Graph2D

Kai Nickel

COLLABORATORS

	TITLE : Graph2D		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Kai Nickel	July 22, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Graph2D

1.1 Table of contents

```
                           G r a p h   2 D

                    Functionplotter and -analyzer
                      Version 2.10 (03.03.1995)
                         Author: Kai Nickel

Note: This documentation has not been completely updated for this Version
      yet. Some of the new features of Graph2D are not described here.
      So please read chapter "3.2 History" to know about the changes!
```

1.2 Short description

1.1 Short description

Graph2D is a function-plotter that can create graphs of math. functions and is able to discuss them. That includes zero-points, extrema, turning-points, monotony and symmetrie. You can derive and simplify the functions, calculate value-tables, integrate functions numerically, create tangents, etc. As a special feature functions can be presented in a SIRDS-picture giving you the impression of "true" three- dimensionality. The Graph2D package contains an installer-script and online-help.

Feature overview:

- Plots functions into highly configurable coordinate-systems
- Generates SIRDS for three-dimensional functions
- Work with many functions in many windows at the same time
- Generation of discussions (zero-points, extrema, turning-points, monotony, symmetry)
- symbolic derivation and numerical integration of funktion-terms
- use your own functions to define other functions
- Generation of value-tables and tangents
- Shows functions in "mathematical notation"
- system- and user-friendly MUI-interface
- Printing of graphs and discussions
- Online-Help as AmigaGuide
- Installation with the Commodore-Installer
- uncrippeled shareware without limitations

System requirements:

- Amiga with Kickstart2.0 or higher
- installed MUI-System version 2.1 or higher

1.3 Author

1.2 Author

Graph2D and the documentation were written by

```
                  Kai Nickel

  Mail          : Herzogstrasse 29
                  67435 Neustadt
                  Germany

  eMail         : kai@rpsbbs.pfalz.de (Kai Nickel)
                  un7x@rz.uni-karlsruhe.de
```

I would be glad about bug reports or ideas concerning future development of Graph2D.

About this documentation:

As you can see my English is not very good, and I know that there are a lot of mistakes in this documentation. Especially the mathematical expressions are often pure "inventions" because I could not find them in my dictionaries. But despite this I hope that you can at least figure out what I wanted to say... If anybody wants to put this documentation into "real" English or even translate it into any other language: please contact me!

How to get updates:

I will send updates of Graph2D to Aminet and FRAS. You may also call one of the following (german) BBSs:

```
  - RPSBBS (06329-1624)
    Login "GAST", path "/Lokal/Support/Amiga/Graph2D"
    (my username is "Kai")

  - SchneeII (06347-92071)
    Login "GAST"
    (my username is "Kai")
```

1.4 Copyright

1.3 Copyright

Graph2D is Copyright © 1994-95 Kai Nickel.

Graph2D is shareware. The author owns the copyright for this programm. Anybody who wants to use Graph2D seriously has to register himself at the author.

Spreading of the unregistered version is permitted as long as there are no commercial interests connected to the spreading. The price of a floppy-disk with Graph2D on it must not be higher than US$3. The inclusion into FD-series is hereby permitted, when nothing of the above said things is injured. You must not change any of the files in the original~archive or remove or add files from or to it. It must remain complete and unchanged!

The author cannot guarantee the correct function of Graph2D and cannot be made responsible for any negative consequences that may result from the use of Graph2D. Updates or bugfixes are not guaranteed.

Graph2D was developed with the Amiga-Oberon-Compiler V3.20 by the A+L AG. The license of that compiler prohibits the use of Graph2D in the military sector.

Graph2D uses the MUI-system by Stefan Stuntz.

MagicWB and some of the icons in the Graph2D package are copyright by Martin Huttenloher.

The SIRDS algorithm was developed by Kilian Singer and has been included to Graph2D with his permission.

