日本 Ruby 会議 2008 (2008/06/21)

# Web ばかりが UI じゃない！

## ～ 新しくなった Ruby/Tk ～


永井 秀利 (*nagai@ai.kyutech.ac.jp*)

九州工業大学 / Rubyist 九州


# Ruby と UI

- **今どき，UI なら Web でしょ？**
  - 猫も杓子も「on browser」
  - Ruby と言えば Rails ？
- **ローカルのちょっとしたツールの UI で使うには Web は過大**
  - 多くのメモリを必要とする重たい Web ブラウザ
  - Web サーバの常時稼働が必要
- **インタラクティブ性は GUI が上**
  - Web は「send and response」が基本
  - 「Ajax 等で…」は非常に面倒
  - プログラム（サーバ）からの働きかけや特殊なライブラリ利用は…

**GUI が適切な分野はたくさん存在**


# Ruby/Tk

- **Ruby 用 GUI ライブラリの一つ**
  - GUI ライブラリ中，唯一の標準添付
- **Tcl/Tk の wrapper**
  - Tcl/Tk をオブジェクト指向的に使う
  - ほとんどの Tcl/Tk スクリプトや拡張ライブラリをそのまま利用可能
- **「(La)TeX かワープロか」ー 今風の GUI ライブラリとの違い**
  - Tk： primitive だが多機能な少数のウィジェット集合
    欲しいものは組み立てることが前提
    「好きにコントロールしたい」ー (La)TeX 的？
  - 他： 目的別に多種のウィジェット
    使いそうなものは基本的に存在しているのが前提
    「楽に構築したい」ー ワープロ的？


# Ruby/Tk の特徴

- **お手軽さではピカイチ**
  - Unix, Windows, Mac OS X 等多くのプラットフォームで動く
  - 面倒な「前置き」処理の必要なし

**例：Hello World ボタン**

```
require 'tk'
TkButton.new(:text=>'Hello, World!!',
             :command=>proc{puts 'Hello, Ruby/Tk!!'}).pack
Tk.mainloop
```

Linux/Tk8.4

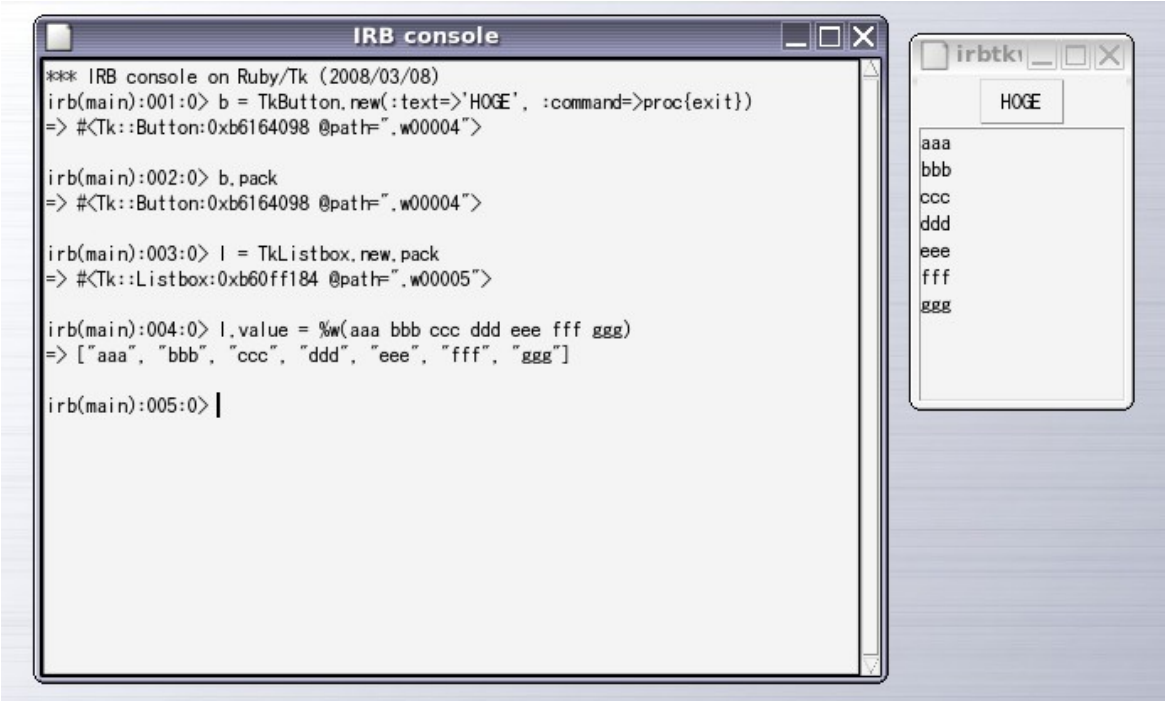
Linux/Tk8.5

Windows/Tk8.4

- **極めて高機能なキャンバスウィジェットとテキストウィジェット**
  - これらがあるからこそ Tk から離れられないという人も


# irbtkw.rbw

- **インタラクティブ Ruby/Tk with コンソールウィンドウ**
  - Ruby ソースアーカイブに付録のサンプルスクリプト
  - Tcl/Tk における wish と同様に，インタラクティブに GUI 操作
- **コンソール部も Ruby/Tk**
  - **Windows においても，キーボード入力待ちによるブロックなし**




# Ruby/Tk の情報源

- **ドキュメントの不足は大きな問題点**
  - Ruby⇔Tcl/Tk の仕組みが分かっていれば，Tcl/Tk のマニュアルとRuby/Tk のソースコードとがドキュメントの代わりにはなるが…
- **書籍**（Ruby1.6 の頃のかなり古いものだが，今でも参考になる）
  - Ruby を256倍使うための本ー界道編（アスキー）
  - Ruby アプリケーションプログラミング（オーム社）
- **Ruby/Tk 講習会資料**（やや古いが比較的まとまった解説かも）
  - URL: *http://www.dumbo.ai.kyutech.ac.jp/~nagai/RubyTk-seminar200709-files.zip*
- **Ruby のソースアーカイブに同梱のサンプル**
  - Widget Demo： デモスクリプトを使った try&error での学習が可能
  - 付録のサンプルスクリプト： そのままクラスとして使えるものも存在


# 新しい Ruby/Tk 情報源：TkDocs

- **注目に値する Tk の総合情報サイト**
  - Tcl に限らず，各種言語の Tk バインディングの情報の掲載が目標
  - 現状は Tcl/Tk と Ruby/Tk の情報を掲載
  - 各プラットフォーム（Windows, Mac OS X, Linux）の比較表示あり
- **チュートリアルでは Tcl/Tk と同時に Ruby/Tk のコードも掲載**
  - Tcl/Tk のコードとの比較がやりやすい
  - Ruby/Tk のコードの記述スタイルがまちまちなのはやや難点かも
- **現状の情報量はまだ十分ではないが，今後の発展に期待**
- URL： *http://www.tkdocs.com/*


