

# **Fitrix**

# **Visual Menus**

# Table of Contents

## **1: Overview of the Menu System**

- 1.1 Introduction
- 1.2 Menu Structure
- 1.3 Features
- 1.4 Installation
- 1.5 The Screen

## **2: The Menu System**

- 2.1 The Menu Command Line
- 2.2 Setup a Menu Structure
- 2.3 Menu Control Language
  - 2.3.1 Menu Control Language Instruction Set
    - 2.3.1.1 The `:ifxscreen:` Instruction
    - 2.3.1.2 The `:ifxreport:` Instruction
    - 2.3.1.3 The `:env:` Instruction
    - 2.3.1.4 The `:exit:` Instruction
    - 2.3.1.5 The `:if:` Instruction
    - 2.3.1.6 The `:show:` Instruction
    - 2.3.1.7 The `:pause:` Instruction
    - 2.3.1.8 The `:system:` Instruction
- 2.4 Version Control
  - 2.4.1 Version Control and `$DBPATH`

## **3: Security**

- 3.1 Overview
- 3.2 Maintaining Users and Groups
- 3.3 Assigning Security

## **4: Menu Configuration**

- 4.1 Overview
- 4.2 The Main Screen
- 4.3 The Sample Menu Screen Area
- 4.4 Color Configuration
- 4.5 Font Configuration
- 4.6 Logo Placement

# 1

## Overview of the Menu System

This section introduces Fitrix Menus and covers the following information:

- ❑ Introduction
- ❑ Menu Structure
- ❑ Features
- ❑ Installation
- ❑ The Screen

# 1.1 Introduction

The Fitrix Menus program provides the menu-level control necessary to create a complete, user-friendly application front-end. The menu system it creates is attractive, easy-to-create, easy-to-use, and completely portable. Menus created with Fitrix Menus can be used to access other menus, run programs or reports on the system, or execute UNIX operating system commands.

Other menu systems may allow you to create menus that execute programs from a menu item, but Fitrix Menus provides you with the ability to control the execution of a "multiple-step" menu item without batch or shell programming. For instance, one menu item may control an entire series of events: running a program or series of related programs, testing for proper execution of each program, and so on. Fitrix Menus is more flexible, more manageable, and more portable than a UNIX shell script.

At the simplest level, Fitrix Menus allows the system administrator to quickly modify a menu. Without having to deal with the UNIX shell or the shell language, the system administrator can provide a user with as much access to the system as is appropriate.

Fitrix Menus displays graphical menus on the screen. They can be created and maintained within the menus themselves by the system administrator.

The menu system allows the user to invoke a set of Menu Instructions from a single menu item. Every option on a menu has a corresponding set of Menu Instructions – these instructions form a Control Language.

## 1.2 Menu Structure

This section will briefly describe the structure of a Fitrix Menus system.

For Fitrix Menus to function properly, a specific table structure is required. This structure is hierarchical and uses the 'folder-file' metaphor familiar to Windows users. For the FG menu system the folder corresponds to a **menu** and provides a way of branching off to other levels of folders or files; the file corresponds set of Menu Instructions that are to be executed by the host. In this documentation the terms 'menu' and 'folder' are used interchangeably.

There are two main tables plus an additional table that concerns security. The main tables are 'cgsmnitm' which contains the text that appears next to each item or option, and 'cgsmncmd' which contains the Menu Instructions (Control Language) associated with the menu's option. The key for the 'cgsmnitm' table is the field '**opt**' (a one character option— a-w, 0-9) plus a '**mname**' field. The menu item's option plus name form the main key. Generally, it would be a good idea to use names that are meaningful and hierarchical in nature. There is an additional field which marks each menu selection as either another folder or a terminal option (field '**mtype**'). A menu structure for an accounting system might have options, names, and mtypes like:

<u>Op</u>	<u>Name</u>	<u>Mtype</u>	
a	armenu.customer	FL	{ 'FL' = folder (menu) }
b	armenu.customer	SC	{ 'SC' = Screen maintenance }
c	armenu.customer	RP	{ 'RP' = Report }

## 1.3 Features

Fitrix Menus instructions (refer to "The Fitrix Menus Instruction Set") are stored in table 'cgsmncmd' and can perform any of the following specific functions:

- ☐ Call up other menus.
- ☐ Create a list of explanatory text that can be used by later instructions to show the user comments about the actions that are to be taken.
- ☐ Pause and show the user explanatory text and control the method of report output.
- ☐ Call up a report program
- ☐ Call up a file maintenance program.
- ☐ Test for the successful completion of any item instruction (particularly those that execute programs or UNIX commands).
- ☐ Change or set an environment variable.

# 1.4 Installation

## 1.4.1 Visual Menus Only (VDT already installed)

### 1.4.1.1 Unix Host Installation

- ❑ Login as root
- ❑ Update Unix host system files:

Modify the Unix networking file ‘/etc/services’ and provide a port for the system.

```
$ vi /etc/services
fgmn 50000/tcp    # FG Menu
```

- ❑ Make sure a valid Fitrix VDT environment is set.

Environment variables ‘\$fg’ and ‘\$INFORMIXDIR’ should be set, and ‘\$fg/bin’ and ‘\$INFORMIXDIR/bin’ should be on ‘\$PATH’.

- ❑ Set environment variable “APPSERVER” to the name or IP address of the applications servers (machine where the menus are installed).
- ❑ Mount the install CD:  
For example:  

```
$ mount -o ro,lower /dev/cd0 /mnt/cdrom
```



- ❑ Run the install script:  
\$ cd /mnt/cdrom/unix  
\$ ./mn\_install.sh

The old '\$fg/bin/mz' program is renamed: '\$fg/bin/mzt', and 'mz' becomes a script that will run either the old menus or the new Visual Menus.

Enter the appropriate licensing information when asked:

Enter license number: 9999999  
Enter number of users: 9999  
Enter activation code: xxxxxxxx

- ❑ Populate the menu tables:

For example:

```
$ cd $mz {Usually something like $fg/accounting/menu}  
$ $fg/bin/menucvt.sh mainmenu
```

Where 'mainmenu' is the top of the menu structure.

## 1.4.2 Visual Menus As Included with the VDT product

See VDT Installation documentation.

### **1.4.1.2 PC Workstation Host Installation**

- ❑ Insert the CD into the drive, change to the appropriate drive and go to the appropriate directory:

For example:

```
C:> F:
```

```
F:> cd workstation
```

- ❑ Run the install program.

```
F:> install.exe
```

When asked, provide the directory path of the original Informix WTK client installation and click 'OK'.

WTK install directory path:

## 1.4.8 Unix Environment Variables:

In addition to the normal Fitrix environment settings—the following variable need special attention:

- ❑ **APPSERVER** – IP address or name of the Application server.
- ❑ **FGLSERVER** - The name of IP address where the D4GL display server is running.

