
SIZE	*Anzahl Elemente*
SQR	*Quadrat*
SQRT	*Wurzel*
STDDEV	*Standardabweichung*
TAN	*Tangens*
TANH	*Tangens Hyperbolicus*
TOTAL	*Summe*
TRUNC	*Ganzzahlanteil*

Beispiel: Zeichnen einer Epizykloide

Ein Punkt des Umfanges eines Kreises, der ohne zu gleiten auf derAußenseite eines festen Kreises rollt, beschreibt eine Epizykloide

```
gli> t := 0(0.01)6.28   ! Wälzwinkel
gli> a := 1             ! Radius des festen Kreises
gli> b := 2             ! Radius des rollenden Kreises
gli> x = (a+b)*cos(b/a*t)-b*cos((a+b)/a*t)
gli> y = (a+b)*sin(b/a*t)-b*sin((a+b)/a*t)
gli> plot x y           ! Zeichne die Epizykloide
```



Dateien

- *GLI unterstützt frei-formatierte Textdateien mit Zahlen- und/oder Textinformation. Die Dateien werden zeilenweise eingelesen und verarbeitet.*

- *Eventuelle Textinformation wird automatisch erkannt und in den Symbolen* **P1-P***n abgelegt*

- *Lesen einer Datei:*

  gli> read *file-name var-name,...*

- *Schreiben einer Datei:*

  gli> write *file-name data-spec,...*

- *Erweitern einer Datei:*

  gli> append *file-name data-spec,...*

Beispiel (Dateien)

```
gli> x := 0(0.05)1
gli> y = sqr(x)
!
! Kreiere eine Datei temp.dat
!
gli> write temp.dat x y
  21 lines written to file t.t
gli> append temp.dat "Beispiel (Dateien)"
gli> $vi temp.dat
.
.
.
gli> initialize
!
! Lese die Datei temp.dat
!
gli> read temp.dat x y
  21 lines read from file temp.dat
  1 record contained textual information
gli> plot x y
gli> show symbol p1
  P1 = "Beispiel (Dateien)"

  Total of 1 symbol.
gli>
```

Kommando-Prozeduren

- *Kommando-Prozeduren sind Dateien, die eine Sequenz von GLI Kommandos enthalten*

- *Eine Kommando-Prozedur wird durch Eingabe eines Klammeraffen (@) gefolgt von dem Dateinamen der Prozedur aufgerufen*

  gli> *@filename*

- *Der Dateityp für eine GLI Kommando-Prozedur ist* **.GLI**

- *Es können maximal acht Parameter an eine Kommando-Prozedur übergeben werden*

  gli> *@filename* [*p1,p2,..p8*]

- *Die Befehle einer Kommando-Prozedur können am Terminal mitprotokolliert werden*

```
gli> set verify
gli> @filename
.
.
.
gli> set noverify
```

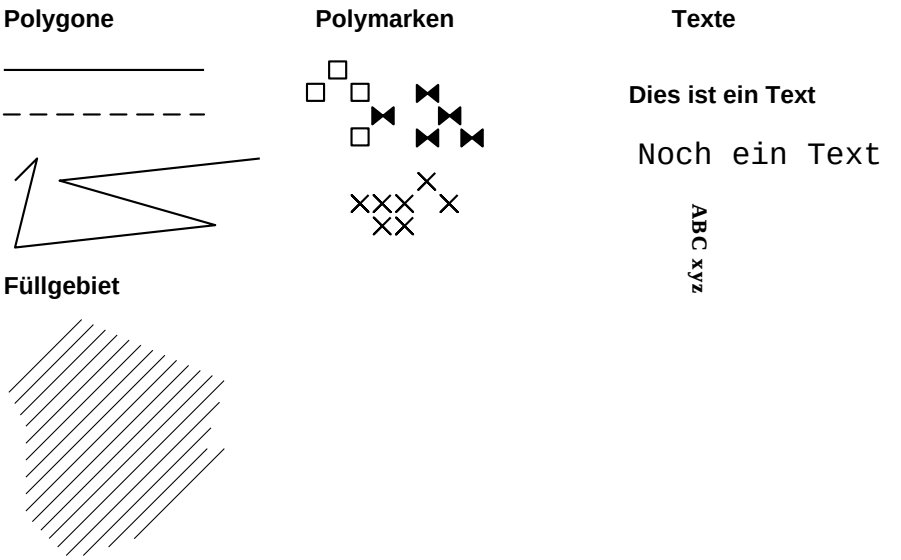
2 Das Graphische Kernsystem (GLI/GKS)

Das Graphische Kernsystem (GKS)

- *Das Graphische Kernsystem GKS ist ein genormtes Basissystem, das die Grundfunktionen für die Erzeugung computergenerierter Bilder bietet*
- *Das Graphische Kernsystem GKS stellt Funktionen zur Erzeugung graphischer Darstellungen unabhängig von einer bestimmten Anwendung bereit*
- *Das GKS stellt seine Fähigkeiten unabhängig von den graphischen Geräten (Arbeitsplätzen) bereit*
- *Die GKS-Norm definiert den Kern eines graphischen Systems unabhängig von einer Programmiersprache*
- *Es existieren Sprachschalen für FORTRAN und C (GKS FORTRAN binding, GKS C binding)*
- *Die Funktionen des GKS sind nach wachsenden Anforderungen in Leistungsstufen gegliedert (0a-2c)*

Darstellungselemente

- *Zeichnen von Linien (**polyline**)*
- *Markierung von Punktmengen (**polymarker**)*
- *Ausgabe von Zeichenfolgen (**text**)*
- *Darstellung von Flächen (**fill area**)*
- *Darstellung von Rasterbildern (**cell array**)*

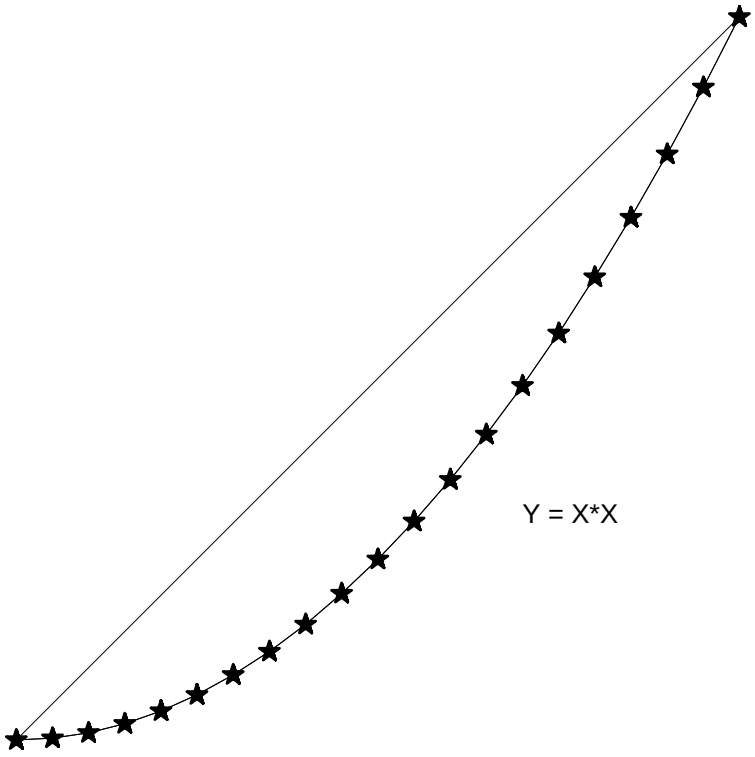


GKS Ausgabe Kommandos

GKS Ausgabefunktion	Kommando
Linien	**gks polyline** *x y*
Marken (Symbole)	**gks polymarker** *x y*
Text	**gks text** *x y text*
Füllgebiet	**gks fill_area** *x y*
Zellmatrix	**gks cell_array** *array [xdim] [first-color last-color]*

Beispiel: GKS Ausgabe Kommandos

```
gli> x := 0(0.05)1
gli> y = sqr(x)
gli> gks polyline x y
gli> gks text 0.7 0.3 Y = X*X
gli> gks polymarker x y
gli> gks fill_area x y
```



Darstellungsattribute

- *Linien (polyline) haben die Attribute*
  Linienart,
  Linenbreitefaktor *und*
  Linienfarbindex

- *Polymarken (polymarker) haben die Attribute*
  Markenart,
  Markenvergrößerungsfaktor *und*
  Polymarkenfarbindex

- *Text (text) hat die Attribute*
  Zeichenhöhe,
  Zeichenaufwärtsrichtung,
  Schreibrichtung,
  Textausrichtung,
  Schriftart *und* ***-qualität***,
  Zeichenverzerrungsfaktor,
  Zeichenabstand *und*
  Textfarbindex

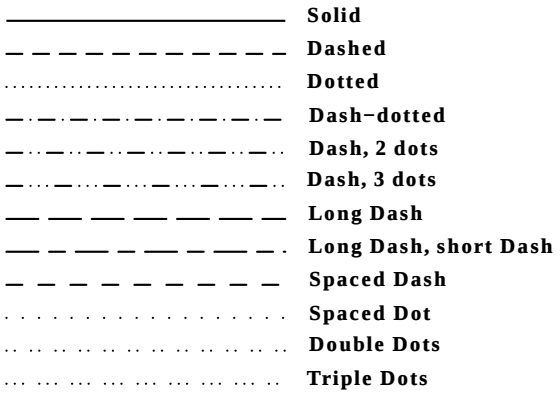
- *Füllgebiet (fill area) hat die Attribute*
  Füllgebietausfüllung,
  Füllgebietausfüllungsindex *und*
  Füllgebietfarbindex