# 最新の Ruby1.8.7，1.9.0 での強化・改良

- Tcl/Tk8.5 の新機能への対応に伴う強化・改良
- ユーザの利便性向上のための強化・改良


# Tcl/Tk8.5 の新機能に伴う強化・改良

- 標準ウィジェットの見掛けの改善
- フォント表示の改善と新たなフォントエイリアス定義の導入
- テーマエンジンの標準装備とそれに伴う新ウィジェット導入
- チェックボタン，ラジオボタンの tristate 属性
- テキストウィジェットの peering
- Tk.messageBox の detail 属性
- TkWindow#wm_manage，TkWindow#wm_forget の新設
- Tk::Wm.attributes の強化
- Tk.inactive，Tk.reset_inactive の新設

… など

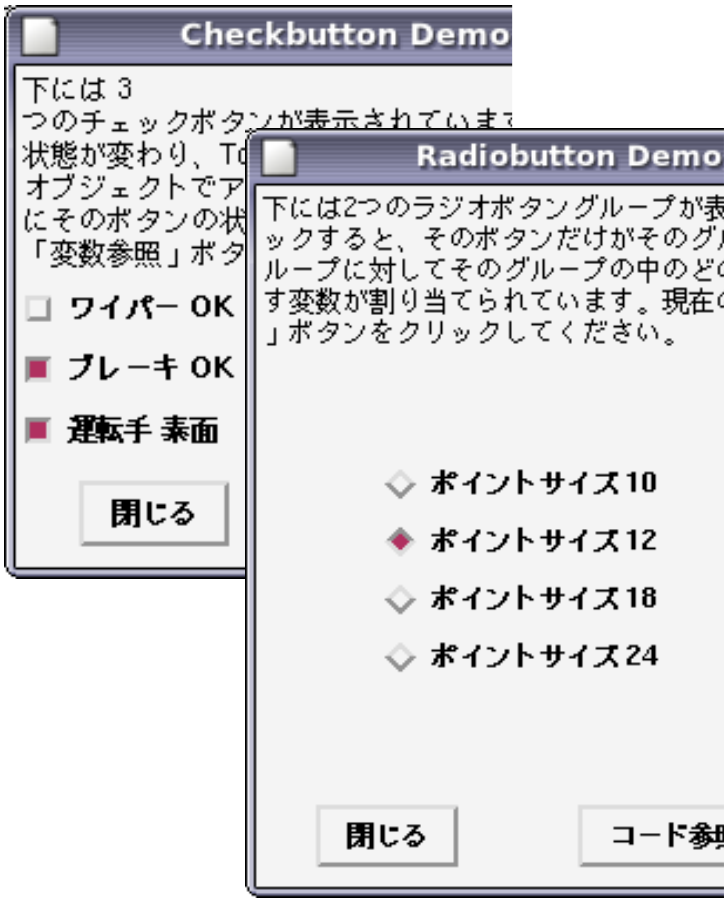

# 利便性向上のための改善

- Ttk 標準化に伴いデフォルトウィジェット集合という概念を導入
- バインディングにおける callback subst key 指定方法の改良
- callback における break，next での脱出
- TkTimer#at_end
- Tk.sleep, Tk.wakeup
- pack, grid のパラメータパターンの増加
- TkWindow#configure 等のオプション取扱いの強化
- 新しいスタイルでの Tcl/Tk 変数の trace 設定に対応
- コンパイル時の configure オプション指定の強化

