Introducing FLTK

Sanel Zukan
sanelz@gmail.com

Outline

- Introduction – what is FLTK
- History
- Hello world and basic examples
- Event handling
- Extending FLTK
- Layout management
- Not covered

Who am I

- EDE maintainer since 2005
- (was) FLTK developer
- Hacking things here and there
- http://sanelz.blogspot.com

Terms

- FLTK – pronounced ‘fulltick’
- EDE – desktop environment built on FLTK
- edelib – library used by EDE
- eFLTK – forked FLTK version for EDE, now obsolete

What is FLTK?

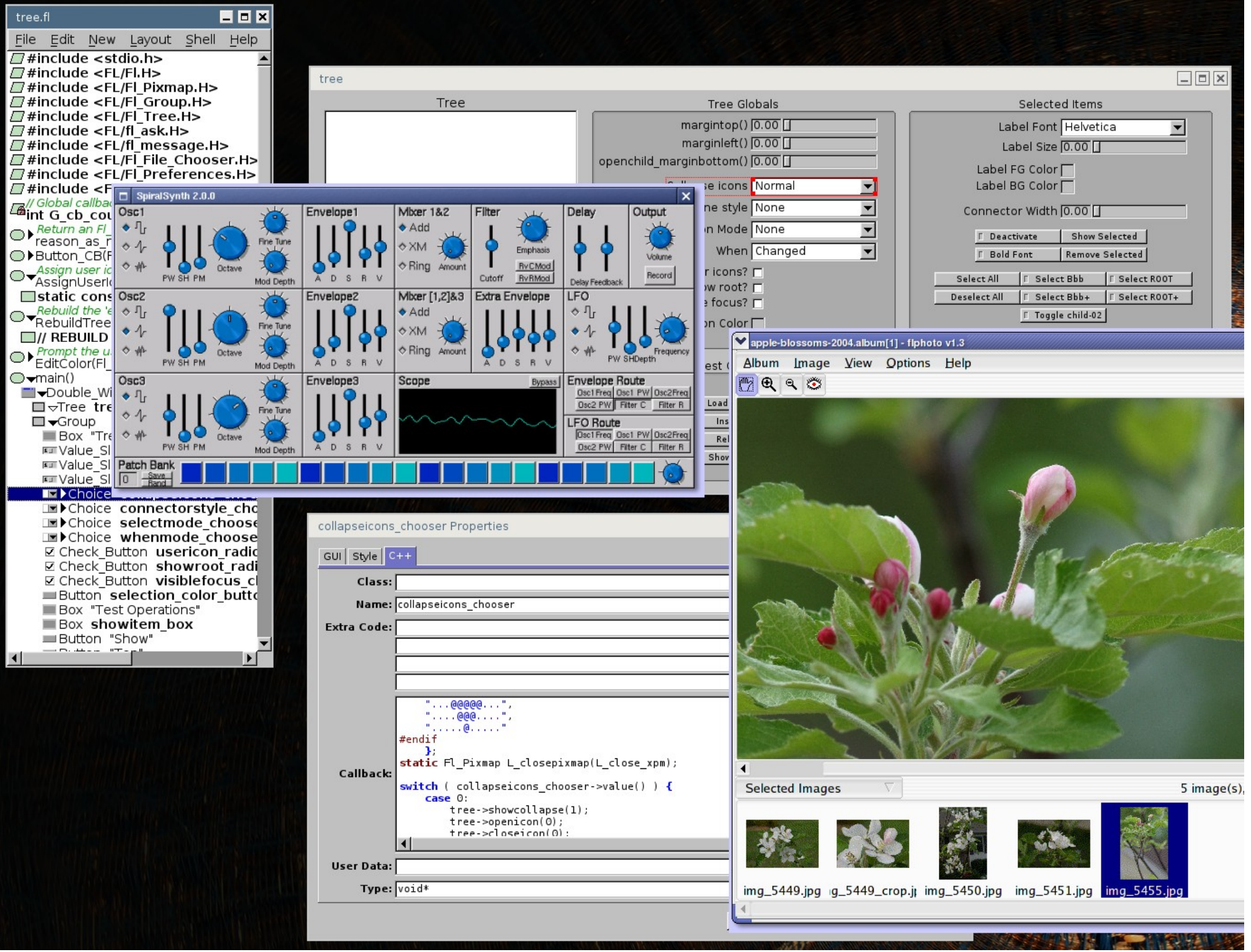
- GUI library only done in C++
- Running on Windows, X server, Nano X, MacOS X, Symbian (S60)
- Very light and small
- OpenGL-like asynchronous API
- Allow static linking
- Permissive license (LGPL with exception)
- Easy to pick up and fun

