Ruby Workshop

# Ruby の GUI 環境について

前橋 孝広

# 内容

- GUI とは
- スクリプト言語と GUI の相性
- Ruby の GUI 拡張ライブラリ
- Ruby/Tk と Ruby/Gtk
- GUI プログラミング
- Ruby のGUI 環境の課題

# GUI とは

- Graphical な User Interface (⇔ CUI)
- 知らなくてもある程度使えるインターフェース
  - できることを選択肢として見せ、選ばせる
- 現実世界との類似性
  - ボタンを押すと引っ込むとか
- 見た目は重要

# GUI の基礎知識

- ウィンドウシステム
  - 直線、四角、ビットマップの描画
  - マウス、キーボードのイベント処理
- ツールキット
  - 頻繁に使うパターンをまとめたもの
  - ウィジェット
    - ボタン、メニュー、スクロールバーなど
  - 例: Tk, Gtk, Motif, Qt など
    - いろいろあるのが良いところでもあり悪いところでもある
- イベントドリブン
  - イベント処理ループからコールバックルーチンを呼び出す

# スクリプト言語と GUI の相性・速度

- 実行時間の多くはインタープリタの制御部分で消費されているのではない
- 動作の大半は、C で書かれたライブラリレベルで実行
  - 例: テキストウィジェット
    - 文字列の挿入・削除、スクロール
- 人間の速度に反応すればいいだけ
- どうしても時間のかかる処理は拡張ライブラリで
  - 例: TkMandel
  - 例: Imlib
- マシンはどんどん高速化している

# スクリプト言語と GUI の相性・生産性

- GUI プログラミングは、割と決まりきったことをごちゃごちゃ書かなければならない
- GUI アプリケーションでは、GUI に特化した細かい処理が7から8割を占める
  - 使うは天国、作るは地獄
  - 作るも天国でないとやだ
- 少し修正しては再実行のサイクルが多い
- 結論: スクリプト言語と GUI の相性は「抜群」

# Ruby と GUI の相性

- GUI ツールキットそのものがオブジェクト指向的考え方
  - 各ウィジェットのクラスからインスタンスを生成
  - ウィジェットはそれぞれ属性を持つ
  - 同じクラスのウィジェットは同じ振る舞いをする
- ウィジェットはオブジェクトそのもの
  - 他のオブジェクトと統一的に扱える
- Ruby の使いやすさで GUI を扱えるのは快感
- 結論: Ruby と GUI の相性は「抜群」

# Ruby の GUI 拡張ライブラリ

[ruby-list:9906] より

- tk
- gtk
- xtoolkit
- xview
- xforms
- ezwgl
- xfltk(はまだ作業にかかってない？)
- openpgl(はGUIツールキットではないか)