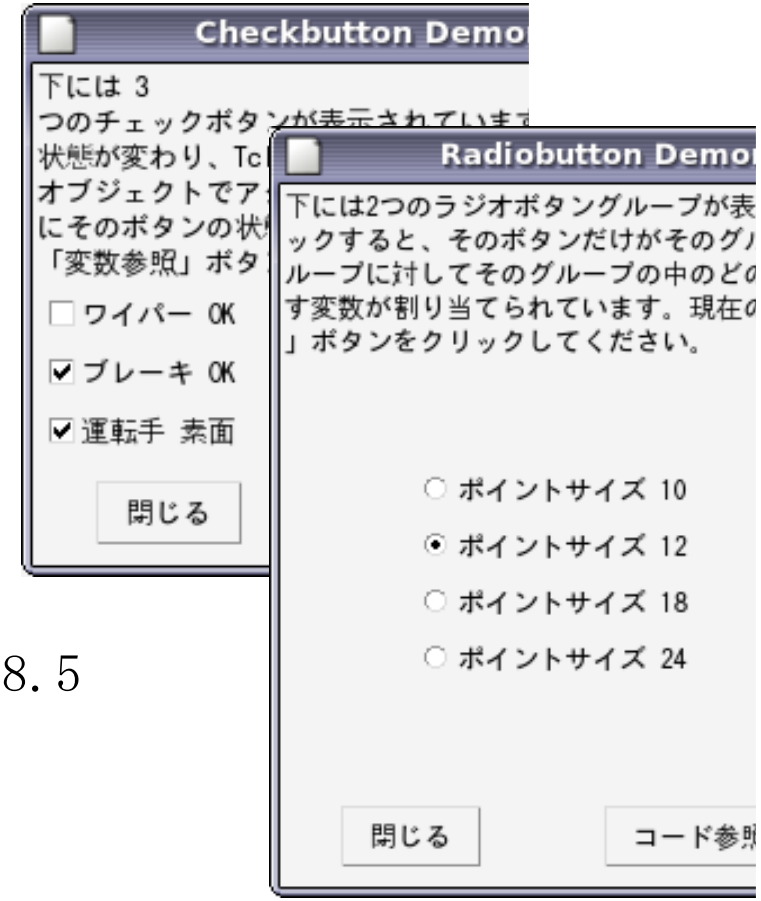
… など


# 標準ウィジェットの見掛けの改善

- **プラットフォームに適合したより現代的な見掛けに改善**
  - 特に X Window System 上のラジオボタン，チェックボタンでの改善が顕著

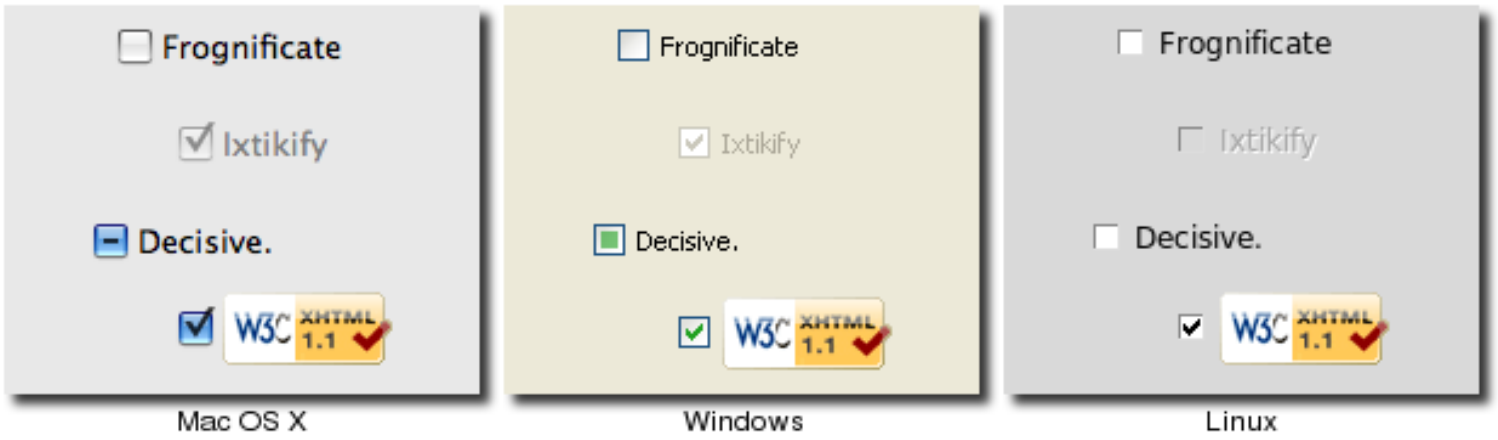



Tcl/Tk8.4 ⇔ Tcl/Tk8.5




# テーマエンジン： Ttk 拡張

- **古くは Tile 拡張とも呼ばれていた Tcl/Tk 拡張ライブラリ**
  - テーマ指定に基づき，ウィジェットの見掛けの動的変更が可能
  - 標準ウィジェットと同種のウィジェットを用意することにより，少ない変更で Tcl/Tk の GUI の見掛けの改善を可能にする

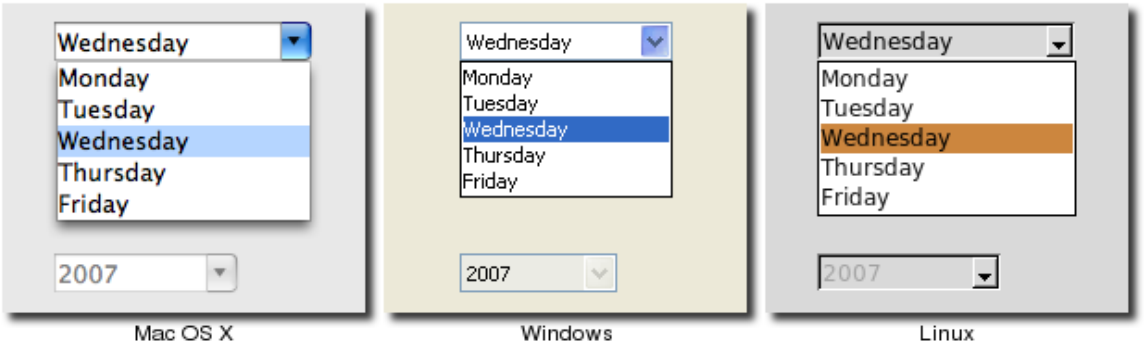


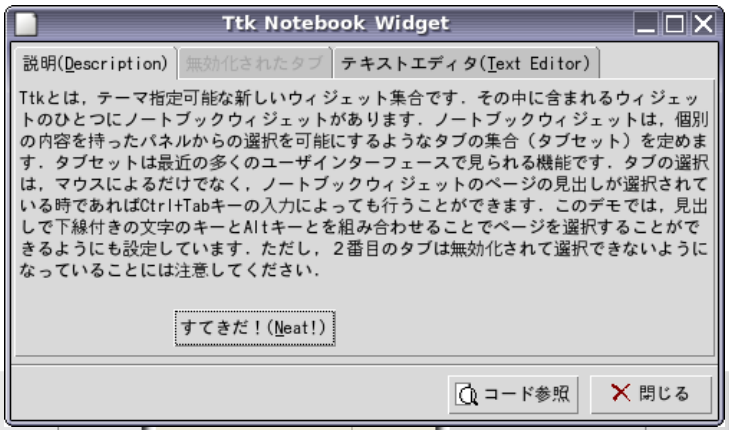

**Ttk 拡張の checkbutton ウィジェットの例**

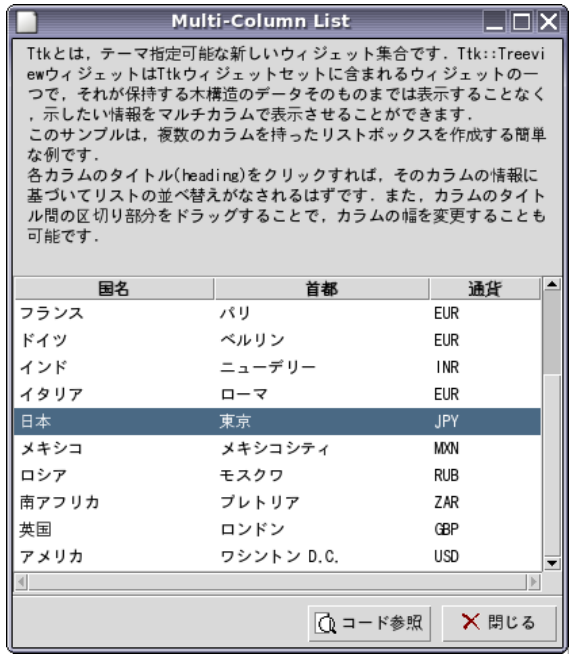
各プラットフォームに合わせた見掛けを持つことに注意
（ 図は TkDocs サイトの TUTORIAL ページより引用 ）

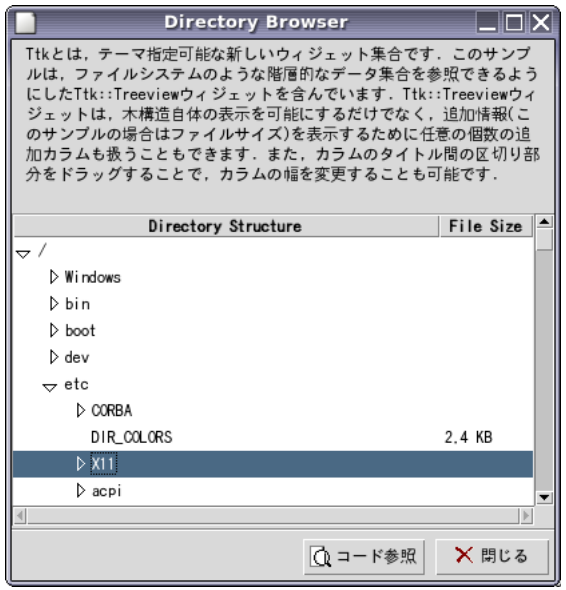
# Ttk 標準化に伴い標準化されるウィジェット

- 近年の GUI でよく用いられる種類のウィジェットの標準化
  - Ttk::Combobox
  - Ttk::Notebook
  - Ttk::Treeview
  - Ttk::Progress
  - Ttk::SizeGrip
  - Ttk::Separator




