McCLIM

Gilbert Baumann

2003-07-10

LSM 2003, Metz, France

What is CLIM?

The “Silica”

Just the usual stuff:

- mediums — graphics
- sheets (windows)
- events (input events, window events, ...)
- gadgets (push-buttons, scroll-bars, ...)
- panes (vbox, hbox, ...)

but CLIM is more ...

Beyond the Silica

central features:

- output recording
- presentations
- incremental redisplay
- command loop

or:

[...] *as the next generation of command-line interfaces*
cli plus hypertext

— Daniel Barlow

Overview

presentations	incremental redisplay	formatting
	output recording	

	gadgets / panes	
media	sheets	events

— backend —

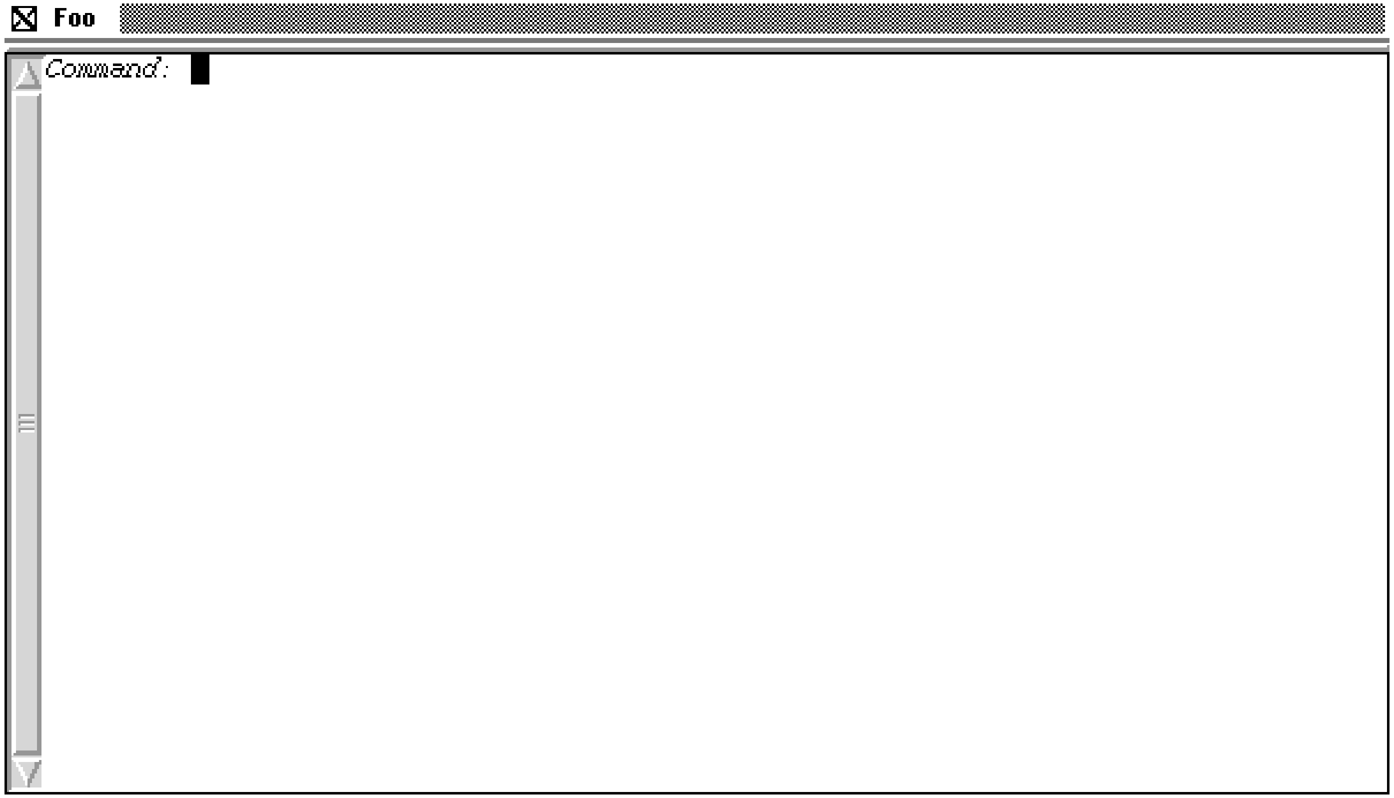
— host window system —
X11, Cocoa, OpenGL, ...

