NAME

tgif − Xlib based interactive 2-D drawing facility under X11. Supports hierarchical construction of drawings and easy navigation between sets of drawings. It's also a hyper-graphics (or hyper-structured-graphics) browser on the World-Wide-Web.

SYNOPSIS

tgif [**-display** displayname] [**-fg** <color>] [**-bg** <color>] [**-bd** <color>] [**-rv**] [**-nv**] [**-bw**] [**-reqcolor**] [**-cwo[+sbwarp]**] [**-hyper**] [**-exec <file>**] [**-dbim {xcin|chinput|xim|kinput2}**] [**-usexlib**] [**-a4**] [**-listdontreencode**] [**-geometry** <geom>] [**=<geom>**] [{*file*[*.obj*]|**-merge** *file1*[*.obj*] *file2*[*.obj*] ...}]

or

tgif -print [**-eps**] [**-p**] [**-ps**] [**-f**] [**-text**] [**-epsi**] [**-tiffepsi**] [**-gif**] [**-png**] [**-jpeg**] [**-xpm**] [**-xbm**] [**-html**] [**-pdf**] [**-netlist**] [**-display** displayname] [**-stdout**] [**-raw[+h[eaderonly]]**] [**-dosepsfilter** [**-previewonly**]] [**-status**] [**-gray**] [**-color | -reqcolor**] [**-adobe | -adobe=<number>/<number> | -adobe=false**] [**-dontreencode=<string> | -listdontreencode**] [**-producedby=<string>**] [**-page <number>**] [**-print_cmd "<command>"**] [**-one_file_per_page**] [**-pepsc**] [**-nolandpdfspd**] [**-dontcondense | -condensed**] [**-a4**] [**-quiet**] [**-bop_hook "<string>"**] [**-eop_hook "<string>"**] [**-tmp_file_mode "<octal number>"**] [**-o**<dir>] [**-exec <file>**] [*file1*[*.obj*] *file2*[*.obj*] ...]

DESCRIPTION

Tgif is an interactive drawing tool that allows the user to draw and manipulate objects in the X Window System. Tgif runs interactively in the first form. In the second form shown in the SYNOPSIS section, tgif just prints *file1.obj*, *file2.obj*, etc. (generated by *tgif*) into PostScript(TM) page description files (without opening windows or fonts) and pipes them to lpr(1) if none of the **-eps**, **-p**, **-epsi**, **-tiffepsi**, **-gif**, **-png**, **-jpeg**, **-xpm**, **-xbm**, **-html**, **-pdf**, **-ps**, **-f**, **-text**, or **-netlist** options are specified. This form of *printing* is tgif's way of *exporting* a tgif file to another format. In this case, any other unrecognized command line options are sent to lpr(1). In this mode, tgif is compatible with the obsoleted *prtgif*. A symbol file (see descriptions below) can also be printed by specifying the *.sym* extension explicitly.

The command line argument *file* specifies a file or an Uniform Resource Locator (URL) of objects to be initially edited by tgif. Only HTTP or FTP URL's are supported. (For a more detailed description of URL and the World-Wide-Web, the reader is referred to [1].)

Tgif is purely based on *Xlib*. It is tested under X11R6, and it requires a 3 button mouse.

OPTIONS

In the first form shown in the SYNOPSIS section, the command line arguments can be:

-fg Foreground color specified in <color>.

-bg Background color specified in <color>.

-bd Border color specified in <color>.

-rv Start tgif in reversed-video mode.

-nv Start tgif in normal-video mode.

-bw Start tgif in black and white mode.

-reqcolor

Same effect as setting the Tgif.PrintUsingRequestedColor X default to true (see the X DEFAULTS section below).

-cwo Canvas Window Only. Only the canvas window (see TGIF SUBWINDOWS section below) will be displayed. This has the same effect as setting the Tgif.CanvasWindowOnly X default to true.

-cwo+sbwarp

If **-cwo+sbwarp** is used, single-button-warp (clicking the left mouse button to warp) is used to activate teleporting (see TELEPORT/HYPERJUMP section below).

-hyper Start tgif in the *hyperspace* mode (see HYPERSPACE section below).

-exec <file>
After tgif starts, execute the internal command in <file> (see INTERNAL COMMANDS section below). If <file> is the string "-", tgif executes internal commands from the standard input.

-dbim *method*
Use *method* as the input method for double-byte fonts (see DOUBLE-BYTE INPUT METHOD section below).

-usexlib
If tgif is compiled with -DUSE_XT_INITIALIZE, X Toolkit initialization routines will be used to setup tgif. Using this commandline option will force tgif to ignore the -DUSE_XT_INITIALIZE compiler option and use Xlib only. This is useful when the system resource file for tgif is not installed properly or messed up and needs to be bypassed.

-a4 Using this option has the same effect as setting the Tgif.PSA4PaperSize X default to true.

-quiet If this option is used, tgif will surpress standard messages.

-listdontreencode=<string>
If this option is used, tgif will print out the list of PostScript font names specified in the -D_DONT_REENCODE compiler option used in compiling tgif.

In the second form shown in the SYNOPSIS section, the command line arguments can be:

-eps (or **-p**)
Generates an Encapsulated PostScript(TM) file in *file.eps*; this file can be included in a LaTeX file through the \psfig, \epsf, or \psfile construct (see the LATEX FIGURE FORMATS section below).

-ps (or **-f**)
Generates a PostScript file in *file.ps*; this file can be printed to a PostScript printer with lpr(1).

-text Generates a text file in *file.txt*; the text file contains all visible text and can be fed to a spell checker.

-epsi Generates an Encapsulated PostScript (EPS) file with a preview bitmap in *file.eps*. Tgif aborts if a valid display is not accessible.

-tiffepsi
Generates an EPS file with a DOS EPS Binary File Header and a trailing TIFF image in *file.eps*. See the GENERATING MICROSOFT WINDOWS EPSI FILES section for more details. Tgif aborts if a valid display is not accessible.

-gif Generates a GIF file in *file.gif*. Please see the notes for Tgif.GifToXpm in the X DEFAULTS section below. Tgif aborts if a valid display is not accessible.

-png Generates a PNG file in *file.png*. Tgif aborts if a valid display is not accessible.

-jpeg Generates a JPEG file in *file.jpg*. Tgif aborts if a valid display is not accessible.

-xpm Generates an X11 pixmap (XPM) file in *file.xpm*. Tgif aborts if a valid display is not accessible.

-xbm Generates an X11 bitmap (XBM) file in *file.xbm*. Tgif aborts if a valid display is not accessible.

-html Generates a GIF file in *file.gif* and an HTML file in *file.html*. Tgif aborts if a valid display is not accessible.

-pdf Generates a GIF file in *file.gif* and an PDF file in *file.pdf*. Please see the notes for Tgif.PsToPdf in the X DEFAULTS section below.

-netlist Generates a text file in *file.net* and a text file in *file.cmp*. *file.net* contains netlist information stored in a table. The first line in it contains column names and each line in it is a port name (surrounded by double-quotes), followed by a comma and a <TAB> character, followed by a signal name (also surrounded by double-quotes). *file.cmp* contains information about components in the file. Each component begins with its name followed by its type. The attributes of a component are printed afterwards (indented by <TAB> characters).

-stdout Sends the output to the standard output instead of generating the output in a file.

-raw Causes the content of the files to be dumped to stdout.

-raw+h
If **-raw+h** is used and if the file is an HTTP URL, the HTTP header is also dumped to stdout.

-raw+headeronly
If **-raw+headeronly** is used and if the file is an HTTP URL, the HTTP header is dumped to stdout.

-dosepsfilter
Makes tgif act as a filter for gettting rid of the DOS EPS Binary File Header and the trailing TIFF image in a DOS/Windows EPS file.

-previewonly
If **-dosepsfilter** is specified, **-previewonly** makes tgif act as a filter for extracting the preview bitmap from the trailing TIFF image in a DOS/Windows EPS file.

-status If this option is used in conjunction with either **-raw**, **-raw+h**, or **-raw+headeronly** causes a status line to be displayed in stderr.

-gray Using this option has the same effect as setting the Tgif.UseGrayScale X default to true (see the X DEFAULTS section below).

-color (or **-reqcolor**)
To print in color, one can use either the **-color** or the **-reqcolor** option. The only difference between the two is that using **-reqcolor** has the same effect as setting the Tgif.PrintUsingRequestedColor X default to true (see the X DEFAULTS section below).

-adobe (or **-adobe=<number>/<number>** **-adobe=false**)
Using this option has the same effect as specifying the Tgif.UsePsAdobeString X default.

-dontreencode=<string>
Using this option has the same effect as specifying the Tgif.DontReencode X default.

-producedby=<string>
Using this option has the same effect as specifying the Tgif.ProducedBy X default.

-page Causes a specified page (specified by <page>) to be printed.

-print_cmd
Using this option has the same effect as specifying the Tgif.PrintCommand X default.

-one_file_per_page
Causes each page to be printed into a separate file.

-pepsc Preserve EPS Comment. This command line option is obsoleted since EPS comments are always preserved starting from tgif-4.0.11.

-nolandpdfspd
Do not generate "segpagedevice" command in exporting landscape PDF files. Using this option has the same effect as setting the Tgif.LandscapePdfSetPageDevice X default to false.

-dontcondense
Using this option has the same effect as setting the Tgif.DontCondensePSFile X default to true.

-condensed
Using this option has the same effect as setting the Tgif.DontCondensePSFile X default to false.

-bop_hook and **-eop_hook**
Using these options have the same effect as specifying the Tgif.PSBopHook and Tgif.PSEpsHook X defaults.

-tmp_file_mode
Using this option have the same effect as specifying the Tgif.TmpFileMode X defaults.

-o If this option is not specified, the output file (eps, ps, etc.) goes into the same directory as the input file. If **-o**dir is specified, the output file goes into the directory specified by **<dir>**.

-merge *file1 file2 ...*
Using this option merges *file1*.obj, *file2*.obj, etc. into a multipage file.

BASIC FUNCTIONALITIES

Primitive objects supported by tgif are rectangles, ovals, rounded-corner rectangles, arcs, polylines, polygons, open-splines, closed-splines, text, X11 bitmaps, some specific forms of X11 pixmaps, and Encapsulated PostScript. (Please note that the splines tgif draw are *not* Bezier curves.) Objects can be grouped together to form a *grouped* object. A primitive or a grouped object can be made into an *icon* object or a *symbol* object through user commands.

Tgif objects are stored in two types of files. A file with a *.obj* extension (referred to as an *object* file) is a file of objects, and a file with a *.sym* extension (referred to as a *symbol* file) specifies a ''building-block'' object. A *teleport* mechanism is provided to *travel* (or *hyperjump*) among the .obj files. A building-block object consists of the *representation* part and the *definition* part (which can be empty) of the object. Tgif supports the ''bottom-up'' construction of hierarchical drawings by providing the capability to ''instantiate'' a building-block object in a drawing. Tgif also supports the ''top-down'' specification of drawings by allowing the user to make any object a *representation* of an un-specified subsystem. Both types of files are stored in the form of Prolog facts. Prolog code can be written to interpret the drawings! (It is left to the user to produce the code. See the PROLOG/C TESTDRIVE section for more details.) Prolog engines are referred to as *drivers* in the sections to follow. (Other types of drivers are also allowed, e.g., written in C.)

Text based *attributes* can be attached to any non-text object. Attributes specified in the representation part of a building-block object are non-detachable when such an object is instantiated. See the ATTRIBUTES section for details.

Tgif can generate output in a few different formats. By default, the output is in the PostScript format (color PostScript is supported), and it is generated into a file named /tmp/Tgifa* (produced by mktemp() calls) where * is a number; this file is piped to lpr(1). This takes place when the laser-printer icon is displayed in the Choice Window (see the TGIF SUBWINDOWS section for the naming of tgif windows). This output can be redirected to a file with a *.ps* extension. This takes place when the *PS* icon is displayed in the Choice Window. When the *PDF* icon is displayed in the Choice Window, the output is generated into a file with a *.pdf* extension. By default, tgif calls ps2pdf(1) from the ghostscript(1) package to convert a *PS* file to a *PDF* file. When the *LaTeX* (or *EPSI*) icon is displayed in the Choice Window, the output is generated into a file with a *.eps* extension. This file is in the Encapsulated PostScript (or Encapsulated PostScript Interchange) format; it can be included in a LaTeX document with the *\psfig* or the *\epsf* construct; this will be discussed later. The only difference between the EPS and EPSI formats is that an EPSI file contains a preview bitmap. However, it takes time to generate the preview bitmap. If the EPS/EPSI file is to be incorporated into some tool that does not know how to use the preview bitmap, time can be saved by not using the EPSI format. When the *T* icon is displayed in the Choice Window, the output generated into a file with a *.txt* extension. This is a text file containing all visible text; it can be fed to a spell checker. When the *x11bm* (X11 bitmap) icon is displayed in the Choice Window and color output is *not* selected, tgif generates the output with the *.xbm* extension; the output is in the X11 bitmap format. However, if the x11bm icon is displayed in the Choice Window and color output *is* selected (through the ^#k keyboard command -- ^ denotes the <Control> and # denotes the <Meta> or <Alt> key), then tgif generates the output with the *.xpm* extension, and the output is in the X11 pixmap format (the version of this XPM format depends on the settings of the Tgif.XPmOutputVersion X default). When the *GIF* icon is displayed in the Choice Window, the output is generated into a file with a *.gif* extension. By default, tgif calls xpmtoppm and ppmtogif from the netpbm(1) package to convert an *XPM* file to a *GIF* file.

X11 bitmap files, certain forms of X11 pixmap files (such as the one generated by tgif; see the section on X11 PIXMAP for details), GIF files, and Encapsulated PostScript (EPS) files can be *imported* into tgif and be represented as tgif primitive objects. Files in other raster formats (e.g, JPEG, TIFF, etc.) can also be imported into tgif if external tools can be used to convert them into X11 pixmap files. Please see the

IMPORT RASTER GRAPHICS section for details.

Tgif drawings are supposed to be printed on letter size paper (8.5in by 11in). Both landscape and portrait page styles are supported by tgif. Reduction (or magnification) can be controlled by the #% keyboard command to set the reduction/magnification. If the compiler flag -DA4PAPER is defined (in Imakefile or Makefile.noimake), then the output is supposed to be printed on A4 papers (which has approximate dimensions of 8.25in by 11.7in).

GRAPHICAL OBJECTS

An object in an *object* (*.obj*) file can be a primitive object, a grouped object, or an *icon* object. A *symbol* (*.sym*) file can have any number of objects allowed in an object file and exactly one *symbol* object. (Recall that a symbol file specifies a building-block object.) The symbol object in a symbol file is the representation part of the building-block object, and the rest of the symbol file is the definition part of the building-block object. The symbol object is highlighted with a dashed outline to distinguish it from the rest of the objects. When a building-block object is instantiated, the symbol part of the file is copied into the graphics editor, and it becomes the icon for the building-block object.

All objects in tgif can be moved, duplicated, deleted, rotated, flipped, rotated, and sheared. However, in the non-stretchable text mode, text objects can not be stretched. For an text object, if it has not been stretched, rotated, or sheared, flipping it horizontally will cause the text justification to change and flipping it vertically has no effect.

Tgif supports 32 fill patterns, 32 pen patterns, 7 default line widths, 4 line styles (plain, head arrow, tail arrow, double arrows) for polylines and open-splines, 9 dash patterns, 3 types of text justifications, 4 text styles (roman, italic, bold, bold-italic), 11 default text sizes (8, 10, 12, 14, 18, and 24 for the 75dpi fonts and 11, 14, 17, 20, 25, and 34 for the 100dpi fonts), 5 default fonts (Times, Courier, Helvetica, New-Century-Schoolbook, Symbol), and 11 default colors (magenta, red, green, blue, yellow, pink, cyan, cadet-blue, white, black, dark-slate-gray). Additional line widths can be added through the use of Tgif.MaxLineWidths, Tgif.LineWidth#, Tgif.ArrowWidth#, and Tgif.ArrowHeight# X defaults. Additional text sizes can be added through the use of Tgif.FontSizes X default. Additional fonts can be added through the use of Tgif.AdditionalFonts X default. If the defaults fonts are not available, their replacement fonts can be specified by Tgif.HasAlternateDefaultFonts and related X defaults. Additional colors can be added through the use of Tgif.MaxColors, and Tgif.Color# X defaults. One can also select AddColor() from the Edit Menu to add a color.

Most commands in tgif can either be activated by a popup menu or by typing an appropriate non-alphanumeric key. All operations that change any object can be undone and then redone. Commands such as zoom, scroll, change fonts while no text objects are selected, etc. are not undoable. The undo/redo history buffer size can be set using the Tgif.HistoryDepth X default.

TGIF SUBWINDOWS

The tgif windows are described in this section.

Top Window

Displays the current domain and the name of the file tgif is looking at. Mouse clicks and key presses have no effect.

Menubar Window

This window is right under the Top Window. Pull-down menus can be activated from it with any mouse buttons. Key presses have no effect. If HideMenubar() is selected from the Layout Menu, this window becomes invisible. If ShowMenubar() is selected from the Layout Menu (which can be activated from the Canvas Window below), this window becomes visible.

The View, Text, and Graphics pull-down menus are cascading menus and can not be *pinned* (see the *Popup Menus* subsection below for a description).

Message Window

This is right under the Menubar Window and to the left. It displays tgif messages. Clicking the left mouse button in this window scrolls the messages towards the bottom, clicking the right mouse button scrolls towards the top, and clicking or dragging the middle mouse button scrolls to

the location in the message history depending on where the mouse is clicked. If the <Shift> (or <Control>) key is held down when clicking the left/right mouse button, it scrolls right/left.

Panel (Choice) Window

This is the window to the right of the Message Window, and it contains a collection of icons (not to be confused with the tgif icon objects) reflecting the current state of tgif. In top/bottom, left/right order, it displays the current drawing mode, the page style (portrait or landscape), edit (see below), print/export mode, zoom factor, move and stretch mode (constrained or unconstrained), radius for rounded-corner rectangles, text rotation, page number or row/column, page layout mode (stacked or tiled), horizontal alignment (L C R S -), vertical alignment (T M B S -), font, text size, vertical spacing between lines of text within the same text object, text justification, shape (see below), stretchable or non-stretchable text mode, dash pattern, line style, polyline, spline, or interpolated spline, line width, fill pattern, pen pattern, color, and special (see below). Key presses have no effect in this window.

In addition to displaying the current state of tgif, the icons in the Choice Window can also be used to change the current state. Each icon is associated with a particular state variable of tgif. Clicking the left mouse button on top of an icon cycles the state variable associated with the icon forward; clicking the right mouse button cycles the state variable backwards. Dragging the middle mouse button on top of an icon usually generates a popup menu which corresponds to an entry in the Main Menu for the Canvas Window below. (The ‘‘edit’’, ‘‘shape’’, and ‘‘special’’ icons mentioned above are dummy icons that allow the ‘‘edit’’, ‘‘shape’’, and ‘‘special’’ menus to be accessed in the Choice Window. They do not respond to left and right mouse clicks.) The response to the dragging of the middle mouse button is different for the zoom, radius, and vertical spacing icons. Dragging the mouse left or up increases the zoom or decreases the radius or vertical spacing; dragging the mouse right or down has the opposite effect.

If there are objects selected in the canvas window, then the action of the mouse will cause the selected objects to change to the newly selected mode; note that in this case, the current choice won’t change if the middle mouse button is used (unless the Tgif.StickyMenuSelection X default is set to true).

The settings of the horizontal and vertical alignments determine how objects (or vertices) align with each other when the ^l keyboard command is issued, how each individual object (or vertex) aligns with the grids when the ^t keyboard command is issued, how objects or vertices distribute spatially with respect to each other when the #l keyboard command is issued, and how each icon replaces the old icon when the ^#u keyboard command is issued. The horizontal alignments are left (L), center (C), right (R), space (S), and ignore (-). The vertical alignments are top (T), middle (M), bottom (B), space (S), and ignore (-). In aligning operations, the space (S) and the ignore (-) settings have the same effect. The space settings are used to distribute objects such that the gaps between any two neighboring objects are equal. In vertex mode, any non-ignore setting will cause the selected vertices to be spaced out evenly. The best way to understand them is to try them out.

The text vertical spacing determines the vertical distance to advance when a carriage return is pressed during text editing. If the user tries to set the value too negative, such that the next line is exactly at the same position as the current line, such a setting will not be allowed (this distance depends on the current font and font size).

Canvas Window

This is the drawing area. The effects of the actions of the mouse are determined by the current drawing mode. Before tgif-4.x, dragging the right mouse button will generate the Mode Menu. This is disabled by default in tgif-4.x, but you can turn it on using the Tgif.Btn3PopupModeMenu X default.

The drawing modes are (in order, as they appear in the Mode Menu) select, text, rectangle, corner oval, center oval, edge circle, polyline (open-spline), polygon (closed-spline), arc (center first), arc (endpoints first), rounded-corner rectangle, freehand polyline (open-spline), select vertices, and rotate/shear. When drawing a rectangle, an oval, or a rounded-corner rectangle, if the <Shift> key is held down, a square, a circle, or a rounded-corner square is drawn. Dragging the middle mouse

button will generate the Main Menu.

In the select mode, left mouse button selects, moves, stretches, and reshapes objects (double-click will ''de-select'' all selected objects in vertex mode). When an object is selected, it is highlighted by little squares (referred as handles here) at the corners/vertices (using the Tgif.HandleSize X default, the sizes of the handles can be customized). Dragging one of the handles stretches/reshapes the selected object. If one wants to move a selected object, one should not drag the handles. Instead, one should drag other parts of the object. For example, if the object is a hollow rectangle (the fill is NONE and the pen is not NONE), in order to select the rectangle, one should to click on the outline of the rectangle with the left mouse button. If one would like to move the rectangle, one should drag the outline of the rectangle with the left mouse button. If the object is a filled rectangle (fill is not NONE), one can click inside the rectangle to select it and drag anywhere inside the rectangle to move it.

Holding down the <Shift> key and clicking the left mouse on an object which is not currently selected will add the object to the list of already selected objects. The same action applied to an object which is already selected will cause it to be de-selected. When stretching objects (not reshaping poly-type objects), holding down the <Shift> key *after* stretching is initiated activates proportional stretching (basically, a scale operation is being performed). In non-stretchable text mode, text objects can not be stretched or scaled.

Double-clicking or clicking the middle mouse button while the <Shift> key is held down will activate the *teleport* (or *travel*), the *launch*, or the *execute internal command* mechanism. See the sections on TELEPORT/HYPERJUMP, LAUNCH APPLICATIONS, and INTERNAL COMMANDS for details. Teleporting has precedence over launching, which has precedence over executing an internal command. In the text drawing mode, dragging the middle mouse button while the <Cntrl> key is held down inside the edit text area will move the edit text area.