If a WTK workstation client running ‘wtk.exe’:

# FGLSERVER=myworkstation      ;export FGLSERVER

If an X-Windows Unix host running ‘fglX11d’ daemon:

# FGLSERVER=bigmachine ;export FGLSERVER

- ❑ **remoteshell** – Name of the remote shell used by the platform:

SCO = rcmd

Linux = rsh

For other platforms—see your system administrator

- ❑ **ifxproject** – root directory for the application – for FG Accounting this would be: \$fg/accounting
- ❑ **company** – Company/database to run against

**Fitrix Software**  
*Visual Menus User Reference*

- ❑ **cust\_key** – Single three character directory extension used by the menus
- ❑ **cust\_path** - Series of three character directory extensions used by programs
- ❑ **printerlist** - Points to a file containing a list of Unix host printers
- ❑ **LPDEST** - Default printer
- ❑ **pcopies** - Number of print copies
- ❑ **lpflag** – Additional flags for host print command. Usually flag that prefaces number of copies—**BE CAREFUL** of spaces between this switch and the actual number of copies. (SCO: -n ; LINUX: -#).
- ❑ **filedir** – Where report files are temporarily kept
- ❑ **mn\_buttonbar** – Set to ‘0’ if button bar should not appear, set to ‘1’ if should appear, set to ‘2’ if shell button also appears (default is ‘1’).
- ❑ **mn\_menubar** – Set to ‘0’ if menu bar should not appear, set to ‘1’ if should appear, set to ‘2’ if only File-Exit button appears (default is ‘1’).
- ❑ **mn\_nolocalprinter** – Set to anything if local printer suppressed (default is unset)—see switch -l.
- ❑ **mz\_nomail** – Set to anything if mail suppressed (default is unset) — see switch -m.
- ❑ **MAILER** – Mail program.

**Fitrix Software**  
*Visual Menus User Reference*

- ❑ **mn\_nofax** – Set to anything if fax suppressed (default is unset) —see switch -f.
- ❑ **faxnum** – Fax program.
- ❑ **mn\_maintitle** – Set to menu title bar text (default “Mn --” , maximum 80 characters)
- ❑ **mn\_reportprinter** – Set to ‘1’ if close printer dialog after print to printer. (default “0”)
- ❑ **mn\_installdir** – Directory on workstation where client software and resources were installed. (default “.”, i.e. same as installation directory for Informix ‘WTK’)

## 1.4.9 Troubleshooting:

- ❑ There are three log files created to help troubleshoot installation issues.

1) On the Unix host: '\$HOME/mnc.log' and '\$HOME/mnp.log'. The execution directory and program execution line is written into 'mnc.log'. The critical lines will look like:

```
P LOG: Dir: </fourgen/accounting/ar.4gm/i_custr.4gs>
P LOG: Fork: </fourgen/bin/fglrun>
</fourgen/bin/fglrun>
<i_custr.42r>
<-c>
<sample>
```

2) On the workstation: <WTK installation directory>/fgss\_bin/mn.log

- ❑ Menu conversion

- 1) Look at the file 'menucvr.err' for errors logged during menu conversion.
- 2) When loading menus from flat files only 'one character' keys are allowed for a menu item.
- 3) Obviously no duplicates are allowed within a 'menu' file.
- 4) The delimiter for loading from flat files into a database is normally '|'. Be careful not to use this character anywhere within a flat file or use the environment variable 'DBDELIMITER' to change the delimiter character.

□ Printing problems on a Unix host:

Let's say your entry in \$printerlist was:

:lexmark1:lpr -Plexmark1:

and your environment variables were set as follows:

lpflag="-n"

ncopies="2"

and the file to be printed was "report.out".

The program concatenates the various pieces together like this:

<printerlist\_command><space><lpflag><ncopies><space><path\_to\_file>

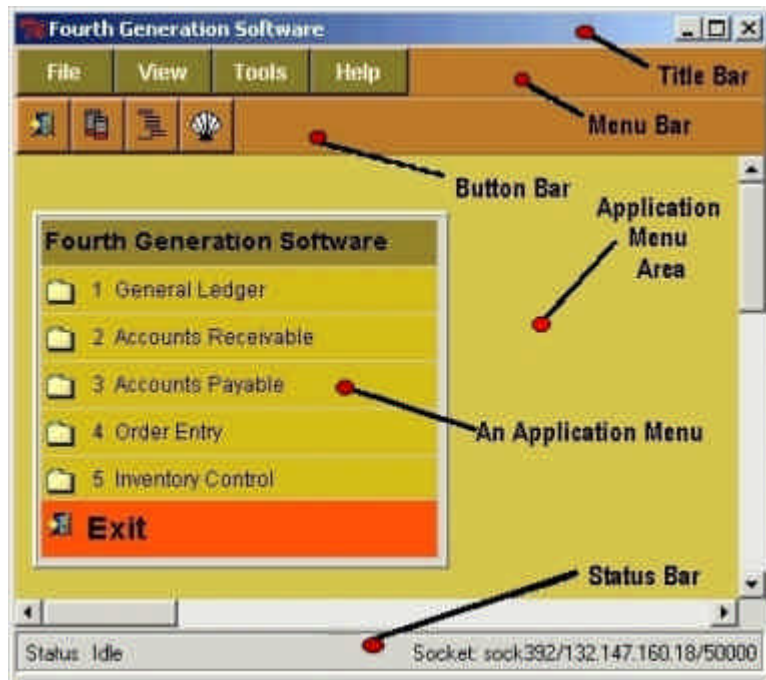
so that the command would be:

lpr -Plexmark1 -n2 report.out

## 1.5 The Screen

### 1.5.1 Classic View

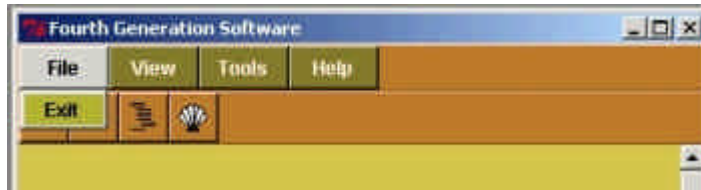
The Menu screen shown below is composed of: 1) Title bar, 2) Menu bar, 3) Button bar, 4) Application Menu Area, 4a) An Application Menu, 5) Status bar.





**Fitrix Software**  
*Visual Menus User Reference*

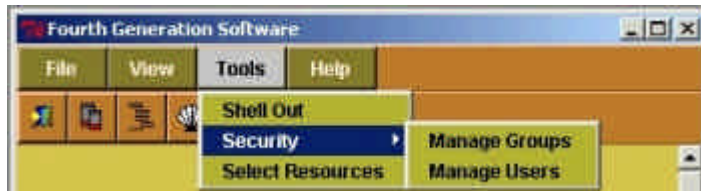
❑ **Menu Bar: File**



❑ **Menu Bar: View**



❑ **Menu Bar: Tools**



❑ **Menu Bar: Help**

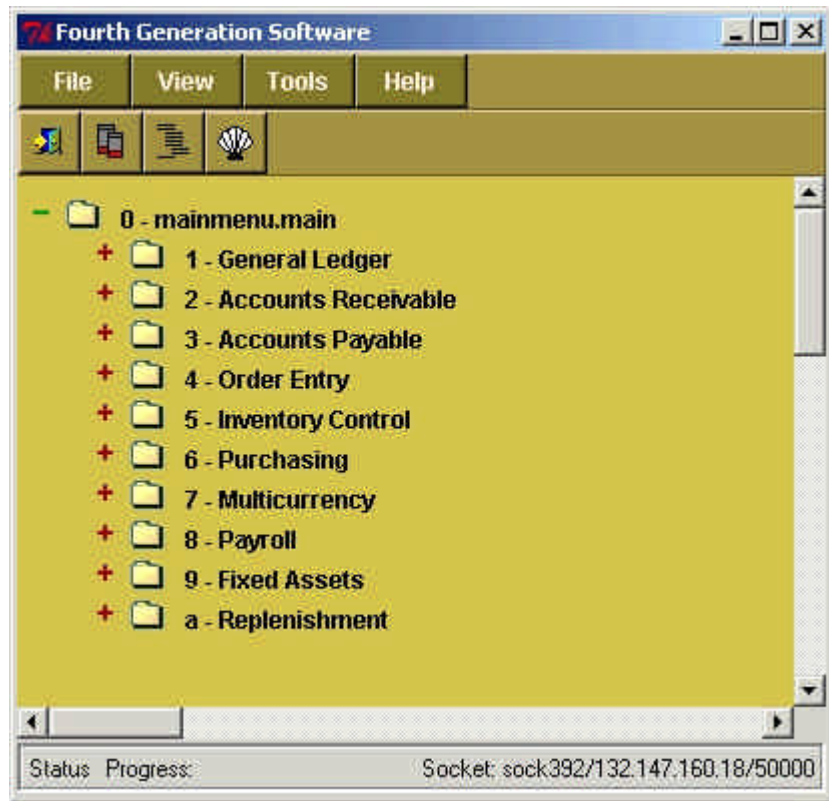


□ **Button Bar:**



## 1.5.2 Explore View

Now click on the  button:



## Sample Menu Structure

For the example below, the table 'cgsmnitm' can be viewed as a header table and table 'cgsmncmd' is detail, joined by the fields 'opt' and 'mname'. Fields 'mtype' and 'txt' are header information while 'cmd' is detail (indented).