A minimal CLIM application

```
(define-application-frame foo ()
  ()
  (:panes
    (cli :interactor))
  (:layouts
    (default
       cli)))

(defun foo ()
  (run-frame-top-level (make-application-frame 'foo)))
```

A minimal CLIM application



The *cli* aspect

let's define a *Quit* command:

```
(define-foo-command com-quit ()
  (frame-exit *application-frame*))
```

The *cli* aspect

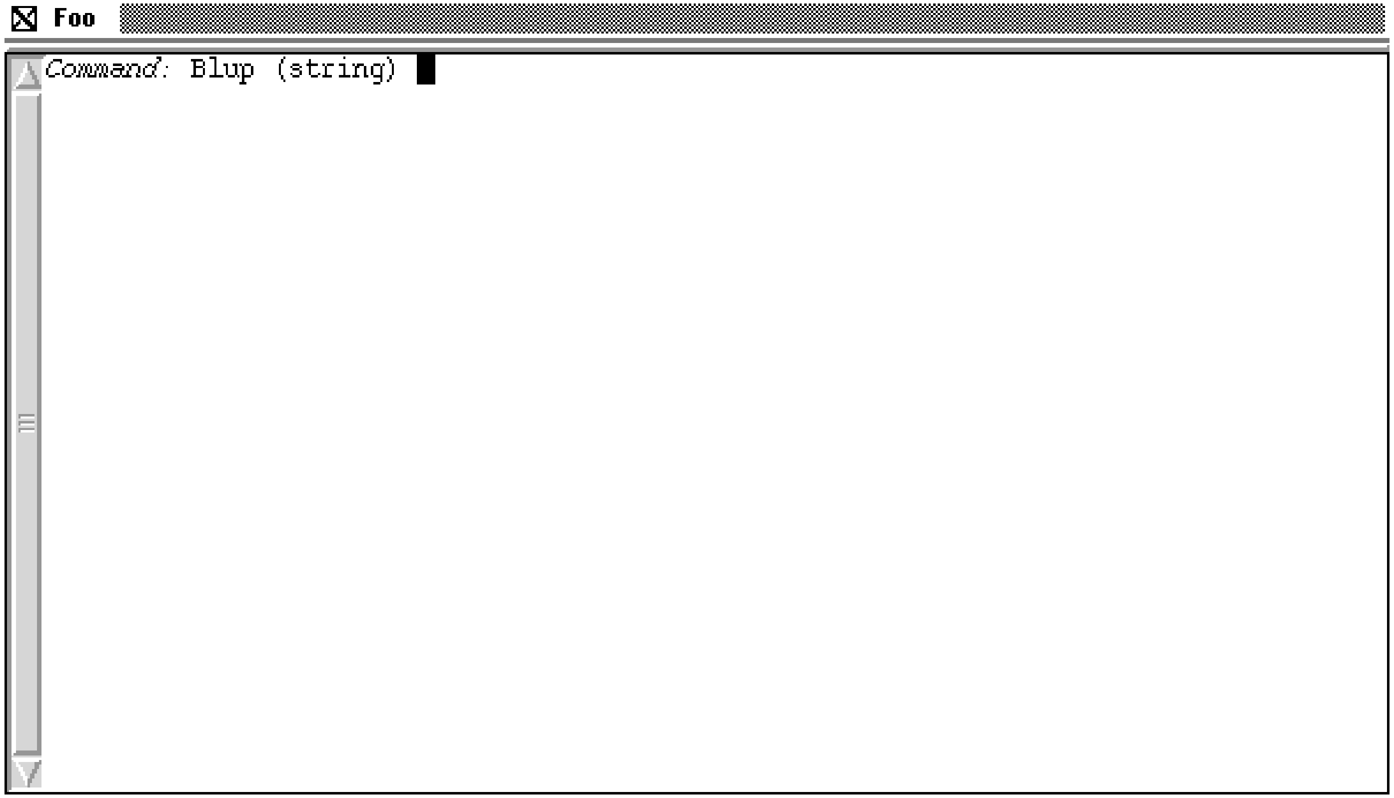
let's define a *Quit* command:

```
(define-foo-command com-quit ()
  (frame-exit *application-frame*))
```

