The $\mathcal{EZ}$ Widget and Graphics Library

Maorong Zou
Department of Mathematics
The University of Texas at Austin
Austin, TX 78712
Email: `mzou@ma.utexas.edu`

February 2, 2001

Contents

Chapter 1

Introduction

This manual explains the application programming interface (API) for the *EZ* Widget and Graphics Library (henceforth, *EZwgl*). It serves as both a specification and a users guide. This document referes to *EZwgl* release 1.50.

EZwgl is a library of C functions based on Xlib. Its purpose is to provide a tool that simplifies the task of graphical user interface (GUI) programming under the X window system environment. If you are a C programmer and want to build a nice graphical user interface for your applications, or you want to do some graphics programming under the X window system programming environment, then *EZwgl* is for you.

EZwgl implements about thirty or so commonly used widgets. Currently implemented widgets are:

- **buttons**, push button, check button, radio button, arrow button and menu button.
- **dial**, a numerical value editor.
- **entry**, a one line text editor.
- **fancy listbox**, a multi-column listbox.
- **file selector**, a Motif^{TM} style file selection dialogue.
- **frame/labled-frame**, a simple container widget.
- **gradient bar**
- **histogram**
- **icon**, a toplevel label/frame that is invisible by your window manager.
- **label**, textual or bitmap/pixmap/image label.
- **listbox**, a simple textual item selector.
- **list tree**, a widget for displaying a tree of items.
- **menus**, popup menu and pull-down menu.
- **menubar**
- **notebook**, a container widget for grouping widgets into pages.
- **optional entry**, a comboBox.
- **pane-handle**, a special widget for dividing its parent into panes.
- **ruler**
- **sliders**, horizontal and vertical sliders.
- **spread sheet**, a simple spread sheet.
- **thumb wheel**,
- **text**, an editable text browser. Text widget supports a few hardcoded text properties: fonts, foreground/background color and bitmaps/pixmaps.

- **raw X window**, a widget that gives you a native X window.
- **work area**, a special container widget capable of managing items and other widgets.
- **lcd**, a widget mimics the liquid crystal display.
- **led**, a widget simulates the light emitting diode display.
- **statusMeter** , a widget implements the progress meter.
- **term** , a vt100 terminal emulator.
- **3D canvas**, a widget specifically for the *EZ* graphics library.
- **free label/frame**. A freely movable label/frame whose geometry is not managed by the geometry manager.

EZwgl also implements a set of about 100 OpenGL^{TM} like graphics commands that provide high-level support for 3D graphics. It currently supports

- **zbuffer**
- **doublle buffer**, supports pixmap or XImage back buffer.
- **RGB or Colormap mode**, supports a simulated RGB mode (dithering) for 8bit visuals.
- **flat or smooth shading**
- **lighting**
- **fog**
- **polygon, line and points primitives**
- **bitmap fonts and a vector font**
- **mouse and keyboard input devices**

EZwgl is designed to work on displays with an at least 8 bit deep *PseudoColor, DirectColor, TrueColor, StaticColor, GrayScale*, or *StaticGray* visual.

EZwgl has been tested on the following platforms, SunSparc/X11R5, SGI-Indigo/X11R5, Pentium-Linux/X11R6 and Alpha OSF/X11R5/X11R6. It should work on any Unix systems with X11R5/X11R6.

The current release is 1.50

1.1 Conventions

- **Naming** All functions and constants in *EZwgl* begin with the prefix `EZ_` to avoid name clashing. In this document, *EZwgl* functions are printed in a `typewriter` font.
- **Screen Distance** Screen distance in the *EZ* widget library is measured in pixels.
- **Coordinates** Coordinates are always parent relative, measured from the Upper-Left corner of the widget window.
- **Colors** Colors are specified by either the symbolic names, like **"red"**, or hexdecimal strings like **"#7f00ae"**.
- **Fonts** The *EZ* widget library adopts X's convention in naming fonts.

1.2 Getting *EZwgl*

The latest release of *EZwgl* is always available at
`http://www.ma.utexas.edu/~mzou/EZWGL/EZWGL-latest.tgz`

1.3 Credits

The following people have either sent suggestions or contributions. I hereby thank them sincerely.

```
hydra@pato.vaala.fi
piemail!arnodab@relay.NL.net
orava@nether.tky.hut.fi
danields@UTKUX.UTCC.UTK.EDU
alstrup@diku.dk
O.Bedford@uni-koeln.de
N.Hatzis@dcs.shef.ac.uk
mjr@iki.fi
manabe@Roy.dsl.tutics.tut.ac.jp
mallors@ips1.msfc.nasa.gov
wuebben@math.cornell.edu
stevel@r4002.3dem.bioch.bcm.tmc.edu
ingar@ace.ulyssis.student.kuleuven.ac.be
miguel@roxanne.nuclecu.unam.mx
arturo@nuclecu.unam.mx
jbailie@wwdc.com
DFreese@osc.uscg.mil
maderassoriano@redestb.es
turnere@MimeStar.com
fmantegazza@cea.fr
root@uxi1.uxi.com
```

The html version of this document is maintained by `arturo@nuclecu.unam.mx`.

1.4 License

EZwgl is released under the **GNU Library General Public License** and is copyrighted by Maorong Zou. For more information about the GNU LGPL, consult the file LICENSE from the *EZwgl* distribution. Below is a brief description of this license. It is an excerpt from the header of each source file.

Chapter 2

Using EZ Widgets

There are two classes of widgets in the EZ widget library: a class of primitive widgets each of which is implemented using exactly one X window, and a class of composite widgets which are made from a group of primitive widgets. *Label, frame, buttons, menu-items, entry* and *sliders* are examples of simple widgets while *text widget*, *list box*, *notebook* and *notebook-page* are composite widgets.

Widgets in *EZwgl* are event-driven. Interaction between an application and the EZ widget library is achieved via *callbacks* and *event handlers*. Typically, an *EZwgl* application contains five components: *initialization, creating and configuring widgets, registering widget callbacks and/or event handlers, mapping widgets* and *handling events*.

EZwgl supports application resources. Appearance of an *EZwgl* application is completely configurable by resource files.

EZwgl does not support event-action translations. Functionality of an *EZwgl* cannont be modified by resources.

EZwgl supports geometry management. It includes four geometry managers: row geometry manager (layout children in rows), column geometry manager (layout children in columns), row-column geometry manager (aka the grid bag geometroy manager) and a simplified version of it (the default).

There are some subtle differences between primitive widgets and composite widgets.

- For a primitive widget, events are dispatched to the widget event handler; callbacks, event handlers and client data are registered to the widget data structure. For a composite widget, however, events are dispatched to whatever children widgets that are involved; callbacks, event handlers and client data, if set, are attached to the most relevent widget. For example, in a Listbox widget, these information are automatically registered to the Internal Listbox widget.
- For a primitive widget, children widgets are the direct descendants of the widget; for a composite widget, children widgets are arranged as the children of the top-most frame of the composite widget.
- If one sends an event to a composite widget, the event goes to the top-most frame, generating some undesired results. For example, if one sends a `EZ_KEY_PRESS` event to a text widget, nothing will happen because the frame ignores key events. To really send a key event to a text widget, one has to extract the internal-text-widget component and send the event to it.

2.1 Data Types

Most of the data types in *EZwgl* are hidden from the user. There are only a few special data types an application programmer needs to be aware of. The important ones are the prototypes for callbacks and event handlers. Below is a list of these data types.

```
/* these can only be used as pointers */
typedef void       EZ_Widget;
typedef void       EZ_LabelPixmap;
typedef void       EZ_TextProperty;
typedef void       EZ_TextProp;
typedef void       EZ_Item;
typedef void       EZ_TreeNode;
typedef void       EZ_Timer;
typedef void       EZ_Bitmap;
typedef void       EZ_Pixmap;
typedef void       EZ_Input;
```

```
/* prototypes for func ptrs */
typedef  void      (*EZ_CallBack)(void *object, void *data));
typedef  void      (*EZ_InputCallback)(void *object, void *data, int fd, int mask));
typedef  void      (*EZ_EventHandler)(void *object, void *data, int etype, XEvent *event));
typedef  void      (*EZ_MessageHandler)(void *msgIn, void *client_data);
typedef  int       (*EZ_DnDEncoder)(void *object, void *data, char **msg,int *len, int *needfree));
typedef  int       (*EZ_DnDDecoder)(void *object, void *data, char *msg,int len));
typedef  int       (*EZ_SChartHandle)(EZ_Widget *ssheet, void *chart, int r1, int c1,
                                      int r2, int c2, int kind, void *data);
typedef  void      *(*EZ_DirTreeNodeCreator)(char *path, void *data);
typedef  void      (*EZ_WorkAreaGeometryManager)(void *GMData, EZ_Item **data, int nitems,
                                         int Rx,int Ry,int Rw, int Rh, int xsp, int ysp,
                                         int *w_ret, int *h_ret));

/*
 * Used to attach client data to a widget/item.
 */
typedef  union {
  char           c;
  short          s;
  int            i;
  long           l;
  unsigned char  uc;
  unsigned short us;
  unsigned int   ui;
  unsigned long  ul;
  float          f;
  void           *p;
}  EZ_UnknownDataType;

/*
 *  application resource values.
 */
typedef struct EZ_Value_ {
  int  type;   /* EZ_INT ... see below */
  union {
    char    c;    unsigned char  uc;
    short   s;    unsigned short us;
    int     i;    unsigned int   ui;
    float   f;
    void    *p;
    void    *function;
    char    *str;
    unsigned long l;
  } value;
} EZ_Value;

#define EZ_INT     0
#define EZ_FLOAT   1
#define EZ_POINTER 2
#define EZ_ULONG   3
#define EZ_LONG    4
#define EZ_UCHAR   5
#define EZ_CHAR    6
#define EZ_USHORT  7
#define EZ_SHORT   8
#define EZ_UINT    9
#define EZ_STRING  10
#define EZ_ENUM    11
#define EZ_BOOL    12
#define EZ_FUNCTION 13
#define EZ_FONTNAME 14
#define EZ_FILENAME 15

/*
 *  message
 */
typedef struct {
  Atom       messageType;
  int        messageLength;
  char       *message;
  int        messageId;    /* set by the sender, transaction serial # */
  int        replyId;
  int        isReply;
  /* src info */
  Window     sender;
  Atom       senderClass;
  Atom       senderInstance;
  int        senderIsSelf;
} EZ_Message;

/* used in the GL */
typedef void       EZ_GLImage;

typedef float      EZ_Vector[4];
typedef float      EZ_Vector3[3];
typedef float      EZ_Matrix[4][4];
typedef float      EZ_Material;
typedef float      EZ_LightSrc;
typedef float      EZ_LightModel;
```

2.2 Structure of an *EZwgl* Application

An *EZwgl* program should contain the following components.

- Include the header file `EZ.h`.
- Initialize the library by calling `EZ_Initialize` or `EZ_InitializeXrm`.
- Create and configure widgets by using `EZ_CreateWidget` or `EZ_CreateWidgetXrm`.
- Register widget callbacks and/or event handlers by using `EZ_AddWidgetCallBack` and `EZ_AddEventHandler`.
- Map your widgets by invoking `EZ_DisplayWidget` on your toplevel widgets.
- Processing events by using `EZ_EventMainLoop` or `EZ_ServiceEvents` or `EZ_WaitAndServiceNextEvent`.

2.2.1 Hello World

Listed below is a simple "Hello World" program. It demonstrates the steps outlined above for writting an *EZwgl* application. When run, this program displays a toplevel push button labeled "Hello World". If the widget window is resized, the position of the lable will be adjusted accordingly. If you press the left mouse button (Button1) over the widget window, it exits.

```
/************************* Example 1 ********************************/
#include "EZ.h"                              /* the header file      */

void cleanExit(EZ_Widget *widget, void *dat)  /* push button callback */
{
  EZ_Shutdown();/* shutdown EZWGL */
  exit(0);      /* and exit       */
}

main(int argc, char **argv)
{
  EZ_Widget *hello;

  EZ_Initialize(argc,argv,0);       /* initialize EZWGL    */

          /* create a button and set its foreground red    */
  hello = EZ_CreateWidget(EZ_WIDGET_NORMAL_BUTTON, NULL,
                          EZ_LABEL_STRING, "Hello World",
                          EZ_FOREGROUND, "red",
                          0);

  EZ_AddWidgetCallBack(hello,       /* register call back  */
        EZ_CALLBACK, cleanExit, NULL, 0);

  EZ_DisplayWidget(hello);          /* show the button     */

  EZ_EventMainLoop();               /* handle mouse inputs */
}
/************************* Example 1 ********************************/
```

2.2.2 The *EZwgl* Header File

The one and only header file `EZ.h` must be included in all source files of an *EZwgl* application.

`#include ‘‘EZ.h’’`

This files contains type definitions, symbolic constants and forward declarations needed by all applications.

2.2.3 Initialization

The first order of business of an *EZwgl* application is to initialize the *EZwgl* library. This is achieved by calling

```
EZ_Initialize(int argc, char **argv, int initGL);
```
or

```
EZ_InitializeXrm(char *className,
                 XrmOptionDescRec *optsTable,  int noptions,
                 int *argc_in_out,  char **argv_in_out,
                 char *fallbackResrc,  int initgl);
```

These two functions parse the command line arguments, opens the X server connection and initialize the internal data structure for *EZwgl*.

2.2.4 Creating and Configuring Widgets

The next two functions are responsible for allocating and initializing widgets.

```
EZ_Widget *EZ_CreateWidget(int type, EZ_Widget *parent, ...);

EZ_Widget *EZ_CreateWidgetXrm(int type, EZ_Widget *parent,
                    char *instanceName, char *ClassName, ...);
```

These functions create a widget of the given type and initialize widget using the supplied configuration options. They are also responsible for scanning the resource database and configure the widget according to its resources. Configuration options passed to these two functions can be regarded as fallback resources as they will be overriden by application defaults.

2.2.5 Registering Callbacks and Event Handlers

Callbacks and event handlers are the only means that an application interacts with EZ widgets. When the user acts on a widget, e.g. pressing a checkButton, the callbacks or the event handlers for the corresponding widget, if any, will be invoked.

Callbacks and event handlers are registered by using

```
void EZ_AddWidgetCallBack(EZ_Widget *widget, int which_list,
                EZ_CallBack callback, void *data,int where);

void EZ_AddEventHandler(EZ_Widget *widget, EZ_EventHandler hdlr,
                        void *data, int where);
```

We have already used callbacks in our first example. The procedure `cleanExit` will be executed when the left mouse button is pressed and released over the button.

There are four procedure lists for each widget.

- a list of normal callbacks,
- a list of destroy callbacks,
- a list of motion callbacks, and
- a list of event handlers.

Normal callbacks are invoked when the default action has occured on a widget, e.g., press a push button, a double click on a listbox item or drag a slider button.

Destroy callbacks are invoked at the time a widget is being destroyed. This is useful if your application attaches a complex data structure to a widget, it provides a convenient way to clean up your complex data structure associated with a widget, e.g. free allocated memory.

Motion callbacks are available only for Entry, Slider, Scrollbar, Dial, OptionalEntry, FreeLabel, ListBox, FancyListBox, WorkArea, ListTree, gradient bar, histogram and spread sheet widgets. They are invoked whenever the text changes or the selection of a widget changes or the location of a FreeLabel changes etc.

Event handlers are used to define or alter the behavior of a widget to user events. Private event handlers are called before the default event handler. A private event handler may remove an event it handles by using `EZ_RemoveEvent`, which prevents the event in question from being dispatched to other handlers, including the default handler.

2.2.6 Prototypes for Callbacks and Event Handlers

Normal callbacks, motion callbacks and destroy callbacks have the following prototype:

```
void callback(EZ_Widget *widget, void *data)
```

where `widget` is the widget that `callback` is registered to. `data` is a pointer to a client data that will be passed to `callback` when invoked.

Event handlers have the prototype

```
void eventHandler(EZ_Widget *widget, void *data,
                          int eType, XEvent *xevent);
```

where `widget` is the widget that `eventHandler` is registered to; `data` is a pointer to a client data that will be passed to the event handler; `eType` is the internal EZ event type; and `xevent` is the pointer to the actual X event, it contains all the information about the event. Normally, actions of your event handler should be based on `xevent`. However, if your event handler just wants to check whether a key has been pressed or the the state of a button, you may find that using `eType` is more convenient than deciphering `xevent`.

2.2.7 Displaying a Widget

Once a widget has been created, it can be displayed by invoking

```
void EZ_DisplayWidget(EZ_Widget *widget);
```

The first time this function is called on a widget, it performs two tasks:

- Traverses the whole widget tree, computes and finalizes the geometry of all widgets on the widget tree.
- Creates X windows for all widgets on the widget tree and maps them to the screen.

If it is called on a widget which has been previously displayed. The behavior depends on whether or not `widget` is current viewable. If it is, then this function updates as necessory the geometry of `widget` and updates the display. If `widget` is not currently viewable, the behavior is the following:

- If `widget` is a toplevel widget, then its window is remapped.
- If `widget` is not a toplevel widget, and one ancestor of `widget` is not viewable, then this function does nothing visually.
- If `widget` is not a toplevel widget, and all its ancestors are viewable, then the widget window is remapped.

2.2.8 Processing Events

After an application has created and displayed its widgets, it is ready to process events. There are three routines that can be used for event processing.

```
void EZ_EventMainLoop(void);
void EZ_ServiceEvents(void)
void EZ_WaitAndServiceNextEvent(void)
```

The first function is the main event loop. It is an infinite loop that **waits** and processes user inputs. It will call as necessary the callbacks and event handlers registered with *EZ* widget library. It never returns.

The second function checks the event queue, processes them if there are any, and returns. This function is useful for the "polling" style programming, especially for interactive graphics applications.

The third function checks the event queue, if there are no events on the queue, it waits for the next event to arrive. It processes the first event on the queue and then returns.

2.3 Compling and Linking

The *EZ* widget and graphics library is named `libEZ.**`. *EZwgl* applications need to link with this library, the standard X11 library, the X extension library, the math library and the C library. A sample complile line would look like:

```
cc -o Example1 Example1.c -lEZ -lX11 -lXext -lm
```

If *EZwgl* is compiled with jpeg/png/tiff support enabled, you also need to link your program with the relevent image libraries.

2.4 Running an *EZwgl* Application

When invoking an *EZwgl* program, you may pass a subset of the following common *command line options*:

- `-display DISPLAY_NAME`

  This option sets the display name. It is the most common way to temporarily override the `DISPLAY` environment.

- `-geometry WIDTHxHEIGHT±XOFFSET±YOFFSET`

  This option sets the geometry and/or location of your application's main window. The `WIDTHxHEIGHT` part specifies a prefered dimension and the `±XOFFSET±YOFFSET` specifies a prefered location. You may pass only a subset of the geometry specification. For example, the following specifications are all valid.

  - `-geometry 300x200+10+10`
  - `-geometry 300x200`
  - `-geometry 300`
  - `-geometry x200`
  - `-geometry 300+10-10`

  - `-geometry x200+10-10`
  - `-geometry -10+10`
  - `-geometry -10+`
  - `-geometry +10`

- `-visual VISUAL_TYPE`
  This option selects a visual type supported by your X server. Supported visual types are:
  - `PseudoColor`
  - `TrueColor`
  - `DirectColor`
  - `Grayscale`
  - `StaticColor`
  - `StaticGray`

- `-privateColormap`

  This option is relevent only for PseudoColor/grayScale visuals. It tells *EZwgl* to allocate a private colormap instead of using the default colormap of your display. It is useful when your application is color hungry, (e.g. an image viewer).

- `-readWriteColormap`

  This option is relevent only for PseudoColor/grayScale visuals. It tells *EZwgl* to use the default colormap of your display and allocate read/write color cells. Use this option if you need exact colors and don't want to use a private colormap. EZWGL will switch to the default colormap if it cannot allocate a minimum of 44 read/write color cells.

- `-background COLOR` or `-bg COLOR`

  This option sets the **default** background color of your widgets.

- `-foreground COLOR` or `-fg COLOR`

  This option overrides the **default** foreground color of your widgets. The default foreground color is black.

- `-iconic`

  This option informs *EZwgl* to start the application in *iconic* mode. This option only affects your first toplevel widgets. It has no affects if the `initial_state` window manager hint is explicitly set.

- `-noBackingStore`

  This option informs *EZwgl* not to set the **backing store** attributes for all widget windows.

- `-xrmfile resourceFile`

  This option tells an application to read the application resources from the specified file.

- `-xrm resourceString`

  This option sets application resources from the specified resource string.

- `-name  name`

  This option alters the default application name.

- `-dispXrmDB`

  Use this option to print the application resources database. It is meant for debugging only.

- `-synchronous`

  This option tells *EZwgl* to run in synchronous mode (for debugging).

2.5 Resources

Resources are used to customize applications by the user. *EZwgl* supports widget resources. The appearance of widgets is completely customizable by resources. However, the functionality of widgets cannot be altered via resources. *EZwgl* does not support event-action translations.

2.5.1 The Resources Aware Functions

Three new routines were introduced in version 1.40. `EZ_InitializeXrm`, `EZ_CreateWidgetXrm`. and `EZ_CreateItemXrm`. They are the resources-aware replacements to `EZ_Initialize`, `EZ_CreateWidget` and `EZ_CreateItem`. An application should use these resource-aware functions in order to use the resource manager fully.

2.5.2 Class and Instance Names

An application has a ClassName and an instanceName. The class name is set by EZ_InitializeXrm. It is normally the application name with the first letter captalized, unless the first letter is 'X' or 'x', in which case the first two letters are capitalized. The instance is the command name, or any string set by the -name command line option. These two names are used by the resource manager to identify the application resources for the application.

The command line option -className can be used to set the class name.

Each widget has a ClassName and a instanceName also. The two names are introduced solely for the purpose of using Xlib's resources manager. They otherwise have nothing to do with the structure of the library. Consequently, one can use “class name” to group an arbitary set of widgets together and use their class name to set a common attribute, like foreground color.

Widget class and instance names are set by EZ_CreateWidgetXrm. For widgets implemented internally, the class and instance names can also set by using EZ_CreateWidget with the EZ_NAME and EZ_CLASS attributes set.

2.5.3 Default Widget Class and Instance Names

The default widget class and instance names are listed in the following table.

