

**Pick Systems
GraPICK**

**Installation Manual and
User's Guide**

PickTM
S Y S T E M S

**Pick Systems
GraPICK**

**Installation Manual and
User's Guide**

Pick Systems GraPICK v1.2.2 (Mar. 01, 1995)

© Copyright 1995, Pick Systems. All rights reserved.
Printed in the United States of America.

PROPRIETARY INFORMATION

This document contains information that is proprietary and considered a trade secret of Pick Systems. It is expressly agreed that it shall not be reproduced in whole or part, disclosed, divulged, or made available to any third party either directly or indirectly. Reproduction of this document for any purpose is prohibited without the prior express written authorization of Pick Systems.

While every precaution has been taken in the preparation of this publication, the information in this document is subject to change without notice and does not represent a commitment on the part of Pick Systems. Pick Systems assumes no liability for any errors or omissions in this manual.

TRADEMARK CREDITS

The following are registered trademarks of Pick Systems, Irvine, California, U.S.A.

Pick ®
Pick Systems ®
Pick Systems of America ®
Pick Operating System ®
Pick Operating Software ®
Pick Open Architecture ®
Pick Open Database ®
Pick PC/XT System ®
Advanced Pick ®
PickWare ®
Pick News ®
PickFair ®
Pick Partners ®
Pick Pavilion ®
PickNet ®
PickWorld ®

Access is a trademark of Pick Systems.
Advanced Pick Reference Guide is a trademark of Pick Systems.
EPick is a trademark of Pick Systems.
FlashBASIC is a trademark of Pick Systems.
GraPICK is a trademark of Pick Systems.
Output processor is a trademark of Pick Systems.
Pick/BASIC is a trademark of Pick Systems.
UPdate processor is a trademark of Pick Systems.

rademark of Apple Computer Corp.

lemark of Data General Corp.

rademark of Data General Corp.

RISC System/6000 and RS/6000 are trademarks of International Business Machines Corp.

is a trademark of Apple Computer Corp.

Windows and MS-Windows are trademarks of Microsoft Corp.

gistered trademark of The Santa Cruz Operation Inc.

gistered trademark of Unix System Laboratories Inc.

PREFACE

Document Conventions

This manual uses different typefaces to indicate different kinds of information. Please see the appendix to determine the different typefaces and type styles we use and what they mean.

Keyboard Conventions

- <Ctrl>** Words in darts (<>) are keys found on the keyboard. For example, **hello<Enter>** means to type the letters **hello**, followed by pressing the “Enter” key.
- Any Key** Any key on the keyboard, except the **<Caps Lock>**, **<Shift>**, or **<Alt>**. We recommend using the **<space bar>**.
- Enter:** Input the specified commands or text as shown in the instruction, and then press the carriage return key, a key usually labeled as **<Enter>**, **<Return>**, or **<Line>** on your keyboard. We usually use the term “enter” when we are referring to a whole line of information to be typed in to the computer.
- Type:** Input the specified commands or text as shown in the instruction. Do not press the carriage return key unless instructed. We usually use the term “type” when we are referring to a word or two of information that is to be typed into the computer.
- Press:** Press the single specified character (or combination of characters) as shown. We usually use the term “press” when we are referring to a single key. For example: Press the **<F1>** key.

style

In this manual, we will be using this typeface for the body of our text.
Referring to a dialog between the user and the computer, we will use this typeface.

When describing a dialog or interaction between the user and the machine, we will use the type styles to denote different actions:

Normal Words in a normal style indicate typical prompt or output by the machine.

Bold Words or characters in **boldface** indicate data entry by the user.
Words in **bold upright** are to be entered (typed) as shown.
These boldface words are commands, filenames, options and other keywords recognized by the system.
For example, **logto dm** would mean that you type it in exactly as it is shown.

Italics Words in *italics* are parameters to be replaced by the actual term. It might be an actual name, word, or number.
For example, *baud.rate* might be replaced by **9600**.

Information or commands displayed within braces `< { } >` is optional. The braces themselves are not entered.

When the letters *n* or *m* are shown, they represent numbers.
Numbers are usually input as a series of digits without commas.
Numbers may also be used to define a range, as in this designation. The starting and ending numbers are typed with a hyphen in between.
For example, *n*{ *-m*} might be replaced by **10** or it might be replaced by **1-31**.

The vertical bar or pipe sign `< | >` separates available choices. The vertical bar itself is not entered.
For example, **yes | no** would be replaced with either **yes** or **no** (but not both).

[?] A required entry where the possible choices are displayed in brackets `< [] >`. Only one of the list of available entries is to be used.
For example, **[300 | 600 | 1200 | 2400 | 4800 | 9600]** would be replaced with only one of those values.

PREFACE	
FORMAT CONVENTIONS	
GRAPICK INSTALLATION GUIDE	
PICK DISTRIBUTION DISKETTES	
MINIMUM EQUIPMENT REQUIRED.....	
LIMITATIONS.....	
INSTALLATION PROCEDURE.....	
<i>Installing the GraPICK Client</i>	
<i>Installing the GraPICK Server</i>	
<i>Enabling GraPICK for the User</i>	
GRAPICK INSTALLATION TROUBLESHOOTING.....	
GRAPICK USER'S GUIDE	
INTRODUCTION	
STARTING GRAPICK ON THE CLIENT	
STARTING GRAPICK ON THE SERVER	
THE GRAPICK CLIENT ENVIRONMENT - A USER'S POINT OF VIEW	
<i>Executing TCL Commands</i>	
<i>Entering or Editing Data</i>	
<i>Windows Features</i>	
Windows Data Box	
Text Editing Using the Mouse	
Toolbar Buttons.....	
<i>Windows-style Text Editing</i>	
<i>GraPICK Features</i>	
The UPDATE Processor	
Highlighted Attributes.....	
Cruising.....	
Zooming	
Closing a Zoomed-to Window	
GraPICK Toolbar Buttons.....	
THE GRAPICK ENVIRONMENT - A PROGRAMMER'S POINT OF VIEW	
<i>GraPICK Filesystem Changes</i>	
Form Item.....	
w{n} Processing Code	
Opf Item	
<i>GraPICK UPDATE Processor Changes</i>	
UP (w) Option.....	
UP Form Connective.....	
<i>GraPICK Pick/BASIC Changes</i>	
Data Statement Extension	

ing and Exporting Data.....	38
ick/BASIC 'gui.lib' Subroutine Library.....	39
ors.....	39
roperties.....	40
vate.menu().....	42
menu.item().....	42
lge.object().....	42
lge.wdw.title().....	42
sk.menu.item().....	42
r.data.obj().....	43
r.tb.....	43
r.wdw.....	43
te.menu().....	43
te.object().....	43
te.submenu().....	43
te.value().....	44
menu().....	44
menu.item().....	44
value().....	44
wdw().....	44
n.fn().....	45
n.vseq().....	45
color().....	45
/..menu.item().....	45
OFF.....	45
ON.....	46
hlight.menu.item().....	46
menu.item().....	46
ress.menu.item().....	46
range.menu.....	46
range.objects().....	47
..menu.item().....	47
B().....	47
wdw.prg().....	47
focus().....	47
ine().....	48
lg.box().....	49
lsg.box().....	50
MING USING PICK/BASIC.....	51
g a Form Item.....	52
g the Form Item.....	53
3 FORM ITEMS.....	55
dit Mode.....	55
mode.....	55
Input Information From an Object.....	56
WINDOWS TERM DEFINITIONS.....	57
.....	57

<i>Bitmap Graphics</i>	
<i>DDE</i>	
<i>Data Boxes</i>	
<i>Events</i>	
<i>Focus</i>	
<i>Forms</i>	
<i>Form Title</i>	
<i>FTP</i>	
<i>GUI</i>	
<i>Menu Items</i>	
<i>ODBC</i>	
<i>Objects</i>	
<i>Object Properties</i>	
<i>OLE</i>	
<i>Properties</i>	
<i>Tag Fields</i>	
<i>3-D Presentation</i>	
<i>Window</i>	
EXAMPLES	
<i>demo.1</i>	
<i>main</i>	
<i>u entity (w)</i>	
GRAPHICK TROUBLESHOOTING	
INDEX	
READER'S COMMENTS	

GraPICK Installation Guide

Distribution Diskettes

consists of two packages:

st package is the Advanced Pick (AP) extension part (the "server").
istributed on one (or more) 3-1/2" high-density diskette(s).

cond package is the MS-DOS Windows terminal part (the "client").
istributed on one 3-1/2" high-density diskette.

Server diskettes are specific to the platform of Advanced Pick you are running. Make sure
ve the proper diskettes for your platform. The available GraPICK platforms are:

ICK Server for Advanced Pick DOS
ICK Server for Advanced Pick Native
ICK Server for Advanced Pick 386 (SCO, Protected, AT&T, etc.)
ICK Server for Advanced Pick AIX
ICK Server for Advanced Pick DG AViiON
ICK Server for Advanced Pick Motorola

Minimum Equipment Required

usually requires the use of (at least) two computer systems:

system (the host, or server). GraPICK requires Advanced Pick 5.2.5 or greater, or any
ation program that has been made "GraPICK-aware."

ICK Server requires up to 7.5 Mb of disk space to install.

-DOS system (the terminal, or client) that is set up as a terminal emulator and is connected
AP system. GraPICK requires MS-DOS 3.3 or greater, running Microsoft Windows 3.1 or

ICK Client will require up to 1 Mb of disk space to install.

installing GraPICK into AP/DOS, then the server and the client may be one and the same

P/DOS system will be the server.

ICK Server requires about 7.5 Mb of disk space to install.

me machine running Windows will be the client.

ICK Client will require about 1 Mb of disk space to install.

Limitations

This release of GraPICK is a preliminary release and is known to have the following limitations:

- Support of several different types of Windows objects have yet to be implemented.
- The GraPICK client currently only supports RS232, direct-connect to AP/DOS, and communications protocols.
- Windows-style edit mode does not fully support the function keys.
- “AP Edit” mode (from the File menu's Windows Properties feature) is not fully implemented.

Installation Procedure

There are two parts to the installation. The first installs GraPICK into the Windows environment (the client), and the second part of the install allows the AP environment (the server) to be made GraPICK-compatible.

Client Registration

At the start, please take a few moments to fill out and send in the registration card, found with the software. Registration allows you access to Pick Systems Customer Service and allows us to keep track of product updates and enhancements.

Installing the GraPICK Client into Windows

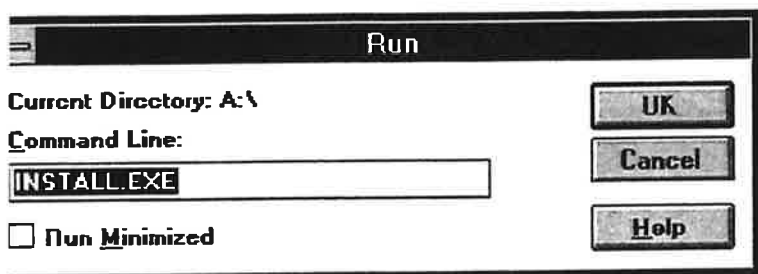
On your machine running Windows, insert the disk labeled: GraPICK Client for Windows Volume #1/1 into the floppy drive.