The arrow keys can also be used to move selected objects. However, if no objects are selected, using the arrow keys will scroll the drawing area by a small amount, and using the arrow keys when <Control> key is held down will scroll a screen full.

In the select vertices mode, left mouse button selects and moves vertices. Only the top-level polyline/open-spline and polygon/closed-spline objects which are selected when the vertex mode is activated are eligible for vertex operations. In this mode, all eligible objects have their vertices highlighted with squares. When a vertex is selected (using similar mechanism as selecting objects described above), it is doubly highlighted with a '+' sign. Operations available to these doubly highlighted vertices are move, delete, align (with each other), distribute (space them equally), and align to grid. The arrow keys can also be used to move selected vertices.

Objects can be locked (through the #< keyboard command). Locked object are shown with gray handles, and they can not be moved, stretched, flipped, rotated, or sheared. When objects are grouped, the resulting grouped object will also be locked if any one of it's constituents is locked. Locked objects can have their properties, such as color, font, pen, etc., changed; furthermore, they can be deleted.

If the current move/stretch mode is of the constrained type (activated and deactivated by the #@ keyboard command), top-level polylines will have the following behavior. In a move operation, if both endpoints of a polyline lie inside the objects being moved, then the whole polyline is moved; otherwise, if only one endpoint falls inside the objects being moved, then that endpoint is moved. The vertex that is the neighbor of the moved endpoint may also be moved either horizontally or vertically. If the last line segment is horizontal or vertical, then the neighbor vertex may be moved so that the direction of the last line segment is maintained. In a stretch (not reshape) operation, if an endpoint of a polyline lies inside the objects being moved, that endpoint will be moved. The vertex that is the neighbor of the moved endpoint will also be moved in the same manner as described above.

When the drawing mode is set to text (a vertical-bar cursor is shown), clicking the left mouse button causes selected text to go into edit mode. Dragging the left mouse button or clicking the left

mouse button while the <Shift> key is held down highlights substrings of the text. Double-clicking causes a word to be selected. In edit mode, key presses are treated as text strings being inputed, and arrow keys are used to move the current input position. If a key press is preceded by an <ESC> key, then the character's bit 7 is turned on. This allows non-ASCII (international) characters to be entered. One can use xfd(1) to see what the corresponding international character is for an ASCII character. For the Symbol font, symbols such as the integral, partial derivative, and copyright symbols can all be found in this range. There are some characters that are supported by X11 but not by PostScript; these characters are not accepted by tgif. If the text being edited is an attribute of a object, <Meta><Tab> will move the cursor to the next visible attribute and <Shift><Tab> will move the cursor to the previous visible attribute.

If the drawing mode is set to draw polygons (not closed-splines) and if the <Shift> key is held down, the rubber-banded polygon will be self-closing.

The freehand drawing mode can be used to draw polylines and open splines. All intermediate points are specified by moving the mouse (as opposed to clicking the mouse buttons as in the polyline mode). The second endpoint is specified by releasing the mouse button.

In all drawing modes (other than the text mode), pressing the <ESC> key cancels the drawing (creation) of the current object.

Middle mouse button always generates the main tgif popup menu. Holding down the <Shift> key and clicking the right mouse button will change the drawing mode to *select*. Key presses with the <Control> or <Meta> key held down (referred to as *non-alphanumeric* key presses since they can also generate control characters) are treated as commands, and their bindings are summarized in the next section. Users can also define single key commands to emulate the functions of the non-alphanumeric key commands. The SHORTCUTS section will describe the details.

Scrollbars

Clicking the left mouse button in the vertical/horizontal scrollbar causes the canvas window to scroll down/right by a small distance; clicking the right mouse button has the reverse effect. (The scrollbars in the popup windows for selecting file names and domain names behave similarly.) Clicking with the <Shift> key held down will scroll a window full. Clicking or dragging the middle button will cause the page to scroll to the location which corresponds to the gray area in the scrollbars. (Tgif insists that the left-top corner of the Canvas Window is at a distance that is a non-negative multiple of some internal units from the left-top corner of the actual page.)

Rulers

They track the mouse location. Mouse clicks and key presses have no effect. When the page reduction/magnification is set at 100%, the markings in the rulers correspond to centimeters when the metric grid system is used, and they correspond to inches when the English grid system is used. When the page reduction/magnification is not set at 100%, the markings do not correspond to the above mentioned units any more (this is considered as a known bug).

Interrupt/Hyperspace Window

This window is right below the Message Window and to the left of the horizontal ruler. When the Tgif.IntrCheckInterval X default has a positive value, an interrupt icon is visible when the Canvas Window is being redrawn. If the user clicks on this window when the interrupt icon is visible, tgif aborts the repainting of the objects. If this is done when a file is being opened (either through Open() or Push()), the drawing of objects is stopped, but the reading of the file continues (reading of the file is not aborted).

If tgif is currently in the *hyperspace* mode (please see the HYPERSPACE section below for more details), a space ship icon will be displayed when the interrupt icon is not being displayed. Clicking any button in this window will switch tgif in and out of the hyperspace mode.

Page Control Window

The Page Control Window is to the left of the horizontal scrollbar. This window is empty if the current page mode is set to the *tijled* page mode. If the current page mode is set to the *stacked* page mode, each page has a tab in tabs subwindow of this window. Clicking the left mouse button

on a tab goes to the corresponding page. Clicking the middle mouse button brings up the Page Menu. When there are too many pages in a drawing so that one can not see the tabs for all the pages, one can use the icons to the left side of the Page Control Window to scroll the tabs subwindow. Clicking on the first icon scrolls the tabs subwindow such that the first tab is visible. Clicking on the 4th icon scrolls the tabs subwindow such that the last tab is visible. Clicking on the 2nd icon scrolls the tabs subwindow towards the first tab by one tab and clicking on the 3rd icon scrolls the tabs subwindow towards the last tab by one tab.

Status Window

This window is below the horizontal scrollbar. It shows what action will be taken if a mouse button is depressed. When a menu is pulled down or popped up, this window shows what action will be taken if a menu item is selected. It also displays miscellaneous status information. Mouse clicks and key presses have no effect. If HideStatus() is selected from the Layout Menu, this window becomes invisible. If ShowStatus() is selected from the Layout Menu, this window becomes visible.

By default, when this window is displaying mouse button status, right-handed mouse is assumed. Setting the Tgif.ReverseMouseStatusButtons X default to true will reverse the status (as if a left-handed mouse is used).

Popup Menus

When a menu is popped up by a mouse drag, the menu can be *pinned* if it is dragged far enough horizontally (the distance is determined by the setting of the Tgif.MainMenuPinDistance X default). Clicking the right mouse button in a pinned menu will cause it to disappear. Dragging the left mouse button in a pinned menu will reposition the menu (except when the Tgif.TitledPinnedMenu X default is set to true in which case the left mouse button performs the same function as the middle mouse button). Clicking the middle mouse button in it will activate the item right below the mouse.

NON-ALPHANUMERIC KEY BINDINGS

Most operations that can be performed in tgif can be activated through non-alphanumeric keys (some operations can only be activated through popup menus or shortcut keys). This section summarizes the operations that can be activated by a key stroke with the <Control> and/or the <Meta> key held down. ‘‘^’’ denotes the <Control> key and ‘‘#’’ denotes the <Meta> key in the following description. (The ‘‘*keys.obj*’’ file, distributed with tgif, also summarizes the same information, but it is organized differently. This file can be viewed with tgif, and if installed properly, it can be found in the same directory as the ‘‘tgificon.obj’’ file, mentioned in the FILES section of this document.)

^a	select all
^b	send selected objects to the back
^c	copy selected objects into the cut buffer
^d	duplicate selected objects
^e	save/restore drawing mode
^f	send selected objects to the front
^g	group selected objects (the grouped object will be brought to the front)
^i	instantiate a building-block object
^k	pop back to (or return to) a higher level and close the symbol file (reverse of ^v)
^l	align selected objects according to the current alignment settings
^n	open a new un-named object file
^o	open an object file to edit
^p	print the current page (or export in XBM, XPM, GIF, HTML, PDF, EPS, or PS formats)
^q	quit tgif
^r	redraw the page
^s	save the current object/symbol file
^t	align selected objects to the grid according to the current alignment
^u	ungroup selected objects
^v	paste from the cut buffer

^w	change the drawing mode to text
^x	delete all selected objects
^y	change domain
^z	escape to driver
^,	scroll left
^.	scroll right
^-	print the current page with a specified command
#a	attach selected text objects to a selected non-text object as attributes
#b	escape to driver
#c	rotate selected objects counter-clockwise
#d	decrement the grid size
#e	send a token on a selected polyline
#f	flash a selected polyline
#g	show/un-show grid points
#h	flip the selected objects horizontally
#i	increment the grid size
#j	hide the attribute names of the selected objects
#k	change the drawing mode to select
#l	distribute selected objects according to the current alignment
#m	move/justify an attribute of a selected object
#n	show all the attribute names of the selected objects
#o	zoom out
#p	import a .obj or a .sym file into the current file
#q	change the drawing mode to polyline/open-spline
#r	change the drawing mode to rectangle
#s	escape to driver
#t	detach all the attributes of the selected objects
#u	undo
#v	flip the selected objects vertically
#w	rotate the selected objects clockwise
#x	escape to driver
#y	escape to driver
#z	zoom in
#9	create a user-specified arc (12 o’clock position is 0 degree)
#0	update the selected objects according to current settings
#,	scroll up
#.	scroll down
#-	show all the attributes of the selected objects
#[	align the left sides of objects
#=	align the horizontal centers of objects
#]	align the right sides of objects
#{	align the top sides of objects
#+	align the vertical centers of objects
#}	align the bottom sides of objects
#"	make the selected polygon regular (fit the original bounding box)
#%	set the percent print reduction (if < 100%) or magnification (if > 100%)
#:	go to default zoom
#‘	zoom out all the way so that the whole page is visible
#~	saved selected objects in a new file
#;	cut and/or magnify a selected bitmap/pixmap object
#_	abut selected objects horizontally
#|	abut selected objects vertically
##	break up text objects into single character text objects

#^	scroll to the origin set by SaveOrigin()
#@	toggle between constrained and unconstrained move (stretch) modes
#$	change the drawing mode to select vertices
#&	align selected objects to the paper according to the current alignment
#*	redo
#(	import an Encapsulated PostScript file
#)	scale selected objects by specifying X and Y scaling factors
#<	lock the selected objects (can't be moved, stretched, flipped, or rotated)
#>	unlock the selected objects

^#a	add points to the selected poly or spline
^#b	change the text style to bold
^#c	change to center justified text
^#d	delete points from the selected poly or spline
^#e	change the drawing mode to rounded-corner rectangles
^#f	reverse-video the selected bitmap objects
^#g	toggle snapping to the grid points
^#h	hide all attributes of the selected objects
^#i	make the selected object iconic
^#j	make the selected icon object a grouped object
^#k	select color or black-and-white output
^#l	change to left justified text
^#m	make the selected object symbolic
^#n	make the selected symbol object a grouped object
^#o	change the text style to roman
^#p	change the text style to bold-italic
^#q	change the drawing mode to polygon/closed-spline
^#r	change to right justified text
^#s	save the file under a new name
^#t	change the text style to italic
^#u	update iconic representations of selected objects
^#v	change the drawing mode to oval
^#w	toggle between poly and spline
^#x	cycle among the various output file formats
^#y	push into (or edit) the definition part of a building-block (icon) object
^#z	change the drawing mode to arcs
^#.	import an X11 bitmap file
^#,	import an X11 pixmap file
^#-	toggle between English and Metric grid systems
^#=	repeat the last Find command

SHORTCUTS

The user can define single character *shortcut* keys to emulate the function of the non-alphanumeric key presses to activate commands. This is done through the use of the Tgif.ShortCuts X default. (Please note that these shortcut keys are only active when the drawing mode is *not* set to the text mode.) The Tgif.ShortCuts consists of a list of items, each of which specifies the bindings between a key (may be case sensitive) and a command. The items are separated by blanks, and each item is interpreted as follows. It consists of two parts, KEY and COMMAND, which are concatenated together with a ':' character. The format of the KEY part is one of *:<Key>x*, *!<Key>x*, or *<Key>x* (here the character 'x' is used as an example; furthermore, the substring *<Key>* must be spelled exactly the way it appears here). The first 2 formats are equivalent, they specify the *lower case* x; the 3rd format specifies both the characters 'x' and 'X'. The COMMAND part is a string that matches strings in tgif's popup menus with space characters removed (exceptions are noted below). This is illustrated by the following example. In the Edit menu, two of the entries are,

"Delete ^x"

"SelectAll ^a"

which means that <Control>x activates and Delete() command, and <Control>a activates the SelectAll() command. Therefore, both Delete() and SelectAll() are valid names for the COMMAND part of a shortcut specification. To complete the example, the following line can be used to bind the lower case 'x' to Delete() and 'a' or 'A' to SelectAll():

```
Tgif.ShortCuts: !<Key>x:Delete() \n\
                <Key>a:SelectAll()
```

For more examples, please see the sample X defaults file, tgif.Xdefaults, included in the tgif distribution.

Here is a list of exceptions where the COMMAND does not match a command name in a menu entry. The left entry is a proper COMMAND name, and the right is a list of strings that's shown in popup menus which the COMMAND would correspond to.

CyclePrintFormat() Printer, LaTeXFig, RawPSFile, XBitmap, TextFile, EPSI, GIF/ISMAP, TiffEPSI, NetList
ToggleBW/ColorPS() BlkWhtPS, ColorPS
ToggleGridSystem() EnglishGrid, MetricGrid
ToggleMapShown() ShowBit/Pixmap, HideBit/Pixmap
ToggleUseGrayScale() UseGrayScale, NoGrayScale
ToggleMoveMode() ConstMove, UnConstMove
ToggleShowMeasurement() ShowMeasurement, HideMeasurement

ToggleLineType() (advances between different curved shapes)
ScrollPageUp() (scroll up a window full)
ScrollPageDown() (scroll down a window full)
ScrollPageLeft() (scroll left a window full)
ScrollPageRight() (scroll right a window full)
FreeHandMode() (change the drawing mode to freehand poly/open-spline)
CenterAnEndPoint() (move an endpoint of a polyline object to the center of another object)
ToggleNamedAttrShown(<x>=) (toggle name shown for the attribute <x>)
ToggleSmoothHinge() (convert smooth to hinge and hinge to smooth points)
ToggleShowMenubar() ShowMenubar, HideMenubar
ToggleShowStatus() ShowStatus, HideStatus
ToggleShowMode() ShowMode, HideMode
ToggleOneMotionSelMove() OneMotionSelMove, ClickSelClickMove
ToggleHyperSpace() GoHyperSpace, LeaveHyperSpace
ImportOtherFileType(<x>) (import using a filter named <x>)
BrowseOtherType(<x>) (browse using a filter named <x>)

In addition to the above list, the following are also valid COMMAND names (having the obvious meaning): ScrollLeft(), ScrollRight(), ScrollUp(), ScrollDown(), SelectMode(), DrawText(), DrawBox(), DrawOval(), DrawPoly(), DrawPolygon(), DrawRCBox(), DrawArc(), and SelectVertexMode().

COLORS AND COLORMAPS

In most X environments, only 256 colors can be displayed at once. In these environment, if an application needs 128 colors and another application needs a totally different 129 colors, both applications can not be displayed at once with all the colors they want. X solves the problem by allowing applications to use their own colormaps (known as private colormaps). Each private colormap can have at most 256 colors. There is also a shared colormap available for applications that do not wish to use private colormaps. The main problem with using private colormaps is that a user will see the the well-known *colormap flashing* phenomenon when he/she switchs in and out of applications that use private colormaps.

Tgif uses the shared colormap initially. When it needs more color than what is available in the shared colormap, it will use a private colormap automatically. When tgif no longer needs the extra colors, it does *not* automatically revert to using the shared colormap because it needs to be able to undo operations that use the extra colors. If one does no longer needs the objects in the undo buffer, one can select FlushUndoBuffer()

from the Edit Menu to flush the undo buffer. At this point, tgif will attempt to use the shared colormap to avoid the colormap flashing problem. If one often uses XPM and GIF objects, one can bind the <Shift>f key to the FlushUndoBuffer() operation by setting the following X default and uses the <Shift>f key to regain entries in the colormap when an XPM/GIF object is deleted:

```
Tgif.ShortCuts: !<Key>F:FlushUndoBuffer()
```

Even when a private colormap is used, only 256 colors can be used at once. Therefore, it is not possible to import two 256-colors GIF files into the same drawing unless the colors are somehow reduced to fit in the 256-colors colormap. This can be done through *dithering* which is described in the IMPORT RASTER GRAPHICS section below.

IMPORT RASTER GRAPHICS

The native raster graphics formats that tgif supports are the XBM and XPM formats. In order to import color raster graphics file of another format, tgif can work with external tools that can convert non-XPM format files to an XPM files. A popular raster format conversion toolkit is the *pbmplus*(1) (also known as the *netpbm*(1)) toolkit. It can convert a GIF file (e.g., "foo.gif") to an XPM file (e.g., "foo.xpm") with the following command (*giftopnm* is in netpbm; an earlier version of it called *giftoppm* exists in pbmplus):

```
giftopnm foo.gif | ppmtoxpm > foo.xpm
```

When working with tgif, a GIF file name will be supplied by tgif and the output of ppmtoxpm will be directly read by tgif through a pipe; therefore, the previous sequence is replaced by an X default containing the following form (which happens to be the default setting for the Tgif.GifToXpm X default):

```
giftopnm %s | ppmtoxpm
```

The "%s" is to be replaced by a GIF file name. The above is referred to as a *filter*.

To be able to import other types of raster graphics files, one can use Tgif.MaxImportFilters and Tgif.ImportFilter# X defaults to specify additional filters. The following example adds a JPEG filter:

```
Tgif.MaxImportFilters: 1
Tgif.ImportFilter0: \n\
    JPEG-222 jpg;jpeg \n\
    djpeg -gif -colors 222 %s | \n\
    giftopnm | ppmtoxpm
```

The "JPEG-222" above is the name given to the filter (must not contain any space character). The "jpg;jpeg" are possible file extensions separated by semicolons. The rest is the filter specification. The djpeg(1) program is part of the libjpeg distribution. It can convert a JPEG file to a GIF file. The above filter also restrict the output to have a maximum of 222 colors. (The 222 is chosen arbitrarily. Many XPM files use some ‘‘standard’’ 32 colors, so one may want to leave room form them.)

To invoke a filter, one can select ImportOtherFile() or BrowseOther() commands from the File Menu. This will bring up a dialogbox listing the available filters by their names (e.g., "JPEG-222"). After selecting a filter, tgif continues in a similar manner as with invoking ImportXPixmap() or BrowseXPixmap() commands from the File Menu.

The above example is not suitable for the BrowseOther() command because only 256 colors can be used in a drawing (as explained in the COLORS AND COLORMAPS section above). In order for BrowseOther() to work well, one can use *dithering* to represent an image with a *dithered* image that only uses a set of standard colors. The example below uses ppmdither from the pbmplus/netpbm toolkit:

```
Tgif.MaxImportFilters: 2
Tgif.ImportFilter0: \n\
    JPEG-222 jpg;jpeg \n\
    djpeg -gif -colors 222 %s | \n\
    giftopnm | ppmtoxpm
Tgif.ImportFilter1: \n\
    JPEG-dithered jpg;jpeg \n\
    djpeg -gif %s | \n\
```

giftopnm | ppmdither | ppmtoxpm

If one is working with one JPEG image, one can select ImportOtherFile() then select "JPEG-222" to get as many as 222 colors. If one is browsing for JPEG images, one can select BrowseOther() then select "JPEG-dithered".

OBJECT NAMES

If an object contains an attribute (please see the ATTRIBUTES sections below for details) whose name is the string *"name"* (case-sensitive), the value part of the attribute is the *name* of the object. Subobject of a composite object can be named using a *path*, e.g., *<t>!<s1>!<s2>!...*, where *<t>* is the name of a top-level object which directly contains *<s1>* which directly contains *<s2>*, etc. *!** refers to the currently selected object (if more than one object is selected, the top-most object in the stacking order is used). *!*<s1>!<s2>* names the *<s2>* subject of the *<s1>* subject of the currently selected object.

The following is *not fully* supported, yet (only the *#<page>* form is supported at this time). Every object in a tgif file can be uniquely named using the notation *#<page>!<path>*, where *<page>* can be a string that specifies the name of a page or *#<number>* which specifies a page number. The *<path>* is described in the previous paragraph. If an object *o1* is referenced by another object *o2* within the same file (no file name or URL is specified before *#*) and *<page>* is omitted, then *o1* must be on the same page as *o2*. If a file name or URL is specified before *#* and *<page>* is omitted, then *o1* must be on the first page.

ATTRIBUTES

Attributes are text strings of the form *name=value* or *value* which are attached to either the current drawing or any non-text objects. An attribute attached to the current drawing is called a *file attribute*; otherwise, it is a *regular attribute*. Attributes can be attached and detached from these objects except in the following case:

> Attributes appearing in the symbol object in a building-block object file can not be detached when the building-block object is instantiated. These attributes are considered to be the ‘‘inherited’’ attributes of the icon object. (If it is really necessary to detach inherited attributes of an icon object, the icon object can be ‘‘de-iconified’’ by using UnMakeIconic() in the Special Menu to make it a grouped object; then the attributes can be detached.)

A file attribute is always invisible. For a regular attribute, the user has control over which part of the attribute is displayed. An entire attribute can be made invisible, or only its name can be made invisible (accomplished through the commands under the special menu, such as #m, #n, #j, #-, and ^#h).

TELEPORT/HYPERJUMP

Tgif provides the mechanism to travel between .obj and .sym files. If the middle mouse button is clicked on an object with the <Shift> key held down (or double-clicking such an object), tgif looks for an attribute named *warp_to* (by default) or *href* of that object. The only difference between *warp_to* and *href* is that ".obj" is automatically appended to the value of a *warp_to* attribute while the value of a *href* attribute is taken as is. (Please note that *warp_to* is obsolete now. It is still supported for the sake of compatibility.) If such an attribute is found, the value part of the attribute is interpreted as the name of a .obj file to *travel* to. (If tgif is in the *hyperspace* mode, then clicking the left mouse button has the same effect.) If there are multiple *href* attributes on the object, but are in different colors, tgif will use the one that has the same color as the current color appearing in the Choice Window. If the current file is modified, the user is prompted to save the file before traveling to the next file. If the value part of the *href* attribute starts with the '/' character, the value is treated as an absolute file name; otherwise, it is treated as a relative file name.

HYPERSPACE

Tgif provides a *hyperspace* mode to facilitate traveling between .obj files. The hyperspace mode is entered when GoHyperSpace() is selected from the Navigate Menu. In hyperspace mode, the little window below the Message Window will show a little space ship. The hyperspace mode is also automatically entered when a remote URL is opened (unless the Tgif.AutoHyperSpaceOnRemote X default is set to false).