The following Fitrix Menus application structure is provided as an example:

<b>mname</b>	<b>opt</b>	<b>mtype</b>	<b>txt</b>	<b>cmd</b>
mainmenu.main	1	FL	"General Ledger"	:menu:glmenu.main:
<b>mainmenu.main</b>	<b>2</b>	<b>FL</b>	<b>"Accounts Receivable"</b>	<b>:menu:armenu.main:</b>
mainmenu.main	3	FL	"Accounts Payable"	:menu:apmenu.main:
mainmenu.main	4	FL	"Order Entry"	:menu:oemenu.main:
mainmenu.main	5	FL	"Inventory Control"	:menu:icmenu.main:
armenu.main	1	FL	"Receivable Ledger"	:menu:armenu.ledger:
<b>armenu.main</b>	<b>2</b>	<b>FL</b>	<b>"Customer Information"</b>	<b>:menu:armenu.customer:</b>
armenu.main	3	FL	"Setup Receivables"	:menu:armenu.arsetup:
armenu.customer	a	SC	"Customer Information"	:ifxscreen:ar:i_custr:::
armenu.customer	b	FL	"Print Customer Info"	:menu:armenu.custinfo:
armenu.customer	c	FL	"Print Customer Labels"	:menu:armenu.label:
armenu.customer	d	SC	"Update Customer Terms"	:ifxscreen:ar:i_termr:::
armenu.customer	e	RP	"Print Customer Terms"	:show:Print Customer Terms: :show:List of available customer terms: :pause:p: :ifxreport:ar:o_termrls::default:::
armenu.customer	f	SC	"Update Cust Ship-To's"	:ifxscreen:ar:i_shipr:::

**Fitrix Software**  
*Visual Menus User Reference*

The menu 'mainmenu.main' is seen below:



The folder icons on the left mean that each item is a menu (folder) that will link to another menu. Clicking on 'Accounts Receivable' will show the menu below:



Clicking on 'Customer Information' will show the menu below:



# 2

## The Menu System

- ❑ The Menu Command Line
- ❑ Setup a Menu Structure
- ❑ Menu Control Language
- ❑ Menu Control Language Instruction Set
  - ❑ The **:ifxscreen:** Instruction
  - ❑ The **:ifxreport:** Instruction
  - ❑ The **:env:** Instruction
  - ❑ The **:exit:** Instruction
  - ❑ The **:if:** Instruction
  - ❑ The **:show:** Instruction
  - ❑ The **:pause:** Instruction
  - ❑ The **:system:** Instruction
- ❑ Version Control
- ❑ Version Control and \$DBPATH

## 2.1 The menu command line

To use a Fitrix Menus menu, you must run the Fitrix Menus runtime program. The command to execute from the UNIX operating system prompt is:

**mn [-v] [-h] [-m] [-f] [-l] [-r] [-x option.menu\_name] [-d directory]**

<b>-v</b>	Print version information and exit.
<b>-h</b>	Print help on command line options and exit.
<b>-m</b>	Suppress mail option for printer dialog output. Can also use environment variable: 'mz_nomail'.
<b>-f</b>	Suppress fax option for printer dialog output. Can also use environment variable: 'mn_nofax'.
<b>-l</b>	Suppress local printer option for printer dialog output. Can also use environment variable: 'mn_nolocalprinter'.
<b>-r</b>	This flag prevents the user from accessing the UNIX shell or any shell command.
<b>-x option.name</b>	The -x option.menu_name This option allows you to select a specific option and menu combination to run with showing the main screen.
<b>-d directory</b>	By default, the Fitrix Menus automatically uses the key: opt='0', mname='topmenu.main' to lookup the command to use as a starting folder. The -d flag is used to override this default setting. For example: -c armenu_main

## 2.2 Setup a Menu Structure

Setting up a menu structure begins by initializing the appropriate tables for a particular company. The data can then either: 1) be manually entered into the tables via the Menu system (by the system administrator), or 2) loaded from a previous version of the 'mz' system.

### Enter data manually

First initialize the tables:

```
$ company=standard ;export company  
$ menuini.sh
```

Select a top menu name: **mainmenu**

**The menu name must use lowercase alphanumeric characters  
plus '\_' and '.'.**

The tables 'cgsmnitm', 'cgsmncmd', and 'cgsmnsec' are created. Now run the 'mn' and you will see:





**Fitrix Software**  
*Visual Menus User Reference*

To edit the menu structure which is now empty, you must switch to 'Explorer View'.

Click on the  icon and you will see:

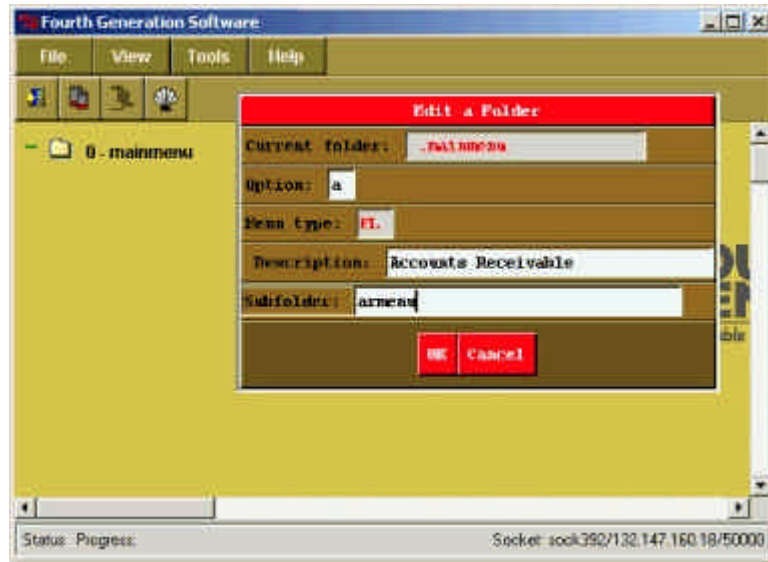


This odd looking folder is a dummy place holder for the top or root menu. For ordinary users (not 'root' or not members of group 'root') this folder will be hidden. Right click on the word 'mainmenu' (when you created the empty menu structure this is the name that was selected for the top or root menu) and select 'New Folder'.