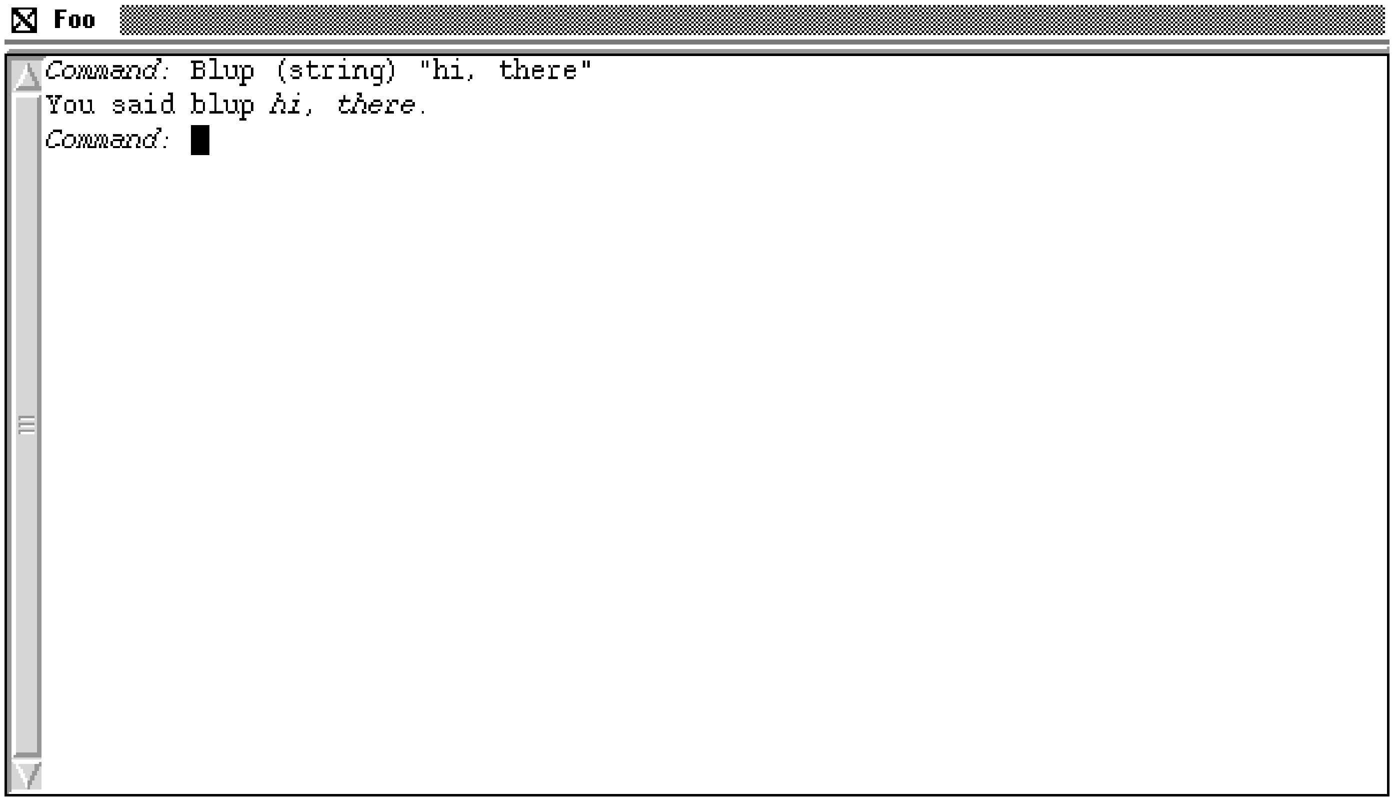
another command:

```
(define-foo-command com-blup ((x 'string))
  (format t "You said blup ")
  (with-text-face (t :italic)
    (format t "~A" x))
  (format t ".~%"))
```

A minimal CLIM application



A minimal CLIM application



A Class Browser

we want to develop a class browser, here's a sketch:

```
+-----------------------------------------+
|  class tree                             |
|                                         |
|                                         |
|                                         |
|                                         |
+-----------------------------------------+
| interactor                              |
| (command line)                          |
+-----------------------------------------+
```

A Class Browser — application frame

```
(define-application-frame class-browser ()
  ;; slots here
  ((current-class
     :initform (find-class 'sheet)
     :documentation "The class which is the root of the tree."))
  (:panes
   (class-tree :application
               :scroll-bars t
               :display-function 'display-class-tree)
   (cli :interactor) )
  (:pointer-documentation t)
  (:layouts
   (default
     (vertically ()
       class-tree
       (100 cli)))))
```

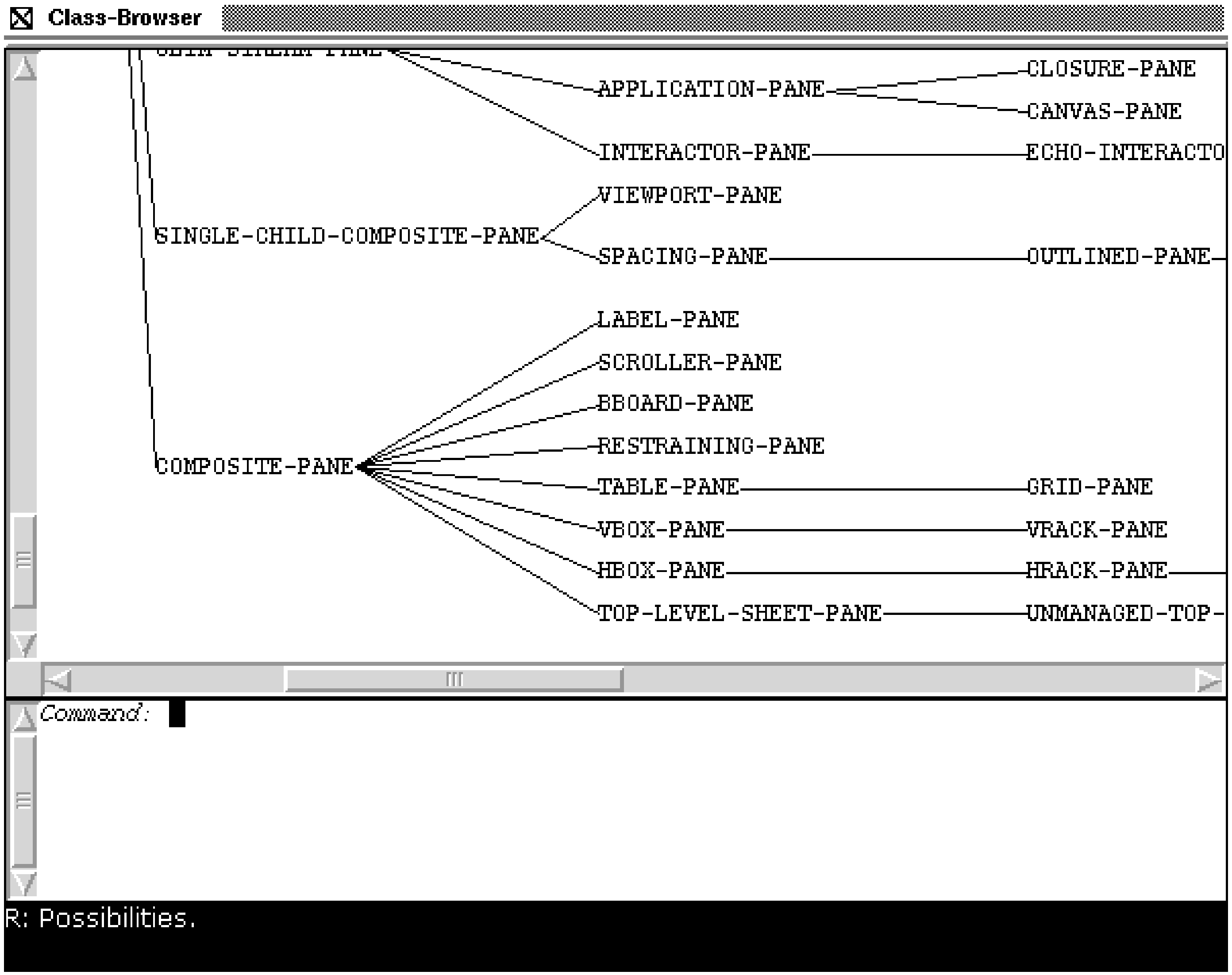
A Class Browser – display function

```
(defun display-class-tree (frame *standard-output*)
  (with-slots (current-class) frame
    (format t "current class: ~S." current-class)))
```