GKS Attribut Kommandos

GKS Attribut	Kommando
Linienart	**gks set pline linetype** *type*
Linienbreitefaktor	**gks set pline linewidth** *width*
Linienfarbindex	**gks set pline color_index** *color*
Markenart	**gks set pmark type** *type*
Markenvergrößerungsfaktor	**gks set pmark size** *size*
Polymarkenfarbindex	**gks set pmark color_index** *color*
Zeichenhöhe	**gks set text height** *height*
Zeichenaufwärtsrichtung	**gks set text upvec** x y
Schreibrichtung	**gks set text path** *path*
Textausrichtung	**gks set text align** *halign valign*
Schriftart und -qualität	**gks set text fontprec** *font prec*
Zeichenverzerrungsfaktor	**gks set text expfac** *factor*
Zeichenabstand	**gks set text spacing** *spacing*
Textfarbindex	**gks set text color_index** *color*
Füllgebietausfüllung	**gks set fill int_style** *int_style*
Füllgebietausfüllungsindex	**gks set fill style_index** *index*
Füllgebietfarbindex	**gks set fill color_index** *color*

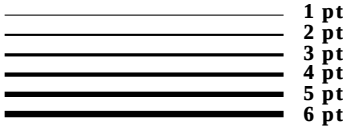
GKS Attribute

Linetypes

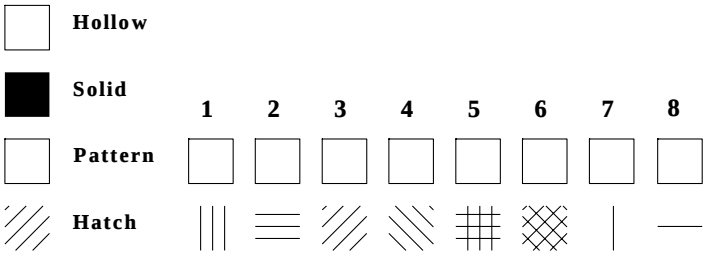


Markertypes

Linewidths



Fill Styles



Schlüsselworte für GKS Attribute

- *Linienart (***gks set pline linetype***)*
  SOLID, DASHED, DOTTED, DASH_DOTTED, DASH_2_DOT, DASH_3_DOT, LONG_DASH, LONG_SHORT_DASH, SPACED_DASH, SPACED_DOT, DOUBLE_DOT, TRIPLE_DOT

- *Markenart (***gks set pmark type***)*
  DOT, PLUS, ASTERISK, CIRCLE, DIAGONAL_CROSS, SOLID_CIRCLE, TRIANGLE_UP, SOLID_TRI_UP, TRIANGLE_DOWN, SOLID_TRI_DOWN, SQUARE, SOLID_SQUARE, BOWTIE, SOLID_BOWTIE, HOURGLASS, SOLID_HGLASS, DIAMOND, SOLID_DIAMOND, STAR, SOLID_STAR, TRI_UP_DOWN, SOLID_TRI_LEFT, SOLID_TRI_RIGHT, HOLLOW_PLUS, OMARK

- *Füllgebietsausfüllung (***gks set fill int_style***)*
  HOLLOW, SOLID, PATTERN, HATCH

- *Schriftqualität (***gks set text fontprec***)*
  STRING, CHAR, STROKE

- *Textschreibrichtung (***gks set text path***)*
  RIGHT, LEFT, UP, DOWN

- *Textausrichtung (***gks set text align***)*

  horizontal: **NORMAL, LEFT, CENTER, RIGHT**
  vertikal: **NORMAL, TOP, CAP, HALF, BASE, BOTTOM**

Attribute in Normalschrift *sind implementierungsabhängig!*

Farben

- *Die Farbe kann für jedes Ausgabeelement getrennt gesetzt werden*
- *Die Farben 0 bis 7 sind für direkte Ausgabe im GLI bestimmt*
- *GKS Attributfunktionen*

GKS Ausgabeelement	***Attributfunktion***
Linien	**gks set pline color_index** *color*
Marken (Symbole)	**gks set pmark color_index** *color*
Text	**gks set text color_index** *color*
Füllgebiet	**gks set fill color_index** *color*

- *Farbzuordnung*

GKS Farbindex	***Farbe (color)***
0	**white**
1	**black**
2	**red**
3	**green**
4	**blue**
5	**cyan**
6	**yellow**
7	**magenta**

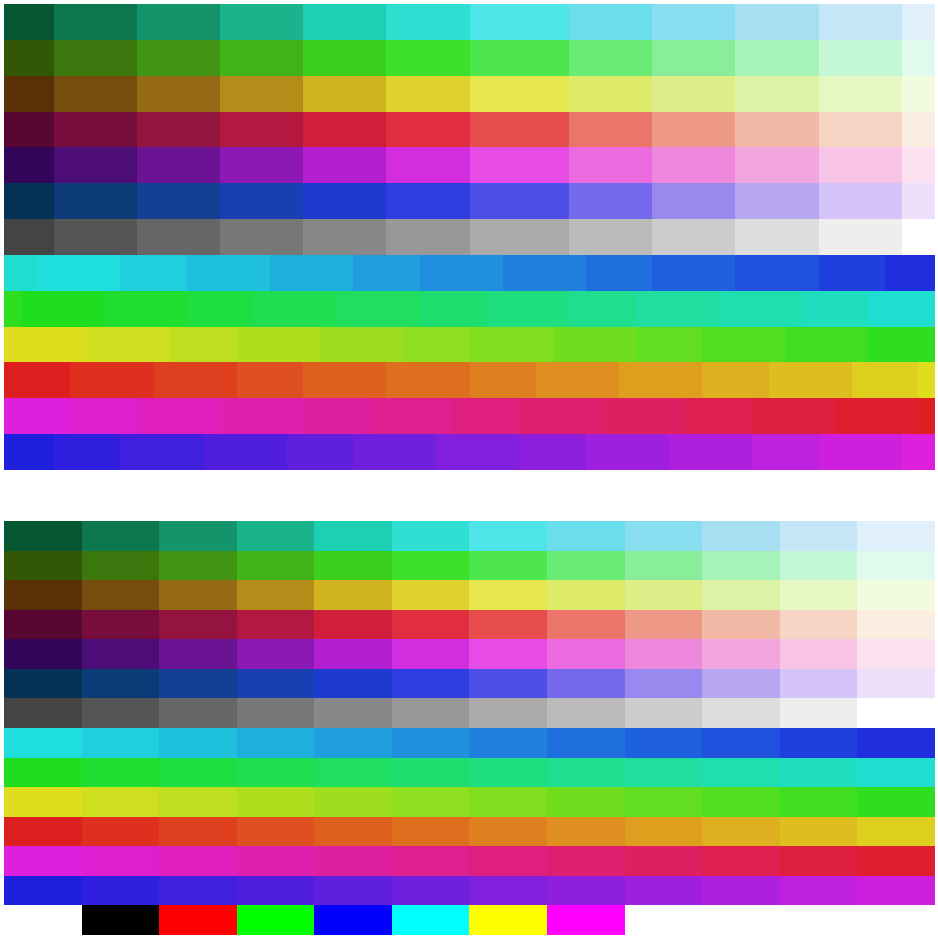
- *Die Farben 0 bis 7 können einzeln verändert werden*

  gli> gks set color_rep *index red green blue*

- *Die Farben 0 - 7 werden (wenn möglich) als lese-/schreibbare Farben angelegt*
- *Die Farben 8 bis 79 sind für GLI-interne Bibliotheks-Programme reserviert und können mit vorgefertigten Farbtabellen belegt werden*
- *Die Farben 80 bis 163 sind für GKS-internen Gebrauch reserviert*
- *Die Farben 258 bis 979 sind für die GR-Software reserviert*
- *Die Farben 80 - 979 werden erst bei Bedarf angelegt*
- *Die Farben 8 - 979 werden auf s/w Geräten automatisch in Graustufen umgerechnet (Dithering)*

Beispiel: Farben

```
gli> gks set viewport 0.05 0.95 0.05 0.45
gli> c := 0..7 0 0 0 0 8..163
gli> gks cell c 12
gli> gks set viewport 0.05 0.95 0.5 0.95
gli> c := 257..312 312..367 367..422 422..477 477..532
gli> c := c 532..587 588..979
gli> gks cell c 56
```



Koordinatensysteme und Transformationen

- *Die Transformation von Anwenderdaten auf Gerätekoordinaten erfolgt zweistufig:*
  - *Normalisierungstransformation (Weltkoordinaten −> normalisierte Gerätekoordinaten)*
  - *Gerätetransformation (normalisierte Gerätekoordinaten −> Gerätekoordinaten)*
- *Auswahl der Transformation*

  gli> gks select xform (ndc|wc)

- *Definition der Weltkoordinaten (Benutzerkoordinaten)*

  gli> gks set window *xmin xmax ymin ymax*

- *Definition der Darstellungsfläche*

  gli> gks set viewport *xmin xmax ymin ymax*

- *Klippen am Darstellungsfeld und/oder Gerätefenster wahlfrei*

  gli> gks set clipping *(*on|off*)*

- *Definition eines Arbeitsplatzfensters (Ausschnittbildung)*

  gli> gks set ws_window *xmin xmax ymin ymax*

- *Definition des Arbeitsplatzdarstellungsbereichs (Bildgröße)*

  gli> gks set ws_viewport *xmin xmax ymin ymax*

- *Das Ansichtsverhältnis von Arbeitsplatzfenster und Arbeitsplatzdarstellungsbereich muß übereinstimmen*