WidgetType	ClassName	instanceName
EZ_WIDGET_FRAME	Frame	frame
EZ_WIDGET_LABEL	Label	label
EZ_WIDGET_NORMAL_BUTTON	Button	button
EZ_WIDGET_CHECK_BUTTON	CheckButton	checkButton
EZ_WIDGET_RADIO_BUTTON	RadioButton	radioButton
EZ_WIDGET_MENU_BUTTON	MenuButton	menuButton
EZ_WIDGET_MENU_SEPARATOR	MenuSeparatorItem	menuSeparatorItem
EZ_WIDGET_MENU_SUBMENU	MenuSubmenuItem	menuSubmenuItem
EZ_WIDGET_MENU_NORMAL_BUTTON	MenuButtonItem	menuButtonItem
EZ_WIDGET_MENU_CHECK_BUTTON	MenuCheckButton	menuCheckButton
EZ_WIDGET_MENU_RADIO_BUTTON	MenuRadioButton	menuRadioButton
EZ_WIDGET_VERTICAL_SLIDER	VerticalSlider	verticalSlider
EZ_WIDGET_HORIZONTAL_SLIDER	HorizontalSlider	horizontalSlider
EZ_WIDGET_ENTRY	Entry	entry
EZ_WIDGET_3D_CANVAS	ThreeDCanvas	threeDCanvas
EZ_WIDGET_VERTICAL_SCROLLBAR	VerticalScrollbar	verticalScrollbar
EZ_WIDGET_HORIZONTAL_SCROLLBAR	HorizontalScrollbar	horizontalScrollbar
EZ_WIDGET_ILIST_BOX	IListBox	iListBox
EZ_WIDGET_LIST_BOX	ListBox	listBox
EZ_WIDGET_ITEXT	IText	iText
EZ_WIDGET_TEXT	Text	text
EZ_WIDGET_NOTE_BOOK	NoteBook	noteBook
EZ_WIDGET_NB_PAGE	NoteBookPage	noteBookPage
EZ_WIDGET_NW_LABEL	NWLabel	nWLabel
EZ_WIDGET_PANE_HANDLE	PaneHandle	paneHandle
EZ_WIDGET_ARROW_BUTTON	ArrowButton	arrowButton
EZ_WIDGET_EXECUTOR	Executor	executor
EZ_WIDGET_OPTIONAL_ENTRY	OptionalEntry	optionalEntry
EZ_WIDGET_FILE_SELECTOR	FileSelector	fileSelector
EZ_WIDGET_ITREE	ITree	iTree
EZ_WIDGET_LIST_TREE	Tree	tree
EZ_WIDGET_IFANCY_LIST_BOX	IFancyListBox	iFancyListBox
EZ_WIDGET_FANCY_LIST_BOX	FancyListBox	fancyListBox
EZ_WIDGET_IWORK_AREA	IWorkArea	iWorkArea
EZ_WIDGET_WORK_AREA	WorkArea	workArea
EZ_WIDGET_RAW_XWINDOW	RawXWindow	rawXWindow
EZ_WIDGET_MENU_TEAR_OFF_BAR	TearOffBar	tearOffBar
EZ_WIDGET_LCD	Lcd	lcd
EZ_WIDGET_LED	Led	led
EZ_WIDGET_MENU	Menu	menu
EZ_WIDGET_FREE_LABEL	FreeLabel	freeLabel
EZ_WIDGET_EMBEDER	Embeder	embeder
EZ_WIDGET_STATUS_METER	StatusMeter	statusMeter
EZ_WIDGET_MENU_BAR	MenuBar	menuBar
EZ_WIDGET_TERM	Term	term
EZ_WIDGET_ITERM	ITerm	iTerm
EZ_WIDGET_HORIZONTAL_RULER	HorizontalRuler	horizontalRuler
EZ_WIDGET_VERTICAL_RULER	VerticalRuler	verticalRuler
EZ_WIDGET_DIAL	Dial	dial
EZ_WIDGET_GRADIENT_BAR	GradientBar	gradientBar
EZ_WIDGET_HISTOGRAM	Histogram	histogram
EZ_WIDGET_SPREAD_SHEET	SpreadSheet	spreadSheet
EZ_WIDGET_THUMB_WHEEL	ThumbWheel	thumbWheel

2.5.4 Resource Files

When the application statrs up, EZ_InitializeXrm reads and mergs resource files from the following sources, in this order:

- The fallback resource string passed to EZ_InitializeXrm.

- The file `/usr/lib/X11/app-defaults/`**ClassName**.
- The file `XAPPLRESDIR/`**ClassName** and the file `XUSERFILESEARCHPATH/`**ClassName**.
- Resources set by the string `RESOURCE_MANAGER`.
- Resources set by the string `SCREEN_RESOURCE`.
- The file `HOME/.Xdefaults`
- The file `XENVIRONMENT/.Xdefaults` or `HOME/.Xdefaults-`**HOST**.
- Resource file specified by the `-xrmfile` command line option.
- Resources set on the command line.

2.5.5 Forms of Resource Specifications

Resources are specified by name/value pairs. There are two type of resources: application resources and widget resources. The syntax for specifying an application resource is like:

`application_name.resource_name: value`

The next table lists the set of internally defined application resource names and their values.

Resource Name	Default value
"labelFont"	"-Adobe-Helvetica-Bold-R-Normal--*-120-*-*-*-*-*-*",
"textFont"	"-*-Courier-Medium-R-Normal--*-120-*-*-*-*-*-*",
"menuTitleFont"	"-Adobe-Times-Bold-I-Normal--*-120-*-*-*-*-*-*",
"highlight"	true
"sliderDepression"	true
"labelOffset"	true
"ditherImage"	false
"bubbleHelp"	off
"initialState"	normalState
"visual"	
"name"	argv[0]
"noBackingStore"	false
"privateColormap"	false
"readWriteColormap"	false
"focusHighlightColor"	"black"
"foreground"	"black"
"background"	"gray74"

Specifying a widget resource is a bit more complicated, since there may be several instances of the same widget in one application. Consequently, one must use the whole pathname of a widget, starting with the application name, to specify a widget resource. For example, the line

`example1.box.label.labelString: Welcome!`

sets the lable of a widget named *label*, which is a child of a widget named *box* which is a toplevel widget in an application called *example1*, to the string *"Welcome!"*.

There is a wildcard syntax which may be used to simplify resources specifications. One may use an asterisk instead of a dot and omitting one or more terms of the full pathname. For example, the above line may be simplified as

`*label.labelString: Welcome!`

For a complete description of resource specification and matching, please consult the "Xlib Programming Manual" by Adrian Nye.

2.5.6 Widget Resource Names

Each configuration options has a corresponding resource name. Some resources apply to all widgets while others are relevent only to some widgets. Please consult the relevent sections in this manual for a complete list of resources available for each widget.

2.5.7 Standard Command Line Options

All *EZwgl* applications accept the set of standard command line options listed in the next table. At startup time, the command line is parsed first in search of the standard options first, matching options are removed. The altered command line is then send to `XrmParseCommand` to match against the command line option table passed to `EZ_InitializeXrm`.

```
static XrmOptionDescRec EZ_StandardCommandLineOptions[] = {
  {"-display",          ".display",          XrmoptionSepArg,  NULL},
  {"-visual",           ".visual",           XrmoptionSepArg,  NULL},
  {"-geometry",         ".geometry",         XrmoptionSepArg,  NULL},
  {"-background",       ".background",       XrmoptionSepArg,  NULL},
  {"-bg",               ".background",       XrmoptionSepArg,  NULL},
  {"-foreground",       ".foreground",       XrmoptionSepArg,  NULL},
  {"-fg",               ".foreground",       XrmoptionSepArg,  NULL},
  {"-iconic",           ".initialState",     XrmoptionNoArg,  "IconicState"},
  {"-dualLabel",        ".dualLabel",        XrmoptionNoArg,  "True"},
  {"-noBackingstore",   ".noBackingStore",   XrmoptionNoArg,  "True"},
  {"-privateColormap",  ".privateColormap",  XrmoptionNoArg,  "True"},
  {"-readWriteColormap",".readWriteColormap", XrmoptionNoArg,  "True"},

  /* the next five options are for debugging only */
  {"-xrm",             NOTHING,            XrmoptionSepArg,  NULL},
  {"-xrmfile",         NOTHING,            XrmoptionSepArg,  NULL},
  {"-className",       NOTHING,            XrmoptionSepArg,  NULL},
  {"-name",            NOTHING,            XrmoptionSepArg,  NULL},
  {"-sychnronous",     NOTHING,            XrmoptionNoArg,   NULL},
};
```

2.5.8 Application Resources

EZwgl provides a limited support for application resources. In particular, there is no automatic type conversion facility as seen in Xt. *EZwgl* can only convert application resources to one of the four basic data types, namely `int,float,ulong` and `string`. Consequently, if you need to convert resources to other data types, you must provide ways to convert resource values (strings) to the desired data types.

Only global variables can be set via application resources.

Automatic Resource Value Convertion

To convert resource values to one of the four basic data types. One first define a list of resource specifications and a procedure specifies how the resource values will be used, then call `EZ_RegisterApplicationResources` to register the resources. This function must be called right after the library has been initialized.

A resource specification is a special data structure of type

```
typedef struct {
  int     option;     /* reference id. must > 1024 */
  char    *iname;     /* instance name */
  char    *cname;     /* class name, may be NULL */
  char    *vtypes;    /* value type, e.g  "float, float" */
} EZ_ResourceSpec;
```

Here is an example.

```
static int   int1 = 0,   int2 = 0; /* var set by resources */
static float float2 = 0;
static  EZ_ResourceSpec exampleResources[] =
{
   { 1234, "aPairOfInt",  "APairOfInt", "int,int"   },
   { 1235, "intFloat",    "IntFloat",   "int,float" },
};
```

The configure procedure looks like

```
int configure(EZ_Widget *notused, int option, EZ_Value *values)
{
  switch(option)
   {
     case 1234:  int1 = values[0].value.i; int2 = values[1].value.i; break;
     case 1235:  int1 = values[0].value.i; float2 = values[1].value.f; break;
     default:  break;
   }
}
```

And here is how the resources above should be used.

```
#include "EZ.h";
main(int ac, char **av)
{
  EZ_Initialize(ac, av, 1);
  EZ_RegisterApplicationResources(2, exampleResources, configure);
  printf("int1=%d, int2=%d, float2=%f\n", int1, int2, float2);
  /* more stuff ... */
}
```

Using Symbolic Constants

EZwgl includes a few functions that may be helpful for manipulating symbolic names.

One special data type is introduced for associating symbolic names with numerical values.

```
typedef struct EZ_Value_ {
  int  type;    /* value type: EZ_INT, EZ_FLOAT, EZ_POINTER, EZ_ULONG */
  union {
    char    c;    unsigned char  uc;
    short   s;    unsigned short us;
    int     i;    unsigned int   ui;
    float   f;
    void    *p;
    void    *function;
    char    *str;
    unsigned long l;
  } value;
} EZ_Value;
```

The few relevent functions are:

`void   EZ_InstallSymbol(char *name, EZ_Value value);`

install the `name-value` pair in an internal table for later lookup.

`EZ_Value  *EZ_RetrieveSymbol(char *name);`

retrieve the value assigned to `name`. Return a pointer or NULL.

`int EZ_GetApplicationResource(char *rname, char **value_ret);`

retrieve the resource value assigned to `rname`. Return True or False.

`int EZ_GetWidgetResource(EZ_Widget *wgt, char *rname, char **v_ret);`

retrieve the widget resource value for `rname`. Return True or False.

`char *EZ_GetInstanceName();`
`char *EZ_GetClassName();`

retrieve the instanceName and className.

There are also four pairs of convenience routines for `EZ_InstallSymbol`, then are

```
int   EZ_InstallSymbolicInt       (char *name, int value);
int   EZ_RetrieveSymbolicInt      (char *name, int *value_ret);

int   EZ_InstallSymbolicFloat     (char *name, float value);
int   EZ_RetrieveSymbolicFloat    (char *name, float *value_ret);

int   EZ_InstallSymbolicPointer   (char *name, void *value);
int   EZ_RetrieveSymbolicPointer  (char *name, void **value_ret);

int   EZ_InstallSymbolicULong     (char *name, unsigned long value);
int   EZ_RetrieveSymbolicULong    (char *name, unsigned long *value_ret);
```

An Example

Suppose that in a hypothetical application we need to get the variable `shape` from the application resource named *shape*. The variable `shape` takes only 4 different values: `triangle`, `square`, `pentagon` and `hexagon`. These values are represented by, say, `0, 1, 2` and `3`. We can specify these values in the resource file, but it is more readable if we use the names instead. The following code segement does the trick.

```
#include <EZ.h>
#define TRIANGLE      0
#define SQUARE        1
#define PETAGON       2
#define HEXAGON       3

static  int   shape = TRIANGLE;

static EZ_ResourceSpec testResources[] =
{
  {1236,  "shape", "Shape", "int"},
};

static int configShape(EZ_Widget *notused, int option, EZ_Value *values)
{
  if(option == 1236)
    {
      int tmp = values[0].value;
      if(tmp >= 0 && tmp <= 3) shape = tmp;
    }
}
```

```
main(int argc, char **argv)
{
   char  *value_ret;

   EZ_InitializeXrm("Test", NULL, 0, &argc, argv, NULL, 0);
   /* install 4 name-value pairs */
   if( EZ_InstallSymbolicInt("triangle", TRIANGLE) < 0 ||
       EZ_InstallSymbolicInt("square",   SQUARE) < 0   ||
       EZ_InstallSymbolicInt("petagon",  PETAGON)< 0   ||
       EZ_InstallSymbolicInt("hexagon",  HEXAGON)< 0 )
   fprintf(stderr,"InstallSymbol Failed");  /* symbol is already in use */

   EZ_RegisterApplicationResources(1, testResources, configShape);

   ... ...

}
```

This code will set the value of `shape` to `2` if the resource file contains a specification

```
Test.shape:  pentagon
```

2.5.9 Internally Defined Symbolic Names

The following (integer) symbols are defined internally.

no	0	bottom	4
yes	1	topLeft	5
No	0	topRight	6
Yes	1	bottomLeft	7
null	0	bottomRight	8
Null	0	empty	0
false	0	square	1
true	1	sunkenSquare	2
False	0	diamond	3
True	1	sunkenDiamond	4
on	1	circle	5
off	0	check	6
On	1	rectangle	7
Off	0	mbutton	8
none	0	tick	9
None	0	row	1
nil	0	column	2
horizontal	1	allRows	4
horizontalLeft	2	allColumns	8
horizontalRight	3	flat	1
horizontalCenter	4	ridge	2
vertical	5	groove	3
verticalTop	6	raised	4
verticalBottom	7	sunken	5
verticalCenter	8	embossed	8
fillNone	0	engraved	9
fillHorizontally	1	shadow	22
fillVertically	2	shadow	23
fillBoth	3	normalState	1
center	0	iconicState	3
left	1	up	10
right	2	down	11
top	3	leftRight	12
upDown	13		

2.5.10 When Are Widget Resources Set

When you create a widget using `EZ_CreateWidgetXrm`, the arguments are processed first, then the resource database is searched for widget resources and relevent resource values are set.

In other words, attributes set by the arguments of `EZ_CreateWidgetXrm` may be thought as the fallback values for the corresponding resources.

To hardcode an widget attribute, use `EZ_ConfigureWidget`. Attributes set by this function overrides the corresponding attributes set by resources (unless you rescan the resource database).

2.5.11 Resources and Composite Widgets

Setting resources for a composite widget is a little more involved, since there is no way to set the instance/class names for its components at the creation time. After `EZ_CreateWidget***` returns, the resource database has already been scaned with relevent resources applied.

There are two ways to set resources for components of a composite widget.

Scan Resource Database At a Later Time

In *EZwgl*, one can scan the resource database anytime after initialization. The procedures responsible for searching the database for widget resources are

```
void EZ_GetAndUseWidgetResources(EZ_Widget *widget);
void EZ_GetAndUseWidgetResources1(EZ_Widget *widget);
```

When invoked, they scans the resource database for resources relevent to widget and set the resource (attribute) values accordingly. The first function scans and sets resources for descendants widgets recursively.

To set resources for components of a composite widget, one first extract the components of the composite widget using `EZ_Get***Components`, then invoke `EZ_ConfigureWidget` on the components to give them an instanceName or ClassName, then call `EZ_GetAndUseWidgetResources` on the composite widget to rescan and set the resources values.

Use the Builtin instance/Class Names

Except fileSelector and optionalEntry, the components of all other composite widgets has the default widget instanceName and ClassName. These default names are explicit enough to specify all subwidgets. For example, a text widget consists of three components: an internal-text-widget, a h-scrollbar and a v-scrollbar. The path names for these components are:

```
text_widget_name.iText
text_widget_name.horizontalScrollbar
text_widget_name.verticalScrollbar
```

Components of an OptionalEntry

OptionalEntry presents a unique problem, it contains a toplevel popup listbox, which is not a descendant of the the optionalEntry. The builtin name for the popup listbox is `popup` (ClassName `Popup`). *EZwgl* provides a configuration option for optionalEntry, namely, `EZ_POPUP_NAME`, to override this name. The componet names are listed below.

```
                ( entry
                (
 optionalEntry-->( arrowButton
                (         ( iListBox
          ....... popup-->( horizontalScrollbar
                          ( verticalScrollbar
```

Components of a FileSelector

Since there are repetitions of buttons and entries on a fileSelector widget, using the default widget instanceName is not sufficient in distinguishing components. The builtin instanceNames for all components of a fileSelector is listed below.

```
                (           ( filterLabel
                (           (                          ( iListBox
                (  frameA-->( filterEntry ...popupA-->( horizontalScrollbar
                (                                      ( verticalScrollbar
                (          (           ( directoryLabel
                (          (           (                     ( iListBox
                (          ( frameB1-->( directoryListBox-->( horizontalScrollbar
                (          (                                 ( verticalScrollbar
                (          (           ( fileLabel
                (          (           (               ( iListBox
 fileSelector-->( frameB-->( frameB2-->( fileListBox-->( horizontalScrollbar
                (          (           (               ( verticalScrollbar
                (
                (          ( selectionLabel
                (          (                           ( iListBox
                ( frameC-->( selectionEntry ...popupB-->( horizontalScrollbar
                (                                      ( verticalScrollbar
                (          ( okButton
                ( frameD-->( filterButton
                (          ( hideButton
```

The instance names for the popup listboxes can be oriverriden by the configuration option `EZ_POPUP_NAME`. *EZwgl* appends the letter `A` and `B` at the end of the configured name and assign them to the two popups.

Components of Notebook and NoteBookPage

The notebook widget is made from three frames. The outermost frame is the notebook. The two inner frames are named `topFrame` and `bottomFrame`. `topFrame` is the container for page tabs.

The notebook page widget can only be the child of a notebook widget. A notebook page consists of two widgets, a button and a frame. They frame is called `notebookPage`. The two subwidgets are actually descendants of the notebook widget.

```
           ( topFrame--> notebookPageTab
notebook-->(
           ( bottomFrame--> notebookPage
```

2.6 Geometry Management

The purpose of the geometry manager is to automate widget layout. Each widget in *EZwgl* has attributes that control the placement of children widgets as well as attributes to hardcode the geometry of widgets. Unless absolutely necessary, it is recommended that you do not hard code the geometry of your widgets.

EZwgl implements four geometry managers: layout widget in rows/columns/grid-bag and a simplified grid-bag (the default). The geometry manager is controled by the `EZ_GEOMETRY_MANAGER` configuration option or the `geometryManager` resource. Possible resource values are: -1 (the default), 0 (grid bag), 1 (row), 2 (column) and 3 (none).

2.6.1 The Default Geometry Manager

The default geometry manager arranges children widgets in one row or one column. To compute the layout, it uses the following set of attributes of container widgets

- `EZ_ORIENTATION` (how to stack subwidgets)
- `EZ_FILL_MODE` (how to distribute extra spaces)
- `EZ_SIDE` (how to align subwidgets)
- `EZ_PADX` (padding in the horizontal direction)
- `EZ_PADY` (padding in the vertical direction)
- `EZ_IPADX` (padding between subwidgets, in the horizontal direction)
- `EZ_IPADY` (padding between subwidgets, in the vertical direction)
- `EZ_WIDTH`
- `EZ_HEIGHT`

and the following set of attributes of children widgets

- `EZ_EXPAND`
- `EZ_WIDTH`
- `EZ_HEIGHT`
- `EZ_X`
- `EZ_Y`

2.6.2 The Grid Bag Geometry Manager

The grid bag geometry manager arranges children widgets in multiple rows and columns. It is more powerful and flexible than the default geometry manager. The down side is that it is not very intuitive and can be extremly complex.

The grid bag geometry manager uses the row/column contrains of the container widget and cell geometry attributes of children widgets to compute the layout. Row/column contrains are set by the following configuration options

- `EZ_GRID_COLUMN_CONSTRAINS`
- `EZ_GRID_ROW_CONSTRAINS`

Cell geometry of children widgets are set by the following two configuration options

- `EZ_GRID_CELL_GEOMETRY`
- `EZ_GRID_CELL_PLACEMENT`

2.6.3 The Row and Column Geometry Manager

Row/column geometry manager layout children widgets in rows/columns inside the parent widget window. The size of children widgets does not depend on the geometry of the parent, only the location does.

2.6.4 About EZ_PROPAGATE

Geometry managers in *ℰ𝒵wgl* use a recursive algorithm to compute the layout of widgets. If the geometry of a widget changes, the change by default propagates to the whole widget tree. If your interface has many widgets packed as descendants of one toplevel widget, the perfermence may not be satisfactory if the geometry of leave widgets change constantly. Updating the geometry and move/resize/redisplay all widget windows can be slow. This is especially noticable if the application is running on a different machine across the network.

The attribute `EZ_PROPAGATE` is designed to alter the the default behavior about propagating geometry changes. When set `True` for a widget, change of geometry of the widget will no longer propagate.

Chapter 3

Widget Resources

3.1 Description of Pre-defined Widget Resources

In this section, we describe briefly all widget resources and configuration options defined in *EZwgl*.

- `EZ_X, EZ_Y, EZ_LOCATION`

configOption	resourceName	valueType
`EZ_X`	`x`	`int`
`EZ_Y`	`y`	`int`
`EZ_LOCATION`	`location`	`int_x int_y`

These options set the position of the widget window. Coordinates are parent relative, measured in pixels from the upper-left corner of the widget window. In general, one should not hardcode the position of widgets. It will be determined by the geometry manager.

- `EZ_WIDTH, EZ_HEIGHT, EZ_SIZE`

configOption	resourceName	valueType
`EZ_WIDTH`	`width`	`int`
`EZ_HEIGHT`	`height`	`int`
`EZ_SIZE`	`size`	`int_w int_h`

These options set the dimension of a widget, in pixels. **Configured dimension will always be honored** even if it is smaller than the minimal size required to hold its content.

- `EZ_GEOMETRY`

configOption	resourceName	valueType
`EZ_GEOMETRY`	`geometry`	`int_x int_y int_w int_h`

This option combines `EZ_LOCATION` and `EZ_SIZE`. It sets the geometry of a widget. When set by resource, it also takes the standard geometry specification `WIDTHxHEIGHT±XOFFSET±YOFFSET`

- `EZ_WIDTH_HINT, EZ_HEIGHT_HINT, EZ_SIZE_HINT`

configOption	resourceName	valueType
`EZ_WIDTH_HINT`	`widthHint`	`int`
`EZ_HEIGHT_HINT`	`heightHint`	`int`
`EZ_SIZE_HINT`	`sizeHint`	`int_w int_h`

These options sets the size hint of a widget. Size hint is not guaranteed to be honored.

- `EZ_BORDER_TYPE, EZ_BORDER_WIDTH`

configOption	resourceName	valueType
`EZ_BORDER_TYPE`	`borderType`	`enum`
`EZ_BORDER_WIDTH`	`borderWidth`	`int`

These two options set the border of a widget. Border types are listed in the following table.