Class-Browser
current class: #<STANDARD-CLASS SHEET {282B3585}>.
Command:
R: Possibilities.

A Class Browser – display function

```
(defun display-class-tree (frame *standard-output*)
  (with-slots (current-class) frame
    (format-graph-from-roots
      ;; graph roots:
      (list current-class)
      ;; printing a node
      #'(lambda (class *standard-output*)
          (format t "~S" (class-name class)))
      ;; children of a node
      #'class-direct-subclasses)))
```



Class-Browser
APPLICATION-PANE
CLOSURE-PANE
CANVAS-PANE
INTERACTOR-PANE
ECHO-INTERACTO
VIEWPORT-PANE
SINGLE-CHILD-COMPOSITE-PANE
SPACING-PANE
OUTLINED-PANE
LABEL-PANE
SCROLLER-PANE
BBOARD-PANE
RESTRAINING-PANE
COMPOSITE-PANE
TABLE-PANE
GRID-PANE
VBOX-PANE
VRACK-PANE
HBOX-PANE
HRACK-PANE
TOP-LEVEL-SHEET-PANE
UNMANAGED-TOP-
Command:
R: Possibilities.

Streams

most output in CLIM goes through *streams*.

like say a tty, but supports:

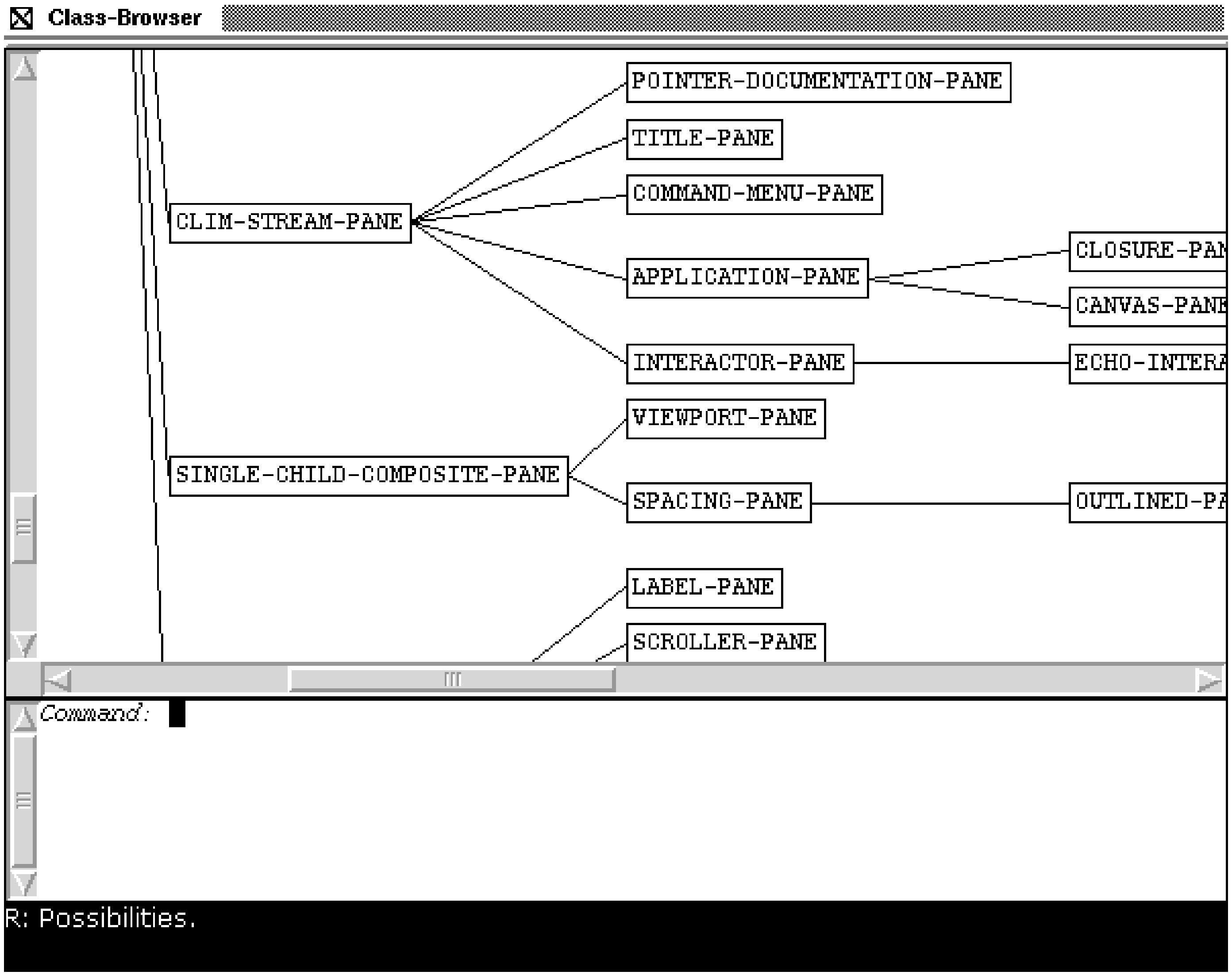
- text styles (bold, italic, fonts)
- color
- graphics
- formatting
  - formatting-table
  - formatting-graph
  - surrounding-output-with-border
  - indenting, line-breaking, ...