- *Logarithmische und Polarkoordinaten werden im GKS nicht behandelt*

Erfragefunktionen

```
gli> gks inquire gks_state
GKS state on Thu Oct 19 08:51:25 1995

      POLYLINE  Linetype: Solid
                Linewidth scale factor: 1.00
                Color: Black
    POLYMARKER  Marker Type: Asterisk
                Marker Size Scale Factor: 2.00
                Color: Black
          TEXT  Font: Helvetica Medium
                Precision: String
                Expansion factor: 1.00
                Spacing factor: 0.00
                Color: Black
                Height: 0.027
                Text Up Vector: (0.00, 1.00)
                Path: Right
                Alignment: (Normal, Normal)
     FILL AREA  Interior Style: Hollow
                Style Index: 1
                Color: Black
TRANSFORMATION  Number: 1, Clipping: on
                Viewport: (0.2, 0.9, 0.2, 0.9)
                Window: (0, 1, 0, 1)
gli>
```

Arbeitsplätze

- *GKS basiert auf der Konzeption abstrakter graphischer Arbeitsplätze*

- *Die Ausgabe kann auf mehreren Arbeitsplätzen gleichzeitig erfolgen*

- *Aufbau einer Verbindung zu einem Arbeitsplatz*

```
gli> gks open_ws name [type]
```

- *Freigabe einer Verbindung*

```
gli> gks close_ws name
```

- *Ein-/Ausschalten der Ausgabe*

```
gli> gks activate_ws name
gli> gks deactivate_ws name
```

- *GLI öffnet (und aktiviert) beim Aufruf implizit einen* **Terminal** *Arbeitsplatz*

```
gli> gks open_ws terminal
```

Ausgabegeräte

Name	Typ	Bedeutung	Ausgabe
terminal	16	*VT 330 (s/w)*	-
	17	*VT 340 (farbig)*	-
	72	*TEK 401x*	-
	82	*TEK 42xx*	-
	201	*TAB 132/15-G*	-
	204	*Monterey*	-
	207	*IBM/PC*	-
	210	*X display (nur Ausgabe)*	-
	211	*X display*	-
	217	*X display mit Framepuffer*	-
metafile	2	*GKSM output metafile*	`gli.gksm`
cgm_file	7	*CGM binär*	`gli.cgm`
	8	*CGM Klartext*	`gli.cgm`
laser_printer	38	*DEC LN03+*	`gli.lw`
plotter	53	*HP-GL (HP Plotter)*	`gli.pl`
figure_file	61	*PostScript (s/w)*	`gli.eps`
	62	*Color PostScript*	`gli.eps`
	63	*CompuServe GIF dump (s/w)*	`gli`*nn*`.gif`
	64	*CompuServe GIF dump (farbig)*	`gli`*nn*`.gif`
	92	*DEC LJ250 Companion Color Printer*	`gli`*nn*`.six`
	214	*X display mit Sun Rasterfile dump*	`gli.rf`
	215	*X display mit PCM dump*	`gli.pcm`

Die Arbeitsplatztypen `63`, `64` *und* `92` *benutzen die Display PostScript Erweiterung (DPS extension, Adobe-DPS-Extension) des X Servers*

GKS Kontrollfunktionen

GKS Kontrollfunktion	*Kommando*
GKS öffnen	**gks open_gks**
GKS schließen	**gks close_gks**
Arbeitsplatz öffnen	**gks open_ws** *name* [*type*]
Arbeitsplatz schließen	**gks close_ws** *name*
Arbeitsplatz aktivieren	**gks activate_ws** *name*
Arbeitsplatz deaktivieren	**gks deactivate_ws** *name*
Zeichenbereich löschen	**gks clear_ws**
Ausgabepuffer leeren	**gks update_ws**
GKS unbedingt schließen	**gks emergency_close**

Forschungszentrum Jülich GmbH
Institut für Festkörperforschung
- J. Heinen -

Segmente

- *GKS bietet einen arbeitsplatzunabhängigen Segmentspeicher (**W**orkstation **I**ndependent **S**egment **S**torage)*

- *Der WISS wird im GKS als normaler Arbeitsplatz behandelt*

```
gli> gks open_ws wiss 5
```

- *Mit der Segmenttransformation kann ein Segment verschoben, skaliert und gedreht werden (2 x 3 Segmenttransformationsmatrix)*

```
gli> gks set seg_xform 0.5 0.5 0 0 pi/2 1 1
```

- *Segmente können aus dem WISS auf alle aktiven Arbeitsplätze kopiert werden*

```
gli> gks copy_sg
```

- *Anwendungsgebiet: z.B. Hardcopy*

Beispiel: Segmente

```
!
! Öffne den WISS
!      Segment wird geöffnet
!
gli> gks open_ws wiss 5
!
! Erstelle ein einfaches Bild
!
gli> x := 0(0.01)1
gli> y = sqr(x)
gli> gks polyline x y
!
! Lösche Arbeitsplatz
!      Segment wird geschlossen
!      WISS wird deaktiviert
!
gli> gks clear_ws
!
! Drehe um 90 Grad um den Punkt 0.5, 0.5;
! keine Verschiebung oder Skalierung
!
gli> gks set seg_xform 0.5 0.5 0 0 pi/2 1 1
!
! Schreibe das Bild auf eine PostScript Datei
!
gli> gks open_ws fig 62
gli> gks copy_sg
gli> gks close_ws fig
```

CGM Bilddatei

- *Der CGM-Standard (**C**omputer **G**raphics **M**etafile, ISO 8632) beschreibt ein Format zur Speicherung graphischer Daten*
- *GLIGKS enthält einen CGM-Treiber*
- *GLIGKS unterstützt:*
  - *CGM Klartext-Format*
  - *CGM Binär-Kodierung*
- *Darstellung von CGM-Dateien mit* **`cgmview`**
- *Anwendungsgebiete:*
  - *Langzeitspeicherung von Bildern (Archivierung)*
  - *Austausch von Bildern zwischen verschiedenen Systemen*
  - *Exportierung z.B. in Textverarbeitungssysteme*

Beispiel: CGM

```
!
! Öffne binäre CGM-Datei
!
gli> gks open_ws cgm 7
!
! Zeichne Kurve
!
gli> x := 0(0.05)1
gli> y = sqr(x)
gli> gks polyline x y
!
! Schließe CGM-Datei
!
gli> gks close_ws cgm
!
! Verlasse GLI und starte CGM Viewer
!
gli> quit
% cgmview gli.cgm
```

GKSM Bilddatei

- *GLIGKS enthält einen GKSM-Ausgabetreiber*
- *Der GKSM Treiber ist (wegen des geringen Overheads) sehr effizient*
- *Darstellung von GKSM-Dateien mit* **`gligksm`**
- *Debugging-Hilfe für beliebige GKS-Anwendungen (*`gligksm -v`*)*
- *Anwendungsgebiete:*
  - *Kurzzeitspeicherung von Bildern*
  - *Batchorientierte Graphik-Ausgabe*
  - *Austausch von Bildern zwischen verschiedenen Anwendungen (auf gleicher Rechnerarchitektur)*
  - *Hardcopy*

Beispiel: GKSM

```
!
! Öffne GKSM-Ausgabe Bilddatei
!
gli> gks open_ws metafile 2
!
! Zeichne Kurve
!
gli> x := 0(0.05)1
gli> y = sqr(x)
gli> gks polyline x y
!
! Aktualisiere GKSM-Datei
!
gli> gks update_ws
!
! Starte externen GKSM Viewer
!
gli> $ 'GLI_HOME'/gligksm gli.gksm
```

3 Graphics Utility System (GUS)

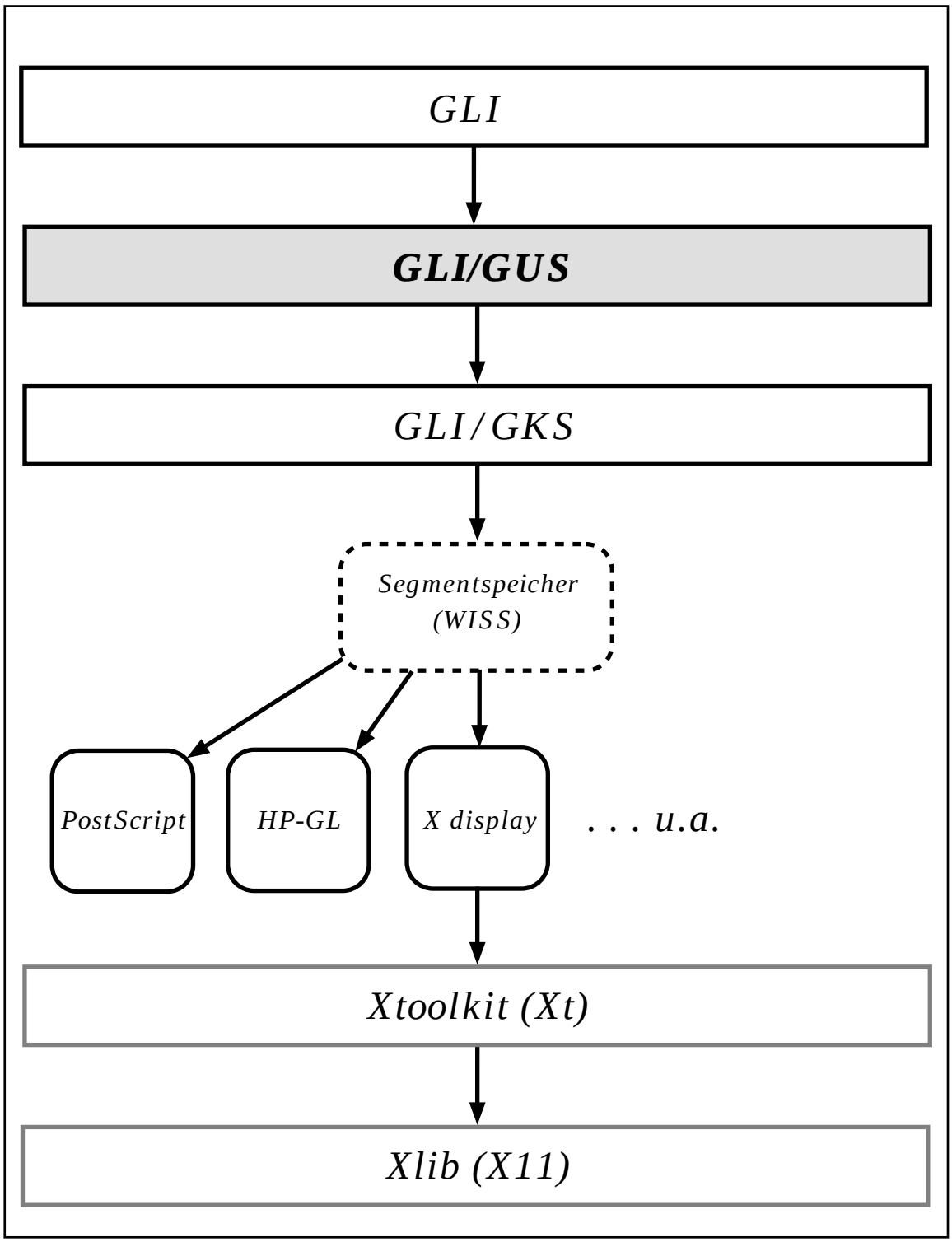
Graphics Utility System (GUS)