SymbolicConstant	resourceValue
EZ_BORDER_NONE	none
EZ_BORDER_FLAT	flat
EZ_BORDER_RIDGE	ridge
EZ_BORDER_GROOVE	groove
EZ_BORDER_RAISED	raised
EZ_BORDER_SUNKEN	sunken
EZ_BORDER_EMBOSSED	embossed
EZ_BORDER_ENGRAVED	engraved
EZ_BORDER_SHADOW	shadow
EZ_BORDER_SHADOW3	shadow3

- `EZ_PADX, EZ_PADY, EZ_FOCUS_PAD`

configOption	resourceName	valueType
EZ_PADX	padx	int
EZ_PADY	pady	int
EZ_FOCUS_PAD	focusPad	int

`EZ_PADX` and `EZ_PADY` set the horizontal and the vertical paddings between the widget border and its contents, respectively. `EZ_FOCUS_PAD` sets the padding outside the widget border for highlighting keyboard focus.

- `EZ_FOREGROUND, EZ_BACKGROUND`

configOption	resourceName	valueType
EZ_FOREGROUND	foreground	string
EZ_BACKGROUND	background	string

`EZ_FOREGROUND` sets the foreground color of a widget. The foreground color is used to color textual labels and bitmap labels. `EZ_BACKGROUND` sets the background color of a widget. For each configured background color, *EZwgl* allocates three additional colors for rendering borders. Configured background colors will be honored only when *EZwgl* can allocate all four colors. Background colors propagates, i.e., children widgets inherit parent's background, unless a different background color is set for them.

- `EZ_BG_IMAGE_FILE, EZ_BG_IMAGE_FILE_B`

configOption	resourceName	valueType
EZ_BG_IMAGE_FILE	bgImageFile	string
EZ_BG_IMAGE_FILE_B	bgImageFileB	string
EZ_BG_PIXMAP	N/A	EZ_Bitmap *
EZ_BG_PIXMAP_B	N/A	EZ_Bitmap *

`EZ_BG_IMAGE_FILE` sets the pixmap to tile the background of a widget. `EZ_BG_IMAGE_FILE_B` is available for sliders, scrollbars and composite widgets that uses scrollbars. It sets an additional pixmap to tile the slider/scrollbar buttons. Like background color, these two resources propagates to children widgets.

- `EZ_ORIENTATION, EZ_FILL_MODE, EZ_SIDE`

configOption	resourceName	valueType
EZ_ORIENATTION	orientation	enum
EZ_FILL_MODE	fillMode	enum
EZ_SIDE	side	enum
EZ_IPADX	iPadx	int
EZ_IPADY	iPady	int

This resource is used by the default geometry manager to layout children widgets.

`EZ_ORIENATTION` sets the orientation of a container widget. Orientation can be one of the following.

SymbolicConstant	resourceValue
EZ_HORIZONTAL	horizontal
EZ_HORIZONTAL_LEFT	horizontalLeft
EZ_HORIZONTAL_RIGHT	horizontalRight
EZ_HORIZONTAL_CENTER	horizontalCenter
EZ_VERTICAL	vertical
EZ_VERTICAL_TOP	verticalTop
EZ_VERTICAL_BOTTOM	verticalBottom
EZ_VERTICAL_CENTER	verticalCenter

`EZ_SIDE` specifies how to align children widgets in the direction perpendicular to the orientation. It takes one of the following values.

SymbolicConstant	resourceValue
EZ_CENTER	center
EZ_LEFT	left
EZ_RIGHT	right
EZ_TOP	top
EZ_BOTTOM	bottom

EZ_IPADX and EZ_IPADY set the amount of spaces to be inserted between children widgets in the horizontal and the vertical direction respectively.

- EZ_EXPAND, EZ_PROPAGATE

configOption	resourceName	valueType
EZ_EXPAND	expand	boolean
EZ_PROPAGATE	propagate	boolean

EZ_EXPAND specifies whether or not a child should be stretched in one direction to the size of its parent. This attribute is used only by the default geomtry manager. A child widget can only be expanded in the direction perpendicular to its parent's orientation.

EZ_PROPAGATE specifies whether a change of geometry of a child should propagate to the whole widget tree.

- EZ_GRID_CONSTRAINS

configOption	resourceName	valueType
EZ_GRID_CONSTRAINS	gridConstrains	enum_row_column int_idx int_minsize int_weight int_pad

This option sets the row or column contrains. This resource is used by the grid bag geometry manager to layout children widgets. Values for enum_row_column are

SymbolicConstant	resourceValue
EZ_ROW	row
EZ_COLUMN	column
EZ_ALL_ROWS	allRows
EZ_ALL_COLUMNS	allColumns

- EZ_CELL_GEOMETRY, EZ_CELL_PLACEMENT

configOption	resourceName	valueType
EZ_GRID_CELL_GEOMETRY	gridCellGeometry	int_x int_y int_w int_h
EZ_GRID_CELL_PLACEMENT	gridCellPlacement	enum_fill_mode enum_gravity

EZ_GRID_CELL_GEOMETRY sets the starting location and the number of cells a widget occupies (in both the horizontal and the vertical directions) in a grid bag. EZ_GRID_CELL_PLACEMENT specifies the rules to place the widget in the case when the cells assigned to it are larger than its dimension.

- EZ_LABEL_STRING, EZ_FONT_NAME, EZ_TEXT_LINE_LENGTH

configOption	resourceName	valueType
EZ_LABEL_STRING	labelString	string
EZ_FONT_NAME	fontName	string
EZ_FONT_ID	fontId	int
EZ_TEXT_LINE_LENGTH	textLineLength	int

EZ_LABEL_STRING sets the textual label. EZ_FONT_NAME specifies the font. EZ_FONT_ID specifies a font id.

EZ_TEXT_LINE_LENGTH sets the length limit of one line. Long labels will be broken into multiple lines according to this limit.

- EZ_LABEL_POSITON, EZ_JUSTIFICATION

configOption	resourceName	valueType
EZ_JUSTIFICATION	justification	enum
EZ_LABEL_POSITION	labelPosition	enum

EZ_JUSTIFICATION specifies the justification of multiple line textual lables. Possible values are

SymbolicConstant	resourceValue
EZ_CENTER	center
EZ_LEFT	left
EZ_RIGHT	right

EZ_LABEL_POSITION sets the anchor location for the label. Valid values are

SymbolicConstant	resourceValue
EZ_CENTER	center
EZ_LEFT	left
EZ_RIGHT	right
EZ_TOP	top
EZ_BOTTOM	bottom
EZ_TOP_LEFT	topLeft
EZ_TOP_RIGHT	topRight
EZ_BOTTOM_LEFT	bottomLeft
EZ_BOTTOM_RIGHT	bottomRight

- `EZ_LABEL_SHADOW`

configOption	resourceName	valueType
EZ_LABEL_SHADOW	labelShadow	int_xoffset int_yoffset

`EZ_LABEL_SHADOW` specifies the way to render a shadow for a textual label. It is available for all widgets which has textual labels. The option values specify the XY offset to render the shadow.

- `EZ_PIXMAP_FILE, EZ_IMAGE_FILE`

configOption	resourceName	valueType
EZ_PIXMAP_FILE	pixmapFile	string
EZ_IMAGE_FILE	imageFile	string
EZ_BITMAP_DATA	N/A	string
EZ_PIXMAP_DATA	N/A	char **
EZ_X_PIXMAP	N/A	Pixmap
EZ_LABEL_PIXMAP	N/A	EZ_Bitmap *

`EZ_PIXMAP_FILE` makes an image lable from a XPM image file or an X11 bitmap file. `EZ_IMAGE_FILE` makes an image label from a GIF, XPM, JPEG, PNG or TIFF image file.

The other four options can only be used as configuration options. One can not set them by resources.

When an image label is set succefully. Textual label will be ignored.

- `EZ_SLIDER_LENGTH, EZ_SLIDER_WIDTH, EZ_SLIDER_BORDER_WIDTH`

configOption	resourceName	valueType
EZ_SLIDER_LENGTH	sliderLength	int
EZ_SLIDER_WIDTH	sliderWidth	int
EZ_SLIDER_BORDER_WIDTH	sliderBorderWidth	int
EZ_SLIDER_STYLE	sliderStyle	enum

`EZ_SLIDER_LENGTH` and `EZ_SLIDER_WIDTH` set the size of the slider button. `EZ_SLIDER_BORDER_WIDTH` sets the border width of a slider. `EZ_SLIDER_STYLE` selects one of the 6 kinds (small variations) of sliders. Values are one of 0–5.

- `EZ_SLIDER_RANGE, EZ_SLIDER_RESOLUTION`

configOption	resourceName	valueType
EZ_SLIDER_DISPLAY_VALUE	sliderDisplayValue	boolean
EZ_SLIDER_RANGE	sliderRange	float_min float_max
EZ_SLIDER_RESOLUTION	sliderResolution	float
EZ_SLIDER_INIT_VALUE	sliderInitialValue	float

`EZ_SLIDER_DISPLAY_VALUE` toggles display numerical values on/off. `EZ_SLIDER_RANGE` specifies the range of a slider. `EZ_SLIDER_RESOLUTION` specifies the unit of increment. `EZ_SLIDER_INIT_VALUE` sets the slider value.

- `EZ_SHOW_TICK`

configOption	resourceName	valueType
EZ_SHOW_TICK	showTicks	int, int

`EZ_SHOW_TICK` controlls the tickmarks on a slider or a dial. This resource takes a pair of integers, the first one specifies the number of ticks to be displayed, the second value specifies the length of tickmarks.

- `EZ_SCROLLBAR_WIDTH, EZ_HSCROLL_INCREMENT`

configOption	resourceName	valueType
EZ_SCROLLBAR_WIDTH	scrollbarWidth	int
EZ_SCROLLBAR_BORDER_WIDTH	scrollbarBorderWidth	int
EZ_HSCROLL_INCREMENT	hScrollIncrement	int
EZ_VSCROLL_INCREMENT	vScrollIncrement	int

EZ_HSCROLLBAR_WIDTH and EZ_HSCROLLBAR_BORDER_WIDTH set the geometry of a scrollbar. EZ_HSCROLL_INCREMENT and EZ_VSCROLL_INCREMENT set the scroll unit for the arrow buttons on the scrollbar. These resources are also available for composite widgets that use scrollbars (text, listbox, fancyListbox, tree and workarea).

- EZ_INDICATOR_TYPE, EZ_INDICATOR_COLOR

configOption	resourceName	valueType
EZ_INDICATOR_TYPE	indicatorType	enum
EZ_INDICATOR_COLOR	indicatorColor	string
EZ_INDICATOR_SIZE_ADJUST	indicatorSizeAdjust	int

These options set check button and radio button indicators. EZ_INDICATOR_TYPE specifies the kind of indicator. Must be one of

SymbolicConstant	resourceValue
EZ_EMPTY_INDICATOR	none
EZ_SQUARE_INDICATOR	square
EZ_SUNKEN_SQUARE_INDICATOR	sunkenSquare
EZ_DIAMOND_INDICATOR	diamond
EZ_SUNKEN_DIAMOND_INDICATOR	sunkenDiamond
EZ_CIRCLE_INDICATOR	circle
EZ_CHECK_INDICATOR	check
EZ_RECTANGLE_INDICATOR	rectangle
EZ_MENU_BUTTON_INDICATOR	mbutton
EZ_TICK_INDICATOR	tick

EZ_INDICATOR_COLOR sets the foreground color of the indicator. EZ_INDICATOR_SIZE_ADJUST adjusts the size of the indicator. Value must be in the range from -4 to 4.

- EZ_CHECK_BUTTON_ON_VALUE, EZ_CHECK_BUTTON_ON_OFF

configOption	resourceName	valueType
EZ_CHECK_BUTTON_ON_VALUE	checkButtonOnValue	int
EZ_CHECK_BUTTON_OFF_VALUE	checkButtonOnValue	int
EZ_CHECK_BUTTON_ON_OFF	checkButtonOnOff	boolean

These options set the value or state of a check button.

- EZ_RADIO_BUTTON_GROUP, EZ_RADIO_BUTTON_VALUE

configOption	resourceName	valueType
EZ_RADIO_BUTTON_GROUP	radioButtonGroup	int
EZ_RADIO_BUTTON_VALUE	radioButtonValue	int

These options sets the value or state of a radio button.

- EZ_CLIENT_PTR_DATA, EZ_CLIENT_INT_DATA

configOption	resourceName	valueType
EZ_CLIENT_PTR_DATA	N/A	void*
EZ_CLIENT_INT_DATA	N/A	int

The EZ_Widget structure has two slots for applications to hook in client data. These two options set the data in these fields.

- EZ_CALLBACK, EZ_EVENT_HANDLER

configOption	resourceName	valueType
EZ_CALLBACK	N/A	void* void*
EZ_MOTION_CALLBACK	N/A	EZ_CallBack void*
EZ_DESTROY_CALLBACK	N/A	EZ_CallBack void*
EZ_EVENT_HANDLER	N/A	EZ_EventHandler void*

These four options register callbacks or event handlers to a widget. Callbacks and event handlers are not settable via resources. Use EZ_AddWidgetCallBack and EZ_AddEventHandler if you want to register more than one callbacks or event handlers on a widget.

- EZ_SHORTCUT_KEY, EZ_UNDERLINE

configOption	resourceName	valueType
EZ_SHORTCUT_KEY	shortcutKey	string
EZ_UNDERLINE	underline	int

EZ_SHORTCUT_KEY sets a keyboard accelerator for a button type widget. Valid values are strings like C-a, M-x or Ctrl-s. EZ_UNDERLINE sets the keyboard accelerator from the button label. The button can be activated via Alt-underlined_char.

- EZ_LCD_NDIGITS, EZ_LCD_FONT_SIZE

configOption	resourceName	valueType
EZ_LCD_NDIGITS	lcdNDigits	int
EZ_LCD_FONT_SIZE	lcdFontSize	int
EZ_LCD_FOREGROUND	lcdForeground	string
EZ_LCD_BACKGROUND	lcdBackground	string

These options sets up a LCD, a widget displaying fixed width numerical values.

- EZ_LED_WIDTH, EZ_LED_PIXEL_SIZE

configOption	resourceName	valueType
EZ_LED_WIDTH	ledWidth	int
EZ_LED_HEIGHT	ledHeight	int
EZ_LED_PIXEL_COLOR	ledPixelColor	string
EZ_LED_BACKGROUND	ledBackground	string
EZ_LED_PIXEL_SIZE	ledPixelSize	int
EZ_LED_COLOR_FUNCTION	N/A	(ulong *)(int int)

These options sets up a LED, a widget with a fixed size that simulate a real LED.

- EZ_OPTIONAL_ENTRY_HISTORY, EZ_OPTIONAL_ENTRY_EDITABLE

configOption	resourceName	valueType
EZ_OPTIONAL_ENTRY_HISTORY	optionalEntryHistroy	boolean
EZ_OPTIONAL_ENTRY_EDITABLE	optionalEntryEditable	boolean
EZ_OPTIONAL_ENTRY_STYLE	optionalEntryStyle	enum

These three options modifies the characteristics of an optional entry. The first option toggles histroy mode on/off. If turned on, entry selections will be inserted in the popup listbox. The second option turns editing on/off. The last option selects one of the two display styles (represented by 0 and 1) for the optional entry.

- EZ_OPTIONAL_HSCROLLBAR, EZ_OPTIONAL_VSCROLLBAR

configOption	resourceName	valueType
EZ_OPTIONAL_HSCROLLBAR	optionalHScrollbar	boolean
EZ_OPTIONAL_VSCROLLBAR	optionalVScrollbar	boolean

EZ_OPTIONAL_HSCROLLBAR turns the optional horizontal scrollbar on a composite widget on/off. Likewise, EZ_OPTIONAL_VSCRO turns the optional vertical scrollbar on/off.

- EZ_HIGHLIGHT_FOREGROUND, EZ_HIGHLIGHT_BACKGROUND

configOption	resourceName	valueType
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	string
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	string

These resources specifies the colors to highlight the current selections.

- EZ_SELECTION_FOREGROUND, EZ_SELECTION_BACKGROUND

configOption	resourceName	valueType
EZ_SELECTION_FOREGROUND	selectionForeground	string
EZ_SELECTION_BACKGROUND	selectionBackground	string
EZ_TEXT_BACKGROUND	textBackground	string

These resources are available for ListBox, Text, FancyListBox, Tree and WorkArea widgets. The first two resources set the selection background and foreground. These two colors are used to highlight the relevent selections. EZ_TEXT_BACKGROUND sets the background color for the internal display window.

- EZ_HIGHLIGHT_MODE, EZ_RUBBER_BAND

configOption	resourceName	valueType
EZ_HIGHLIGHT_MODE	highlightMode	enum
EZ_RUBBER_BAND	rubberBand	boolean

These two resources are available for fancyListBox, Tree and WorkArea widgets. `EZ_HIGHLIGHT_MODE` selects one of the four modes for highlighting display items. Values are 0,1,2 and 3. The default value is 0. The value is used by *EZwgl* in the following way:

- if bit 1 is on, turn off the marching ants.
- if any other bits are on, highlight pixmap and textual lable individually.

`EZ_RUBBER_BAND` turns rubber band on and off. When turned on, multiple items may be selected.

- `EZ_ROW_BG`

configOption	resourceName	valueType
`EZ_ROW_BG`	`rowBg`	`boolean string string`

This option is available to FancyListBox and Tree widgets. It sets up two colors to be used as alternating row backgrounds. It needs three values, the first one is a boolean value which turns this feature on and off. The last two are colors names, or `nil`, in which case a default color will be selected.

- `EZ_TEXT_WIDGET_EDITABLE, EZ_TEXT_SPACING`

configOption	resourceName	valueType
`EZ_TEXT_SPACING`	`textSpacing`	`int`
`EZ_TEXT_WIDGET_EDITABLE`	`textWidgetEditable`	`boolean`

These two resources are available only for text widget. `EZ_TEXT_SPACING` sets the insertions between text lines. `EZ_TEXT_WIDGET_EDITABLE` turns text editing on/off.

- `EZ_FANCY_LISTBOX_TYPE, EZ_FANCY_LISTBOX_COLUMNS`

configOption	resourceName	valueType
`EZ_FANCY_LISTBOX_TYPE`	`fancyListBoxType`	`enum`
`EZ_FANCY_LISTBOX_COLUMNS`	`fancyListBoxColumns`	`int`

`EZ_FANCY_LISTBOX_TYPE` specifies the selection mode of a fancyListBox. Values are either 0 or 1. When sets to 0, individual items can be selected; when set to 1, only rows can be selected.

`EZ_FANCY_LISTBOX_COLUMNS` specifies the number of columns in a fancyListBox widget.

- `EZ_BACKING_STORE, EZ_TRANSIENT, EZ_CURSOR`

configOption	resourceName	valueType
`EZ_BACKING_STORE`	`backingStore`	`boolean`
`EZ_OVERRIDE_REDIRECT`	`overrideRedirect`	`boolean`
`EZ_TRANSIENT`	`transient`	`boolean`
`EZ_CURSOR`	`cursur`	`???`

`EZ_BACKING_STORE` sets the backing store window attributes on the widget window. `EZ_OVERRIDE_REDIRECT` sets the override redirect window attributes on the widget window. `EZ_TRANSIENT` sets the transientFor window manager hint on the widget window. `EZ_CURSOR` defines the widget window cursor. When set by resource, value is a cursor name. When set by `EZ_CreateWidget`, value is a symbolic constant.

- `EZ_CLASS, EZ_NAME`

configOption	resourceName	valueType
`EZ_CLASS`	`class`	`string`
`EZ_NAME`	`name`	`string`

These two resources set the widget class name and the widget instance name. They override the default widget class and instance names.

- `EZ_SMETER_VALUE, EZ_SMETER_RANGE, EZ_SMETER_VALUE_N`

configOption	resourceName	valueType
`EZ_SMETER_RANGE`	`statusMeterRange`	`float_min float_max`
`EZ_SMETER_RANGE_N`	`statusMeterRange`	`int_n float_min float_max`
`EZ_SMETER_VALUE`	`statusMeterValue`	`float`
`EZ_SMETER_VALUE_N`	`statusMeterValueN`	`int_n float_v`

A status meter can monitor 5 quantities simultaneously. These options set the range and value of these quantities.`EZ_SMETER_R` sets the range of the nth quantity. `EZ_SMETER_RANGE` sets the range of the first quantity. `EZ_SMETER_VALUE_N` sets the values of the nth quantity. `EZ_SMETER_VALUE` sets the value of the first quantity.

- EZ_SMETER_STYLE, EZ_SMETER_ORIENTATION

configOption	resourceName	valueType
EZ_SMETER_STYLE	statusMeterStyle	int int
EZ_SMETER_ORIENTATION	statusMeterOrientation	enum

EZ_SMETER_STYLE defines the display style of a status meter. Value is a pair of integers representing the width of on/off rectangles. EZ_SMETER_ORIENTATION selects the orientation of a status meter, value is EZ_HORIZONTAL or EZ_VERTICAL.

- EZ_SMETER_FOREGROUND, EZ_SMETER_BACKGROUND, EZ_SMETER_FOREGROUND_N

configOption	resourceName	valueType
EZ_SMETER_BACKGROUND	statusMeterBackground	string
EZ_SMETER_FOREGROUND	statusMeterForeground	string
EZ_SMETER_FOREGROUND_N	statusMeterForegroundN	int_n string

EZ_SMETER_BACKGROUND sets the background color of a status meter. EZ_SMETER_FOREGROUND sets the color to display the first quantity a status meter monitors. EZ_SMETER_FOREGROUND_N sets the foreground color to display the nth quantity.

- EZ_SMETER_STOP, EZ_SMETER_SHOW_VALUE

configOption	resourceName	valueType
EZ_SMETER_STOP	statusMeterStop	boolean
EZ_SMETER_SHOW_VALUE	statusMeterShowValue	boolean

EZ_SMETER_STOP activate or deactivate a status meter. EZ_SMETER_SHOW_VALUE toggle numeric display on off.

- EZ_WM_WINDOW_NAME, EZ_WM_ICON_NAME, EZ_WM_INITIAL_STATE

configOption	resourceName	valueType
EZ_WM_WINDOW_NAME	wmWindowName	string
EZ_WM_ICON_NAME	wmIconName	string
EZ_WM_INITIAL_STATE	wmInitialState	enum

These three options set the window manager hints for windowName, iconName and initialState. Value for wmInitialState is normalState or iconState. These options are relevent only for toplevel widgets.

- EZ_WM_ICON_LOCATION, EZ_WM_ICON_XPM_FILE

configOption	resourceName	valueType
EZ_WM_ICON_LOCATION	wmIconLocation	int_x int_y
EZ_WM_ICON_XPM_FILE	wmIconXpmFile	string
EZ_WM_ICON_PIXMAP	N/A	EZ_Bitmap *

These options set the window manager hints for iconPosition and iconPixmap. The last option sets the icon pixmap to be an *EZwgl* internal pixmap, it cannot be set via resources. These options are relevent only for toplevel widgets.