# テーマエンジン (Ttk) の標準化に伴う強化

- **従来の標準ウィジェット群の ::Tk モジュール下への移動**
  - 例： ::TkButton → ::Tk::Button
- **Ttk(Tile) 拡張モジュール (::Tk::Tile) の別名として ::Ttk を定義**
  - 標準化に合わせて，短い名前でも定義
- **Ttk 拡張の有無に両対応のスクリプトの作成を容易に**
  - トップレベル定義 (TkButton 等 ) の alias 化
  - Tk.default_widget_set= によるデフォルトウィジェット集合の切替え
    - Tk.default_widget_set = :Tk　の時，TkButton → Tk::Button
    - Tk.default_widget_set = :Ttk　の時，TkButton → Ttk::Button
  - Tk::Button, Ttk::Button などと明示することで，デフォルトウィジェット集合の設定に依存しない指定が可能


# Ttk を活用するサンプル: ttk_wrapper.rb

- 旧来の Ruby/Tk スクリプトを (100% ではないが) テーマエンジンに対応させる wrapper スクリプト

**例: tkhello.rb**

```
require 'tk'
TkButton.new(:text=>'hello',
             :command=>proc{puts 'hello'}).pack(:fill=>:x)
TkButton.new(:text=>'quit',
             :command=>proc{exit}).pack(:fill=>:x)
Tk.mainloop
```

ruby tkhello.rb


⇒

ruby ttk_wrapper -t *theme* tkhello.rb

*theme*: alt


*theme*: clam

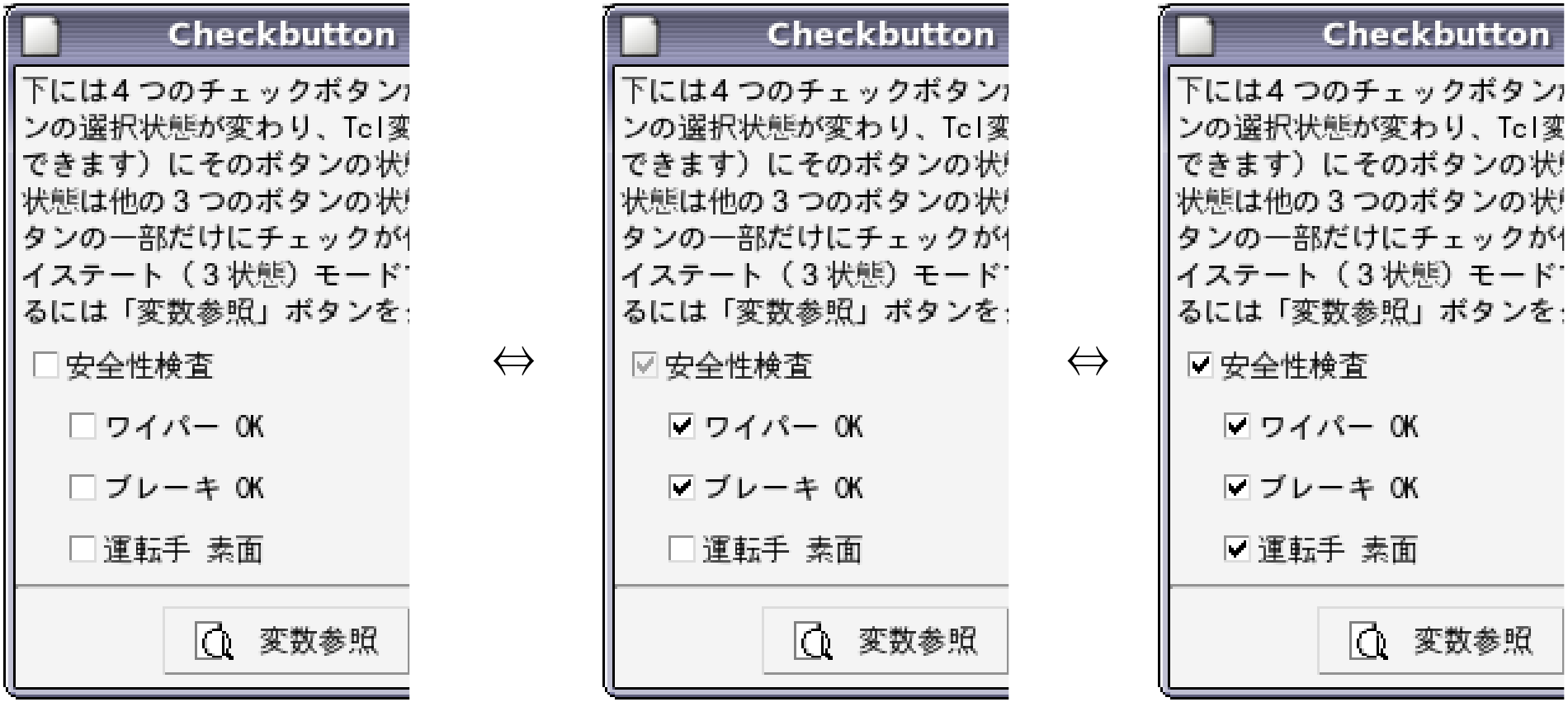

*theme*: classic


*theme*: blue


*theme*: kroc


# チェックボタン，ラジオボタンの tristate

- 「いずれも選択されていない」や「部分的に選択されている」を表現することが容易に




非選択状態　　トライステート　　選択状態

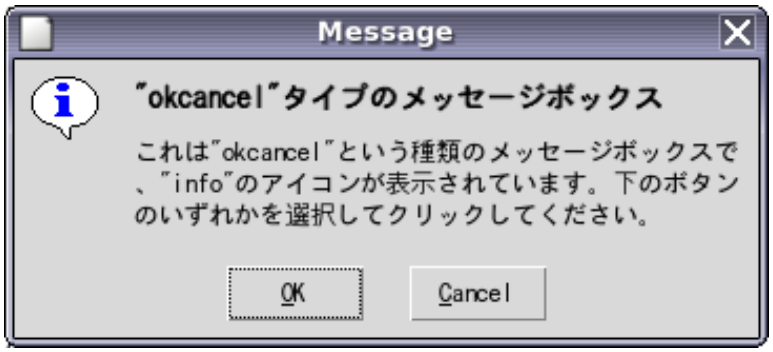

# テキストウィジェットの peering

- **複数のテキストウィジェット（peer）間で内容を対等化する**
  - いずれかの peer でテキストを変更すると，他の peer の内容も自動的に同じになる
  - カーソル位置などの情報は各 peer で個別に保持
- **TkText::Peer クラスのオブジェクトとして生成**
  - Peering 対象のテキストウィジェットを new の引数として渡す
  - 対等化するものが同一の親ウィジェットを持つ必要なし
- **マルチバッファのエディタなどを容易に実現可能**
  - テキストウィジェットは十分な編集機能を持つため，peer を生成するだけで編集用バッファを生成したのと同等になる


