ESP Widget Set for FLTK

ESPWS−2.0

Easy Software Products

Table of Contents

Introduction

The ESP Widget Set for FLTK provides check button, file chooser, help viewer, and wizard widgets for FLTK. These widgets are provided under the terms of the GNU General Public License.

The widget set comes with a configure script for UNIX systems and Visual C++ workspace and project files for Windows systems.

Compiling the Widget Set under UNIX

First you must configure the makefile for use on your system; in most cases you just need to do:

```
% ./configure ENTER
```

If the configure script is unable to determine what C and C++ compilers to use, set the `CC` and `CXX` environment variables to the corresponding program.

The `CFLAGS`, `CXXFLAGS`, and `LDFLAGS` environment variables can also be set to tell the compilers and linker where to find include files and libraries.

Once you have successfully configured the makefile, just type:

```
% make ENTER
```

to build the widget set and all the examples.

Compiling the Widget Set under Windows

The Visual C++ project file was created using Visual C++ 6.0. We do not recommend using older versions of Visual C++.

To build the widget set and all the examples, open the "espws" workspace and do a batch build.

The File Chooser

The `FileChooser` widget provides a standard file selection dialog for your applications. It supports three different selection modes − single file selection, multiple file selection, and new file selection (for use in "save as" situations).

Implementation

The `FileChooser` widget is based upon four classes:

- `FileBrowser` − this is a subclass of `Fl_Browser` that lists files in a specified directory or all drives/mount points on a system.
- `FileChooser` − this is the high−level class that provides the dialog window and controls.
- `FileIcon` − this class provides icon images for the `FileBrowser` widget and associates filetypes/extensions with those icon images. The current code understands SGI ".fti" files used under IRIX and ".xpm" files used by CDE.
- `FileInput` − this is a subclass of `Fl_Input` that remaps the Tab key. The new mapping allows a user to move the cursor to the end of the current selection using the Tab key (i.e. to accept filename completion), but if there is no selection the Tab key performs navigation instead.

The `FileIcon` class allows you to add icons for individual file types. Normally you'll just use the `FileIcon::load_system_icons()` method to load icons specific to your system; if the system icons can't be loaded then generic file and folder icons are used instead. Icons are only shown if you have loaded them.

Using the FileChooser Widget

To use the `FileChooser` widget in your program, do:

```
#include "FileChooser.h"

...

{
  FileIcon::load_system_icons(); // Optional...

  FileChooser fc("pathname", "*.pattern", type, "title");

  fc.show();
  while (fc.visible())
    Fl::wait();

  fc.count()  = number of selected files
  fc.value()  = value of first (or only) selected file
  fc.value(n) = value of selection "n"
}
```

The Help Dialog

The `HelpDialog` widget displays HTML files and allows the user to click on links to do navigation. Currently most HTML 2.0 elements are supported except for images. Table support is primitive at best.

Even so, it provides enough capabilities to be used for on−line help and other HTML applications.

Implementation

The `HelpDialog` widget combines a dialog window, history buttons, text size buttons, and the `HelpView` widget to provide an embedded HTML file viewing dialog.

Using the HelpDialog Widget

To use the `HelpDialog` widget in your program, do:

```
#include "HelpDialog.h"

...

{
  HelpDialog help();

  help.load("filename.html");
  help.show();
  while (help.visible())
    Fl::wait();
}
```

The Help Application (flsurf)

The `HelpApp` widget provides a simple web browser application, complete with bookmarks, proxy support, and so forth. The `flsurf` application included with the distribution demonstrates how to use it.

Implementation

The `HelpApp` widget is composed of several windows and control widgets. The HTML viewing portion is handled by the `HelpView` widget, while file selection is handled by the `FileChooser` widget.

HTTP access to files is handled by the CUPS HTTP functions. Currently only a single file is cached, so any navigation results in a reload.