Start the installation routine by doing the following:
1. Go to the Windows **File Manager**,

2. Click on the appropriate diskette drive button to view the contents of the GraPICK Client diskette,

3. Highlight **install.exe**,

4. Pull down the File Manager's **File** menu and click on **Run**.



5. Click on the **OK** button to continue, or click on the **Help** button for more Windows information.



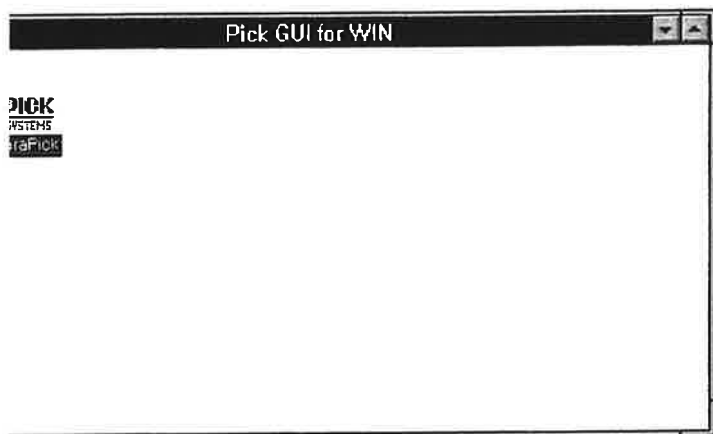
- Enter the subdirectory name where you want GraPICK to install and click on the **OK** button when you are ready.
- GraPICK will install itself, and when it is done, it will display:



Click on the **OK** button when you are ready.

point, the GraPICK client is installed on your system.

Machine will show a group window with the GraPICK icon in it. Double-clicking on this icon starts the GraPICK client.



If you are installing GraPICK on AP/DOS, then there are a couple of additional steps necessary.

It may be necessary to reinstall AP/DOS. (The installation process will only tell you it is necessary to reinstall if it encounters a problem, so it would be wise to back your files up first.)

If you do become necessary to reinstall AP/DOS, then the following step *must* be performed immediately before loading the abs diskettes. Otherwise, the following step may be performed at any time.

Put the "GraPICK Client for Windows" diskette in the drive, and copy the programs "pickmon.exe", "pick.pif", and "pickmon.pif" from the "\PICK" subdirectory on the diskette to your "K:" subdirectory (or whatever you have named it) on your disk drive:

```
a:\pick\*. * c:\pick /y
3 file(s) copied
```

Run your AUTOEXEC.BAT file to execute this program.

Your AUTOEXEC.BAT file would look like this:

```
echo off
prompt $p$g
path c:\dos;c:\windows;c:\msoffice
set temp=c:\dos
```

```
C:\dos\smartdrv.exe /x
lh /L:1,6400 c:\dos\doskey

chkdsk/f < c:\dos\xdos\yes.dat > nul:
if exist file*.chk erase file*.chk

c:\pick\ttsr.exe
win
```

(Note: You may not use the DOS “loadhigh” command to load the “ttsr.exe” r

When the program is invoked, it should report:
hooking interrupt 0060

You may, if you wish, use the command:
c:\pick\ttsr.exe > nul:
to avoid viewing this output.

Configuring the GraPICK Server to make Advanced Pick become K-aware

On the system already running Advanced Pick, and place the disk labeled GraPICK Server for Advanced Pick *platform* Volume #1/n in the floppy drive. GraPICK floppies *must* match the target platform.

Unless you are installing GraPICK for AP/DOS, this will *not* be the same physical machine as the client machine running Windows, above.

Use the "dm" account.

```
logto dm
```

Use the appropriate tape device.

```
set-floppy (ah) (if your floppy device is drive
```

ah) Tape device is assigned to 3 1/2" high density (1.44M) floppy using drive

```
t-rew
```

```
size: 500
```

```
not
```

```
unt-restore gui.demo
```

```
File name on tape: gui.demo
```

```
gui.demo > filename
```

The system will ask for the other diskettes, as necessary)

```
Restore from incremental save tape (y/n)? n
```

```
Restore from transaction log (y/n)? n
```

(Restoring indexes)

```
Restore completed.
```

```
logto gui.demo
```


6a. If you are installing this on an AP/Pro platform, then perform this step, otherwise go to

- Enter: `up dict nsm.gui 'mode'`

```
01 6386
02 virtual
03 translation
04 sensitive
05 1
06 5.2.6.A3
```

(the contents may vary somewhat between)
- Go to line 01 and change 6386 to `pro`
Type: `pro<Ctrl>e`
- Press: `<Ctrl>xf` to file the change.

7. Enter: `grapick.install`

It will display various new modes as it generates a new gui abs. It will take a few minutes.
Creating abs.gui.temp ... Please Wait ...

```
:
Finally, it will say:
:
--- Completed ---
--- Execute the 'setup.grapick' verb to install GraPICK to another account ---
```

8. Enter: `catalog dm,bp, term`
[244] 'term' cataloged

9. Enter: `sgs`
`day mon year platform`
--- GraPICK is set up ---

At this point, GraPICK is installed on your server system.

To use GraPICK, just:

```
logto gui.demo
sgs
```

and run your Pick/BASIC programs, or use the new features of the Update processor.

When you type “**sgs**” from a non-gui terminal, the cursor may “disappear.” This is normal. If the terminal type back to “mm-mon” will make the cursor reappear.

Due to a timing diagnostic in AP/DOS 5.2.7.14, you may (or may not) see the following message when you boot AP/DOS on faster 486 and pentium machines:

```
: # 30016 _
```

Press <Enter> to continue, and the machine will continue to boot up to the logon prompt without trouble.

This message occurs due to an instruction diagnostic command (in AP/DOS) which sometimes returns an error code to be erroneously reported back to Pick. It is harmless, and the problem will be fixed in a future release.

Enabling GraPICK for the User

To set up GraPICK in other user accounts, do the following:

1. Enter: `logto gui.demo`
2. Enter: `setup.grapick`
target account: `account.name`
----- Installation completed -----

Repeat the above process for each additional account you wish to use with GraPICK..

GRA PCK Installation Troubleshooting

If you have any problems, try re-installing the product first.
The problems you may encounter are:

```
gui.demo' exists on file
You are trying to install GraPICK over an existing version.
Delete the old one by doing:
delete-account gui.demo
ls,, 'gui.demo' size = 44
pe d
se/dict 120396
dulo 37

spriv sys2
stification L
dth 12
allocation
DISPLAY FILES BEFORE DELETING (<Y>,N)? n
DO YOU STILL WANT TO DELETE THE ACCOUNT (<N>,Y)? y
ls,, 'gui.demo' size 44 deleted.
/ installing it again.

missing!
allocation stopped ---
You are trying to invoke GraPICK, but you haven't run the "setup.grapick" routine on your
system.

>gto gui.demo
>setup.grapick
Go back to your account, and try again.
```

If you have any other problems installing GraPICK, please contact:

Pick Systems
Customer Service
1691 Browning Ave.
Irvine, CA 92714
United States

Phone: [1] 714/261-1875
Fax:[1] 714/261-5308

Pick Systems Support - Europe
The Manor
Haseley Business Centre
Warwick CV CV35 7LS
United Kingdom

Phone: [44] 1203-537027
Fax:[44] 1203-537045

Pick Systems Africa (Pty) Ltd.
P.O. Box 15277
Vlaeberg 8018
South Africa

Phone: [27] 21-240656
Fax:[27] 21-247761

Pick Systems Ltd.
12th Floor, Profsojuznaja Str. 84/32
M. V. Kaldish Applied Mathematics Institute
Russian Academy of Sciences
Moscow, GSP-7 Russia
Russian Federation

Phone: [7] 095-33-36456
Fax:[7] 095-33-38000

GraPICK User's Guide

Introduction

This document offers GraPICK as an add-on to existing Pick software, which enables Advanced Pick users to use the Windows environment.

As defined, GraPICK is a module within Advanced Pick that can be attached to (or driven by) programs, processes, or end-user applications to tap the full graphical, messaging, database and transaction capabilities of the local client environment. In this version, the client environment is Windows. Future GraPICK implementations may support other client environments such as OS/2, Apple Macintosh, Motif, etc.

A more limited definition of GraPICK would be a graphical user interface (GUI) program running in a Windows environment that acts primarily as an enhanced terminal emulator. It allows a Pick user to create and manage Windows objects and facilitates interfacing with other Windows programs.

GraPICK is thus made up of two parts - an extension to Advanced Pick that makes the system more Windows aware, and a Windows connection package that permits a PC running Windows to interface more effectively with Pick.

You can use GraPICK in either of two ways:

Changes to the UPDATE processor, Attribute correlatives and the new "form" connective lets you use the Windows environment without having to write a single line of Pick/BASIC code.

Changes to the Pick/BASIC subroutine library allows application programs to be customized to run in the Windows environment.

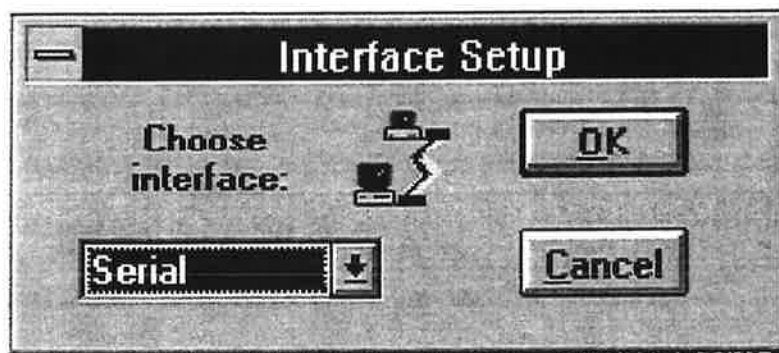
Windows does not come without a price. You, as an applications developer, must learn how to manage Windows "forms" and "objects." One goal of GraPICK is to reduce the management of the forms and their objects, down to simple database transactions, where your desktop objects exist simply as a file in the Pick database, accessible and maintainable like any other Pick object.

We assume that you, as a user or a developer, are familiar with Microsoft Windows, and that you understand the concept of using a mouse, clicking, double-clicking, dragging, highlighting, and so on. We also assume that you, as a programmer, are familiar with Microsoft Windows terms such as windows, menus, buttons and so on. If this is not the case, points in this guide may seem confusing. We assume that you spend some time learning about the Windows environment before trying to make use of GraPICK.

Starting GraPICK on the Client

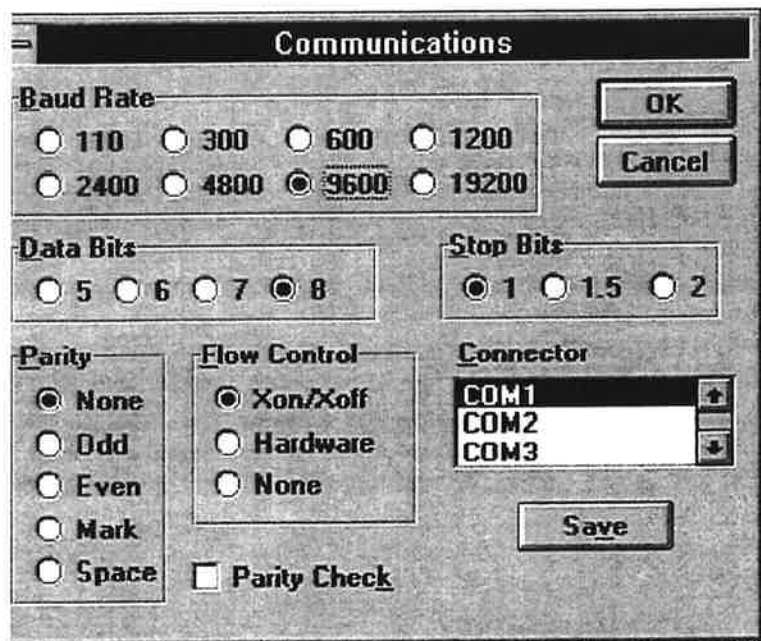
To start running GraPICK, open the PICK GUI for WIN group and then double-click the **GraPICK** icon.

- The first time you invoke GraPICK, you must define the type of connection to Advance you will be using.
- Indicate which interface you wish to use.
Pull down the **GraPICK Settings** menu, and choose **Interface**.