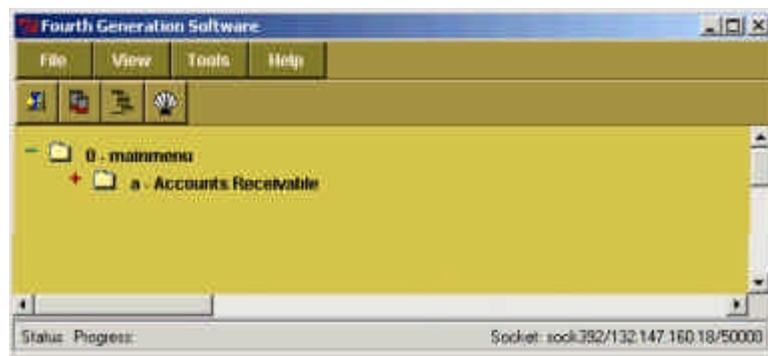


### Visual Menus User Reference

You will see a ‘Edit a Folder Dialog Box’:



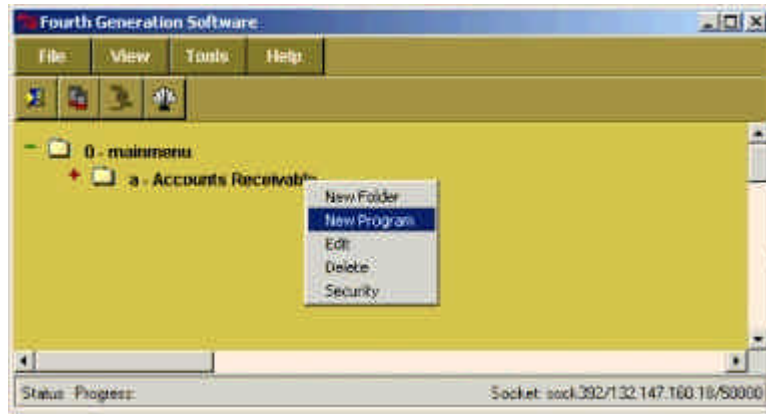
Click ‘OK’ when you have filled in ‘Option:’, ‘Description:’, and ‘Subfolder’ and you will see:



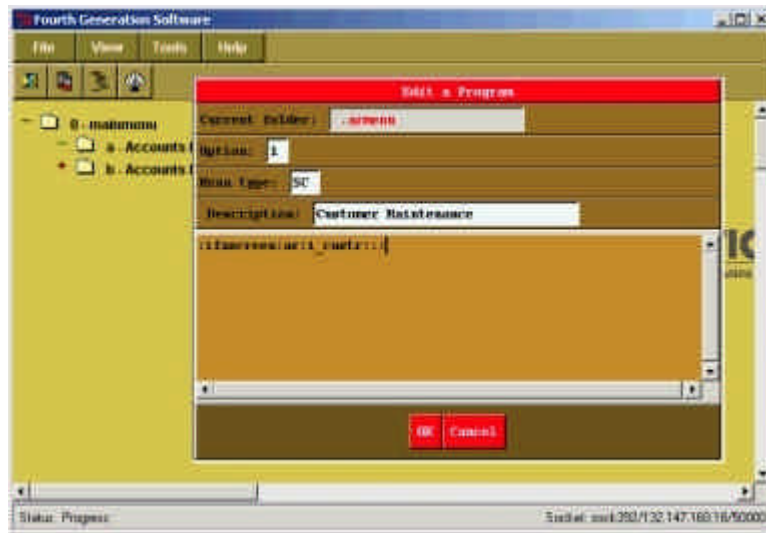
## Fitrix Software

### *Visual Menus User Reference*

Now after adding another folder under 'mainmenu' called 'Accounts Payable', right click on the 'Accounts Receivable' folder and select 'New Program'.

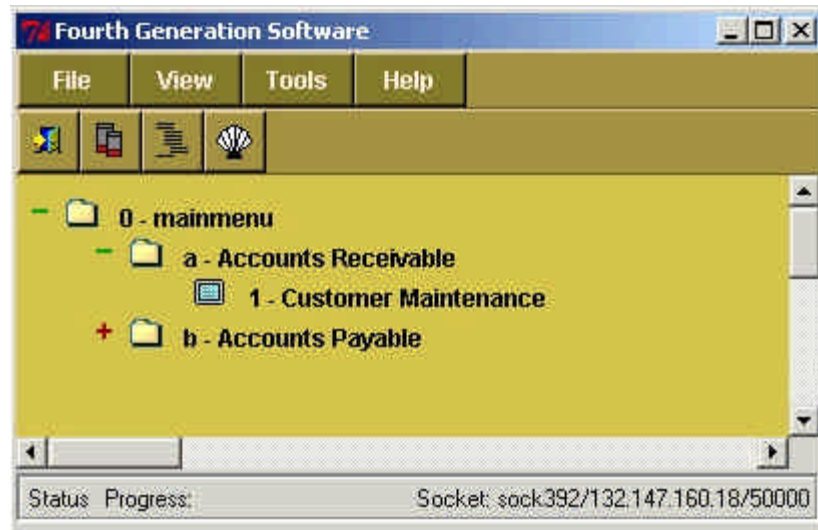


You will see the the 'Edit a Program Dialog Box':



**Fitrix Software**  
*Visual Menus User Reference*

After filling in 'Option', 'Menu Type', 'Description', and the instructions, you will now see the menu modified as follows:



As in 'Microsoft Windows Explorer', the '+' sign in front of the folder icon provides a way of opening and exposing the items within the folder and the '-' sign will reverse this process and collapse the folder's contents. The icon in front of 'Customer Maintenance' is a 'screen' and derives from the 'Menu type' field in the 'Edit a Program' dialog box. You can also use 'RP' for 'report' or 'OT' for 'other'.

The text scrollable area in the 'Edit a Program' dialog box that contains the text:

**:ifxscreen:ar:i\_custr:::**

needs a good deal more explanation. This is the "Control Language" that determines what happens when a user clicks an item.

## 2.3 Menu Control Language

Each line of Control Language contains an instruction always beginning and ending with a colon ( : ). Following this initial colon is the instruction name, followed by a colon, followed by the instruction parameters, which are separated from each other with colons. For example:

```
:ifxscreen:ar:i_custr:::
```

A Fitrix Menus set of item instructions may consist of any number of instructions and comments

Both the instruction and its parameters must be written in lowercase (except for parameters consisting of text that is to be displayed, which may appear in upper or lowercase). Some of these instruction parameters are required, others are optional. After selecting a menu item, processing begins; if an instruction is missing a required parameter, the system reports a format error.

### 2.3.1 Menu Control Language Instruction Set

Fitrix Menus has two special instructions designed to assist developers working in an INFORMIX-4GL environment. These instructions are the root of most sets of item instruction. They allow you to run a specific INFORMIX-4GL report or a program.

The **:ifxreport:** instruction simplifies the entire process of selecting reports to be run, passing arguments to those reports, and routing the output to a specific destination.

The **:ifxscreen:** instruction searches a directory structure for a specified program, passes arguments to that program, and runs the program.

## 2.3.1.1 The **:ifxscreen:** Instruction

The **:ifxscreen:** instruction searches a directory structure for a specified INFORMIX-4GL program, passes arguments to that program, and then runs the program.

The format:

```
:ifxscreen:module:program:[flags]:[x]:
```

An example:

```
:ifxscreen:ar:i_invce:::
```

Explanation:

- ❑ **module:** An environment variable, \$ifxproject, points to a directory in which module directories reside. The name of the module directory is the value given in the "module" field with a ".4gm" extent. In the example, "ar.4gm" is a module (directory) containing accounts receivable directories.
- ❑ **program:** Two directories may exist below the module directory. The names of these two directories are the value given in the "program" parameter with a ".4gc" and ".4gs" extension. Both directories may exist. Fitrix Menus searches the .4gc directory first for the program to be run.

Either the .4gc or .4gs directory can contain an executable program. The name of this program is the value given in the "program" parameter with a ".42r" extension. The program is executed with a 'runner' or 'interpreter' for INFORMIX-4GL D4GL programs.

- ❑ **flags:** Any "flags" set are sent to the program being called.
- ❑ **x:** When used, if an abnormal exit occurs during processing, Fitrix Menus returns the user to the menu, rather than proceeding to any remaining instructions within the set of item instructions.