In the hyperspace mode, certain objects are considered *hot-links*. When the cursor is placed on top of these object, it will change from a pointer to a hand to indicate that clicking on the left mouse button will invoke some actions. An object is a hot-link if it contains an attribute described in either the TELEPORT/HYPERJUMP, LAUNCH APPLICATIONS, or INTERNAL COMMANDS section.

The hyperspace mode is exited when the drawing mode is changed or the LeaveHyperSpace() is selected from the Navigate Menu.

LAUNCH APPLICATIONS

Tgif provides the mechanism to launch applications. If the middle mouse button is clicked on an object with the <Shift> key held down (or double-clicking such an object), tgif looks for an attribute named *launch* (by default) of that object. If such an attribute is found, the value part of the attribute is interpreted as a sh(1) command to execute. Same color rule applies as described in the TELEPORT/HYPERJUMP section above. If the command ends with the '&' character, tgif forks itself (what actual happens depends on whether the _BACKGROUND_DONT_FORK compiler flag is defined or not at compile time) and the command is executed by the child process; otherwise, popen() is used to execute the command (in this case, if the command hangs, there is no way provided to terminate the command, and tgif will not be able to recover from it). Within the command, values of other attributes of the same object can be used. The syntax is: *$(attr)*, where *attr* is the name of another attribute.

For example, if one wants to perform a man(1) function, one can draw a box; enter a line of text "title=tgif"; enter another line of text "launch=xterm -rw -e man $(title)"; select all three objects using ^a keyboard command; attach the text strings to the box using #a keyboard command; and launch the man(1) command by clicking the middle mouse button on the box (or the text strings) withe the <Shift> key held down. If one wants to be more fancy, the box can be replaced by an X11 pixmap object; the 'launch' attribute can be made invisible; and the 'title' attribute can be center justified and with its name hidden using the #m keyboard command.

By default, launching of an application is by default disabled in the *hyperspace* mode for security considerations (this can be overridden by the Tgif.AllowLaunchInHyperSpace X default setting). If a lunch command is encountered in the *hyperspace* mode, the command is displayed and the user is prompted to see if he/she wants to execute the command.

INTERNAL COMMANDS

Tgif provides the mechanism to execute internal commands. If the middle mouse button is clicked on an object with the <Shift> key held down (or double-clicking such an object), tgif looks for an attribute named *exec* (by default) of that object. If such an attribute is found, the value part of the attribute is interpreted as a list of internal commands (separated by semicolor) to execute. Same color rule applies as described in the TELEPORT/HYPERJUMP section above. A command usually takes the form:

<cmd_name> (<arg1>, <arg2>, ..., <argN>)

An argument of a command can be a string argument or a numeric argument. A string argument must be enclosed in double-quotes. A numeric argument can be a numerical value or a string of the form "$(*x*)", where *x* is the name of another attribute (this form is referred as the substitution form). A string argument can also contain substitution form. Please note that only one-level substitution are performed (the collection of internal commands should be viewed as a simple scripting language and *not* a declaration language).

When an attribute is referenced in an internal command, the attribute name can be in the form, <*obj_name*>.<*string*>, where <*obj_name*> must be in the form specified in the OBJECT NAMES section above and <*string*> contains only alphanumeric characters and the underscore ('_') character. If the first 2 characters of an attribute name is "!.", the rest of the attribute name names a file attribute. If the first 2 characters of an attribute name is "!*", the rest of the attribute name names an attribute of the currently selected object (if more than one object is selected, the top-most object in the stacking order is used).

The following internal commands are supported:

launch(<attr_name>)

The value of the attribute specified by <attr_name> is interpreted as a sh(1) command to execute. Please see the LAUNCH APPLICATIONS section above for more details.

exec(<attr_name>)
: The value of the attribute specified by <attr_name> is interpreted as an internal command to execute. This is similar to a subroutine call. Please note that the internal command is executed in the context of the top-level which contain the attribute.

mktemp(<str>,<attr_name>)
: This command makes a unique file name. The <str> argument is a template string, e.g., "/tmp/TgifXXXXXX", and it requires at least two "/" in it. The result of mktemp() is stored as the value of the attribute specified by <attr_name>. Please see the man pages of the C library function on mktemp(3C) for more details. (If tgif is compiled with the -D_USE_TMPFILE compiler option, then tempnam(3S) is used instead.)

create_file_using_simple_template(<template>,<output>,<str>,<attr_name>)
: The file specified by <template> is scanned for a line that matches <str>. When such a line is found, that line is replaced by the value of the attribute specified by <attr_name>. The result is put into the file specified as <output>.

update_eps_child(<eps_file_name>)
: This only works if the object being executed is a composite object. If the object has a component which is an imported EPS (Encapsulated PostScript) object, it is replaced by the EPS file specified by <eps_file_name>. If the object does not contain an EPS subobject, an EPS subobject is created.

update_xbm_child(<xbm_file_name>)
: This only works if the object being executed is a composite object. If the object has a component which is an imported XBM (X11 bitmap) object, it is replaced by the XBM file specified by <xbm_file_name>. If the object does not contain an XBM subobject, an XBM subobject is created.

update_xpm_child(<xpm_file_name>)
: This only works if the object being executed is a composite object. If the object has a component which is an imported XPM (X11 pixmap) object, it is replaced by the XPM file specified by <xpm_file_name>. If the object does not contain an XPM subobject, an XPM subobject is created.

delete_eps_child(<obj_name>)
: This only works if the object named <obj_name> is a composite object. If the object has a component which is an EPS (Encapsulated PostScript) object, it is deleted. If the object does not contain an EPS subobject, no operation is performed.

delete_xbm_child(<obj_name>)
: This only works if the object named <obj_name> is a composite object. If the object has a component which is an XPM (X11 pixmap) object, it is deleted. If the object does not contain an XPM subobject, no operation is performed.

delete_xpm_child(<obj_name>)
: This only works if the object named <obj_name> is a composite object. If the object has a component which is an XBM (X11 bitmap) object, it is deleted. If the object does not contain an XBM subobject, no operation is performed.

flip_deck(<times>,<frames_per_second>,<style>)
: This only works if the object being executed is a composite object and all subobjects of the composite object are X11 bitmap or X11 pixmap objects and have identical positions and sizes. The <times> argument specifies the number of times the deck is flipped. It can be a number or the string "infinite". The <frames_per_second> argument must be a number between 1 and 60. The <style> argument can be either "linear" or "ping_pong". When this command is being executed, any mouse button click or key click aborts command execution.

read_file_into_attr(<file_name>,<attr_name>)
: This command reads a file into an attribute. The <file_name> argument names a file, e.g., "/tmp/foo". The content of the file is read as the value of the attribute specified by <attr_name>.

If the file can not be opened for read, the attribute's value is set to an empty string.

write_attr_into_file(<attr_name>,<file_name>)
This command writes the value of an attribute into a file. The <file_name> argument names a file, e.g., "/tmp/foo". The value of the attribute specified by <attr_name> is written into <file_name>.

append_attr_into_file(<attr_name>,<file_name>)
This command appends the value of an attribute into a file. The <file_name> argument names a file, e.g., "/tmp/foo". The value of the attribute specified by <attr_name> is appended into <file_name>.

select_obj_by_name(<obj_name>)
This command silently (no highlighting handles) selects an object named <obj_name>. Please see the OBJECT NAMES section above for the specification of object names.

select_top_obj()
This command silently (no highlighting handles) selects the top object. This command fails if there is no object in the current page.

delete_selected_obj()
This command deletes all selected objects. This command fails if no object is selected.

unselect_all_obj()
This command de-selects all selected objects. If the select_obj_by_name() command is used, this command must be used eventually.

move_selected_obj_relative(<dx>,<dy>)
This command moves the selected object by <dx> absolute units in the x direction and <dy> absolute units in the y direction.

repeat(<cmd_attr_name>,<times>)
This command executes the internal command in the <cmd_attr_name> attribute <times> times.

hyperjump(<attr_name>)
This command teleports to the file name or URL name found in the <attr_name> attribute.

make_cgi_query(<dest_attr_name>,<url_name>,<list_attr_name>)
This command constructs an URL in the Common Gateway Interface (CGI) format in the <dest_attr_name> attribute. <url_name> names the CGI server script and <list_attr_name> names an attribute whose value are comma-separated attribute names. For example, if an object has the following attributes:

```
attr_list=last_name,first_name
last_name=Cheng
first_name=Bill
final_url=
exec=make_cgi_query(final_url,
   http://bourbon.cs.umd.edu:8001/cgi-bin/test-cgi,
   attr_list)
```

Executing this object will construct the following string in final_url:

```
http://bourbon.cs.umd.edu:8001/cgi-bin/test-cgi?last_name=Cheng&first_name=Bill
```

An subsequent hyperjump(final_url) command can be invoked to execute the corresponding "test-cgi" CGI server script with the last_name and first_name arguments.

For a detailed description of CGI scripts, the reader is referred to [2].

wait_click(<cursor_name>,<grab>,<attr_name>)
This command displays the <cursor_name> cursor and waits for the user to click a mouse button. If <cursor_name> is the string *NULL* (case-sensitive), the cursor will not change. If <Btn1> is clicked, the command terminates and 1 is placed in <attr_name>. If <Btn2> is clicked, 2 is placed in <attr_name>, etc. If <grab> set to *TRUE* (case-sensitive), then the mouse is grabbed by tgif.

Valid <cursor_name> can be found in <X11/cursorfont.h> (without the XC_ prefix).

sleep(<cursor_name>,<ms_interval>)
This command displays the <cursor_name> cursor and waits for <ms_interval> milliseconds to elapse. If <cursor_name> is the string *NULL* (case-sensitive), the cursor will not change. This command can be interrupted (and aborted) by any mouse clicks or key strokes. Valid <cursor_name> can be found in <X11/cursorfont.h> (without the XC_ prefix).

begin_animate()
This command is used to start an animation sequence. By default, tgif prepares for undo/redo. For a long animation sequence, the undo/redo records may take up a lot of memory. In this case, disable_undo() (described below) should be used before this command.

end_animate()
This command is used to terminate an animation sequence.

set_redraw(<true_or_false>)
This command is used to temporarily disable redraw if <true_or_false> is *FALSE* (case-sensitive). If a shuffle_obj_to_top() command is used before a move command, set_redraw(FALSE) and set_redraw(TRUE) should be used immediately before and immediately after, respectively, the shuffle_obj_to_top() command.

set_selected_obj_color(<color_str>)
This command changes the color of the selected object to <color_str>. If no object is selected, the current color will be changed to <color_str>.

set_selected_obj_fill(<fill_index>)
This command changes the fill pattern of the selected object to <fill_index>, which must be between 0 (for no fill) and 31. If no object is selected, the current fill pattern will be changed to <fill_index>.

set_selected_obj_pen(<pen_index>)
This command changes the pen of the selected object to <pen_index>, which must be between 0 (for no pen) and 31. If no object is selected, the current pen will be changed to <pen_index>.

set_selected_obj_line_width(<width>,<arrow_w>,<arrow_h>)
This command changes the line width, arrow width, and arrow height of the selected object to <width>, <arrow_w>, and <arrow_h>, respectively. If <arrow_w> or <arrow_h> is -1, the arrow width or arrow height, respectively, is not changed. If no object is selected, the current line width will be changed to the one that matches <width>, <arrow_w>, and <arrow_h> most closely. (Closeness is measured such that the difference is width is counted 10 times the diffference in arrow width and arrow height.)

set_selected_obj_spline(<spline_type>)
This command changes the spline type of the selected object to <spline_type>, which can be *straight*, *spline*, or *interpolated*. If no object is selected, the current spline type will be changed to <spline_type>.

set_selected_obj_arrow(<arrow_type>)
This command changes the arrow type of the selected object to <arrow_type>, which can be *none*, *right*, *left*, or *double*. If no object is selected, the current arrow type will be changed to <arrow_type>.

set_selected_obj_dash(<dash_index>)
This command changes the dash type of the selected object to <dash_index>, which must be between 0 (solid) and 8. If no object is selected, the current dash type will be changed to <dash_index>.

set_selected_obj_trans_pat(<trans_pat>)
This command changes selected object to have opaque pattern if <trans_pat> is 0; it changes selected object to have transparent pattern if <trans_pat> is any other numeric value. If no object is selected, the current fill and pen pattern will be opaque if <trans_pat> is 0 and will be

transparent if <trans_pat> is any other numeric value.

set_selected_obj_rcb_radius(<rcb_radius>)
This command changes the rcbox radius of the selected object to <rcb_radius>, which must be greater or equal to 4. If no object is selected, the current rcbox radius will be changed to <rcb_radius>.

set_selected_text_vspace(<vspace>)
This command changes the text vspace of the selected object to <vspace>. If no object is selected, the current text vspace will be changed to <vspace>.

set_selected_text_just(<justification>)
This command changes the text justification of the selected object to <justification>, which can be *left*, *center*, or *right*. If no object is selected, the current text justification will be changed to <justification>.

set_selected_text_font(<ps_font_name>)
This command changes the font and text style of the selected object to match <ps_font_name>. Examples of valid <ps_font_name> can be found when one selects CopyProperties() from the Properties Menu. The item listed under *text font* is a valid <ps_font_name>. If no object is selected, the current font and text style will be changed to match <ps_font_name>. This command fails if no match can be found,

set_selected_text_size(<size>)
This command changes the text size of the selected object to <size>. If <size> ends with the substring "pt", then point size is used instead of text size. If such as size cannot be found in the Size Menu, the closest size in the Size Menu will be used. If no object is selected, the current text size will be changed to <size> or the closest size.

set_selected_text_underline(<underline>)
This command removes text underline from the selected object if <underline> is 0; it underlines text in the selected object if <underline> is any other numeric value. If no object is selected, the current text underline will be changed accordingly.

inc(<attr_name>,<expr>)
This command increment <attr_name> by the expression <expr>. Both the value of <attr_name> and <expr> must be integers. Please see the ARITHMETIC EXPRESSIONS section below for details about expressions.

dec(<attr_name>,<expr>)
This command decrement <attr_name> by <expr>. Both the value of <attr_name> and <expr> must be integers.

shuffle_obj_to_top(<obj_name>)
This command move <obj_name> to the top. If <obj_name> is a subobject, it is raised to the top, relative to its siblings. This command is useful in animation where a selected frame (subobject) can be raised to the top.

disable_undo()
This command cleans up the undo/redo records and disable undo (and stop recording undo/redo information). The original history depth is saved away. This command should be used before a long animation sequence.

enable_undo()
This command restores the history depth saved away by the disable_undo() command and enables undo/redo. This command should be eventually used after disable_undo() is called.

get_drawing_area(<ltx_attr>,<lty_attr>,<rbx_attr>,<rby_attr>)
This command stores the absolute coordinate of the current drawing area in the specified attributes. <ltx_attr> stores the left-top X coordinate, <lty_attr> stores the left-top Y coordinate, <rbx_attr> stores the right-bottom X coordinate, and <rby_attr> stores the right-bottom Y coordinate.

get_selected_obj_bbox(<ltx_attr>,<lty_attr>,<rbx_attr>,<rby_attr>)
This command stores the absolute coordinate of the bounding box of the selected object in the specified attributes. <ltx_attr> stores the left-top X coordinate, <lty_attr> stores the left-top Y coordinate, <rbx_attr> stores the right-bottom X coordinate, and <rby_attr> stores the right-bottom Y coordinate. The bounding box is computed assuming that all lines are of width 0.

get_named_obj_bbox(<obj_name>,<ltx_attr>,<lty_attr>,<rbx_attr>,<rby_attr>)
This command stores the absolute coordinate of the bounding box of the object named <obj_name> in the specified attributes. <ltx_attr> stores the left-top X coordinate, <lty_attr> stores the left-top Y coordinate, <rbx_attr> stores the right-bottom X coordinate, and <rby_attr> stores the right-bottom Y coordinate. The bounding box is computed assuming that all lines are of width 0.

move_selected_obj_absolute(<ltx>,<lty>)
This command moves left-top corner of the selected object to (<ltx>,<lty>).

assign(<attr_name>,<expr>)
This command assigns <expr> to the attribute specified by <attr_name>. <expr> must be evaluated to a numeric value.

strcpy(<attr_name>,<string>)
This command copies <string> into the attribute specified by <attr_name>.

copy_string_to_cut_buffer(<string>)
This command copies <string> into the cut buffer.

strcat(<attr_name>,<string>)
This command appends <string> to the attribute specified by <attr_name>.

while(<expr>,<cmd_attr_name>)
This command keeps executing the internal command in <cmd_attr_name> until <expr> evaluates to 0.

if(<expr>,<then_cmd_attr_name>,<else_cmd_attr_name>)
If <expr> evaluates to 0, the internal command in <else_cmd_attr_name> is executed; otherwise, the internal command in <then_cmd_attr_name> is executed. <then_cmd_attr_name> or <else_cmd_attr_name> can be the string *NULL* (case-sensitive); in this case, no corresponding action is taken.

get_current_file(<attr_name>)
This command stores the full path name of the current file in <attr_name>.

get_current_export_file(<attr_name>)
This command stores the full path name of the output (print/export) file in <attr_name>.

get_current_dir(<attr_name>)
This command stores the current directory in <attr_name>.

getenv(<attr_name>,<env_var_name>)
This command stores the environment variable named <env_var_name> in <attr_name>.

strlen(<attr_name>,<string>)
This command assigns the number of characters in <string> to <attr_name>.

substr(<attr_name>,<string>,<start_index>,<length>)
This command copies <length> characters, starting from the character index <start_index>, of <string> into <attr_name>. The <start_index> is zero-based.

strstr(<attr_name>,<string>,<sub_string>)
This command finds the first occurrence of <sub_string> in <string> and copies <sub_string> and the rest of the string into <attr_name>.

strrstr(<attr_name>,<string>,<sub_string>)
This command finds the last occurrence of <sub_string> in <string> and copies <sub_string> and the rest of the string into <attr_name>.

unmake_selected_obj_iconic()
This command has the same effect as selecting UnMakeIconic() from the Special Menu except that at least one object must be selected already.

hyperjump_then_exec(<attr_name>,<attr_name_to_exec>)
This command teleports to the file name or URL name found in the <attr_name> attribute then executes the internal command specified by the <attr_name_to_exec> attribute in the new file.

show_attr(<attr_name>)
This command makes the <attr_name> attribute visible.

hide_attr(<attr_name>)
This command makes the <attr_name> attribute invisible.

show_attr_name(<attr_name>)
This command makes the name part of the <attr_name> attribute visible.

hide_attr_name(<attr_name>)
This command makes the name part of the <attr_name> attribute invisible.

show_value(<attr_value>)
This command makes the attribute whose name is empty and whose value is <attr_value> visible.

hide_value(<attr_value>)
This command makes the attribute whose name is empty and whose value is <attr_value> invisible.

get_attr_bbox(<ltx_attr>,<lty_attr>,<rbx_attr>,<rby_attr>,<attr_name>)
This command stores the absolute coordinate of the bounding box of the <attr_name> attribute in the specified attributes. <ltx_attr> stores the left-top X coordinate, <lty_attr> stores the left-top Y coordinate, <rbx_attr> stores the right-bottom X coordinate, and <rby_attr> stores the right-bottom Y coordinate. The bounding box is computed assuming that all lines are of width 0.

size_selected_obj_absolute(<abs_w>,<abs_h>)
This command stretches the right-bottom corner of the selected object so that its width becomes <abs_w> and height becomes <abs_h>.

size_named_obj_absolute(<obj_name>,<abs_w>,<abs_h>)
This command stretches the right-bottom corner of the object named <obj_name> so that its width becomes <abs_w> and height becomes <abs_h>.

message_box(<attr_name>,<msg>,<title>,<style>)
This command displays a messagebox with <title> as the title and <msg> as the message. <style> can be the string "info", "ync", "yn", or "stop". The messagebox display an OK button for the "info" or "stop" styles, YES/NO/CANCEL buttons for the "ync" style, YES/NO buttons for the "yn" style. When the user click a button in the messagebox, the name of the button will be placed in <attr_name>. If the user cancels the messagebox by typing the <ESC> key, <attr_name> will be set to the string "CANCEL". If <attr_name> is the string *NULL* (case-sensitive), the information about which button is clicked is not written anywhere. If <title> is the string *NULL*, *Tgif* will be the title for the messagebox.

get_user_input(<attr_name>,<msg1>,<msg2>)
This command displays a dialogbox with <msg1> in the first line and <msg2> in the second line. If <msg2> is the string "USE_CURRENT_DIR", the second line displays the current directory. The user can type in a line in the dialogbox which get placed in <attr_name>. If the user cancels the dialog by typing the <ESC> key, <attr_name> will be set to the empty string.

add_attr_to_selected_obj(<attr_name>,<attr_value>,<abs_x>,<abs_y>)
This command adds <attr_name>=<attr_value> to a selected object and place the attribute at (<abs_x>,<abs_y>). If <attr_name> is the string *NULL* (case-sensitive), the attribute's name will be the empty string. If <abs_x> and <abs_y> are both *NULL* (case-sensitive), the attribute will be placed below the lower left corner or the object. If <attr_name> starts with "!.", a *file attribute* will be added.

delete_attr_from_selected_obj(<attr_name>)
This command deletes an attribute named <attr_name> from a selected object. If <attr_name> starts with "!.", a *file attribute* will be deleted.

user_end_an_edge(<attr_name>,<abs_x>,<abs_y>)
This command starts a polyline/open-spline at (<abs_x>,<abs_y>), switches the drawing mode to the draw polyline/open-spline, and lets the user finishes the polyline/open-spline. If the endpoint falls in an object having an attribute *type=port*, that object's name will be placed in <attr_name>, if <attr_name> is not the string *NULL* (case-sensitive).

user_draw_an_edge(<start_attr_name>,<end_attr_name>)
This command switches the drawing mode to the draw polyline/open-spline and lets the user draw a polyline/open-spline. If the first endpoint falls in an object having an attribute *type=port*, that object's name will be placed in <start_attr_name>, if <attr_name> is not the string *NULL* (case-sensitive). If the last endpoint falls in an object having an attribute *type=port*, that object's name will be placed in <end_attr_name>, if <attr_name> is not the string *NULL* (case-sensitive).

get_a_poly_vertex_absolute(<x_attr_name>,<y_attr_name>,<obj_name>,<index>)
This command stores the absolute coordinate of the <index>*th* vertex of <obj_name> in attributes specified by <x_attr_name> and <y_attr_name>. The object specified by <obj_name> must be either a poly/open-spline or a polygon/closed-spline object.

move_a_poly_vertex_absolute(<obj_name>,<index>,<abs_x>,<abs_y>)
This command moves the <index>*th* vertex of <obj_name> to the absolute coordinate (<abs_x>,<abs_y>). The object specified by <obj_name> must be either a poly/open-spline or a polygon/closed-spline object.