- Highlight the appropriate interface:
 - Use a standard RS232 serial connection to the host (server) machine
-or-
 - Use the FTP (File Transfer Protocol) TCP/IP interface to the host (server) machine
Note: If FTP has not been installed on your Windows workstation, then it will not be an option.
-or-
 - Use the AP/DOS interface to connect this machine (as a client) to this machine (server).
- Click on the **OK** button when you are ready.

For a serial connection, you will need to verify the serial port setup characteristics:
 Select the **Settings** menu, and click on **Communications**.



The defaults are 9600 baud, 8 data bits, 1 stop bits, no parity, Xon/Xoff enabled and COM1.

To modify the settings appropriate for your system, click on the **Save** button to save the new settings and then click on the **OK** button.

1. Make the connection to your host Pick machine and log on.
2. Activate the host's GraPICK abs and then inform your Pick process that you are on GraPICK Windows terminal by specifying its term type as "gui":

- Enter: **sgs**

Note: the macro `sgs` contains the the following commands:

```
exec  gui.demo,abs.gui.temp
term  gui
```

You may wish to use them instead of the macro, but if you do, these two lines together.

Note 1: You may wish to add the above command(s) to your logon macro in your user "dm,users," file).

Note 2: Many user logon macros (in the "dm,users," file) already have an "exec" command, which may interfere with the "exec gui.demo,abs.gui.temp," command. This will prevent GraPICK from functioning properly for the current process. Be sure the "exec" command that gets executed is the "exec gui.demo,abs.gui" command.

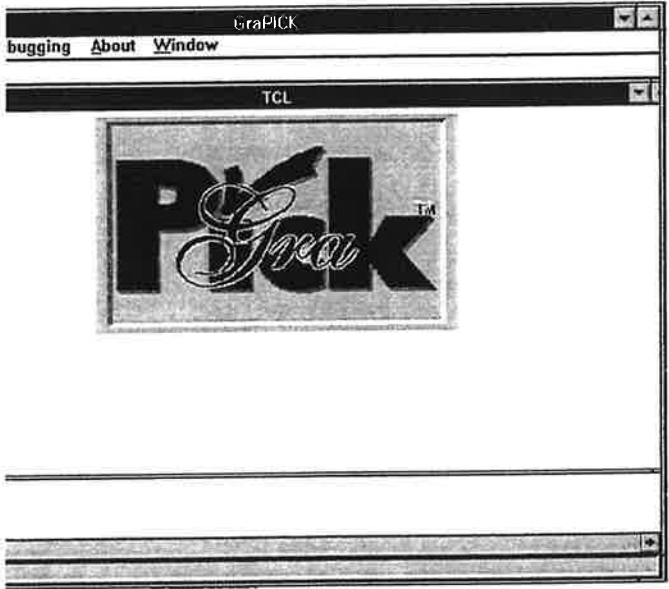
Starting GraPICK on the Server

Once you have installed GraPICK on your server, there is nothing more for you to do. The capability is there waiting for the user to take advantage of it.

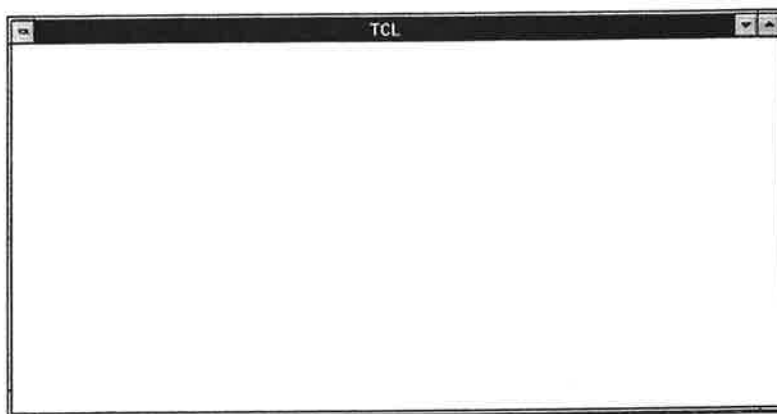
Note: if you create additional accounts later, you will have to run the "setup.grapick" to allow it to use GraPICK.

GraPICK Client Environment - A User's Point of View

The GraPICK client terminal (or GUI terminal) lets you access an Advanced Pick system from a Windows environment, along with all the options and features available to a Windows user.



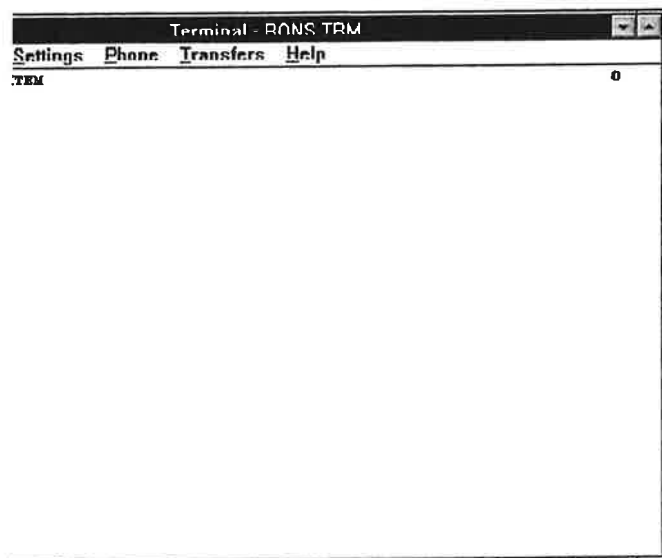
Executing TCL Commands



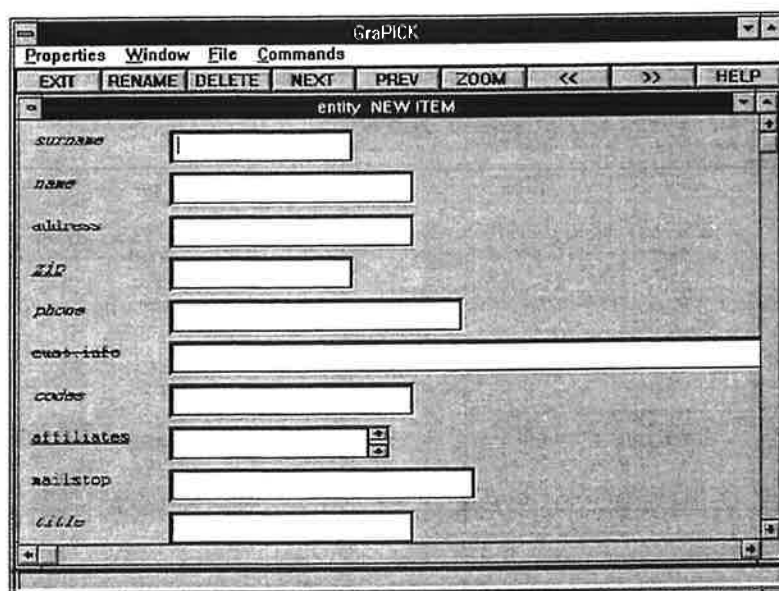
GraPICK offers a windowed terminal screen for communicating with the Pick host (serv TCL commands as you usually do. Pull down the **Settings** menu and choose **Font** to display font.

ig or Editing Data

y Update processor entry screen through the Microsoft Windows Terminal emulator might is:



In GraPICK, using the (w) option, the screen would look like this:



The screenshot shows a window titled "GraPICK" with a menu bar containing "Properties", "Window", "File", and "Commands". Below the menu bar is a toolbar with buttons for "EXIT", "RENAME", "DELETE", "NEXT", "PREV", "ZOOM", "<<", ">>", and "HELP". The main area of the window is titled "entity NEW ITEM" and contains a list of attributes with corresponding input fields:

Attribute	Input Field
<i>surname</i>	Text input field
<i>name</i>	Text input field
<i>address</i>	Text input field
<i>zip</i>	Text input field
<i>phone</i>	Text input field
<i>extra-info</i>	Text input field
<i>course</i>	Text input field
<i>affiliates</i>	Text input field with a dropdown arrow
<i>mailstop</i>	Text input field
<i>title</i>	Text input field

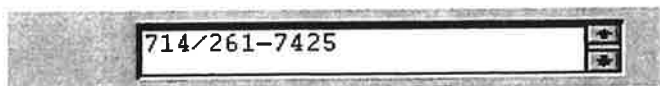
Note that each line of data entry is clearly defined and type styles of the attribute and different features available for that attribute.

Windows Features

Standard Windows Data Box

Uses a standard Windows data box to display all the attributes to be edited in an item. If all attributes will not fit within a single screen, then use the scroll buttons on the side or bottom of the

Attribute name is shown with a "slot" (or data box) alongside it for the data. The width of the data box is determined by the attribute-defining item. The height of the data box is always 1, unless the attribute-defining item has a "wr" as an input-processing code, or unless the form was previously set to wrap (see Local Edit Mode and AP Edit Mode).



If the value exceeds the data box's width, or if there are multiple values, then it will stack the data on multiple lines (as if it had a justification type of "tx"). and automatically display a scroll bar on the right side of the data box. Click on the up/down button to scroll the data in the slot.

Editing Using the Mouse

Use the standard Windows text-editing features while entering or editing information. Use the mouse or the arrow keys to position the cursor, highlight letters or words for deletion or insertion. Use <Ctrl><Delete> to delete words, etc.

Refer to your Windows documentation for a more complete explanation.

Toolbar Buttons

Toolbar buttons let you perform predefined commands, like a macro. Mouse-click on the button performs the operation. See below for the predefined GraPICK Toolbar buttons.



EXIT	Exits the UPdate processor (<Ctrl>xe).
RENAME	Renames the item-id. It will prompt you for the new name.
DELETE	Deletes the item (<Ctrl>xo).
NEXT	Goes to the next item in an active list (if any). Otherwise, it exits the UPdate processor.
PREV	Goes to the previous item in an active list (if any). Otherwise, it has no function.
ZOOM	Zooms to this item in a remote file (<Ctrl>g<Ctrl>g).
<<	Cruises to the previous key (<Ctrl>y).
>>	Cruises to the next key (<Ctrl>u).
HELP	Displays the Help information on this attribute, if any ("?").

Windows-style Text Editing

All the typical Advanced Pick data entry/editing features are useable in the GraPICK environment. In addition, the Windows editing features using the mouse and arrow-keys, and Windows-style editing features are also available. You can also "cut and paste" using Windows' clipboard using the <Alt><PrintScreen> within data entry forms.

Note: Only partial function key support is included in this release, with full function key support to be included in a later release. Future versions will give you even more convenient editing keys.

2K Features

Update Processor (w) Option

The processor now works in both character-mode and windows-mode. Windows-mode may be enabled by merely including the “(w)” option to the TCL-command line (or the input-macro in the defining item) for the sentence, e.g.,

```
entity name phone zip comments (w)
```

If windows-mode is designated, it will propagate to all subsequent zooming and cruising windows, and the process exits the UPdate processor and returns to TCL.

Highlighted Attributes

Attributes allow you to “cruise” through the current file using existing indexes, or “cruise” backwards on an index of a remote file and/or “zoom” into this same remote item. These attributes now have their attribute names highlighted to inform you of the added functionality of this attribute. Pick uses the following highlighting conventions:

Attribute characteristic	meaning
lined	“local” cruising and zooming enabled
<u>lined</u>	“remote” cruising and zooming enabled
<u>lined italics</u>	“local” and “remote” cruising and zooming enabled
⌘t	attribute is display-only

When you see an attribute name highlighted in such a way, it will be apparent that this attribute has the capability to “cruise” or “zoom” or both.

Using

To cruise on an item by selecting that attribute, and:

1. Use the typical UPdate processor <Ctrl>u | <Ctrl>y commands, or

2. Left-click on the left mouse button, or

3. Click on the “>>” button.

The available indexed attribute value will display in the data field.

See the *Advanced Pick Reference Guide* (formerly *Epick*) for more information on UPdate processor

options.