$\Rightarrow$ Easy to have pleasing output.

Formatting Example

In the class tree display function

```
:
:
;; printing a node
#'(lambda (class *standard-output*)
    (surrounding-output-with-border (t)
      (format t "~S" (class-name class))))
:
:
```


Class-Browser
POINTER-DOCUMENTATION-PANE
TITLE-PANE
COMMAND-MENU-PANE
CLIM-STREAM-PANE
CLOSURE-PAN
APPLICATION-PANE
CANVAS-PANE
INTERACTOR-PANE
ECHO-INTERA
VIEWPORT-PANE
SINGLE-CHILD-COMPOSITE-PANE
SPACING-PANE
OUTLINED-PA
LABEL-PANE
SCROLLER-PANE
Command:
R: Possibilities.

Formatting Example

```
(formatting-table ()
  (loop for slot in  (pcl:class-direct-slots class)
    (formatting-row ()
      (formatting-cell ()
        (format t "~S" (pcl:slot-definition-name slot)))
      (formatting-cell ()
        (format t "~{~S~^ ~}" (pcl:slot-definition-initargs slot)))
      (formatting-cell ()
        (format t "~S" (pcl:slot-definition-initform slot))))))
```

redisplay-needed	:display-time	nil
incremental-redisplay	:incremental-redisplay	nil
scroll-bars	:scroll-bars	nil
display-function	:display-function	'clim-stream-pane-default-display-function
label	:label	
text-margin	:text-margin	nil
vertical-spacing	:vertical-spacing	nil
end-of-line-action	:end-of-line-action	wrap
end-of-page-action	:end-of-line-action	scroll
output-history	:output-history	(make-instance 'standard-tree-output-history)
user-width		nil
user-min-width		nil
user-max-width		nil
user-height		nil
user-min-height		nil
user-max-height		nil
updating-record	:updating-record	nil

Streams / output recording

CLIM remembers what you draw.

- no more *redraw* methods to write
- no more redraw glitches
- redraw handling is shared across all applications
  $\Rightarrow$ optimizations done in McCLIM benefit all applications

A bit of output gets recorded $\rightarrow$ *output records*

There are also *compound* output records.

$\Rightarrow$ output records form a hierarchy.

$\Rightarrow$ *output record protocol* to manipulate and implement output records.

Streams / output recording

Example

```
(defclass backed-by-ink-or (standard-sequence-output-record)
  ((ink :initarg :ink)))

(defmethod replay-output-record :before ((record backed-by-ink-or) stream
                                         &optional region x-offset y-offset)
  (with-slots (ink) or
    (with-bouning-rectangle* (x1 y1 x2 y2) or
      (draw-rectangle* stream x1 y1 x2 y2 :ink ink :filled t))))

(defmacro backing-output-by-ink ((stream &key ink) &body body)
  `(with-new-output-record (stream 'backed-by-ink-or :ink ,ink)
     ,@body))
```

Presentations

Idea: Associate the *object* drawn with its output.

Example:

Instead of simply doing

```
(format t "~S" (class-name class))
```

do

```
(with-output-as-presentation (t class 'class)
  (format t "~S" (class-name class)))
```

CLIM now remembers that the output done:

- presents the object *class*.
- presents it as *presentation type* `class`.