Quick history

- Started as free implementation of XForms for SGI
- Created by Bill Spitzak
- Maintained by small group of developers
- Today we have 3 branches:
  - Stable 1.3.x
  - Unstable 2.x (obsoleted)
  - Future 3.0

Written using FLTK

- Nuke – the reason why FLTK exists
- EDE desktop
- CinePaint
- Dillo web browser
- Rush render queue (commercial)
- ...

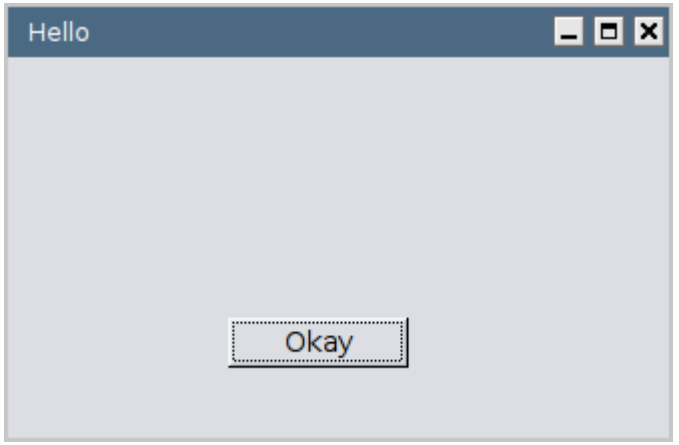

tree.fl
File Edit New Layout Shell Help
#include <stdio.h>
#include <FL/Fl.H>
#include <FL/Fl_Pixmap.H>
#include <FL/Fl_Group.H>
#include <FL/Fl_Tree.H>
#include <FL/fl_ask.H>
#include <FL/fl_message.H>
#include <FL/Fl_File_Chooser.H>
#include <FL/Fl_Preferences.H>
Choice connectorstyle_cho
Choice selectmode_choose
Choice whenmode_choose
Check_Button usericon_radio
Check_Button showroot_radi
Check_Button visiblefocus_cl
Button selection_color_butto
Box "Test Operations"
Box showitem_box
Button "Show"
tree
Tree
Tree Globals
margintop() 0.00
marginleft() 0.00
openchild_marginbottom() 0.00
Normal
None
None
Changed
Selected Items
Label Font Helvetica
Label Size 0.00
Label FG Color
Label BG Color
Connector Width 0.00
Deactivate
Show Selected
Bold Font
Remove Selected
Select All
Select Bbb
Select ROOT
Deselect All
Select Bbb+
Select ROOT+
Toggle child-02
SpiralSynth 2.0.0
Osc1
Envelope1
Mixer 1&2
Filter
Delay
Output
Osc2
Envelope2
Mixer [1,2]&3
Extra Envelope
LFO
Osc3
Envelope3
Scope
Envelope Route
LFO Route
Patch Bank
apple-blossoms-2004.album[1] - flphoto v1.3
Album Image View Options Help
Selected Images
5 image(s)
img_5449.jpg
img_5449_crop.jp
img_5450.jpg
img_5451.jpg
img_5455.jpg
collapseicons_chooser Properties
GUI Style C++
Class:
Name: collapseicons_chooser
Extra Code:
Callback:
User Data:
Type: void*
#endif
static Fl_Pixmap L_closepixmap(L_close_xpm);
switch (collapseicons_chooser->value()) {
case 0:
tree->showcollapse(1);
tree->openicon(0);

Hello world

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>

int main(int argc, char *argv[]) {
   Fl_Window *w = new Fl_Window(330, 190, “Hello”);
   w->show(argc, argv);
   return Fl::run();
}

/* g++ hello.cpp -o hello `fltk-config –cflags –ldflags` */
```


Hello

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Button.H>

int main(int argc, char *argv[]) {
  Fl_Window *w = new Fl_Window(330, 190, “Hello”);
  w->begin();
  new Fl_Button(110, 130, 90, 25, "Okay");
  w->end();
  w->show(argc, argv);
  return Fl::run();
}
```