1.5 Registration

1.4 Registration

Graph2D is shareware. Everybody using the program someway serious, is asked to register himself at the author.

The registration fee is DM 20 or US$15.

Anybody using Graph2D longer than a certain evaluation period (lets say 2 weeks) without registering injures my copyright and breaks the license! The unregistered version is not limited in any way but there are nerv-requesters that appear sometimes and remind you about registering. Please concern that I have invested many month of work in Graph2D and the US$15 are really not that much money. I just want to have a kind of positive feedback for my efforts. This may also motivate me for further developments or bugfixes.

After registering you get a personal keyfile which gives you the right to use Graph2D and future updates (if there are any) permanently. Registration of course also makes the nerv requesters disappear!

How to register:

You send me a letter containing the registration fee and a filled out Graph2D.RegForm because i need your address to generate your personal keyfile. After a couple of days you should receive a letter from me containing a floppy-disc with your keyfile on it and an installation guide. If anything went wrong and you did not get a response from me after lets say 4 weeks you should consider contacting me with another mail (don’t get me wrong: You shall not send the money twice :-))

Thank you for supporting the shareware idea!

1.6 Installation

```
1.5 Installation


Normal:

The best way to get Graph2D installed is to start the script
"Graph2D-Install". Look for the script in the drawer of your preferred
language! You can start Graph2D after the installation just by clicking it
from the workbench.

Don’t be afraid: If you chose "expert user" at the installation you may
                 confirm really every action!


Manually:

If you don’t have the Commodore installer you alternatively can also
install Graph2D by hand:

  - Create a Graph2D drawer and copy the main program into it

  - Copy the guide and the reg-form of your preferred language into
    the drawer

  - Copy the "GarbageCollector.library" to the "LIBS:" drawer of your
    system or to PROGDIR:Libs/.

  - Copy the "Graph2D.catalog" to your system’s "Locale:Catalogs/English"
    or to PROGDIR:Catalogs/Libs/

  - optional: Copy the subdirectories "Functions" and "Graphs" to your
              drawer

  - optional: Replace the icons by the icons of the "MWBIcons" drawer


Systemrequirements:

To use Graph2D you need an Amiga with Kickstart 2.0 or higher (2.1 for
localisation). You also have to have the Mui-System by Stefan Stuntz
Version 2.1 or higher installed. Graph2D won’t work without MUI!


Originalarchive:

  The original Graph2D-archive contains the following files. If files
  have been added or removed then you have an illegal copy!

  Graph2D                          - main program
  Graph2D-Install                  - installation script
  Docs/Deutsch/Graph2D.guide       - german documentation/onlinehelp
  Docs/Deutsch/Graph2D.RegForm     - german registration form
  Docs/Deutsch/Graph2D.LiesMich    - german description
  Docs/English/Graph2D.guide       - documentation/onlinehelp
  Docs/English/Graph2D.RegForm     - registration form
  Docs/English/Graph2D.ReadMe      - description
  Docs/English/Product-Info        - description in fish-format
  Catalogs/english/Graph2D.catalog - english catalog
  Libs/garbagecollector.library    - library
  Libs/GarbageCollector.LiesMich   - german infotext for the library
  Graphs/Standard.2D               - 2D-standard coordinatesystem
  Graphs/Standard.SIRDS            - SIRDS-standard coordinatesystem
  Graphs/Trigonometrie.2D          - example of a trigonometrical system
  Funktions/Bsp2D.fkt              - example of a functionlist
  Funktions/Bsp3D.fkt              - example of a list of 3D-functions
  MWBIcons/ClickForColors          - MagicWB-palette (try it!)
  MWBIcons/Graph2D.info            - MagicWB-style icon
  MWBIcons/Graph2D.guide.info      - MagicWB-style icon
  MWBIcons/Graph2D.RegFrom.info    - MagicWB-style icon
  MWBIcons/Drawer.info             - MagicWB-style icon
  MWBIcons/Funktions.info          - MagicWB-style icon
  MWBIcons/Graphs.info             - MagicWB-style icon
  MWBIcons/Catalogs.info           - MagicWB-style icon
  MWBIcons/Libs.info               - MagicWB-style icon
```

1.7 Q&A

1.6 Questions and answers

So how exactly can I display the graph of a function?