## 2.3.1.2 The **:ifxreport:** Instruction

Menu options that print reports utilize the **:ifxreport:** instruction to find and print reports. The **:ifxreport:** instruction searches a directory structure for a specified INFORMIX-4GL program, passes arguments to that program, runs the program, and routes the output to a designated destination.

The format:

```
:ifxreport:module:program:[flags]:[destination]:[x]:
```

An example:

```
:ifxreport:ar:p_invpst:-p:default:x:
```

Explanation:

- ❑ **module:** An environment variable, \$ifxproject, points to a directory in which module directories reside. The name of the module directory is the value given in the "module" field with a ".4gm" extent. In the example, "ar.4gm" is a module (directory) containing accounts receivable directories.
- ❑ **program:** Two directories may exist below the module directory. The names of these two directories are the value given in the "program" parameter with a ".4gc" and ".4gs" extension. Both directories may exist. Fitrix Menus searches the .4gc directory first for the program to be run.

Either the .4gc or .4gs directory can contain an executable program. The name of this program is the value given in the "program" parameter with a ".42r" extension. The program is executed with a 'runner' or 'interpreter' for INFORMIX-4GL D4GL programs.

- ❑ **flags:** Any "flags" set are sent to the program being called.
- ❑ **destination:** The "destination" field may be set to "screen," or "default." The "default" destination may be set with a prior **:pause:p:** instruction; otherwise, the default printer is used.

**Fitrix Software**  
*Visual Menus User Reference*

Fitrix Menus passes the arguments "destin /tmp/ifaxunique" to an Informix report program. In order for you to print or redirect properly from Fitrix Menus, your program must parse for these command line arguments. This can easily be accomplished utilizing a call to the `get_arg()` function in the `$fg/standard.4gs` library, which is included with . If you do not have this library we have listed the function below for your convenience.

The following code shows an example of how you might set your program up to output its report to the filename where Fitrix Menus expects to find output.

First, your program must read the filename from the command line, the `-c $company` variable, and then output to that filename.

The following function call puts the filename into a variable called `destin_variable`. The function `get_arg()` previously placed the filename in the scratch variable.

```
if get_arg("destin") then
    let destin_variable = scratch
end if
```

The following function outputs your report to the `destin_variable`:

```
#####
#####
function start_rpt()
#####
#####
#
start report some_report to destin_variable
    return
end function
# ct_start_rpt()
```



**Fitrix Software**  
*Visual Menus User Reference*

- ❑ **x:** When used, if an abnormal exit occurs during processing, Fitrix Menus returns the user to the menu, rather than proceeding to any remaining instructions within the set of item instructions.

The `:ifxreport:` instruction prints any partial report that was generated even if it exits with a non-zero exit status.

### 2.3.1.3 The **:env:** Instruction

The **:env:** instruction is used to set variable values or to remove variables from the environment.

Format:

**:env:variable:[value]:**

Examples:

**:env:myname:root:**

**:env:pst\_status:\$mz\_status:**

**:env:pst\_status::**

In the first example, the variable myname is set to "root". Any subsequent reference to \$myname uses this value. In the second example, the variable pst\_status is being set to the current value of 'mz\_status'. In the third example, the variable pst\_status is being removed from the :environment. A variable set with the **:env:** instruction is available to all subsequent programs called by Menus.

## 2.3.1.4 The **:exit:** Instruction

The `:exit:` instruction halts the processing of the Item Instruction Set and returns the user to the menu. It is usually used in conjunction with `:if:` statements to cut off processing if certain conditions exist.

Format:

**:exit:**

Examples:

```
:ifxreport:gl_post::default::  
:if:test "$mz_status" = "1":exit:
```

## 2.3.1.5 The **:if:** Instruction

The **:if:** instruction conditionally executes any other Fitrix Menus instruction if the UNIX command returns a "true" (0) status. Any return status other than 0 skips the **:if:** instruction parameter.

Format:

**:if:UNIX\_command:other\_FourGen\_Menus\_instruction:**

Example:

**:if:test "\$mz\_status" = 1:post:gl.chart:totals:a:Posting Totals:**

## 2.3.1.6 The :show: Instruction

This instruction is used to announce to the user what menu item has been selected and what it does. The :show: instruction also allows the user a chance to verify his menu selection. It is especially important to include item instructions prior to instructions that print or post data.

Format:

**:show:text:**

:Example:

**:show:Print Customer Terms:**

**:show:Sorted by term code:**

**:pause:p:**

**:ifxreport:ar:o\_terminals::default::**

**Select a Printer**

Print Customer Terms  
Sorted by Term Code

Number of Copies: 1

☐ screen

☒ Host Printer lexmark1

☐ File Name:

Last process: 0 - Unknown

**OK Cancel**

### 2.3.1.7 The **:pause:** Instruction

The **:pause:** instruction is used to pause during processing until the user clicks [OK] or [Cancel] to continue, and to allow the user to define or change the destination of any program output.

Format:

**:pause: [x|p] :**

If the x flag is used, the user is given the option of exiting the current set of Instructions.

If the p flag is used, the user is able to exit, redirect output, or change the number of copies to print.

The **:pause:** instruction is commonly used after **:show:** instructions to give the user the opportunity to confirm his selection of the menu item.

Example:

**:show:Print Customer Terms:**  
**:show:Sorted by term code:**  
**:pause:p:**  
**:ifxreport:ar:o\_terminals::default::**

Explanation:

The use of the optional x flag following the **:pause:** instruction gives the user the opportunity to quit processing and return to the menu.

If the p option were used, the user would also have the option to change the default printer setting and the number of copies to print.

**Fitrix Software**  
*Visual Menus User Reference*

Below is the ‘Select a Printer’ Dialog box that the ‘pause:p’ instruction produces

**Select a Printer**

Print Customer Terms  
Sorted by Term Code

Number of Copies: 1

☐ Screen

☒ Host Printer: lexmark1

☐ Client Printer: **Select Printer** **Printer Setup**

☐ Mail To: Subject:

☐ Fax Number:

☐ File Name:

Last process: 0 - Unknown

**OK** **Cancel**

At the top are the ‘:show:’ instructions that just precede the ‘:pause:p:’ instruction.

Next the user can either key in the number of copies desired or can use the selector arrows to set the number copies. A series of radio buttons on the left control the selection of the output 'media'.

☐ screen

The option presents the report to the screen via a screen pager (See 'report\_page' environment variable).

☐ Host Printer

Key in or select from a drop down box of available printers (See environment variables: 'LPDEST', 'printerlist', 'filedir').

☐ Client Printer

The buttons 'Select Printer' and 'Printer Setup' execute standard Windows Print Dialog Boxes. (See environment variable: mn\_nolocalprinter) (Note the command line switch -l)

☐ Mail

Enter the name of the recipient and the subject line of the message. (See environment variables: mz\_nomail, MAILER) (Note the command line switch -m)

☐ Fax

Enter the fax number. (See environment variables: mn\_nofax, faxnum) (Note the command line switch -f)

☐ File



Enter a valid file name (no path). (See environment variables: `filedir`)

### **2.3.1.8    The `:system:` Instruction**

The `:system:` instruction is used to execute a one-line operating system command, a program, or a shell script. If the command, program, or script to be executed cannot be found on the user's regular command path, then it must include the entire path name, just as if it were run outside of Fitrix Menus. If the command, program, script, or batch file fails to execute correctly, you can optionally return the user to the menu without continuing with the rest of the instructions in the set of Item Instructions. You can use the `:system:` instruction to change your current working directory while in a menu.

Format:

`:system:system_command_1:[system_command_2:]...`

Example:

`:system:ls -la:`

## 2.4 Version Control

Version control allows you to run a particular version of a program. It allows you to take advantage of Version Control, which is a feature in Fitrix Screen that allows programmers to have different versions of a program within the same directory structure without multiple copies of files.