post_attr_and_get_cgi_result(<url_attr>,<query_attr>,<result_attr>)
This command makes an HTTP request using the *POST* method. <url_attr> names the attribute that contains the URL (which usually names a CGI server script). <query_attr> names the attribute whose value is the data to be posted. <result_attr> names the attribute for receiving the results. For example, if an object has the following attributes:

```
url=http://bourbon.cs.umd.edu:8001/cgi-bin/echo-post
query=Hello World!
result=
exec=post_attr_and_get_cgi_result(url,query,result)
```

Executing this object will post "Hello World!" to the specified CGI script. In this case, the result of executing the script just echoes "Hello World!" back (along with some other bookkeeping information).

navigate_back()
This command performs the same operation as if the NavigateBack() is selected from the Navigate Menu.

stop() This command stops the execution of all internal commands.

sqrt(<attr_name>,<expr>)
This command assigns the square-root of <expr> to <attr_name>. <expr> must be evaluated to a non-negative numeric value.

random(<attr_name>)
This command assigns a random integer to <attr_name> using the C library function *rand()*. 0 is used as a seed for the random number generator.

srand48(<use_cur_time_as_seed>)
: This command seeds the random generator used by the C library function *drand48()*. If <use_cur_time_as_seed> is 0, 0 will be used as a seed. Otherwise, the current time will be used as a seed.

drand48(<attr_name>)
: This command assigns a floating pointer number between 0.0 and 1.0 to <attr_name> using the C library function *drand48()*.

round(<attr_name>,<expr>)
: This command assigns the round of <expr> to <attr_name>.

redraw_obj(<obj_name>)
: This command redraws the area occupied by <obj_name>.

redraw_drawing_area()
: This command redraws the whole drawing area (visible through the Canvas Window).

itox(<attr_name>,<digits>,<expr>)
: This command assigns <attr_name> to be the hex value of <expr>. <digits> (which must be between 1 and 8, inclusive) is the final width of the hex value (zeroes are added on the left).

for_i(<attr_name>,<min_val>,<max_val>,<increment>,<cmd_attr_name>)
: This command is the same as the following sequence of commands:

```
assign(<attr_name>,<min_val>);
while($(<attr_name>) <= <max_val>,loop)
```

where *loop* has the following value:

```
exec(<cmd_attr_name>);
inc(<attr_name>,<increment>)
```

Please note that <min_val>, <max_val>, and <increment> are only evaluated once prior the execution of this command.

set_file_not_modified()
: This command sets the file modified flag to false.

new_id(<attr_name>)
: This command generates an object ID, which is (unique in the current drawing, and stores it in <attr_name>.

rotate_selected_obj(<angle>)
: This command rotates the selected object by <angle> degrees. Positive angle is clockwise.

call_simple_shortcut(<shortcut_name>)
: This command calls a shortcut named <shortcut_name> which takes no arguments. Please see the SHORTCUTS section for a description of shortcuts.

call_one_arg_shortcut(<shortcut_name>,<arg>)
: This command calls a shortcut named <shortcut_name> that takes one argument and pass <arg> to it. Please see the SHORTCUTS section for a description of shortcuts.

substitute_attr(<attr_name>,<src_attr_name>,<replace_attr_name>,<pattern_str>)
: This command replaces occurrances of <pattern_str> in the value part of the attribute specified by <src_attr_name> by the value of the attribute specified by <replace_attr_name> and write the result into the attribute specified by <attr_name>.

get_file_size(<attr_name>,<file_name>)
: This command puts the size of file specified by <file_name> in the attribute specified by <attr_name>.

is_file(<attr_name>,<file_name>)
: This command puts a "1" in the attribute specified by <attr_name> if the file specified by <file_name> exists. It puts a "0" otherwise.

index(<attr_name>,<string>,<sub_string>)
: This command finds the first occurrence of <sub_string> in <string> and copies the zero-based index into <attr_name>.

rindex(<attr_name>,<string>,<sub_string>)
: This command finds the last occurrence of <sub_string> in <string> and copies the zero-based index into <attr_name>.

get_number_of_lines_in_attr(<result_attr>,<attr_name>)
: This command counts the number of lines in the attribute specified by <attr_name> and writes the count into <result_attr>.

get_line_in_attr(<result_attr>,<attr_name>,<line_number>)
: This command copies the *n*th line of the attribute specified by <attr_name> into <result_attr>, where *n* is a zero-based index specified by <line_number>.

trim(<attr_name>)
: This command removes leading and trailing blank characters from the attribute specified by <attr_name>.

is_attr(<result_attr>,<attr_name>)
: This command writes a "1" into <result_attr> if the attribute specified by <attr_name> exists. It writes a "0" into <result_attr> otherwise.

find_obj_names(<result_attr>,<obj_name>,<attr_name_value>)
: This command finds all objects that are direct sub-objects of the object specified by <obj_name> and writes their names into <result_attr>. If <obj_name> is an empty string, all top-level objects are scanned.

  <attr_name_value> specifies a filter for the objects. If <attr_name_value> is the empty string, all qualifying objects are selected. If <attr_name_value> is of the form "<string>=*", an object is selected if it has an attribute named <string>. If <attr_name_value> is of the form "<string>=<value>", an object is selected if it has an attribute named <string> and its corresponding value is <value>. If <attr_name_value> does not contain the '=' character, an object is selected if it has an attribute whose name is empty and the corresponding value is identical to <attr_name_value>.

  If *n* objects are matched, the attribute specified by <result_attr> is updated with *n+1* lines. The value of the zeroth line becomes *n* and the object names becomes lines 1 through *n* of <result_attr>. The get_line_in_attr() internal command can be used to retrieve the object names.

tokenize(<result_attr>,<string>,<separator>)
: This command breaks <string> into tokens which are separated by the <separator> character and writes the tokens (in the same fashion as in the find_obj_names command above) into <result_attr>. <separator> must be a string of length of 1 and it must not be the space character, the single-quote character, or the double-quote character. If a token contains the separator character, the token can be surrounded by a pair of single-quotes or double-quotes which are automatically removed when this command is executed.

  If *n* tokens are found, the attribute specified by <result_attr> is updated with *n+1* lines. The value of the zeroth line becomes *n* and the tokens becomes lines 1 through *n* of <result_attr>. The get_line_in_attr() internal command can be used to retrieve the tokens.

move_attr_relative(<attr_name>,<dx>,<dy>)
: This command moves the attribute whose name is <attr_name> by <dx> absolute units in the x direction and <dy> absolute units in the y direction.

get_number_of_vertices(<result_attr>,<obj_name>)
: This command copies the number of vertices of the object specified by <obj_name> into <result_attr>. The specified object must be a polyline (open-spline) or a polygon (closed-spline).

is_obj_transformed(<result_attr>,<obj_name>)
: This command writes a "1" into <result_attr> if the object specified by <obj_name> is transformed (rotated or sheared). It writes a "0" into <result_attr> otherwise.

make_selected_obj_iconic(<sym_path>)
: This command works like the MakeIconic() command from the Special Menu, except that the user is not prompted for the name of the icon. Instead, <sym_path> is used to specify the full path name of the icon.

get_tgif_version(<major_attr,minor_attr,patchlevel_attr,build_attr>)
: This command writes tgif's major version number, minor version number, patchlevel, and build information into <major_attr>, <minor_attr>, <patchlevel_attr> and <build_attr>, respectively. If an argument is the string *NULL* (case-sensitive), that information is skipped.

get_tgif_dir(<result_attr>)
: This command writes "$HOME/.Tgif" into <result_attr> where $HOME is the home directory of the user.

get_profile_string(<result_attr>,<section>,<key>,<def_value>,<ini_path>)
: This command gets the value associated with the key specified by <key> from the section specified by <section> in the file specified by the full path <ini_path> and stores it into the attribute specified by <result_attr>. If there is not value associated with the specified key, <def_value> is stored into <result_attr>. If <key> is an empty string, all the key names in <section> of <ini_path> will be written (in the same fashion as in the find_obj_names command above) into <result_attr>. If <section> is an empty string, all the section names in <ini_path> will be written (in the same fashion as in the find_obj_names command above) into <result_attr>.

write_profile_string(<section>,<key>,<value>,<ini_path>)
: This command sets the value associated with the key specified by <key> of the section specified by <section> in the file specified by the full path <ini_path> to be <value>. If <key> is an empty string, all key/value pairs in <section> of <ini_path> will be cleared. <section> should not be an empty string.

select_additional_obj(<obj_name>)
: This command silently (no highlighting handles) selects an additional object named <obj_name>. Please see the OBJECT NAMES section above for the specification of object names.

open_file(<file_number>,<file_name>,<file_mode>)
: This command opens the file specified by <file_name> in the mode specified by <file_mode> and assigns the opened file a file reference number of <file_number>. <file_number> must be 0 or between 3 and 15. Opening file 0 *rewinds* the standard input. Examples of modes are "r" for reading, "w" for writing, and "a" for appending. A file is always opened in text (non-binary) mode.

close_file(<file_number>)
: This command closes the file associated with file reference number <file_number>. <file_number> must be 0 or between 3 and 15.

read_file(<file_number>,<result_attr>)
: This command reads a line from the file associated with file reference number <file_number> and put the line in the attribute specified by <result_attr>. <file_number> must be between 0 (for standard input) or between 3 and 15.

write_file(<file_number>,<string>)
: This command writes <string> to the file associated with file reference number <file_number>. <file_number> must be between 1 and 15. Numbers 1 and 2 are for standard output and standard error files.

flush_file(<file_number>)
This command flushes the file associated with file reference number <file_number>. <file_number> must be between 1 and 15. Numbers 1 and 2 are for standard output and standard error files.

append_file(<dest_file_name>,<src_file_name>)
This command appends the file specified by <src_file_name> to the file specified by <dest_file_name>.

set_output_format(<format>,<color_output>)
This command sets the output format to <format>. If <color_output> is 0, black and white output (printing) mode will be used; otherwise, color output (printing) mode will be used. Please see the Tgif.WhereToPrint X default for a list of possible formats.

set_export_clip_rect(<ltx>,<lty>,<rbx>,<rby>)
This command sets the export clipping rectangle to be a rectangular region with left-top corner at (<ltx>,<lty>) and right-bottom corner at (<rbx>,<rby>). <ltx> must be strickly less than <rbx> and <lty> must be strickly less than <rby>.

import_file(<file_name>,<format>,<ltx>,<lty>)
This command imports the file specified by <file_name> and place it at (<ltx>,<lty>). The file is expected to be in the format specified by <format>, which can be "XBM", "XPM", "GIF", "PNG", "JPEG", and names specified by the Tgif.ImportFilter# X defaults.

set_xpm_output_version(<version_number>)
This command sets the XPM version number when outputting in the X11 pixmap format to be <version_number>. <version_number> can take on values 1 or 3.

edit_ini_section(<attr_name>,<title>,<section>,<ini_path>)
This command brings up a dialogbox to edit the section specified by <section> in the file specified by the full path <ini_path>. If the user press the OK button in the dialogbox, the section is cleared and the content of the dialogbox is written back into the file, and "OK" is placed in the attribute specified by <attr_name>. If the user press the CANCEL button in the dialogbox, the file is unmodified, and "CANCEL" is placed in the attribute specified by <attr_name>.

select_from_ini_section(<attr_name>,<title>,<section>,<ini_path>)
This command brings up a list to select an entry from the section specified by <section> in the file specified by the full path <ini_path>. If nothing is selected, the attribute specified by <attr_name> will be cleared. Otherwise, the selected entry will be written into the attribute specified by <attr_name>.

append_line_into_attr(<attr_name>,<string>)
This command appends the line specified by <string> to the attribute specified by <attr_name>.

insert_line_into_attr(<attr_name>,<string>,<line_number>)
This command inserts the line specified by <string> as the *n*th line of the attribute specified by <attr_name>, where *n* is a zero-based index specified by <line_number>. *n* must be at least 1. If *n* is larger the number of lines in the attribute, blank lines are automatically inserted.

clear_attr(<attr_name>)
This command clears the attribute value of the attribute specified by <attr_name> and deletes all other lines of the attribute if the attribute contains multiple lines.

create_text_obj(<abs_x>,<abs_baseline_y>,<string>)
This command creates a text object at the location (<abs_x>,<abs_baseline_y>) with the text specified by <string>.

create_box_obj(<abs_ltx>,<abs_lty>,<abs_rbx>,<abs_rby>)
This command creates a rectangle defined by (<abs_ltx>,<abs_lty>) and (<abs_rbx>,<abs_rby>).

create_corner_oval_obj(<abs_ltx>,<abs_lty>,<abs_rbx>,<abs_rby>)
This command creates a corner oval defined by (<abs_ltx>,<abs_lty>) and (<abs_rbx>,<abs_rby>).

create_center_oval_obj(<abs_x>,<abs_y>,<radius>)
This command creates a center oval centered at (<abs_x>,<abs_y>) with radius specified by <radius>.

create_edge_oval_obj(<abs_ltx>,<abs_lty>,<abs_rbx>,<abs_rby>)
This command creates an edge circle defined by (<abs_ltx>,<abs_lty>) and (<abs_rbx>,<abs_rby>).

create_rcbox_obj(<abs_ltx>,<abs_lty>,<abs_rbx>,<abs_rby>)
This command creates a rounded-corner rectangle defined by (<abs_ltx>,<abs_lty>) and (<abs_rbx>,<abs_rby>).

create_arc_obj(<abs_x>,<abs_y>,<radius>,<dir>,<angle1>,<angle2>)
This command creates an arc centered at (<abs_x>,<abs_y>) with radius, direction, start angle, and end angle specified by <radius>, <dir>, <angle1>, and <angle2>, respectively. The <radius>, <dir>, <angle1>, and <angle2> are specified in the same way as they are specified in the SpecifyAnArc() command under the CreateObject submenu of the Edit Menu. <dir> can be "+" or "-" where "+" is clockwise. <angle1> and <angle2> are in degrees with 0 degree at the 12 o'clock position.

create_first_vertex(<abs_x>,<abs_y>)
This command is used in conjunction with the create_next_vertex() and create_poly_obj() commands to create a polyline/open-spline object. It can also be used in conjunction with the create_next_vertex() and create_polygon_obj() commands to create a polygon/closed-spline object. This command sets the starting point of the polyline/open-spline object or the polygon/closed-spline object to be at (<abs_x>,<abs_y>).

create_next_vertex(<abs_x>,<abs_y>)
This command is used in conjunction with the create_first_vertex() and create_poly_obj() commands to create a polyline/open-spline object. It can also be used in conjunction with the create_first_vertex() and create_polygon_obj() commands to create a polygon/closed-spline object. This command sets the next vertex of the polyline/open-spline object or the polygon/closed-spline object to be at (<abs_x>,<abs_y>).

create_poly_obj()
This command is used in conjunction with the create_first_vertex() and create_next_vertex() commands to create a polyline/open-spline object.

create_polygon_obj()
This command is used in conjunction with the create_first_vertex() and create_next_vertex() commands to create a polygon/closed-spline object.

start_create_group_obj()
This command is used in conjunction with the create_group_obj() command to create a grouped object. This command marks the beginning of the group.

create_group_obj()
This command is used in conjunction with the start_create_group_obj() command to create a grouped object. This command groups all objects created since the last start_create_group_obj() call into a grouped object.

set_allow_interrupt(<true_or_false>)
If <true_or_false> is *FALSE* (case-sensitive), this command is used to temporarily disable an user interrupt when tgif is executing internal commands. If a user interrupt is received when interrupt is disabled, it will be queued and will interrupt the execution of internal commands when set_allow_interrupt() is called again with <true_or_false> being *TRUE* (case-sensitive).

select_each_obj_and_exec(<attr_name_to_exec>)
This command first unselects any object that is selected. It then selects each object in the current drawing in turn and executes the internal command specified by the <attr_name_to_exec> attribute. If this command is executed as a result of a mouse click over an object, only objects in

the current page will be scanned for execution. If this command is executed from a script file, objects in every page will be scanned for execution.

edit_attr_in_text_mode(<attr_name>)
When this command is executed, tgif enters the text drawing mode and edits the attribute specified by <attr_name>.

ARITHMETIC EXPRESSIONS

Certain internal commands allow arithmetic expressions as arguments. Infix notation is used. Supported operators (and their precedences) are listed below.

?	1	if-then-else, e.g. <rel> ? <iftrue> : <else>
:	2	if-then-else, e.g. <rel> ? <iftrue> : <else>
||	3	logical OR
&&	4	logical AND
|	5	bit-wise OR
^	5	bit-wise XOR
&	5	bit-wise AND
==	6	equal
!=	6	not-equal
>	7	greater than
<	7	less than
>=	7	greater than or equal to
<=	7	less than or equal to
<<	8	shift left
>>	8	shift right
+	9	add
-	9	subtract
*	10	multiple
/	10	divide
//	10	integer divide
%	10	mod
!	11	logical NOT
~	11	bit-wise invert/NOT
)	12	closed parenthesis
(	13	open parenthesis

GENERATING IMAGEMAP FILES

This section describes how to generate NCSA imagemap and CERN clickable image files. The Tgif.ImageMapFileFormat X default decides whether to generate a NCSA imagemap or a CERN clickable image file. Since the two formats are very similar, we will only discuss how to generate NCSA imagemap files. For more information about NCSA imagemap, please see [3]. For more information about CERN clickable image, please see [4].

The Tgif.GenerateImageMap X default should be set to ‘‘true’’ to enable the imagemap generation. When printing in the GIF format (see the BASIC FUNCTIONALITIES section about printing), an XPM file (which will be removed at the end of this process) is generated first. (The value specified by the Tgif.InitExportPixelTrim X default is used to trim extra pixels. Using these values forms an escape mechanism to fix an idiosyncrasy that tgif can not figure out exactly how big the whole image is.)

The XPM version is specified by the Tgif.XPmOutputVersion X default unless the Tgif.UseXPmVersion1ForImageMap X default is set to ‘‘true’’, which forces the XPM1 format. Then the command specified by the Tgif.XpmToGif X default is executed to convert the XPM file into a GIF (Generic Interchange Format) file which can be used by software such as NCSA’s Mosaic(1). The file extension for the GIF file is specified by the Tgif.GifFileExtension X default. Together with the GIF file, an imagemap file with file extension specified by the Tgif.ImageMapFileExtension X default is generated. The content of the imagemap is generated as follows.

Tgif first looks for a *file attribute* with attribute name *href*. The value of the attribute is written as the

default URL. If such a file attribute can not be found, imagemap generation is aborted. If it is found, then all objects in the file are scanned. For an object having an attribute named *href*, the value of the attribute is written as the *URL* for a *method* line in the imagemap. If the object is neither a circle nor a poly/polygon, the *rectangle* method is used.

Similar mechanism is used when printing in the HTML format, except that a generic HTML file is generated with an imagemap in the Spy Glass Client-side Imagemap format. You can generate a custom HTML file if you specify an HTML export template using SetHTMLExportTemplate() from the File Menu. Details about the template file is described below.

HTML EXPORT TEMPLATE

If an HTML export template file is specified with the SetHTMLExportTemplate() from the File Menu, custom HTML files can be generated when printing in the HTML format. The customization is done through the use of variables embedded in the HTML export template file. These variables have the syntax of an HTML character entity. They all starts with "&tgv" and ends with ";". They are:

&tgvfilename;
: This variable will be replaced by the name of the file (without file extension).

&tgvcurnum;
: This variable will be replaced by current page number.

&tgvfirstnum;
: This variable will be replaced by the first page number (usually 1).

&tgvlastnum;
: This variable will be replaced by last page number.

&tgvprevnum;
: This variable will be replaced by the previous page number (with wrap around).

&tgvprevnumnowrap;
: This variable will be replaced by the previous page number (with no wrap around).

&tgvnextnum;
: This variable will be replaced by the next page number (with wrap around).

&tgvnextnumnowrap;
: This variable will be replaced by the next page number (with no wrap around).

&tgvtitle;
: This variable will be replaced by the title the page or of the file.

&tgvmapobjs;
: This variable will be replaced by the objects (specified as <AREA> tabs) in a client-side image map.

For example, if a template specifies:

```
<IMG SRC="&tgvfilename;-&tgvcurnum;.gif"
        USEMAP="#p0">
<MAP NAME="p0">
&tgvmapobjs;
<AREA SHAPE="RECT"
        COORDS="0,0,&tgvmapwidth;,&tgvmapheight;"
        HREF="&tgvfilename;-&tgvnextnum;.html">
</MAP>
```

Exporting using PrintOneFilePerPage() with this template may get (for page 2 of a file name "foo.obj" with 5 pages):

```
<IMG SRC="foo-2.gif"
        USEMAP="#p0">
<MAP NAME="p0">
```

```
<AREA SHAPE="RECT" ...>
 ...
<AREA SHAPE="RECT" ...>
<AREA SHAPE="RECT"
        COORDS="0,0,145,97"
        HREF="foo-3.html">
</MAP>
```

GENERATING MICROSOFT WINDOWS EPSI FILES

Some Microsoft Windows (TM) applications do not understand standard PostScript %%BeginPreview, %%EndImage, and %%EndPreview comments. This section describes how to generate an EPSI file which is understood by them. This feature is invoked when the current print format is TiffEPSI. In this case, the generated EPSI file will contain 30 bytes of binary information in the beginning of the file and a TIFF image (also binary) at the end of the file. This file also will not contain the %%BeginPreview, %%EndImage, and %%EndPreview comments. A file in this format is normally not considered to be a PostScript file except under Windows.

When this feature is enabled, tgif generates a normal EPSI file first, then dump the current content of the file into an X11 bitmap file. The command specified in Tgif.XbmToTiff is executed to generate a TIFF image which is then append at the end of the EPSI file.

LOCKING OBJECTS

Objects can be locked and unlocked using #< and #> keyboard commands. When a selected object is locked, it is shown with gray handles. A locked object can be moved, stretched, flipped, or rotated; however, its properties, such as fill pattern, width, etc., can be changed. Locked objects can also be deleted. When a locked object is grouped with other objects, the resulting grouped object is also locked. A locked object can be used as an anchor to align other objects; however, DistributeObjs() command will fail if any objects are locked. Locked objects do not participate in any operations in the select vertex mode.

UNDO/REDO

Most operations can be undone and redone. The Tgif.HistoryDepth X default controls the size of the undo buffer. If it is set to -1, then the undo buffer's size is infinite. The undo buffer is flushed when the New() or Open() commands are executed (from the File Menu), when the FlushUndoBuffer() command is executed from the Edit Menu, or when Pop() is executed from a .sym file. If a private colormap is used (automatically done when new colors can not be allocated from the default colormap), executing FlushUndoBuffer() will attempt to reset the colormap (if the -DDONT_FREE_COLORMAP compile option is not used).

DOMAINS

A *domain* is a collection of library symbols suitable for instantiations. A library is implemented as a directory of .sym files, and therefore, a domain is implemented as a search path. If there are symbols with the same file name which reside in different directories specified in the search path, then the one closer to the front of the search path will be made available for the user to instantiate.