- *GLI/**GUS** ist eine Anwendungsbibliothek des Graphischen Kernsystems (GKS)*
- ***GUS** beinhaltet Funktionen, die durch den GKS Standard nicht abgedeckt werden*
  - *erweiterte Ausgabe*
  - *Transformationen*
- ***GUS** definiert zwei vom GKS unabhängige Koordinatensysteme*
  - *Transformation von logarithmischen (Raum-) Koordinaten in lineare Koordinaten*
  - *Transformation von (linearen) Raumkoordinaten in Weltkoordinaten*
- ***GUS** setzt auf der GKS FORTRAN Sprachschale auf und kann aus **C** und **FORTRAN** Programmen aufgerufen werden*

Funktionalität

- *Achsen, Gitter*
- *Logarithmische und/oder gespiegelte Koordinatensysteme*
- *Splines*
- *Fehlerbalken*
- *Balken- und Kreisdiagramme*
- *Kontur- und Oberflächendiagramme*
- *Einfache Textausgabe (mit mathematischen Symbolen, griechischen Zeichen)*
- *Maskengesteuerte Diagrammerstellung*

Funktionsweise



Achsen

- *Format*

```
gus axes_2d      x-tick
                 y-tick
                 x-org
                 y-org
                 major-x
                 major-y
                 tick-size
```

Parameter	Bedeutung
x-tick	*Schrittweite auf der X-Achse*
y-tick	*Schrittweite auf der Y-Achse*
x-org	*Ursprung der X-Achse*
y-org	*Ursprung der Y-Achse*
major-x	*Anzahl Teilintervalle auf der X-Achse*
major-y	*Anzahl Teilintervalle auf der Y-Achse*
tick-size	*Länge der Teilstriche*

- *Die Skalierungsart wird mit dem* **gus set scale** *Befehl festgelegt*

gli> gus set scale *scale [flip]*

scale	*Skalierung*
linear	*lineare Einteilung*
x_log	*X-Achse logarithmisch*
y_log	*Y-Achse logarithmisch*
xy_log	*X- und Y-Achse logarithmisch*

flip	*Spiegelung*
none	*keine*
flip_x	*X-Achse gespiegelt*
flip_y	*Y-Achse gespiegelt*
flip_xy	*X- und Y-Achse gespiegelt*

- *Sonderfälle*

Parameter	**Wirkung**
`x-tick, y-tick <= 0`	*keine Achse*
`major-x, major-y < 0`	*keine Beschriftung*
`major-x, major-y = 0, 1`	*keine Teilstriche*
`x-tick, y-tick = 1`	*Beschriftung in Normalformat, z.B.* `0.001` *oder* `1E-6`
`x-tick, y-tick = 2`	*Beschriftung mit hochgestellten Exponenten, z.B.* 10^2

Achsen (3D)

- *Format*

```
gus axes_3d      x-tick
                 y-tick
                 z-tick
                 x-org
                 y-org
                 z-org
                 major-x
                 major-y
                 major-z
                 tick-size
```

Parameter	Bedeutung
x-tick	Schrittweite auf der X-Achse
y-tick	Schrittweite auf der Y-Achse
z-tick	Schrittweite auf der Z-Achse
x-org	Ursprung der X-Achse
y-org	Ursprung der Y-Achse
z-org	Ursprung der Z-Achse
major-x	Anzahl Teilintervalle auf der X-Achse
major-y	Anzahl Teilintervalle auf der Y-Achse
major-z	Anzahl Teilintervalle auf der Z-Achse
tick-size	Länge der Teilstriche

- *Die Skalierungsart wird mit dem* **`gus set scale`** *Befehl festgelegt*

  `gli>` `gus set scale` *`scale [flip]`*

scale	*Skalierung*
linear	*lineare Einteilung*
x_log	*X-Achse logarithmisch*
y_log	*Y-Achse logarithmisch*
xy_log	*X- und Y-Achse logarithmisch*
z_log	*Z-Achse logarithmisch*
xz_log	*X- und Z-Achse logarithmisch*
yz_log	*Y- und Z-Achse logarithmisch*
xyz_log	*X-, Y- und Z-Achse logarithmisch*

flip	*Spiegelung*
none	*keine*
flip_x	*X-Achse gespiegelt*
flip_y	*Y-Achse gespiegelt*
flip_xy	*X- und Y-Achse gespiegelt*
flip_z	*Z-Achse gespiegelt*
flip_xz	*X- und Z-Achse gespiegelt*
flip_yz	*Y- und Z-Achse gespiegelt*
flip_xyz	*X-, Y- und Z-Achse gespiegelt*

- *Sonderfälle*

Parameter	Wirkung
`x-tick, y-tick, z-tick <= 0`	*keine Achse*
`major-x, major-y, major-z < 0`	*keine Beschriftung*
`major-x, major-y, major_z = 0, 1`	*keine Teilstriche*
`x-tick, y-tick, z-tick = 1`	*Beschriftung in Normalformat, z.B.* `0.001` *oder* `1E-6`
`x-tick, y-tick, z-tick = 2`	*Beschriftung mit hochgestellten Exponenten, z.B.* 10^2

- *Die Z-Achse wird mit dem* **`gus set space`** *Befehl definiert*

  `gli>` `gus set space` *`zmin zmax rotation tilt`*

Parameter	Bedeutung	Standardeinstellung
zmin	*Minimum Z*	0
zmax	*Maximum Z*	1
rotation	*Drehwinkel (XY-Ebene)*	30
tilt	*Kippwinkel*	60

Beispiele: Achsen

```
! Standard-Fenster mit linearen Achsen
gli> initialize
gli> gus axes_2d
.
.
.
! logarithmisches Koordinatensystem
gli> gks set window 1 100 0.1 1000
gli> gus set scale xy_log
gli> gus axes_2d
.
.
.
! halb-logarithmisches Koordinatensystem mit
! zusätzlicher Einteilung (rechts oben)
gli> gks clear_ws
gli> gks set window 0 100 0.1 1000
gli> gus set scale y_log
gli> gus axes_2d 10 2 0 0.1 2 1 0.01
gli> gus axes_2d 10 2 100 1000 -2 -1 -0.01
.
.
.
! ...oder in einfacher Ausführung
gli> gks clear_ws
gli> gus axes_2d
.
.
.
! Drei-dimensionales Koordinatensystem
gli> gks clear_ws
gli> gus axes_3d
```

Splines

- *Format*

  gus spline *x y* [*smoothing*]

Parameter	Bedeutung
x	*X-Werte*
y	*Y-Werte*
smoothing 0 -1 -2	*Spline-Typ* *natürlicher Spline* *geglätteter Spline* *B-Spline*

- *Beispiel*

```
gli> x := 0.1 0.3 0.5 0.7 0.9
gli> y := 0.4 0.1 0.3 0.2 0.7
gli> gus polymarker x y
gli> gus spline x y      ! natuerlicher Spline
.
.
.
gli> gus spline x y -1   ! geglaetteter Spline
.
.
.
gli> gus spline x y -2   ! B-Spline
```

Fehlerbalken

- *Format*

```
gus error_bars  x
                y
                e1
                e2
                orientation
```

Parameter	Bedeutung
x	*X-Werte*
y	*Y-Werte*
e1	*negativer Fehler*
e2	*positiver Fehler*
orientation vertical horizontal	*Orientierung* *vertikale Fehlerbalken* *horizontale Fehlerbalken*

- *Beispiel*

```
gli> x := 0.1 0.3 0.5 0.7 0.9
gli> y := 0.4 0.1 0.3 0.2 0.7
gli> sd := 0.1 0.05 0.15 0.12 0.09
gli> e1 := y-sd
gli> e2 := y+sd
gli> gus error_bars x y e1 e2 vertical
```

Balken- und Kreisdiagramme

- *Format*

  gus bar_graph *x y*
  gus pie_chart *y*

Parameter	Bedeutung
x	*X-Werte*
y	*Y-Werte*

- *Beispiel*

```
gli> x := 0.1 0.3 0.5 0.7 0.9
gli> y := 0.4 0.1 0.3 0.2 0.7
gli> gks set fill int_style solid
gli> gus bar_graph x y
.
.
.
gli> gks set fill int_style pattern
gli> gus bar_graph x y     .
.
.
.
gli> gks clear_ws
gli> gks set fill int_style solid
gli> gus pie_chart y
```

Konturen

- *Format*

```
gus contour      x
                 y
                 h
                 z
                 [dimension-x]
                 [major-h-count]
```