Hello
Okay

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Button.H>

int main(int argc, char *argv[]) {
  Fl_Window *w = new Fl_Window(330, 190, “Hello”);
  w->begin();
  new Fl_Button(110, 130, 90, 25, "Okay");
  w->end();
  w->show(argc, argv);
  return Fl::run();
}
```

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Button.H>

int main(int argc, char *argv[]) {
  Fl_Window *w = new Fl_Window(330, 190, “Hello”);
  w->begin();
  new Fl_Button(110, 130, 90, 25, "Okay");
  new Fl_Button(110, 155, 90, 25, "Okay #2");
  w->end();
  w->show(argc, argv);
  return Fl::run();
}
```

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Button.H>
#include <FL/Fl_Group.H>

int main(int argc, char *argv[]) {
  Fl_Window *w = new Fl_Window(330, 190, “Hello”);
  w->begin();
  Fl_Group *group = new Fl_Group(110, 130, 90, 155);
  group->begin();
  new Fl_Button(110, 130, 90, 25, "Okay");
  new Fl_Button(110, 155, 90, 25, "Okay #2");
  group->end();
  w->end();
  w->show(argc, argv);
  return Fl::run();
}
```

Event handling

Event handling

- Simple C callbacks
- No signal & slot bloat
- ...but you can use it through external libraries

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Button.H>

int main(int argc, char *argv[]) {
  Fl_Window *w = new Fl_Window(330, 190, “Hello”);
  w->begin();
  new Fl_Button(110, 130, 90, 25, "Okay");
  w->end();
  w->show(argc, argv);
  return Fl::run();
}
```

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Button.H>

static void close_window(Fl_Widget*, void *o) {
   Fl_Window w = (Fl_Window*)o;
   w->hide();
}

int main(int argc, char *argv[]) {
   Fl_Window *w = new Fl_Window(330, 190, “Hello”);
   w->begin();
   Fl_Button *b = new Fl_Button(110, 130, 90, 25, "Close");
   b->callback(close_window, w);
   w->end();
   w->show(argc, argv);
   return Fl::run();
}
```

C++ callbacks?

C++ style callbacks

- Fl_Slot
- Fl_Signal
- Fl_CallbackPlus
- ...

Fl_CallbackPlus demo

http://svn.easysw.com/public/fltk/applications/trunk/Fl_CallbackPlus
(http://bit.ly/XKo1WW)

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Button.H>
#include “Fl_CallbackPlus.h”

class MainWindow : public Fl_Window {
private:
   Fl_Button *button;
   Fl_CallbackList<MainWindow> set_callback;

public:
   MainWindow() : Fl_Window(370, 215, “Sample”) {
      close = new Fl_Button(270, 180, 90, 25, "&Close");
      set_callback(this, &MainWindow::on_close, close);
   }

   void on_close(void) { hide(); }
};

int main(int argc, char *argv[]) {
   MainWindow win;
   win.show();
   return Fl::run();
}
```

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Button.H>
#include “Fl_CallbackPlus.h”

class MainWindow : public Fl_Window {
private:
   Fl_Button *button;
   Fl_CallbackList<MainWindow> set_callback;

public:
   MainWindow() : Fl_Window(370, 215, “Sample”) {
      close = new Fl_Button(270, 180, 90, 25, "&Close");
      set_callback(this, &MainWindow::on_close, close);
     /* set_callback(this, close, &MainWindow::on_close); */
   }

   void on_close(void) { hide(); }
};

int main(int argc, char *argv[]) {
   MainWindow win;
   win.show();
   return Fl::run();
}
```

Extending FLTK

Extending FLTK

- Writing custom widgets
- Handling unknown OS events (e.g. X events)
- Creating custom scheme (aka theme)
- Trying to add new backend

```
#include <FL/Fl_Button.H>

class MyButton : public Fl_Button {
public:
  MyButton(int x, int y, int w, int h, const char *label) :
  Fl_Button(x, y, w, h, label) { }

  int handle(int event) {
    switch(event) {
      case FL_PUSH:
        /* do something on push */
        return 1;
      case FL_ENTER:
        /* do something on enter */
        return 1;
    }

    return Fl_Button::handle(event);
  }
};
```

```
#include <FL/Fl_Button.H>

class MyButton : public Fl_Button {
public:
  MyButton(int x, int y, int w, int h, const char *label) :
  Fl_Button(x, y, w, h, label) { }

  void draw(void) {
    if (image() && (damage() & FL_DAMAGE_ALL))
      image()->draw(x, y);

    fl_font(labelfont(), labelsize());
    fl_color(FL_BLACK);
    fl_draw(label(),  x, y, label_w, label_h, align(), 0, 0);
  }
};
```