The number of domains is specified by the MaxDomains X default, and the names of the domains are specified by the DomainPath# X default. The library search paths are specified by csh environment variables. See the section on X DEFAULTS for more details.

Domain information can also be loaded into the ~/.Tgif/domain.ini file by setting Tgif.DomainInIni to true and selecting Reload Domain Info From X from the Domain submenu of the File Menu.

SELECTING A NAME FROM A POPUP WINDOW

When selecting a file name, a symbol name, or a domain name, tgif pops up a window with appropriate names for the user to choose from. The user can use mouse clicks to select an entry. Key strokes can also be used to specify the desired name; however, tgif attempts to match the key strokes with names in the selection on the fly. If a match can not be found, the key strokes are ignored. ^n, ^j, or the DownArrow key advances the selection down by 1 entry; ^p, ^k, or the UpArrow key advances the selection up by 1 entry. ^f, ^d, or the DownArrow key with <Control> key held down advances the selection down by 10 entries; ^b, ^u, or the UpArrow key with <Control> key held down advances the selection up by 10 entries. '$' will select the last entry, while '^' will select the first entry. ^w or ^y un-select the selected entry. If the selected

entry is a directory, hitting <CR> will change directory; if not, hitting <CR> finishes the selection process and the selected entry is returned.

In selecting file names to open or import, typing '/' is interpreted as going to the root directory or specifying an URL. At this point, the automatic matching of key strokes is temporarily disabled until either a <TAB> or a <CR> is pressed. Also, clicking the middle mouse button in the file name area pastes from the clipboard.

The automatic appending of *index.obj* or *.obj* (introduced in version 2.16) is obsoleted and an URL is never modified.

The current selection is displayed near the top of the popup window. Back-space should be used with caution because it might change the current directory to the parent directory.

IMPORTING EPS FILES

Encapsulated PostScript (EPS) files can be imported using the #(keyboard command. If the EPS file has a preview bitmap (can be generated using the *pstoepsi* tool), tgif will display it (HideBit/Pixmap() from the Layout Menu can be used to disable the displaying of bitmap/pixmaps). When the EPS object is saved in a .obj or .sym file, neither the preview bitmap, nor the PostScript content of the EPS file is saved. Therefore, when printing such a file (either from tgif or using prtgif), the EPS file must be present at the same place from which it was originally imported.

ADDITIONAL FONTS

In addition to the Times, Courier, Helvetica, NewCentury, and Symbol fonts, additional fonts can be specified using the Tgif.AdditionalFonts X default. (The default screen fonts can also be replaced, please see Tgif.HasAlternateDefaultFonts in the X DEFAULTS section for more details.) Each additional font requires *4* parts, one for each font style (in the order of Roman, Bold, Italic, and BoldItalic). Each part contains *3* strings. The first string specifies the family, weight, slant, and width of the font (please see the man pages for xfontsel(1) for more details; also there is a second form which is described below). The second string specifies the registry and encoding of the font (see xfontsel(1) again). (One can use xlsfonts(1) to see what fonts are available and pick out the just mentioned two strings from the output.) The third string specifies the PostScript font name.

For example, if one wants to use the X Lucida font to represent the PostScript ZapfChancery-MediumItalic font, one can set Tgif.AdditionalFonts as follows:

```
Tgif.AdditionalFonts: \n\
    lucida-medium-r-normal \n\
    iso8859-1 \n\
    ZapfChancery-MediumItalic \n\
    \n\
    lucida-demibold-r-normal \n\
    iso8859-1 \n\
    ZapfChancery-MediumItalic \n\
    \n\
    lucida-medium-i-normal \n\
    iso8859-1 \n\
    ZapfChancery-MediumItalic \n\
    \n\
    lucida-demibold-i-normal \n\
    iso8859-1 \n\
    ZapfChancery-MediumItalic
```

The above maps all four font styles of the Lucida font to the ZapfChancery-MediumItalic font (similar to how Symbol font is handled).

The first string can also be specified in a second form which is identified by having "%d" as part of the string. For example, one can use "lucidasans-%d" as the first string. In this case, the actual X font used will be the specified string with "%d" replaced by the font size. The encoding string (second string) is

ignored (but must be present). The font name prefix (please see Tgif.FontNamePrefix entry in the X DEFAULTS section) is also ignored.

Sometimes, different encodings of the same PS font is needed. This can be accomplished with the Tgif.PSFontAliases and Tgif.AdditionalDontReencode X defaults. For example, if one wants to use the adobe-fontspecific encoding of the Utopia font (instead of ISO-Latin-1), one can use the following X defaults:

```
Tgif.AdditionalFonts: \n\
    utopia-medium-r-normal \n\
    adobe-fontspecific \n\
    UtopiaFS-Regular \n\
    \n\
    utopia-bold-r-normal \n\
    adobe-fontspecific \n\
    UtopiaFS-Bold \n\
    \n\
    utopia-medium-i-normal \n\
    adobe-fontspecific \n\
    UtopiaFS-Italic \n\
    \n\
    utopia-bold-i-normal \n\
    adobe-fontspecific \n\
    UtopiaFS-BoldItalic
Tgif.PSFontAliases: \n\
    UtopiaFS-Regular=Utopia-Regular \n\
    UtopiaFS-Bold=Utopia-Bold \n\
    UtopiaFS-Italic=Utopia-Italic \n\
    UtopiaFS-BoldItalic=Utopia-BoldItalic
Tgif.AdditionalDontReencode: UtopiaFS
```

In the above example, a fake PS font name prefix of UtopiaFS is used to represent the adobe-fontspecific encoding of the Utopia font. Tgif.PSFontAliases maps the fake PS font names to their real PS font names and Tgif.AdditionalDontReencode tells tgif that do not use ISO-Latin-1 reencodings for the specified font (or list of fonts).

SQUARE DOUBLE BYTE FONTS

Starting with version 4.0 of tgif, double-byte fonts are supported. But only double-fonts where every character has the same width and height are supported. Double-byte fonts is specified using the Tgif.SquareDoubleByteFonts X default. The format of this X default is similar to that of the Tgif.AdditionalFonts X default described in the ADDITIONAL FONTS section above with differences described here. Each double-byte font requires *4* parts, one for each font style (in the order of Roman, Bold, Italic, and BoldItalic). Each part contains *3* strings. The first string specifies the name of the font. It must contain a "%d" as part of the string. The actual X font used will be the specified string with "%d" replaced by the font size. The second string can be either "*", "H", or "V". When it is the "V" string, each character is rotated 90 degrees counter-clockwise. Otherwise, the characters are not rotated. The third string specifies the PostScript font name.

Using input methods (specified by the Tgif.DoubleByteInputMethod X default) one can mix english (single-byte) substrings within a double-byte string. The font to use for the english substring is specified by the Tgif.DefaultSingleByteFont X default.

For example, if one wants to use the X Song Ti font to represent PostScript GB-Song-Regular font, one can set Tgif.SquareDoubleByteFonts as follows:

```
Tgif.DefaultSingleByteFont: Helvetica
Tgif.GBShowFontChar: 271372
Tgif.SquareDoubleByteFonts: \n\
```

```
-isas-song ti-*-*-*--%d-*-*-*-*-*-gb2312.1980-0 \n\
        * \n\
        GB-Song-Regular \n\
\n\
-isas-song ti-*-*-*--%d-*-*-*-*-*-gb2312.1980-0 \n\
        * \n\
        GB-Song-Regular \n\
\n\
-isas-song ti-*-*-*--%d-*-*-*-*-*-gb2312.1980-0 \n\
        * \n\
        GB-Song-Regular \n\
\n\
-isas-song ti-*-*-*--%d-*-*-*-*-*-gb2312.1980-0 \n\
        * \n\
        GB-Song-Regular
```

In the above example, the Song Ti font doesn't have styles such as italic and bold, so all four parts are identical. The Tgif.GBShowFontChar X default specifies a double-byte octal character to be used to represent this font in the Choice Window when this font is selected.

Below is another example of using the X JIS fonts to represent PostScript Ryumin-Light-EUC-H and Ryumin-Light-EUC-V fonts: as follows:

```
Tgif.RyuminShowFontChar: 244242
Tgif.SquareDoubleByteFonts: \n\
    -jis-fixed-*-*-*--%d-*-*-*-*-*-jisx0208.1983-* \n\
        H \n\
        Ryumin-Light-EUC-H \n\
    -jis-fixed-*-*-*--%d-*-*-*-*-*-jisx0208.1983-* \n\
        H \n\
        Ryumin-Light-EUC-H \n\
    -jis-fixed-*-*-*--%d-*-*-*-*-*-jisx0208.1983-* \n\
        H \n\
        Ryumin-Light-EUC-H \n\
    -jis-fixed-*-*-*--%d-*-*-*-*-*-jisx0208.1983-* \n\
        H \n\
        Ryumin-Light-EUC-H \n\
    \n\
    -jis-fixed-*-*-*--%d-*-*-*-*-*-jisx0208.1983-* \n\
        V \n\
        Ryumin-Light-EUC-V \n\
    -jis-fixed-*-*-*--%d-*-*-*-*-*-jisx0208.1983-* \n\
        V \n\
        Ryumin-Light-EUC-V \n\
    -jis-fixed-*-*-*--%d-*-*-*-*-*-jisx0208.1983-* \n\
        V \n\
        Ryumin-Light-EUC-V \n\
    -jis-fixed-*-*-*--%d-*-*-*-*-*-jisx0208.1983-* \n\
        V \n\
        Ryumin-Light-EUC-V
```

MULTIPAGE DRAWING

An object file can contain multiple pages. Two layout modes, *stacked* and *tiled*, for a multipage drawing are supported. In *stacked* layout mode, pages are considered to be stacked on top of each other, and therefore, an object can only appear on one page. In *tiled* layout mode, pages are tiled to form a large logical page; in this case, an object can exist on several physical pages simultaneously. Swiching between the two modes are considered rare events and can not be undone. Tgif does not allow switching from the tiled

layout mode to the stacked mode when there exists an object that spans physical page boundaries because it can not decide which physical page the object belongs.

Page numbers are supported through the use of page numbering objects. A page number objecting is an object that contains an attribute whose name is *!PAGE_NUM* (the name is case-sensitive) and the name part of that attribute is not shown (hiding an attribute name can be achieved by using the Move/JustifyAttr() command under the Attribute submenu of the Special Menu). The value of the attribute determines how the page number is printed. If the value of the attribute contains a *!(STACKED_PAGE_NUM)* substring, that part of the substring will be replaced by the page number if the page layout mode is *stacked*. If the page layout mode is tiled, the string will be printed out as is. If the value of the attribute contains a *!(STACKED_NUM_PAGES)* substring, that part of the substring will be replaced by the number of pages if the page layout mode is *stacked*. If the value of the attribute contains a *!(TILED_PAGE_ROW)* or *!(TILED_PAGE_COL)* substring, that part of the substring will be replaced by the row number or the column number of the physical page if the page layout mode is *tiled*.

SPECIAL ATTRIBUTES

There are a few special attributes that tgif recognized. There are described in this section. They are all case-sensitive.

!PAGE_NUM=<page_number>

This specifies the page numbers in a multipage drawing. Please see the MULTIPAGE DRAWING section for details.

auto_center_attr

If an attribute's name is empty and the value is *auto_center_attr*, then all the visible attributes of the owner object will automatically be centered relative to the bounding box of the owner object. It doesn't really make sense to have multiple visible attributes because they will overlap. This attribute is useful for making simple flowchart elements.

unmakeiconic_on_instantiate

If a symbol object's attribute has an empty attribute name and the value is *unmakeiconic_on_instantiate*, then when the symbol is instantiated, the following commands are performed on the just-instantiated icon object: 1) UnMakeIconic() command from the Special Menu, 2) UnGroup() command from the Arrange Menu, and 3) the "unmakeiconic_on_instantiate" text object is removed. This attribute is useful for making simple flowchart segments.

unmakeiconic_on_instantiate_delete_attrs

If a symbol object's attribute has an empty attribute name and the value is *unmakeiconic_on_instantiate_delete_attrs*, then when the symbol is instantiated, the following commands are performed on the just-instantiated icon object: 1) UnMakeIconic() command from the Special Menu, 2) delete all attributes from this object, and 3) UnGroup() command from the Arrange Menu. This attribute is useful for putting a group of "useful" objects into a symbol object.

retracted_arrows

If an attribute's name is empty and the value is *retracted_arrows* for a polyline or open-spline object with more than 2 vertices, then the arrows of the spline object is retracted by one vertex.

auto_retracted_arrows

This is very similar to the *retracted_arrows* above except that the object must be an interpolated open-spline with only one arrow head. The spline object is forced to have 3 vertices and the middle vertex of the spline object is automatically adjusted when an endpoint is moved.

auto_exec=<internal_command>

If such a file attribute exists, the value is executed when the file is opened (unless the file is opened as a result of executing the hyperjump_then_exec() internal command).

edit_attrs_in_context_menu=...

If an object has an attribute named *edit_attrs_in_context_menu*, the values (starting from the 2nd line and separated by linebreaks) of this attribute are treated as attribute names. The named attributes will be visible in the Edit Attribute In Editor submenu of the Context Menu. For

example, if an object has the following attributes:

```
edit_attrs_in_context_menu=
        x
        y
        z
w=greetings
x=hello
y=world
z=how are you
good-bye
```

the Edit Attribute In Editor submenu of the Context Menu will only show "x=hello", "y=world", and "z=how are you".

EXPORT TO TABLE

When the ExportToTable() command is selected from the Table submenu of the Special Menu, certain attributes of selected objects are written into a user-specified output file in a format which can be easily imported by a spreadsheet program or to be used by the MergeWithTable() command described in the next section. An output file contains columns of strings. Two columns are separated by a single <TAB> character. The first row of a output file contains the column names and all other rows contain values.

The names of the attributes to be written are specified by the *file attribute* named *TABLE_ATTRS* (which is denoted by *!.TABLE_ATTRS* here). The value of the TABLE_ATTRS file attribute is a list of comma-separated attribute names. When ExportToTable() command is executed, the attribute names specified by !.TABLE_ATTRS are written to the output file first. Then, for each selected object, every one of its attribute which appears in the list specified by !.TABLE_ATTRS are written to the output file in one line. If an object has no attributes that match the specification, no corresponding line is generated.

MERGE WITH TABLE

When the MergeWithTable() command is selected from the Table submenu of the Special Menu, a selected object is *merged* (also known as *mail-merged* on PCs) with a table (data) file (in the same format as the output file described in the previous section) to generate a new multipage drawing having the *stacked* page layout mode.

The selected object contains formating information, and it is also used as a template to be replicated for each data row in the table file. If an attribute of the replica matches the column name of the table, the attribute value is set to the value in the table file. The replicas are tiled horizontally first.

Eight attributes must be specified in the template object. They are all case-sensitive. The ones that measure distances can be specified in inches ("in"), centi-meters ("cm"), or pixels (if no units are specified).

PAPER_WIDTH
: This specifies the width of the paper.

PAPER_HEIGHT
: This specifies the height of the paper.

LEFT_MARGIN
: This specifies the distance to the left edge of the paper.

TOP_MARGIN
: This specifies the distance to the top edge of the paper.

H_PITCH
: This specifies the distance between the left edges of the replicas.

V_PITCH
: This specifies the distance between the top edges of the replicas.

NUM_COLS
: This specifies the number of replicas to tile horizontally before moving down to the next row.

NUM_ROWS

This specifies the number of replicas to tile vertically before moving to the next page.

After each replica is generated, filled with the data from the table file, and placed, its attribute named *exec* is executed (unless an attribute named *EXEC_AFTER_MERGE* is specified, in which case, the attribute named by the *EXEC_AFTER_MERGE* attribute is executed instead). If there is no attribute named by the *EXEC_AFTER_MERGE* attribute, nothing is executed. (Please see the INTERNAL COMMANDS section for details on command execution.) One can use the *exec* command to construct other attributes from the attributes associated with the data table.

If an attribute whose name is empty and whose value is the string *USER_PLACEMENT*, the user will be asked to place the replica (object name will be displayed in the Status Window when the object is being placed). In this case, the 8 placement related attributes are ignored.

If an attribute whose name is empty and whose value is the string *STRIP_DOUBLE_QUOTES*, data fields surrounded by double-quotes are stripped.

MIME TYPES AND MAILCAPS

When an URL names an HTTP server, the HTTP server sends the *Content-type* of the URL along with the remote file referenced by the URL to tgif. The *Content-type* contains information such as the type/subtype of the file plus some optional fields. If the file is not a tgif file, the following mechanism is used to view the file.

First, the X defaults are looked up to see if there is an external viewer specified for the file. Please see Tgif.@@@Viewer in the X DEFAULTS section below for details. If there's no match, the type/subtype is matched against entries in the MIME-types file. The default MIME-types file is *.mime.types* in user's home directory. Please see Tgif.MimeTypesFile in the X DEFAULTS section on how to override the default MIME-types file. The first field in each line of the MIME-types file specifies type/subtype information. If there is a type/subtype match in the MIME-types files, the MailCap files are consulted as follows.

A line in a MailCap file consists of fields separated by semi-colons. The first field specifies the type/subtype and the second field specifies a *view command* for viewing a file that matches the type/subtype. For tgif, the view command must contains a single *%s* substring to be replaced by local copy of the URL. Only the *%t* and the *%{}* optional fields are supported by tgif. The *multipart* MIME-type is not supported. The type/subtype information of the remote file is matches against the MailCap files. If a match is found, the corresponding view command is executed. If no match is found, but the type of the remote file is either *application*, *audio*, *image*, or *video*, the file is saved and no external viewer is launched. Otherwise, the remote file is assumed to be pure text and tgif will create a text object to view the text.

The MailCap files are the (colon-separated) files specified by the MAILCAP environment variable (if defined). If MAILCAP is not defined, the *.mailcap* file in the user's home directory is used.

MIME is the Multipurpose Internet Mail Extensions specified in RFC1521, and MAILCAP is specified in RFC1524.

HOW TO MAKE A BUILDING-BLOCK OBJECT (SYMBOL FILE)

Here are the steps for defining a building-block object, to be used in a hierarchical design.

1) Draw the representation part of the building-block object. Group everything together. Select this grouped object.

2) Popup the main menu with the middle mouse button; select ''Special''. Select ''MakeSymbolic'' from the next popup menu. The selected object becomes a symbol and gets a dashed boundary.

3) Type in attributes as individual text strings. Select the symbol object and all the text strings to be attached to the symbol. Type #a (for *Attach*) to attach attributes to the symbol.

4) (This step is optional.) Build the definition part of the building-block object. Look at the ''flip-flop.sym'' file for an example. To look at that file, first, instantiate a ''flip-flop'' by typing ^i (for *Instantiate*). Select the flip-flop from the popup window; place the flip-flop; select the flip-flop and type ^v (for *Push*) to see the symbol file.

5) Save and name the file. If the current library path contains the current directory (or '.'), the symbol just built should be instantiatable by typing ^i.

X11 PIXMAP (XPM) FORMATS

Tgif can only import X11 pixmaps that satisfy the constraints described here. The format of the X11 pixmap must be either 1 (XPM1) or 3 (XPM3). Only a subset of the XPM3 format is supported, namely, the *key* field for the color specification must be 'c' (for color visuals). Tools that generate XPM1 format files are (they might have been upgraded to support XPM3), *pbmplus* (or *netpbm*), which is a set of bitmap and pixmap conversion freeware (together with *xv*, the colors for pixmap objects can be manipulated), and *xgrabsc*, another freeware; also, *xloadimage* can display XPM1 files. Tools that can generate XPM3 format files are, for example, *xsnap*(1) and *sxpm*(1). For each color specified in the color string, a color cell is allocated. If the allocation fails, the current color will be used for that color string. If the first color character is a back-quote (‘) or a space, then the corresponding color is substituted with the *background* color of the tgif window if the Tgif.GuessXPmBgColor X default is set to ‘‘true’’. (This design choice is made because the pixmap will then look ‘‘right’’ on both regular and reverse video.) The following is an example of a very small pixmap file (in XPM1 format).

```
#define arrow_format 1
#define arrow_width 5
#define arrow_height 3
#define arrow_ncolors 3
#define arrow_chars_per_pixel 1
static char *arrow_colors[] = {
  "‘", "Black",
  "a", "red",
  "b", "yellow"
};
static char *arrow_pixels[] = {
"‘a‘‘‘",
"aabbb",
"‘a‘‘‘"
};
```

LATEX FIGURE FORMATS

Here we show how to make a figure for a LaTeX file, first with the *\psfig* (or *\epsf*) special construct, then with the *psfile* special construct. (The author does not recommend the *psfile* construct.) An example of both can be found in ‘‘example.tex’’ which is included with the tgif distribution.

To print a tgif file to be included in a LaTeX document with the *\psfig* or *\epsf* special construct (files generated will be in the *Encapsulated PostScript* format), first select LaTeX format in the panel window (click the left mouse button on the laser printer icon), then type ^p to generate the Encapsulated PostScript file. If the file name is ‘‘an-sr-flip-flop.obj’’, then the LaTeX figure file generated will be named ‘‘an-sr-flip-flop.eps’’. This file can be included in a LaTeX document as follows,

```
\input{psfig}
\begin{figure*}[htb]
\centerline{\psfig{figure=an-sr-flip-flop.eps}}
\caption{An SR flip-flop.  \label{fig:an-sr-flip-flop}}
\end{figure*}
```

An alternative way is to use the *\epsf* construct as follows,

```
\input{epsf}
\begin{figure*}[htb]
\centerline{\epsffile{an-sr-flip-flop.eps}}
\caption{An SR flip-flop.  \label{fig:an-sr-flip-flop}}
\end{figure*}
```

The \centerline command above centers the picture. If one has multiple tgif figures in one's LaTeX

document, one only have to include the psfig macro (\input{psfig} or \input{epsf}) once, right after the \begin{document} statement.

If Encapsulated PostScript is not available, the *psfile* special construct can be used as described here. In this case, since LaTeX doesn't not know where the bounding box of the drawing is, it takes some practice to get this just right. Here is something that seems to work. First, center the picture on the page (e.g., the width of a portrait style page is 8.5 inch, so the center of the page is at the 4.25 inch mark), and make the top object in the picture about 1/4 inch away from the top of the page. Select the LaTeX format in the panel window, then print in the LaTeX format. As with the *psfig* construct, a file with the *.eps* extension will be generated. This file can be included in a LaTeX document as follows,

```
\begin{figure*}[htb]
\special{psfile="an-sr-flip-flop.eps" hoffset=-40}
\rule{0in}{1.1in}
\caption{An SR flip-flop.  \label{fig:an-sr-flip-flop}}
\end{figure*}
```

The \rule{0in}{1.1in} above specifies an invisible box of 1.1 inches high, which is the total height of the picture in an-sr-flip-flop.

CONNECTING OBJECTS

In the world of E-CAD, an icon object can represent an electronic component and a line object can represent a connection between a pair of pins of two electronic components. When a component moves, the endpoint of a wire connecting to the component will also move with the component. Tgif simulates these functionalities in a limited fashion.