First you have to generate a new function via "New", enter the functionterm (e.g. "f(x)=sin(x)^2"), press <return> to interpret the function and open a "New Graph". To make the function be plotted into the graph-window you now have to mark the function as "plot" and "Redraw" the graph to make the cange visible. The function should now be visible - of course only, if parts of the function are inside the choosen interval of the graph. You can change the visible interval, if you "edit" the graph.

1.8 Basics

2.1 Basics

Graph2D should be easy to use and look nice with every font and configuration thanks to MUI.

This documentation is also available as context-sensitive online help. You can read the help just by pressing the "Help"-key on the keyboard or by selecting "Help..." from the "Project"-menu

After the start by double-clicking the Graph2D icon a main-window - the so called functioneditor - opens in which you can enter and edit functions.

If there are functions present, you can look at them in graph-windows or mathematically analyze them.

1.9 Project-menu

2.1.1 The project-menu

The project-menu is available from the function-editor, from graph- and notation-windows.

Project

```
  New Graph...        - Creates a new graph-window. The graph type
                        (2D-graph or SIRDS-graph) has to be specified
                        in a little requester.
  Presets...          - Opens the presets-editor
  Help...             - Opens the context-sensitive online-help like
                        pressing the HELP-key
  About...            - Opens an info-window
  Quit                - Quit Graph2D
```

1.10 Function editor

2.2 Function editor

The funktioneditor is a window, in which alle present functions are listed and can be edited.

A function from the list-gadget appears in the function gadget by clicking on the function. If finish the entered or modified string with pressing <return>, the functionterm gets automatically interpreted.

If the interpreter detects an error in the function, the error gets displayed in the "error" gadget below, and the "bad" function will appear in italics in the function-list. Of course such a bad function cannot be plotted or analyzed!

Pressing the HELP-key over the function or error gadget shows immediately the "Functions"-chapter of the documentation where you can find information about the function-syntax.

In the string-gadget "Graph" with the popup-list all existing graphs are shown by their names. You can give an individual name to every graph. When you double-click a graph in the list it will appear in the foreground.

For a selected function you can define, whether and in which color and pattern it should be visible in the selected graph. A double-click to a function toggles the visible-mark as if you had clicked onto the visible-gadget. A function that appears in the selected graph ist marked with a preceding ">" in the function-list.

Changes been made on a visible function will then be executed when you click "Redraw" in the graph-window - not before! This allows you to do more than one change at once without a time-consuming redraw after every change.

The image-buttons stand for the following things you can do with the selected function:

math. notation	- Opens the notation-window with the selected function
Derive	- The derivation of the selected function will be added to the list
Simplify	- A simplified aequivalent of the selected function will be added to the list
Diskussion	- Initiates a selectable discussion
num. integration	- Calculates the integral
Value-table	- Creates a value-table
Tangente	- Creates a tangente

As long as the function-editor is active, the following menus can be selected:

Project	- The project-menu

Functions

Clear all	- The functionlist will be deleted
Load ...	- Load a functionlist
Append...	- Append a functionlist (without deleting the actual one)
Save as...	- Save the functionliste

Edit

New	- A new, "empty" function will be added to the list at the selected position and can be immediately edited
Cut	- The selected function will be cut out of the list
Paste	- A former cut-out function will be added to the list at the selected position

Miscellanious

Simplify	- like the according image-button
Derive	- "
Diskussion	- "
num. integration...	- "
Value-table...	- "
Tangente ...	- "
math. notation...	- "

1.11 math. notation

2.2.1 Mathematical notation

In a window the selected function will be displayed in a more readable, a more "normal" way. Divisions are shown by horizontal lines and exponents are really at high-position. I call this the "mathematical notation"...