Parameter	Bedeutung
x	X-Werte
y	Y-Werte
h 1 Wert mehr als 1 Wert	Konturhöhen berechne Konturhöhen Konturhöhen wie angegeben
z	Z-Werte
dimension-x	X-Dimension der Z-Werte
major-h-count 0 1 > 1	Zähler für Beschriftung der Höhenlinien keine Beschriftung automatische Beschriftung Beschriftung wie angegeben

Beispiel: Konturen

```
gli> x := -3(0.15)3
gli> y := x
gli> h := 0(0.2)2.8
gli> z = exp(sin(y)*cos(x**2))
gli> gks set window -3 3 -3 3
gli> gus set space 0 3 0 90
gli> gus contour x y h z
!
! Konturlinien mit Beschriftung
!
gli> gks clear_ws
gli> gus contour x y h z size(x) 2
!
! Perspektivische Darstellung
!
gli> gks clear_ws
gli> gus set space 0 6 30 60
gli> gus contour x y h z
```

Oberflächen

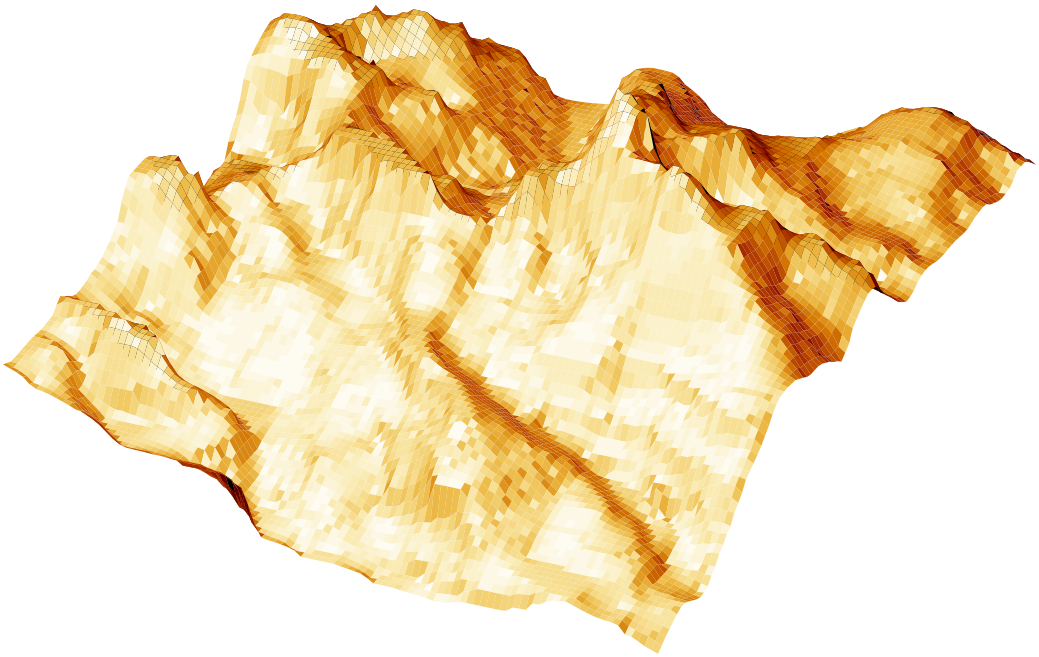
- *Format*

```
gus surface     x
                y
                z
                [option]
                [dimension-x]
```

Parameter	Bedeutung
x	*X-Werte*
y	*Y-Werte*
z	*Z-Werte*
option lines mesh filled_mesh colored_mesh z_shaded_mesh shaded_mesh cell_array	*Darstellungsart* *Liniendarstellung* *Gitterdarstellung* *" (ausgefüllt)* *farbige Darstellung* *" (nach Z-Werten)* *schattierte Darstellung* *Zellmatrixdarstellung*
dimension-x	*X-Dimension der Z-Werte*

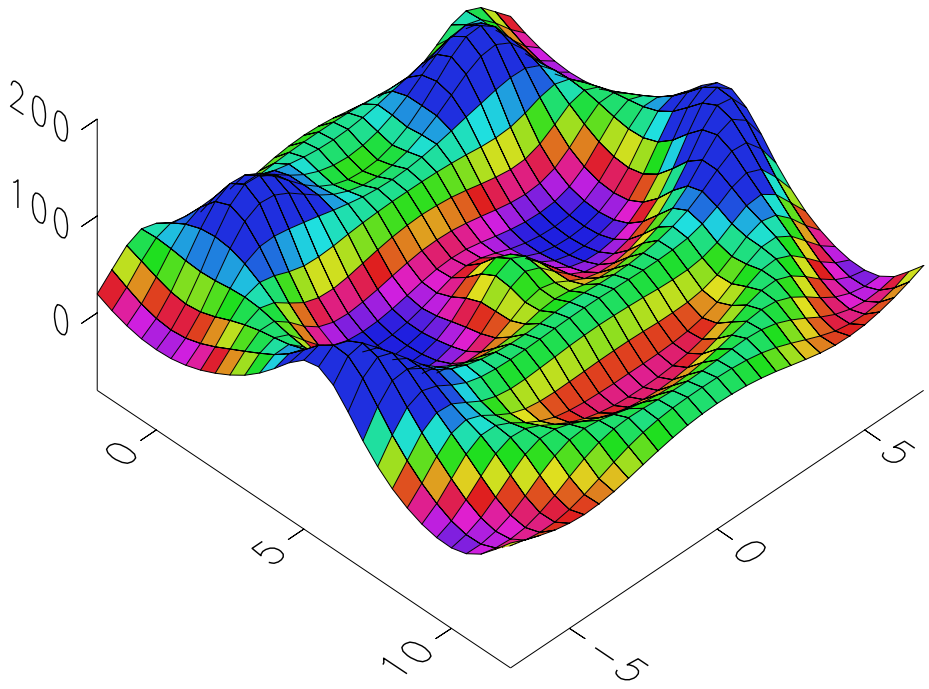
Beispiel: Oberflächen

```
gli> read 'GLI_DEMO'xdemo5.dat z
gli> x := 1..100
gli> y := 1..100
!
! Zeichne eine schattierte Oberflaeche
!
gli> gks set window 1 100 1 100
gli> gks set viewport 0 1 0 1
gli> gus set space 2400 5000 30 60
gli> gus set color glowing
gli> gus surface x y z shaded_mesh
```



Beispiel: Oberflächen (Forts.)

```
gli> initialize
gli> x := -2(.5)12
gli> y := -7(.5)7
gli> r1 = sqrt((x-5)**2+y**2)
gli> r2 = sqrt((x+5)**2+y**2)
gli> z = (exp(cos(r1))+exp(cos(r2))-0.9)*25
gli> gks set window -2 12 -7 7
gli> gks set viewport .1 .9 .1 .9
gli> gus set space -80 200 45 70
gli> gus axes_3d
gli> gus surface x y z z_shaded_mesh
gli> gus surface x y z mesh
```



Text

- *Format*

```
gus text   x
           y
           string
```

Parameter	Bedeutung
x	X-Position
y	Y-Position
string	Text

- *Der (/) Operator wird in Bruchschreibweise umgesetzt*

- *Der (**) Operator erzeugt einen hochgestellten Exponenten*

- *Indizes, Exponenten, Subscripts und Superscripts können mit Makros erzeugt werden*

  <*argument*[*index*[*exponent*[*subscripts*[*superscript*]]]]**>**

- *Wurzelausdrücke*

  <ROOT*argument*[*degree*]>**

- *Griechische Buchstaben*

  <*keyword***>**

keyword	***Symbol***	***keyword***	***Symbol***
Alpha	α, A	Omicron	ο, Ο
Beta	β, Β	Pi	π, Π
Chi	χ, Χ	Vartheta	ϑ
Delta	δ, Δ	Rho	ρ, Ρ
Epsilon	ε, Ε	Sigma	σ, Σ
Phi	ϕ, Φ	Tau	τ, Τ
Gamma	γ, Γ	Upsilon	υ, Υ
Eta	η, Η	Psi	ψ, Ψ
Iota	ι, Ι	Omega	ω, Ω
Kappa	κ, Κ	Xi	ξ, Ξ
Lambda	λ, Λ	Zeta	ζ, Ζ
Mu	μ, Μ	Theta	θ, Θ
Nu	ν, Ν	Varphi	φ

- *Alle GKS Textattribute werden von der* **gus text** *Funktion als normalisierte Gerätekoordinaten interpretiert*

Beispiel: Text

```
gli> gks set text height 0.048
gli> gks set text align center half
gli> gus text .3 .9 <A\ind\exp\sub\sup>
gli> gus text .7 .7 <<alpha>\<beta>>
gli> gus text .3 .5 1/x + y**2
```

$$A_{\mathrm{sub}}^{\mathrm{sup}}{}_{\mathrm{ind}}^{\mathrm{exp}}$$

$$\alpha_{\beta}$$

$$\frac{1}{x} + y^{2}$$

GUS/Autoplot

- *GUS/**Autoplot** ist eine menu-orientierte Umgebung zur Erzeugung einfacher 2-D Graphiken*

- *Aufruf*

```
gli> autoplot
```

- *Funktionalität*

  - *automatische Skalierung*
  - *lineare oder logarithmische Skalierung*
  - *Achsen-Beschriftung*
  - *Legenden*
  - *verschiedene Darstellungsarten*

FORTRAN Beispiel

- *Quellcode*

```
program myprog
integer asf(13)
data asf /13*1/
call gopks(6, -1)
call gsasf(asf)
call gopwk(1, 0, 0)
call gacwk(1)
call gsvp(1, 0.2, 0.9, 0.2, 0.9)
call gselnt(1)
call gschh(0.027)
call gstxfp(3, 0)
call gusax(0.2, 0.2, 0.0, 0.0, 1, 1, -0.01, ierr)
call guwk(1, 0)
read(*, *)
call geclks
end
```

```
gli> axes
```