Using the HelpApp Widget

The source for `flsurf` is shown below:

```
#include "HelpApp.h"
#include <FL/x.H>

int                             // O - Exit status
main(int  argc,                 // I - Number of command-line arguments
     char *argv[])              // I - Command-line arguments
{
  HelpApp       *app;           // Help application


  fl_open_display();

  app = new HelpApp;

  if (argc >= 2)
    app->load(argv[1]);

  app->show();

  Fl::run();

  delete app;

  return (0);
}
```

Widgets

The widget set includes several base and composite widgets. The base widgets include:

- `CheckButton`, a better looking check/radio button.
- `FileBrowser`, a browser widget that supports file icons.
- `FileIcon`, a class for managing and drawing file icons.
- `Fl_Wizard`, an adaptation of the `Fl_Tabs` widget for "wizard" interfaces.
- `HelpView`, a simple HTML viewing widget.

Technically, the `FileIcon` class is not a FLTK widget, however it does provide a `label()` method for widgets, much like `Fl_Pixmap`.

The composite widgets include:

- `FileChooser`, a complete file chooser dialog using the `FileBrowser`, `FileIcon`, and `FileInput` widgets.
- `HelpDialog`, a complete help dialog using the `HelpView` widget.
- `HelpApp`, a simple web browser using the `FileChooser` and `HelpView` widgets.

The `HelpDialog` widget is designed for embedded use within a FLTK application, while the `HelpApp` widget provides the framework for a complete application, such as the flsurf application included with the widget set.

class CheckButton

Class Hierarchy

```
Fl_Group
   |
   +----CheckButton
```

Include Files

```
#include "CheckButton.h"
```

Description

The `CheckButton` widget provides a better looking check or radio button. It is otherwise identical to the `Fl_Check_Button` widget.

Methods

CheckButton(int xx, int yy, int ww, int hh, const char *l = 0)

The constructor creates the `CheckButton` widget at the specified position and size.

~CheckButton()

The destructor destroys the widget and frees all memory that has been allocated.

class FileBrowser

Class Hierarchy

```
Fl_Browser
   |
   +----FileBrowser
```

Include Files

```
#include "FileBrowser.h"
```

Description

The `FileBrowser` widget displays a list of filenames, optionally with file−specific icons.

Methods

FileBrowser(int xx, int yy, int ww, int hh, const char *l = 0)

The constructor creates the `FileBrowser` widget at the specified position and size.

~FileBrowser()

The destructor destroys the widget and frees all memory that has been allocated.

void iconsize(uchar s)
uchar iconsize() const

Sets or gets the size of the icons. The default size is 20 pixels.

void filter(const char *pattern)
const char *filter() const

Sets or gets the filename filter. The pattern matching uses the `filename_match()` function in FLTK.

int load(const char *directory)

Loads the specified directory into the browser. If icons have been loaded then the correct icon is associated with each file in the list.

class FileChooser

Class Hierarchy

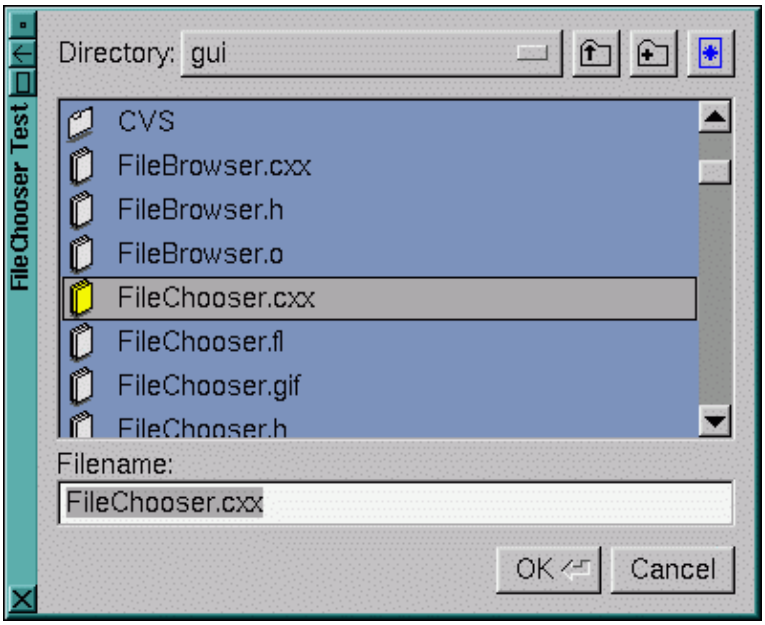
```
Fl_Group
   |
   +----FileChooser
```

Include Files

```
#include "FileChooser.h"
```

Description

The `FileChooser` widget displays a standard file selection dialog that supports various selection modes.