# Tk.messageBox の detail 属性

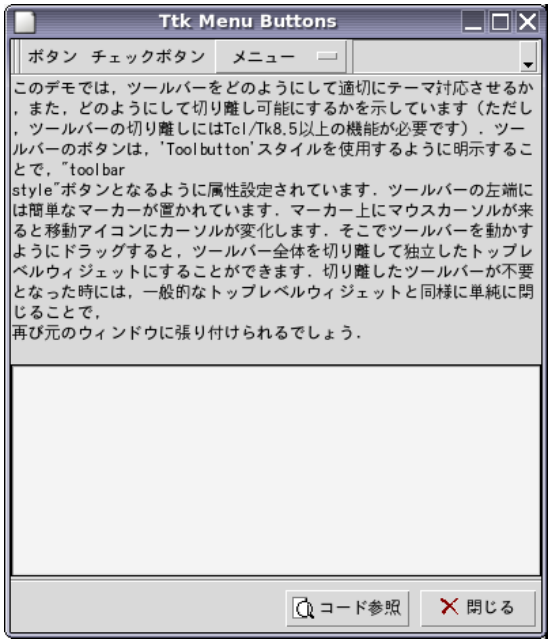
- ちょっとした確認ダイアログの表示のために頻繁に使うメソッド
- 表示テキストのための属性は従来は message 属性だけ


- 主題となるメッセージと詳細説明テキストとを区別して表示できるようにするために detail 属性を追加

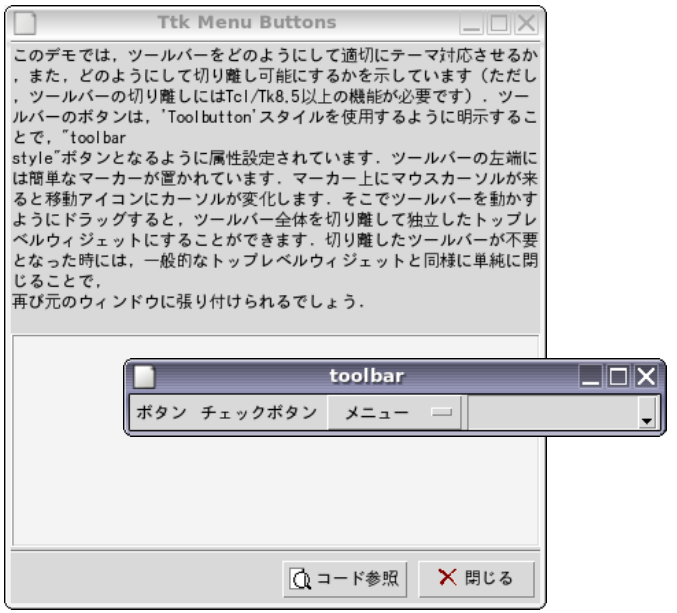



# TkWindow#wm_manage, wm_forget

- **任意のウィジェットを直接ウィンドウマネージャの管理下に置いたり，それを取り消したりできるようにするメソッド**
  - 通常はフレームウィジェットに対して用いる
  - 特定のフレーム（およびその上に配置されたもの）を独立したウィンドウとして切り離したり元に戻したりすることが可能になる

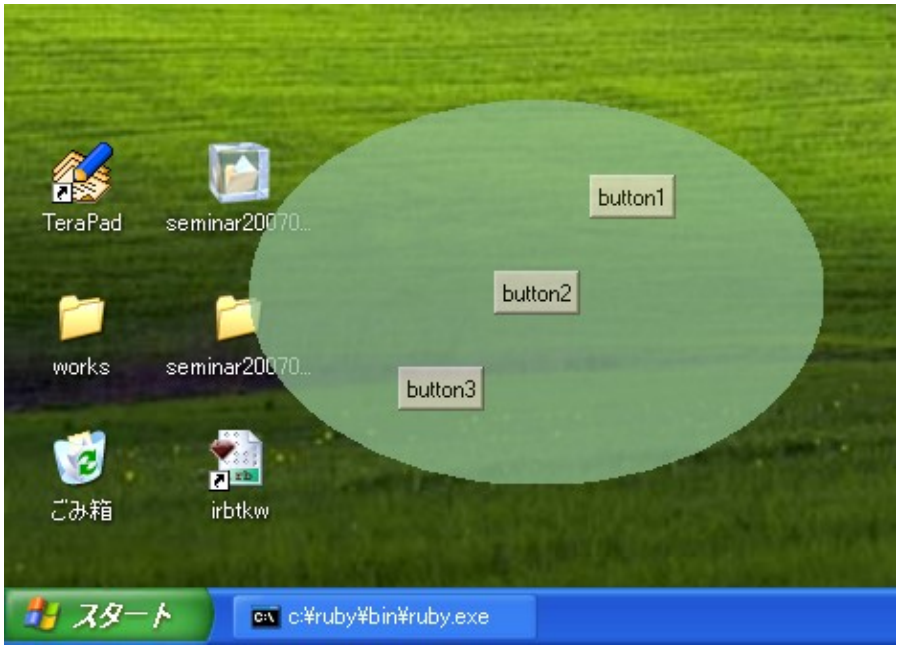



⇔



# Tk::Wm.attributes の強化

- 全プラットフォームで fullscreen 属性，topmost 属性に対応
- プラットフォームに依存したウィンドウ属性への対応の増強
  - 例えば Windows での alpha 属性，transparentcolor 属性への対応により，不定型（非矩形）・半透過の GUI 作成が可能


（サンプルの GUI 下地は単なる楕円図形だが，画像を使えば完璧！）


# Tk.inactive , Tk.reset_inactive

- ユーザがウィンドウシステム上で何らかの操作を最後に行ってからの経過時間 (ms) の参照とリセット
- サポートされているかどうかはウィンドウシステムに依存
- システムに対するユーザのアクティビティの調査，およびそれに基づく処理の実装が可能になる


# バインディングにおける subst key

- **これまでのイベント処理のための情報取得例**

```
widget.bind('Control-KeyPress'){|ev|
              w = ev.widget; x = ev.x; y = ev.y
              k_sym = ev.keysym; k_code = ev.keycode .... }
```

  - 無駄な情報変換(不要な情報までもすべてが一旦処理される)を伴う

```
widget.bind('Control-KeyPress',
            '%W %x %y %K %k'){|w, x, y, k_sym, k_code| .... }
```