If the notation exceeds the visible range of the window you can scroll around with the scrollbars. The window gets closed by pressing the close-gadget. You can have opened more than one notation-windows at the same time an work in another part of Graph2D without problems.

You can use the following menus:

Project	- The project-menu

Notation

Print	- Prints the notation via Workbench-drivers and settings. You can edit a scaling factor here.

1.12 Discussion

2.2.2 Discussion

Graph2D can discuss fuction with one variable in a selectable test-interval. This interval should always be choosen as small as possible for best results. Graph2D shows a report of the results in a textviewer and may be printed from there.
A dicussion may take (especially on 68000er Amigas) sometimes quite a long time and it is also very recommended to verify the results by having a look to the graph of the function. So you probably will see soon, that certain types of zero-points will sometimes really not be detected.

A complete discussion consists of the following parts:

Zero-points: $f(x) = 0$

The zero-points of the selected function will be detected. In addition to the x-value you will also get the following information about the type of the zero-point:

-+	$f'(x) > 0$
+-	$f'(x) < 0$
++	$f'(x) = 0$
--	$f'(x) = 0$

Extrema: $f'(x) = 0$

min	$f''(x) > 0$
max	$f''(x) < 0$

```
Turning-points: f''(x) = 0

  left->right   f'''(x) < 0
  right->left   f'''(x) > 0

Monotony:

  The test-intervall will be divided into parts where the selected function
is:

  monotonous raising       :  f'(x) >= 0
  monotonous falling       :  f'(x) <= 0
  strong monotonous raising:  f'(x) >  0
  strong monotonous falling:  f'(x) <  0

Symmetry:

In this part the selected function will be tested whether it is

  - symmetric to the origin: f(x) = - f(-x)
  - symmetric to the y-axis: f(x) =   f(-x)

If both is not correct, then "No symmetry" will be reported.
```

1.13 num. integration

```
2.2.3 numerical integration


  Allows you to calculate the integral of the selected function in an
intervall you have to specify via the "Simpson"-formula.

  a, b: Integrationlimits
  k   : Subdivisions

  a
  /          b-a
  | f(x)dx = ----- * ( f(x ) + 4*f(x ) + 2*f(x ) + ...
  /          k*3         0         1         2
  b

             ... + 2*f(x   ) + 4*f(x   ) + f(x ) )
                        k-2         k-1       k
```

1.14 Value-table

```
2.2.4 Value-table


  A value-table of the selected function in a definable intervall and with
a selectable step-width will be generated.
```

1.15 Tangente

```
2.2.5 Tangente


  A new function will be added to the list which represents a tangete on a
definable point x of the selected function.
```

1.16 Taylor

```
2.2.6 Taylor


  A function can be approximated in a range around a selectable point x0
with the Taylor-polynom. The higher you choose the degree of that polynom
the more exact the approximation will be. But beware of choosing "degree"
too high (> 10) because the need of memory and time will increase
dramatically! The calculated polynom will be inserted into your function
list and is marked with a preceding "T" in its name.


Taylor:

   oo      (n)
  ----    f   (x ) *  x
  \             0      0              n
   |    ------------------- * (x - x )
  /                                  0
  ----          n!
  n=0


Graph2D cannot sum up infinite terms and uses the following expression to
generate a Tylor-polynome with a selectable degree n:

   n         n      (k)           l
  ----      ----   f  (x ) * (-x )
  \      k  \          0       0
   |    x    |   --------------------
  /         /
  ----      ----     l! * (k - l)!
   k=0       l=k
```

1.17 2D-graph-window

```
2.3 2D-graph-window


  2D-graph-windows can be opened by the user at any number. With the graphs
you can look at functions (also more than one in a graph) in a coordinate-
system.
```

Which function gets displayed or not in a graph-window can be selected in the functioneditor. The graph-windows and the function-editor run parallel so that you can switch between them and work simultaneously in both of them. Graph-windows are resizeable.
The present intervall and the name of the graph get displayed in the graph-windows title. As long as you press the left mousebutton the coordinates according to the mouseposition get displayed in the text-line above the coordinate-system.