Presentations (ii)

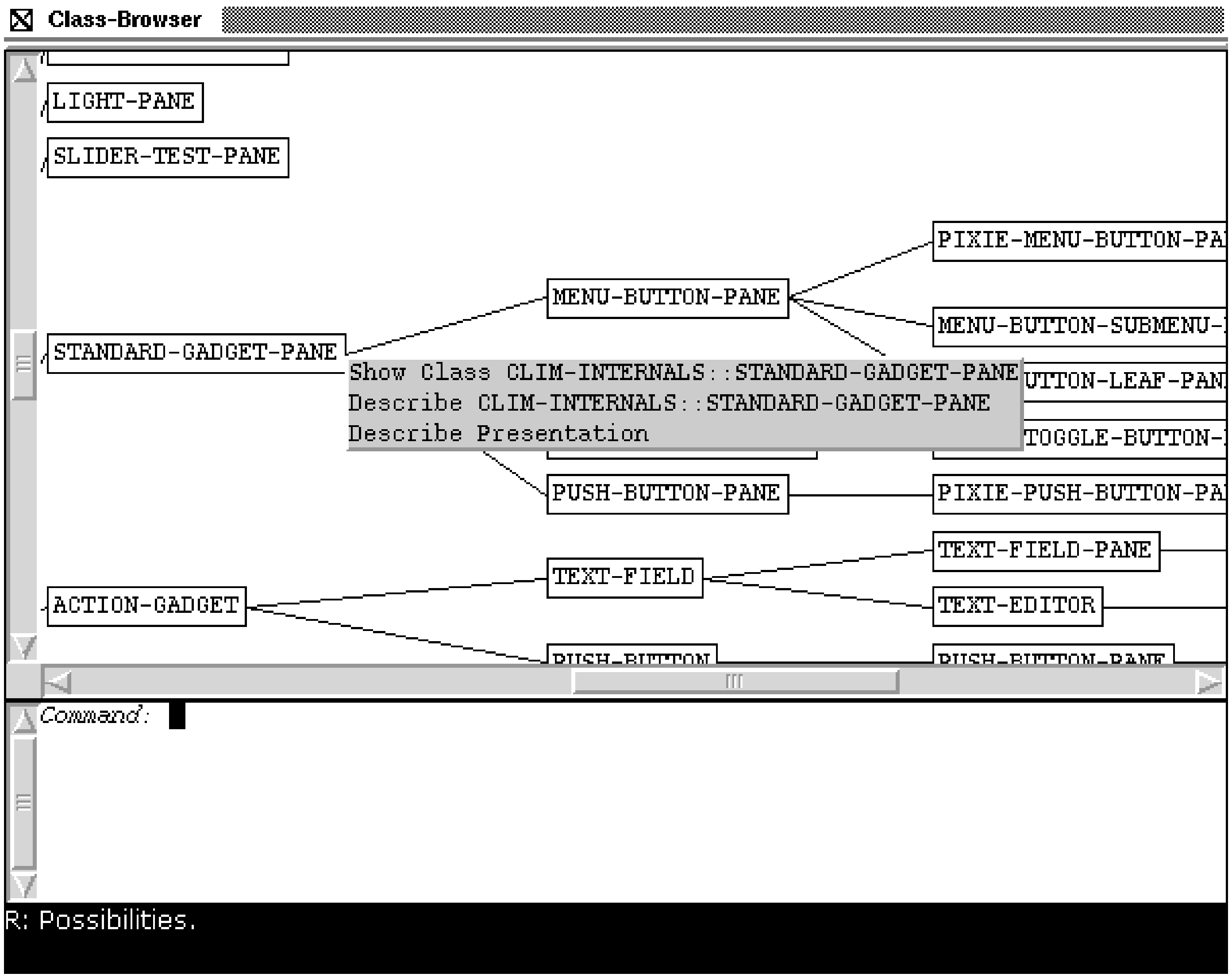
```
(with-output-as-presentation (t class 'class)
  (format t "~S" (class-name class)))
```