Zooming

You can zoom to an item by first selecting that attribute, and then:

- use the typical UPdate processor <Ctrl>g<Ctrl>g command, or
- double-click on the right mouse button, or
- click on the **ZOOM** button.

When you zoom to a remote item, a new window will open for that zoomed item. Edit (item as usual.

Closing a Zoomed-to Window

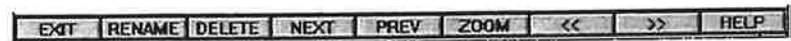
You can exit the zoomed-to item by:

- using the typical UPdate processor <Ctrl>xf | <Ctrl>xe | <Ctrl>xo | etc. cor
- by clicking on the **exit** button, or
- use the pull-down File menu and exit, or
- double-clicking on the top-left box in the window frame.

(See *The Advanced Pick Reference Guide* (formerly *Epick*) for more information on UPd: commands.)

GraPICK Toolbar Buttons

While in the UPdate processor, GraPICK also has the following Toolbar buttons availa use:



EXIT	Exits the UPdate processor (<Ctrl>xe).
RENAME	Renames the item-id. It will prompt you for the new name.
DELETE	Deletes the item (<Ctrl>xo).
NEXT	Goes to the next item in an active list (if any). Otherwise, it exits the UPdate
PREV	Goes to the previous item in an active list (if any). Otherwise, it has no funct
ZOOM	Zooms to this item in a remote file (<Ctrl>g<Ctrl>g).
<<	Cruises to the previous key (<Ctrl>y).
>>	Cruises to the next key (<Ctrl>u).
HELP	Displays the Help information on this attribute, if any (“?”).

GraPICK Client Environment - A Programmer's Point of

Improvements have been made to the Advanced Pick Operating System to make it easier to use GraPICK applications.

GraPICK Filesystem Changes

Form Item

You need to modify your existing dictionaries to use GraPICK screens (forms). It will generate windows form for you automatically.

A form in GraPICK is the existing screen layout generated directly by an application, where it makes various assumptions. GraPICK builds a form from explicit object definitions sent as references to it from the application (whether the UPDATE processor, Pick/BASIC, etc.).

A form can specify a form name for reference in other routines.

To save a form on either the DOS (client) or Pick (server) side, go to the GraPICK Main Menu, pull down the Properties menu, and choose **Save Properties**. The UPDATE processor will generate a unique name for this form based on the list of attribute names and save it. If the UPDATE processor is invoked with the same attribute list, it will recall the same form automatically.

If you wish, modify the form and save it again, or save it as a custom form. If you wish to name a form item a name yourself, then use the `form` connective in the UP command line. (See New Form connective.)

"w{n}" Processing Code

The "w{n}" processing code has been added to allow you to specify the size of an attribute's data box on the screen (or "form"). Normally, the data box size has a default value of one. But to have a data box display three values for, say, the "phone" attribute, add a "w3" to the Input-conversion of the re-defining item "phone" in the dictionary of the file. You can also override this using the "w" command in GraPICK.

New opf Item

.Opf files contain compressed definitions of all objects in the form starting from the Win Forms are placed on the dictionary of corresponding Pick file on the server. If the form is Pick/BASIC application, it is saved in the dictionary of the "pd.forms" file. The structures on DOS and .opf items on Pick are quite similar.

Forms saved on Pick have the following structure:

Attribute #	Term	Description
1	f	specifies a form-defining item.
2	<i>date/time</i>	time the form was saved.
3	<i>dosfile</i>	the name of corresponding DOS .opf file.
4	.opf	.opf file imported from DOS.

Pick Update Processor Changes

P (w) Option

The processor now works in both character-mode and windows-mode. Windows-mode may be achieved by merely including the "(w)" option to the TCL command line (or the input-macro in the defining item) for the Update sentence, e.g.,

```
item name phone zip comments (w)
```

Once you designate windows-mode, it propagates to all subsequent zooming and cruising until the process exits the Update processor and returns to TCL.

P Form Connective

The "P" connective may be used to override the use of a default form, or to create a new form (the (existing) default form).

would be:

```
file name form formname {attribute ...} {(options)}
```

file name is the name of the (Pick) file to update.

form name is the name (item.id) of the form which can be found in the dictionary of the specified file.

attribute is the attribute name (or attribute list) to use.

options are the options available to the Update processor.

See *Advanced Pick Reference Guide* (formerly *Epick*) for more information on Update processor).

GraPICK Pick/BASIC Changes

New Data Statement Extension

When working with a GraPICK client, an enhanced data statement support is implemented means that by using the Pick/BASIC "data" statement or "access (21)" function for stacked data, you can simulate almost any keystroke combination for Windows (including other applications, passing them data, etc.).

Each key is represented by one or more characters. To specify a single keyboard character itself, enclosed in double-quotes < " >. For example, to represent the letter statement for update looks like:

```
data "a"
```

Where a data statement character string is enclosed in single-quotes < ' >.

If you want to represent more than one character, include them all within a double-quoted :

```
data "abc"
```

The plus < + >, caret < ^ >, percent sign < % > and parentheses < () > characters have special meanings (see below). To specify one or more of these special characters, enter the character < { } >. For example, to specify the plus sign, use:

```
{+}.
```

To send the opening brace < { > or closing brace < } >, use:

```
{{} or
```

```
{}}.
```

To specify other keyboard keystroke characters such as <Enter> or <Tab>, use the following

Key	Code
Backspace	{backspace} or {bs} or {bksp}
Break	{break}
Caps Lock	{capslock}
Del	{delete} or {del}
Down Arrow	{down}
End	{end}
Enter	{enter}
	In UP-specific data, <Ctrl>M also will be interpreted as <Enter> (i.e., <code>data "abc"m'</code> will be interpreted as <code>abc{enter}</code>).
Esc	{escape} or {esc}
Home	{home}
Insert	{insert}

```

row      {left}
lock     {numlock}
down    {pgdn}
up      {pgup}
screen  {prtsc}
Arrow   {right}
Lock    {scrolllock}
        {tab}
ow      {up}
ough F12 {f1} through {f12}

```

keys combined with any combination of <Shift>, <Ctrl> and <Alt> key(s), precede the key with one or more of these codes:

Code

```

+
^
%
```

For example, if you wanted the data to be <Shift>bob, you could use **Bob** or **+bob**.

Characters < () > may be used to group character codes together. To indicate that <Shift>, <Ctrl> or <Alt> should be held down while several keys are pressed, enclose the key in < >. For example, to hold down the <Shift> key while pressing E and C, use + (EC). To hold down <Shift> while pressing E, followed by an unshifted C, use +EC. The ever-popular <Alt> sequence would be represented as (^%{del}).

To indicate a repeating character, use the syntax:

```
number}
```

There is a space between *key* and *number*. For example, {left 14} means press the "left" key fourteen times; {x 5} means press the character x five times.

Importing / Exporting Data Between the Host and the C GraPICK

GraPICK provides the user capability to exchange data between host and client.

The “gui.import” routine transfers DOS files into Pick items.

Syntax:

```
gui.import pickfile {item.ids} {(options)}
from: {dosfile} {dosfile} ...
```

pickfile The Pick filename to write to.

item.ids The Pick item(s) to write.
If no item.id is specified, then it will use all items, or if there is an
will refer to the list.

options Available options:
b binary items.
f items are Forms (DOS “.opf” files).

dosfile The DOS filename to read.

The “gui.export” routine transfers Pick items into DOS files.

Syntax:

```
gui.export pickfile {item.ids} {(options)}
to: {dosfile} {dosfile} ...
```

pickfile The Pick filename to read.

item.ids The Pick item(s) to read.
If no item.id is specified, then it will use all items, or if there is an
will refer to the list.

options Available options:
b binary items.
f items are Forms (DOS “.opf” files).

dosfile The DOS filename to write to.

Note: A “modified Kermit” protocol is currently implemented for the data transfers; however, future versions will fully support Kermit, Xmodem, Ymodem, Zmodem, etc. protocols.

Pick/BASIC "gui.lib" Subroutine Library

Include files and subroutines are in the "gui.lib" file for use with Pick/BASIC programs. Their source listings are provided, we recommend that you *do not* modify the source code in order to ensure compatibility with later releases.

```
rs
Color Table
apr kp
-----
C(28)

) = 16777215    ;* white
) = 0          ;* black
) = 12632256   ;* gray (light)
) = 4210752    ;* gray (dark)
) = 255        ;* red
) = 128        ;* red (dark)
) = 65535      ;* yellow
) = 32896      ;* yellow (dark)
) = 65280      ;* green
) = 32768      ;* green (dark)
) = 16776960   ;* sea-green
) = 8421376    ;* sea-green (dark)
) = 16711680   ;* blue
) = 8388608    ;* blue (dark)
) = 16711935   ;* lilac
) = 8388736    ;* lilac (dark)
) = 8454143    ;* yellow (light)
) = 4227200    ;* khaki
) = 8453888    ;* green (light)
) = 4210688    ;* gray-green
) = 16777088   ;* ice-blue (light)
) = 16744448   ;* ice-blue
) = 16744576   ;* violet (light)
) = 8404992    ;* violet
) = 8388863    ;* pink
) = 8388672    ;* violet (dark)
) = 4227327    ;* brown (light)
) = 16512      ;* brown
```



```

$properties
001 * properties definitions ( equal the dict PD file )
002 * 1 Jul 94 kp
003 *-----
004 equ COLOR.FILE to "pd.support,color.tbl"    ;* file with palette
005 equ SET.ON to 1, SET.OFF to 0, RIGHT to 1
006
007 equ MAIN to 0, SUBMENU to 0, POPUP to 1, SYSTEM to 2    ;* menu types
008
009 *--- values returned by W.MSG.BOX() subr
010 equ YES to 'Y', NO to 'N', CANCEL to 'C'
011 equ IGNORE to 'I', RETRY to 'R', OK to char(13)
012 *
013 *-- attributes/properties of an object
014 equ OBJ.TYPE to 1    ;* object type (see below for valid types)
015
016 equ X to 4    ;* horizontal coordinate (pixels)
017 equ Y to 5    ;* vertical coordinate (pixels)
018 equ WIDTH to 6    ;* width of an object (pixels)
019 equ HEIGHT to 7    ;* height of the object (pixels)
020
021 equ BOLD to 10    ;* future !!!
022 equ ITALIC to 11    ;* font italic if SET.ON
023 equ UNDERLINE to 12    ;* font underlined if SET.ON
024 equ STRIKEOUT to 13    ;* font stikeout if SET.ON
025 equ FSIZE to 14    ;* size of font (pixels)
026 equ FNAME to 15    ;* name of font ('script','modern' etc, see
027    ;*the 'PD.SUPPORT, FONTS' file for valid fontname
028 *-- for COLORS :
029 * RGB-code of the color from the $colors item.
    See the 'PD.SUPPORT,COLOR.TBL' file
030 equ BACKCOLOR to 16    ;* color of background
031 equ FORECOLOR to 17    ;* color of foreground
032 equ BORDERCOLOR to 18    ;* future
033
034 equ CAPTION to 19    ;* text to be written on a control
035 equ PICTURE to 19    ;* path for .bmp
036 equ TITLE to 19    ;* title of the WINDOW object
037 equ STYLE3D to 21    ;* Style 3D if SET.ON
038 equ BYCLOSE to 30    ;* string to be sent by closing the Window
039 equ BYLCLICK to 31    ;* string to be sent by left mouse click
040
041 *-- object types (valid value for OBJ.TYPE)
042 equ WINDOW to 0, DATA to 1
043 equ BUTTON to 2, TBBUTTON to 3, GBOX to 4, BMP to 5, STATEXT to 6
044 equ RBUTTON to 7, CHBOX to 8
045
046 *- properties of menus
047 equ GRAY to 40    ;* menu item is grayed if SET.ON
048 equ BAR.BREAK to 41    ;* menu item is 'bar broken' if SET.ON
049 equ BREAK to 42    ;* menu item is 'broken' if SET.ON
050 equ ALIGN to 43    ;* menu item is right aligned if RIGHT
051 equ CHECK to 44    ;* menu item is checked if SET.ON
052 equ SEPARATE to 45    ;* menu item is separated if SET.ON
053

```

ESC to 32 ;* string to be sent by ESC key pressed

subroutine activate.menu(*x*, *y*)

Subroutine to activate a predefined standalone popup menu.

x x-coordinate of the screen where the menu is to be placed.

y y-coordinate of the screen where the menu is to be placed.

subroutine add.menu.item(*path*, *item*, *MenuType*)

Subroutine to add an item at the end of the submenu.

path #Menu.level_1 : <am> : ... : #Menu.level_n
defines the hierarchy of menus this menu is connected to.
item properties of the menu item. (see the "\$properties" item above for details)
MenuType POPUP if standalone popup submenu.
SYSTEM if system menu.
otherwise, it is ordinary submenu.

subroutine change.object(*Nwdw*, *Nobj*, *Properties*)

Subroutine to change properties of the object. The object must be already created.

