Lisp GUIs

or “Bloodied but Unbowed”

Robert P. Goldman

TC Lispers

2009-08-18 Tue

Introduction

- Every now and then I have to write code for someone who doesn't realize inferior lisp mode in emacs *is* a GUI.
  - This is a classic Phil Agre hassle: "hassles, which are small bits of trouble that recur frequently in routine patterns of activity."
- I have tried three solutions, only one of which works satisfactorily:
  1. Use a pure CL solution:
     - CLIM (McCLIM, specifically)
     - Garnet
  2. Use a web browser as the GUI, employing AJAX;
  3. *Have someone else write a GUI that will use my CL program as a server.*
- There's one that doesn't work at all: FFI to another language's GUI package.
- Here's a bit about each of these.

Other Languages' UI Toolkits

- Why don't you just use *this toolkit* implemented in *this language*?
- *this language* = Java:
  - No portable link to Java.
- *this language* = C++
  - C++ FFI is a big pain
  - To know the package (e.g., Qt) you need to know about how the horrible C++ class system works
- These frameworks are dismayingly static.
  - REPL is critical for GUI programming.
  - Libraries typically have boatloads of parameters and are fussy about them.

CLIM: Common Lisp Interface Manager

The Good

- All CL solution.
- `PRESENT` and `ACCEPT`.
  - `PRESENT` arbitrary Lisp data structures.
  - Whenever you need input of a particular type, just `ACCEPT` it from everywhere on the display.
- Multiple backends.

The Bad

- UIs that feel unnatural to users. Gadgets versus `PRESENT` and `ACCEPT`.
- Huge learning curve, crummy documentation.
- McCLIM (open source CLIM) is *unpredictably* unfinished.
- The API is huge, so making it work is daunting.

Garnet: Constraint-based UI development

The Good

- All CL solution.
- Constraint-based layout.
- Huge set of built-in components.
- Natural feeling UIs.
- Brilliant documentation.
- Great for building novel UI components.

The Bad

- Huge learning curve: built on prototype-based object system.
- Back-end has bitrotted — X has moved while Garnet has been static.
- No non-X backend.
- Default look and feel is ugly (but this is readily fixable).

Using Web Browsers as GUI

What do you do if you:

- Don't really know or love C++ or Java;
- Want a cross-platform GUI;
- Want a repl-style development framework;
- Want a rich set of widgets that are reasonably native-looking?

Use the browser!

Preliminary Lessons Learned

- Build your lisp program as a web server. There are a plethora of server frameworks.
- Use AJAX-style client-server interactions.
  - Plenty of libraries for XML and JSON serialization from CL.
  - Use JSON instead of XML. I learned the hard way.
- Consider going native and programming Mozilla:
  - Evade the sandbox;
  - Cross-browser compatibility is a horror;
  - Get a richer set of widgets.
- Still can be ugly:
  - Still a horrific learning curve;
  - JavaScript can be cumbersome (get a good library!);
  - JS debugging and repl use is still cumbersome.

Got anything better?