Methods

- textcolor
- textfont
- textsize
- type
- value
- visible

FileChooser(const char *pathname, const char *pattern, int type, const char *title)

The constructor creates the `FileChooser` dialog pictured above. The `pathname` argument can be a directory name or a complete file name (in which case the corresponding file is highlighted in the list and in the filename input field.)

The `pattern` argument can be a `NULL` string or `"*"` to list all files. See the FLTK documentation on `filename_match()` for other kinds of patterns.

The `type` argument can be one of the following:

- `SINGLE` − allows the user to select a single, existing file.
- `MULTI` − allows the user to select one or more existing files.
- `CREATE` − allows the user to select a single, existing file or specify a new filename.

The `title` argument is used to set the title bar text for the `FileChooser` window.

~FileChooser()

Destroys the widget and frees all memory used by it.

void color(Fl_Color c)
Fl_Color color()

Sets or gets the background color of the `FileBrowser` list.

int count()

Returns the number of selected files.

void directory(const char *pathname)
const char *directory()

Sets or gets the current directory.

void filter(const char *pattern)
const char *filter()

Sets or gets the current filename filter pattern.

void hide()

Hides the `FileChooser` window.

void iconsize(uchar s)
uchar iconsize()

Sets or gets the size of the icons in the `FileBrowser`. By default the icon size is set to 1.5 times the `textsize()`.

void label(const char *l)
const char *label()

Sets or gets the title bar text for the `FileChooser`.

void rescan()

Reloads the current directory in the `FileBrowser`.

void show()

Shows the `FileChooser` window.

void textcolor(Fl_Color c)
Fl_Color textcolor()

Sets or gets the current `FileBrowser` text color.

void textfont(uchar f)
uchar textfont()

Sets or gets the current `FileBrowser` text font.

void textsize(uchar s)
uchar textsize()

Sets or gets the current `FileBrowser` text size.

void type(int t)
int type()

Sets or gets the current type of `FileChooser`.

const char *value(const char *pathname)
const char *value(int file)
const char *value()

Sets or gets the current value of the selected file.

int visible()

Returns 1 if the `FileChooser` window is visible.

class FileIcon

Class Hierarchy

```
FileIcon
```

Include Files

```
#include "FileIcon.h"
```

Description

The `FileIcon` class manages icon images that can be used as labels in other widgets and as icons in the `FileBrowser` widget.

Methods

FileIcon()

The constructor creates a new `FileIcon` with the specified information.

~FileIcon()

The destructor destroys the icon and frees all memory that has been allocated for it.

short *add(short d)

Adds a keyword value to the icon array, returning a pointer to it.

short *add_color(short c)

Adds a color value to the icon array, returning a pointer to it.

short *add_vertex(int x, int y)
short *add_vertex(float x, float y)

Adds a vertex value to the icon array, returning a pointer to it. The integer version accepts coordinates from 0 to 10000, while the floating point version goes from 0.0 to 1.0. The origin (0.0) is in the lower−lefthand corner of the icon.

void clear()

Clears all icon data from the icon.

void draw(int x, int y, int w, int h, Fl_Color ic, int active = 1)

Draws the icon in the indicated area.

static FileIcon *find(const char *filename, int filetype = ANY);

Finds an icon that matches the given filename and file type.

static FileIcon *first()

Returns a pointer to the first icon in the list.

void label(Fl_Widget *w)

Applies the icon to the widget, registering the `FileIcon` label type as needed.

static void labeltype(const Fl_Label *o, int x, int y, int w, int h, Fl_Align a)

The labeltype function for icons.

void load(const char *f)

Loads the specified icon image. The format is deduced from the filename.

void load_fti(const char *fti)

Loads an SGI icon file.

static void load_system_icons(void)

Loads all system−defined icons. This call is useful when using the `FileChooser` widget and should be used when the application starts:

```
FileIcon::load_system_icons();
```

void load_xpm(const char *xpm)

Loads an XPM icon file.

const char *pattern()

Returns the filename matching pattern for the icon.

int size()

Returns the number of words of data used by the icon.

int type()

Returns the filetype associated with the icon, which can be one of the following:

- `FileIcon::ANY`, any kind of file.
- `FileIcon::PLAIN`, plain files.
- `FileIcon::FIFO`, named pipes.
- `FileIcon::DEVICE`, character and block devices.
- `FileIcon::LINK`, symbolic links.
- `FileIcon::DIRECTORY`, directories.

short *value()

Returns the data array for the icon.

class FileInput

Class Hierarchy

```
Fl_Input
   |
   +----FileInput
```

Include Files

```
#include "FileInput.h"
```

Description

The `FileInput` widget has got to be a candidate for the shortest widget ever. The only difference between it and the `Fl_Input` widget is that when text is selected in the input field, the **`Tab`** key will move the cursor to the end of the selection and clear the selection, instead of moving to the next input field.

Methods

FileInput(int xx, int yy, int ww, int hh, const char *l = 0)

The constructor creates the `FileInput` widget at the specified position and size.

~FileInput()

The destructor destroys the widget and frees all memory that has been allocated.

class Fl_Wizard

Class Hierarchy

```
Fl_Group
   |
   +----Fl_Wizard
```

Include Files

```
#include "Fl_Wizard.h"
```

Description

The `Fl_Wizard` widget is based off the `Fl_Tabs` widget, but instead of displaying tabs it only changes "tabs" under program control. Its primary purpose is to support "wizards" that step a user through configuration or troubleshooting tasks.

As with `Fl_Tabs`, wizard panes are composed of child (usually `Fl_Group`) widgets. Navigation buttons must be added separately.

Methods

Fl_Wizard(int xx, int yy, int ww, int hh, const char *l = 0)

The constructor creates the `Fl_Wizard` widget at the specified position and size.

~Fl_Wizard()

The destructor destroys the widget and its children.

void next()

This method shows the next child of the wizard. If the last child is already visible, this function does nothing.

void prev()

This method shows the previous child of the wizard. If the first child is already visible, this function does nothing.

void value(Fl_Widget *w)
Fl_Widget *value()

Sets or gets the child widget that is visible.

class HelpApp

Class Hierarchy

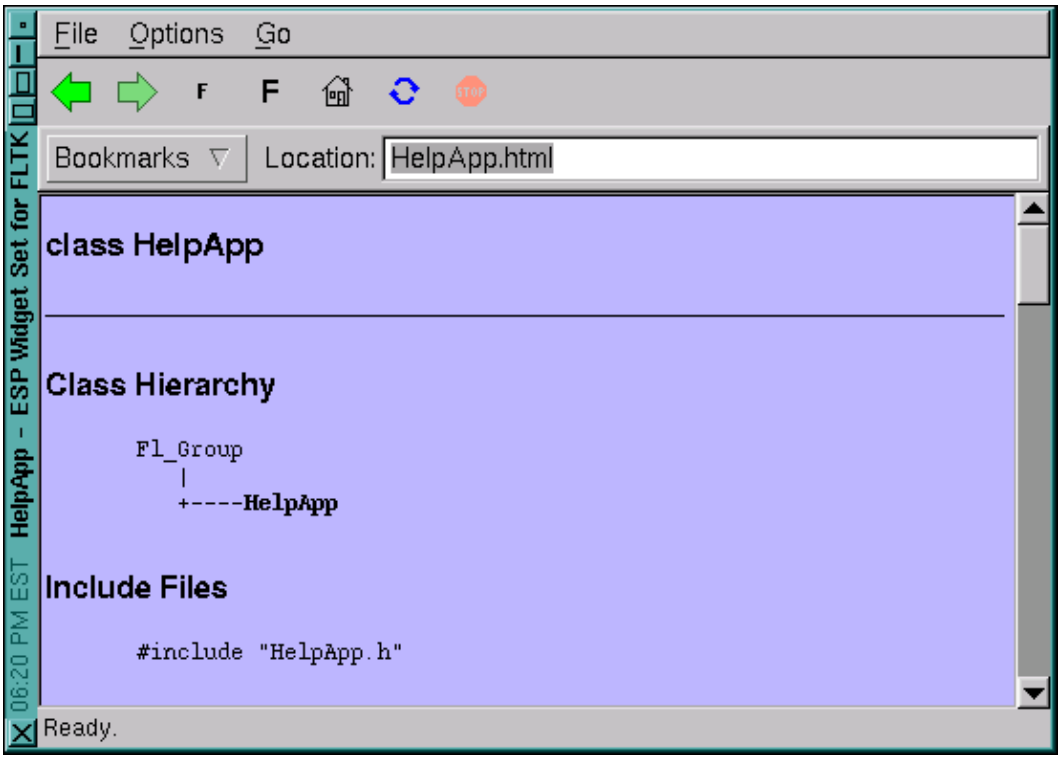
```
Fl_Group
   |
   +----HelpApp
```

Include Files

```
#include "HelpApp.h"
```

Description

The `HelpApp` widget displays a simple web-browser window using the `HelpView` widget.