# Ruby/Tk

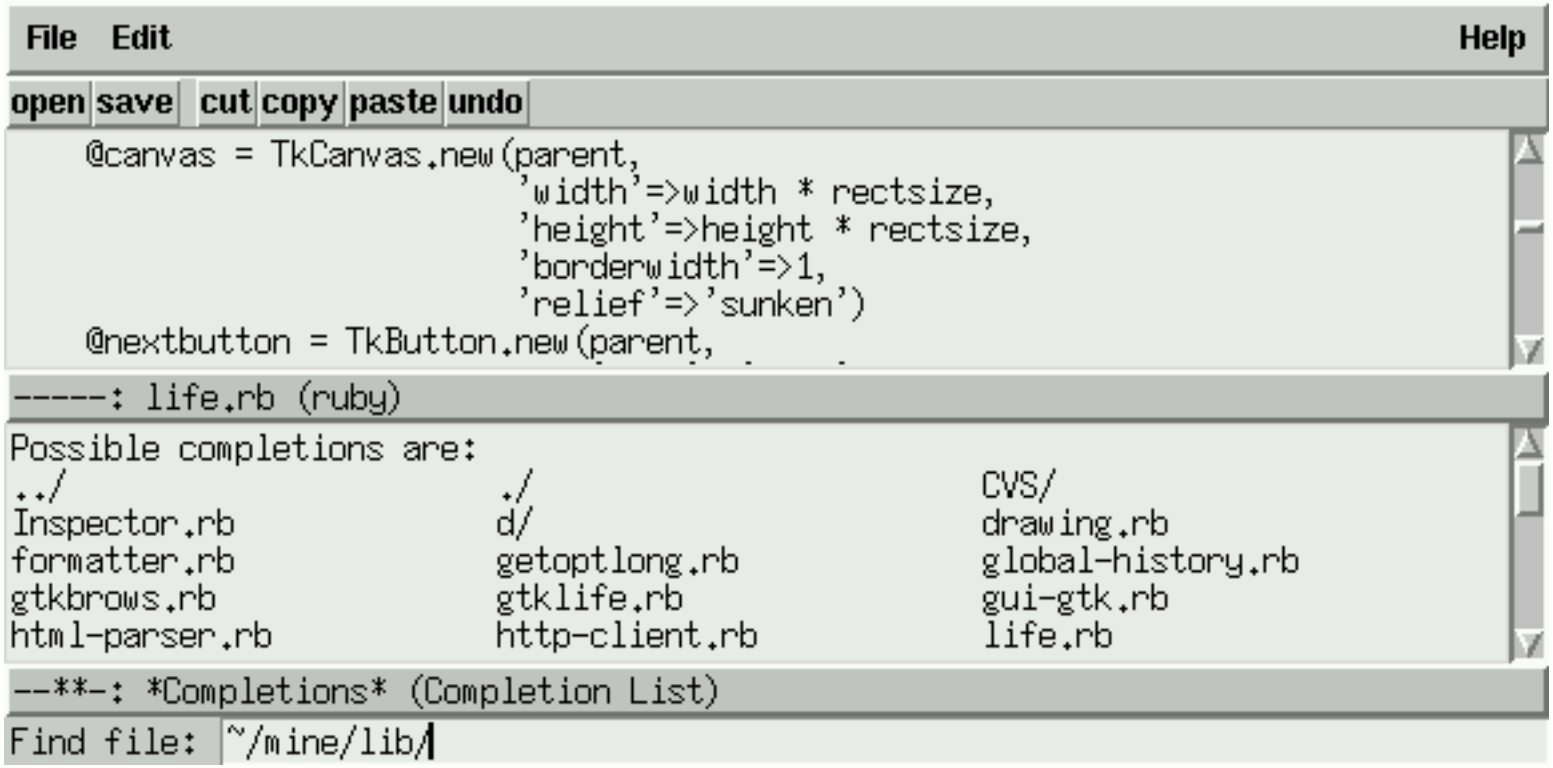
- Ruby らしい記述で Tk の GUI を実現
- TkObject を頂点とするクラス階層
- TkObject
  - TkWindow
    - TkToplevel
    - TkFrame
    - TkLabel
    - ...
  - TkImage
  - ...
- Tcl/Tk 自体はオブジェクト指向言語ではない

# Ruby/Tk

- Tcl/Tk への wrapper
  - 昔は内部で wish を起動していた
  - Tcl は邪魔:-p
  - 将来は pTk(portable Tk)?
- コールバックを実現している部分がややトリッキー
- ruby + tk.rb + tcltklib.so + tkutil.so + libtcl + libtk
- Ruby/Tk チュートリアル
  - http://www.cdrom.co.jp/~hiramatu/#RubyTK

# Ruby/Tk の application

- SGmail -- IMAP4, POP3 対応のGUI なメーラ
- mine -- マルチバッファなテキストエディタ
  - Ruby mode, Ruby interaction mode
  - 簡易クラスブラウザ、インスペクタ、ライフゲーム内蔵
  - 無制限 undo
  - Mine is not Emacs



# Ruby/Gtk

- Ruby から Gtk+ ライブラリを使うための拡張ライブラリ
- Gtk+ はGIMP のために開発、GNOME と共に発展中
- Gtk+ は最初からオブジェクト指向なツールキット
- Gtk+ は最初からインタープリタバインディングを考慮
  - Perl, Python, Guile, Pike, ...
- Windows 環境でも動く