Nwdw window number.

Nobj object number.

Properties dynamic array, defining properties of the object (see the "\$properties" item above for details).

subroutine change.wdw.title(*NewTitle*)

Subroutine to change a title of the window.

NewTitle new title for the window.

subroutine check.menu.item(*path*, *mode*, *MenuType*)

Subroutine to make a line of the menu checked/unchecked.

path #Menu.level_1 : <am> : ... : #Menu.level_n.
defines the hierarchy of menus this menu is connected to.
mode 0 means "set checked", else "set unchecked."
MenuType 0 - ordinary submenu.
1 - standalone popup submenu.
2 - system menu.

`line clear.data.obj(Nwdw, Nobj)`
 routine to clear all values of the "data" object.

 window number.
 object number.

`line clear.tb`
 routine to clear the tool bar of the currently active window.

`line clear.wdw(Nwdw)`
 routine to clear the specified window.

 window number.

`line create.menu(Menu)`
 routine to create a main menu.

 properties of the menu (see the the "\$properties" item above for details).

`line create.object(Nwdw, Nobj, Properties)`
 routine to create an object with specified properties.

 window number.
 object number.

`Properties` dynamic array, defining properties of the object (see the "\$properties" item above for details).

`line create.submenu(path, Menu, isPopUp)`
 routine to create a submenu.

 #Menu.level_1 : <am> : ... : #Menu.level_n.
 defines the hierarchy of menus this menu is connected to.
 properties of the menu (see the the "\$properties" item above for details).

`isPopUp` "0" means standalone popup menu.
 If it is a standalone popup menu, it will have to be activated by "call activate.menu(x,y)."

subroutine create.value(*Nwdw*, *Nobj*, *Nval*, *data*)
 Subroutine to create a value of the object.

Nwdw window number.
Nobj object number.
Nval value number.
data text data to fill the value.

subroutine del.menu(*MenuType*)
 Subroutine to clear and delete menu(s). If it is the system menu, it restores the original

MenuType 0 - ordinary submenu.
 1 - standalone popup submenu.
 2 - system menu.

subroutine del.menu.item(*path*, *MenuType*)
 Subroutine to delete an item from the menu.

path #Menu.level_1 : <am> : ... : #Menu.level_n.
 defines the hierarchy of menus this menu is connected to.
MenuType POPUP if standalone popup submenu.
 SYSTEM if system menu.
 otherwise, it is ordinary submenu.

subroutine del.value(*Nwdw*, *Nobj*, *Nval*)
 Subroutine to delete a value from the object.

Nwdw window number.
Nobj object number.
Nval value number to delete.

subroutine del.wdw(*Nwdw*)
 Subroutine to clear and delete the specified window.

Nwdw window number.
 If *Nwdw* = "" or *Nwdw* = 0 then the currently active window is delet

`line form.fn(Nwdw, formname)`
Routine to send *formname* to the client. This name will be used when the form is saved ("save ty" mode).

Nwdw window number.
formname name of the ".opf" file/item on server where the form will be saved to.

`line form.vseq(Properties, vseq, chg.flg)`
Level common subroutine to reformat "properties" to GUI virtual sequence.

Properties dynamic array, defining properties of the object (see the "\$properties" item above for details).
vseq sequence to be returned.
chg.flg 0 - if creating *vseq* to CREATE the object.
1 - if creating *vseq* to CHANGE properties of the object.

`line get.color(code, color)`
Routine to get color value through its number.

code color code to be returned.
color color value desired.

`line gray.menu.item(path, mode, MenuType)`
Routine to set an item of the menu grayed/ungrayed.

path #Menu.level_1 : <am> : ... : #Menu.level_n.
defines the hierarchy of menus this menu is connected to.
0 - means "set grayed."
else "set ungrayed."
Type 0 - ordinary submenu.
1 - standalone popup submenu.
2 - system menu.

`line hg.OFF`
Routine to set cursor shape from "hourglass" to "arrow" and allow input through the keyboard/mouse.

subroutine hg.ON

Subroutine to set cursor shape to "hourglass" and block any input from the keyboard/n

subroutine highlight.menu.item(*path*, *mode*, *MenuType*)
Subroutine to set an item in the menu highlighted/normal.

path #Menu.level_1 : <am> : ... : #Menu.level_n.
 defines the hierarchy of menus this menu is connected to.
mode 0 - "set highlight."
 else "set lowlight."
MenuType 0 - ordinary submenu.
 1 - standalone popup submenu.
 2 - system menu.

subroutine ins.menu.item(*path*, *item*, *MenuType*)
Subroutine to insert an item into the submenu.

path #Menu.level_1 : <am> : ... : #Menu.level_n.
 defines the hierarchy of menus this menu is connected to.
item properties of the menu item (see the "\$properties" item above for de
MenuType POPUP if standalone popup submenu.
 SYSTEM if system menu.
 otherwise, it is ordinary submenu.

subroutine process.menu.item(*path*, *item*, *MenuType*, *code*)
Low-level subroutine to add / delete / replace / insert a menu item.

path #Menu.level_1 : <am> : ... : #Menu.level_n.
 defines the hierarchy of menus this menu is connected to.
item properties of the menu item (see the "\$properties" item above for de
MenuType POPUP if standalone popup submenu.
 SYSTEM if system menu.
 otherwise, it is ordinary submenu.

subroutine rearrange.menu

Subroutine to rearrange all menus that were changed, otherwise, the changes don't tak

`.ne rearrange.objects(Nwdw)`
 routine to activate all predefined objects.

Nwdw window number.

`.ne repl.menu.item(path, item, MenuType)`
 routine to replace an item in the submenu.

path #Menu.level_1 : <am> : ... : #Menu.level_n.
 defines the hierarchy of menus this menu is connected to.
 properties of the menu item (see the "\$properties" item above for details).
MenuType POPUP if standalone popup submenu.
 SYSTEM if system menu.
 otherwise, it is ordinary submenu.

`.ne RGB(r, g, b, color)`
 routine to return a color value as a mixture of values of red, green and blue.

r 8-bit value for red.
 g 8-bit value for green.
 b 8-bit value for blue.
 color Composite 24-bit color value returned where:
 xxxxxxxx = the "red" value.
 xxxxxxxx = the "green" value.
 xxxxxxxx = the "blue" value.

`.ne run.wdw.prg(PrgName)`
 routine to run the specified window program.

PrgName name of the program (for example : "pr.exe").

`.ne set.focus(Nwdw, Nobj, Nval)`
 routine to set focus pointed to the value of the object.

Nwdw window number.
 Nobj object number.
 Nval value number.


```
subroutine sts.line( txt )
```

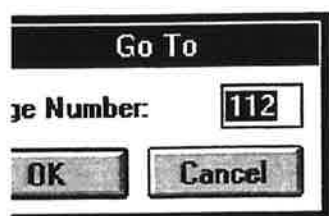
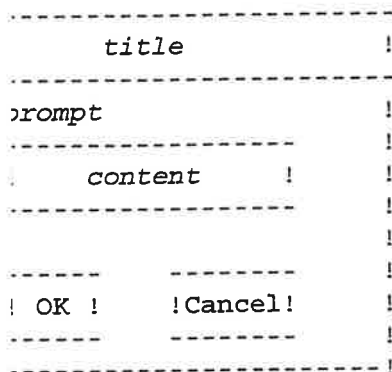
Subroutine to output text to the status line (at the bottom of the screen).

txt

Text to output on the status line.

line `w.dlg.box(title, prompt, content, ret)`
 line to get input through standard window modal dialog box.

- `title` window dialog box title.
- `prompt` window dialog box prompt.
- `content` window dialog box input string.
- `ret` string to be sent to server by click.



box that contains three Windows control elements:
 "title" window for an input string ("content" above)
 "OK" button and
 "Cancel" button.

When the user enters data (if any) into the "content" area, and presses <Enter> or a button.
 The "OK" button sends the entered string (if any) and a char(13) to the server.
 The "Cancel" button sends <Esc> to the server. Any input data will be lost.

subroutine w.msg.box(*Nicon*, *ButtonCode*, *title*, *prompt*, *ret*)
 Subroutine to create a standard window message box.

Nicon icon name (numeric).
 1 - "STOP" symbol.
 2 - "?" question mark.
 3 - "!" exclamation mark.
 4 - "i" information sign.

ButtonCode (the default button is underlined):

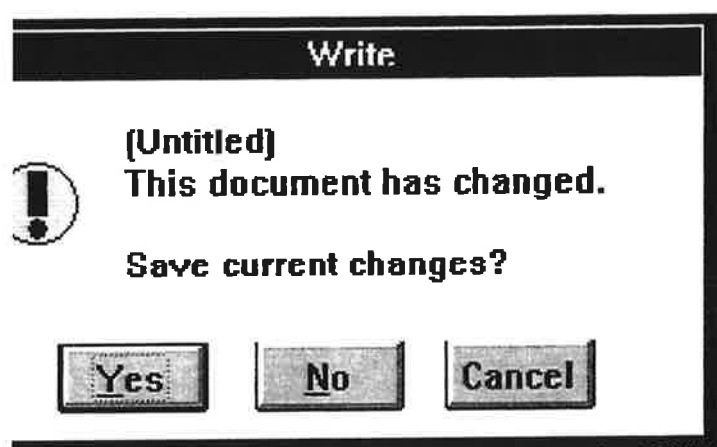
Code	Button definition(s)			Returns		
1	<u>OK</u>			x'0D'		
2	<u>Yes</u>	No		Y	N	
3	Yes	<u>No</u>		Y	N	
4	<u>Yes</u>	No	Cancel	Y	N	C
5	Yes	<u>No</u>	Cancel	Y	N	C
6	Yes	No	<u>Cancel</u>	Y	N	C
7	<u>Abort</u>	Retry	Ignore	A	R	I
8	Abort	<u>Retry</u>	Ignore	A	R	I
9	Abort	Retry	<u>Ignore</u>	A	R	I
10	<u>OK</u>	Cancel		x'0D'	C	
11	OK	<u>Cancel</u>		x'0D'	C	
12	<u>Retry</u>	Cancel		R	C	
13	Retry	<u>Cancel</u>		R	C	

title message box title text.
prompt message box prompt text.
ret returns pressed.

```

-----
!-!          title          !
-----
!
!          ICON          prompt !
! (depends on Nicon)      !
!
!          LINE OF BUTTONS    !
!          []          []          [] !
! (depends on ButtonCode) !
-----

```



mming Using Pick/BASIC

il Strategy

ly straightforward to generate GraPICK screens by a Pick/BASIC application using the
vided by GraPICK in the "gui.lib" file:

Creating a Form Item

Building a form layout is simply the process of object-by-object creation (in Windows). It should be described before its creation. To describe an object means to define its type (window, graphic, tag, menu, etc.) and other properties (coordinates, size, colors, etc.). Once it is described, it may be created by the appropriate subroutine out of `gui.lib`.