Effect

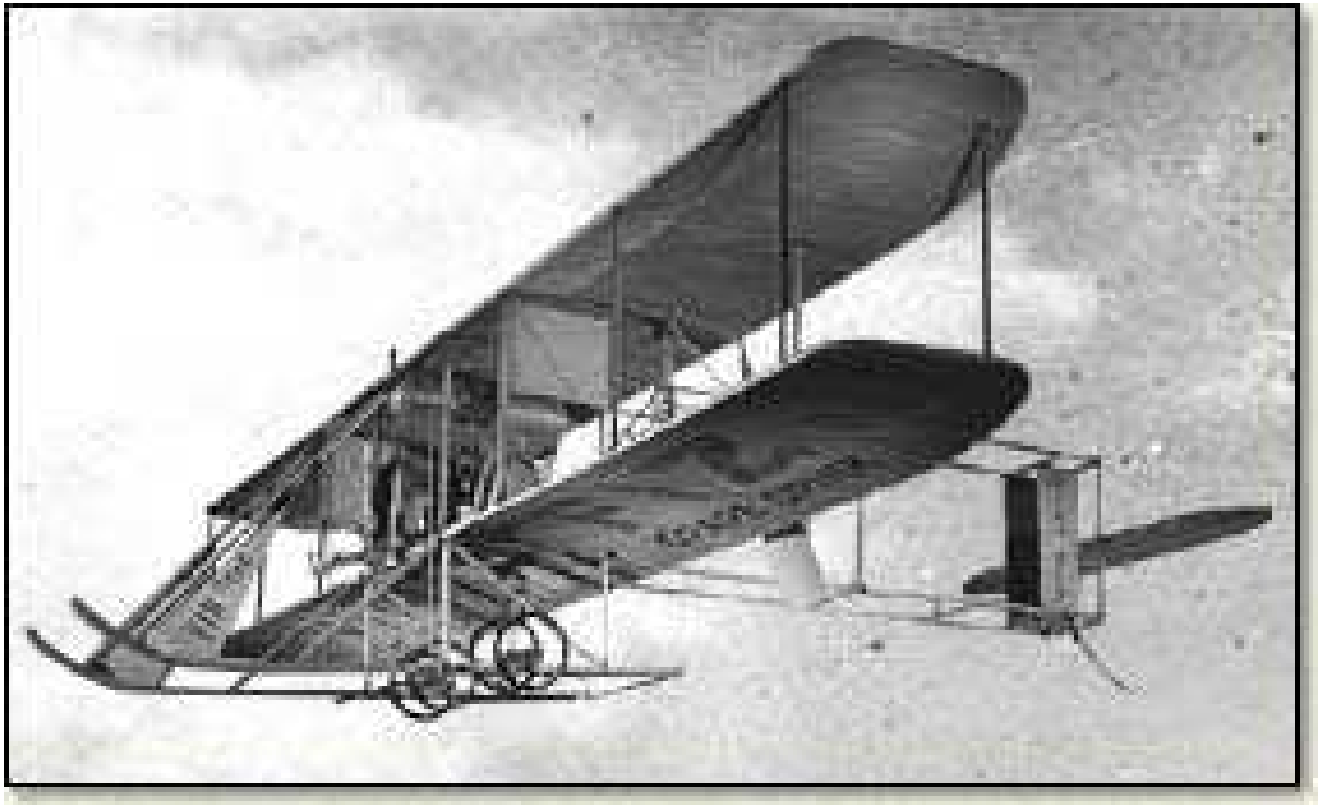
- When the application can *accept* a class as input, the output becomes mouse sensitive.
- Given some commands defined on classes (like perhaps *view sub class graph*, *edit*, ...) these automatically become available in a context menu by right-clicking on the output.

Putting it to use

```
(define-class-browser-command com-show-class ((x 'class))
   ...)
```


Class-Browser
LIGHT-PANE
SLIDER-TEST-PANE
STANDARD-GADGET-PANE
MENU-BUTTON-PANE
PIXIE-MENU-BUTTON-PA
MENU-BUTTON-SUBMENU-
UTTON-LEAF-PAN
TOGGLE-BUTTON-
Show Class CLIM-INTERNALS::STANDARD-GADGET-PANE
Describe CLIM-INTERNALS::STANDARD-GADGET-PANE
Describe Presentation
PUSH-BUTTON-PANE
PIXIE-PUSH-BUTTON-PA
TEXT-FIELD-PANE
TEXT-FIELD
ACTION-GADGET
TEXT-EDITOR
PUSH-BUTTON
PUSH-BUTTON-PANE
Command:
R: Possibilities.

The History



The History

1998	Mike McDonald started hacking horizontal approach
1998	Gilbert Baumann: started hacking too, vertical approch hacking chapter 3 til 5.
2000	Robert Strandh: picked it up, hacking chapter 7 til 13 Arthur Lemmens: initial gadgets at the LSM 2000: we merged Mike McDonald's code with Gilbert Baumann's
2001	Alexey Dejneka: table formatting, bordered output, Postscript backend Tim Moore: started hacking presentations Brian Spilsbury: i18n
2002	Tim Moore: incremental redisplay Closure ran
2003	Andy Hefner: optimizations, fixes, new ideas, Working on a Listener Tim Moore: accepting-values.

The Present

The Present

we still focus on getting the spec implemented

presentations	incremental redisplay	formatting
90%	75%	90%

output recording
100%

gadgets / panes
80%

media	sheets	events
100%	100%	100%

— backend —
75%

The Present, applications

applications

- Weird-Irc, IRC client — Jochen Schmidt
- gsharp, Score Editor — Robert Strandh
- cl-reversi, game — Kevin M. Rosenberg
- Closure, Web Browser — Gilbert Baumann

The Future

We are the McCLIM.
Resistance is futile.
Toolkits will be assimilated.