```
#include <FL/Fl_Button.H>

class MyButton : public Fl_Button {
public:
  MyButton(int x, int y, int w, int h, const char *label) :
  Fl_Button(x, y, w, h, label) { }

  void draw(void) {
    fl_color(FL_BLACK);
    fl_line_style(FL_DOT);

    fl_push_matrix();
    fl_begin_loop();
      fl_vertex(x, y);
      fl_vertex(x + width, y);
      fl_vertex(x + width, y + height);
      fl_vertex(x, y + height);
      fl_vertex(x, y);
    fl_end_loop();
    fl_pop_matrix();

    fl_line_style(0);
  }
};
```

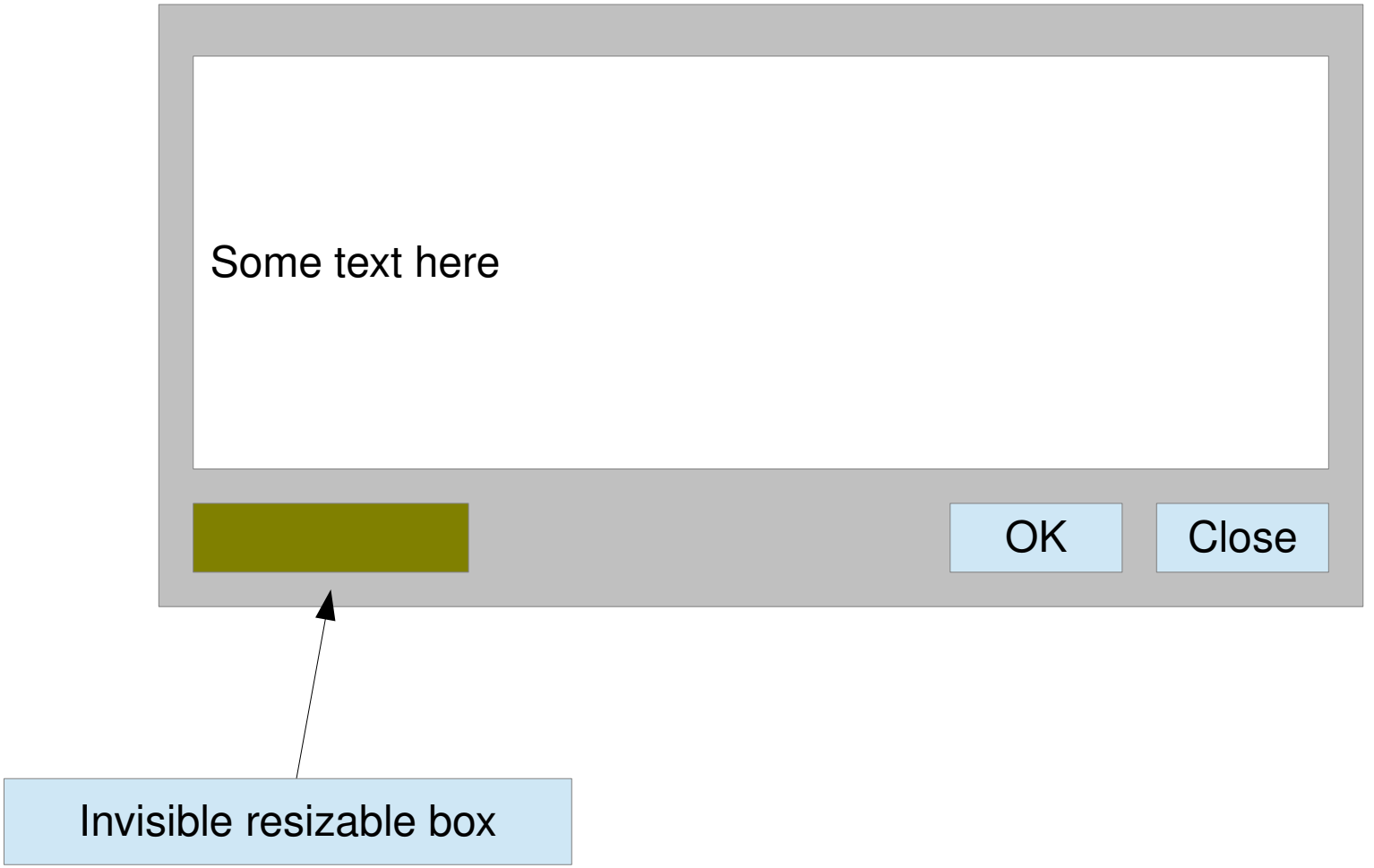
Handling unknown events

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/x.H>

static Atom net_num_of_desktops;

static int unknown_events(int event) {
   if (fl_xevent->xproperty.atom == net_num_of_desktops) {
      /* notify change */
   }
   return 0;
}

int main(int argc, char *argv[]) {
   fl_open_display();

   net_num_of_desktops = XinternAtom(fl_display,
                                     "_NET_NUMBER_OF_DESKTOPS", False);
   Fl_Window *w = new Fl_Window(330, 190, "Hello");
   w->show(argc, argv);
   Fl::add_handler(unknown_events);
   return Fl::run();
}
```

```
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/x.H>

static Atom net_num_of_desktops;

static int unknown_events(int event) {
   if (fl_xevent->xproperty.atom == net_num_of_desktops) {
      /* notify change */
   }
   return 0;
}

int main(int argc, char *argv[]) {
   fl_open_display();

   net_num_of_desktops = XinternAtom(fl_display,
                                     "_NET_NUMBER_OF_DESKTOPS", False);
   Fl_Window *w = new Fl_Window(330, 190, "Hello");
   w->show(argc, argv);
   Fl::add_handler(unknown_events);
   return Fl::run();
}
```