  - 暗号のような情報指定文字が必要

- **新しくサポートする指定方法**

```
widget.bind('Control-KeyPress',
            :widget, :x, :y,
            :keysym, :keycode){|w, x, y, k_sym, k_code| .... }
```

  - 無駄な変換を避けつつ,「暗号」も回避


# callback からの break，next での脱出

- **以前の Ruby/Tk におけるイベント処理の制御**
  - Tk.callback_break や Tk.callback_continue を呼ぶ必要があった
    - Tk.callback_break： 今回のイベントに関するイベント処理を打ち切る
    - Tk.callback_continue： 次のバインドタグに処理を移す
- **Ruby の break，next が機能するように改善**
  - それぞれ Tk.callback_break，Tk.callback_continue に対応

```
Label = TkLabel.new(:text=>'HOGE').pack
label.bind('ButtonRelease-1'){
  p :label; Tk.callback_continue; p 1
}
Tk.root.bind('ButtonRelease-1'){
  p :root; Tk.callback_break; p 2
}
TkBindTag::ALL.bind('ButtonRelease-1'){
  p :ALL; p 3
}
```

```
Label = TkLabel.new(:text=>'HOGE').pack
label.bind('ButtonRelease-1'){
  p :label; next; p 1
}
Tk.root.bind('ButtonRelease-1'){
  p :root; break; p 2
}
TkBindTag::ALL.bind('ButtonRelease-1'){
  p :ALL; p 3
 }
```


# TkTimer ( TkRTTimer ) #at_end

- **TkTimer ( TkRTTimer )クラス**
  - 時間間隔を空けての繰り返し処理用のクラス
  - イベントループのタイマー処理を用いるため Thread 生成は不要
  - TkRTTimer はインターバル誤差累積の影響を抑制する機能付き
    （ リアルタイム処理を行うわけではなく，インターバルを誤差補正により変動させることで規定時間内の規定回数の繰り返し実行をほぼ保証するもの ）
- **TkTimer#at_end{ … 終了時処理 … }**
  - タイマーの繰り返しが正常に終了する時に呼び出す処理を登録
  - 従来は，規定回数を繰り返した後の終了時に特定の処理を呼び出すというようなコードが書きづらかったことへの対策

**例:ボタンの点滅（点滅中の再度のクリックを禁止）**

```
b = Tk::Button.new(:text=>'HOGE').pack
t = TkTimer.new(100,6){b.fg,b.bg = b.bg,b.fg}.at_end{b.state(:normal)}
b.command{t.start{b.state(:disable)}})
```


# Tk.sleep，Tk.wakeup

- **Ruby/Tk のコールバック中での Ruby の sleep 実行**
  - イベントループが停止してイベントが処理されなくなるため危険！
  - sleep している間，GUI も無反応になってしまう
- **対策は？**
  - a)Thread を生成して処理を任せ，コールバックはすぐに終了する
  - b)Tcl/Tk の after コマンドと変数トレースとを組み合わせる
- **Tk.sleep(ms=nil, id=nil)，Tk.wakeup(id)**
  - 上記 (b) の方法を楽に行うためのもの
  - ミリ秒単位で指定する( nil なら永久に sleep )
  - id ( TkVariable オブジェクトまたは変数名となる文字列)を指定していなければ wakeup できないことに注意


# ところで…　GUI とネットワーク

- **GUI は基本的にはローカルマシンで動かすもの**
  - X Window System なら remote 実行可能だが…
- **Ruby/Tk では RemoteTklp というのも提供してはいるが…**
  - Tcl/Tk の send コマンドを用いて，異なるマシン上で稼働している Ruby/Tk (Tcl/Tk) 間で相互制御を可能にする仕組み (Rubyist Magazine 0003 号でも少し触れている)
  - 色々な意味で制約多し
- **リモートデスクトップなどの方法もなくはないが…**
  - 基本的にはウィンドウシステムを丸々提供してしまう形になる (Windows Server 2008 の新しいターミナルサービスならあるいは？)
  - GUI アプリだけを Web アプリのように広く一般に提供とはいかない
- **GUI アプリだけをネットワーク越しにプラットフォームをあまり気にせず手軽に使うなんてのは，やはり無理な話か？**


# 「YAHOO知恵袋」より…

質問日時： 2007/7/29 02:34:54　解決日時： 2007/8/12 03:31:42

解決済み **rubyで、GUIアプリも作れるようですが、サーバ側にそのソフト（ruby/tk?）が備わっ...**

rubyで、GUIアプリも作れるようですが、サーバ側にそのソフト（ruby/tk?）が備わっていれば、クライアントはただのIE6でいいのですか？

回答数： 2　　質問した人：

**ベストアンサーに選ばれた回答**

回答日時： 2007/7/29 09:28:58

あほか。
サーバで動いているGUIアプリの画面をクライアントからどうやって見るんだ。

**ベストアンサー以外の回答**

回答日時： 2007/7/30 09:34:58

NOです。
ＧＵＩアプリを作るというのは、Webでシステムを作るのとはまったく違います。
通常はそのマシンで動かすものですから、「クライアントはただのIE6」も何も必要ありません。