In tgif, a connection is represented by matching signal names. A wire is defined as a polyline object having a *type=tgWire* attribute and an attribute named *signal_name*. The definition of a *pin* is more complicated. It is described in the next paragraph. If two pins have identical values for the *signal_name* attribute, they are considered to be connected (they do not have to be visually connected by a wire).

A *pin* object must have a *type=port* attribute and attributes named *signal_name* and *name*. But not all objects having such attributes are pins. In addition, a pin object must be either:

1) a top-level symbol or an icon object

or:

2) an immediate subobject of a *owner* symbol or icon object. or:

3) an immediate subobject of a *owner* grouped object which has a *type=tgBroadcastWire* attribute.

In (2) above, the *owner* object must also have an attribute named *name* and must *not* be a subobject of another symbol or icon object. If the *owner* object is a subobject of a grouped object, the *name* attributes of the grouped object will be ignored.

In (3) above, that grouped object can be created using the ConnectPortsToBroadcastWire() command in the PortsAndSignals submenu of the Special Menu when a polyline object and some floating port objects are selected.

A *pin* object can have a connected view and a disconnected view. A connected view is a subobject with a *view=conn,FILL,PEN* attribute and a disconnected view is a subobject with a *view=disconn,FILL,PEN* attribute; FILL and PEN are numeric values between 0 and 31 (inclusive). The value corresponds to patterns in the Fill Menu and the Pen Menu. Normally, only 0 or 1 should be used. When the *signal_name* attribute of a pin object is changed from an empty string to a non-empty string, the pen and fill of the subobject that corresponds to the disconnected view will be set to 0 (meaning NONE) and the pen and fill of the subobject that corresponds to the connected view will be set to the values specified in the *view* attribute of the connected view. When the *signal_name* attribute of a pin object is changed from a non-empty string to an empty string, the pen and fill of the subobject that corresponds to the connected view will be set to 0 and the pen and fill of the subobject that corresponds to the disconnected view will be set to the values specified in the *view* attribute of the disconnected view.

A connection can be created using the ConnectTwoPortsByAWire() command from the PortsAndSignals

submenu of the Special Menu. Please note that if a pin is part of another object, that object must also have a *name* attribute with a non-empty value. When two pins are connected using this command, the *signal_name* attributes of the pins and the wire will be set to have the same value.

The moving of endpoints when a component moves is implemented in tgif using the constrained move mode from the MoveMode Menu (please see Tgif.ConstrainedMove in the X DEFAULTS section for additional information). Please note that a connected wire that is not visually connected will not automatically extends itself to follow a connected component even in the constrained move mode. Also, when a wire object is deleted, the *signal_name* attributes of connected pins do not change (since they are not really "connected").

X DEFAULTS

Tgif.Geometry: WIDTHxHEIGHT+X+Y

Tgif.IconGeometry: +X+Y

Tgif.Foreground: COLORSTRING
The default foreground color is Black.

Tgif.Background: COLORSTRING
The default background color is White.

Tgif.BorderColor: COLORSTRING
If not specified, the foreground color will be used.

Tgif.ReverseVideo: [on,off]
For black and white terminal, reverse video ‘‘on’’ means the background is black. For color terminal, reverse video ‘‘on’’ means the background is specified by the Tgif.Foreground color. The default is off.

Tgif.InitialFont: [Times,Courier,Helvetica,NewCentury,Symbol]
This specifies the initial font. The default is Courier.

Tgif.InitialFontStyle: [Roman,Bold,Italic,BoldItalic]
This specifies the initial font style. The default is Roman.

Tgif.InitialFontJust: [Left,Center,Right]
This specifies the initial font justification. The default is Left.

Tgif.InitialFontDPI: [75,100]
Obsoleted.

Tgif.InitialFontSizeIndex: [0,1,2,3,4,5]
Obsoleted.

Tgif.InitialFontSize: NUMBER
This specifies the size of the start-up font. The default is 14.

Tgif.MsgFontSizeIndex: [0,1,2,3,4,5]
Obsoleted.

Tgif.MsgFontSize: NUMBER
This specifies the size of the font used for messages, menues, and popup windows. The default is 14.

Tgif.RulerFontSize: NUMBER
This specifies the size of the font used for ruler windows. The default is 10.

Tgif.DefaultFontSize: NUMBER
This specifies the size of the font to be used when a requested for a font size can not satisfied. This size *must* exist for *all* fonts used in tgif. The default is 14.

Tgif.FontSizes: NUMBER1 NUMBER2, ...
This specified the font sizes. The default is *8 10 11 12 14 17 18 20 24 25 34*. An alternative form allows "pt" to be specified immediately after a NUMBER (with no space between "pt" the the

NUMBER).

Tgif.AdditionalFonts: FONT_SPEC1 FONT_SPEC2 ...

In addition to the Times, Courier, Helvetica, NewCentury, and Symbol fonts, additional fonts can be specified here. Please see the ADDITIONAL FONTS section for details.

Tgif.FontNamePrefix: [-, *]*

This specified the prefix to be used when tgif makes a request to the X server. The default is -*. Certain fonts have obscure font names (e.g., does not start with the - character). In order to use these fonts, this X default can be set to *.

Tgif.HasAlternateDefaultFonts: [true,false]

The default value of this X default is false. If it set to ‘‘false’’, tgif uses the iso8859 registry with ASN1 encoded screen fonts, and it look for "times", "courier", "helvetica", "new century schoolbook", and "symbol" as part of the screen font names. Some X servers do not support these fonts. In this case, this X default can be used to make tgif use user specified screen and PostScript fonts. If this X default is set to ‘‘true’’, tgif will look for additional X defaults of the form *Tgif.<ps_font_name>*, where *<ps_font_name>* can be one of the following strings:

Times-Roman
Times-Bold
Times-Italic
Times-BoldItalic
Courier-Roman
Courier-Bold
Courier-Oblique
Courier-BoldOblique
Helvetica-Roman
Helvetica-Bold
Helvetica-Oblique
Helvetica-BoldOblique
NewCenturySchlbk-Roman
NewCenturySchlbk-Bold
NewCenturySchlbk-Italic
NewCenturySchlbk-BoldItalic
Symbol

The corresponding value of the X default must contain "%d" as part of the string, and the "%d" string will be replaced by the font size when the font is requested. For example, The following lines will use the Times New Roman screen font instead of the Times screen font and use the Bookman PostScript font instead of the Times PostScript font, if Tgif.HasAlternateDefaultFonts is ‘‘true’’:

Tgif.Times-Roman: *-times new roman-medium-r-*--%d-*,Bookman-Light
Tgif.Times-Bold: *-times new roman-bold-r-*--%d-*,Bookman-Demi
Tgif.Times-Italic: *-times new roman-medium-i-*--%d-*,Bookman-LightItalic
Tgif.Times-BoldItalic: *-times new roman-bold-i-*--%d-*,Bookman-DemiItalic

Please note that certain X servers require the right-hand-side font specifications to have all the dashes in place.

Tgif.DefaultCursor: [x_cursor,arrow,...]

This specifies the select cursor. Entries in <X11/cursorfont.h> (without the XC_ prefix) are valid names of the cursor. The default is arrow.

Tgif.DrawCursor: [x_cursor,arrow,...]

This specifies the cursor used when drawing objects. Entries in <X11/cursorfont.h> (without the XC_ prefix) are valid names of the cursor. The default is the same as Tgif.DefaultCursor.

Tgif.DragCursor: [x_cursor,arrow,...]
This specifies the cursor used when dragging. Entries in <X11/cursorfont.h> (without the XC_ prefix) are valid names of the cursor. The default is hand2.

Tgif.VertexCursor: [x_cursor,arrow,...]
This specifies the cursor used in the select vertices mode. Entries in <X11/cursorfont.h> (without the XC_ prefix) are valid names of the cursor. The default is plus.

Tgif.FreeHandCursor: [x_cursor,arrow,...]
This specifies the cursor used in freehand drawing mode. Entries in <X11/cursorfont.h> (without the XC_ prefix) are valid names of the cursor. The default is pencil.

Tgif.RubberBandColor: COLORSTRING
This specifies color used for rubber-banding (XORing). The default color is the same as the foreground color.

Tgif.MaxColors: NUMBER
This specifies the maximum number of colors. Color0 through ColorMax, where Max is NUMBER-1, in X defaults are queried. If NUMBER is greater than the default of 11, Color11 through ColorMax *must* all exist in X defaults. Please see the GRAPHICAL OBJECTS section for a list of the default colors.

Tgif.Color#: COLORSTRING
This specifies the correspondence between a color number and a color.

Tgif.DefaultColorIndex: NUMBER
This specifies the default color index if a certain color can not be found. The default is 0.

Tgif.ShortCuts: ITEM1 ITEM2 ...
The ITEM specifies the correspondence between a key (may be case sensitive) and a non-alphanumeric key command. Please see the SHORTCUTS section for details.

Tgif.MaxLineWidths: NUMBER
This specifies the maximum number of line widths. LineWidth0 through LineWidthMax, ArrowWidth0 through ArrowWidthMax, and ArrowHeight0 through ArrowHeightMax, where Max is NUMBER-1, in X defaults are queried. If NUMBER is greater than the default value of 7, LineWidth7 through LineWidthMax, ArrowWidth7 through ArrowWidthMax, and ArrowHeight7 through ArrowHeightMax *must* all exist in X defaults. Some default values will be used for those that are not specified in the X defaults.

Tgif.DefaultLineWidth: NUMBER
This specifies the initial line width *index*. The default is 0.

Tgif.LineWidth#: NUMBER
This specifies a line width. The default line widths are 1, 2, 3, 4, 5, 6, and 7.

Tgif.ArrowWidth#: NUMBER
This specifies the width (when the arrow is pointing horizontally) of the arrow head for arc and open-spline objects. The default arrow widths are 8, 10, 12, 14, 18, 20, and 22.

Tgif.ArrowHeight#: NUMBER
This specifies *half* the height (when the arrow is also pointing horizontally) of the arrow head for arc and open-spline objects. The default arrow heights are 3, 4, 5, 6, 7, 8, and 9.

Tgif.MaxDomains: NUMBER
This specifies that NUMBER is the number of domains. DomainPath0,DomainPath1,...,DomainPathM all must exist in X defaults. Here M=NUMBER-1.

Tgif.DomainPath#: DOMAINSTRING
This specifies the correspondence between a domain number, a domain name, and the path associated with a domain. Hence one DomainPath# X default is required for each domain defined. Here the # should be replaced with a domain number. The domain numbers should be 0,1,...,MAXDOMAINS-1, where MAXDOMAINS is set in the MaxDomain X default above. The MaxDomain X

default in combination with the DomainPath# X default are required to use domains.

DOMAINSTRING contains strings which are separated by the ':' symbol. The first string is the name of the domain. Each of the rest of the strings specifies a directory where symbol files are to be searched when the Instantiate command is executed (please see the HOW TO MAKE A BUILDING-BLOCK OBJECT section for details). Another way to look at the DOMAINSTRING specification is that removing the first string (which specifies the domain name) and the first ':' symbol, a DOMAINSTRING has the form of the *PATH* csh(1) environment variable. For example, to specify the symbol path for domain *DEFAULT* to look for symbol files, first in the library directory /tmp/tgif/symbols, then in the current directory, DOMAINSTRING should be set to the following value:

DEFAULT:/tmp/tgif/symbols:.

Tgif.DefaultDomain: NUMBER

This specifies the default domain when tgif starts up. The default is 0.

Tgif.PrintCommand: COMMAND

This specifies the print command used for printing the PostScript file. The default is lpr(1). An example would be *lpr -h -Pprintername*. If COMMAND contains a *%s* substring, the *%s* will be replaced by the full path name of the PostScript file which is normally send to the print command. Therefore, COMMAND without a *%s* substring behaves identically to *COMMAND %s*. Please note that this only works when running tgif without the -print command line option. This can be used to send a font file to the printer before the tgif PostScript file is sent as in the following example:

cat /somewhere/sansfex.pfa %s | lpr -Pmyprinter

Tgif.WhereToPrint: STRING

This specifies the initial print/export destination/format. STRING can be Printer, EPS, PS, Bitmap, Text, EPSI, GIF, HTML, PDF, WinEPSI, PNG, JPEG, or NetList. The default is EPS.

Tgif.PrintDirectory: PATH

This specifies the print directory when the output destination is not the printer. The default is a null string, which means that the output goes into the directory in which the current file resides.

Tgif.NoTgifIcon: [true,false]

If set to ''true'', tgif will not use its own icon window. In this case, one should also set Tgif.UseWMIconPixmap described below to true. The default is false.

Tgif.UseWMIconPixmap: [true,false]

If set to ''true'', tgif will use the standard icon pixmap. Also, Tgif.NoTgifIcon will be ignored. The default is true.

Tgif.DontShowVersion: [true,false]

If set to ''true'', the tgif version will not be displayed on top of the tgif window. The default is true.

Tgif.XBmReverseVideo: [true,false]

If set to ''true'', an invert bitmap operation will be performed when importing an X11 bitmap file. The default is false.

Tgif.AskForXBmSpec: [true,false]

If set to ''true'', the user will be asked to specify magnification and geometry for an X11 bitmap file being imported. Format of the specification is *MAG=WxH+X+Y*, where MAG is the magnification, W and H specifies the width and height, and the location specification can be +X+Y, +X-Y, -X+Y, and -X-Y. The '=' is mandatory if any of the geometry information is specified. The default is false.

Tgif.AskForXPmSpec: [true,false]

If set to ''true'', the user will be asked to specify magnification and geometry for an X11 pixmap file being imported. The format of the specification is the same as for AskForXBmSpec. The

default is false.

Tgif.StripEPSComments: (obsolete)
This X default is obsolete in tgif-4.0.11 because it turns out that it's not always okay to strip PS comments (it should always be set to false).

Tgif.GuessXPmBgColor: [true,false]
If set to ''true'', then when tgif imports an X11 pixmap file with the first color string being ' ' (the space character) or ''' (the back quote character), it will treat the first color as a *background* color. This means that the specified color in the X11 pixmap file will be changed to the current background color. The default is false. (Please note that this default was *true* before patch 2 of tgif-2.7. This X default is there for compatibility reasons; it should be considered obsolete.)

Tgif.XPmOutputVersion: NUMBER
This specifies the XPM version number when outputting in the X11 pixmap format. NUMBER can take on values 1 or 3. The default is 1.

Tgif.XPmInXGrabSCFormat: [true,false]
If Tgif.XpmOutputVersion is set to 1, setting this to ''true'' will force the X11 pixmap output to resemble what xgrabsc generates. The default is false.

Tgif.UseGrayScale: [true,false]
If set to ''true'', gray scales will be used for tiling patterns to speed up printing. The default is false.

Tgif.AutoPanInEditText: [true,false]
If set to ''true'', auto panning will be used such that the text cursor is always visible in text edit mode (except when the cursor is to the left or on top of the paper). This should probably be turned off on slow servers. The default is true.

Tgif.PercentPrintReduction: NUMBER
The specifies the initial percent print reduction/magnification. The default is 100.

Tgif.ConstrainedMove: [true,false]
This specifies the initial move mode. When set to ''true'', moving or stretching an object will cause the endpoints of all polylines or open-splines, whose endpoints fall within the object, and may be the neighboring vertices, to be moved. Please see the IDIOSYNCRASIES section for more details. The default value is false.

Tgif.DoubleQuoteDoubleQuote: [true,false]
When set to ''true'', output of the double-quote character will be preceded by a double-quote character; when set to false, output of the double-quote character will be preceded by a back-slash character. The default value is false.

Tgif.GridSystem: [English,Metric]
This sets the initial grid system. The default is English.

Tgif.InitialGrid: NUMBER
This specifies the initial grid size. For the English grid system, NUMBER can be -2, -1, 0, +1, or +2 for grid sizes of 1/32, 1/16, 1/8, 1/4, and 1/2 inch. For the Metric grid system, NUMBER can be -1, 0, +1, or +2 for grid sizes of 1mm, 2mm, 5mm, and 1cm. The default value is 0.

Tgif.DropObsIconAttrWhenUpdate: [true,false]
If set to ''true'', obsolete icon attributes will be dropped without confirmation when the UpdateSymbols command is executed. If set to ''false'', a popup window will prompt the user to specify what to do with the obsoleted icon attributes. The default is false.

Tgif.UseRecentDupDistance: [true,false]
If set to ''true'', the most recent change in position produced by a combination of a duplicate and a move command will be used for the new duplicate command. Otherwise, some default distance will be used to position the duplicate. The default is true.

Tgif.SplineTolerance: NUMBER
This specifies the tolerance of spline drawing. The smaller the number, the smoother the spline. The default is 9 (min is 3 and max is 13).

Tgif.SplineRubberband: [true,false]
If set to ‘‘true’’, spline rubber-bands will be used in drawing, moving, and stretching open and closed splines. (This might not be desirable if the spline contains too many vertices.) The default is true.

Tgif.Synchronize: [on,off]
XSynchronize is called if this default is set to ‘‘on’’. The default is off.

Tgif.DoubleClickUnIconify: [true,false]
If set to ‘‘true’’, double mouse clicks are used to de-iconify the icon window (in this mode, the icon window ignores single mouse clicks and drags). The default is false.

Tgif.MainMenuPinDistance: NUMBER
This specifies the horizontal distance (in pixels) the user needs to drag a popup menu before the popup menu is to be pinned down. The default is 80. (If pinned popup menus are not desired, then this should be set to a value greater than the screen width.) Dragging the left mouse button can be used to move the pinned popup menu; clicking the right button in the popup menu will remove it.

Tgif.DoubleClickInterval: NUMBER
This specifies the maximum interval (in milliseconds) between two mouse clicked to be recognized as one double-click. The default is 300.

Tgif.HandleSize: NUMBER
This specifies (half) the size of the handle used to highlight objects. Its allowable value is between 2 and 6. The default is 3.

Tgif.HistoryDepth: NUMBER
This specifies the size of the undo/redo buffer; negative values mean that the buffer is unbounded. The default is -1.

Tgif.SaveTmpOnReturn: [true,false]
If set to ‘‘true’’, a tmpmodel file will be saved automatically before returning to the driver. Otherwise, no files will be saved automatically. The default is true.

Tgif.ImportFromLibrary: [true,false]
If set to ‘‘true’’, the library directories specified by the current domain are searched for .obj, .sym, xbitmap/xpixmap, and EPS files to import. Otherwise, the current directory will be used as the starting point. The default is false.

Tgif.WarpToWinCenter: [true,false]
If set to ‘‘true’’, the mouse is warped to the center of popup windows. Otherwise, the mouse is not warped. The default is true.

Tgif.SaveCommentsInSaveNew: [true,false]
If set to ‘‘true’’, "%%" type comments in the file will be stored in the newly created file. The default is true.

Tgif.CanvasWindowOnly: [true,false]
If set to ‘‘true’’, only the canvas window will be displayed (this is kind of the ‘‘demo’’ mode). The default is false.

Tgif.UsePsAdobeString: [true,false,NUMBER_1/NUMBER_2]
If set to ‘‘true’’, the first line in the PS or EPS file will be "%!PS-Adobe-2.0 EPSF-1.2". If set to ‘‘false’’, it is just "%!". If the PS-Adobe string confuses the document manager (such as Transcript) on your site, you should set it to ‘‘false’’. If the third form is used,, the first line will be "%!PS-Adobe-NUMBER_1 EPSF-NUMBER_2". The default is false.

Tgif.HalfToneBitmap: [true,false]
If set to ‘‘true’’, the Floyd-Steinberg half-tone method will be used when printing in the X11 bitmap format. This is useful when the drawing contains X11 pixmap objects. The default is false.

Tgif.ThresholdBitmap: [true,false]
If set to ‘‘true’’, a simple thresholding method will be used to decide whether a bit is turned on or off when printing in the X11 bitmap format. If Tgif.HalfToneBitmap is set to true, this X default is ignored. The default is false.

Tgif.BitmapThreshold: NUMBER
This specifies the threshold value used in either the Floyd-Steinberg half-tone algorithm or the simple thresholding algorithm. NUMBER must be between 0 and 1. This X default is only active when either the Tgif.HalfToneBitmap or the Tgif.ThresholdBitmap X default is set to true. The default value is 0.5 if Tgif.HalfToneBitmap is true, and is 1.0 if Tgif.ThresholdBitmap is true (basically, anything that is not white will be black).

Tgif.GroupedTextEditable: [true,false]
If set to ‘‘false’’, only top level text objects and attributes of top level objects can be edited when the drawing mode is set to the text mode. If set to ‘‘true’’, text objects and attributes everywhere can be edited. The default is false.

Tgif.DefaultEPSScaling: NUMBER
This specifies the scaling factor applied to an imported PS or EPS image. As mentioned in the IDIOSYNCRASIES section below, tgif treats 128 pixels as an inch and PostScript treats 72 points as an inch. In order to have real-size PostScript images, this parameter should be set to 1.7778 (which is 128/72). The default value is 1.

Tgif.IntrCheckInterval: NUMBER
This specifies the number of objects drawn before tgif checks for interrupts. If this is set to be 0 or less, interrupt is not allowed. The default value is 10.

Tgif.TiledPageScaling: NUMBER
This specifies the scaling value used when a multipage drawing in tiled page mode is printed. Since most PostScript printers do not use the full page as the drawing area, setting this number to 1 may get truncated output. The default value is 0.9.

Tgif.TGIFPATH: STRING
This specifies the directory where the files, mentioned in the FILES section below, can be found. The TGIFPATH environment variable may override this option. The default value is specified by the compiler option TGIF_PATH.

Tgif.TGIFICON: STRING
This specifies the name of the object file to be displayed when tgif is iconified. If it starts with a / character, absolute path is used; otherwise, the actual path of the icon file is $TGIFPATH/STRING where TGIFPATH is either defined using the X defaults or an environment variable. The default value is ‘‘tgificon.obj’’.

Tgif.StickyMenuSelection: [true,false]
If set to ‘‘true’’, when patterns/linewidths/linestyles/... of objects are changed using a menu action, the corresponding pattern/linewidth/linestyle/... becomes the current selection. The default is true.

Tgif.PSBopHook: STRING
If specified, the following PostScript line is added at the beginning of each page when printing to the printer or to a PS file,

```
userdict /STRING known { STRING } if
```

This option should only be used if one is very familiar with PostScript. (Setting STRING to "tgif-bop-hook" is recommanded since it would not have a name conflict with existing software, such as dvips(1).)

Tgif.PSEopHook: STRING
If specified, the following PostScript line is added at the end of each page when printing to the printer or to a PS file,

userdict /STRING known { STRING } if

This option should only be used if one is very familiar with PostScript. (Setting STRING to "tgif-eop-hook" is recommanded since it would not have a name conflict with existing software, such as dvips(1).)