The image-buttons have the following meanings:

```
  Redraw   - The graph gets recalculated showing all changes that have
             been made to the functions since the last time
  Edit     - Open the 2D-graph-editor
  Zoom in  - After marking an area with the mouse a new graph
             opens displaying just the selected area
  Zoom out - A new graph opens containing a four-times larger
             visible intervall
  Print    - Prints the visible graph
```

As long as a 2D-graph-window is active, you can select one of the following menus:

```
Project              - The project-menu


Graph

  Redraw             - like the according image-button
  Edit...            - "
  Zoom in            - "
  Zoom out           - "
  Print              - "
```

1.18 2D-graph-editor

2.3.1 2D-graph-editor

With this editor you are able to adjust the visible intervall, the scaling of the axis and some other attributes of a graph.

You can set the following attributes for each the x- and the y-axis:

from ... to   - The visible intervall for the axis. The value also depends of the choosen unit!

Unit   - The basical unit of the axis. The other values of the axis will be (internally) multiplied with with the unit - so the unit is a kind of a scaling factor. Usually you set the unit to "1". But maybe if you want to generate a coordinate-system especially for trigonometric functions a unit of "3.141" would make sense. If you then enter a visible intervall of e.g. [-2; 4] it would mean, that the axis in fact shows an intervall of [-2*3.141; 4*3.141] = [-6.282; 12.564]. You can chose the unit free as long it stays >0.

Draw axis - Display the axis or not. Even if displayed not, the visible intervall of an axis is of course of importance!

Title - You can give a title to every axis which will be shown in the graph next to the axis. If you maybe chose a unit of e.g. "3.141" then it is important, to make that "public" by e.g. titeling the axis as "x-axis in pi" or something like this.

Marks - An axis can optionally be divided by many small marks. Now here you can define after how many units a mark should be drawn. Furthermore you may place a certain number of (smaller) "Submarks" between every two regular marks. A submark-value of "0" means: no submarks. If you choose too many marks for a graph so that one could not visually tell one from another then Graph2D will not draw the marks.

Text - The marks can optionally be equipped with numbers next to them. You can decide after how many marks a number should be drawn. The number again is dependant of the unit you have chosen. If you draw too many numbers so that they would overleap one another, Graph2D just leaves out some of them.

Now you can decide for both axis wheter there should be a "Raster", wheter the raster should appear as "dots" or as "lines" and how large the distance between the rasterlines in x- and y-direction should be. Again: this is unit-dependant.

"Connect" makes Graph2D to connect the single calculated function- values by lines. So if the graph of a function "jumps" in the visible intervall, maybe because of a low accuracy, the otherwise existing gaps can be avoided. Of the other hand "connect" could draw connections that in reality just do not exist (e.g. sgn(x))!

And last but not least you can choose an "Accuracy", with that a graph should be calculated. At a lower accuracy only fewer function-values will be calculated and the output-speed will increase - but the graph of the function may get a bit unprecise ans show some "steps".

"Font" allows you to choose the font for the whole graph.

You can load and save Graph-settings with any filename with "Load..." and "Save...".

After leaving the graph-editor with "Ok" the active graph gets redrawn to make the probably made changes visible. When leaving the editor via "Cancel" or simply closing the window nothing will be changed of course and the old settings stay valid.

1.19 SIRDS-graph-window

2.4 SIRDS-graph-windows

SIRDS-graph-windows can be opened by the user at any number. With the graphs you can look at functions in a three-dimensional SIRDS coordinate-system.