Say you have several customers all using the same program, but ABC customer and XYZ customer both needed slight variations. With multi-version control, Menus allows you to run any one of those versions by setting a new environment variable called \$cust\_key that corresponds to the extension on the directory and files for the particular version, e.g., .abc, .xyz, .4gc, or .4gs. What Menus will do is run the program version connected with that particular \$cust\_key. If a particular version of the program doesn't exist for the \$cust\_key, it will search the program directories for the program to run in a hierarchical fashion starting with customer-specific (.abc or .xyz), to custom (.4gc), and then to base (.4gs). In the example below, we have a general ledger (gl.4gm) application with its different program directories. There are four different versions of the application that may be executed:

```
gl.4gm -----|-- i_chart.abc
                |-- i_chart.4gc
                |-- i_chart.4gs
                |
                |-- o_income.4gc
                |-- o_income.4gs
                |
                |-- p_genled.xyz
                |-- p_genled.4gs
                |
                |-- i_genjrn.4gs
```

**Fitrix Software**  
*Visual Menus User Reference*

Depending on what the \$cust\_key is set to, Menus will execute the different versions of the program as shown below.

cust_key=4gs	i_chart.4gs o_income.4gs p_genled.4gs i_genjrn.4gs	
cust_key=4gc	i_chart.4gc o_income.4gc p_genled.4gs i_genjrn.4gs	- since there is no p_genled.4gc - since there is no i_genjrn.4gc
cust_key=xyz	i_chart.4gc o_income.4gc p_genled.xyz i_genjrn.4gs	- since there is no i_genjrn.xyz - since there is no i_genjrn.xyz - since there is no i_genjrn.xyz or i_genjrn.4gc
cust_key=abc	i_chart.abc o_income.4gc p_genled.4gs i_genjrn.4gs	- since there is no o_income.abc - since there is no p_genled.abc or p_genled.4gc - since there is no i_genjrn.abc or i_genjrn.4gc

So at any time, you can set the \$cust\_key variable, run Menus, and see what the product looks lil for any specific customer (abc or xyz), or see your value-added product (4gc).

The sequence of directories in which an application looks for a specified program is as follows:

1. If there is a \$cust\_key variable and a program exists in the directory with the customer key as the extension, then it runs that program.
2. If there is no directory with the \$cust\_key extension, it looks for a custom directory with the .4gc extension and runs the program in that directory if it exists.
3. If it still has not found a program to execute, it then goes into the 4GL source directory with the .4gs extension and runs the program in that directory.

This sequence allows you to maintain the standard software and any number of specific modifications in the same directory structure.

## **2.4.1      Version Control and \$DBPATH**

When using version control, directories listed in your \$cust\_path are automatically prepended to your \$DBPATH. This allows the program to locate .42f files that may not appear in your local directory.

If your \$cust\_path is "4gc:4gs" and the directory that the program is running in is the .4gc (custom) directory, then the \$DBPATH variable is prepended with "../{prog\_name}.4gs:".

If the directory that the program is running in is the \$cust\_key (customer specific) directory, then the \$DBPATH variable is prepended with "../{prog\_name}.4gc:.../{prog\_name}.4gs:" where {prog\_name} is the name of the program that is being run.

This allows you to keep form (.42f) files that are the same throughout different versions in their original location and avoid copying these files. At runtime, the form file will be found even if it isn't in the custom directory because the \$DBPATH variable also acts as a form path for INFORMIX-4GL.

# 3

## Security

- ❑ **3.1 Overview**
- ❑ **3.2 Maintaining Users and Groups**
- ❑ **3.3 Assigning Security**

## 3.1 Overview

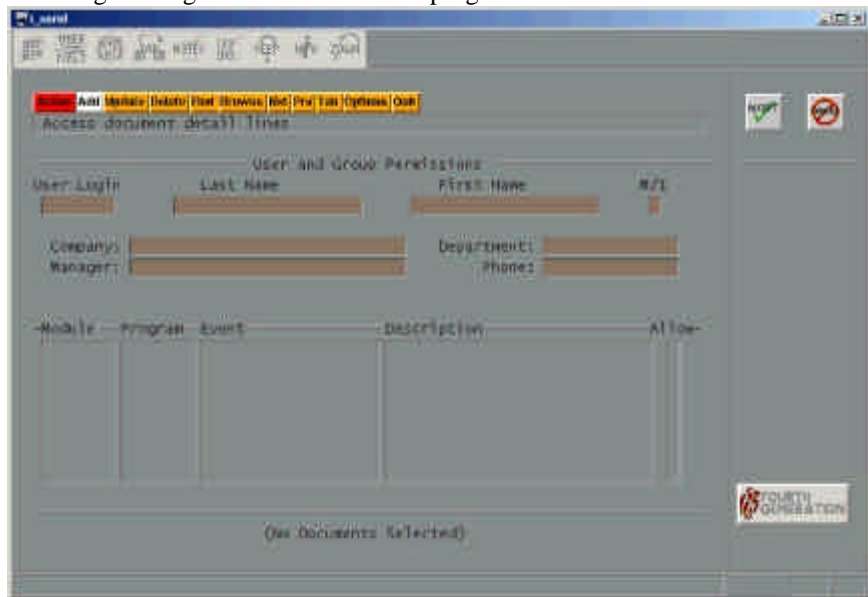
Security for the menu system is based upon two tables, one that contain users (stxsecur) and one that contains groups (stxgropr). There is one additional table that controls which user/group is allowed or disallowed to execute a menu item. Maintenance of users and groups is accomplished thru D4GL programs launched from the Menu Bar while maintenance of security for a particular menu item is controlled by right clicking on the item from the Explorer View.

## 3.2 Maintaining Users and Groups

The Main Menu Screen has a drop-down menu that provides access to the main to file maintenance programs.



Selecting 'Manage Users' launches the program below:



Fill in the user-login and name fields.

**Fitrix Software**  
*Visual Menus User Reference*

Selecting 'Manage Groups' launches the program below:

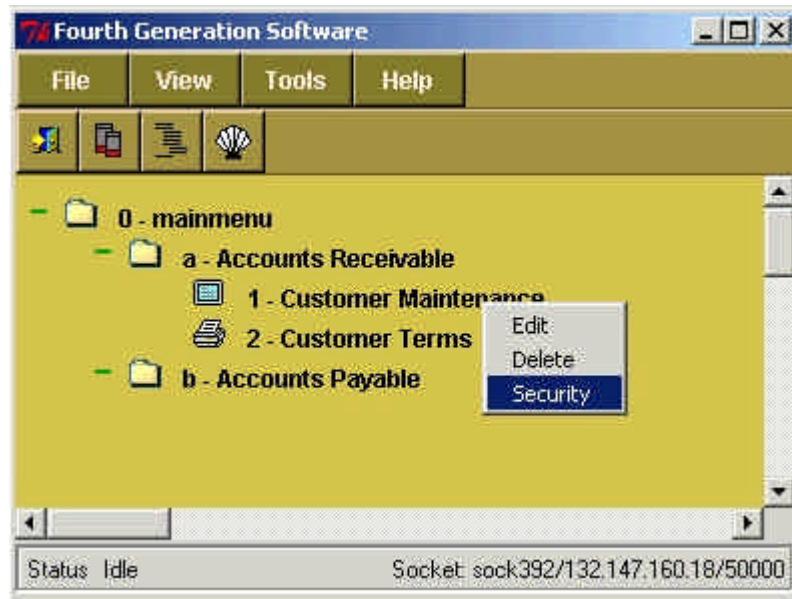
The screenshot shows a window titled "i\_group" with a menu bar containing "Add", "Modify", "Delete", "Find", "Show", "Hide", "Print", "Options", and "Quit". Below the menu bar is a toolbar with icons for these functions. The main area is titled "Security Groups" and contains a "Create a new document" button. Below this are two input fields: "Group Code:" and "Description:". At the bottom, there are five columns labeled "user Login", "root Login", "user Login", "root Login", and "user Login", each containing a list of empty rows for data entry. A status bar at the bottom indicates "(0 documents selected)". On the right side, there are two small icons: a green checkmark and a red X, and a "SECURITY" logo at the bottom right.