- EZ_WM_LOCATION, EZ_WM_SIZE_HINT, EZ_WM_MAX_SIZE_HINT

configOption	resourceName	valueType
EZ_WM_LOCATION	wmLocation	int_x int_y
EZ_WM_SIZE_HINT	wmSizeHint	int_w int_h
EZ_WM_MIN_SIZE_HINT	wmMinSizeHint	int_w int_h
EZ_WM_MAX_SIZE_HINT	wmMaxSizeHint	int_w int_h

These options set the window manager hints for position, size, minSize and maxSize respectively. These options are relevent only for toplevel widgets.

- EZ_WMHINTS

configOption	resourceName	valueType
EZ_WM_HINTS	wmHints	int_mask

This option sets the window manager hint, value is bitwise or'ed from PMinSize, PMaxSize, PAspect and PWinGravity. These options are relevent only for toplevel widgets.

- EZ_INTERIOR_BORDER

configOption	resourceName	valueType
EZ_INTERIOR_BORDER	interiorBorder	int_width enum_style

This option sets the internal border of a labeled frame or a labeled icon. It takes two values, a border width and a border style. The later must be one of the valid border types defined in *EZwgl*.

- `EZ_ARROW_LABEL`

configOption	resourceName	valueType
EZ_ARROW_LABEL	arrowLabel	enum

This option is available only for Button, Label and NoWindowLabel. It labels the widget with an internally defined symbol (arrow). Value is a symbolic constant listed below. These symbolic constants are not internally defined as resource values. Therefore, when set via resources, the resource value must be an integer between 1 and 36.

```
#define EZ_UP_TRIANGLE          1       #define EZ_DOUBLE_UP_TRIANGLE    19
#define EZ_DOWN_TRIANGLE        2       #define EZ_DOUBLE_DOWN_TRIANGLE  20
#define EZ_LEFT_TRIANGLE        3       #define EZ_LEFT_RIGHT_ARROW      21
#define EZ_RIGHT_TRIANGLE       4       #define EZ_UP_DOWN_ARROW         22
#define EZ_UP_ARROW             5       #define EZ_LEFT_BAR_TRIANGLE     23
#define EZ_DOWN_ARROW           6       #define EZ_RIGHT_BAR_TRIANGLE    24
#define EZ_LEFT_ARROW           7       #define EZ_UP_BAR_TRIANGLE       25
#define EZ_RIGHT_ARROW          8       #define EZ_DOWN_BAR_TRIANGLE     26
#define EZ_UP_TRIANGLE_BAR      9       #define EZ_ELLIPSE               27
#define EZ_DOWN_TRIANGLE_BAR    10      #define EZ_RECTANGLE             28
#define EZ_LEFT_TRIANGLE_BAR    11      #define EZ_UP_TRIANGLE_UP        29
#define EZ_RIGHT_TRIANGLE_BAR   12      #define EZ_UP_TRIANGLE_DOWN      30
#define EZ_UP_ARROW_BAR         13      #define EZ_DOWN_TRIANGLE_UP      31
#define EZ_DOWN_ARROW_BAR       14      #define EZ_DOWN_TRIANGLE_DOWN    32
#define EZ_LEFT_ARROW_BAR       15      #define EZ_LEFT_TRIANGLE_UP      33
#define EZ_RIGHT_ARROW_BAR      16      #define EZ_LEFT_TRIANGLE_DOWN    34
#define EZ_DOUBLE_LEFT_TRIANGLE  17     #define EZ_RIGHT_TRIANGLE_UP     35
#define EZ_DOUBLE_RIGHT_TRIANGLE 18     #define EZ_RIGHT_TRIANGLE_DOWN   36
```

- `EZ_SCROLL_TYPE`

configOption	resourceName	valueType
EZ_SCROLL_TYPE	scrollType	enum

This option is available only for scrollButton widget. It specifies how to render the scroll arrow. Valid values are listed below.

SymbolicConstant	resourceValue
EZ_LEFT	left
EZ_RIGHT	right
EZ_UP	up
EZ_DOWN	down
EZ_LEFT_RIGHT	leftRight
EZ_UP_DOWN	upDown

- `EZ_DOCKABLE, EZ_DRAGHANDLE`

configOption	resourceName	valueType
EZ_DOCKABLE	dockable	boolean
EZ_DRAG_HANDLE	dragHandle	enum

These two resources are available for Frames and MenuBars. `EZ_DOCKABLE` sets or removes the dockable attribute on a frame or menubar. `EZ_DRAG_HANDLE` sets the location to draw the drag handle. Valid values are

SymbolicConstant	resourceValue
EZ_LEFT	left
EZ_RIGHT	right
EZ_TOP	top

- `EZ_LOCATOR_XY`

configOption	resourceName	valueType
EZ_LOCATOR_XY	locatorXY	float,float

This resource is available only for the locator widget. It sets the location of the locator (inside [0,1]x[0,1]).

- `EZ_SPIN_VALUE, EZ_SPIN_FUNCTION`

configOption	resourceName	valueType
EZ_SPIN_VALUE	spinValue	int,string
EZ_SPIN_FUNCTION	spinValue	func,pointer

These resources are available only for the spin button widget. They alter the state of a spin button. `EZ_ENTRY_STRING` sets the current value and its position in the list of possible values of the spin button. `EZ_ENTRY_FUNCTION` register the function to compute the next value for the spin button. It is a function of type:

```
char *(*func)(int current, int next, void *clientData)
```

When invoked, `current/next` are the index of the current and the next spin value, `clientData` is arbitary client data registered to the spin function.

- `EZ_FORGET_LOCATION, EZ_FORGET_SIZE, EZ_FORGET_GEOMETRY`

configOption	resourceName	valueType
EZ_FORGET_X	forgetX	boolean
EZ_FORGET_Y	forgetY	boolean
EZ_FORGET_W	forgetW	boolean
EZ_FORGET_H	forgetH	boolean
EZ_FORGET_LOCATION	forgetLocation	boolean
EZ_FORGET_SIZE	forgetSize	boolean
EZ_FORGET_GEOMETRY	forgetGeometry	boolean

These options informs the geometry manager to forget about the configured geometry after using it. They are meant to be used by applications to set the initial geometry of internal widgets.

- `EZ_ENTRY_STRING`

configOption	resourceName	valueType
EZ_ENTRY_STRING	entryString	string

This resource sets the entry string for Entry and OptionalEntry widgets.

- `EZ_BUBBLE_STRING`

configOption	resourceName	valueType
EZ_BUBBLE_STRING	bubbleString	string

This resource sets the bubble help message for a widget.

- `EZ_DND_BUBBLE_STRING`

configOption	resourceName	valueType
EZ_DND_BUBBLE_STRING	dndBubbleString	string

This resource sets the bubble message for a drop target.

- `EZ_DND_DRAG_CURSOR`

configOption	resourceName	valueType
EZ_DND_DRAG_CURSOR	dndDragCursor	string

This option sets the drag cursor for a drag source. When set by resource, value is a cursor name. When set by `EZ_CreateWidget`, value is a symbolic constant.

- `EZ_RETURN_VALUE`

configOption	resourceName	valueType
EZ_RETURN_VALUE	returnValue	int

This option sets return value for menu-items. The return value field in the widget structure can also be used to hold an extra client data.

- `EZ_GLOB_PATTERN`

configOption	resourceName	valueType
EZ_GLOB_PATTERN	globPattern	string

This resource sets the glob pattern for a `FileSelector` widget.

- `EZ_SHAPED_WINDOW`

configOption	resourceName	valueType
EZ_SHAPED_WINDOW	shapedWindow	boolean

This resource sets shape window attribute on a button or label widget. When set, the widget window will be masked by the shape mask of its pixmap label.

- EZ_SHAPE_FILE

configOption	resourceName	valueType
EZ_SHAPE_FILE	shapedFile	string

This resource sets shape mask resource. Value is a X11 bitmap file. If set succesfully, it sets the EZ_SHAPED_WINDOW resource also.

- EZ_SHAPE_PIXMAP

configOption	resourceName	valueType
EZ_SHAPE_FILE	N/A} & {\verbpointer+	

This resource sets shape mask resource. Value is an internal pixmap allocated by one of the EZ_CreateLabelPixmapFrom....

- EZ_MENU_TEAR_OFF

configOption	resourceName	valueType
EZ_MENU_TEAR_OFF	menuTearOff	boolean

This resource inserts or removes the tear-off bar on a popup menu.

- EZ_POPUP_NAME

configOption	resourceName	valueType
EZ_POPUP_NAME	popupName	string

This resource is available for OptionalEntry and FileSelector widgets. For OptionalEntry, it sets the instance name of the pull down listbox. For FileSelector, it sets the instance names for the two pull down listboxes, to be the specified name with the letter 'A' and 'B' appended.

- EZ_IS_EMBEDER

configOption	resourceName	valueType
EZ_IS_EMBEDER	isEmbeder	boolean

This option sets the embeding attribute on a widget. When set, the widget swallows other *EZwgl* client's window when moved over to the widget window.

- EZ_MOVABLE

configOption	resourceName	valueType
EZ_MOVABLE	movable	boolean

This option is available for WorkArea and FreeLabel. For a WorkArea widget, it enable/disable moving items around by the left mouse button. For FreeLabel, it enable/disable moving the widget window by the left mouse button.

- EZ_LABELED_ICON

configOption	resourceName	valueType
EZ_LABELED_ICON	movable	int

This option is available for toplevel FreeLabels. It selects one of the three display styles (correspond to 0,1,2) for icons. If value is 0, the icon ignores the textual label when it has a pixmap label. If value is 1, it displays both the textual and the pixmap label, and it shape mask the pixmap label unless the EZ_SHAPED_WINDOW resource is set to False. If value is 2, it shape mask both the pixmap label and the textual label, unless the EZ_SHAPED_WINDOW resource is set to False.

- EZ_MARK_CLEAN

configOption	resourceName	valueType
EZ_MARK_CLEAN	N/A	boolean

What is this for?

- EZ_RESERVE_MENU_BUTTON

configOption	resourceName	valueType
EZ_RESERVE_MENU_BUTTON	reserveMenuButton	boolean

This resource is available for 3DCanvas only. It tells *EZwgl* to reserve the right mouse button as the menu button.

- `EZ_GEOMETRY_MANAGER`

configOption	resourceName	valueType
EZ_GEOMETRY_MANAGER	N/A	???

This configuration option is available for container widgets. It sets the geometry manager for laying our children widgets.

- `EZ_DIAL_RANGE, EZ_DIAL_SPAN`

configOption	resourceName	valueType
EZ_DIAL_RANGE	dialRange	float float
EZ_DIAL_RESOLUTION	dialResolution	float
EZ_DIAL_SPAN	dialSpan	float
EZ_DIAL_VALUE	dialValue	float
EZ_DIAL_DISPLAY_VALUE	dialDisplayValue	int
EZ_DIAL_STYLE	dialStyle	enum

These resources modify the specified dial attributes.

- `EZ_BAR_RANGE, EZ_BAR_COLOR_N`

configOption	resourceName	valueType
EZ_BAR_RANGE	barRange	float float
EZ_BAR_WIDTH	barWidth	int
EZ_BAR_EDITABLE	barEditable	bool
EZ_BAR_ORIENTATION	barOrientation	enum
EZ_BAR_COLOR_N	barColorN	int str ...

These resources modify the specified attributes of a gradient bar.

- `EZ_SHEET_HEADER_SIZE, EZ_SHEET_CELL_SIZE`

configOption	resourceName	valueType
EZ_SHEET_HEADER_SIZE	sheetHeaderSize	int int
EZ_SHEET_CELL_SIZE	sheetCellSize	int int

These options modify the specified attributes of a spread sheet.

3.2 Adding Private Widget Resources

You can add private widget resources to one widget or to all widgets of a given type, e.g frames. For this you need

- a list of resource names and their value types, and
- a configuration procedure specifies how the resources will be used.

The resource specification is of the following type.

```
typedef struct {
  int     option;     /* referenced in configure, must >1024 */
  char    *iname;     /* instance name */
  char    *cname;     /* class name, may be NULL */
  char    *vtypes;    /* value type, e.g  "float, float" */
} EZ_ResourceSpec;
```

A list of resources looks like the next example.

```
#define COUNTER_INIT_VALUE   1025
#define COUNTER_INCREMENT    1026
#define COUNTER_TEST         1027
EZ_ResourceSpec counterResources[] =
{
  {COUNTER_INIT_VALUE,  "initValue", "InitValue", "int"},
  {COUNTER_INCREMENT,   "increment", "Increment", "int"},
  {COUNTER_TEST,        "test",      "Test",      "int,float,string,pointer,int"},
};
```

The configuration procedure has the prototype

```
   int (*configure)(EZ_Widget *widget,int opt, EZ_Value *values);
```

and here is a sample procedure.

```
int counterConfigure(EZ_Widget *widget, int opt, EZ_Value *values)
{
  switch(opt)
   {
    case COUNTER_INIT_VALUE:
      {
        int i = values[0].value.i;
        ez_SetCounter(widget, i);
      }
     break;
    case COUNTER_INCREMENT:
      {
        int i = values[0].value.i;
        ez_counter *ptr = (ez_counter *)EZ_GetWidgetPtrData(widget);
        ptr->increment1 = i;
      }
     break;
    case COUNTER_TEST:
      {
        int   i = values[0].value.i;
        char  c = values[1].value.c;
        float f = values[2].value.f;
        char *str = value[3].value.p;  /* have to copy in real applications */
        fprintf(stderr, "%d, %c,%f, %s\n", i, c, f, str);
      }
     break;
    default:
     break;
 }
}
```

The command responsible for registering resources to all widgets of a given type is

```
void EZ_RegisterResourcesAll(int widget_type, int nresrc, EZ_ResourceSpec *resources,
         int (*configure)(EZ_Widget *widget, int opt, EZ_Value *values) );
```

To register resources for a single widget, use the configuration option `EZ_RESOURCES_HANDLE` in `EZ_CreateWidget`. For example,

```
 counter = EZ_CreateWidget(Counter_TYPE,
           EZ_RESOURCES_HANDLE, 3, counterResources, counterConfigure,
           COUNTER_INIT_VALUE, 10,
           0);
```

3.3 Convert Private Resource Values

Resource values are strings. *EZwgl* can only convert resource values to the four basic data types: `int, float, ulong` and `string`. Although pointers can be used as configuration values, they cannot be specified by resources directly. If you need to convert resource values to pointers, you have to provide a convertion mechanism to translate strings to pointers. In the next example, we use resource to select a callback procedure for a button. Since there is only one private resource in this example, we'll extract it directly instead of registering it to the button widget.

```
#include "EZ.h"

static void sayHi(EZ_Widget *widget, void *data)
{  printf("Hello, there\n"); }

static void sayBye(EZ_Widget *widget, void *data)
{  printf("Goodbye!\n"); }


static  EZ_ResourceSpec exampleResources[] =
{
   { 1234, "callback",  "Callback", "string"   },
};


int configure(void *widget, int option, EZ_Value *values)
{
  if(option == 1234)
   {
     if(!strncmp(values[0].value.str, "hi",2))
       EZ_AddWidgetCallBack(widget, EZ_CALLBACK, sayHi, NULL, 0);
     else if(!strncmp(values[0].value.str, "bye",3))
       EZ_AddWidgetCallBack(widget, EZ_CALLBACK, sayBye, NULL, 0);
   }
}

main(int ac, char **av)
```

```
{
  EZ_Widget *btn;

  EZ_Initialize(ac, av, 0);
  fprintf(stderr, "\nRun with: %s -xrm \"*btn.callback: hi\" \nor\n         %s -xrm \"*btn.callback: bye\"\n", av[0],av[0]);

  btn  =   EZ_CreateWidgetXrm(EZ_WIDGET_NORMAL_BUTTON,    NULL,
                              "Btn",               "btn",
                              EZ_LABEL_STRING,     "The callbacks of this button\nis set by Resource",
                              EZ_RESOURCES_HANDLE,  1, exampleResources, configure,
                              0);
  EZ_DisplayWidget(btn);
  EZ_EventMainLoop();
}
```

Chapter 4

The List of *EZ* Widgets

In this chapter, we list and describe all widgets currently available in *EZwgl*.

4.1 Frame

Frames are the most commonly used container widgets. The primiary purpose of frames is to serve as containers for grouping other widgets together.

A frame may have an optional textual/image lable or both, in which case it appears as a labeled frame with the label displayed at the top (left, center or right) of the frame window.

A frame may have an optional docking handle. The docking handle may be display at either the left or right or top of the frame window.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_FRAME`
Default instance name:	`"frame"`
Default Class name:	`"Frame"`

Widget Interface Routines

`int EZ_OnFrameDragHandle(EZ_Widget *widget, int x, int y);`

return 1 if point (x,y) is on the drag handle.

`void EZ_HighlightFrameDragHandle(EZ_Widget *widget, int onoff);`

turn the highlight on the drag handle on or off.

`int EZ_DetachFrame(EZ_Widget *widget, int x, int y);`

detach an internal frame or menubar from its parent.

`int EZ_DockFrame(EZ_Widget *widget, int force);`

try dock a detached frame.

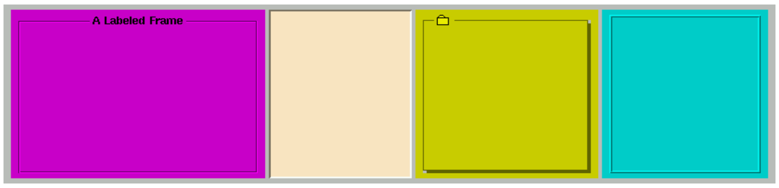
`int EZ_XYIsAtCorner(int x,int y,int w,int h);`

returns 1 if point (x,y) is on the corner of a detached frame.

`int EZ_FrameIsDetached(EZ_Widget *widget);`

returns 1 if frame is detached.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"Frame"`
EZ_NAME	name	`"frame"`

EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	

EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	

EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`0`

EZ_PADX	padx	`8`
EZ_PADY	pady	`6`
EZ_IPADX	iPadx	`0`
EZ_IPADY	iPady	`0`

EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`
EZ_TRANSIENT	transient	`false`

EZ_ORIENTATION	orientation	`horizontal`
EZ_FILL_MODE	fillMode	`fillNone`
EZ_SIDE	side	`center`

EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*Bold*"`
EZ_LABEL_STRING	labelString	`null`
EZ_LABEL_POSITION	labelPosition	`center`
EZ_JUSTIFICATION	justification	`left`
EZ_TEXT_LINE_LENGTH	textLineLength	`30`

EZ_INTERIOR_BORDER	interiorBorder	`"1 groove"`
EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`

EZ_PIXMAP_FILE	pixmapFile	`null`
EZ_IMAGE_FILE	imageFile	`null`
EZ_BITMAP_DATA	N/A	`null`
EZ_PIXMAP_DATA	N/A	`null`
EZ_X_PIXMAP	N/A	`none`
EZ_LABELPIXMAP	N/A	`null`
EZ_RAW_RGB_DATA	N/A	`null`
EZ_SHAPED_WINDOW	shapedWindow	`false`
EZ_SHAPE_FILE	shapeFile	`NULL`
EZ_SHAPE_PIXMAP	N/A	`NULL`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`

EZ_DRAG_HANDLE	dragHandle	`none`
EZ_DOCKABLE	dockable	`false`
EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_BACKING_STORE	backingStore	`false`

EZ_GRID_CELL_GEOMETRY	gridCellGeometry	
EZ_GRID_CELL_PLACEMENT	gridCellPlacement	
EZ_GRID_CONSTRAINS	gridConstrains	
EZ_GRID_ROW_CONSTRAINS	gridRowConstrains	
EZ_GRID_COLUMN_CONSTRAINS	gridColumnConstrains	

EZ_WMHINTS	wmHints	`0`
EZ_WM_WINDOW_NAME	wmWindowName	
EZ_WM_INITIAL_STATE	wmInitialState	`normalState`
EZ_WM_ICON_NAME	wmIconName	
EZ_WM_ICON_XPM_FILE	wmIconXpmFile	
EZ_WM_ICON_PIXMAP	wmIconPixmap	
EZ_WM_ICON_LOCATION	wmIconLocation	
EZ_WM_LOCATION_HINT	wmLocation	
EZ_WM_SIZE_HINT	wmSizeHint	
EZ_WM_MIN_SIZE_HINT	wmMinSizeHint	
EZ_WM_MAX_SIZE_HINT	wmMaxSizeHint	

4.2 Label

A label widget displays either a textual and/or a bitmap/pixmap label. A textual label may contain multiple lines of text and can be left, center or right justified. If the displayed dimension of a label is larger that the minimal size needed to hold its label, the textual or bitmap label may be anchored at different locations inside its window.

Labels do not handle useful events.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_LABEL`
Default instance name:	`"label"`
Default Class name:	`"Label"`

Widget Interface Routines

`int EZ_SetWidgetLabelString(EZ_Widget *widget, char *str);`

set the textual label.

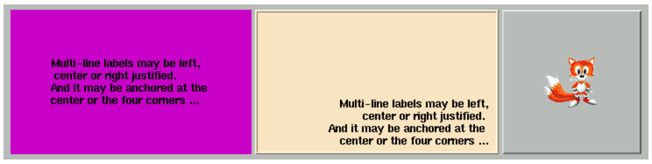
`void EZ_SetWidgetLabelPixmap(EZ_Widget *widget, EZ_Bitmap *pixmap);`

set the pixmap label.

`char *EZ_GetWidgetLabelString(EZ_Widget *widget);`

return the textual label.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"Label"`
EZ_NAME	name	`"label"`
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	
EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	
EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`0`
EZ_PADX	padx	`8`
EZ_PADY	pady	`6`
EZ_IPADX	iPadx	`0`
EZ_IPADY	iPady	`0`

EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`
EZ_TRANSIENT	transient	`false`

EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*Bold*"`
EZ_LABEL_STRING	labelString	`null`
EZ_LABEL_POSITION	labelPosition	`center`
EZ_JUSTIFICATION	justification	`left`
EZ_TEXT_LINE_LENGTH	textLineLength	`30`
EZ_ARROW_LABEL	arrowLabel	`0`

EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`

EZ_PIXMAP_FILE	pixmapFile	`null`
EZ_IMAGE_FILE	imageFile	`null`
EZ_BITMAP_DATA	N/A	`null`
EZ_PIXMAP_DATA	N/A	`null`
EZ_X_PIXMAP	N/A	`none`
EZ_LABELPIXMAP	N/A	`null`
EZ_RAW_RGB_DATA	N/A	`null`
EZ_SHAPED_WINDOW	shapedWindow	`false`
EZ_SHAPE_FILE	shapeFile	`NULL`
EZ_SHAPE_PIXMAP	N/A	`NULL`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`

EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_BACKING_STORE	backingStore	`false`

EZ_GRID_CELL_GEOMETRY	gridCellGeometry	
EZ_GRID_CELL_PLACEMENT	gridCellPlacement	

4.3 Normal Button

Normal buttons are also refered to as push buttons. They are similar to labels in appearance except they respond to the mouse or keyboard.

When the pointer is over a push button, the button becomes highlighted. This highlighting indicates that the button is ready for selection. When mouse button 1 (left button) is pressed, the push button indicates that it has been selected by changing its border to make it look pressed down. When the mouse button is released over the button, the buttons callbacks are invoked. If the pointer is moved off of the button before the pointer button is released, the button reverts to its normal appearance, and releasing the mouse button has no effect. This behavior allows the user to cancel an action.