# Ruby/Gtk

- ruby + gtk.so + libgtk
- Gtk のクラス階層がそのまま Ruby のクラス階層に対応
- (繰り返さない)イテレータによる signal_connect
- APIドキュメント
  - http://www.ueda.info.waseda.ac.jp/~igarashi/ruby/gtk-ja.html
- Ruby/Gtk のチュートリアルページ
  - http://ruby.freak.ne.jp/gtk/

# Ruby/Gtk

- Gtk を C から使う場合に比べてより自然なオブジェクト指向的な記述ができる

- C の場合
  - gtk_container_border_width(GTK_CONTAINER (window), 10);

- Ruby の場合
  - window.border_width = 10

# Ruby/Tk, Ruby/Gtk の比較

- 余計なもの(tcl)が介在するだけ Ruby/Tk のほうが遅い?
  - しかし実用的には両者とも十分
- Ruby/Tk は Thread との相性が良くない
- Ruby/Tk の方がお手軽なかんじ

# Ruby/Tk, Ruby/Gtk の比較

- Ruby/Gtk は、ウィジェットの配置が面倒
- Gtk の方がかっこいい? 流行っている?
- Gtk はドキュメントが少ない

# Ruby/Tk プログラム例

```
require "tk"
TkButton.new(nil, ’text’=>’hello’,
        ’command’=>
        proc{print "hello\n"}).pack(’fill’=>’x’)
TkButton.new(nil, ’text’=>’quit’,
        ’command’=>’exit’).pack(’fill’=>’x’)
Tk.mainloop
```

# Ruby/Gtk プログラム例

```
require ’gtk’
window = Gtk::Window::new(Gtk::WINDOW_TOPLEVEL)
box = Gtk::VBox::new(FALSE, 0)
window.add(box)
button = Gtk::Button::new("hello")
button.signal_connect("clicked") {print "hello\n"}
box.pack_start(button, TRUE, TRUE, 0)
button.show
button = Gtk::Button::new("quit")
button.signal_connect("clicked") {window.destroy; exit}
box.pack_start(button, TRUE, TRUE, 0)
button.show
box.show
window.show
Gtk::main()
```

# GUI プログラムの流れ

- ウィジェットの生成
- ウィジェットの各プロパティ設定
- コールバックルーチンをイベントにバインド
- ウィジェットの配置・表示
- イベントループに突入

# GUI プログラミングの注意点

- 上から下への直列の実行ではない
  - どんな順序でイベントが発生するかわからない
- コールバックルーチンは短時間で処理を終えなければならない
  - そうしないと、他のイベントは待たされる
- 長い処理時間がかかるものは分割してタイマーで実行
  - アニメーションなど
- ブロックする I/O を扱う場合は特に注意
- →すみやかにイベントループに制御を戻す

# 部品の組合わせによるプログラミング

- 大きな一枚岩のようなプログラムではなく、小さな部品の組み合わせによるプログラミング
  - 開発が容易
  - 再利用可能

- 部品どうしをつなぐ接着剤(glue)としてスクリプト言語を活用

- 部品
  - ツールキット
    - ウィジェット
  - 組み込みクラス
  - 他の拡張ライブラリ

# オブジェクト指向と GUI プログラミング

- ウィジェットの抽象化・カプセル化
  - 実装を隠蔽
- オブジェクト指向の本質は継承ではない
  - 下手な継承は上位クラスの実装に依存する
- ウィジェットの組合わせで新しいウィジェットを作る
  - 継承よりオブジェクトコンポジション
- クラスどうしの結び付きを弱くする
- オブジェクトとして統一的な操作

# Ruby の GUI環境: 今後の課題

- ドキュメントをもっと充実させる
  - Ruby を覚え、Tcl/Tk あるいは Gtk を覚え、C のソースを読む
  - 参考になるサンプルも少ない
- GUI Builder があった方が良い?
- キラーアプリケーション

# まとめ

□スクリプト言語と GUI の相性は抜群

□Ruby と GUI の相性も抜群

□ドキュメントの充実を