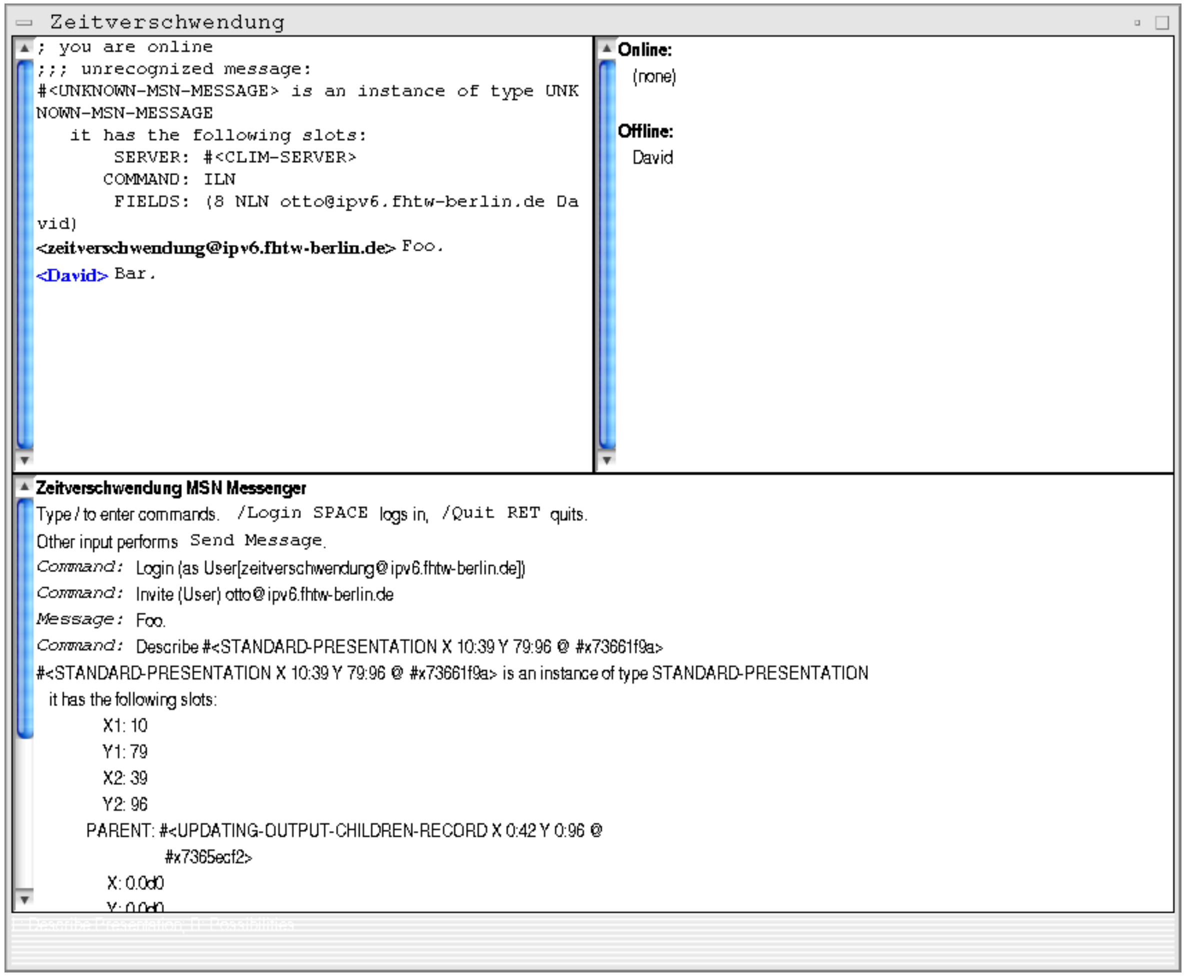
Future

future

- new backend of MacOS X (Cocoa) — Tim Moore
- Theme support — David Lichtenblau
- maybe a GTK backend — anonymous
- maybe OpenGL backend

experimental stuff

- X render extension support for anti-aliased text and translucency.
- pretty-printer integration
- Lisp listener and debugger — Andy Hefner


Zeitverschwendung
; you are online
;;; unrecognized message:
#<UNKNOWN-MSN-MESSAGE> is an instance of type UNK
NOWN-MSN-MESSAGE
it has the following slots:
SERVER: #<CLIM-SERVER>
COMMAND: ILN
FIELDS: (8 NLN otto@ipv6.fhtw-berlin.de Da
vid)
<zeitverschwendung@ipv6.fhtw-berlin.de> Foo.
<David> Bar.
Online:
(none)
Offline:
David
Zeitverschwendung MSN Messenger
Type / to enter commands. /Login SPACE logs in, /Quit RET quits.
Other input performs Send Message.
Command: Login (as User[zeitverschwendung@ipv6.fhtw-berlin.de])
Command: Invite (User) otto@ipv6.fhtw-berlin.de
Message: Foo.
Command: Describe #<STANDARD-PRESENTATION X 10:39 Y 79:96 @ #x73661f9a>
#<STANDARD-PRESENTATION X 10:39 Y 79:96 @ #x73661f9a> is an instance of type STANDARD-PRESENTATION
it has the following slots:
X1: 10
Y1: 79
X2: 39
Y2: 96
PARENT: #<UPDATING-OUTPUT-CHILDREN-RECORD X 0:42 Y 0:96 @
#x7365ecf2>
X: 0.0d0

We are the McCLIM.
Resistance is futile.
Toolkits will be assimilated.