Fill in the group code and Description fields.



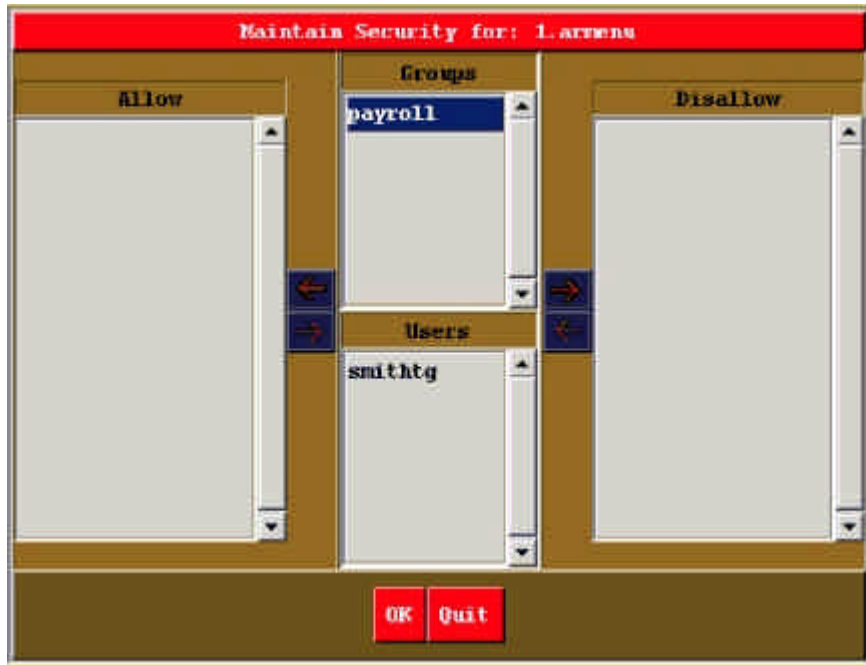
## 3.3 Assigning Security

To assign security privileges, switch to 'Explorer View' and right click on any item (folder or otherwise)-- select 'Security' from the pop-up menu.



**Fitrix Software**  
*Visual Menus User Reference*

The 'Maintain Security' Dialog Box will appear. The user has clicked on the group 'Payroll' in preparation for assigning it.



In the center you will see a scrollable list of groups (top) and a scrollable list of users (bottom). The group 'payroll' is highlighted after being clicked and the top arrows have been highlighted and the lower arrows have been dimmed since the only possibilities are to move the group either to the 'Allow' (left) or 'Disallow' (right) column.

**Fitrix Software**  
*Visual Menus User Reference*

Assume the user clicks on the upper left arrow (move to 'Allow'). This will move the group to the 'Allow' list. Now assume that the user has clicked on 'payroll \*' again (the asterisk is an indicator that the name was a group):



The user may now return the group to it's original position (out of the 'Allow' list and back to the 'Groups' list).

# 4

## Menu Configuration

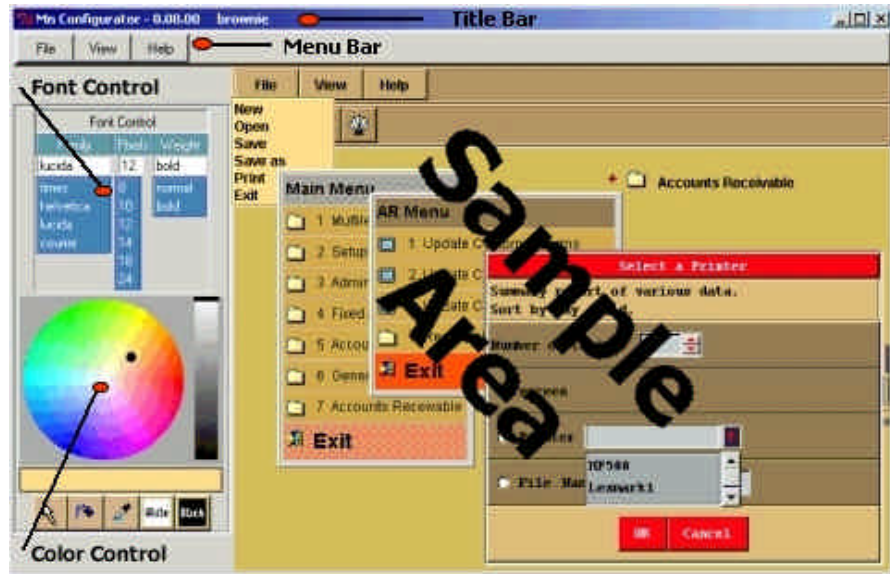
- ❑ **Overview**
- ❑ **The Main Screen**
- ❑ **The Sample Menu Screen Area**
- ❑ **Color Configuration**
- ❑ **Font Configuration**
- ❑ **Logo Placement**

## 4.1 Overview

Each workstation PC can be individually configured in terms of colors and fonts by maintaining a resource file called: **<WTK>/fgss\_bin/mn.res**. This file is the file opened by the 'mn.exe' client menu program. There may, however, be a number of resource files (all ending with the '.res' extension) on the workstation—for example: 'brown.res', 'plum.res', 'sepia.res'. From the menu system, any of these may be selected and will be copied to 'mn.res'.

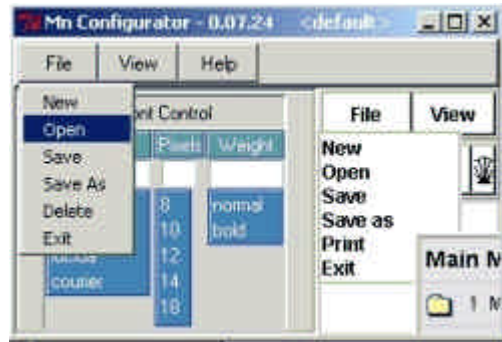
## 4.2 The Main Screen

Upon executing the configuration program: 'mnconfig.exe', the program's screen will appear:

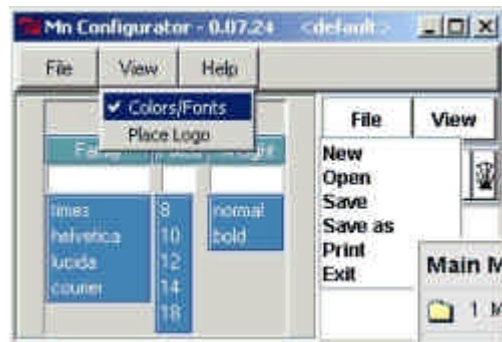


At the top is the Title Bar, and underneath is the Menu Bar (File-View-Help). On the right side of the screen are the controls for Font and Color. At the right is a sample View of the Menus.