Note: The WINDOW object should be created *first*. The following table may help you in choosing the proper subroutine to create the object:

Object	Object type	Subroutines to create object	Notes
1. Window	WINDOW	<code>create.object()</code>	WINDOW object should be <i>first</i> !
2. Databox	DATA	<code>create.object()</code> <code>create.value()</code>	Creates an empty data box. Fills the box by value.
3. Tag	STATEXT	<code>create.object()</code>	
4. Buttons	BUTTON TBBUTTON RBUTTON	<code>create.object()</code> <code>create.object()</code> <code>create.object()</code>	
5. Groupbox	GBOX	<code>create.object()</code>	
6. Checkbox	CHBOX	<code>create.object()</code>	
7. Graphic	BMP	<code>create.object()</code>	
8. Menus	MAIN	<code>create.menu()</code> <code>rearrange.menu()</code>	Creates the main menu. Makes the precreated menu
9. Submenus	SUBMENU	<code>create.submenu()</code> <code>rearrange.menu()</code>	Creates the submenu. Makes the precreated menu
10. Submenus	POPUP	<code>create.submenu()</code> <code>activate.submenu()</code>	Creates the standalone popup menu. Activates the precreated standalone popup menu

Changing the Form Item

Objects are created, you may modify them at any time. There are a number of "gui.lib" routines that provide the Pick/BASIC application with this possibility. The table below will help you to choose the appropriate one.

Only the following property may be changed, except "Object Type."

Object type	Subroutines to modify properties
WINDOW	change.object()
DATA	change.object() create.value()
STATEXT	change.object()
BUTTON	change.object()
TBBUTTON	change.object()
RBUTTON	change.object()
GBOX	change.object()
CHBOX	change.object()
BMP	change.object()
POPUP	add.menu.item()
SYSTEM	del.menu.item()
SUBMENU	ins.menu.item() ins.menu.item() repl.menu.item() highlight.menu.item() gray.menu.item() check.menu.item()

The list below describes functions supported by GraPICK:

Function	Subroutine to perform
1. Output to status line	sts.line()
2. Change title of the Window	chg.wdw.title()
3. Set focus to the value	set.focus()
4. Change cursor to "Hourglass"	hg.ON / hg.OFF
5. Activate Message Box	w.msg.box()
6. Activate Dialog Box	w.dlg.box()
7. Notify client about FormName	form.fn()
8. Clear Tool Bar	clear.tb()
9. Clear window	clear.wdw()

Editing Form Items

Local Edit Mode

Local Edit mode of the Properties menu lets you inspect or modify properties of the objects on the client side.

Clicking on the object invokes the Properties window for this object.

Changes of properties values on the Properties window are propagated to the screen when you click the OK button.

Clicking the Local Edit mode item of the Properties menu closes the local edit mode.

Screen modifications can be saved on DOS and/or Pick for further usage *after* you select the Properties mode of the Properties menu.

AP Edit Mode

AP Edit mode on the Properties menu lets you to inspect or modify properties of the objects on the host (server) through the UPDATE processor.

When in AP Edit mode, the form (“opf” file) is transferred to Pick and every object on the screen is presented as an item on the PD (PickDesktop) file with attributes reflecting the current state of the object. If you change any value of these properties, the new values are immediately reflected to the screen.

Clicking on the “Id” field or right mouse-click on the object to invoke the UPDATE processor for this object.

Clicking the AP Edit mode on the Properties menu closes the AP Edit mode.

Screen modifications are *not* saved on DOS and/or Pick for further usage until you select the Properties mode of the Properties menu.

AP Edit mode is not fully implemented for most screen or object modifications. The Local Edit mode is currently the preferred technique.

Getting Input Information From an Object

Once an event is performed on an object (based on the event being previously defined to via one of its properties), such as when a mouse-click occurs on it (ByLClick, ByESC, B; the application can receive this input string using the Pick/BASIC in statement.

Note: When programming properties such as ByLClick etc., the return string must be terminated with a carriage return, or char(13).

Windows Term Definitions

Working with GraPICK means dealing with Microsoft Windows. Microsoft has its own terminology for concepts and ways of doing things. Here are a few of the more primary terms you will need to know.

Form - a DOS filename extension used by GraPICK for storing its modified form definitions, containing all of the form's objects with their definitions and component objects, etc.

Graphics (.bmp)

Graphics (BMP) are objects to display graphical data. Bitmaps can be created/modified either in the AP/Local Edit modes of the Properties menu or routines of the "gui.lib" file. By making a call to the "showicon" routine on the output conversion of a dictionary item (say, "pic"), this would allow the Update processor to display a 256-color bitmap ("*.bmp") by simply indicating its DOS path (e.g., "d:\dos\path\pic.bmp") of where the graphic is located in DOS. Also, see the program "main" for examples on how to reference a BMP from a BASIC program.

While in the Update processor, only one (1) BMP may be simultaneously displayed in any form (window). Also, the default location of the BMP is the top-right section of the window, so your window should be resized accordingly. The graphic may be moved around in the window and subsequently all graphics will be displayed in that location, if the form is saved.

Public Data Exchange. A Microsoft standard for messaging and exchanging data between Windows applications.

Records are objects to hold/display data of databases, for example, providing access to the data and allowing selected actions to be performed on the data (such as "cruising" from item to item in the Update processor.) Objects of this type can be created/modified through either the AP/Local Edit modes of the Properties Menu or routines of the "gui.lib" file.

Events are the attribute values for a given object. They define actions to be performed when the events associated with the object occur. The following list describes correspondence between the events of the "pd" (PickDesktop) file and events defined for the object:

Attribute #	Event
1	string to be sent by closing the window.
2	string to be sent by left mouse-click on the object.
3	string to be sent by <Esc> key pressed when on menu.

33	(Reserved)
34	(Reserved)
35	(Reserved)
36	(Reserved)
37	(Reserved)
38	(Reserved)
39	(Reserved)

Focus

A term indicating which data field is currently being examined.

Form

A window object that makes up part of an interface. When the GraPICK screen layout window object should be created first. Window objects can be created/modified through AP/Local Edit modes of the Properties menu or routines of the "gui.lib" file.

Form Title

The property of the window object to determine the text displayed in the Window's title bar. When the window is minimized, this text is displayed below the icon.

FTP

File Transfer Protocol. A protocol defined for transferring information between computers.

GUI

Graphical User Interface (usually pronounced "goeey"). A term used to denote a user interface using graphical images, icons, pointers and other images. Windows is a GUI interface on DOS.

Menu Items

Control objects used to display a customized menu for the application. The menu is programmed and/or modified through routines of the "gui.lib" file.

ODBC

Open database connectivity. A Microsoft standard for interfacing with databases.

Objects

Elements defining the layout of a screen. There are three categories of objects:

- window,
- data and
- control.

Every object is explicitly defined by the list of its properties.

Properties

These are the values for a given object, such as color, type of font, size, location, etc. This includes a list of actions to be performed when various predefined events occur that have been attached to the object. The following list describes the correspondence between attributes of the PickDesktop) file and properties of the object:

Attribute #	Property
	object type (WINDOW, DATA, BUTTON, TBBUTTON, GBOX, BMP, STATEXT, RBUTTON, CHBOX)
	(Reserved)
	(Reserved)
	horizontal coordinate (pixels)
	vertical coordinate (pixels)
	width of an object (pixels)
	height of the object (pixels)
	(Reserved)
	(Reserved)
	(Reserved)
	font italic
	font underlined
	font strikeout
	size of font (pixels)
	name of font ("script", "modern", etc.)
	color of background
	color of foreground
	(Reserved)
	text to be displayed on a control
	full path name of .bmp
	title of the WINDOW object
	(Reserved)
	Style 3D
	(Reserved)
	(Reserved)
	(Reserved)
	(Reserved)
	(Reserved)
	(Reserved)
	(Reserved)
	(Reserved)
	(Reserved)
	See: Events
	See: Events
	See: Events
	See: Events
	See: Events

35	See: Events
36	See: Events
37	See: Events
38	See: Events
39	See: Events
40	menu item grayed
41	menu item "bar broken"
42	menu item "broken"
43	menu item right aligned
44	menu item checked
45	menu item separated

Note: As these are being extended, their attribute numbers are subject to change.

OLE

Not a Spanish expression of excitement, but an acronym for Object Linking and Embedding. A Microsoft standard for supporting embedded objects within objects.

Properties

See "Object Properties."

Tag Fields

Fields are static control objects logically bound to corresponding Data Boxes. Cruising operations can be performed when on the Tag Field as well as when on the bound Data Boxes. Objects can be created/modified through either the AP/Local Edit modes of the Properties or routines of the "gui.lib" file.

3-D Presentation

An optional style of the window. This property is inherited by all objects of the window.

Note : When in 3-D mode, objects are system-colored and use a fixed border style.

Window

See "form."

les

ing are some sample routines that demonstrate some of the features of GraPICK.

les of GraPICK via Pick/BASIC programs

ib,samples" file provides you with two programs to demonstrate methods of writing pick/BASIC applications: "demo.1" and "main."

emo.1"

e shows an example of how an ordinary Pick/BASIC data-entry screen can look after being PICK-aware."

andard" is also provided for reference. It is the same routine as "demo.1," but it was the sion, before it was modified to work with GraPICK.

es prompt the user to input data and check validity of inputs against the length restrictions ks.

nteresting part of "demo.1" is the "data.entry.screen" subroutine that creates a Windows-n and prompts for values. It performs the following:

es access to GraPICK constants and definitions, such as names of properties, valid values of ties etc. (line #42).

: a WINDOW object with Exit event being performed when the window is closed (lines).

: the Exit Tool Bar Button to close and delete the window on the condition that all data ed are received,

firmation on the exit is got (lines #59-67).

: prompts and empty data fields for input, sized according to their definitions (lines #69-

es the data-input mode through the Pick/BASIC "in" statement until the Exit event is (lines #106-158).

When a value is received, it is examined. If it is a data-entry value (lines #128-141) or the sent by the Exit button or the user's attempt to close the window (lines #144-156), it deletes adow when the Exit event occurs.

t, click on the Exit button.

```

demo.1
001 !
002 * demo data-entry screen program
003 * 17 may 94 kp
004 *
005 *----- data defining grapick screen and input data
006 header = "Welcome to GraPICK" ;* window title
007 *- prompts ----- number of values -- length restrict -- input maska
008 tags      ="Customer Id:"; Vqty      =1;   Length      =8 ;   Maska      ='mcp'
009 tags<-1>="Name:"          ; Vqty<-1>=1;   Length<-1>=25;  Maska<-1>='mc/n
010 tags<-1>="Phone(s):"      ; Vqty<-1>=2;   Length<-1>=9;   Maska<-1>='mc/a
011 tags<-1>="Address:"       ; Vqty<-1>=3;   Length<-1>=25;  Maska<-1>='
012
013 Input = "" ;* input data. will be return by the 'data.entry.screen' su
014
015 *-- call subr which 1.creates a grapick screen 2.gets input data like
016 *-- "input @(col,row) var 'maska'" <problems with 'input var,len:{_}'>
017 *-- returns dim array Input and 'done'-flg (=0 if data-entry interrupted
018 *-- by user not completed )
019
020 gosub data.entry.screen
021
022 crt; crt '----- RESULTS ':
023 if done then crt '(completed)': else crt '(not completed)':
024 crt '-----'
025 mask = 'm#':len+2
026 for i=1 to max
027     crt tags<i> mask:Input<i,1>
028     for j=2 to Vqty<i>; crt space(len+2):Input<i,j>; next j
029 next i
030 crt '-----'
031
032
033 STOP ;* end of demo input
034 *-----
035 data.entry.screen:
036  !!subroutine data.entry.screen(header,tags,Vqty,Length,Maska,Input,don
037 *
038 equ AM to char(254), VM to char(253)
039 equ ENTER to 13
040 equ EXIT to char(24), CTRLZ to char(26)
041
042 $include gui.lib $properties
043
044 Wn = system(16) ;* window name
045
046 *--- create a screen
047 P = ""
048 P<OBJ.TYPE> = WINDOW
049 P<BACKCOLOR> = "#15"
050 P<FORECOLOR> = "#3"
051 *- 1. create a window < with Exit button and ExitByClose >
052 P<X> = 1; P<Y> = 1; P<WIDTH> = 400; P<HEIGHT> = 350
053 P<TITLE> = header
054 P<STYLE3D> = SET.OFF