- *Übersetzen, Linken*

  auf Systemen mit shared libraries, z.B. Digital UNIX:

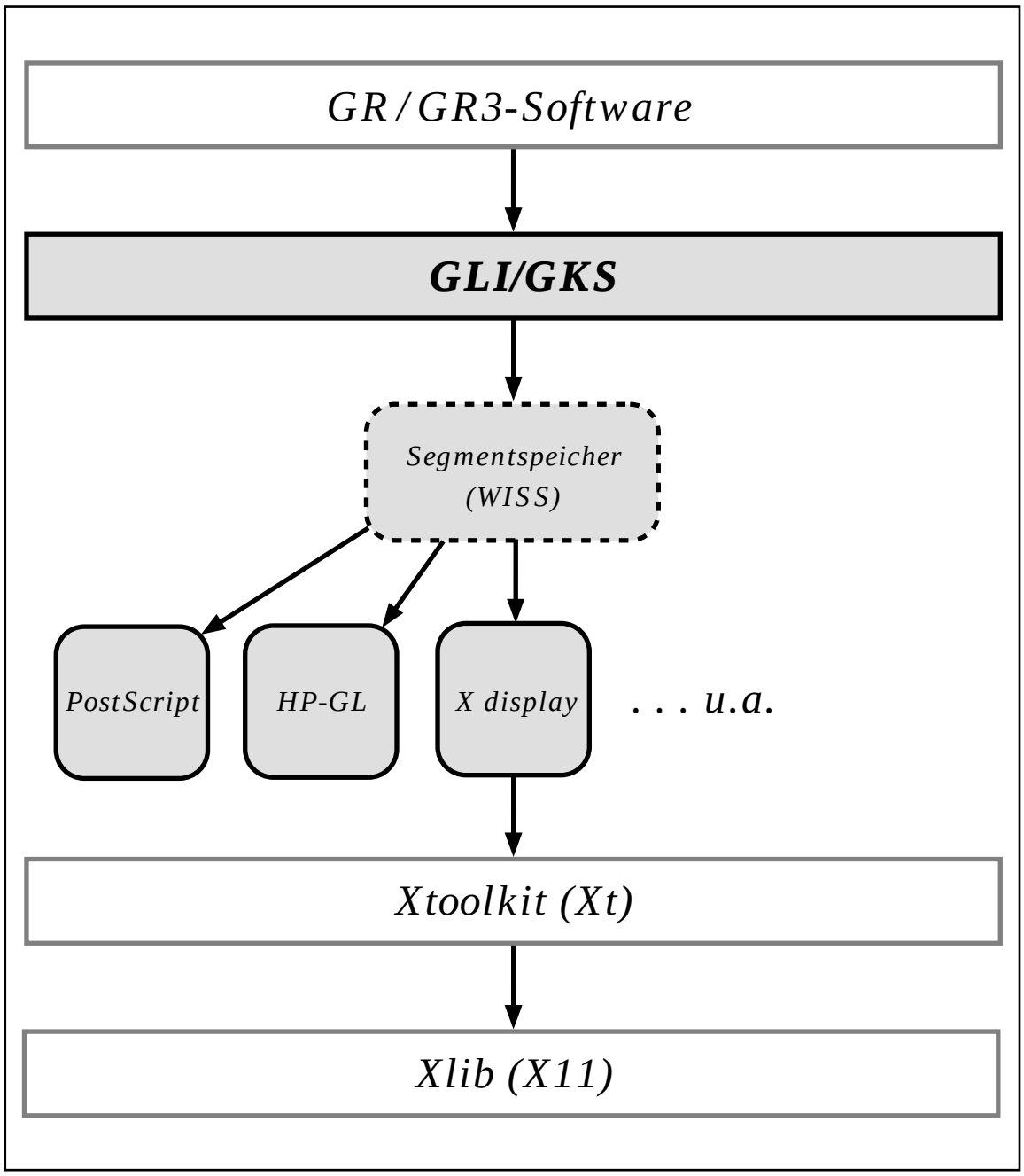
```
f77 -o myprog myprog.f -lgus -lgks
```

  sonst:

```
f77 -o myprog myprog.f -L/usr/local/gli -lgus \
-lcrtl -lfrtl -lgks -lXt -lX11
```

4 Aufruf von GR-Unterprogrammen aus GLI

Funktionsweise



GR Kommandos

GR Name	GLI/GR Kommando (grsoft)
GR00DG	**gr 00dg**
GR90DG	**gr 90dg**
GRARRW	**gr arrw** *xp yp xtip ytip alen awid icode*
GRAXLIN	**gr axlin** *x1 y2 x2 y2 s1 s2 links kindax*
GRAXLOG	**gr axlog** *x1 y2 x2 y2 s1 s2 links kindax*
GRAXS	**gr axs** *lopt opt ltxtx textx ltxtxy textxy*
GRBLD	**gr bld** *xcm ycm isk jsk xmin xmax ymin ymax nkurv*
GRCHN	**gr chn** *xx yy m nr*
GRCHNC	**gr chnc** *xx yy m nr*
GRCHRC	**gr chrc** *height angle idummy*
GRCLP	**gr clp** *iclip*
GRCRCL	**gr crcl** *xm ym r ph1 ph2*
GRDEL	**gr del**
GRDN	**gr dn** *idin xcm ycm*
GRDRAX	**gr drax** *ltx tx lty ty ltz tz szt*
GRDRDM	**gr drdm** *drdmpa nrow tab xx yy*
GRDRDU	**gr drdu** *drdmpa nrow tab xx yy*
GRDRHS	**gr drhs** *drdmpa npn pnkt xx yy*
GRDRKU	**gr drku** *drdmpa n xyz*
GRDRLG	**gr drlg** *drdmpa textx texty textz iopt x y z ix iy iz*
GRDRNE	**gr drne** *parm nrow xyz*
GRDRW	**gr drw** *x y*

GR Kommandos (Forts.)

GR Name	*GLI/GR Kommando (*grsoft*)*
GRRDWS	**gr drws** *x y nr*
GRDSH	**gr dsh** *a1 a2 a3*
GREND	**gr end**
GRENDE	**gr ende**
GRFILL	**gr fill** *n xx yy istyle itype*
GRFONT	**gr font** *ifont*
GRFRBN	**gr frbn** *ifu ika iks ira iri*
GRGFLD	**gr gfld** *imax f istax incx nx istay incy ny kind hoehe breite winkel dicke ier*
GRJMP	**gr jmp** *x y*
GRJMPS	**gr jmps** *x y nr*
GRHHFL	**gr hhfl** *nx ix vx ny iy vy nw vw ivfa ndx tab intact*
GRHHNL	**gr hhnl** *n1 tab n3 xx n4 yy l1 wert int option istax incx istay incy*
GRHPAN	**gr hpan** *n1 tab n3 xx n4 yy l1 wert int option istax incx istay incy*
GRLGND	-
GRLN	**gr ln** *xx yy m*
GRLNCN	**gr lncn** *xx yy m*
GRDRDU	**gr drdu** *drdmpa nrow tab xx yy*
GRMRKS	**gr mrks** *height*
GRMSKN	**gr mskn**
GRMSKF	**gr mskf**
GRNWPN	**gr nwpn** *i*

GR Kommandos (Forts.)

GR Name	GLI/GR Kommando (grsoft)
GRNXTF	**gr nxtf**
GRPTS	**gr pts** *xx yy m nr*
GRSCAX	**gr scax** *cx cy czl czr*
GRSCDL	**gr scdl** *n x y lintyp cha nh xa yh*
GRSCLC	**gr fill** *xa ya xb yb*
GRSCLP	**gr sclp** *xcm ycm rahmen*
GRSCLV	**gr sclv** *xa ya xb yb*
GRSHD	**gr shd** *x1 y1 x2 y2 abst winkel n1 n2*
GRSHOW	**gr show**
GRSPHR	**gr sphr** *ns nr vs infa chi psi zp hs ipo mdm ier*
GRSPTS	**gr spts** *int*
GRSTRT	**gr strt** *camera ddnumb*
GRTXT	**gr txt** *x y ltext text*
GRTXTC	**gr txtc** *ltext text*
GRVAR	**gr var** *x y var n koax*
GRVFLD	**gr vfld** *ndim1 va vb ifa istax incx nx istay incy ny kind hoehe breite winkel dicke ier*
GRWIN	**gr win** *x1 y1 x2 y2 idummy*
KURVEF	**gr kurvef** *x y ist isy*
PSKURF	**gr pskurf** *x y ist form z*
PSKURL	**gr pskurl** *x y ist form z ksk*
SKURF	**gr skurf** *x y ist form z*
SKURL	**gr skurl** *x y ist form z ksk*

Beispiel (GR-Software)

```
!
! teslog.gli
!
gr strt -1 -1
l := 6
m := 25
n := l*m
i := 1..150
x := i/m
y = log(2**x+sin(9*x))
gr sclc 3 3 36.5 25.7
gr chrc .5 0 18
gr sclv 0 2 l 2**l
gr dsh 1 .5 1
gr axsl 0 1001 1
gr sclv 0 2 l 2**l
gr axsl 0 -12 1
gr axs -1 "X=-1,F=(2),M=3.,U=1" -1 " " 0 " "
gr axs -1 "X=-2,F=(2),M=3.,U=1,XLB=-2,I" -1 " " 0 " "
gr clp 1
gr nwpn 4
gr dsh 1 0 1
gr spts 22
gr ln x y n
gr clp 0
gr font -51
gr nwpn 1
gr sclc 0 27.7 39.5 28.7
gr sclv 0 0 1 1
gr chrc .7 0
gr gstxal 2 0
gr txt .5 0 -1 "Lin. und log.(y) Achse in normaler Lage,
       Clipping (GRAXS(L), GRCLP)"
gr gstxal 0 0
del var l m n x
del fun y
gr nxtf
gr end
```