Which function gets displayed or not in a graph-window can be selected in the functioneditor. The graph-windows and the function-editor run parallel so that you can switch between them and work simultaneously in both of them. Graph-windows are resizeable. The present intervall and the name of the graph get displayed in the graph-windows title.

The image buttons:

```
Redraw - will cause the graph to become actualized and make every
         changes visible that may have been made to the function since the
         last redraw.

Edit   - open the SIRDS-graph-editor

Print  - Prints the visible graph
```

As long as a SIRDS-graph-window is active, you can select one of the following menus:

```
Project              - The project-menu

Graph

  Redraw             - like the according image button
  Edit...            - "
  Print              - "
```

1.20 SIRDS-graph-editor

2.4.1 SIRDS-graph-editor

In this editor the visible intervall, the scaling of the axis and some other attributes of a SIRDS-graph can be set.

For each the x-, y- and z-axis a visible intervall must be choosen x- and y-axis run as known horizontally and vertically on the monitor while the z-axis seems to "touch" the user magically...

With "colors" you can choose how many colors should be used to create the SIRDS-graph. The number of colors has no effect on the output speed.

And last but not least you can choose an "Accuracy", with that a graph should be calculated. At a lower accuracy only fewer function-values will be calculated and the output-speed will increase - but the graph of the function may get a bit unprecise ans show some "steps". But in a SIRDS graph the low resolutions are of a high importance because it sometimes may take really long to create a SIRDS...

You can load and save graph-settings with any filename with "Load..." and "Save...".

After leaving the graph-editor with "Ok" the active graph gets redrawn to make the probably made changes visible. When leaving the editor via "Cancel" or simply closing the window nothing will be changed of course and the old settings stay valid.

1.21 Textviewer

2.5 Textviewer

Textviewers show all kind of texts and lists in Graph2D. With the scroller- bar you can scroll through the text, print it with "Print" and close the window with "Ok" or the close-gadget.

A textviewer does not have to been closed to work on with Graph2D!

1.22 Preset-editor

2.6 Presets-editor

You can do the following presets for Graph2D:

```
Settings directory - this path is used for the file-requesters in the
                     graph editors

2D-graph           - this 2D-settings file is used whenever a new 2D-graph
                     will be opened

SIRDS-graph        - this SIRDS-settings file is used whenever a new
                     SIRDS-graph will be opened

Font               - the font used for math. notation windows.
```

"Use" uses the presets until the program ends, "Save" makes your selections permanent.

1.23 Print

2.7 Print-Requester

With this requester you can print a graphic via the workbench-printer-drivers and the workbench settings. The size of the print can be selected with scale.

1.24 Functions

3.1 Functions

The following chapters are also part of the topic "functions":

A function in Graph2D has to consist of the following parts:

Name

A function needs a name. The name must not contain leading figures or space and mus not exceed an length of 30 chars. You may use (international) chars, figures (not at first position) and the apostroph "’".

Argument list

The name is followed by an argument list in bracktes "(" ")". The list consists of (many) variable names seperated by colons ",". The variable names must follow the same conventions as the function names. A variable must not appear twice or more in an argument list. The list can also be empty - in this case you do not even have to write the brackets. Name and argument-list together are called "functionheader".

"="

Between functionheader and functiontext you must place an "="...

Function text

The definition of the function-text follows the "usual" rules for the notation of mathematical terms. Graph2D is case-sensitive! The functiontext may consist of:

- Operators see also chapter internal~Operators

- Functions see also chapter internal~Functions

  You also can include other user-defined functions out of the function-list into one of your functions.

- Constantes

  A Constante is e real number in the range \ensuremath{\pm}9.22337177E18

- Variables

  Of course you can use only those variables in a function you did define in the argument-list of the function.