If a push button is configured with a keyboard accelerator, it can also be invoked by typing the corresponding key strokes.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_NORMAL_BUTTON`
Default instance name:	`"button"`
Default Class name:	`"Button"`

Widget Interface Routines

`int EZ_SetWidgetLabelString(EZ_Widget *widget, char *str);`

set the textual label.

`void EZ_SetWidgetLabelPixmap(EZ_Widget *widget, EZ_Bitmap *pixmap);`

set the pixmap label.

`char *EZ_GetWidgetLabelString(EZ_Widget *widget);`

return the textual label.

Screenshot

Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"Button"`
EZ_NAME	name	`"button"`
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	
EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	2
EZ_BORDER_TYPE	borderType	raised
EZ_FOCUS_PAD	focusPad	2

EZ_PADX	padx	8
EZ_PADY	pady	6
EZ_IPADX	iPadx	8
EZ_IPADY	iPady	6

EZ_EXPAND	expand	false
EZ_PROPAGATE	propagate	true
EZ_TRANSIENT	transient	false

EZ_FONT_ID	fontID	3
EZ_FONT_NAME	fontName	"*Bold*"
EZ_LABEL_STRING	labelString	null
EZ_LABEL_POSITION	labelPosition	center
EZ_JUSTIFICATION	justification	left
EZ_TEXT_LINE_LENGTH	textLineLength	30
EZ_ARROW_LABEL	arrowLabel	0

EZ_UNDERLINE	underline	-1
EZ_CURSOR	cursor	"XC_left_ptr"
EZ_BUBBLE_STRING	bubbleString	null

EZ_CALLBACK	N/A	null null
EZ_DESTROY_CALLBACK	N/A	null null
EZ_EVENT_HANDLER	N/A	null null

EZ_PIXMAP_FILE	pixmapFile	null
EZ_IMAGE_FILE	imageFile	null
EZ_BITMAP_DATA	N/A	null
EZ_PIXMAP_DATA	N/A	null
EZ_X_PIXMAP	N/A	none
EZ_LABELPIXMAP	N/A	null
EZ_RAW_RGB_DATA	N/A	null
EZ_SHAPED_WINDOW	shapedWindow	false
EZ_SHAPE_FILE	shapeFile	NULL
EZ_SHAPE_PIXMAP	N/A	NULL

EZ_FOREGROUND	foreground	"black"
EZ_BACKGROUND	background	"gray74
EZ_BG_IMAGE_FILE	bgImageFile	null
EZ_BG_PIXMAP	N/A	none
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_CLIENT_PTR_DATA	N/A	null
EZ_CLIENT_INT_DATA	clientIntData	0

EZ_DND_DRAG_CURSOR	dndDragCursor	none
EZ_DND_BUBBLE_STRING	dndBubbleString	null

EZ_BACKING_STORE	backingStore	false
EZ_SHORTCUT_KEY	shortcutKey	null

EZ_GRID_CELL_GEOMETRY	gridCellGeometry	
EZ_GRID_CELL_PLACEMENT	gridCellPlacement	

4.4 Check Button

A check button is typically used to toggle between two choices, represented by the "on" and "off" states of the check button. When mouse button 1 is pressed over a check button, it switches to a state different from its current state, and its callbacks are called.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_CHECK_BUTTON`
Default instance name:	`"checkButton"`
Default Class name:	`"CheckButton"`

Widget Interface Routines

```
int     EZ_SetWidgetLabelString(EZ_Widget *widget, char *str);
```
set the textual label.

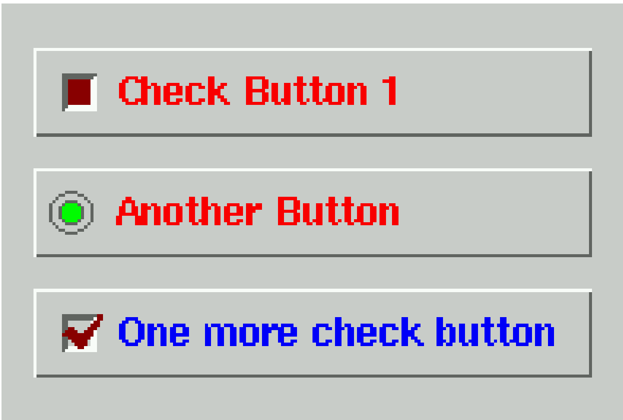
```
void    EZ_SetWidgetLabelPixmap(EZ_Widget *widget, EZ_Bitmap *pixmap);
```
set the pixmap label.

```
char    *EZ_GetWidgetLabelString(EZ_Widget *widget);
```
return the textual label.

```
void    EZ_SetCheckButtonState(EZ_Widget *widget, int onoff);
```
set the state of a check button.

```
int     EZ_GetCheckButtonState(EZ_Widget *widget, int *value);
```
retrieve the state and state value of a check button.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"CheckButton"`
EZ_NAME	name	`"checkButton"`

EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	

EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	

EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`2`

EZ_PADX	padx	`8`
EZ_PADY	pady	`6`
EZ_IPADX	iPadx	`8`
EZ_IPADY	iPady	`6`

EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`
EZ_TRANSIENT	transient	`false`

EZ_ORIENTATION	orientation	`horizontal`
EZ_FILL_MODE	fillMode	`fillNone`
EZ_SIDE	side	`center`

EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*Bold*"`
EZ_LABEL_STRING	labelString	`null`
EZ_LABEL_POSITION	labelPosition	`center`
EZ_JUSTIFICATION	justification	`left`
EZ_TEXT_LINE_LENGTH	textLineLength	`30`

EZ_UNDERLINE	underline	`-1`
EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`

EZ_PIXMAP_FILE	pixmapFile	`null`
EZ_IMAGE_FILE	imageFile	`null`
EZ_BITMAP_DATA	N/A	`null`
EZ_PIXMAP_DATA	N/A	`null`
EZ_X_PIXMAP	N/A	`none`
EZ_LABELPIXMAP	N/A	`null`
EZ_RAW_RGB_DATA	N/A	`null`
EZ_SHAPED_WINDOW	shapedWindow	`false`
EZ_SHAPE_FILE	shapeFile	`NULL`
EZ_SHAPE_PIXMAP	N/A	`NULL`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_BACKING_STORE	backingStore	`false`
EZ_SHORTCUT_KEY	shortcutKey	`null`

EZ_INDICATOR_TYPE	indicatorType	`EZ_SQUARE_INDICATOR`
EZ_INDICATOR_COLOR	indicatorColor	`darkRed`
EZ_INDICATOR_SIZE_ADJUST	indicatorSizeAdjust	`0`

EZ_CHECK_BUTTON_ON_VALUE	checkButtonOnValue	`1`
EZ_CHECK_BUTTON_OFF_VALUE	checkButtonOffValue	`0`
EZ_CHECK_BUTTON_ON_OFF	checkButtonOnOff	`off`

EZ_GRID_CELL_GEOMETRY	gridCellGeometry	
EZ_GRID_CELL_PLACEMENT	gridCellPlacement	

4.5 Radio Button

Radio buttons are usually arranged in groups and are used to select one from a group of mutualy exclusive choices, such as one of the several fonts or one of several colors. Associate to each group of radio buttons there is one internal group variable. When button 1 is pressed over a radio button, the group variable is set to the value associated with the radio button and the appearance of the button changes (border changes or indicator lights up). And its callbacks are then invoked.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_RADIO_BUTTON`
Default instance name:	`"radioButton"`
Default Class name:	`"RadioButton"`

Widget Interface Routines

`int EZ_SetWidgetLabelString(EZ_Widget *widget, char *str);`

set the textual label.

`void EZ_SetWidgetLabelPixmap(EZ_Widget *widget, EZ_Bitmap *pixmap);`

set the pixmap label.

`char *EZ_GetWidgetLabelString(EZ_Widget *widget);`

return the textual label.

`void EZ_SetRadioButtonGroupVariableValue(EZ_Widget *widget, int value);`

set the value of the group variable corresponding to a radio button.

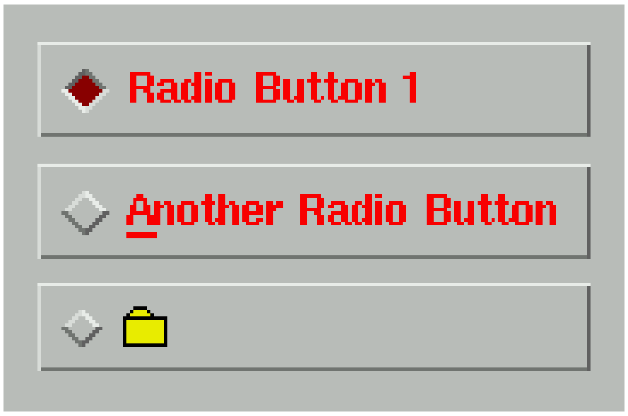
`int EZ_GetRadioButtonGroupVariableValue(EZ_Widget *widget);`

retrieve the value of the group variable corresponding to a radio button.

`int EZ_GetRadioButtonGroupID(EZ_Widget *widget);`

retrieve the group identifier of a radio button.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"RadioButton"`
EZ_NAME	name	`''radioButton"`

EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	

EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	

EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`2`

EZ_PADX	padx	`8`
EZ_PADY	pady	`6`
EZ_IPADX	iPadx	`8`
EZ_IPADY	iPady	`6`

EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`
EZ_TRANSIENT	transient	`false`

EZ_ORIENTATION	orientation	`horizontal`
EZ_FILL_MODE	fillMode	`fillNone`
EZ_SIDE	side	`center`

EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*Bold*"`
EZ_LABEL_STRING	labelString	`null`
EZ_LABEL_POSITION	labelPosition	`center`
EZ_JUSTIFICATION	justification	`left`
EZ_TEXT_LINE_LENGTH	textLineLength	`30`

EZ_UNDERLINE	underline	`-1`
EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`

EZ_PIXMAP_FILE	pixmapFile	`null`
EZ_IMAGE_FILE	imageFile	`null`
EZ_BITMAP_DATA	N/A	`null`
EZ_PIXMAP_DATA	N/A	`null`
EZ_X_PIXMAP	N/A	`none`
EZ_LABELPIXMAP	N/A	`null`
EZ_RAW_RGB_DATA	N/A	`null`
EZ_SHAPED_WINDOW	shapedWindow	`false`
EZ_SHAPE_FILE	shapeFile	`NULL`
EZ_SHAPE_PIXMAP	N/A	`NULL`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_BACKING_STORE	backingStore	`false`
EZ_SHORTCUT_KEY	shortcutKey	`null`

EZ_INDICATOR_TYPE	indicatorType	`EZ_DIAMOND_INDICATIOR`
EZ_INDICATOR_COLOR	indicatorColor	`darkRed`
EZ_INDICATOR_SIZE_ADJUST	indicatorSizeAdjust	`0`

EZ_RADIO_BUTTON_GROUP	radioButtonGroup	
EZ_RADIO_BUTTON_VALUE	radioButtonValue	

EZ_GRID_CELL_GEOMETRY	gridCellGeometry	
EZ_GRID_CELL_PLACEMENT	gridCellPlacement	
EZ_GRID_CONSTRAINS	gridConstrains	

4.6 Menu Buttons

Menu buttons are very similar to normal buttons except that they serve as hooks to pull-down menus. When user press mouse button 1 over a menu button it posts its associated menu under the menu button. Then the user can slide the mouse down over the menu with button still pressed and release over a desired item. When the button is released over a menu item, the menu is unposted and if the menu item has any callbacks, they are invoked first. In the mean time, the selected menu entry and its return identifier are recorded in both the most recently posted menu (parent of the selected item) and the responsible menu button. If the menu has callbacks, they are invoked next. If the menu button has callbacks, they are invoked the last. This feature makes it convenient to manage menu callbacks. For the complex cases, one can associate a callback to each menu item. For simple cases, one may just register one callback to a menu button.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_MENU_BUTTON`
Default instance name:	`"menuButton"`
Default Class name:	`"MenuButton"`

Widget Interface Routines

`int    EZ_SetWidgetLabelString(EZ_Widget *widget, char *str);`

set the textual label.

`void   EZ_SetWidgetLabelPixmap(EZ_Widget *widget, EZ_Bitmap *pixmap);`

set the pixmap label.

`char   *EZ_GetWidgetLabelString(EZ_Widget *widget);`

return the textual label.

`void  EZ_SetMenuButtonMenu(EZ_Widget *widget, EZ_Widget *menu);`

associate a menu to a menu button.

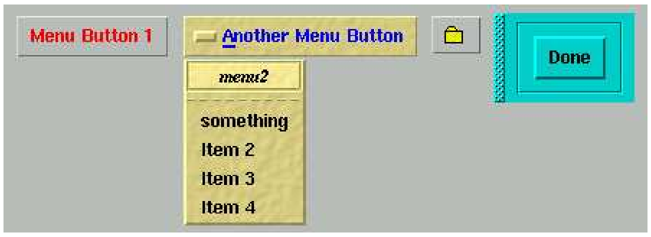
`EZ_Widget *EZ_GetMenuButtonMenu(EZ_Widget *widget);`

retrieve the menu associated to a menu button.

`int EZ_GetWidgetReturnedData(EZ_Widget *widget);`

retrieve the returned data from the last selected menu item or -1.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"MenuButton"`
EZ_NAME	name	`"menuButton"`

EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	

EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	

EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	`2`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`2`

EZ_PADX	padx	`8`
EZ_PADY	pady	`6`
EZ_IPADX	iPadx	`8`
EZ_IPADY	iPady	`6`

EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`
EZ_TRANSIENT	transient	`false`

EZ_ORIENTATION	orientation	`horizontal`
EZ_FILL_MODE	fillMode	`fillNone`
EZ_SIDE	side	`center`

EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*Bold*"`
EZ_LABEL_STRING	labelString	`null`
EZ_LABEL_POSITION	labelPosition	`center`
EZ_JUSTIFICATION	justification	`left`
EZ_TEXT_LINE_LENGTH	textLineLength	`30`

EZ_UNDERLINE	underline	`-1`
EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`

EZ_PIXMAP_FILE	pixmapFile	`null`
EZ_IMAGE_FILE	imageFile	`null`
EZ_BITMAP_DATA	N/A	`null`
EZ_PIXMAP_DATA	N/A	`null`
EZ_X_PIXMAP	N/A	`none`
EZ_LABELPIXMAP	N/A	`null`
EZ_RAW_RGB_DATA	N/A	`null`
EZ_SHAPED_WINDOW	shapedWindow	`false`
EZ_SHAPE_FILE	shapeFile	`NULL`
EZ_SHAPE_PIXMAP	N/A	`NULL`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_BACKING_STORE	backingStore	`false`
EZ_SHORTCUT_KEY	shortcutKey	`null`

EZ_GRID_CELL_GEOMETRY	gridCellGeometry	
EZ_GRID_CELL_PLACEMENT	gridCellPlacement	

4.7 Menus

A menu is a widget that contains a collection of items arranged in a column. If a menu is tied to a menuButton, it behaves a a pull-down menu; i.e, they are activated by pressing the mouse button 1 over a menu button. If a menu is tied to a mouse button, it will behave like a popup menu.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_MENU`
Default instance name:	`"menu"`
Default Class name:	`"Menu"`

Widget Interface Routines

`int EZ_DoPopup(EZ_Widget *widget, int where);`

post a menu.

`int EZ_DoPopupAt(EZ_Widget *widget, int x, int y);`

post a menu at the specified location.

`void EZ_SetMenuSetupProcedure(EZ_Widget *menu, EZ_CallBack p, void *data);`

register a procedure to be invoked before posting the menu.

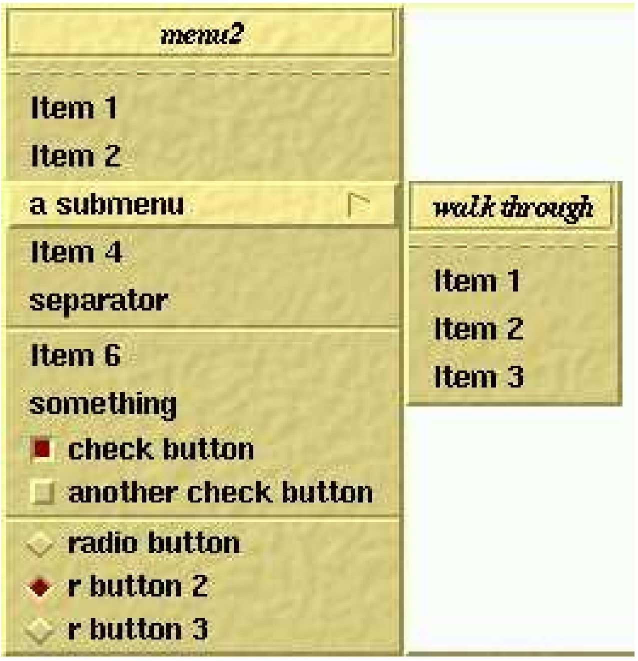
`EZ_Widget *EZ_GetPopupMenuTitleWidget(EZ_Widget *menu);`

return the title widget (a lanel) on a menu.

`EZ_Widget *EZ_GetSelectedMenuItem(EZ_Widget *menu);`

return the last selected item.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"Menu"`
EZ_NAME	name	`"menu"`

EZ_BORDER_WIDTH	borderWidth	`2`
EZ_BORDER_TYPE	borderType	`raised`
EZ_FOCUS_PAD	focusPad	`0`

EZ_PADX	padx	`0`
EZ_PADY	pady	`0`
EZ_IPADX	iPadx	`0`
EZ_IPADY	iPady	`0`

EZ_ORIENTATION	orientation	`vertical`
EZ_FILL_MODE	fillMode	`fillBoth`
EZ_SIDE	side	`left`

EZ_CURSOR	cursor	`"XC_left_ptr"`

EZ_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`

EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_MENU_TEAR_OFF	menuTearOff	`true`

4.7.1 Creating Simple Menus

EZwgl provides a convenient way to create simple menus, menus that uses only menu-normal-buttons, menu-submenus and menu-separators.

`int EZ_CreateSimpleMenu(char *str, ...)`

Specifing a Simple Menu

A simple menu is specified by a string and an optional set of arguments that provide callback procedures and/or submenus for the menu and menu items.

The string is a number of fields separated by `|`'s. Each field may contain any number of the following specifications:

- `%t` Makes non-formatting text in this item as the menu title string.
- `%F` Gets the callback-procedure, client-data for the callback of the menu from the optional arguments.
- `%f` Gets the callback-procedure, client-data for the callback of this menu-item from the optional arguments.
- `%l` Inserts a menu-separator.
- `%m` Makes this item as a menu-submenu item and gets the menu from the optional arguments.
- `%n` Sets the instance name of the this menu item to be the substring after the current formating characters and before the next formating character or `|`.
- `%c` Sets the class name of the this menu item to be the substring after the current formating characters and before the next formating character or `|`.
- `%u` Set the index of the underline character for the current item.
- `%s` Set the keyboard shortcut for the current item.
- `%x`[0-9]+ Sets the return value for this item. This value overrides the default position-based value assigned to this menu item. You must enter the numeric value as the part of the text
- `%k[int,int,int]` Make this item a menu check button. The three integers are the on value, off value and initial state (on/off).

- `%r[int,int,int]` Make this item a menu radio button. The three integers are the radio button group id, the values this item represents and the initial value of the group variable.

Here are some examples.

```
void       submenu1Callback(EZ_Widget *, void *);
void       Item2Callback(EZ_Widget *, void *);

EZ_Widget *submenu1 = EZ_CreateSimpleMenu(‘‘%F|red|green%x123 %s C-g|blue %u 0|%l|||’’,
                                                submenu1Callback, NULL);

EZ_Widget *menu = EZ_CreateSimpleMenu(‘‘%T Test Menu|Item 1|Item 2%f|submenu%m|last item’’
                                                Item2Callback, NULL, submenu1);
```

The first two lines declare two callbacks, one for a submenu and one for a menu item.

The third line creates a simple menu. This submenu has callback `submenu1Callback` with client data `NULL`. It contains 5 menu items and a menu-separator. The first three items are labeled by “red”, “green” and “blue” respectively. The second item has a numeric return value of `123` and a keyboard shortcut “C-g”. The letter ’b’ in the third item will be underlined and will be used as a keyboard accelerator. The other items have the default return values (their position in the menu). The last two items have no lables specified, so the default lable `item #` will be used for them.

The forth line create a menu with title “Test Menu” and no callbacks. It contains four items. The second item has a callback procedure `Item2Callback` with call-data `NULL`. The third menu is a submenu-item with menu `submenu1`.

Specifing a Menu Tree

One can create a menu hierarchy with a single call to `EZ_CreateMenuFromList`. This function requires an array of menu descriptions of the followingn type.

```
typedef struct menuEntry_ {
  int           type;           /* entry type, e.g. EZ_WIDGET_MENU_NORMAL_BUTTON */
  char          *name;          /* instance name, may be NULL */
  char          *label;         /* label string */
  char          *accelerator;   /* keyboard accelerator, e.g. C-a,  may be NULL */
  int           underline;      /* index of label to underline */
  int           returnValue;    /* return value */
  EZ_CallBack   callback;       /* callback  */
  void          *clientData;    /* client data to be passed to callback */
  int           v1,v2,v3;       /* for radioBtn, groupId, buttonValue, groupVarValue */
                                /* for checkBtn, onValue, offValue, initialOnOffState */
} EZ_MenuEntry;
```

For example, the next code segments defines a menu with a submenu.

```
static  void btnCb(EZ_Widget *, void *);
static  void rbtnCb(EZ_Widget *, void *);
static  void cbtnCb(EZ_Widget *, void *);

static EZ_MenuEntry menuEntries[] = {
  {EZ_WIDGET_MENU_NORMAL_BUTTON,  NULL,  "normal Button",  NULL,  0,     100,   btnCb,    NULL,  0, 0, 0 },
  {EZ_WIDGET_MENU_NORMAL_BUTTON,  NULL,  "Another Button", "M-b", 0,     200,   btnCb,    NULL,  0, 0, 0 },
  {EZ_WIDGET_MENU_SEPARATOR,      NULL,  NULL,             NULL,  0,     0,     NULL,     NULL,  0, 0, 0 },
  {EZ_WIDGET_MENU_SUBMENU,        NULL,  "Walk Through",   "C-s", 0,     300,   NULL,     NULL,  0, 0, 0 },
     {EZ_WIDGET_MENU_NORMAL_BUTTON,  NULL, "S 1",           NULL,  0,     400,   btnCb,    NULL,  0, 0, 0 },
     {EZ_WIDGET_MENU_NORMAL_BUTTON,  NULL, "S 2",           NULL,  0,     500,   btnCb,    NULL,  0, 0, 0 },
     {EZ_WIDGET_MENU_SEPARATOR,      NULL,  NULL,           NULL,  0,     0,     NULL,     NULL,  0, 0, 0 },
     {EZ_WIDGET_MENU_CHECK_BUTTON,   NULL, "C 1",           NULL,  0,     600,   cbtnCb,   NULL,  0, 1, 1 },
     {EZ_WIDGET_MENU_CHECK_BUTTON,   NULL, "C 2",           NULL,  0,     700,   cbtnCb,   NULL,  0, 1, 0 },
     {EZ_WIDGET_MENU_SEPARATOR,      NULL,  NULL,           NULL,  0,     0,     NULL,     NULL,  0, 0, 0 },
     {EZ_WIDGET_MENU_RADIO_BUTTON,   NULL, "R 1",           "C-r"  0,     800,   rbtnCb,   NULL,  5, 0, 0 },
     {EZ_WIDGET_MENU_RADIO_BUTTON,   NULL, "R 2",           NULL,  0,     900,   rbtnCb,   NULL,  5, 1, 0 },
     {EZ_WIDGET_MENU_RADIO_BUTTON,   NULL, "R 3",           NULL,  0,     1000,  rbtnCb,   NULL,  5, 2, 2 },
     {0,                             NULL, NULL,            NULL,  0,     0,     NULL,     NULL,  0, 0, 0 },
  {EZ_WIDGET_MENU_NORMAL_BUTTON,  NULL,  "Last Button", NULL,     0,     2000,  btnCb,    NULL,  0, 0, 0 },
  {0,                      NULL, NULL,            NULL,       0, 0,     NULL,     NULL,  0, 0, 0 },
  NULL,
};

EZ_Widget *menu = EZ_CreateMenuFromList(menuEntries);
```

4.7.2 Menu Separator Item

A menu separator is a decorating item used on menus. When displayed, it shows a sunken horizontal line across the menu.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_MENU_SEPARATOR`
Default instance name:	`"menuSeparatorItem"`
Default Class name:	`"MenuSeparatorItem"`

Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"MenuSeparator"`
EZ_NAME	name	`"menuSeparator"`
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	`10`
EZ_HEIGHT	height	`3`
EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`

4.7.3 Menu Submenu Item

A submenu is an item used on menus. When selected, it posts a popup menu registered to the submenu item.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_MENU_SUBMENU`
Default instance name:	`"menuSubmenuItem"`
Default Class name:	`"MenuSubmenuItem"`

Widget Interface Routines

```
void  EZ_SetSubMenuMenu(EZ_Widget *widget, EZ_Widget *menu);
```

associate a menu to a submenu item.

Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"MenuSubmenuItem"`
EZ_NAME	name	`"menuSubmenuItem"`
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_BORDER_WIDTH	borderWidth	`2`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`0`
EZ_PADX	padx	`8`
EZ_PADY	pady	`6`
EZ_IPADX	iPadx	`8`
EZ_IPADY	iPady	`6`
EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*Bold*"`
EZ_LABEL_STRING	labelString	`null`
EZ_LABEL_POSITION	labelPosition	`left`
EZ_JUSTIFICATION	justification	`left`
EZ_TEXT_LINE_LENGTH	textLineLength	`30`
EZ_UNDERLINE	underline	`-1`
EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_PIXMAP_FILE	pixmapFile	`null`
EZ_IMAGE_FILE	imageFile	`null`
EZ_BITMAP_DATA	N/A	`null`
EZ_PIXMAP_DATA	N/A	`null`
EZ_X_PIXMAP	N/A	`none`
EZ_LABELPIXMAP	N/A	`null`
EZ_RAW_RGB_DATA	N/A	`null`
EZ_SHAPED_WINDOW	shapedWindow	`false`
EZ_SHAPE_FILE	shapeFile	`NULL`
EZ_SHAPE_PIXMAP	N/A	`NULL`
EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_SHORTCUT_KEY	shortcutKey	`null`

4.7.4 Menu Normal Button Item

Menu normal buttons are like normal buttons, except they are used only on popup menus. When mouse button 1 is pressed on a menu normal button, its callbacks are invoked.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_MENU_NORMAL_BUTTON`
Default instance name:	`"menuButtonItem"`
Default Class name:	`"MenuButtonItem"`

Widget Interface Routines

`int EZ_SetWidgetLabelString(EZ_Widget *widget, char *str);`
set the textual label.

`void EZ_SetWidgetLabelPixmap(EZ_Widget *widget, EZ_Bitmap *pixmap);`
set the pixmap label.

`char *EZ_GetWidgetLabelString(EZ_Widget *widget);`
return the textual label.

Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"MenuButtonItem"`
EZ_NAME	name	`"menuButtonItem"`
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_BORDER_WIDTH	borderWidth	`2`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`0`
EZ_PADX	padx	`8`
EZ_PADY	pady	`1`
EZ_IPADX	iPadx	`8`
EZ_IPADY	iPady	`6`
EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`
EZ_TRANSIENT	transient	`false`
EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*Bold*"`
EZ_LABEL_STRING	labelString	`null`
EZ_LABEL_POSITION	labelPosition	`left`
EZ_JUSTIFICATION	justification	`left`
EZ_TEXT_LINE_LENGTH	textLineLength	`30`
EZ_UNDERLINE	underline	`-1`
EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`
EZ_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`

EZ_PIXMAP_FILE	pixmapFile	`null`
EZ_IMAGE_FILE	imageFile	`null`
EZ_BITMAP_DATA	N/A	`null`
EZ_PIXMAP_DATA	N/A	`null`
EZ_X_PIXMAP	N/A	`none`
EZ_LABELPIXMAP	N/A	`null`
EZ_RAW_RGB_DATA	N/A	`null`
EZ_SHAPED_WINDOW	shapedWindow	`false`
EZ_SHAPE_FILE	shapeFile	`NULL`
EZ_SHAPE_PIXMAP	N/A	`NULL`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_SHORTCUT_KEY	shortcutKey	`null`

4.7.5 Menu Check Button Item

Menu check buttons are similar to check buttons, except they are used only on popup menus.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_MENU_CHECK_BUTTON`
Default instance name:	`"menuCheckButtonItem"`
Default Class name:	`"MenuCheckButtonItem"`

Widget Interface Routines

```
int     EZ_SetWidgetLabelString(EZ_Widget *widget, char *str);
```
set the textual label.

```
void    EZ_SetWidgetLabelPixmap(EZ_Widget *widget, EZ_Bitmap *pixmap);
```
set the pixmap label.

```
char    *EZ_GetWidgetLabelString(EZ_Widget *widget);
```
return the textual label.

```
void    EZ_SetCheckButtonState(EZ_Widget *widget, int onoff);
```
set the state of a check button.

```
int     EZ_GetCheckButtonState(EZ_Widget *widget, int *value);
```
retrieve the state and state value of a check button.

Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"MenuCheckButtonItem"`
EZ_NAME	name	`"menuCheckButtonItem"`
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`0`
EZ_PADX	padx	`8`
EZ_PADY	pady	`0`
EZ_IPADX	iPadx	`8`
EZ_IPADY	iPady	`6`
EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`
EZ_TRANSIENT	transient	`false`
EZ_ORIENTATION	orientation	`horizontal`
EZ_FILL_MODE	fillMode	`fillNone`
EZ_SIDE	side	`center`
EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*Bold*"`
EZ_LABEL_STRING	labelString	`null`
EZ_LABEL_POSITION	labelPosition	`center`
EZ_JUSTIFICATION	justification	`left`
EZ_TEXT_LINE_LENGTH	textLineLength	`30`

EZ_UNDERLINE	underline	`-1`
EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`

EZ_PIXMAP_FILE	pixmapFile	`null`
EZ_IMAGE_FILE	imageFile	`null`
EZ_BITMAP_DATA	N/A	`null`
EZ_PIXMAP_DATA	N/A	`null`
EZ_X_PIXMAP	N/A	`none`
EZ_LABELPIXMAP	N/A	`null`
EZ_RAW_RGB_DATA	N/A	`null`
EZ_SHAPED_WINDOW	shapedWindow	`false`
EZ_SHAPE_FILE	shapeFile	`NULL`
EZ_SHAPE_PIXMAP	N/A	`NULL`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_SHORTCUT_KEY	shortcutKey	`null`

EZ_INDICATOR_TYPE	indicatorType	`EZ_SQUARE_INDICATOR`
EZ_INDICATOR_COLOR	indicatorColor	`darkRed`
EZ_INDICATOR_SIZE_ADJUST	indicatorSizeAdjust	`3`

EZ_CHECK_BUTTON_ON_VALUE	checkButtonOnValue	`1`
EZ_CHECK_BUTTON_OFF_VALUE	checkButtonOffValue	`0`
EZ_CHECK_BUTTON_ON_OFF	checkButtonOnOff	`off`

4.7.6 Menu Radio Button Item

Menu radio buttons behave just like radio buttons, except they are used only on popup menus.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_MENU_RADIO_BUTTON`
Default instance name:	`"menuRadioButtonItem"`
Default Class name:	`"MenuRadioButtonItem"`

Widget Interface Routines

`int EZ_SetWidgetLabelString(EZ_Widget *widget, char *str);`
set the textual label.

`void EZ_SetWidgetLabelPixmap(EZ_Widget *widget, EZ_Bitmap *pixmap);`
set the pixmap label.

`char *EZ_GetWidgetLabelString(EZ_Widget *widget);`
return the textual label.

`void EZ_SetRadioButtonGroupVariableValue(EZ_Widget *widget, int value);`
set the value of the group variable corresponding to a radio button.

`int EZ_GetRadioButtonGroupVariableValue(EZ_Widget *widget);`
retrieve the value of the group variable corresponding to a radio button.

`int EZ_GetRadioButtonGroupID(EZ_Widget *widget);`
retrieve the group identifier of a radio button.

Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"MenuRadioButtonItem"`
EZ_NAME	name	`"menuRadioButtonItem"`
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`0`
EZ_PADX	padx	`8`
EZ_PADY	pady	`0`
EZ_IPADX	iPadx	`8`
EZ_IPADY	iPady	`6`
EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*Bold*"`
EZ_LABEL_STRING	labelString	`null`
EZ_LABEL_POSITION	labelPosition	`center`
EZ_JUSTIFICATION	justification	`left`
EZ_TEXT_LINE_LENGTH	textLineLength	`30`
EZ_UNDERLINE	underline	`-1`
EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`
EZ_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`

EZ_PIXMAP_FILE	pixmapFile	`null`
EZ_IMAGE_FILE	imageFile	`null`
EZ_BITMAP_DATA	N/A	`null`
EZ_PIXMAP_DATA	N/A	`null`
EZ_X_PIXMAP	N/A	`none`
EZ_LABELPIXMAP	N/A	`null`
EZ_RAW_RGB_DATA	N/A	`null`
EZ_SHAPED_WINDOW	shapedWindow	`false`
EZ_SHAPE_FILE	shapeFile	`NULL`
EZ_SHAPE_PIXMAP	N/A	`NULL`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_SHORTCUT_KEY	shortcutKey	`null`

EZ_INDICATOR_TYPE	indicatorType	`EZ_DIAMOND_INDICATIOR`
EZ_INDICATOR_COLOR	indicatorColor	`darkRed`
EZ_INDICATOR_SIZE_ADJUST	indicatorSizeAdjust	`3`

EZ_RADIO_BUTTON_GROUP	radioButtonGroup	
EZ_RADIO_BUTTON_VALUE	radioButtonValue	

4.8 Sliders

A slider widget is a sunken rectangle on which a button moves in one dimension, either vertically or horizontally. A slider appears as a linear scale with optional numerical labes and an optional short label. Sliders are typically used to edit a numerical value. The user adjust the value by clicking mouse button 1 or button 2 in the slider, or by dragging the slider button with mouse button 1. If a slider has callbacks, they are invoked whenever its numeric value changes.

Sliders may be displayed in 6 different styles controlled by `sliderStyle` resource. An optional set of tickmarks is controlled by the `showTick` resource.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_VERTICAL_SLIDER`
Default instance name:	`"verticalSlider"`
Default Class name:	`"VerticalSlider"`
Symbolic Widget type:	`EZ_WIDGET_HORIZONTAL_SLIDER`
Default instance name:	`"horizontalSlider"`
Default Class name:	`"HorizontalSlider"`

Widget Interface Routines

`void EZ_SetSliderValue(EZ_Widget *widget, float value);`

set the slider value.

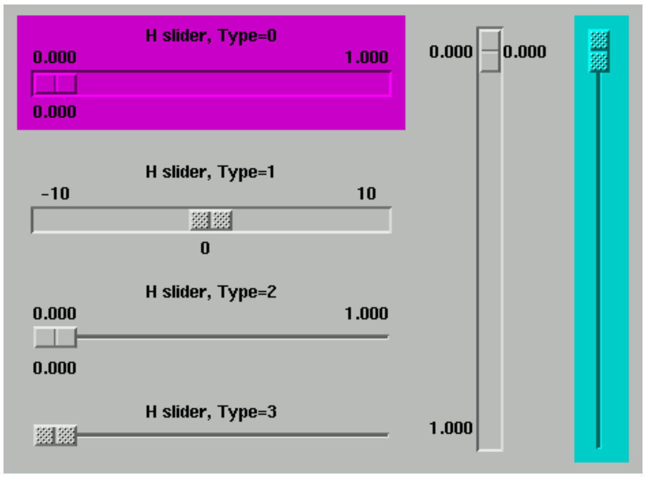
`float EZ_GetSliderValue(EZ_Widget *widget);`

retrieve the slider value.

`void EZ_SetSliderRange(EZ_Widget *widget, float f, float t);`

set the range of a slider.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"VerticalSlider"`
		or `"HorizontalSlider"`
EZ_NAME	name	`"verticalSlider"`
		or `"horizontalSlider"`

EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	

EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	

EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`2`

EZ_PADX	padx	`8`
EZ_PADY	pady	`6`
EZ_IPADX	iPadx	`0`
EZ_IPADY	iPady	`0`

EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`
EZ_TRANSIENT	transient	`false`

EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*Bold*"`
EZ_LABEL_STRING	labelString	`null`
EZ_TEXT_LINE_LENGTH	textLineLength	`30`

EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_CALLBACK	N/A	`null null`
EZ_MOTION_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_BG_IMAGE_FILE_B	bgImageFileB	`null`
EZ_BG_PIXMAP_B	N/A	`none`
EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_BACKING_STORE	backingStore	`false`

EZ_SLIDER_LENGTH	sliderLength	`100`
EZ_SLIDER_WIDTH	sliderWidth	`16`
EZ_SLIDER_BORDER_WIDTH	sliderBorderWidth	`2`
EZ_SLIDER_RESOLUTION	sliderResolution	`-1.0`
EZ_SLIDER_RANGE	sliderRange	`0.0 1.0`
EZ_SLIDER_INIT_VALUE	sliderInitialValue	`0.0`
EZ_SLIDER_DISPLAY_VALUE	sliderDisplayValue	`1`
EZ_SHOW_TICKS	showTick	`0 0`
EZ_SLIDER_STYLE	sliderStyle	`0`

4.9 Entry

An entry is a one line text editor. It is typically used to obtain textual input from user, such as a file name.

To enter text into an entry the user moves the pointer into the entry, causing the entry to take focus and making a red cursor to appear. The user can then enter characters and they will be inserted at the point of the cursor. The cursor can be moved by clicking mouse button 1 anywhere in the entry's text. Text in an entry can be selected by pressing and dragging mouse button 1. Selections can be pasted into an entry by pressing mouse button 2 in the entry. The entry widget supports a few emacs like editing commands.

When the return key or the tab key is pressed in an entry, and if it has any callbacks, the callbacks are invoked.

If motion callbacks are registered to an entry widget, they are invoked whenever the text string in the entry field is modified.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_ENTRY`
Default instance name:	`"entry"`
Default Class name:	`"Entry"`

Widget Interface Routines

`void EZ_SetEntryString(EZ_Widget *widget, char *str);`

set or replace the text in an entry widget.

`char *EZ_GetEntryString(EZ_Widget *widget);`

return the text in an entry widget. Don't modify the returned text !

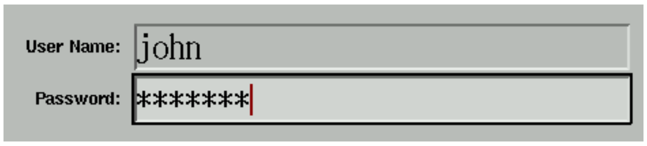
`void EZ_OnOffSecretEntry(EZ_Widget *widget, int onoff);`

toggle secret entry on and off.

Editing Routines

```
void  EZ_EntryEditDeleteChar(EZ_Widget *widget, int direction);
void  EZ_EntryEditBeginOfInput(EZ_Widget *widget);
void  EZ_EntryEditBeginningOfInput(EZ_Widget *widget);
void  EZ_EntryEditEndOfInput(EZ_Widget *widget);
void  EZ_EntryEditForeward(EZ_Widget *widget);
void  EZ_EntryEditBackward(EZ_Widget *widget);
void  EZ_EntryEditSetMarker(EZ_Widget *widget);
void  EZ_EntryEditKillRegion(EZ_Widget *widget);
void  EZ_EntryEditKillEndOfLine(EZ_Widget *widget, int flag);
void  EZ_EntryCopySelectedText(EZ_Widget *widget, int start, int end);
void  EZ_EntryPasteSelection(EZ_Widget *widget);
void  EZ_EntryEditInsertString (EZ_Widget *widget, char *str);
void  EZ_EntryEditForwardWord(EZ_Widget *widget);
void  EZ_EntryEditBackwardWord(EZ_Widget *widget);
void  EZ_EntryEditGotoPosition(EZ_Widget *widget, int p);
```

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"Entry"`
EZ_NAME	name	`"entry"`

EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	

EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	

EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`2`

EZ_PADX	padx	`0`
EZ_PADY	pady	`3`
EZ_IPADX	iPadx	`0`
EZ_IPADY	iPady	`0`

EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`
EZ_TRANSIENT	transient	`false`

EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*fixed*"`
EZ_LABEL_STRING	labelString	`null`
EZ_TEXT_LINE_LENGTH	textLineLength	`30`

EZ_CURSOR	cursor	`"XC_xterm"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_CALLBACK	N/A	`null null`
EZ_MOTION_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_ENTRY_STRING	entryString	`null`
EZ_ENTRY_EDITABLE	entryEditable	`true`

EZ_GRID_CELL_GEOMETRY	gridCellGeometry	
EZ_GRID_CELL_PLACEMENT	gridCellPlacement	

4.10 Scrollbars

A scrollbar is used to scroll an object. To each scrollbar, there are three state variables: a total size, a display size and a starting position. Total size is the size of the object being scrolled, e.g. the total number of lines in a text widget; display size is the maximal size can be displayed simultaneously in a widget window, e.g. the number of lines in a text widget window; and starting position is the starting position of the object in the scrolled window, e.g. the index of the first line displayed in the text widget window.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_VERTICAL_SCROLLBAR`
Default instance name:	`"verticalScrollbar"`
Default Class name:	`"VerticalScrollbar"`

Symbolic Widget type:	`EZ_WIDGET_HORIZONTAL_SCROLLBAR`
Default instance name:	`"horizontalScrollbar"`
Default Class name:	`"HorizontalScrollbar"`

Widget Interface Routines

`void EZ_UpdateScrollbar(EZ_Widget *widget, int maxsz, int pagsz, int start);`

update a scrollbar.

`void EZ_DoScroll(EZ_Widget *widget, int *offset, int mode);`

scroll a scrollbar.

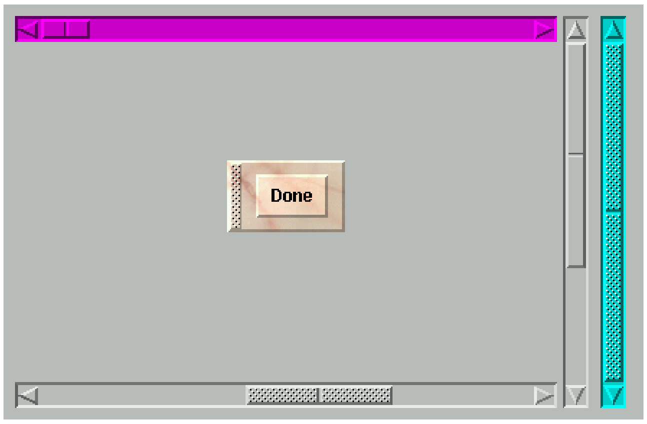
`void EZ_GetScrollbarState(EZ_Widget *widget, int *ms, int *ps, int *st);`

retrieve the state of a scrollbar.

`int EZ_SetScrollbarDiscreteSpeed(EZ_Widget *widget, int speed);`

set the scroll step size. It's the speed of scrolling when the arrow is pressed.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"VerticalScrollbar"`
		or `"HorizontalScrollbar"`
EZ_NAME	name	`"verticalScrollbar"`
		or `"horizontalScrollbar"`

EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	

EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	

EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	`2`
EZ_BORDER_TYPE	borderType	`sunken`
EZ_FOCUS_PAD	focusPad	`2`

EZ_PADX	padx	`0`
EZ_PADY	pady	`0`
EZ_IPADX	iPadx	`0`
EZ_IPADY	iPady	`0`

EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`
EZ_TRANSIENT	transient	`false`

EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_CALLBACK	N/A	`null null`
EZ_MOTION_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_BG_IMAGE_FILE_B	bgImageFileB	`null`
EZ_BG_PIXMAP_B	N/A	`none`
EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_SCROLLBAR_WIDTH	scrollbarWidth	`10`
EZ_SCROLLBAR_BORDER_WIDTH	scrollbarBorderWidth	`2`
EZ_HSCROLL_INCREMENT	hScrollIncrement	`1`
EZ_VSCROLL_INCREMENT	vScrollIncrement	`1`
EZ_GRIP_BUTTON_STYLE	gripButtonStyle	`0`

4.11 ListBoxes

A listbox contains a list of items formatted into rows. When one of the items is selected, it is highlighted. When a double click of mouse button 1 occures on an item, the callbacks of the listbox are invoked.

The contents of a listbox is scrollable in both directions.

There are two types of listboxes in *EZwgl*, an internal listbox and a composite listbox. The latter is a composite widget made from a frame, an internal listbox and two scrollbars. From the application programmer's point of view, these two listboxes are the same.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_LISTBOX`
Default instance name:	`"listBox"`
Default Class name:	`"ListBox"`

Symbolic Widget type:	`EZ_WIDGET_ILISTBOX`
Default instance name:	`"iListBox"`
Default Class name:	`"IListBox"`

Subwidgets Info

Subwidgets of ListBox are listed below.

```
          ( iListBox
listBox-->( horizontalScrollbar
          ( verticalScrollbar
```

Widget Interface Routines

```
void  EZ_GetListBoxWidgetComponents(EZ_Widget *widget, EZ_Widget **ilb,
                                    EZ_Widget **hsc,  EZ_Widget **vsc);
```
return the subwidgets of a composite listbox.

```
void  EZ_ClearListBox(EZ_Widget *widget);
```
clear a listbox.

```
void  EZ_SetListBoxItems(EZ_Widget *widget, char **items, int nitems);
```
set up the list in a listbox.

```
void  EZ_AppendListBoxItem(EZ_Widget *widget, char *item);
```
append one item at the end of a listbox.

```
void  EZ_InsertListBoxItem(EZ_Widget *widget, char *str, int idx);
```
insert an item at the given location.

```
void  EZ_DeleteListBoxItem(EZ_Widget *widget, int idx);
```
delete the item at the specified location.

```
void  EZ_DeleteListBoxString(EZ_Widget *widget, char *str);
```
find the item and delete it.

```
char  *EZ_GetListBoxSelectedItem(EZ_Widget *widget);
char  *EZ_GetListBoxSelection(EZ_Widget *widget);
```
return the selected item in a listbox.

```
int   EZ_GetListBoxSelectionLineNumber(EZ_Widget *widget);
```
return the selection index.

```
void  EZ_ModifyListBoxItem(EZ_Widget *widget, char *nitem, int idx);
```
replace the item at the specified location by the given string.

```
void  EZ_ListBoxSelectItem(EZ_Widget *widget, char *str);
```

find the specified item and make it the current selection.

```
void  EZ_ListBoxSelectItemUsingIdx(EZ_Widget *widget, int idx);
```

make the item at the specified location the current selection.

```
void  EZ_GetListBoxState (EZ_Widget *widget, int *xoff, int *line1);
```

return the current xoffset and yoffset (index of the first line).

```
void  EZ_SetListBoxState (EZ_Widget *widget, int xoff, int line1);
```

set the current xoffset and yoffset (index of the first line).

```
void  EZ_GetListBoxData(EZ_Widget *widget, listboxentry **dat, int *cnt);
```

return the internal listbox data. One should not modify them.