Tgif.MinimalEPS: [true,false]
If set to ‘‘false’’, comments such as %%Pages, %%DocumentFonts, %%EndComments, %%BeginProlog, %%EndProlog, %%Page, %%Trailer, and %%EOF will be generated in an EPS output. These comments may confuse certain ‘‘document managers’’. Therefore, the default is true if Tgif.UsePsAdobeString is not specified (and the default is false if Tgif.UsePsAdobeString is specified).

Tgif.InitialPrintInColor: [true,false]
If set to ‘‘true’’, color output (printing) mode is enabled on startup. Please note that in black and white PS/EPS/EPSI printing mode, the *white* color will be be printed as black (only *background* will be printed as white). The default is true (except when the **-print** command line option is used).

Tgif.InitialShowGrid: [true,false]
If set to ‘‘false’’, showing grid is disabled on startup. The default is true.

Tgif.InitialSnapOn: [true,false]
If set to ‘‘false’’, snapping to the grid points is disabled on startup. The default is true.

Tgif.NoMenubar: [true,false]
If set to ‘‘true’’, no menubar will be shown initially. The default is false.

Tgif.NoStatusWindow: [true,false]
If set to ‘‘true’’, no Status Window will be shown initially. The default is false.

Tgif.ReverseMouseStatusButtons: [true,false]
If set to ‘‘true’’, the left mouse status and the right mouse status are swapped. This should be used when a ‘‘left-handed mouse’’ is used. The default is false.

Tgif.MinimalMenubar: [true,false]
If set to ‘‘false’’, the menu items in the Menubar Window will be the same as the main popup menu. This would take up much more space. If set to ‘‘true’’, the Page, PageLayout, HoriAlign, VertAlign, and MoveMode menus are collapsed into the View cascading menu; the Font, TextStyle, and TextSize menus are collapsed into the Text cascading menu; and the LineDash, LineStyle, LineType, LineWidth, Fill, and Pen menus are collapsed into the Graphics cascading menu. The default is true.

Tgif.ColorBgInPrintingColorPS: [true,false]
If set to ‘‘true’’, the window background color is used as the background color when generating color PostScript output. If set to ‘‘false’’, no background color is used. The default is false.

Tgif.ScrollBarWidth: NUMBER
This specifies the width of a scroll bar. NUMBER must be between 2 and 16. The default is 16.

Tgif.InitialPaperSize: STRING
The STRING specifies the initial width and height of the paper. STRING is in the "<width> x <height>" form. <width> and <height> is a numeric value immediately followed by either "in" (inch) or "cm" (centi-meter). The " x " that separate the <width> and <height> is mandatory. If A4PAPER is defined in the Makefile, the default value is "21cm x 29.7cm". If A4PAPER is not defined in the Makefile, the default value is "8.5in x 11in".

Tgif.UpdateChildUsingAlignment: [true,false,no_overlap]
If set to ‘‘true’’ or ’no_overlap’, when update_eps_child(), update_xbm_child(), or update_xpm_child() internal command is executed, the current horizontal and vertical alignments are used to place the EPS/XBM/XPM subobject. If the horizontal alignment is L, C, R, S, or -, the subobject is aligned to the left, center, right, center, or left, respectively, to the parent object. If the vertical alignment is T, M, B, S, or -, the subobject is placed above, middle, below, middle, or below the parent object if this X default is set to ’no_overlap’; the subobject is aligned to the top, middle, bottom, middle, or below the parent object if this X default is set to ‘‘true’’. If this X default is set to ‘‘false’’, the subobject is placed left aligned and below the parent object. The default is false.

Tgif.GenerateImageMap: [true,false]
If set to ‘‘true’’, NCSA *imagemap* or CERN *Clickable Image* files will be generated when print in GIF format. In this case, Tgif.XpmToGif, Tgif.ImageMapFileExtension, Tgif.GifFileExtension, Tgif.ImageMapFileFormat, and Tgif.UseXPmVersion1ForImageMap X defaults, described below, will be interpreted; otherwise, they are ignored. Please see the section on GENERATING IMAGEMAP FILES for details. The default is false.

Tgif.XpmToGif: STRING
The STRING specifies a command used to convert an XPM file to a GIF file. The STRING *must* contain a *%s* substring to be replaced by the full path name of the XPM file. The default is "xpm-toppm %s | ppmtogif".

Tgif.ImageMapFileExtension: STRING
The STRING specifies the file extension for a imagemap file. The default is "map".

Tgif.GifFileExtension: STRING
The STRING specifies the file extension for a GIF file. The default is "gif" (lower case).

Tgif.ImageMapFileFormat: [NCSA,CERN]
The STRING specifies either the NCSA imagemap or the CERN clickable image format. The default is NCSA for the NCSA imagemap format.

Tgif.UseXPmVersion1ForImageMap: [true,false]
The setting of this X default should depend on the setting of the Tgif.XpmToGif X default above. If set to ‘‘true’’, XPM1 file is generated regardless of the setting of the Tgif.XPmOutputVersion X default. The default is true.

Tgif.UsePaperSizeStoredInFile: [true,false]
If set to ‘‘true’’, the paper size information stored in a just opened file is used. The default is true.

Tgif.OneMotionSelMove: [true,false]
If set to ‘‘true’’, one can select and move an object in one motion. The default is false.

Tgif.TiffEPSI: (obsolete)
This X default is obsolete because TiffEPSI became a supported export format since tgif-4.0.

Tgif.XbmToTiff: STRING
The STRING specifies a command used to convert an XBM file to a TIFF file. The STRING *must* contain either one of two *%s* substring. The first *%s* substring is to be replaced by the full path name of the XBM file. The optional second *%s* substring is to be replaced by the full path name of the generated TIFF image. The default is "xbmtopbm %s | pnmtotiff -none > %s".

Tgif.EPSIExportExtension: STRING
STRING specifies the file extension used for exporting *EPSI* files. The default is "eps".

Tgif.HotListFileName: STRING
STRING specifies a full path name of a file used to store the hot file list. By default, this file is .Tgif_hotlist in the user’s home directory.

Tgif.@@@Viewer: STRING
STRING specifies an external viewer for an remote URL with a file extension of @@@. STRING can be in 3 forms. It can be the string "NONE" to indicate that when such a remote file is encountered, tgif should retrieve the file into a user specified directory. For example, if one wishes to retrieve .gz files, one can use:

Tgif.gzViewer: NONE

STRING can also contain the string %S (S is capitalized), this indicates that %S is to be replaced by the URL. For example, if one wishes to view .html files using xmosaic, one can use:

Tgif.htmlViewer: xmosaic %S

Another form of STRING contains the string %s (S is lower-case), this indicates that the remote file is to be retrieved into a user specified directory and view by a tool. For example, if one wishes to view .gif files using xv, one can use:

Tgif.gifViewer: xv %s

Please note that this mechanism has precedence over the mechanism described in the MIME TYPES AND MAILCAPS section above.

Tgif.AutoHyperSpaceOnRemote: [true,false]
If set to ‘‘false’’, tgif will not go into the *hyperspace* mode when a remote URL is visited. The default is true.

Tgif.AllowLaunchInHyperSpace: [true,false]
If set to ‘‘true’’, launching of applications is enabled in the *hyperspace* mode when a remote URL is visited. This is potentially very dangerous because the application may do catastrophic damages. Therefore, it is strongly recommanded that it is set to false. The default is false.

Tgif.CanChangeAttrColor: [true,false]
If set to ‘‘true’’, color of an attribute can be changed when it is attached to an object. The default is false.

Tgif.MimeTypesFile: STRING
STRING specifies a full path name of the MIME-types file. Tgif only uses the type/subtype field in the MIME-types file and ignores all other fields. The default MIME-types file is *.mime.types* in user’s home directory.

Tgif.LocalRGBTxt: STRING
If one would like to override certain system colors, one can use STRING to specify a full path name of a file to be consulted first before looking up the color in the server. The file must be in the same format as the *rgb.txt* file. Namely, each line contains 4 fields, the first 3 fields correspond to the red, green, and blue components of the color, and the 4th field is the name of the color. A color component must have a value between 0 and 255 (inclusive).

Tgif.PrintUsingRequestedColor: [true,false]
If set to ‘‘true’’, the color PostScript file being printed will use the *requested* color instead of the color returned by the X server. The default is false.

Tgif.ShowMeasurement: [true,false]
If set to ‘‘true’’, the location of the cursor and the width and height of the object begin drawn/dragged/stretched will be shown. The default is false.

Tgif.ShowMeasurementUnit: [pixel,inch,cm]
The STRING specifies the unit used to display the measurement cursor. The default is pixel.

Tgif.PageStyleLandscape: [true,false]
If set to ‘‘true’’, tgif comes up in landscape mode. The default is false.

Tgif.QueryZoomInPoint: [true,false] or [always,no_select,no_query,never]
If set to ‘‘true’’ (or ‘‘always’’), the user will be asked to select a center point when zooming in. If set to ‘‘no_select’’, the user will be asked to select a center point when zooming in if no objects

are selected. If set to ‘‘no_query’’, the position of the mouse is the zoom-in point. In this case, it is not desirable to zooms in using a menu selection. The default is false (or never).

Tgif.GZipCmd: STRING
The STRING specifies a command used to gzip a .obj file. The command *must* produce output into its stdout. If the command contains a *%s* substring, the *%s* will be replace by the full path name of a temporary copy of the .obj file. The default is "gzip -c".

Tgif.GUnZipCmd: STRING
The STRING specifies a command used to unzip a zipped tgif file (with extension .obj.gz or .sym.gz) into a tgif file. The command *must* produce output into its stdout. If the command contains a *%s* substring, the *%s* will be replace by the full path name of a temporary copy of the zipped file. The default is "gunzip -c".

Tgif.HttpProxy: STRING
The STRING specifies a host name and a port number of an HTTP proxy server. Format of the specification is *<host>:<port>*. If *:<port>* is omitted, 80 is used as the default port number. The environment variable *http_proxy* has precedence over this X default. The default is not to use an HTTP proxy server.

Tgif.FtpProxy: STRING
The STRING specifies a host name and a port number of an FTP proxy server. Format of the specification is *<host>:<port>*. If *:<port>* is omitted, 21 is used as the default port number. The environment variable *ftp_proxy* has precedence over this X default. The default is not to use an FTP proxy server.

Tgif.InitialArrowStyle: [NONE,RIGHT,LEFT,DOUBLE]
This specifies the initial arrow style for polyline/open-splines/arcs. The default is RIGHT.

Tgif.ShowPageInEPS: [true,false]
If set to ‘‘true’’, a *showpage* PostScript command will be generated for an EPS or EPSI file. The default is true.

Tgif.MaxNavigateCacheBuffers: NUMBER
This specifies the number of cache buffers allocated to cache remote files (to minimize communication). NUMBER must be non-negative. The default is 40.

Tgif.NumberFileInPrintOnePage: [true,false]
If set to ‘‘true’’, when PrintOnePage from the Print Menu is selected for a stacked multipage drawing (e.g., file.obj), file_*N* with the proper file extension will be generated, where *N* corresponds to the selected page number. The default is false.

Tgif.OneMotionTimeout: NUMBER
When Tgif.OneMotionSelMove is set to true, moving an object is considered to be making a selection if the elapse time between mouse-down and mouse-up is smaller than the timeout value specified by this X default (in milliseconds). The default is 200.

Tgif.MinMoveInterval: NUMBER
When Tgif.OneMotionSelMove is set to false, moving an object is considered to be making a selection if the elapse time between mouse-down and mouse-up is smaller than the interval specified by this X default (in milliseconds). The default is 0.

Tgif.GifToXpm: STRING
The STRING specifies a command used to convert a GIF file to an XPM file. The STRING *must* contain a *%s* substring to be replaced by the full path name of the GIF file. The default is "giftopnm %s | ppmtoxpm".

Tgif.InitExportPixelTrim: LEFT_NUMBER,TOP_NUMBER,RIGHT_NUMBER,BOTTOM_NUMBER
The numbers specify the number of pixels to trim when printing or exporting in the XBM, XPM, or GIF format. The use of these values forms an escape mechanism to fix an idiosyncrasy that tgif can not figure out exactly how big the whole image is. The default values are all 0’s.

Tgif.QuantizingLevels: NUMBER
Some image functions such as Sharpen() uses convolution and may generate an image that uses more than 256 colors which tgif can not handle. The NUMBER specifies the number of colors to quantize down to when such a situation occurs. The default is 222.

Tgif.RotateCursor: [x_cursor,arrow,...]
This specifies the cursor used in the rotate mode. Entries in <X11/cursorfont.h> (without the XC_ prefix) are valid names of the cursor. The default is crosshair.

Tgif.ColorLayers: [true,false]
If set to ''true'', each color is considered to be a different layer which can be individually turned on and off. If a color layer is turned off, primitive objects in that layer will not be visible. A grouped object only becomes invisible when all its constituent objects are invisible. The default is false.

Tgif.TiffToXbmCmd: STRING
The STRING specifies a command used to convert a TIFF file to an XBM file. This command is used when importing an EPSI file generated by a Windows application. The STRING *must* contain a *%s* substring to be replaced by the full path name of the TIFF file. The default is "tifftopnm %s | pgmtopbm | pbmtoxbm"

Tgif.DefFixedWidthFont: STRING
The STRING specifies a font to be used as the default font for the Status Window, menus, dialogboxes, etc. The default is "-*-courier-medium-r-normal-*-14-*-*-*-*-*-iso8859-1".

Tgif.DefFixedWidthRulerFont: STRING
The STRING specifies a font to be used in the horizontal and vertical ruler windows. The default is "-*-courier-medium-r-normal-*-10-*-*-*-*-*-iso8859-1".

Tgif.MenuFont: STRING
The STRING specifies a font to be used in menus. If this X default is not specified, the default font is used in menus.

Tgif.BoldMsgFont: STRING
The STRING specifies a bold font to be used in buttons and dialogboxes. If this X default is not specified but Tgif.MenuFont is specifed, this will take on the value of Tgif.MenuFont. If this X default and Tgif.MenuFont are not specified, the default font is used in bold messages.

Tgif.MsgFont: STRING
The STRING specifies a thin font to be used in the Status Window and dialogboxes. If this X default is not specified, the default font is used in messages.

Tgif.BggenToXpm: STRING
The STRING specifies a command for generating an X11 pixmap file to be executed when RunBggen() is selected from the ImageProc Menu. The STRING *must* contain two *%s* substrings. The first *%s* is to be replaced by a user specified string. The second *%s* is to be replaced by the geometry of the image. The default is "bggen %s -g %s | ppmquant 64 | ppmtoxpm".

Tgif.DefaultErrorDiffuseLevels: R_NUMBER G_NUMBER B_NUMBER
The NUMBERs specify the number of bits of red, green, and blue to be used when ReduceToDefaultColors() or DefaultErrorDiffuse() are selected from the ImageProc Menu. These values determine a set of default colors to be used for color quantization for the ReduceToDefaultColors() and DefaultErrorDiffuse() methods. R_NUMBER+G_NUMBER+B_NUMBER must be less than or equal to 8, and each number must be greater than 0. The default is 2 2 2.

Tgif.MaxImportFilters: NUMBER
This specifies the maximum number of import filters. ImportFilter0 through ImportFilterMax, where Max is NUMBER-1, in X defaults are queried. The default is 0.

Tgif.ImportFilter#: FILTERSTRING
This specifies an import filter. FILTERSTRING has 3 parts (separated by space characters). The first part is the name of the filter. It must not contain a space character. The second part contains

semicolon-separated file extensions. The third part is the actual filter command for converting the named external file type to an X11 pixmap file. Please see the IMPORT RASTER GRAPHICS section for details.

Tgif.ShowFileNameOnBrowse: [true,false]
If set to ‘‘true’’, file names will be shown when BrowseXBitmap(), BrowseXPixmap(), or BrowseOther() are selected from the File Menu. The default is true.

Tgif.HtmlFileExtension: STRING
The STRING specifies the file extension used when printing in the HTML format. The default is "html".

Tgif.GenerateHtmlHref: [true,false]
If set to ‘‘true’’ and when printing in the HTML format, the value of an *href* attribute is parsed. If the value references a .obj file, it’s changed to have a HTML file extension. If it is set to ‘‘false’’, no conversion will be performed. The default is true.

Tgif.RotationIncrement: NUMBER
This specifies the initial rotation increment in degrees. The default is 45.

Tgif.PSA4PaperSize: [true,false]
If set to ‘‘true’’ and A4 size paper is specified, the following line is added to a PS/EPS/EPSI file (before "%%EndComments"):

```
%%DocumentPaperSizes: a4
```

The default is false.

Tgif.ShapeShadowSpec: STRING
The STRING specifies the initial horizontal and vertical offsets of a shape shadow. If both values are zeroes, a shape is created without a shadow. When creating a shape with a shadow, background fill pattern (3rd pattern in the first column of the Fill Menu) usually gives the best result. The default is "0,0".

Tgif.StretchableText: [true,false]
If set to ‘‘true’’, stretchable text mode is the initial mode. The default is true.

Tgif.EditTextSize: NUMBER
This specifies the text size to be used in editing existing text objects. NUMBER should either be 0 or a value between 4 and 34 (inclusive). If NUMBER is 0, the actual text size is used in editing existing text objects. The value of the edit text size can later be changed by selecting SetEditTextSize() from the Properties Menu. The default is 0.

Tgif.IconPixmap: STRING
STRING specifies the path of an XBM or XPM file to be used as tgif’s desktop icon. If STRING starts with a / character, absolute path is used; otherwise, the actual path of the icon file is $TGIFPATH/STRING where TGIFPATH is either defined using the X defaults or an environment variable. This X default is only enabled if Tgif.UseWMIcon is set to true. The default value is ‘‘tgificon.xbm’’ (which is compiled into tgif).

Tgif.TmpFileMode: NUMBER (OCTAL)
This specifies file mode for temporary and exported files. NUMBER must be an octal number. If NUMBER is 0, no attempt is made to change the file mode. If this value is specified (even if it’s 0), it overrides the PSFILE_MOD compile option. There is no default value.

Tgif.TitledPinnedMenu: [true,false]
If set to ‘‘true’’, pinned menu will have a title bar and left button is used for selecting menu items in a pinned menu. The default is true.

Tgif.ColorFromXPixmap: STRING
STRING specifies the path of an XPM file to be used to load the initial colors. If this X default is specified, the Tgif.Color# X defaults are ignored.

Tgif.VectorWarpSoftness: NUMBER
This specifies the softness value used when VectorWarp() is selected from the ImageProc Menu. VectorWarp() lets the user warp pixels in an X11 pixmap object by specifying a vector. The size of the affected area is controled by this value, which must lie between 1.0 and 4.0. The larger the value, the larger the affected area. The default value is 2.0.

Tgif.ChangePropertiesOfAttrs: [true,false]
If set to ‘‘true’’, changing a property (such as font, font size, color, etc.) of an object will change the property of the attributes attached to the object in the same way. The default is false.

Tgif.ShiftForDiagMouseMove: [true,false]
If set to ‘‘true’’, certain mouse movements are restricted to multiple of 45 degrees. The default is true.

Tgif.UseRecentForDiagMouseMove: [true,false]
If set to ‘‘true’’, the object that is used as anchor for measuring the moving distance is used as an anchor when objects. This only works if Tgif.UseRecentDupDistance and Tgif.ShiftForDiag-MouseMove are both set to true, The default is false.

Tgif.FlushColormapOnOpen: [true,false]
If set to ‘‘true’’, colormap is flushed the initial tgif colors are reloaded when a new file is opened. The default is false.

Tgif.TransparentPattern: [true,false]
If set to ‘‘true’’, fill and pen patterns are transparent initially. The default is false.

Tgif.DontReencode: STRING
For fonts that are not iso8859-1 encoded, non-ASCII portion of the font (characters with bit 7 on) is by default reencoded as if it iso8859-1 encoded. If this is not desirable for a font, reencoding can be bypassed by including the first part of the PostScript font name of the font in STRING. Fields in STRING are colon-separated. For example, if STRING is "Times:Courier:Helvetica", PostScript fonts that begins with "Times", "Courier", or "Helvetica" will not be reencoded. (Please note that this X default overwrite the fonts specified by -D_DONT_REENCODE at compile time.) Please also see the ADDITIONAL FONTS section for an example.

Tgif.AdditionalDontReencode: STRING
Use this X default to augment Tgif.DontReencode (or the fonts specified by -D_DONT_REEN-CODE at compile time).

Tgif.UnsignedInXBmExport: [true,false]
If set to ‘‘true’’, *unsigned char* will be used instead of *char* in exported XBM files. The default is false.

Tgif.CommentInBitmapExport: [true,false]
If set to ‘‘true’’, a blank *RCS Header* comment will be prepended to exported XBM and XPM files. The default is false.

Tgif.ShowFontSizeInPoints: [true,false]
If set to ‘‘true’’, font sizes are displayed in the unit of point sizes. The default is false.

Tgif.DontCondensePSFile: [true,false]
By default, PS/EPS files generated by tgif are condensed. If this X default is set to ‘‘true’’, tgif will not generate condensed PS/EPS files. The default is true.

Tgif.StripCondensedPSComments: (obsolete)
This X default is obsolete in tgif-4.0.11 because it turns out that it’s not always okay to strip PS comments (it should always be set to false).

Tgif.PdfFileExtension: STRING
The STRING specifies the file extension used when printing in the PDF format. The default is "pdf".

Tgif.PsToPdf: STRING
The STRING specifies a command used to convert a PS file to a PDF file. The STRING *must* contain 2 *%s* substrings to be replaced by the full path name of the PS file and the full path name of the PDF file. The default is "ps2pdf %s %s".

Tgif.3DLook: [true,false]
If set to ‘‘false’’, no 3D decoration of windows and buttons will be used. The default is true.

Tgif.XpmDeckToGifAnim: STRING
The STRING specifies a command used to convert a list of GIF file to a GIF animation file. The STRING *must not* contain any *%s* substring. The default is "gifsicle -lforever --delay 10". Gifsicle’s home page is <URL:http://www.lcdf.org/gifsicle/>. One can also set this X default to "whirlgif -loop -time 10". Whirlgif’s home page is <URL:http://www.msg.net/utility/whirlgif/>.

Tgif.GifAnimExplode: STRING
The STRING specifies a command used to explode an animated GIF file into its constituent GIF files. The STRING *must not* contain any *%s* substring. The constituent GIF files must have the following file names. If the animated GIF file is named "foo.gif", the constituent GIF files must be named "foo.gif.0", "foo.gif.1", etc. The default is "gifsicle -eU". Gifsicle’s home page is <URL:http://www.lcdf.org/gifsicle/>.

Tgif.Btn3PopupModeMenu: [true,false]
If set to ‘‘true’’, pressing the right mouse button in the canvas window will generate the Mode Menu. The default is false.

Tgif.ScriptFraction: NUMBER
This specifies the size of the super/subscript relative to the size of the normal text. The default value is 0.6.