Examples for correct functions:

```
  "p(x)=x^2+98",
  "Sum(a,b)=a+b",
  "g=9.81",
  "f(x)=sin(x^2)+1/2*x",
  "function23a(iks,yps,zett)=iks^yps*zett+(-1.23+1.0356E-5)"
```

1.25 Internal functions and operators

3.1.1 Internal functions and operators

The following functions and operators are by default known to Graph2D:

```
  sin(x), arcsin(x), sinh(x),
  cos(x), arccos(x), cosh(x),
  tan(x), arctan(x), tanh(x),
  arctanh(x)    - The well-known trigonometrical functions
  abs(x)        - The absolute value of the argument, also known as |x|
  int(x)        - The integer value of the argument
  ln(x)         - natural logarithm (to the base 2)
  sqrt(x)       - The square root x^(1/2)
  sgn(x)        - returns 1, 0, -1 in dependance of x>0, x=0, x<0
  fak(x)        - known as x!. e.g. fak(4)=4*3*2*1=24
                  (only defined for positive integer values)
  if(a, b)      - returns b if a is TRUE (=1) otherwise 0
  binom(n, k)   - binomialcoefficient
  exp(x)        - exponential-function e^x
  e             - 2.71...
  pi            - 3.1415...

  + , -
  * , /
  ^             - I think you know that, don’t you? Note: a^b^c = (a^b)^c
  = , > , <,
  >=, <=, <>    - Compare the two operands
                  e.g. a > b results 1 if a > b otherwise a > b results 0
  AND, OR, NOT - Boolean operators. Only use them for values like
                  1 (TRUE) and 0 (FALSE). Do not expect a bit-wise
                  work of those operators!
```

1.26 Include functions

3.1.2 Include user functions

User defined functions (functions defined by you) can include other user-defined functions into their own definition. If you do that, you should take care of the following things:

Changes to the included function cause no changes to the including function! If you want Graph2D to accept the changes you have to reinter-

```
pret the including function.

  You absolutely have to avoid (even indirect) circle-definitions! Never do
e.g. this: a(x)=c(x); b(x)=a(x); c(x)=b(x)

Examples:

  "Sum(a,b)=a+b"
  "f(x)=2^Sum(x,5)+x^2"                correct

  "k=5.6"
  "foobar(x,y)=k*x+y+1"                correct

  "f(x)=g(x)/6"
  "g(x)=2^f(x)"                        error! (circle-definition!)

  "u(x)=u(1/x)+3.14"                   error! (circle-definition!)
```

1.27 EBNF

```
3.1.3 EBNF-Syntax


Spaces are allowed (except inside of qualifiers/names) in the whole
functiontext and may be used to structure the term.

  Function   = Qualifier [ "(" [ Qualifier { "," Qualifier } ] ")" ]
               "=" Expression .

  Expression = Prio1 [ ( "=" | ">" | "<" | "<=" | ">=" | "<>" )
               Expression ] .

  Prio1      = [ "+" | "-" ] Prio2 [ ( "+" | "-" | "OR" ) Prio1 ] .

  Prio2      = Prio3 [ ( "/" | "*" | "AND" ) Prio2 ] .

  Prio3      = [ "NOT" ] Prio4 [ "^" Prio3 ] .

  Prio4      = Number | "(" Expression ")" | Qualifier | Call .

  Call       = Qualifier [ "(" Expression { "," Expression } ")" ] .


  Number     = Figure { Figure } [ "." Figure { Figure } ]
               [ "E" ["-"] Figure { Figure } ] .

  Qualifier  = Letter { Letter | Figure } .

  Figure     = "0" | .. | "9" .

  Letter     = "A" | .. | "Z" | "a" | .. | "z" | "'".
```

