Rice University

Department of Psychology

Lab Transition From MCL to Clozure Common Lisp

Author:
Clayton Stanley

Supervisor:
Dr. Mike Byrne

August 9, 2012

Contents

1 Introduction

The purpose of this code is to enable Clozure Common Lisp to read in GUI source code written for Macintosh Common Lisp. This enables task environments written in MCL (e.g., Phaser, Votebox, NextGen) to work with CCL with minimal code modifications.

Currently, three files are generated during the build to achieve this purpose:

1. ccl-simple-view.lisp: submodules/actr6/support
2. device.lisp: submodules/actr6/devices/ccl/device.lisp
3. uwi.lisp: submodules/actr6/devices/ccl/uwi.lisp

1. provides the language layer for CCL, so that CCL can read in MCL gui source code. You'll see a subset of Digitool's GUI specification implemented for CCL in this file.

2. and 3. provide the ACT-R device for CCL, so that models that interacted with the task environments written for MCL can now work with CCL. These files are essentially copies of the analogous files for MCL, except at the top they require 1., so that the MCL language layer is available in CCL before the rest of the code is read in.

ccl-simple-view.lisp is a concatenation of many smaller .lisp files in this repo. This was done because I find it easier / more productive to work with many smaller files during development, but it's easier / more straightfoward to provide a single file that does a single purpose to the user.

ccl-simple-view.lisp implements Digitool's GUI specification by levaraging CLozure's Objective C bridge and Apple's Cocoa framework. The majority of the structure of this file is written in CLOS style, and is built on top of Clozure's provided 'easygui' package.

2 Directory Structure

- actr6: All .lisp files needed to build an ACT-R device for CCL that uses a Cocoa display. Note the share.lisp file lives here for now, which contains the MCL GUI interface for CCL

- bincarbon: All original MCL bincarbon code. Just a few files have minor tweaks to work with CCL. Timer is broken on CCL currently though. Other files, like CFBundle.lisp don't work at all on CCL, but aren't necessary on CCL. The main goal for this folder is to document all changes to bincarbon files when migrating to CCL, while maintaining backwards compatibility with MCL.
- binccl: CCL-specific utility files. Contains a CCL implementation of an interface for managing resource files (e.g., sounds, images)
- build: Code to generate a single source file from current code in the repo.
- docs: Archived documentation and reference manuals found online.
- easygui: Any .lisp extensions to CCL's easygui package. Contains a few bug fixes, and an extension which provides a Cocoa view that does not respond to mouse activity
- rmcl: Any lisp code from RMCL that was either directly copied to this distro (some of the digitool GUI code could be bootstrapped after the CCL interface was defined) or rewritten for this distro (e.g., thermometer.lisp, the modal dialog implementation)
- scratch: Stray lisp code, experimental, etc. None of this is loaded for any of the tests
- submodules: Source code for actr and ccl
  - actr is newest version (as of Jun 2012). All code is same except that loader files (loader.lisp) are inside each of the tutorial folders
  - ccl is newest version (as of Jun 2012). Original image. Used to reference ccl src and run basic ccl core file
  - lisp-dev is a git repo for building a custom ccl core that I use during development. I'm using some of the src for the utilities in the GUI code, so the builds here grab some of that src.
  - rmcl is the newest version; no changes; used to reference rmcl src, and run rmcl for each of the tests.

- testing: Source code for testing the ccl code
  - testCCLDevice.lisp: Used to test basic functionality of building a Cocoa display by writing lisp code that meets the uwi.lisp interface spec
  - testTutorials.lisp: Runs ACT-R through all tutorials; checks that ACT-R can 'see' CCL's Cocoa device
  - testImages.lisp: Tests that the CCL resources.lisp file correctly manages images
  - testVotebox.lisp: Tests to run Votebox on CCL
  - testPhaser.lisp: Tests to run Phaser on CCL
- tools: Various shell scripts and tools used to manage the repo
  - merge-and-verify-driver. Used by git when I want to do a merge by hand.

3 To Run A Test

- Mount RedGiant volume
- Navigate to the Projects/mcl-migration folder on RedGiant
- Either access the data in the folder over the network (preferred), or copy the folder locally if you want to access it w/o internet.
- Download Clozure CL from the App Store and install if you don't have it yet.
- Launch Clozure CL
- Use the Cocoa listener to load one of the test files.

e.g.: Launch Clozure CL, and then load Votebox in ./testing/testVotebox.lisp

3.1 Notes about the Tests

Each test loads a bootstrap file, and then any necessary files to run that particular test. The bootstrap file (bootstrap.lisp) loads up ccl-simple-view and any shared lisp code that is needed to run a test file (particularly the unit test framework). The necessary files to run the test are defined in ./testing/file-lists/[testName]/[testName].txt. If you wanted to bootstrap the test code manually, load each lisp file in the order in that file.

4 Conclusion

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien, venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

References

[Figueredo and Wolf, 2009] Figueredo, A. J. and Wolf, P. S. A. (2009). Assortative pairing and life history strategy - a cross-cultural study. *Human Nature*, 20:317–330.