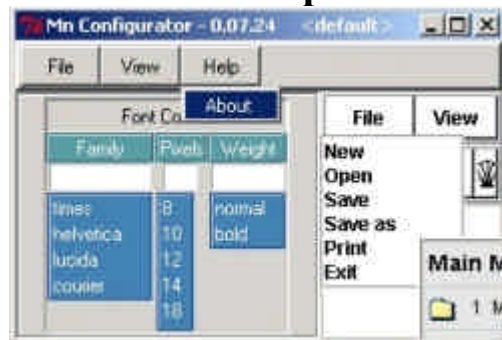
## Menu Bar – File



## Menu Bar – View

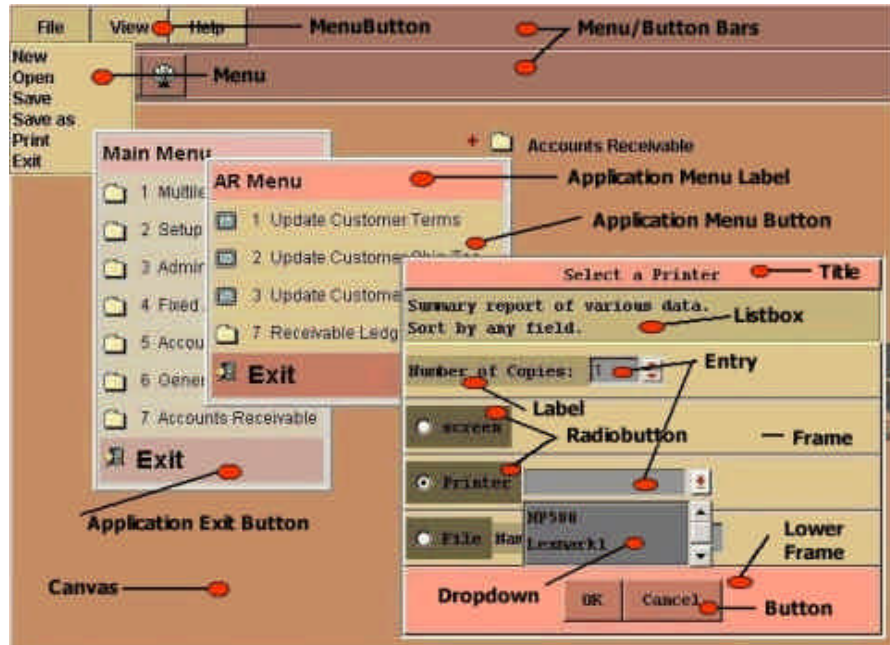


## Menu Bar – Help



## 4.3 Sample Area

Click on **File** and select **Open**, next use the drop-down box and open 'sepia'.

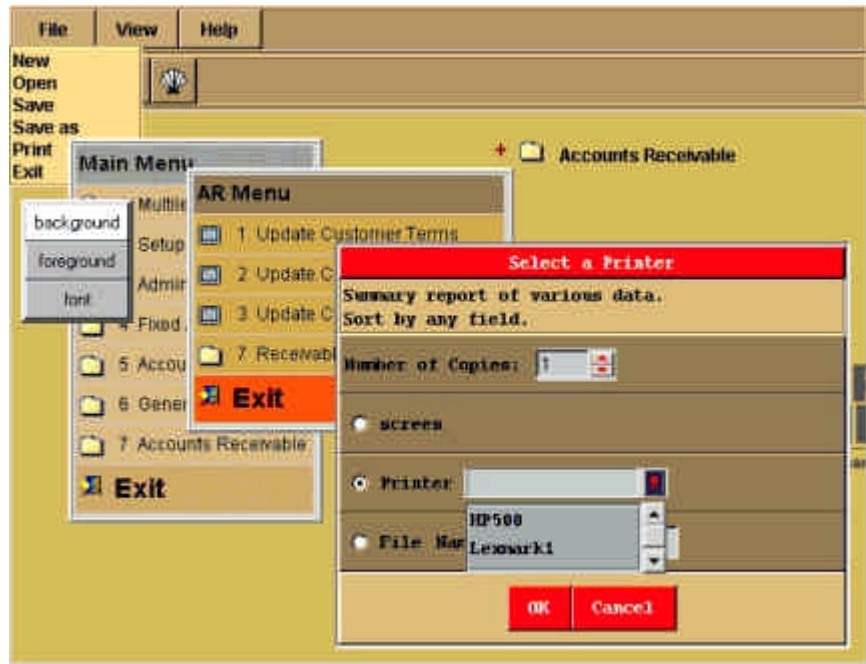


The Sample Area above is composed of discrete areas (widgets) that can be individually configured as to background color and foreground (text) color and font. Not all widgets have text.



**Fitrix Software**  
*Visual Menus User Reference*

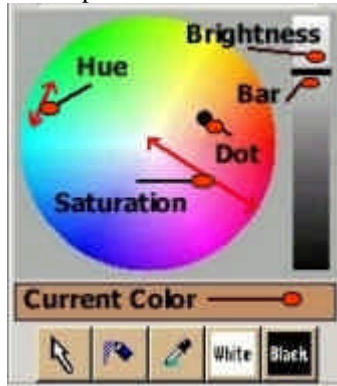
To configure a widget, begin by right clicking on it and selecting from a pop-up menu which characteristic to configure:



The user has clicked on the canvas area and has selected 'background'.

## 4.4 Color Configuration

Next the current color will be filled into the 'Current Color Rectangle'. See the close up below:



To change the color the user must manipulate the controls labeled above.

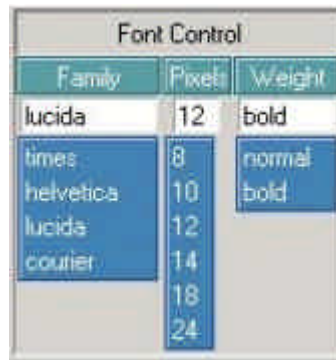
- ☐ Hue: Position around the perimeter of the color circle.
- ☐ Saturation: Distance from the center of the color circle.
- ☐ Brightness: Vertical distance within the right hand rectangle
- ☐ Dot: User left clicks and moves the dot to select hue and saturation.
- ☐ Bar: User left clicks and moves the bar to select brightness.
- ☐ Current Color: When the 'Dot' or 'Bar' are released the color selected is represented here and the background or foreground color of the widget is changed.

Not labeled are the buttons below the 'Current Color' rectangle. They are:

- ☐ Pointer: Normal mode for selecting a widget within the 'Sample Area'
- ☐ Spray can: Next time user clicks on a widget within the 'Sample Area' the 'Current Color' will be filled in.
- ☐ Eye Dropper: Next time user clicks on a widget within the 'Sample Area' the background color will become the 'Current Color'.
- ☐ White: Widget selected (background or foreground color) will be made white.
- ☐ Black: Widget selected (background or foreground color) will be made black.

## 4.5 Font Configuration

If the user selects 'Font' from the widget's pop-up menu the 'Font Control' area is used to change the font. Study the close-up below.



First, notice that the current font of the widget has been filled in. Double clicking on a 'Family', 'Pixels', or 'Weight' selection will change the text appropriately.

## 4.6 Logo Placement

First, from the Menu Bar select **View** and the **Place Logo**



The Sample Area will change to an blank canvas showing only the logo file—  
'logo.gif'.

## Fitrix Software

### *Visual Menus User Reference*



The new cursor ( ) represents where to place the upper left corner of the logo. Left click and release and the logo will move.



# Appendix A

## □ Make a change to upper level libraries

To distinguish between ordinary program errors (1) and the "No rows to process" error (7) make this change.

```
# cd $fglibdir/lib/report.4gs
# vi ./exitnrow.4gl
```

```
#####
function exit_nrows()
#####
# stub function called by flow_control.  if a program must do
# something when no rows are selected for a report, they can create
# this function locally.
#

    # if no_exit is set, do not exit the program.  just return.
    if no_exit then return end if

# TGS 09/25/2000
    CALL set_exit_nrow_stat(7)
```

```
# fg.make -mn -a -R
```

Now go into each 'report' directory and...

```
# fg.make -l
```