1.28 History

```
3.2 History


V2.10 (03.03.95):

  - printing works now with full printer-resolution and selectable size
  - graphs and functions can be saved as IFF-pictures
  - you can choose between image-buttons, text-buttons or no buttons
  - internal constantes "e" and "pi"
  - numbers are displayed with higher precision
  - 2D-graph-editor now split into 2 windows
  - display "-x" instead of "0-x" now
  - graph-windows now remember their old positions
  - some GUI-layout changes
  - SIRDS-calculation can be cancelled with ESC
  - different handling of including functions


V2.00 (04.02.95):

  - improved discussion in speed and success, new: discussion-requester
  - new GUI-layout, images and imagebuttons
  - taylor approximation
  - fixed some terrible bugs in the mathematical routines
  - better memory-handling and speed-up in all calculations
  - extended and restructured online-help
  - new internal functions "binom(n, k)" and "exp(x)"
  - fixed a bug in math. notation
  - fixed another bug after loading functions...


V1.60 (08.01.95):

  - preset editor allows to configure a font for the notation window,
    a path for graphsettings and preset-settings for both graph-types
  - fixed bug after loading functions
  - 2D-graph: selectable font, new raster-type, smarter draw-algorithm
    for better-looking coordinate-systems
  - math. notation now uses "·" for multiplication and "!" for "fak",
    added menus and printing-option to the notation window
  - printing graphs now without window-border
  - all windows fit on a NTSC 640*200 / Topaz 8 screen
  - changes in the archive content


V1.50 (10.12.94):

  - The first localized english version of Graph2D
    if you are interested in the former version please have a look into
    the german documentation :-)
```

1.29 Mui

3.3 MUI

MUI-Copyright:

```
                          This application uses

                        MUI - MagicUserInterface

                (c) Copyright 1993/94 by Stefan Stuntz
```

MUI is a system to generate and maintain graphical user interfaces. With the aid of a preferences program, the user of an application has the ability to customize the outfit according to his personal taste.

MUI is distributed as shareware. To obtain a complete package containing lots of examples and more information about registration please look for a file called "muiXXusr.lha" (XX means the latest version number) on your local bulletin boards or on public domain disks.

```
          If you want to register directly, feel free to send


                         DM 30.-  or  US$ 20.-

                                  to

                             Stefan Stuntz
                        Eduard-Spranger-Straße 7
                             80935 München
                                GERMANY
```

Comments on MUI concerning Graph2D:

Graph2D needs MUI 2.1 or higher to be installed. You are allowed to use MUI without registering for it - but when you register you can take advantage of some extended functions in the MUI preferences. It is very recommended to read the MUI documentation carefully - especially of the MUI preferences. Despite that i would like to show you in the following list some advantages of MUI-programs that could be useful using Graph2D:

If you want Graph2D to work on an own screen and not on the Workbench then simply configure Graph2D with the MUI preferences to use any screen you like.

Windows of MUI applications are resizeable and completely fontsensitive what means, that they look fine with every font.

MUI applications may be iconified at every time with an extra-gadget in the windows title bar.

MUI applications can optionally completely be handled with the keyboard. Via Tab-cycling and shortcuts every gadget may be (de-)activated without having to use the mouse. The gadget actually receiving keyboard input is always marked with a border or something like this. Windows can be closed normally by pressing ESC.

MUI applications are known to the system as commodities and can so be handled with the commodity-exchange program.

1.30 SIRDS

3.4 SIRDS

Graph2D uses for the three-dimensional presentation of function a technique known as SIRDS (= Single Image Random Dot Stereogramm). You do not need special glasses or something like this, you may even print out the pictures without loosing the effect.

The only thing you have to do to achieve the 3D-effect ist to look at the pictures in a special way. Instead of looking at the picture itself you have to look behind the picture. Of course you will not see the random dots sharp this way - but this is nescessary for the three-dimensionality.
It may take sometimes until the effect is going to come, and for strange reasons some people never get the "kick". For further instructions how to achieve the effect please refer to the numerous publications concerning SIRDS pictures.

I want to thank Kilian Singer for his small but effective piece of code that makes those pictures possible in Graph2D!

1.31 Index

3.5 Index