CCL/Cocoa Interfaces

Paul Krueger, Ph.D.

November, 2009

Interface Goals:

- Looks native to platform
- Development effort commensurate with interface complexity
- Easy blending with Lisp code
- Usable in either stand-alone executables or from within standard CCL IDE
- Cross-platform/OS portability

My Search:

- First learn Cocoa using Objective-C, Xcode and Interface Builder (IB)
- .../ccl/examples/cocoa/easygui
- Using Apple's Xcode and IB for Lisp (like Ruby & Python do)
- Ease of IB combined with interactivity of CCL

Goal 1:
Native Look and Feel

- Only one answer for Macintosh:
  Cocoa in some form
- Turns out Cocoa may be more portable than
  I initially thought ...

Goal 2: Effort/Complexity Match

- Cocoa UI objects are very capable and therefore also very complex
- Cocoa UI object interaction protocols are quite complex
- Interface Builder (IB) provides a graphical interface that reduces apparent complexity, but permits complete customization

Goal 3:
Easy Blending with Lisp

- Cocoa is based on Objective-C
- Objective-C is a very nice language (as close as a C dialect can get to something Lisp-like)
- CCL has provided a complete bridge to Objective-C and to Cocoa objects/functions.
- Can create Lisp classes that inherit from Objective-C classes
- Class slots accessible to both Lisp and C

Goal 4: Stand-alone or loadable

- All my example code loads into standard CCL IDE
- Dynamically re-define lisp or objective-C functions (but not classes)
- Nothing precludes using same code with minor changes/additions in stand-alone app
- See Mikel Evins' APIS work for stand-alone apps: http://explorersguild.com/mikelevins/Apis_1_0.zip

Goal 5: Portability

- OK, I'm a mac guy and don't really care too much about this, but ...
- The Cocotron: http://www.cocotron.org/Info Objective-C environment for many platforms and operating systems
- Support for Cocotron in CCL Objective-C bridge code
- But... Mac needed for development today

Lisp/Cocoa Interface Projects

- NIB Loading
- SimpleSum
- Menus
- Speech Synthesizer
- Lisp Packages
- Loan Calculator

Project 1: NIB Loading

Key Concepts:

- Interface Builder
- NIB and XIB files
- NIB loading
- Lisp class as NIB file owner

Project 2: SimpleSum

Key Concepts:

- windows
- text boxes
- buttons
- Lisp action methods

Project 3: Menus

Key Concepts:

- Menu creation
- Menu addition
- Object delegates
- "First Responder" and responder chains
- Menu actions

Project 4: Speech Synthesizer

Key Concepts:

- Speech controller
- Radio buttons
- Runtime view definition
- Memory management

Project 5: Lisp Packages

Key Concepts:

- TableViews
- Lisp data sources
- Lisp accessor functions accessible from Objective-C objects

Project 6: Loan Calculator

Key Concepts:

- Bindings
- Number formatters
- Slider controls
- Control enabling/disabling
- Alert panels

Future Projects

- Custom graphic data documents
  - NSDocument
  - Cocoa drawing
  - JPEG input/output
  - Printing support
- OpenGL window of some sort