Methods

HelpApp()

The constructor creates the dialog pictured above.

~HelpView()

The destructor destroys the widget and frees all memory that has been allocated for the current file.

int h()

Return the height of the `HelpApp` window.

void hide()

Hides the `HelpApp` window.

void load(const char *f)

Loads the specified HTML file into the `HelpView` widget. The filename can also contain a target name ("filename.html#target").

void position(int xx, int yy)

Repositions the window.

void resize(int xx, int yy, int ww, int hh)

Resizes and positions the window.

void show()

Shows the `HelpApp` window.

void textsize(uchar s)
uchar textsize()

Sets or gets the default text size.

void topline(const char *n)
void topline(int n)

Sets the top line in the `HelpView` widget to the named or numbered line.

int visible()

Returns 1 if the `HelpApp` window is visible.

int w()

Return the width of the `HelpApp` window.

int x()

Return the horizontal position of the `HelpApp` window.

int y()

Return the vertical position of the `HelpApp` window.

class HelpDialog

Class Hierarchy

```
Fl_Group
   |
   +----HelpDialog
```

Include Files

```
#include "HelpDialog.h"
```

Description

The `HelpDialog` widget displays a standard help dialog window using the `HelpView` widget.

Methods

- HelpDialog
- ~HelpDialog
- hide
- load
- show

- topline
- visible

HelpDialog()

The constructor creates the dialog pictured above.

~HelpView()

The destructor destroys the widget and frees all memory that has been allocated for the current file.

void hide()

Hides the `HelpDialog` window.

void load(const char *f)

Loads the specified HTML file into the `HelpView` widget. The filename can also contain a target name ("filename.html#target").

void show()

Shows the `HelpDialog` window.

void topline(const char *n)
void topline(int n)

Sets the top line in the `HelpView` widget to the named or numbered line.

int visible()

Returns 1 if the `HelpDialog` window is visible.

class HelpView

Class Hierarchy

```
Fl_Group
   |
   +----HelpView
```

Include Files

```
#include "HelpView.h"
```

Description

The `HelpView` widget displays HTML text. Most HTML 2.0 elements are supported, as well as a primitive implementation of tables. Images are not currently displayed (although the ALT text is, if present.)

Methods

HelpView(int xx, int yy, int ww, int hh, const char *l = 0)

The constructor creates the `HelpView` widget at the specified position and size.

~HelpView()

The destructor destroys the widget and frees all memory that has been allocated for the current file.

const char *directory() const

This method returns the current directory (base) path for the file in the buffer.

const char *filename() const

This method returns the current filename for the text in the buffer.

void link(HelpFunc *fn)

This method assigns a callback function to use when a link is followed or a file is loaded (via `HelpView::load()`) that requires a different file or path. The callback function receives the full pathname for the file in question and must return a pathname that can be opened as a local file. This is used by the `HelpApp` widget to support WWW addresses.

int load(const char *f)

This method loads the specified file or URL.

int size() const

This method returns the length of the buffer text in pixels.

void textcolor(Fl_Color c)
Fl_Color textcolor() const

The first form sets the default text color. The second returns the current default text color.

void textfont(uchar f)
uchar textfont() const

The first form sets the default text font. The second returns the current default text font.

void textsize(uchar s)
uchar textsize() const

The first form sets the default text size. The second returns the current default text size.

const char *title()

This method returns the current document title, or NULL if there is no title.

void topline(const char *n)
void topline(int)
int topline() const

The first two forms scroll the text to the indicated position, either with a named destination or by pixel line.

The second form returns the current top line in pixels.

void value(const char *v)
const char *value() const

The first form sets the current buffer to the string provided and reformats the text. The second form returns the current buffer contents.