Screenshot

Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	"ListBox"
		or"IListBox"
EZ_NAME	name	"listBox"
		or"iListBox"
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	
EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	
EZ_BORDER_WIDTH	borderWidth	0
EZ_BORDER_TYPE	borderType	none
EZ_FOCUS_PAD	focusPad	0
EZ_PADX	padx	0 or 8
EZ_PADY	pady	0 or 6
EZ_IPADX	iPadx	0
EZ_IPADY	iPady	0

EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`
EZ_TRANSIENT	transient	`false`

EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*Bold*"`
EZ_TEXT_LINE_LENGTH	textLineLength	`30`

EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_CALLBACK	N/A	`null null`
EZ_MOTION_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_BG_IMAGE_FILE_B	bgImageFileB	`null`
EZ_BG_PIXMAP_B	N/A	`none`
EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_SCROLLBAR_WIDTH	scrollbarWidth	`10`
EZ_SCROLLBAR_BORDER_WIDTH	scrollbarBorderWidth	`2`
EZ_HSCROLL_INCREMENT	hScrollIncrement	`1`
EZ_VSCROLL_INCREMENT	vScrollIncrement	`1`

EZ_SELECTION_BACKGROUND	selectionBackground	
EZ_SELECTION_FOREGROUND	selectionForeground	
EZ_TEXT_BACKGROUND	textBackground	
EZ_ROW_BG	rowBg	`off nil nil`

EZ_OPTIONAL_HSCROLLBAR	optionalHScrollbar	`true`
EZ_OPTIONAL_VSCROLLBAR	optionalVScrollbar	`true`

4.12 Text Widgets

A text widget provides a window that will allow an application to display and edit one or more lines of text. Text widgets provide a minimal set of the common (emacs-like) keyboard editing commands as well as the X11 selection mechanism.

Text widgets also support a set of hardcoded text properties. Current supported text properties are: *font*, *underlining*, *bitmap/pixmap*, *background color* and *foreground color*.

Two text widgets are provided in *EZwgl*, an internal text widget and a composite text widget.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_TEXT`
Default instance name:	`"text"`
Default Class name:	`"Text"`

Symbolic Widget type:	`EZ_WIDGET_ITEXT`
Default instance name:	`"iText"`
Default Class name:	`"IText"`

Subwidgets Info

Subwidgets of Text are listed below.

```
       ( iText
text-->( horizontalScrollbar
       ( verticalScrollbar
```

Widget Interface Routines

```
void EZ_GetTextWidgetComponents(EZ_Widget *widget, EZ_Widget **itxt,
                             EZ_Widget **hscr, EZ_Widget **vscr );
```

return the components of a composite text widget.

```
void EZ_TextLoadFile(EZ_Widget *widget,char *fname);
```

read a text file into a text widget.

```
void EZ_TextClear(EZ_Widget *widget);
```

clear a text widget.

```
int  EZ_TextSaveText (EZ_Widget *widget, char *file)
```

save the contents of a text widget to a file.

```
int  EZ_TextBufferModified(EZ_Widget *widget);
```

return 1 if buffer has been modifed since last save.

```
char *EZ_TextGetBufferString (EZ_Widget *widget);
```

returns the text in a text widget.

Editing Commands

```
int   EZ_TextSetMarker (EZ_Widget *widget):
int   EZ_TextGetPosition (EZ_Widget *widget):
void  EZ_TextGotoPosition (EZ_Widget *widget, int pos):
char  *EZ_TextGetRegionString (EZ_Widget *widget):

void EZ_TextNextLine(EZ_Widget *widget):
void EZ_TextPreviousLine(EZ_Widget *widget):
void EZ_TextBackwardChar (EZ_Widget *widget):
void EZ_TextForwardChar (EZ_Widget *widget):
```

```
void EZ_TextForwardWord (EZ_Widget *widget):
void EZ_TextBackwardWord (EZ_Widget *widget):
void EZ_TextPreviousPage (EZ_Widget *widget):
void EZ_TextNextPage (EZ_Widget *widget):

void EZ_TextEndOfLine (EZ_Widget *widget):
void EZ_TextBeginningOfLine (EZ_Widget *widget):
void EZ_TextEndOfBuffer (EZ_Widget *widget):
void EZ_TextBeginningOfBuffer (EZ_Widget *widget):

void EZ_TextInsertChar (EZ_Widget *widget, char c):
void EZ_TextDeletePreviousChar (EZ_Widget *widget):
void EZ_TextDeleteNextChar (EZ_Widget *widget):
void EZ_TextInsertNewLine (EZ_Widget *widget):

void EZ_TextKillEndOfLine (EZ_Widget *widget):
void EZ_TextKillRegion (EZ_Widget *widget):
void EZ_TextPasteSelection (EZ_Widget *widget):
void EZ_TextReDisplay (EZ_Widget *widget):
void EZ_TextInsertString (EZ_Widget *widget, char *str):

void EZ_TextSetRegionFont (EZ_Widget *widget, char *fontname):
void EZ_TextSetRegionForeground (EZ_Widget *widget, char *clr):
void EZ_TextSetRegionBackground (EZ_Widget *widget, char *clr):
void EZ_TextSetRegionUnderline (EZ_Widget *widget, int udl):

void EZ_TextSetRegionProperty (EZ_Widget *widget, EZ_TextProperty *prop):
void EZ_TextReplaceRegionWithPixmap (EZ_Widget *widget, char *file):
void EZ_TextInsertStringWithProperty (EZ_Widget *widget, char *str,
                                      EZ_TextProperty *prop):
```

Screenshot

Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	"Text"
		or "IText"
EZ_NAME	name	"text"
		or "iText"

EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	

EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	

EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	`0` or `2`
EZ_BORDER_TYPE	borderType	`none` or `sunken`
EZ_FOCUS_PAD	focusPad	`0` or `2`

EZ_PADX	padx	`8` or `0`
EZ_PADY	pady	`6` or `0`
EZ_IPADX	iPadx	`0`
EZ_IPADY	iPady	`0`

EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`
EZ_TRANSIENT	transient	`false`

EZ_FONT_ID	fontID	`3`
EZ_FONT_NAME	fontName	`"*fixed*"`

EZ_CURSOR	cursor	`"XC_xterm"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_CALLBACK	N/A	`null null`
EZ_MOTION_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_BG_IMAGE_FILE_B	bgImageFileB	`null`
EZ_BG_PIXMAP_B	N/A	`none`
EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_SCROLLBAR_WIDTH	scrollbarWidth	`10`
EZ_SCROLLBAR_BORDER_WIDTH	scrollbarBorderWidth	`2`
EZ_HSCROLL_INCREMENT	hScrollIncrement	`1`
EZ_VSCROLL_INCREMENT	vScrollIncrement	`1`

EZ_SELECTION_BACKGROUND	selectionBackground	
EZ_TEXT_BACKGROUND	textBackground	
EZ_TEXT_WIDGET_EDITABLE	textWidgetEditable	`true`
EZ_TEXT_SPACING	textSpacing	`0`
EZ_WRAP_TEXT	wrapText	`1`

EZ_OPTIONAL_HSCROLLBAR	optionalHScrollbar	`true`
EZ_OPTIONAL_VSCROLLBAR	optionalVScrollbar	`true`

4.13 Notebook

A notebook is a special container widget that allows you to pack large interface into notebook pages. A notebook widget is a composite widget consists of three frames, a base frame and two sub-frames stacked vertically.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_NOTEBOOK`
Default instance name:	`"notebook"`
Default Class name:	`"Notebook"`

Subwidget Info

Instance names of subwidgets are listed below.

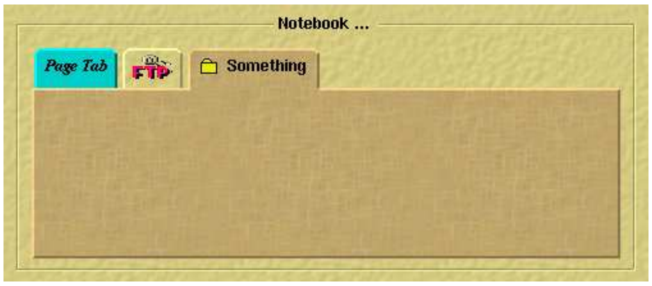
```
            ( topFrame
notebook-->(
            ( bottomFrame
```

Widget Interface Routines

`int EZ_SetNoteBookOrientation(EZ_Widget *widget, int orientation);`

set the orientation of a notebook.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"Notebook"`
EZ_NAME	name	`"notebook"`

EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	

EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	

EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`0`

EZ_PADX	padx	`0`
EZ_PADY	pady	`0`
EZ_IPADX	iPadx	`0`
EZ_IPADY	iPady	`0`

EZ_EXPAND	expand	`false`
EZ_PROPAGATE	propagate	`true`

EZ_ORIENTATION	orientation	`verticalTop`
EZ_FILL_MODE	fillMode	`fillBoth`
EZ_SIDE	side	`left`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

4.14 Notebook Page

A notebook page behaves like a regular frame. It looks like a file folder, with a tab positioned somewhere over the top edge. When mouse button 1 is pressed and released over the tab, the corresponding page (frame) is activated and the current active page is de-activated.

Notebook page is a special composite widget. It is made from a button (tab) and a frame (page).

Widget Info

Symbolic Widget type:	`EZ_WIDGET_NOTEBOOK_PAGE`
Default instance name:	`"notebookPage"`
Default Class name:	`"NotebookPage"`

Subwidgets Info

A notebook page can only be the child of a notebook widget. The subwidgets of a notebook page, a button and a frame, are subwidgets of the notebook.

```
           ( topFrame--> notebookPageTab
notebook-->(
           ( bottomFrame--> notebookPage
```

Widget Interface Routines

```
void   EZ_ActivateNoteBookPage(EZ_Widget *widget);
```

activate the specified page on a notebook widget.

```
EZ_Widget *EZ_GetNoteBookPageTab(EZ_Widget *widget);
```

return the tab (a normal button) of a notebook page.

Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"NotebookPage"`
EZ_NAME	name	`"notebookPage"`
EZ_BORDER_WIDTH	borderWidth	`2`
EZ_BORDER_TYPE	borderType	`raised`
EZ_FOCUS_PAD	focusPad	`2 or 0`
EZ_PADX	padx	`8`
EZ_PADY	pady	`6`
EZ_IPADX	iPadx	`0`
EZ_IPADY	iPady	`0`
EZ_ORIENTATION	orientation	`horizontal`
EZ_FILL_MODE	fillMode	`fillNone`
EZ_SIDE	side	`center`
EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`
EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

4.15 PaneHandle

A paneHandle is a special widget that is used to divide its parent into panes. The size of the panes can be adjusted by moving the paneHandle.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_HANDLE`
Default instance name:	`"paneHandle"`
Default Class name:	`"PaneHandle"`

Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"PaneHandle"`
EZ_NAME	name	`"paneHandle"`
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`1`
EZ_PADX	padx	`0`
EZ_PADY	pady	`0`
EZ_EXPAND	expand	`true`
EZ_PROPAGATE	propagate	`true`
EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`
EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	
EZ_BG_IMAGE_FILE_B	bgImageFileB	`null`
EZ_BG_PIXMAP_B	N/A	`none`
EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`
EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`
EZ_GRIP_BUTTON_STYLE	gripButtonStyle	`0`

4.16 Optional Entry

An optional entry is also refered as a ComboBox. It is a composite widget built from an entry, a button and a listbox. It allows users to enter text via the entry and also provides a list of possible choices via the listbox. The entry can be either editable or non-editable. Option is also provided so that the listbox remembers the history of selected information.

There are two display styles (represented by 0 and 1) for an optional entry. The difference is the location where the arrow button is displayed. In the default style, the button is displayed outside the entry.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_OPTIONAL_ENTRY`
Default instance name:	`"optionalEntry"`
Default Class name:	`"OptionalEntry"`

Subwidgets Info

Instance names of subwidgets are listed below.

```
                 ( entry ....
                 (           arrowButton
 optionalEntry-->(   ........
                 (         ( iListBox
           ....... popup-->( horizontalScrollbar
                           ( verticalScrollbar
```

Widget Interface Routines

```
void EZ_GetOptionalEntryWidgetComponents(EZ_Widget *widget, EZ_Widget **entry,
                                       EZ_Widget **listbox,  EZ_Widget **btn);
```

return the components of an optional entry.

```
void EZ_SetOptionalEntryString(EZ_Widget *widget, char *str);
```

set the text of an optional entry.

```
char *EZ_GetOptionalEntryString(EZ_Widget *widget);
```

retrieve the text in an optional entry.

```
void  EZ_OptionalEntryClearOptions(EZ_Widget *widget);
```

clear the listbox associated to the optional entry.

```
void  EZ_SetOptionalEntryOptions(EZ_Widget *widget, char **items, int count);
```

set items in the listbox associated to an optional entry.

```
void  EZ_OptionalEntryAppendOption(EZ_Widget *widget, char *item);
```

append an item at the end of the listbox associated to the optional entry.

```
void  EZ_OptionalEntryInsertOption(EZ_Widget *widget, char *str, int idx);
```

insert an item into the listbox at the specified location.

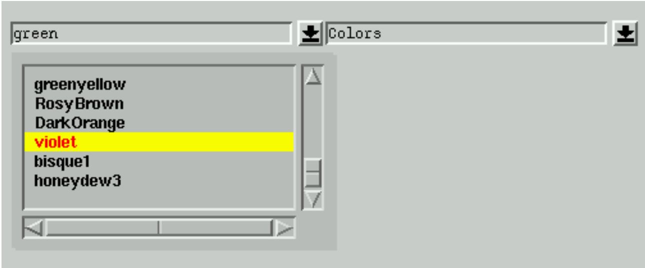
```
void  EZ_OptionalEntryDeleteOption    (EZ_Widget *widget, char *str);
```

delete the specified item in the associated listbox.

```
void  EZ_SetOptionalEntryStyle(EZ_Widget *widget, int style);
```

select one of the two display styles.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"OptionalEntry"`
EZ_NAME	name	`"optionalEntry"`
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	
EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	
EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`0`
EZ_PADX	padx	`0`
EZ_PADY	pady	`0`
EZ_IPADX	iPadx	`2`
EZ_IPADY	iPady	`0`
EZ_ORIENTATION	orientation	`horizontal`
EZ_FILL_MODE	fillMode	`fillHorizontally`
EZ_SIDE	side	`center`
EZ_FONT_ID	fontID	
EZ_FONT_NAME	fontName	`"*fixed*"`
EZ_LABEL_STRING	labelString	`null`
EZ_CURSOR	cursor	`"XC_xterm"`
EZ_BUBBLE_STRING	bubbleString	`null`
EZ_CALLBACK	N/A	`null null`
EZ_MOTION_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`
EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_BG_IMAGE_FILE_B	bgImageFileB	`null`
EZ_BG_PIXMAP_B	N/A	`none`
EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_SELECTION_BACKGROUND	selectionBackground	
EZ_SELECTION_FOREGROUND	selectionForeground	

EZ_ENTRY_STRING	entryString	`null`

EZ_POPUP_NAME	popupName	`"popup"`

EZ_ENTRY_EDITABLE	entryEditable	`true`
EZ_OPTIONAL_ENTRY_HISTORY	optionalEntryHistory	`true`
EZ_OPTIONAL_ENTRY_STYLE	optionalEntryStyle	`0`

4.17 FileSelectors

A fileSelector allows users to select a filename interactively. It is a composite widget consists of six frames, two labels, two listboxes, two optional entries and three buttons. This widget uses the gnu glob library and the gnu tilde library (both are included).

Widget Info

Symbolic Widget type:	`EZ_WIDGET_FILE_SELECTOR`
Default instance name:	`"fileSelector"`
Default Class name:	`"FileSelector"`

Subwidgets Info

Instance names of subwidgets are listed below.

```
                (           ( filterLabel
                (           (                          ( iListBox
                (  frameA-->( filterEntry ...popupA-->( horizontalScrollbar
                (                                      ( verticalScrollbar
                (          (           ( directoryLabel
                (          (           (                     ( iListBox
                (          ( frameB1-->( directoryListBox-->( horizontalScrollbar
                (          (                                 ( verticalScrollbar
                (          (           ( fileLabel
                (          (           (              ( iListBox
 fileSelector-->( frameB-->( frameB2-->( fileListBox-->( horizontalScrollbar
                (          (           (              ( verticalScrollbar
                (
                (          ( selectionLabel
                (          (                            ( iListBox
                ( frameC-->( selectionEntry ...popupB-->( horizontalScrollbar
                (                                       ( verticalScrollbar
                (          ( okButton
                ( frameD-->( filterButton
                (          ( hideButton
```

Widget Interface Routines

```
void EZ_GetFileSelectorWidgetComponents(EZ_Widget *widget,
                EZ_Widget **ptnEntry,   EZ_Widget **fileEntry,
                EZ_Widget **dirListbox, EZ_Widget **fileListbox,
                EZ_Widget **okBtn,      EZ_Widget **filterBtn,
                EZ_Widget **cancelBtn);
```

return the components of a file selector.

```
void EZ_SetFileSelectorInitialPattern(EZ_Widget *widget, char *pattern);
```

set the glob pattern of a file selector.

```
char *EZ_GetFileSelectorSelection(EZ_Widget *widget);
```

return the current selection of a file selector.

Screenshot

Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"FileSelector"`
EZ_NAME	name	`"fileSelector"`

EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	

EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	

EZ_WIDTH_HINT	widthHint	300
EZ_HEIGHT_HINT	heightHint	400
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	0
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	0

EZ_PADX	padx	2
EZ_PADY	pady	4
EZ_IPADX	iPadx	0
EZ_IPADY	iPady	4

EZ_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_HIGHLIGHT_FOREGROUND	highlightForeground	
EZ_HIGHLIGHT_BACKGROUND	highlightBackground	

EZ_BG_IMAGE_FILE_B	bgImageFileB	`null`
EZ_BG_PIXMAP_B	N/A	`none`
EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_GLOB_PATTERN	globPattern	`null`

4.18 Tree Widget

A tree widget lets the user display a "tree" object such as a file tree.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_TREE`
Default instance name:	`"tree"`
Default Class name:	`"Tree"`

Symbolic Widget type:	`EZ_WIDGET_ITREE`
Default instance name:	`"iTree"`
Default Class name:	`"ITree"`

Subwidgets Info

Subwidgets of Tree are listed below.

```
       ( iTree
Tree-->( horizontalScrollbar
       ( verticalScrollbar
```

Widget Interface Routines

```
void EZ_GetListTreeWidgetComponents(EZ_Widget *widget,
                       EZ_Widget **itr, EZ_Widget **hsc, EZ_Widget **vsc);
```

return the components of a Tree widget.

`void EZ_SetListTreeWidgetTree(EZ_Widget *widget, EZ_TreeNode *tree);`

set the tree object to be displayed in a tree widget.

`EZ_TreeNode    *EZ_GetListTreeWidgetTree(EZ_Widget *widget);`

return the tree object displayed in a tree widget.

`EZ_TreeNode    *EZ_GetListTreeWidgetSelection(EZ_Widget *widget);`

return the selected node in a tree widget.

`void EZ_ListTreeWidgetSelectNode(EZ_Widget *widget, EZ_TreeNode *theNode, int *location);`

set the selection in a tree widget and scroll to make the selection visible.

`void EZ_ListTreeWidgetSelectNodeUsingItem(EZ_Widget *widget, EZ_Item *theItem, int *location);`

set the selection in a tree widget and scroll to make the selection visible.

`void EZ_UpdateDirTreeWidget(EZ_Widget *widget);`

if directory tree is displayed, update the directory tree.

`void EZ_GetListTreeWidgetXYOffset (EZ_Widget *widget, int *x, int *y);`

return the XY display offset of a tree widget.

`void EZ_SetListTreeWidgetXYOffset (EZ_Widget *widget, int x, int y);`

set the XY display offset of a tree widget and update the display.

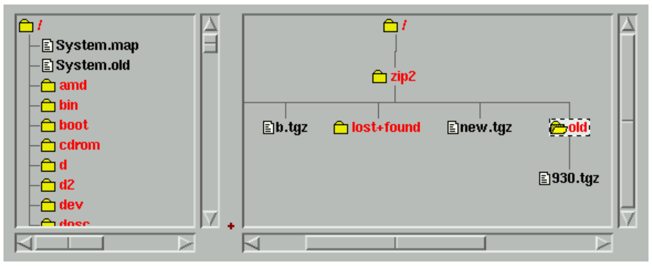
`EZ_TreeNode *EZ_ListTreeFindNodeAtXY(EZ_Widget *tree, int x, int y);`

find and return the node displayed at the specified location.

`void EZ_SetTreeWidgetRowBg(EZ_Widget *widget, int onoff, char *bg1, char *bg2);`

set the alternating background colors for a tree widget.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"Tree"`
		or `"ITree"`
EZ_NAME	name	`"tree"`
		or `"iTree"`
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	
EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	
EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`0`
EZ_PADX	padx	`0`
EZ_PADY	pady	`0`
EZ_IPADX	iPadx	`0`
EZ_IPADY	iPady	`0`
EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`
EZ_CALLBACK	N/A	`null null`
EZ_MOTION_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`
EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_BG_IMAGE_FILE_B	bgImageFileB	`null`
EZ_BG_PIXMAP_B	N/A	`none`
EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`
EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_SCROLLBAR_WIDTH	scrollbarWidth	`10`
EZ_SCROLLBAR_BORDER_WIDTH	scrollbarBorderWidth	`2`
EZ_HSCROLL_INCREMENT	hScrollIncrement	`1`
EZ_VSCROLL_INCREMENT	vScrollIncrement	`1`

EZ_SELECTION_BACKGROUND	selectionBackground	
EZ_SELECTION_FOREGROUND	selectionForeground	
EZ_TEXT_BACKGROUND	textBackground	

EZ_OPTIONAL_HSCROLLBAR	optionalHScrollbar	`true`
EZ_OPTIONAL_VSCROLLBAR	optionalVScrollbar	`true`

EZ_CHECK_BUTTON_ON_VALUE	checkButtonOnValue	`1`
EZ_CHECK_BUTTON_OFF_VALUE	checkButtonOffValue	`0`
EZ_CHECK_BUTTON_ON_OFF	checkButtonOnOff	`off`

EZ_RADIO_BUTTON_GROUP	radioButtonGroup	
EZ_RADIO_BUTTON_VALUE	radioButtonValue	

EZ_HIGHLIGHT_MODE	highlightMode	`0`
EZ_ROW_BG	rowBg	`off nil nil`

EZ_RUBBER_BAND	rubberBand	`false`

4.19 Fancy ListBoxes

A fancyListBox extend the functionalities of a listBox. It displays multi-columns of display-items with an optional header row. The contents of a fancyListBox can be scrolled in both directions.