X-Windowだったら表示先を変更すればできないことないでしょうけど、
画面のほうが「ディスプレイ・サーバ」とサーバにならないといけないよね。


# 「YAHOO 知恵袋」より…

- **質問:**

  ruby で GUI アプリも作れるようですが，サーバ側にそのソフト（ruby/tk？）が備わっていれば，クライアントはただの IE6 でいいのですか？

- **ベストアンサーに選ばれた回答:**

  あほか．サーバで動いている GUI アプリの画面をクライアントからどうやって見るんだ．

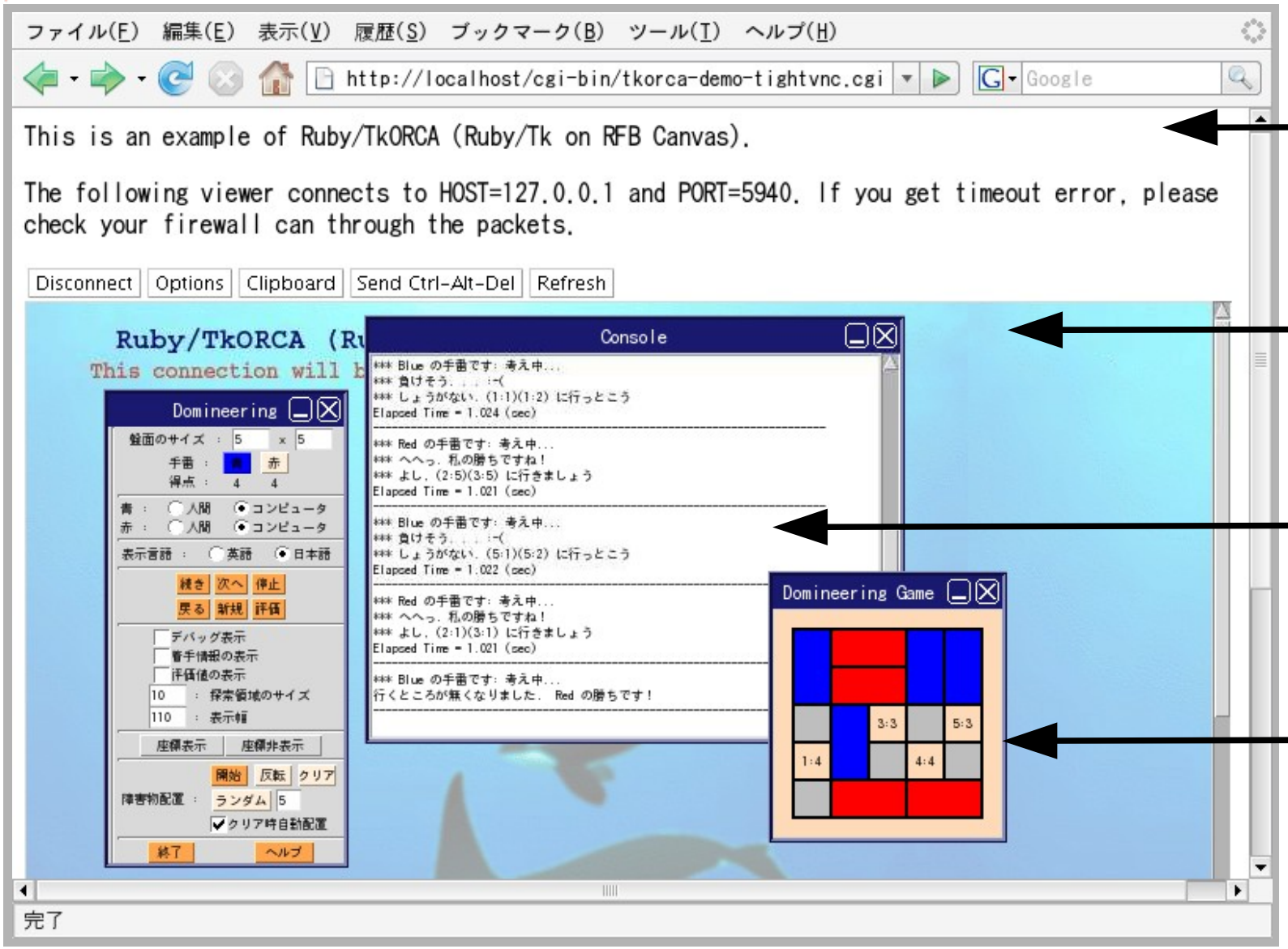
**「Ruby/Tk で」というなら，できなくもないけど．．．**

# 例えばこんな GUI アプリが…



マルチウィンドウ，コンソール利用，…

# 例えばこんな感じで…




普通のブラウザ

Ruby/Tk による
ウィンドウマネージャ

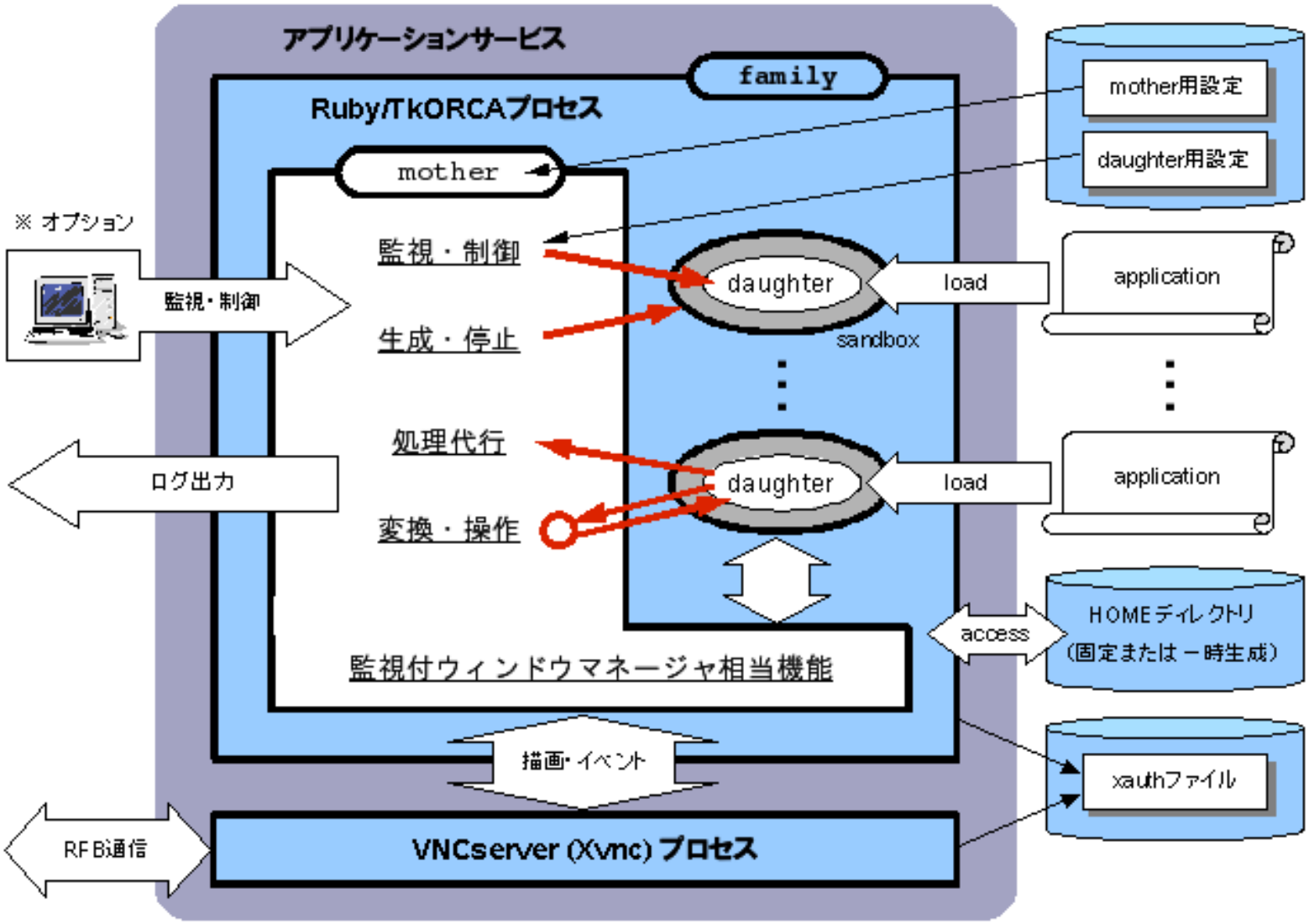
Ruby/Tk による
コンソール

アプリケーションの
ソースは，ほとんど
変更の必要なし

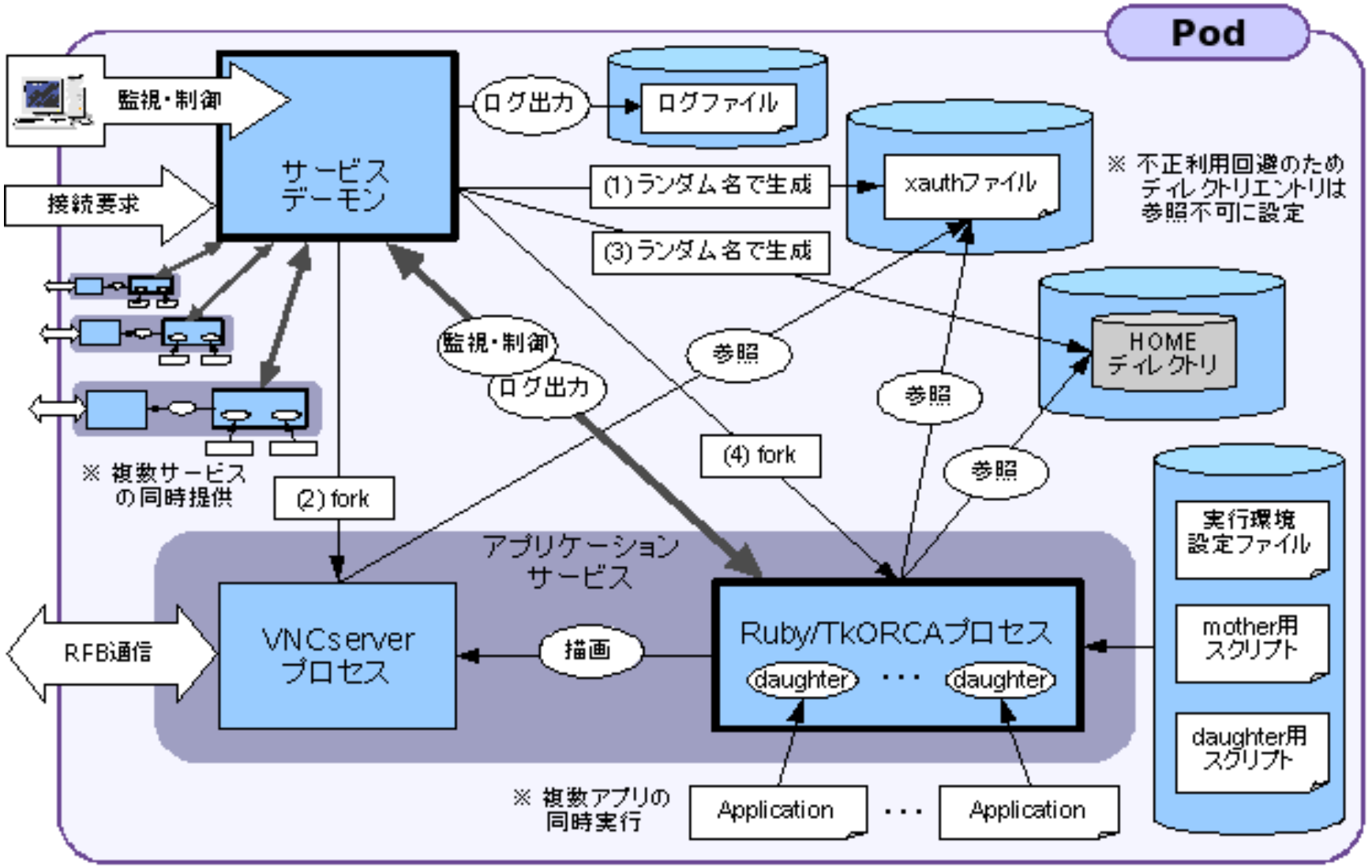

# Ruby/TkORCA (Ruby/Tk on RFB Canvas)

- **Ruby/Tk アプリをネットワーク越しに提供するための枠組**
  - クライアントには Ruby も Tk も必要なし
  - サーバとクライアントとの間で OS が異なっていても問題なし
  - Tk のコンテナウィジェットに埋め込み可能であれば，Tk や Tk 拡張ではない画面描画ライブラリでも利用可能
- **不特定多数に対して同時にサービスを提供することも可能**
  - sandbox を作成してセキュリティを確保
  - 利用者の操作ログを取ることも可能
  - irb を操作するように，管理者が稼働中のサービス（Ruby/Tk アプリ）に介入することも可能
- **通信経路に対する独自のセキュリティ対策は特になし**
  - 必要なら，SSL を使った tunneling 等でセキュリティを確保する