Anmerkungen

- *GLI/GKS arbeitet in Verbindung mit der GR-Software im GKSGRAL-Kompatibilitätsmodus*
- *Die Benutzung einiger GR-Funktionen aus dem GLI heraus ist wegen fehlender Unterstützung mehrdimensionaler Felder teilweise mühsam*
- *Das interaktive Arbeiten mit der GR-Software im GLI ersetzt keine FORTRAN-Programmierung ('rapid prototyping')*
- *Nach Aufruf der* **grend** *bzw.* **grende** *Funktionen befindet sich das GKS im Zustand* **GKOP**, *d.h., kein Arbeitsplatz ist offen*

5 Interaktive Zeichnungserstellung mit GLI/SIGHT

Simple Interactive Graphics Handling Tool (SIGHT)

- *GLI/**SIGHT** ist ein interaktiver objekt-orientierter Zeichnungseditor*
- ***SIGHT** arbeitet nach dem WYSIWYG-Prinzip («**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et»)*
- ***SIGHT** bietet sowohl eine Kommando- als auch eine Point&Click-Schnittstelle (Motif)*
- *Die graphische Ein-/Ausgabe erfolgt ausschließlich über GLI/GUS bzw. GLI/GKS*
- ***SIGHT** verwaltet eine vom GKS Segmentspeicher unabhängige Objektliste für graphische Ausgabeelemente, Attribute und Transformationen*

Die GLI/SIGHT Motif-Oberfläche

- *Die GLI/**SIGHT** Funktionen sind gegliedert in*
  - *Dateifunktionen*
  - *Bearbeitungsfunktionen*
  - *Zeichenfunktionen*
  - *Attributfunktionen*
  - *Ansichtfunktionen*
  - *Einstellungsfunktionen*
- *Die GLI/**SIGHT** Motif-Oberfläche ist in englischer oder deutscher Sprache verfügbar*

```
% setenv GLI_LANG ae_AE
```

oder

```
% setenv GLI_LANG de_DE
```

```
% gli
gli> sight
```

Dateifunktionen

Funktion	Bedeutung
Zeichnung erstellen...	*neue Zeichnung erstellen*
Zeichnung öffnen...	*existierende Zeichnung öffnen*
Speichern als...	*Zeichnung speichern*
Zeichnung schließen	*Zeichnung schließen*
Werte von Datei lesen...	*XY-Werte von einer Datei lesen*
Werte in Datei speichern...	*XY-Werte auf eine Datei schreiben*
Importieren...	*Importieren von CGM-Dateien*
Zeichnung drucken	*Zeichnung drucken*
Druckdatei erstellen	*Druckdatei erstellen (**name**.**eps**)*
Abbrechen	*SIGHT verlassen*
Beenden	*Zeichnung speichern und SIGHT verlassen*

- *Die Druckausgabe kann über die* **GLI_LPR** *Umgebungsvariable eingestellt werden*

```
% setenv GLI_LPR lpr -Plp
% gli
gli> sight
```

Bearbeitungsfunktionen

Funktion	Bedeutung
Zeichnung neu aufbauen	*Zeichnung neu aufbauen*
Auswählen	*Objekte auswählen*
Auswahl aufheben	*Auswahl aufheben*
Ausschneiden	*ausgewählte Objekte löschen*
Einfügen	*Objekte kopieren*
In den Vordergrund setzen	*Objekte in den Vordergrund setzen*
In den Hintergrund schieben	*Objekte in den Hintergrund schieben*
Verschieben	*Objekte verschieben*
Löschen	*Objekte löschen*
Aufnehmen	*Objektdaten übernehmen*
Digitalisieren	*Digitalisieren vom Tableau*
Zeichnung löschen	*Zeichnung löschen*

Zeichenfunktionen

Funktion	Bedeutung
Öffne Zeichenhilfen...	*Zeichenhilfen als Piktogramme darstellen*
Linie	*Zeichnen von Linien*
Spline	*Zeichnen von Splines (natürlich, geglättet, B-Spline)*
Fehlerbalken	*Zeichnen von Fehlerbalken (vertikal, horizontal)*
Symbol	*Zeichnen von Symbolen*
Text	*Textausgabe*
Füllfläche	*Zeichnen von gefüllten Flächen*
Balken	*Zeichnen von Balken*
Bild erstellen	*Bild von XY-Werten erstellen*
Achsen	*Achsenkreuz zeichnen*
Gitter	*Gitter zeichnen*

Attributfunktionen

Funktion	Bedeutung
Öffne Zeichenattribute...	*Zeichenattribute als Piktogramme darstellen*
Linienart	*Strichlierungsart für Linien*
Linienstärke	*Dicke der Linien (in Punkten)*
Linienfarbe	*Linienfarbe*
Markierungsart	*Symbol auswählen*
Markierungsgröße	*Größe der Symbole ändern (in Punkten)*
Markierungsfarbe	*Symbolfarbe*
Schriftart	*Zeichensatz auswählen*
Textgröße	*Texthöhe (in Punkten)*
Textausrichtung	*vertikale Textjustierung einstellen*
Textschreibrichtung	*Orientierung von Texten*
Textfarbe	*Textfarbe*
Füllart	*Ausfüllungsart festlegen*
Füllindex	*Ausfüllungsindex einstellen*
Füllfarbe	*Farbe für gefüllte Flächen*

Ansichtfunktionen

Funktion	Bedeutung
Darstellungsfläche auswählen	*Darstellungsbereich festlegen*
Fenster skalieren	*Benutzerkoordinatenbereich festlegen*
Format	*Ausgabeformat (quer, hoch)*
Skalierungsart	*Skalierungsart (linear, logarithmisch)*
Transformation auswählen	*Transformationsnummer*

- *Bei Mehrfachbildern muß jedem Teilbild eine separate Transformation zugeordnet werden*
- *Die Textpositionierung bezieht sich grundsätzlich auf normalisierte Gerätekoordinaten (Transformation 0)*

6 Bildverarbeitung mit GLI/IMAGE

Bildverarbeitung (GLI/IMAGE)

- *GLI/**IMAGE** ist ein interaktives System zur Bildverarbeitung*
- ***IMAGE** arbeitet nach dem WYSIWYG-Prinzip («**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et»)*
- ***IMAGE** bietet sowohl eine Kommando- als auch eine Point&Click-Schnittstelle (Motif)*
- *Die graphische Ein-/Ausgabe erfolgt ausschließlich über GLI/GKS*
- ***IMAGE** basiert auf dem **pbm**-Dateiformat (**p**ortable **b**it**m**ap)*

Funktionalität

- *Ein-/Ausgabe der verschiedenen **pbm**-Formate*
  - ***p**ortable **b**it**m**ap (**pbm**)*
  - ***p**ortable **g**rey**m**ap (**pgm**)*
  - ***p**ortable **c**olor**m**ap (**pcm**)*
  - ***p**ortable **p**ix**m**ap (**ppm**)*
- *Konvertieren zwischen den verschiedenen **pbm**-Formaten*
- *Geometrische Bildoperationen*
  - *Vergrößern, Verkleinern*
  - *Drehen*
  - *Spiegeln (vertikal, horizontal)*
  - *Scheren (in X- oder Y-Richtung)*

- *Bildmanipulation*
  - *Normalisieren*
  - *Angleichung der Farb- oder Grauwerte (histogram equalization)*
  - *Kontrastverschärfung*
  - *Median-Filter*
  - *Gradientenfilter (vertikal, horizontal)*
  - *(RGB-) Gamma-Korrektur*
  - *Invertieren*
  - *Erkennung von Ecken (edge detection)*
  - *Hervorhebung von Ecken (enhance edges)*
  - *Zuordnung einer Farbtabelle*

- *Transformationen*
  - ***F**ast **F**ourier **T**ransformation (**FFT**) von Grauwert-Bildern*
  - *verschiedene Filter anwendbar (Tiefpass, Hochpass, Bandpass, Notch)*
  - *graphische Auswahl der Fourier-Koeffizienten über Kreisscheiben*
  - *Nutzung schneller FFT-Algorithmen (**D**igital E**x**tended **M**ath **L**ibrary)*

Die GLI/IMAGE Motif-Oberfläche

Gruppe	Funktion	Bedeutung
File	Read Image ...	*Lesen von pbm-Bildern*
	Write Image ...	*Speichern von pbm-Bildern*
	Print Image	*Bild drucken*
	Capture Image	*Druckdatei erzeugen*
	Quit	*IMAGE verlassen*
Edit	Copy	*Bild kopieren*
	Delete	*Bild löschen*
	Rename	*Bild umbenennen*
	Flip	*Bild spiegeln*
	Rotate	*Bild drehen*
	Shear	*Bild scheren*
	Import	*Bild aus GLI importieren*
	Export	*Bild in GLI exportieren*
Modify	Resize	*Vergrößern, Verkleineren*
	Invert	*Invertieren*
	PPM	*in **ppm**-Format konvertieren*
	PCM	*in **pcm**-Format konvertieren* *Anzahl Farben* *Farbtabelle*
	PGM	*in **pgm**-Format konvertieren* *Anzahl Graustufen* *(inverse) FFT* *Filter*
	PBM	*in **pbm**-Format konvertieren* *Schwellwert*

(Forts.)