Tgif.DeleteNextCharWithDelKey: [true,false]
If set to ‘‘true’’, pressing the Delete key on the keyboard will delete the character to the right of the cursor in text mode. The default is true.

Tgif.SquareDoubleByteFonts: FONT_SPEC1 FONT_SPEC2 ...
Starting with version 4.0 of tgif, double-byte fonts are supported. But only double-fonts where every character has the same width and height are supported. Please see the SQUARE DOUBLE FONTS section for details.

Tgif.DefaultSingleByteFont: STRING
Using input methods (specified by the Tgif.DoubleByteInputMethod X default below), one can mix english (single-byte) substrings within a double-byte string. The font to use for the english substring is specified by this X default. The default is Times.

Tgif.@@@ShowFontChar: OCTAL STRING
OCTAL STRING specifies a double-byte octal character to be used to represent a double-byte font in the Choice Window when the font is selected. @@@ should be replaced by the name of the double-byte font. Please see the SQUARE DOUBLE FONTS section for examples.

Tgif.DoubleByteInputMethod: STRING
This specifies the input method for double-byte fonts. Currently, the following values are supported: "xcin", "chinput", "kinput2", and "xim". If you are using xcin-2.5 or above, please use "xim" instead of "xcin". Please see the INPUT METHODS section for details.

Tgif.UseNKF: [true,false]
If set to ‘‘true’’, Network Kanji Filter (NKF) will be used. The default is false.

Tgif.CopyAndPasteJIS: [true,false]
If set to ‘‘true’’, copying and pasting text strings will go through additional JIS to EUC conversion. The default is false.

Tgif.PreeditType: [overthespot,root]
If set to ‘‘overthespot’’, over-the-spot preediting will be used. The default is root.

Tgif.Lang: STRING
This specifies the locale. The environment variables LANG can override this setting.

Tgif.Modifiers: STRING
This specifies the locale modifiers. The environment variables XMODIFIERS can override this setting.

Tgif.ConvSelection: STRING
This specifies the name of the selection used in converting kinput2 strings. The default value is _JAPANESE_CONVERSION.

Tgif.VisibleGridInSlideShow: STRING
If set to ‘‘true’’, grids will be visible in slideshow mode. The default is false.

Tgif.SmoothScrollingCanvas: [off,jump,smooth]
If set to ‘‘smooth’’, scrolling the main canvas window will be smooth. However, there may be a delay when scrolling starts to cache the image. If set to ‘‘jump’’, scrolling the main canvas window will be jumpy. If set to ‘‘off’’, scrolling the main canvas window will not change the canvas until the mouse button is released. The default is jump.

Tgif.LightGrayColor: COLORSTRING
This specifies the color to be used for the background of buttons, menus, etc. The default is gray75.

Tgif.DarkGrayColor: COLORSTRING
This specifies the color to be used for the shadow of buttons, menus, etc. The default is gray50.

Tgif.DefaultObjectBackground: COLORSTRING
This specifies the color to be used for the background of objects. By default, the default background color is used.

Tgif.UseImagePixelsForTrueColorExport: [true,false]
If set to ‘‘true’’, the color table of an exported XPM/GIF file will be obtained from the actual image pixels for a TrueColor visual. The default is false.

Tgif.DialogboxUse3DBorder: [true,false]
If set to ‘‘false’’, dialogboxes will not have 3D borders. This should be used with X servers such as X-Win32 because dialogboxes already have 3D borders. The default is true.

Tgif.MenuFontSet: STRING
This X default is only used if tgif is compiled with the ENABLE_NLS compiler option. The STRING specifies a list of fonts to be used in menus. STRING can be ‘‘none’’ to indicate not to use menu font set. The default is "-*-helvetica-medium-r-normal--12-*-*-*-*-*-*-*,-*-*-medium-r-*--12-*-*-*-*-*-*-*".

Tgif.MsgFontSet: STRING
This X default is only used if tgif is compiled with the ENABLE_NLS compiler option. The STRING specifies a list of fonts to be used in status messages. STRING can be ‘‘none’’ to indicate not to use message font set. The default is "-*-helvetica-medium-r-normal--12-*-*-*-*-*-*-*,-*-*-medium-r-*--12-*-*-*-*-*-*-*".

Tgif.BoldMsgFontSet: STRING
This X default is only used if tgif is compiled with the ENABLE_NLS compiler option. The STRING specifies a list of fonts to be used in messageboxes. STRING can be ‘‘none’’ to indicate not to use bold message font set. The default is "-*-helvetica-bold-r-normal--12-*-*-*-*-*-*-*,-*-*-medium-r-*--12-*-*-*-*-*-*-*".

Tgif.BoldMsgFontDoubleByte: [true,false]
This X default is only used if tgif is compiled with the ENABLE_NLS compiler option. This X default should be set to ‘‘true’’ if the strings used in messageboxes may contain double-byte

characters. The default is false.

Tgif.LocaleDir: STRING
This X default is only used if tgif is compiled with the ENABLE_NLS compiler option. The STRING specifies a full path name of a locale directory.

Tgif.PsRegMarksInTiledPageMode: [true,false]
If set to ‘‘true’’, small crosshairs will be drawn at the corners defining the clipping regions when printing/exporting PS/EPS files in the tiled page mode. The greyness of the cross hairs will be determined by the Tgif.PsRegMarksGray X default. The default is false.

Tgif.PsRegMarksGray: NUMBER
This specifies the greyness of the crosshairs used when Tgif.PsRegMarksInTiledPageMode is set to true. The default value is 0.95

Tgif.PSFontAliases: PSFONTALIAS_SPEC1 PSFONTALIAS_SPEC2 ...
Font aliases can be used to represent different encoding, etc. Please see the ADDITIONAL FONTS section for details.

Tgif.DomainInIni: [true,false]
If set to ‘‘true’’, domain information will be loaded from the ~/.Tgif/domain.ini file and all the menu items in the Domain submenu of the File Menu will be enabled. The default is false.

Tgif.UndoRedoRestoreDrawingMode: [true,false]
If set to ‘‘true’’, the drawing mode just before an undo/redo operation will be restored after undo/redo. The default is true.

Tgif.MenuRowsBeforeScroll: NUMBER
This specifies the maximum number of rows in a user-specifiable text menu (such as the Font Menu and the FontSize Menu) before a vertical scrollbar is automatically used. The default value is 20.

Tgif.MenuColsBeforeScroll: NUMBER
This specifies the maximum number of rows in a user-specifiable bitmap menu (such as the Color Menu) before a horizontal scrollbar is automatically used. The default value is 26.

Tgif.PngToXpm: STRING
The STRING specifies a command used to convert a PNG file to an XPM file. The STRING *must* contain a *%s* substring to be replaced by the full path name of the PNG file. The default is "pngtopnm %s | ppmquant 222 | ppmtoxpm".

Tgif.JpegToXpm: STRING
The STRING specifies a command used to convert a JPEG file to an XPM file. The STRING *must* contain a *%s* substring to be replaced by the full path name of the JPEG file. The default is "djpeg -gif -color 222 %s | giftopnm | ppmtoxpm".

Tgif.XpmToPng: STRING
The STRING specifies a command used to convert an XPM file to a PNG file. The STRING *must* contain a *%s* substring to be replaced by the full path name of the XPM file. The default is "xpmtoppm %s | ppmtopng".

Tgif.PngFileExtension: STRING
The STRING specifies the file extension for a PNG file. The default is "png" (lower case).

Tgif.XpmToJpeg: STRING
The STRING specifies a command used to convert an XPM file to a JPEG file. The STRING *must* contain a *%s* substring to be replaced by the full path name of the XPM file. The default is "xpmtoppm %s | cjpeg".

Tgif.JpegFileExtension: STRING
The STRING specifies the file extension for a JPEG file. The default is "jpg" (lower case).

Tgif.ProducedBy: STRING
When printing/exporting PS/EPS files, STRING will appear in a %%ProducedBy line in a exported PS/EPS file. Please include your name and e-mail address in STRING. The default is "(unknown)".

Tgif.Editor: STRING
STRING specifies a text editor to use for editing attributes. The STRING *must* contain two *%s* substrings to be replaced by the window title and the full path name of the text file. For example, you can use "xemacs -title '%s' '%s'". The default is "xterm -title '%s' -e vi '%s'".

Tgif.GoHyperSpaceInSlideShow: [true,false]
If set to ''true'', *hyperspace* mode will be entered when tgif enters the slideshow mode. The default is false.

Tgif.LineWidthIndexInSlideShow: NUMBER
This specifies the line width *index* to use when tgif is in the slideshow mode. The default value is 4.

Tgif.MaxRecentFiles: NUMBER
This specifies the maximum number of files to remember in the recently used file list. The default value is 10.

Tgif.ResetOriginOnAdvancePage: [true,false]
If set to ''true'', tgif will scroll to the left-top corner of the page when pages are advanced. The default is false.

Tgif.UseMeasureTooltip: [true,false]
If set to ''true'', the location of the cursor and the width and height of the object begin drawn/dragged/stretched will be shown in a tooltip window. This X default only takes effect if Tgif.ShowMeasurement is true. The default is false.

Tgif.MeasureTooltipXFollowMouse: [true,false]
If set to ''true'', the X position of the measurement tooptip will follow the mouse. The default is false.

Tgif.MeasureTooltipYFollowMouse: [true,false]
If set to ''true'', the Y position of the measurement tooptip will follow the mouse. The default is false.

Tgif.MeasureTooltipHorizontalPosition: [left,center,right]
Fix the X position of the measurement tooltip to the left, center, or right. This X default only takes effect if Tgif.MeasureTooltipXFollowMouse is false. The default is left.

Tgif.MeasureTooltipVerticalPosition: [top,middle,bottom]
Fix the Y position of the measurement tooltip to the top, middle, or bottom. This X default only takes effect if Tgif.MeasureTooltipYFollowMouse is false. The default is top.

Tgif.NoMinWinSize: [true,false]
If set to ''false'', tgif will have a minimum window size so that the whole panel window is alway visible. The problem with this setting is that some window manager will show the wrong window size when you resize the window. This setting is left for compatibility reasons. If set to ''true'', a side effect is that the menubar will no longer automatically wraps around when Tgif.MinimalMenubar is set to true. The default is true.

Tgif.AutoWrapMenubar: [true,false]
If set to ''true'', the menubar will automatically wrap around. If Tgif.MinimalMenubar is set to false, menubar will always wrap around automatically. The default is false.

Tgif.AutoEPSPreviewBitmap: [true,false]
If set to ''true'', when importing a PS/EPS file, tgif will automatically generate a preview bitmap if the file does not already contain one. The default is false.

Tgif.PsToXbm: STRING
STRING specifies a command used to convert a PS file to a XBM file. The STRING *must* contain a single *%s* substrings to be replaced by the full path name of the PS file. Please note that the above command usually generates a bitmap that's much larger than image in the file. Tgif automatically trims out the blank space similar to the way pbmtoepsi works. The default is "gs -q -dNOPAUSE -sDEVICE=pbm -sOutputFile=- -- "%s" | pbmtoxbm".

Tgif.TmpDirInHomeDir: [true,false]
If set to ''true'', tgif will use the $HOME/.Tgif directory as the temporary directory (unless the Tgif.TmpDir X default below is used) and the compiler option -DTMP_DIR is ignored. The default is false if the -D_TMP_DIR_IN_HOME_DIR compiler option is used. The default is true if the -D_TMP_DIR_IN_HOME_DIR compiler option is *not* used.

Tgif.TmpDir: STRING
STRING specifies a directory to be used as the temporary directory. The use of this X default is discouraged, especially if tgif is compiled with -DUSE_XT_INITIALIZE and a X resource file found in the directory search path specified by the environment varialbe $XAPPLRESDIR is used. By default, tgif uses /tmp as the temporary directory.

Tgif.ThumbnailGeometry: WIDTHxHEIGHT
This X default specifies the geometry of thumbnails. The defaul tis 160x120.

Tgif.ThumbnailPadding: NUMBER
This specifies the padding (in pixels) for thumbnail images. The default value is 8.

Tgif.ThumbnailXGap: NUMBER
This specifies the horizontal gap (in pixels) for thumbnail images. The default value is 16.

Tgif.ThumbnailYGap: NUMBER
This specifies the vertical gap (in pixels) for thumbnail images. The default value is 0.

Tgif.ThumbnailX: NUMBER
This specifies the starting x location (in pixels) for thumbnail images. The default value is 32.

Tgif.ThumbnailY: NUMBER
This specifies the starting y location (in pixels) for thumbnail images. The default value is 32.

Tgif.ShowWireSignalName: [true,false]
If set to ''false'', when connecting ports, tgif will automatically place the signal name and hide it. Otherwise, the user will be prompted to place the signal name and it will be visible. The default is true.

Tgif.LandscapePdfSetPageDevice: [true,false]
If set to ''true'', when exporting landscape PDF files, tgif will use PostScript "setpagedevice" command to rotate the generated PostScript file 270 degrees before calling ps2pdf(1). This should not be necessary (and is considered a bug in ps2pdf). In the future, this X default can be used to turn off the generation of the "setpagedevice" command when ps2pdf can handle landscape PostScript files correctly. The default is true.

Tgif.DeleteCmdAsCut: [true,false]
If set to ''true'', when Delete is selected from the Edit Menu or if <Cntrl>x is pressed, Cut will be executed instead of Delete. Pressing the <DEL> key on the keyboard will still perform a regular Delete. The default is false. *Tgif.EnableMouseWheel: [true,false]* If set to ''false'', Button4 and Button5 mouse wheel scrolling events will be ignored. The default is true. *Tgif.Btn2PopupMainMenu: [true,false]* If set to ''false'', Button2 events will not bring up the Main Menu in the canvas window. The default is true.

Tgif.NoChoiceWindow: [true,false]
If set to ''true'', no Choice and Message Windows will be shown initially. The default is false.

Tgif.UseXPmVersion1ForXPmDeck: [true,false]
The setting of this X default should depend on the setting of the Tgif.XpmDeckToGifAnim X default above. If set to ‘‘true’’, XPM1 file is generated when a deck of X11 pixmap objects is being coverted to a GIF animation file regardless of the setting of the Tgif.XPmOutputVersion X default. The default is true.

Tgif.SlideShowWindowOffsets: X_OFFSET,Y_OFFSET
The numbers specify the number of pixels to adjust for the slideshow mode. If only one value is given, both X and Y offsets are set to the same value. The default values are all 0’s.

Tgif.SlideShowBorderColor: COLORSTRING
This specifies the color to be used for the area outside of the paper boundary in slideshow mode. By default, the color of the border is the same as the background color.

Tgif.ConvertToBezierSegments: NUMBER
This specifies the number of segments used in converting a polyline/spline object to a Bezier curve. The default value is 50.

Tgif.TickMarkSize: NUMBER
This specifies the size of a tick mark to be used when tick marks are added at a vertex of a polyline/polygon/spline. The default value is 8.

ENVIRONMENT VARIABLE

TGIFPATH
This environment variable should be set such that the files, mentioned in the FILES section below, can be found.

TGIFICON
This environment variable should be set to the name of the object file to be displayed when tgif is iconified. By default, it is set to ‘‘tgificon’’. If it starts with a / character, absolute path is used; otherwise, the icon file is assumed to be $TGIFPATH/$TGIFICON.

TGIF_[Domain]
Obsoleted.

FILES

$TGIFPATH/tgificon.obj contains the default tgif icon.

$TGIFPATH/keys.obj contains a summary of the non-alphanumeric key bindings.

PROLOG/C TESTDRIVE

In the tgif distribution, there are three Prolog files which illustrate a simple Prolog driver. *tgif.pl* contains predicates for parsing tgif files (both .obj and .sym). *frontend.pl* contains predicates for talking to Prolog engines, such as that of Quintus and SISCtus, through the foreign function interface. To use frontend.pl, *frontend11.o* needs to be built (which requires the *frontend11.o* entry to be uncommented from the makefiles). Finally, *testdrive.pl* contains a program which will print out the ID files of all objects in the current drawing, if tgif is escaped with the Solve() (or #s) command. This is also a good way of finding out the structure of a tgif file (especially because the structure is not documented due to the complexity introduced to keep tgif compatible with files created by older versions).

A very simple C driver, *testdrive.c*, is also provided with the tgif distribution which perform the same function as the Prolog driver. The extra code present in this file (and not present in tgif.c) is used to illustrate how the in-memory objects and attributes can be traversed and how new objects can be created and manipulated.

SEE ALSO

latex(1L), **lpr**(1), **ghostscript**(1), **env**(1), **X**(1), **dvips**(1), **csh**(1), **pbmplus**(1), **netpbm**(1), **djpeg**(1), **bitmap**(1), **XPM**(1), **netpbm**(1), **xfontsel**(1), **xlsfonts**(1), **xgrabsc**(1), **xloadimage**(1), **xsnap**(1), **sxpm**(1), **xv**(1), **pstoepsi**(1), **Mosaic**(1), **bggen**(1), **rand**(3C), **ps2pdf**(1)

IDIOSYNCRASIES

When any of the ‘‘escape to driver’’ commands are (accidentally) executed, the current content of the drawing is saved into ‘‘tmpmodel.obj’’ if the drawing indicates that it is a .obj file; then tgif escapes to the driver and returns right away. If the drawing indicates that it is a .sym file, then the content is saved into ‘‘tmpmodel.sym’’, but tgif does not return to the driver.

The paste operation works on a cut buffer generated by tgif or by non-tgif tools (such as *xterm*). If the cut buffer is *not* generated by tgif, its content is treated as a collection of ASCII character strings, which is inserted into the current drawing as a text object (current settings for text objects are used to create the text object). If the cut buffer *is* generated by tgif, then all the current settings are ignored.

The font sizes are the *screen* font sizes (which correspond to the X fonts that are used to draw the text on the screen). They appear smaller on the printout. When a 24 point text is printed, it would correspond to about a 13.5 point PostScript text. This is because tgif treats 128 pixels as an inch, and PostScript treats 72 points as an inch.

Because characters supported by X11 and PostScript are different, not all the characters, especially in the range 128 to 255 (or \200 to \377), which are supported by X11, but are not accepted by tgif. Furthermore, in order to print the supported subset or these characters, character codes must be re-encoded. Therefore, if one would like to hack tgif to support other personalized fonts, one should be careful about the re-encoding mechanism.

The grids are not absolute; they are specified as screen pixels, and they scale with the current zoom. For example, if the grid is set at 16 pixels at maximum zoom, and if the user zooms out once, objects can be drawn, moved, or stretched at 16 screen pixel increments, but this corresponds to 32 pixels in the real coordinate system.

If the vertical text spacing is set to negative values, highlighted text will look a little strange due to XOR operations. If the vertical text spacing is set to be greater than 100 or less than -100, the panel window will not be cleared properly; to clear the panel window, the user may have to close the tgif window and then open it again.

As described in the TGIF SUBWINDOWS section, in constrained move mode, if both endpoints of a not-selected polyline lie inside the object being moved, then the whole polyline is moved. This may look strange sometimes because, for example, if one starts with a line segment pointing to an object, just moving the object will caused the line segment to be ‘‘stretched’’; however, if one eventually moves the object so that the other endpoint is also inside the object, any future movement of the object will cause the whole line segment to move (instead of just moving the original endpoint). The moving of the vertex which is the neighbor of a moved endpoint may also look strange at times. At this point, one should switch to the unconstrained move mode.

Another idiosyncrasy with respect to the constrained move is that right after duplicating an object, the constrained move is disabled temporarily because it is assumed that at this point the user would want to move the new object to a desirable position, and only after this new object is ‘‘settled down’’, the constrained move will be re-enabled. Settling down is signified by doing something other than moving the new object.

Locked objects can be deleted.

Under the Edit Menu, PasteFromFile() reads a file into the drawing. Pasting from a file is different from the normal pasting operation where copying is performed in something like xterm because tabs are automatically converted to spaces. Tabs are ignored when pasting from a file.

When printing a multipage drawing, all pages (even the ones that contains no objects) will be printed. Using the PrintOnePage() command under the Page Menu one can print the selected page (in *stacked* page layout mode, this is the current page; in *tiled* page layout mode, the user is prompted to select a visible page).

Tgif can be setup to use its own icon window (the Tgif.NoTgifIcon and the Tgif.UseWMIconPixmap X defaults must both be set to false). However, it may confuse certain window managers. So, if the effect is undesirable, one can set the Tgif.UseWMIconPixmap X defaults to true.

BUGS

There seems to be a problem with printing Courier fonts with a non-solid pen on the Apple LaserWriter. (Printing single character does seem to work fine.) As pointed out by the PostScript reference manual, Courier is a ‘‘stroked font’’, and it is usually ‘‘difficult’’ to construct character paths for such types of fonts. However, Courier fonts work fine with ghostscript(1) and dxpsview. It’s not clear how this problem can be fixed. The author recommends avoiding Courier fonts when printing in color if a non-solid pen is desired.

Arcs with arrow tips don’t look very sharp (the tip is not pointed as in open-splines with arrow tips).

At high magnifications, *stretching* arcs may cause anomalous behavior due to round off errors.

When page reduction/magnification is not set at 100%, the markings in the Ruler Windows do not correspond to real measurements. behavior due to round off errors.

Copying/pasting large objects might not work because tgif does not use the ‘‘selection’’ mechanism (yet).

If and when tgif crashes, it will try to save the current content of the drawing in a file called ‘‘EmergencySave.obj’’ (or ‘‘EmergencySave.sym’’ if the current drawing specifies a symbol object). Often, the drawing can be restored by loading the ‘‘EmergencySave.obj’’ file. Nevertheless, if the cause of the crash is that some objects are corrupted (due to programming bugs), then the ‘‘EmergencySave.obj’’ file may also be corrupted.

When launching an application, if the command does not end with the ’&’ character and the command does not terminate, tgif also hangs. In this case, kill(1) should be used to send HUP signal to the tgif process if one wants to save the content of tgif in ‘‘EmergencySave.obj’’.

The file *exec.c* may not compile properly on AIX machines. One might have to add -D_BSD to the DEFINES in either the Imakefile or Makefile.noimake.

COPYRIGHT

STATUS

The current status of tgif can be obtained from tgif’s World-Wide-Web home page at <URL:http://bourbon.cs.umd.edu:8001/tgif/>.

AUTHOR

William Chia-Wei Cheng (bill.cheng@acm.org)
<URL:http://bourbon.cs.umd.edu:8001/william/>

REFERENCES

[1] ‘‘*A Beginner’s Guild to HTML*’’, <URL:http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>.

[2] ‘‘*CGI - Common Gateway Interface*’’, <URL:http://www.w3.org/CGI/overview.html>.

[3] ‘‘*NCSA Imagemap*’’, <URL:http://hoohoo.ncsa.uiuc.edu/docs/tutorials/imagemapping.html>.

[4] ‘‘*CERN Clickable Image*’’, <URL:http://www.w3.org/hypertext/WWW/Daemon/User/CGI/HTImageDoc.html>.