```

```

LOSE> = CTRLZ: EXIT: char(ENTER)

create.object(Wn, 1, P)

create an Exit button.
'
.TYPE> = TBBUTTON
= 22; P<Y> = 1; P<WIDTH> = 50; P<HEIGHT> = 16
TION> = "EXIT"
ME> = "Script"
CLICK> = CTRLZ: EXIT: char(ENTER)

create.object(Wn, dcount(tags,am)+1, P)

create tags and empty fields for data objects.
dcount(tags,am)
0
=1 to max
len < len(tags<i>) then len = len(tags<i>)
i
= 8 ;* width of a sym
15; y1 = 10; w = wsym*(len+1); h = 28

'
.TYPE> = STATEXT
KCOLOR> = "#15"
ECOLOR> = "#3"
ERLINE> = SET.ON
'
.TYPE> = DATA
dcount(tags,am)
obj=1 to max
<CAPTION> = tags<Nobj>
<X> = xx; P<Y> = y1+4; P<WIDTH> = w; P<HEIGHT> = h-4
all create.object(Wn, Nobj, P)

<X> = xx+w+s; D<Y> = y1
<WIDTH> = wsym*(Length<Nobj>+3); D<HEIGHT> = h*Vqty<Nobj>
<BYCLICK> = CTRLZ :Nobj: "V"
all create.object(Wn, Nobj, D)

or Nval=1 to Vqty<Nobj>
call create.value(Wn, Nobj, Nval, "")
ext Nval

l = y1 + h*Vqty<Nobj> + s
Nobj

rearrange objects of the form
rearrange.objects(Wn)

loop for input data
= 0 ;* flag all necessary values are entered
= 0 ;* flag if it's an attempt to leave the data-entry screen
= 1; Nval = 1

```



```

110 loop until left do                                ;* main loop to get all necessary values
111   call set.focus(Wn,Nobj,Nval);* set focus (cursor) to the current valu
112   string = ""                                     ;* currently entered value buffer
113   ECHO OFF
114   loop
115     in key
116   until key=ENTER do ;*or Length<Nobj> exceeded
117     string = string : char(key)
118   repeat
119   ECHO ON
120
121   *-- parse the string : info & virtual sequence ByClick
122   posCTRLZ = index( string, CTRLZ, 1)
123   if posCTRLZ then
124     str = string[1, posCTRLZ-1]
125     vir = string[ posCTRLZ+1,9999]
126   end else str = string; vir = ""
127
128   if str # "" then ;* there is entered data
129     cstr = str
130
131     *-- check maska
132     if Maska<Nobj> # "" then cstr = cstr Maska<Nobj>
133
134     *-- check length
135     if Length<Nobj> # "" then cstr = cstr[1,Length<Nobj>]
136
137     *-- re-fill the value if changed
138     call create.value(Wn,Nobj,Nval,cstr)
139
140     Input<Nobj,Nval> = cstr
141   end
142   if not(done) then done = ( Nobj = max and Nval = Vqty<Nobj> )
143
144   if vir = EXIT then ;* user's attempt to leave the screen
145     if done then
146       left=1 ;* ok
147     end else
148       call w.msg.box(2,3,'WARNING','Data entry not completed. Exit?',
149         left = (answ = YES)
150     end
151   end else
152     if vir[1,1] # 'i' then
153       Nobj = field( vir, 'V', 1)
154       Nval = field( vir, 'V', 2)
155     end else execute vir[2,99999]
156   end
157
158 repeat
159
160 *-- End of GraPICK session. deleting the window
161 call del.wdw("")
162
163 return
164

```


program "main"

This routine gives more samples of GrapICK capabilities.

Note: lines #6-7 are the includes of predefined constants and GrapICK definitions, incl color table. The routine demonstrates how to :

- create a WINDOW object (lines #16-26).
- create a Main Menu object (lines #29-40).
- create 2-level Submenus (lines #42-48).
- perform the operations possible on Menu (lines #50-95).
- create objects such as:
 - Tool Bar Button (lines #100-109).
 - Group Box (lines #112-124).
 - Button (lines #126-137).
 - Radio Button (lines #139-151).
 - Check Box (lines #153-165).
 - Graphic (lines #167-176).
- color palette as a set of Static Texts (lines #178 - 197).
- Static Text with updated fonts properties (lines #199 - 212).
- change the Window Title (lines #214-216).
- output to the Status Line (lines #219-221).
- create two Tags/DataFields objects varying fonts, colors, sizes of the objects and p Tags' events by the mouse-click (lines #224-292).
- clear/delete value in the multivalued data field (lines #295-297).
- clear the Tool Bar (line #300).
- get input from all the above objects and react on it (lines #305-344).
- activate Windows Message Box by the mouse-click on Tag (lines #238-239).
- activate Windows Dialog Box (lines #326-328).
- set focus to the data value (line #257).
- delete Window by the user's attempt to close the window or mouse-click on the Exit l #25-26, #108-109).

To exit, click on the **Toolbar** button. Note the effects of using the slide-bar on the "First" c

```

main
001 * program to test GUI.exe performance and/or give samples of Windows
002 * objects creation.
003 * 12 jan 94 kp.
004 * Updated by: 14 jul 94 kp - new version
005 *-----
006 $include gui.lib $properties
007 $include gui.lib $colors          ;* 28-color table (dim TC(28)
008
009 equ AM to char(254), VM to char(253)
010 equ ENTER to 13, CTRLZ to char(26)
011 *-----
012 Nw = system(16)                  ;* Window Name
013 EXIT = 'i':'delete.wdw'
014 DIALOG = 'dialog'
015

```

```

CREATE WINDOW
'
      ;* Properties of an object
.TYPE> = WINDOW
      = 10
      = 15
TH>    = 600
GHT>   = 400
KCOLOR>= TC(22)      ;* ice-blue background
LE>    = "Window Title"
LOSE>  = CTRLZ :EXIT:char(ENTER)
create.object(Nw, 1, P )
-----
YPASS
CREATE MAIN MENU.
'
      ;* Properties of Main Menu.
''; M2 = ''      ;* Properties of SubMenus.
=1 to 3
APTION, i> = "Menu Item &":i
APTION, i> = "Mode &":i
APTION, i> = "SubMode &":i
i
.BREAK, 2> = SET.ON
GN, 3> = RIGHT
EAK, 3> = SET.ON
all create.menu( M )
-----
reate a first level Submenu.
ath = 1; MenuType = SUBMENU
all create.submenu( path, M1, MenuType )
-----
reate a second level Submenu.
ath = 1:am:3;
all create.submenu( path, M2, SUBMENU )
-----
- TESTING MENU PART -----
  Activate the Submenu if it is a standalone PopUp.
n = 50; ym = 125
call activate.menu( xm, ym )
-----
dd an item to the first level Submenu.
= ''
CAPTION> = "NewMenuItem"
= 0
add.menu.item( path, item, SUBMENU )
-----
reate a first level Submenu.
= 3
create.submenu( path, M2, SUBMENU )
-----
  Replace the second item of the first level Submenu.
tem = ''
tem<CAPTION> = "Replaced value"; path = 3:am:2
all repl.menu.item( path, item, SUBMENU )
-----
nsert an item before the third item of the first level Submenu.

```

```

071 item = ''
072 item<CAPTION> = "Inserted value"
073 item<GRAY> = SET.ON
074 path = 3:am:3
075 !!call ins.menu.item( path, item, SUBMENU )
076 *-----
077 *   Delete the second item from the first level Submenu.
078   path = 3:am:2
079 !   call del.menu.item( path, SUBMENU )
080 *-----
081 *   Set the second item of the first level Submenu grayed.
082 !!call gray.menu.item( path, SET.ON, SUBMENU )
083 *-----
084 *   Set the third item of the first level Submenu checked.
085 path = 3:am:3
086 !!call check.menu.item( path, SET.ON, SUBMENU )
087 *-----
088 *   Set the fourth item of the first level Submenu highlight.
089 path = 3:am:4
090 !!call highlight.menu.item( path, SET.ON, SUBMENU )
091 *---- end of TESTING MENU part -----
092 *
093 *   Rearrange menu because of changes
094 call rearrange.menu
095 *-- END of MENU-part -----
096 *-----
097 BYPASS:
098 NCobj = 1      ;* Counter of Controls
099 *-----
100 *   Create a tool bar button.
101 P = ''
102 P<OBJ.TYPE> = TBUTTON
103 P<CAPTION> = "Toolbar Button"
104 P<X> = 30
105 P<Y> = 1
106 P<WIDTH> = 120
107 P<HEIGHT> = 18
108 P<BYLCLICK> = CTRLZ:EXIT:char(ENTER)
109 call create.object( Nw, NCobj, P)
110
111 *-----
112 *   Create a group box
113 NCobj = NCobj+1
114 P = ''
115 P<OBJ.TYPE> = GBOX
116 P<CAPTION> = "Group Box"
117 P<X> = 10
118 P<Y> = 240
119 P<WIDTH> = 270
120 P<HEIGHT> = 80
121 P<BACKCOLOR> = TC(11)      ;* sea blue
122 P<FORECOLOR> = TC(26)     ;* violet dark
123 P<FNAME> = 'script'
124 call create.object(Nw, NCobj, P)
125 *-----

```

```
create a button.  
= NCobj+1  
,  
.TYPE> = BUTTON  
TION> = "Button"  
= 175  
= 270  
FH> = 70  
EHT> = 20  
ME> = 'script'  
CLICK> = "tufta"  
create.object(Nw, NCobj, P)  
-----  
create a radio button  
= NCobj+1  
,  
.TYPE> = RBUTTON  
TION> = "Radio Button"  
= 20  
= 270  
FH> = 120  
EHT> = 20  
KCOLOR> = TC(7) ;* yellow  
ECOLOR> = TC(26) ;* violet (dark)  
LIC> = SET.ON  
create.object(Nw, NCobj, P)  
-----  
create a check box  
= NCobj+1  
,  
.TYPE> = CHBOX  
TION> = "Check Box"  
= 20  
= 295  
FH> = 120  
EHT> = 20  
KCOLOR> = TC(7) ;* yellow  
ECOLOR> = TC(26) ;* violet (dark)  
LIC> = SET.ON  
create.object(Nw, NCobj, P)  
-----  
create a graphic  
= NCobj+1  
,  
.TYPE> = BMP  
TURE> = "advpick.bmp"  
= 290  
= 30  
FH> = 320  
EHT> = 240  
create.object(Nw, NCobj, P)  
-----  
lette
```

```

181 P<OBJ.TYPE> = STATEXT
182 P<X> = 10
183 P<WIDTH> = 19
184 P<HEIGHT> = 30
185 for i=1 to 28 step 2
186     NCobj = NCobj+1
187     P<Y> = yp
188     P<BACKCOLOR> = TC(i)
189     call create.object(Nw, NCobj, P)
190
191     NCobj = NCobj+1
192     P<Y> = yp + P<HEIGHT>
193     P<BACKCOLOR> = TC(i+1)
194     call create.object(Nw, NCobj, P)
195
196     P<X> = P<X> + P<WIDTH>
197 next i
198 *-----
199 * Create a static texts
200 NCobj = NCobj+1
201 P = ''
202 P<OBJ.TYPE> = STATEXT
203 P<CAPTION> = " Static Text"
204 P<X> = 370
205 P<Y> = 295
206 P<WIDTH> = 160
207 P<HEIGHT> = 20
208 P<BACKCOLOR> = TC(14)
209 P<FORECOLOR> = TC(22)
210 P<FSIZE> = 20
211 P<FNAME> = 'courier'
212 call create.object(Nw, NCobj, P)
213 *-----
214 *   Change the window title.
215     NewTitle = "My NEW title"
216 !   call change.wdw.title( NewTitle )
217 *-----
218
219 *   Output to the Status line.
220 txt = "Status Line"
221 call sts.line( txt )
222 *-----
223 bypass1:
224 *   Create the first object/attribute
225 *       Substitute-header ( tag )
226 NCobj = NCobj+1
227 P = ''
228 P<OBJ.TYPE> = STATEXT
229 P<CAPTION> = "First "
230 P<X> = 7
231 P<Y> = 120
232 P<WIDTH> = 60
233 P<HEIGHT> = 20
234 P<BACKCOLOR> = TC(25)      ;* pink
235 P<FORECOLOR> = TC(4)      ;* gray