Gruppe	Funktion	Bedeutung
Filter	Contrast	*Kontrast verschärfen*
	Enhance Edges	*Ecken hervorheben*
	Gamma	*Gamma Korrektur*
	Histogram Equalization	*Farb-/Grauwertverteilung angleichen*
	Median	*Medianfilter*
	Normalize	*Normalisieren*
	Gray	*Gauwert-Bildtransformation* *Erkennung von Ecken* *Gradienten*
	Color	*Gamma Korrektur* *des Rot-Anteils* *des Grün-Anteils* *des Blau-Anteils*
Activities	Display Image	*Darstellen eines Bildes*
	Display	*Bild darstellen* *in das Benutzerfenster* *in Originalgröße* *in das Zeichenfenster*
Help	Help	*Hilfe*

7 GLI/SimplePlot

GLI/SimplePlot

- *GLI/**SimplePlot** ist eine tasten-gesteuerte Umgebung zur Signalverarbeitung*
- *Aufruf*

  ```
  gli> simpleplot
  ```

- *Funktionalität*
  - *FFT's*
  - *Hoch-, Bandpassfilter*
  - *Ausschnittbildung*
  - *Glätten von Signal(abschnitten)*
  - *Zooming*

8 Benutzung von Tcl/Tk aus GLI

Tcl/Tk

- ***Tcl*** *ist eine prozedurale Scriptsprache (**T**ool **c**ommand **l**anguage)*
- ***Tk*** *ist eine Erweiterung zu* ***Tcl*** *zur Erstellung Motif-basierender Benutzeroberflächen*
- ***GLI*** *verbindet Tcl, Tk und eigene Kommandos zu einer komfortablen graphischen 'shell'*
- *Registrierung eines* ***Tcl****-Interpreters im GLI*

```
gli> set tcl
```

- *Die Registrierung gilt nur für die aktuelle Kommandoebene*
- ***GLI*** *Kommandos sollten (zur Unterscheidung von* ***Tcl/Tk*** *Kommandos) mit einem Doppelpunkt (:) eingeleitet und/oder in Großbuchstaben eingegeben werden*
- ***GLI*** *kann direkt auf* ***Tcl****-Variablen zugreifen*

```
gli> : print $tk_version
```

- *Die graphische Ausgabe (**GLI/GKS**) kann in ein **Tk** Widget geleitet werden*

```
gli> canvas .w -width 500 -height 500
gli> pack .w -side left
gli> update
gli> set w \"[winfo screen .w]![winfo id .w]\"
gli> : DEFINE LOGICAL GKSconid $w
gli> : GKS OPEN_WS TERMINAL 212
```

- *Bei komplexeren **Tcl/Tk** Anwendungen sollte der **GLI** (wegen der besseren Performance) mit der* **`-file`** *Option aufgerufen werden*

```
% gli -file tcl-script
```

Tcl/Tk Beispiel

```
! browse.gli
!
set tcl
#
proc demo {script} {
  init
  @ 'GLI_DEMO'$script
}
#
proc init {} {
  global w
  : gks emergency_close
  : gks open_gks
  : gks set xform wc
  : gks set viewport  0.2,0.9 0.2,0.9
  : gks set asf individual
  : gks set pmark size  2
  : gks set pmark type asterisk
  : gks set text font 3 string
  : gks set text height 0.027
  : define logical GKSconid $w
  : gks open_ws terminal 212
  }
#
wm title . "GLI Tcl/Tk Demonstration"
#
button .ascii -text "ASCII Character Set" -command {demo ascii.gli}
button .pie -text "Simple Pie Chart" -command {demo pie.gli}
button .bars -text "Simple Bar Graph" -command {demo bars.gli}
button .greek -text "Greek Letters" -command {demo greek.gli}
button .fill -text "GKS Fill Styles" -command {demo fillstyles.gli}
button .surfcont -text "Surface Plot" -command {demo surfcont.gli}
button .face -text "Face Plot" -command {demo face.gli}
button .rancon -text "Random Contours" -command {demo rancon.gli}
button .flat -text "Light Shading" -command {demo flat.gli}
button .exit -text "Quit" -command {: exit}
pack .ascii .pie .bars .greek .fill .surfcont .rancon .flat .exit \
-ipadx 2m  -fill x
#
canvas .w -width 500 -height 500 -background white
pack .w -side left
#
update
#
set w \"[winfo screen .w]![winfo id .w]\"
```

9 Aufruf von RPC-Programmen

- ***RPC** steht für «**R**emote **P**rocedure **C**all»*

- *Die **RPC**-Technologie erlaubt den Aufruf von externen Unterprogrammen (auf dem lokalen oder einem entfernten Rechner)*

- *Zur Registrierung eines **RPC**-Programms steht die* **gli_registerrpc** *Funktion zur Verfügung*

```
main(argc, argv
    int argc;
    char **argv;
{
    gli_registerrpc(argc, argv, my_proc);
}
```

- *Aufruf eines **RPC**-Programms mit dem* **RPC** *Befehl*

gli> RPC *command [var-name,...]*

RPC-Beispiel

```
#include <stdio.h>
#include <string.h>

int my_proc(message, argc, sizes, data)
    char *message;
    int argc;
    int *sizes;
    float **data;
{
    register int i;
    register float *d;

    if (!strcmp(message, "data"))
    {
        if (argc == 1)
          {
              d = data[0];
              for (i=0; i<*sizes; i++)
                    d[i] = i*i;
        }
        else
            sprintf(message, "?Invalid arguments");
    }
    else
        sprintf(message, "?Unknown command", message);
}

main(argc, argv)
    int argc;
    char **argv;
{
    gli_registerrpc(argc, argv, my_proc);
}
```

```
gli> x := 1..100
gli> y := x
gli> ! user@hostname: sim :1 uid1 uid2 ... uidn &
gli> ! define logical GLI_SERVICE hostname:1
gli> rpc data y
gli> plot x y
```

10 Darstellung von CGM Dateien

Aufruf

- *Der CGM-Viewer wird mit dem* **cgmview** *Kommando aufgerufen*
- **cgmview** *bietet sowohl eine Kommando- als auch eine Motif-Oberfläche*
- *Die Motif-Oberfläche wird aktiviert, wenn man* **cgmview** *ohne Dateinamen aufruft*
- *Mit der* **-h** *Option kann eine Kurzbeschreibung der Kommando-Schnittstelle angezeigt werden*

```
Usage:
cgmview [-c] [-h] [-m] [-p picture] [-t wstype] file
         -c   Specifies the input file to be a clear text format CGM file,
              default is binary format.
         -h   Prints this information.
         -m   Tells cgmview to maximize the output window.
 -p picture   Specifies the picture number, picture.
  -t wstype   Use GKS workstation type wstype.

The present workstation types recognized by gligksm are:
     16:       VT330               7:    CGM Binary
     17:       VT340               8:    CGM Clear Text
     72:       TEK 401x           38:    DEC LN03+
     82:       TEK 42xx       51, 53:    HP-GL Graphics Plotter
    201:       TAB 132/15-G   61, 62:    PostScript (b/w, color)
    204:       Monterey       63, 64:    CompuServe GIF dump (b/w, color)
    207:       IBM/PC             92:    DEC LJ250 Companion Color Printer
    210, 211:  X display    103, 104:    Portable BitMap (72, 75 dpi)
    214:       X display w\ Sun rasterfile dump
    215:       X display w\ PCM dump
    217:       X display w\ frame grabber
```

Beispiele (CGM-Viewer)

- *Darstellen einer binär-kodierten CGM-Datei*

  ```
  % cgmview gli.cgm
  ```

- *Darstellen des zweiten Bildes einer CGM-Datei in Klartextformat*

  ```
  % cgmview -c -p 2 gli.cgm
  ```

- *Konvertieren einer binär-kodierten CGM-Datei in eine farbige PostScript-Datei*

  ```
  % cgmview -t 62 gli.cgm > gli.ps
  ```

- *Aktivieren der* **cgmview** *Motif-Oberfläche*

  ```
  % cgmview
  ```

- *Drucken einer CGM-Datei in Klartextformat*

  ```
  % cgmview -c gli.cgm | lpr -Plp
  ```

11 Darstellung von GKSM Dateien

Aufruf

- *Der GKSM-Viewer wird mit dem* **gligksm** *Kommando aufgerufen*

- *Mit der* **-***h Option kann eine Kurzbeschreibung der Kommando-Schnittstelle angezeigt werden*

```
Usage:
gligksm [-Oorientation] [-factor f] [-h] [-t wstype] [-v] file
         -Oorientation   Select the orientation (landscape, portrait).
             -factor f   Specifies the magnification factor.
                    -h   Prints this information.
             -t wstype   Use GKS workstation type wstype.
                   -v    Prints information about each GKSM item.

The present workstation types recognized by gligksm are:
     16:       VT330                7:    CGM Binary
     17:       VT340                8:    CGM Clear Text
     72:       TEK 401x            38:    DEC LN03+
     82:       TEK 42xx       51, 53:    HP-GL Graphics Plotter
    201:       TAB 132/15-G   61, 62:    PostScript (b/w, color)
    204:       Monterey       63, 64:    CompuServe GIF dump (b/w, color)
    207:       IBM/PC             92:    DEC LJ250 Companion Color Printer
    210, 211:  X display    103, 104:    Portable BitMap (72, 75 dpi)
    214:       X display w\ Sun rasterfile dump
    215:       X display w\ PCM dump
    217:       X display w\ frame grabber
```

Beispiele (GKSM-Viewer)

- *Darstellen einer GKSM-Datei*

```
% gligksm gli.cgm
```

- *Konvertieren einer GKSM-Datei in eine farbige PostScript-Datei*

```
% gligksm -t 62 gli.gksm > gli.ps
```

- *Drucken einer GKSM-Datei in Porträt-Format*

```
% gligksm -Olandscape -t 62 gli.gksm | lpr -Plp
```