  - 例えば stunnel（*http://www.stunnel.org/*）等で容易に実現可能


# Ruby/TkORCA ～ 個々のサービスの構造




# Ruby/TkORCA ～ サーバの構成




# Ruby/TkORCA のデモ

- **ネットワーク越しにブラウザ上や VNC ビューアで動かしてみる**
  - Adobe AIR 等に見られる Web→ローカルのアプローチとは逆方向のローカル→ Web（ネット）のアプローチ
- **Nintendo-DS をクライアントにしてみる**
  - メモリ 4MB のみ，外部記憶なしの小型端末としてのテスト
  - もちろん DS は無改造で，当然 Ruby なし，ウィンドウシステムもなし
  - もう少しネットワーク I/O が速ければもっと実用になるのだが…（最高で 150kbps 程度しか出ないのではさすがに厳しい）

**もし興味を持っていただけたなら…**

β 版（と言いつつ α 版に極めて近い）の入手先：

*http://qwik.jp/ruby-tk/RubyTkORCA.html*

（現状ではまだ多数のバグや Tcl/Tk8.5 対応に不完全な部分があるので注意）


# おわりに

- **「今さら Tk 」？**
  - 「古くから」存在するツールキット ≠「古い」ツールキット
  - Tcl/Tk8.5 に見られるように，Tk の進化は止まってはいない
  - 使われ続けているという事実に注目を
- **毛嫌いせずに試してみては？**
  - 他の GUI ツールキットを試したものの，結局 Tk に戻って来る人も
  - 使ってみたら意外と気に入ることもあるかも
- **「Ruby/Tk 質問・要望受付コーナー」をやります！**
  - 初心者質問も大歓迎！
  - 「こんな機能を」とか「この Tcl/Tk 拡張のサポートを」とかも，ぜひ
  - 少なくともこの発表後の昼休み中は待機している予定です
  - 場所は“ Rubyist 九州のブース”ですので，お気軽にどうぞ


ご清聴ありがとうございました