```

```

ZE>      = 20
VE>      = "Script"
CLICK>   = CTRLZ:"i": "get.msg.box":char(ENTER)
create.object(Nw, NCobj, P)

      Empty field.
= 1          ;* Counter of DATA objects
,
.TYPE>    = DATA
          = 70
          = 120
FH>       = 90
GHT>      = 86
create.object(Nw, NDObj, P)

      Fill the attribute by values.
val=1 to 5
l create.value( Nw, NDObj, Nval, "Value ":Nval )
Nval
      Set focus to the second value
= 2
set.focus( Nw, NDObj, Nval )

      Delete the second value
Nval = 2
call del.value( Nw, NDObj, Nval )
-----

Create the second object/attribute
= NCobj+1
,
.TYPE>    = STATEXT
TION>     = "Second "
          = 170
          = 120
FH>       = 50
GHT>      = 20
KCOLOR>   = TC(25)      ;* pink
ECOLOR>   = TC(4)       ;* gray
ERLINE>   = SET.ON
VE>       = 'script'
CLICK>    = DIALOG:char(ENTER)
create.object(Nw, NCobj, P)

= second data object
= NDObj+1
,
.TYPE>    = DATA
          = 225
          = 120
FH>       = 60
GHT>      = 86
create.object(Nw, NDObj, P)

val=1 to 3

```



```

291 call create.value( Nw, NObj, Nval, "Val ":Nval )
292 next Nval
293 *-----
294 *
295 *   Clear/delete the first object/attribute
296   NObj = 1
297 ! call clear.data.obj( Nw, NObj )
298 *-----
299 *   Clear Tool Bar.
300 !call clear.tb
301 *-----
302 *   Activate all pre-defined objects
303 !call rearrange.objects( Nw )
304 *-----
305 *--- loop for input data
306 left = 0      ;* flag if it's an attempt to leave the data-entry screen
307 loop until left do      ;* main loop to get input
308   string = ""      ;* currently entered value buffer
309   ECHO OFF
310   loop
311     in key
312   until key=ENTER do
313     string = string : char(key)
314   repeat
315     ECHO ON
316
317   *-- parse the string : info & virtual sequence ByClick
318   posCTRLZ = index( string, CTRLZ, 1)
319   if posCTRLZ then
320     str = string[1, posCTRLZ-1]
321     vir = string[ posCTRLZ+1,9999]
322   end else str = string; vir = ""
323
324   if str # "" then
325     if str = DIALOG then
326       content = "delete.wdw"; ttl = "Dialog Box Title"
327       prompt="Dialog Box Prompt"
328       call w.dlg.box( ttl, prompt, content, line)
329       vir = 'i': line ;* TCL command to be executed
330     end else
331       call create.value(Nw,NObj,Nval,str)
332     end
333   end
334   left = (vir = EXIT)      ;* user's attempt to leave the screen
335
336   if vir[1,1] # 'i' then
337     NObj = field( vir, 'V', 1)
338     Nval = field( vir, 'V', 2)
339     call set.focus(Nw,NObj,Nval);* set focus to the current value
340   end else
341     tcl.cmd = vir[2,9999]
342     if tcl.cmd # '' then TCL tcl.cmd
343   end
344 repeat
345

```

nd of GraPICK session.

“u entity (w)” - a sample UPdate command

A sample UPdate processor screen using the “(w)” option.

First, log to the “pa” account. Then, use the UPdate processor to update an item:

Enter: **logto pa**

Enter: **u entity (w)**

Try experimenting with data in the different attributes; and how you can “cruise” and scroll through different values.

Also, try modifying the entry screen by changing the attribute-defining items’ Input Pro with the “wn” procesing code.

See *The Advanced Pick Reference Guide* (formerly *Epick*) for more information on using processor. The documentation herein is mostly supplemental in nature.

PICK Troubleshooting

If you have any problems, try re-installing the product first. The problems you may encounter are:

When running a program "gui.import", user exit "00B2" is not valid" message appears. Have not executed the gui abs. Make sure you have the proper gui abs before running gui applications, usually "abs.gui.temp".

When running a program 'setup.grapick': 'MDS' is not a file name" message appears. Make sure you are in an account that does not have a reference (Q-pointer) to the System Dictionary (MDS). Make sure you have the proper file name.

When running a program you get an error: GUI caused a page fault @ 0016:009B" message appears. Make sure you have loaded EMM386 into high memory ("loadhigh emm386..."), which is not recommended for Workgroups. Make sure you have loaded EMM386 into conventional memory.

When running a program you get an error: Protection Fault in module BC30RTL.DLL @ 001:4F65" message appears. Make sure you are trying to start Pick improperly from an entry in the "AUTOEXEC.BAT" file and then restart Windows. Make sure you are starting Pick by placing the Pick icon in the Windows Startup folder.

When running a program you get an error: does not seem to be working. Editing an item in the Update processor merely brings up the error message and list of items. Make sure you have not executed the gui abs. Make sure you have the proper files and try again. Make sure you have the proper files, then the gui abs may be corrupted, and GraPICK may need to be reinstalled.

When running a program you get an error: seems to work fine when running the Pick/BASIC programs, but it does not work with the Update processor. Make sure you have not executed the gui abs. Make sure you have the proper gui abs before running GUI applications, usually "abs.gui.temp".

When running a program you get an error: get really stuck or lost, you can always close and restart the GraPICK program in Windows. Make sure you are using GraPICK as a terminal emulator, force your application back to TCL.

When running a program you get an error: In a worst case situation (or if the Windows environment is new to you), you may have to restart Windows and start over again. Remember, in such cases, that GraPICK is only the client and should be used as a "terminal" device. The server (host) process should still be running the process that you had when you had to restart GraPICK, so if it doesn't immediately recognize your keyboard you may have to exit the program running on the server using the sequences that the

program allows for; otherwise abort the program using a <Break> and “end” sequer process was enabled for such).

There are numerous ways to close a GraPICK screen:

- You may double-click the mouse on the top-left box of the current window (a Window for closing a Windows program) or
- use a button, such as “Exit” or
- use pull-down menus or
- use the standard application-supported escape sequences through the keyboard (i.e.,

Also, refer to *The Advanced Pick Reference Guide* (formerly *EPick*) for documentation of features and capabilities of the UPdate processor.

For any other problems with GraPICK, please contact:

Business Service
Center
10000
Ave. 92714
Dallas, Texas

Phone: [1] 714/261-1875
Fax:[1] 714/261-5308

Business Support - Europe
Business Centre
10000 CV35 7LS
London, England

Phone: [44] 1203-537027
Fax:[44] 1203-537045

Business Africa (Pty) Ltd.
10000 5277
10000 018
Johannesburg, South Africa

Phone: [27] 21-240656
Fax:[27] 21-247761

Business Ltd.
Profsojuznaja Str. 84/32
Russian Applied Mathematics Institute
Academy of Sciences
10000 SP-7 Russia
Moscow, Russia

Phone: [7] 095-33-36456
Fax:[7] 095-33-38000

Index

- \$**
- \$colors.....39
 - \$properties.....40, 42, 43, 45, 46, 47
- (**
- (w) Option.....31, 35, 75
- 3**
- 3-D Presentation.....60
- A**
- abs.....24
 - abs.gui.....24
 - access(21).....36
 - activate.menu().....42
 - add.menu.item().....42
 - AP Edit mode.....55, 57, 58, 60
- B**
- BMP.....52, 53, 57, 59
 - Button.....52, 53, 59
- C**
- change.object().....42
 - change.wdw.title().....42
 - CHBOX.....52, 53, 59
 - check.menu.item().....42
 - Checkbox.....52, 53
 - clear.data.obj().....43
 - clear.tb().....43
 - clear.wdw().....43
 - client.....9, 10, 11, 15,
.....21, 22, 25, 33, 36, 38, 45, 54, 55, 76
 - Closing a Zoomed-to Window.....32
 - Communications.....23
 - Communications settings.....23
 - create.menu().....43
 - create.object().....43, 52
 - create.submenu().....43
 - create.value().....44
- Cruising**.....
- D**
- DATA.....
 - data box.....21
 - data statement.....
 - Databox.....
 - DDE.....
 - del.menu().....
 - del.menu.item().....
 - del.value().....
 - del.wdw().....
 - demo.1.....
 - demo.1.standard.....
- E**
- Enabling GraPICK for the User.....
 - Entering or Editing Data.....
 - Event.....
 - Examples.....
- F**
- File Transfer Protocol.....
 - Focus.....
 - form.....21, 30, 34, 3
 - form connective.....
 - form file.....
 - form item.....33, 5
 - Form Title.....
 - form.fn().....
 - form.vseq().....
 - Format Conventions.....
 - form-defining item.....
 - FTP.....
- G**
- GBOX.....
 - get.color().....
 - Graphic.....
 - GraPICK
definition.....

Filesystem Changes 33

Pick/BASIC Changes 36

Troubleshooting 76

Windows Term Definitions 57

aware 9

item() 45

..... 52, 53

..... 58

..... 38

..... 38

..... 39, 51, 52, 53, 57, 58, 60, 61

..... 45

..... 46

menu.item() 46

nl Attributes 31

.....

and Exporting Data 38

em() 46

Procedure 11

the GraPICK Client 11

the GraPICK Server 15

n 21

..... 38

..... 10

mode 55, 57, 58, 60

..... 52, 66

..... 58

..... 52

Equipment Required 9

..... 21, 52, 58

erties 59

e 52

..... 58

OLE 60

opf 34, 38, 45, 55, 57

P

pd.forms 34

Pick Distribution Diskettes 9

POPUP 52, 53

process.menu.item() 46

Properties 60

R

RBUTTON 52, 53, 59

Reader's Comments 82

rearrange.menu() 46

rearrange.objects() 47

repl.menu.item() 47

rgb() 47

RS232 10

run.wdw.prg() 47

S

server 9, 11, 16, 22, 24, 76

set up GraPICK in other accounts 18

set.focus() 47

sgs 24

Starting GraPICK on the Client 22

STATEXT 52, 53, 59

sts.line() 48

Submenu 52, 53

SYSTEM 53

T

Tag 52, 53, 60

TBBUTTON 52, 53, 59

TCP/IP 10, 22

The GraPICK Client Environment - A
Programmer's Point of View 33

The GraPICK Client Environment - A User's
Point of View 25

toolbar button 30, 32, 66

U

UPdate Processor 31

W

w.dlg.box()	49
w.msg.box()	50
w{n}	33
WINDOW	53, 58, 59
WINDOW object	52
Windows-style Text Editing	30

X

Xmodem	
--------	--

Y

Ymodem	
--------	--

Z

Zmodem	
Zooming	