Release Notes

Changes in v2.0

- **Merged FileChooser and HelpDialog distributions**
- Got rid of several compiler warnings (converting from float to int).
- Open files in binary mode (workaround for MSVC++ C library problems)
- Fixed scrollbar problems.
- Fixed `HR` problems (wasn't adding x() and y() offset.)
- New `h()`, `position()`, `resize()`, `w()`, `x()`, and `y()` methods to control help dialog window.
- New `link()` callback method − allows you to support loading of data from other locations.
- New `textcolor()` methods to control default text color.
- Now support `TABLE` border and width attributes.
- Now support `BODY` color attributes.
- Now support horizontal alignment.
- Now support table background colors.
- New `flsurf` application that supports limited web browsing, etc. (web browsing requires CUPS library)
- F_OK wasn't defined under Windows.
- FileIcon::load() didn't check the filename extension properly − it was missing the ".".
- FileIcon::load_xpm() needed to use a case−insensitive comparison for color names.
- The KDE load_kde_mimelnk() function had the fopen() check backwards.
- Some of the old FTI code still used NULL for the outline index, but GCC 2.95 didn't like that.
- Added support for column_widths() and column_char() methods from the Fl_Browser widget.
- OS/2 fixes from Alexander Mai.

Changes in HelpView v1.0.1

- Now include a makefile and configure script.

Changes in FileChooser v1.4.2

- FileBrowser::load() didn't handle unreadable directories.
- FileIcon::load_fti() could store the outline color in freed memory.
- FileIcon::load_system_icons() now uses default icons that look like Microsoft Windows icons.
- FileIcon::load_system_icons() now supports GNOME and KDE icons.
- FileIcon::draw() now accepts an "active" flag so that inactive controls are drawn with inactive icons.

Changes in FileChooser v1.4.1

- Now include a makefile and configure script.
- Added missing XBM icon files.

Changes in FileChooser v1.4

- Now provided under the LGPL.
- Pressing the Cancel button, hitting escape, or closing the dialog window clears the FileChooser value.
- The FileIcon class now loads icons for CDE and the IRIX Interactive Desktop (auto−detect at run−time.) Support for GNOME, KDE, and Windows is in the works.
- The FileIcon class now provides a `label()` method for using icons as labels for widgets.

Changes in FileChooser v1.3

- Added new `FileInput` widget (world's shortest widget! :) so that filename completion can be done with the Tab, End, or right arrow key.
- The `FileChooser` widget now uses the `FileInput` widget instead of `Fl_Input`.
- The text selection in the filename field now puts the cursor at the end of the selection (unless you type the Backspace key.)

Changes in FileChooser v1.2

- Fixed VC++ compile problems.
- Filename completion now scrolls the file list to the first matching file and selects it when it matches exactly.
- The `value()` method incorrectly reset the chooser type to `FileChooser::SINGLE` when the type was `FileChooser::CREATE`.
- The `value()` method didn't handle directories without filenames properly (always showed drives/file systems).
- The "new directory" button is now only activated when the chooser type is `FileChooser::CREATE`.
- The "OK" button is now disabled until a filename is selected or entered on the keyboard.
- Previously the chooser would only beep at the user when a non− existing filename was entered for `SINGLE` and `MULTI` type choosers. It now also pops up an alert dialog with the text, "Please choose an existing file!"
- Added `iconsize()`, `textcolor()`, `textfont()`, and `textsize()` methods to the `FileChooser` class.
- The chooser no longer loads any icons by default; call the new `load_system_icons()` method in the `FileIcon` class to get the icons.
- The `FileBrowser` widget now has `iconsize()` methods to control the size of the icons separately from the text font.
- The `FileBrowser` widget now supports tabs and newlines in the item string.
- Added `iconsize`, `textcolor`, `textfont`, and `textsize` methods to the `FileChooser` class.

Changes in FileChooser v1.1

- The `FileChooser::value()` method now has a default argument of 1.
- Changed the `multi` methods and arguments to `type` so that the chooser can limit selection to existing files if necessary.
- Added a `FileChooser::value(pathname)` method to set the current selection and/or directory.

- The file completion code now handles directory navigation when you press "/".
- The filename field now gets the full width of the window.