Layout management

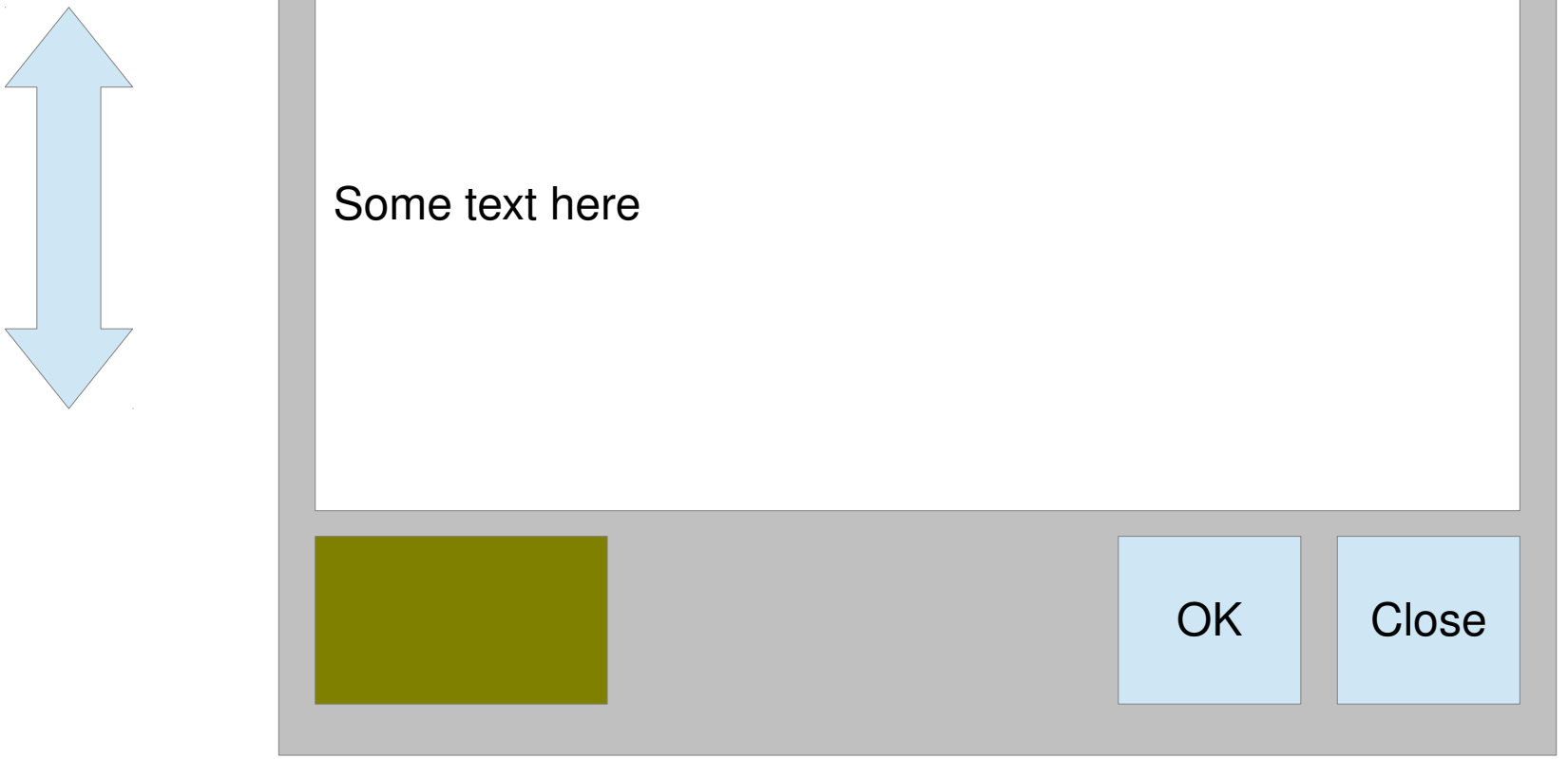
- BorderLayout, BoxLayout, CardLayout, FlowLayout, GridLayout... NO!
- Simple and versatile
- Based on so called “*one resizable child*” -
  only single resizable chiled allowed per group