There are two fancyListBoxes in *EZwgl*, an internal fancyListBox and a composite fancyListBox.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_FANCY_LISTBOX`
Default instance name:	`"fancyListBox"`
Default Class name:	`"FancyListBox"`

Symbolic Widget type:	`EZ_WIDGET_IFANCY_LISTBOX`
Default instance name:	`"iFancyListBox"`
Default Class name:	`"IFancyListBox"`

Subwidgets Info

Subwidgets of FancyListBox are listed below.

```
                ( iFancyListBox
FancyListBox-->( horizontalScrollbar
                ( verticalScrollbar
```

Widget Interface Routines

```
void EZ_GetFancyListBoxWidgetComponents(EZ_Widget *widget,
                  EZ_Widget **iflistBox,
                  EZ_Widget **hscroll, EZ_Widget **vscroll);
```

return the components of a FancyListBox.

```
EZ_Item    **EZ_GetFancyListBoxSelection(EZ_Widget *widget);
```

return the current selection in a fancyListBox.

```
EZ_Item *EZ_GetFancyListBoxItemUsingIdx (EZ_Widget *widget,
                                         int row, int col);
```

return the item at the specified location.

```
void EZ_GetFancyListBoxSelectionIdx(EZ_Widget *widget, int *row, int *col);
```

return the row and column index of the current selection in a fancyListBox.

```
void EZ_SetFancyListBoxData (EZ_Widget *widget, EZ_Item **data, int nrow, int ncol);
```

set the array of display items to be displayed in a fancyListBox.

```
void EZ_FancyListBoxClear (EZ_Widget *widget);
```

clear a fancyListBox.

```
void EZ_GetFancyListBoxXYOffset (EZ_Widget *widget, int *xoff, int *yoff);
```

return the XY display offset of a fancyListBox

```
void EZ_SetFancyListBoxXYOffset (EZ_Widget *widget, int xoff, int yoff);
```

set the XY display offset of a fancyListBox

```
void EZ_SetFancyListBoxHeader (EZ_Widget *widget, EZ_Item **header, int nelts);
```

set the header row of a fancyListBox.

```
void EZ_FancyListBoxInsertRow (EZ_Widget *widget, EZ_Item **items,
                                     int nitems, int where);
```

insert a row of display items into a fancyListBox.

```
void EZ_FancyListBoxDeleteRow (EZ_Widget *widget, int idx);
```

delete the specified row in a fancyListBox.

```
void EZ_FancyListBoxDeleteSomeRows(EZ_Widget *widget, int *idces, int cnt);
```

delete some rows from a fancyListBox.

```
void EZ_FancyListBoxReplaceItem(EZ_Widget *widget, EZ_Item *old,
                                EZ_Item *newItem);
```

replace an item with a new item.

```
void EZ_FancyListBoxReplaceItemUsingIdx(EZ_Widget *widget, int row,
                                        int col, EZ_Item *newItem);
```

replace the item at the specified location by a new item.

```
void EZ_FancyListBoxSelectItemUsingIdx(EZ_Widget *widget, int Srow,
                                       int Scol, int *location);
```

select the specified item and display it at the specified location.

```
void EZ_FancyListBoxSelectItem(EZ_Widget *widget, EZ_Item *item,
                               int *location);
```

select the specified item and display it at the specified location.

```
int EZ_GetFancyListBoxNumberOfRows (EZ_Widget *widget);
```

return the number of rows in a fancyListBox, counting the header row.

```
int EZ_GetFancyListBoxNumberOfColumns (EZ_Widget *widget);
```

return the number of columns in a fancyListBox.

```
EZ_Item ***EZ_GetFancyListBoxData (EZ_Widget *widget);
```

return the array of display items in a fancyListBox. Don't modify them.

```
EZ_Item **EZ_FancyListBoxFindRowAtXY(EZ_Widget *fl, int x,
                                      int y, int *r_ret);
```

return the row of items displayed at the specified location.

```
EZ_Item *EZ_FancyListBoxFindItemAtXY(EZ_Widget *fl, int x, int y,
                                     int *r_ret, int *c_ret);
```

return the item displayed at the specified location.

```
int EZ_FancyListBoxGetHighlightedRows(EZ_Widget *widget, int *ary,
                                                        int size);
```

return the row indices of highlighted items in a fancyListBox.

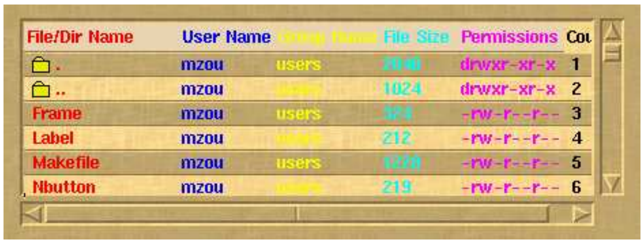
```
int EZ_FancyListBoxGetHighlightedItems(EZ_Widget *widget,
                                    EZ_Item **arg,int siz);
```

return the highlighted items in a fancyListBox.

```
void  EZ_SetFancyListBoxRowBg(EZ_Widget *widget, int mode,
                                     char *bg1, char *bg2);
```

set the alternating background colors.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	"FancyListBox"
		or "IFancyListBox"
EZ_NAME	name	"fancyListBox"
		or "iFancyListBox"
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	
EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	
EZ_BORDER_WIDTH	borderWidth	0
EZ_BORDER_TYPE	borderType	none
EZ_FOCUS_PAD	focusPad	0
EZ_PADX	padx	0
EZ_PADY	pady	0
EZ_IPADX	iPadx	0
EZ_IPADY	iPady	0
EZ_CURSOR	cursor	"XC_left_ptr"
EZ_BUBBLE_STRING	bubbleString	null
EZ_CALLBACK	N/A	null null
EZ_MOTION_CALLBACK	N/A	null null
EZ_DESTROY_CALLBACK	N/A	null null
EZ_EVENT_HANDLER	N/A	null null
EZ_FOREGROUND	foreground	"black"
EZ_BACKGROUND	background	"gray74
EZ_BG_IMAGE_FILE	bgImageFile	null
EZ_BG_PIXMAP	N/A	none
EZ_BG_IMAGE_FILE_B	bgImageFileB	null
EZ_BG_PIXMAP_B	N/A	none
EZ_CLIENT_PTR_DATA	N/A	null
EZ_CLIENT_INT_DATA	clientIntData	0
EZ_DND_DRAG_CURSOR	dndDragCursor	none
EZ_DND_BUBBLE_STRING	dndBubbleString	null

EZ_SCROLLBAR_WIDTH	scrollbarWidth	`10`
EZ_SCROLLBAR_BORDER_WIDTH	scrollbarBorderWidth	`2`
EZ_HSCROLL_INCREMENT	hScrollIncrement	`1`
EZ_VSCROLL_INCREMENT	vScrollIncrement	`1`

EZ_SELECTION_BACKGROUND	selectionBackground	
EZ_SELECTION_FOREGROUND	selectionForeground	
EZ_TEXT_BACKGROUND	textBackground	

EZ_OPTIONAL_HSCROLLBAR	optionalHScrollbar	`true`
EZ_OPTIONAL_VSCROLLBAR	optionalVScrollbar	`true`

EZ_FANCY_LISTBOX_TYPE	fancyListBoxType	`0`
EZ_FANCY_LISTBOX_COLUMNS	fancyListBoxColumns	`0`
EZ_HIGHLIGHT_MODE	highlightMode	`0`
EZ_ROW_BG	rowBg	`off nil nil`

EZ_RUBBER_BAND	rubberBand	`false`

4.20 WorkAreas

A workArea is a free format lister for display items. It is like a canvas but with limited display capabilities. Each workArea widget has a private geometry manager, which arranges the locations of display items.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_WORK_AREA`
Default instance name:	`"workArea"`
Default Class name:	`"WorkArea"`
Symbolic Widget type:	`EZ_WIDGET_IWORK_AREA`
Default instance name:	`"iWorkArea"`
Default Class name:	`"IWorkArea"`

Subwidgets Info

Subwidgets of WorkArea are listed below.

```
           ( iWorkArea
WorkArea-->( horizontalScrollbar
           ( verticalScrollbar
```

Widget Interface Routines

```
void  EZ_GetWorkAreaWidgetComponents(EZ_Widget *widget, EZ_Widget **iwa,
                         EZ_Widget **hscroll, EZ_Widget **vscroll);
```

return the components of a workArea widget.

```
void EZ_WorkAreaInsertItems(EZ_Widget *widget, EZ_Item **item, int nitems);
```

insert some items in a workArea widget.

```
void  EZ_WorkAreaInsertItem(EZ_Widget *widget, EZ_Item *item);
```

insert one item in a workArea widget.

```
void EZ_WorkAreaDeleteItem(EZ_Widget *widget, EZ_Item *item);
```

delete the specified item from a workArea widget and update the display.

```
void EZ_WorkAreaDeleteItem(EZ_Widget *widget, EZ_Item *item,
                                             int redisplay);
```

delete the specified item. Update the display if requested.

```
void  EZ_WorkAreaClear(EZ_Widget *widget);
```

clear a workArea widget.

```
void EZ_WorkAreaRearrange(EZ_Widget *widget);
```

redisplay a workArea widget.

```
void EZ_WorkAreaSelectItem(EZ_Widget *widget, EZ_Item *item,
                              int *location);
```

select the specified item and display it at the specified location.

```
void EZ_WorkAreaSelectItemUsingIdx(EZ_Widget *widget, int idx,
                                                int *location);
```

select the specified item and display it at the specified location.

```
EZ_Item  *EZ_WorkAreaFindItemAtXY(EZ_Widget *wa, int x, int y);
```

find and return the item displayed at the specified location.

```
void EZ_GetWorkAreaXYOffset(EZ_Widget *widget, int *x, int *y);
```

return the XY display offset of a workArea widget.

```
void EZ_SetWorkAreaXYOffset(EZ_Widget *widget, int x, int y);
```

set the XY display offset of a workArea widget.

```
int EZ_GetWorkAreaNumberOfItems(EZ_Widget *widget);
```

return the number of items in a workArea widget.

```
EZ_Item *EZ_GetWorkAreaSelection(EZ_Widget *widget);
```

return the selection of a workArea widget.

```
EZ_Item  *EZ_GetWorkAreaItemUsingIdx(EZ_Widget *wa, int idx);
```

return the item in a workArea widget given index.

```
EZ_Item **EZ_GetWorkAreaData(EZ_Widget *wa);
```

return the array of items displayed in a workArea widget.

```
int EZ_WorkAreaGetHighlightedItems(EZ_Widget *widget,
                                   EZ_Item ** a, int siz);
```

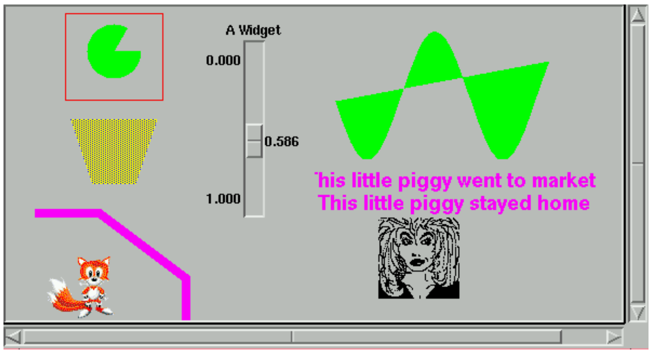
return the highlighted items in a workArea widget.

```
void  EZ_SetWorkAreaGeometryManager(EZ_Widget *widget,
                    EZ_WorkAreaGeometryManager managet, void *pdata);
```

set the geometry manager for a workArea widget.

```
void EZ_WorkAreaInsertItemBefore(EZ_Widget *widget, EZ_Item *item, EZ_Item *newItem);
void EZ_WorkAreaInsertItemAfter (EZ_Widget *widget, EZ_Item *item, EZ_Item *newItem);
void EZ_WorkAreaInsertItemUsingIdx(EZ_Widget *widget, EZ_Item *item, int idx);
void EZ_WorkAreaDeleteItemUsingIdx(EZ_Widget *widget, int idx);
void EZ_WorkAreaDeleteItemUsingIdx0(EZ_Widget *widget, int idx, int dp);
void EZ_WorkAreaDeleteSomeItemsUsingIndices (EZ_Widget *widget, int *ids, int cnt);
```

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	"WorkArea"
		or "IWorkArea"
EZ_NAME	name	"workArea"
		or "iWorkArea"

EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	

EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	

EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`0`

EZ_PADX	padx	`0`
EZ_PADY	pady	`0`
EZ_IPADX	iPadx	`0`
EZ_IPADY	iPady	`0`

EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_CALLBACK	N/A	`null null`
EZ_MOTION_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`

EZ_BG_IMAGE_FILE_B	bgImageFileB	`null`
EZ_BG_PIXMAP_B	N/A	`none`
EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_SELECTION_BACKGROUND	selectionBackground	
EZ_SELECTION_FOREGROUND	selectionForeground	
EZ_TEXT_BACKGROUND	textBackground	

EZ_OPTIONAL_HSCROLLBAR	optionalHScrollbar	`true`
EZ_OPTIONAL_VSCROLLBAR	optionalVScrollbar	`true`

EZ_RUBBER_BAND	rubberBand	`false`
EZ_MOVABLE	movable	`false`

4.21 LCD

An LCD widget simulates a liquid crystal display. It is capable of displaying a fixed width numerical value.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_LCD`
Default instance name:	`"lcd"`
Default Class name:	`"Lcd"`

Widget Interface Routines

`void EZ_SetLcdString(EZ_Widget *parent, char *str);`
set the numeric string to be displayed.

`void EZ_SetLcdInteger(EZ_Widget *lcd, int n);`
set the integer to be displayed.

`void EZ_SetLcdFloat(EZ_Widget *cd, float f);`
set a floating point number to be displayed.

`void EZ_ClearLcd (EZ_Widget *lcd);`
clear a Lcd widget.

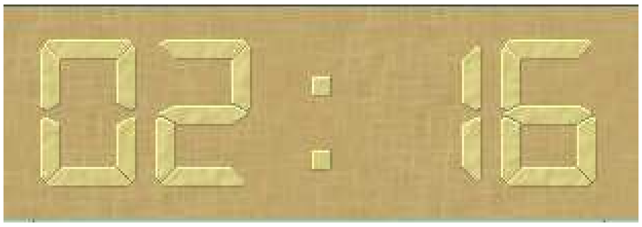
`void EZ_SetLcdNDigits (EZ_Widget *lcd, int n);`
set the width (in terms of digits) of a lcd widget.

`void EZ_SetLcdFontSize (EZ_Widget *lcd, int n);`
set the font size, in terms of pixels.

`void EZ_SetLcdForeground (EZ_Widget *lcd, char *clr);`
set the foreground color.

`void EZ_SetLcdBackground (EZ_Widget *lcd, char *clr);`
set the background color.

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"Lcd"`
EZ_NAME	name	`"lcd"`
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	

EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	

EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`0`

EZ_PADX	padx	`8`
EZ_PADY	pady	`6`
EZ_IPADX	iPadx	`0`
EZ_IPADY	iPady	`0`

EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`

EZ_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`

EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`

EZ_BG_IMAGE_FILE_B	bgImageFileB	`null`
EZ_BG_PIXMAP_B	N/A	`none`
EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`

EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`

EZ_LCD_NDIGITS	lcdNDigits	`6`
EZ_LCD_FONT_SIZE	lcdFontSize	`8`
EZ_LCD_BACKGROUND	lcdBackground	`"gray74"`
EZ_LCD_FOREGROUND	lcdForeground	`"gray74"`

4.22 LED

An LED widget simulates a light emitting diodes display. It implements a few scrolling functions like those on a real LED display.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_LED`
Default instance name:	`"led"`
Default Class name:	`"Led"`

Widget Interface Routines

`void EZ_SetLedString(EZ_Widget *parent, char *str, char *clr);`

set the text to be displayed.

`char *EZ_GetLedString(EZ_Widget *led);`

return the text being displayed.

`void EZ_ClearLed (EZ_Widget *led);`

clear an led widget.

`void EZ_SetLedForeground (EZ_Widget *led, char *clr);`

set the foreground color.

`void EZ_SetLedBackground (EZ_Widget *led, char *clr);`

set the background color.

`void EZ_SetLedOffPixelColor (EZ_Widget *lcd, char *clr);`

set the color to display off pixels.

`void EZ_SetLedFont (EZ_Widget *lcd, char *font);`

set the font.

`void EZ_SetLedWidth (EZ_Widget *lcd, int w);`

set width, in terms of led pixels.

`void EZ_SetLedHeight (EZ_Widget *lcd, int h);`

set height, in terms of led pixels.

`void EZ_SetLedSize(EZ_Widget *lcd, int w, int h);`

set the width and height, in led pixels.

`void EZ_SetLedPixelSize (EZ_Widget *lcd, int s);`

set led pixel size.

`void EZ_RemoveLedAction(EZ_Widget *led, int action);`

remove a queued led scroll function.

`int EZ_LedIsFree (EZ_Widget *lcd);`

return the busy status of a led widget.

`void EZ_SetLedColorFunction(EZ_Widget *led, unsigned long (*colorf)(EZ_Widget *, int, int));`

set a pixel coloring function.

```
int EZ_QueueLedAction(EZ_Widget *led, int type, int delay, int count,
                       char *msg,  char *clr);
```

Quene an led scroll function.

Led Actions

Here is the list of all led actions.

```
#define EZ_LED_SCROLL_LEFT        1
#define EZ_LED_SCROLL_RIGHT       2
#define EZ_LED_SCROLL_UP          3
#define EZ_LED_SCROLL_DOWN        4
#define EZ_LED_SCROLL_CENTE_H     5
#define EZ_LED_SCROLL_CENTER_V    6
#define EZ_LED_SWEEP_LEFT         7
#define EZ_LED_SWEEP_RIGHT        8
#define EZ_LED_SWEEP_UP           9
#define EZ_LED_SWEEP_DOWN         10
#define EZ_LED_SWEEP_CENTER       11
#define EZ_LED_SWEEP_CENTER_V     12
#define EZ_LED_BOUNCE             13
#define EZ_LED_BOUNCE_V           14
#define EZ_LED_SHOW               15
#define EZ_LED_BLINK              16
#define EZ_LED_BLINK_SCROLL_LEFT  17
#define EZ_LED_BLINK_SCROLL_RIGHT 18
#define EZ_LED_SLEEP              19
```

Screenshot

Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"Led"`
EZ_NAME	name	`"led"`
EZ_X	x	
EZ_Y	y	
EZ_WIDTH	width	
EZ_HEIGHT	height	
EZ_LOCATION	location	
EZ_SIZE	size	
EZ_GEOMETRY	geometry	
EZ_WIDTH_HINT	widthHint	
EZ_HEIGHT_HINT	heightHint	
EZ_SIZE_HINT	sizeHint	
EZ_BORDER_WIDTH	borderWidth	`0`
EZ_BORDER_TYPE	borderType	`none`
EZ_FOCUS_PAD	focusPad	`0`
EZ_CURSOR	cursor	`"XC_left_ptr"`
EZ_BUBBLE_STRING	bubbleString	`null`
EZ_CALLBACK	N/A	`null null`
EZ_DESTROY_CALLBACK	N/A	`null null`
EZ_EVENT_HANDLER	N/A	`null null`
EZ_FOREGROUND	foreground	`"black"`
EZ_BACKGROUND	background	`"gray74`
EZ_BG_IMAGE_FILE	bgImageFile	`null`
EZ_BG_PIXMAP	N/A	`none`
EZ_CLIENT_PTR_DATA	N/A	`null`
EZ_CLIENT_INT_DATA	clientIntData	`0`
EZ_DND_DRAG_CURSOR	dndDragCursor	`none`
EZ_DND_BUBBLE_STRING	dndBubbleString	`null`
EZ_LED_BACKGROUND	ledBackground	`"black"`
EZ_LED_PIXEL_COLOR	ledPixelColor	
EZ_LED_WIDTH	ledWidth	`60`
EZ_LED_HEIGHT	ledHeight	`10`
EZ_LED_PIXEL_SIZE	ledPixelSize	`2,2`
EZ_LED_COLOR_FUNCTION	ledColorFunction	`null`

4.23 Status Meters

A status meter is like a progress bar. It has either a vertical or a horizontal orientation. A status meter is capable of displaying five distinct quantities simultaneously.

Widget Info

Symbolic Widget type:	`EZ_WIDGET_STATUS_METER`
Default instance name:	`"statusMeter"`
Default Class name:	`"StatusMeter"`

Widget Interface Routines

`void  EZ_SetStatusMeterRange(EZ_Widget *widget, float f, float t);`

set the range of a status meter.

`void  EZ_SetStatusMeterValue(EZ_Widget *widget, float value);`

set the value of a status meter.

`float EZ_GetStatusMeterValue(EZ_Widget *widget);`

return the status value.

`void  EZ_StopStatusMeter(EZ_Widget *widget);`

stop a status meter.

`void  EZ_StartStatusMeter(EZ_Widget *widget);`

resume a status meter.

`void  EZ_SetStatusMeterForeground(EZ_Widget *widget, char *clr);`

set the foreground color.

`void  EZ_SetStatusMeterBackground(EZ_Widget *widget, char *clr);`

set the background color.

`void  EZ_SetStatusMeterRangeN(EZ_Widget *widget, int idx, float f, float t);`

set the rang of the n*th* status variable.

`void  EZ_SetStatusMeterForegroundN(EZ_Widget *widget, int idx, char *clr);`

set the foreground color to display the n*th* status variable.

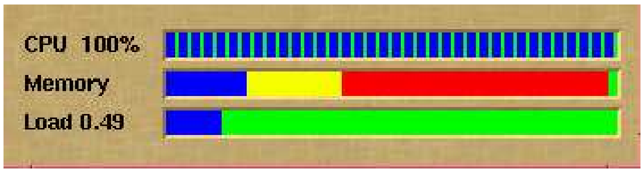
`void  EZ_SetStatusMeterValueN(EZ_Widget *widget, int idx, float value);`

set the value of th n*th* status variable.

`float EZ_GetStatusMeterValueN(EZ_Widget *widget, int idx);`

return the value of the n*th* status variable

Screenshot



Resources

Config_Option	*ResourceName*	*DefaultValue*
EZ_CLASS	class	`"StatusMeter"`
EZ_NAME	name	`"statusmeter"`