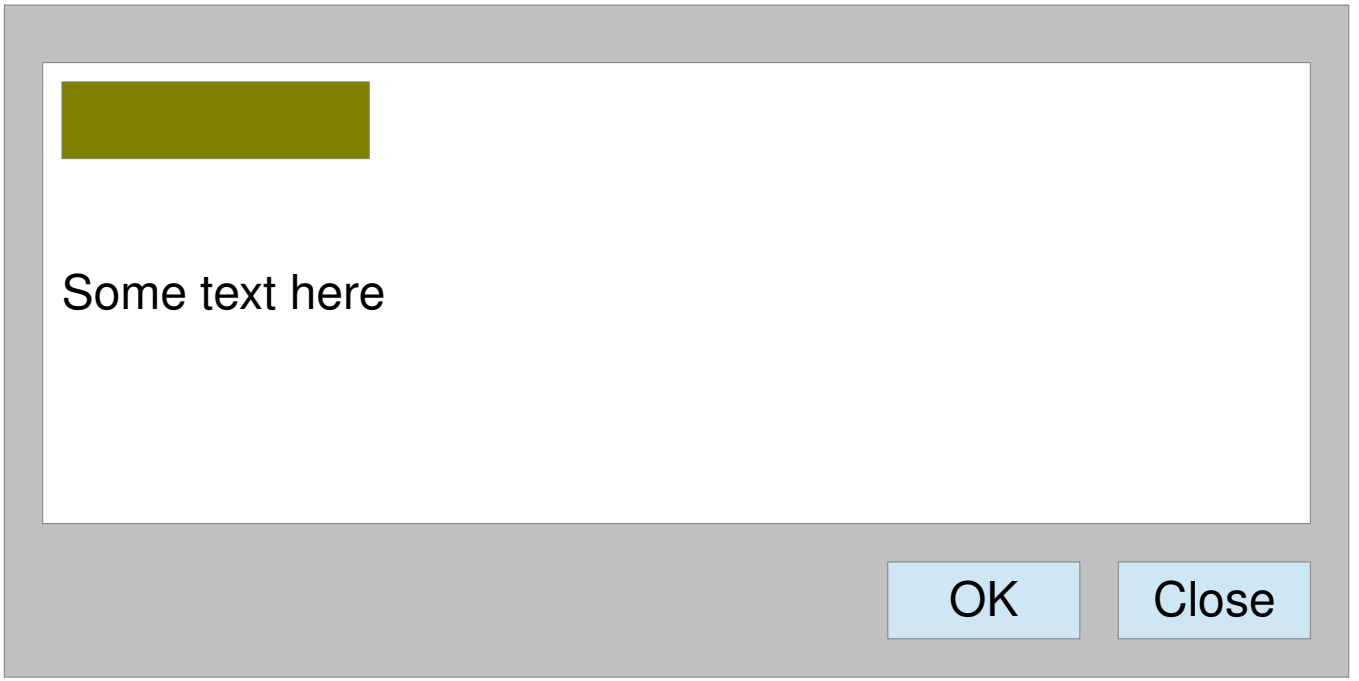

Some text here
OK
Close
Invisible resizable box

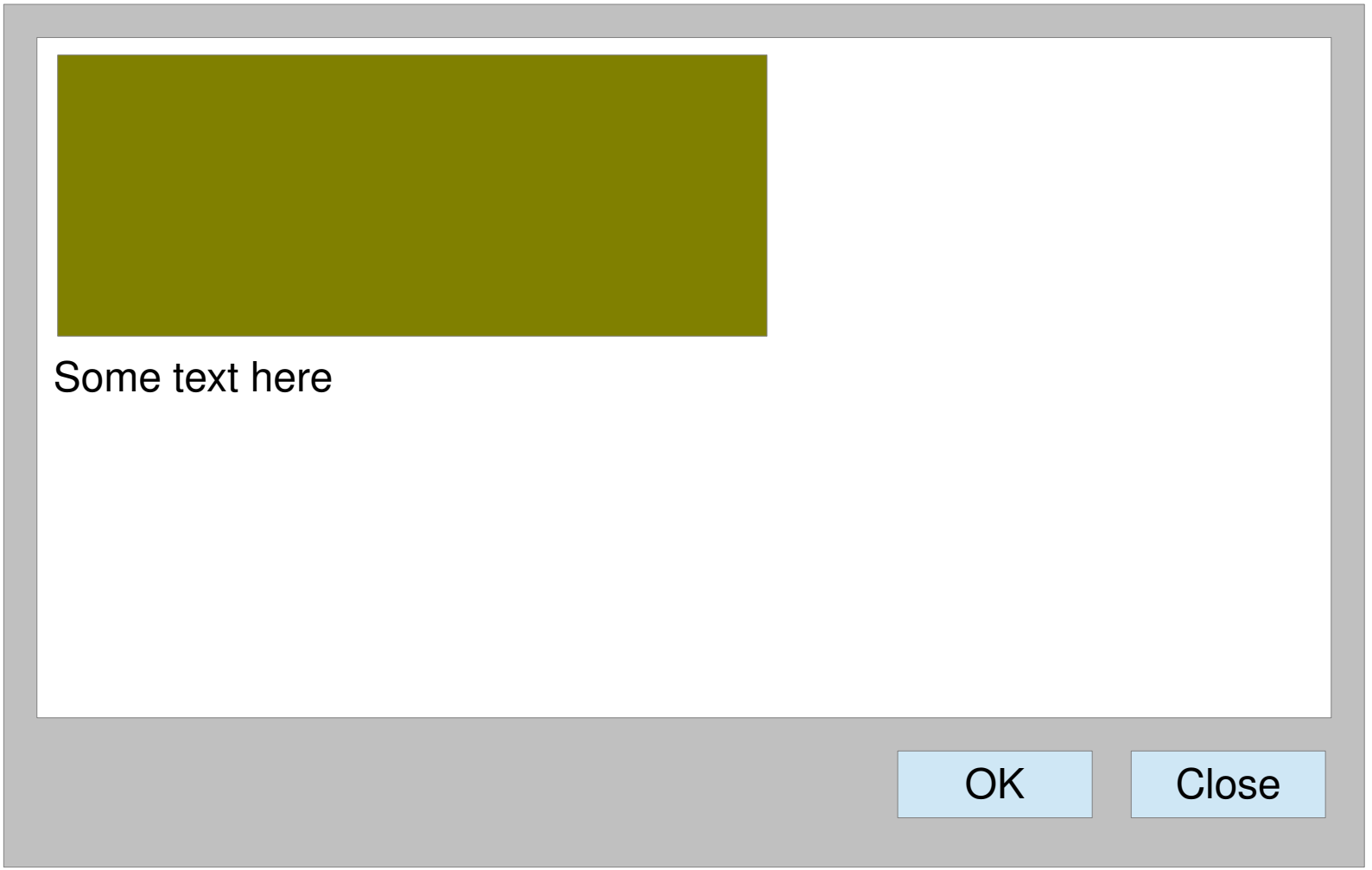
Some text here

OK

Close


Some text here
OK
Close


Some text here
OK
Close


Some text here
OK
Close

Things not covered

- OpenGL, Cairo interaction
- Printing
- Font handling
- RAD development via FLUID
- etc...

Thank you!

http://www.fltk.org

http://www.fltk.org/newsgroup.php

sanelz@gmail.com