CLIM 2.0 Tutorial, Part 2

- Formatted output
- Incremental redisplay
- Defining more complex presentation types
- The input editor
- Gadgets
- Views
- The pointer and the text cursor
- Pixmaps
- Presentation translators, revisited
- Output recording
- The windowing substrate

Formatted output

- Table formatting
- Item list formatting
- Graph formatting
- Bordered output
- Indented output
- Filled output
- Examples of formatted output

Examples of formatted output

- Simple table formatting: **squares-by-rows** and **squares-by-columns**
- Something more complex: **multiplication-tables**
- Mixing formatted output with **clim:accept**: a spreadsheet
- Using the grapher: **graph-classes**
- Filling output: **fill-gettysburg** and **fill-some-names**

Incremental redisplay

- A mostly declarative way of displaying only the output that has changed
- **clim:updating-output** and **clim:redisplay**
- Incremental redisplay with formatted output
  - The **redisplay-graph** example
- Take advantage of the tree-shaped nature of output whenever possible
  - The class browser example

clim:updating-output and **clim:redisplay**

- **clim:updating-output** is used to create an output record that can be redisplayed
  - Unique id ‘‘names’’ a piece of output
  - Cache value determines when a piece of output has changed and should be redisplayed
- **clim:redisplay** is used to redisplay an output record under new parameters

Defining more complex presentation types

- The **clim:accept** and **clim:present** methods
- Parameterized types
- Compound presentation types
- The **clim:highlight-presentation** method
  - Used when you want to customize highlighting behavior
- The **clim:presentation-refined-position-test** method
  - Used when you want to define your own sensitivity region

The **clim:accept** and **clim:present** methods

- **clim:accept** is used to read the representation of a type from the keyboard
  - There can be multiple **clim:accept** methods that specialize on the view
- **clim:present** is used to display the representation of a type
  - There can be multiple **clim:present** methods that specialize on the view
  - Where **clim:accept** methods almost always read textual input, **clim:present** methods need not display textual output
  - For textual views, **clim:accept** and **clim:present** should be inverses

Parameterized types

- Parameterizing a type restricts the membership in the type
- The **clim:presentation-typep** method
  - Used to define what objects are of the type
- The **clim:presentation-subtypep** method
  - Used to define class relationships within the type
- A simple parameterized type: **symbol-in-package**

Compound presentation types

- A compound presentation type is a type with multiple fields
- You may need to interact with the input editor
- Doing in-line prompting
    - The **:prompt** argument to **clim:accept**
    - **clim:input-editor-format**
- Customizing help with **clim:with-accept-help**

The input editor

- **clim:with-input-editing** establishes an input editing environment
- Activation gestures and delimiter gestures
- General control flow in the input editor
- Reading and replacing input
- Completion
- A compound presentation type: **output-destination**

Activation gestures and delimiter gestures

- An activation gesture tells the input editor that the input is complete
  - **clim:with-activation-gestures**
- A delimiter gesture tells the input editor when one field is complete
  - **clim:with-delimiter-gestures**

General control flow in the input editor

- The input editor uses a looping structure
  - Establish a ‘‘catch tag’’ for rescanning
  - Loop reading gestures
  - If the gesture is a ‘‘destructive’’ editing command, then rescan
- This looping structure means you need to be careful if you are maintaining some kind of a buffer of your own

Reading and replacing input

- Use **clim:read-token** to read a single, delimited token
- **clim:replace-input** and **clim:presentation-replace-input**
  - Used to replace a piece of the input buffer with new input
  - Don’t use these when the input editor is rescanning

Completion

- **clim:complete-input** completes over a set of possibilities
- Use **clim:complete-from-possibilities** to complete over a sequence of possibilities
- Use **clim:complete-from-generator** when you have a function that generate possibilities
- Use **clim:completing-from-suggestions** and **clim:suggest** for simple completion

Gadgets

- CLIM gadgets correspond exactly to the gadgets or widgets commonly available in most window system toolkits
- All CLIM gadgets obey a set of generic protocols
- A CLIM gadget is a special type of CLIM pane
- Adaptive look and feel

All CLIM gadgets obey a set of generic protocols

- The generic protocols are the same for all hosts and window system
- Protocol for dealing with events
- Protocol for controlling gadgets
- Protocol for gadget callbacks

Protocol for dealing with events

- **clim:handle-event** to handle all events
  - Pointer enter and exit events
  - Pointer buttons presses and releases
  - Key presses and releases
- **clim:handle-repaint** to display the gadget
- Writing a new gadget consists primarily of defining these methods

Protocol for controlling gadgets

- **clim:activate-gadget** and **clim:activate-gadget**
- **clim:gadget-value** and **(setf clim:gadget-value)**
- **clim:gadget-label** for labeled gadgets
- Other functions for more specific types

Protocol for gadget callbacks

- **clim:armed-callback** and **clim:disarmed-callback**
- **clim:activate-callback** for gadgets such as push buttons
- **clim:value-changed-callback** for gadgets with values
- **clim:drag-callback** for gadgets that slide

A CLIM gadget is a special type of CLIM pane

- Leaf panes
- Composite panes
- All gadgets participate in the pane layout protocol
  - **clim:compose-space** says how much space a pane takes
  - **clim:allocate-space** actually gives space to the pane

Leaf panes

- Scroll bars
- Sliders
- Push buttons
- Toggle buttons
    - Radio boxes
    - Check boxes
- List panes
- Option panes
- Text editor panes
- CLIM stream panes

Composite panes

- Horizontal and vertical box panes
  - **clim:horizontally** and **clim:vertically**
- Table panes
- Border and spacing panes
  - **clim:outlining** and **clim:spacing**
- Scroller panes
  - **clim:scrolling**

Views

- A view specifies what a presentation type looks like
- Views are a way to hook up presentation types and gadgets
- The ‘‘gadget dialog view’’ is an ‘‘indirect’’ view that uses various gadgets for the fields of a dialog based on their presentation types
    - **clim:member** fields use ‘‘one-of’’ radio boxes
    - **clim:subset** fields use ‘‘some-of’’ radion boxes
    - **clim:boolean** fields use check boxes
    - Real number fields use sliders
    - The exit boxes use push buttons

The pointer and the text cursor

- Locating and positioning the pointer or text cursor
- Changing the shape of the pointer's cursor
- **clim:tracking-pointer** is a way to manually track pointer motion and state
- **clim:dragging-output** is a packaged way to move around output records

Locating and positioning the pointer or text cursor

- The pointer
  - **clim:port-pointer**
  - **clim:stream-pointer-position**
  - **clim:stream-set-pointer-position**
  - **clim:port-modifier-state**
- The text cursor
  - **clim:stream-cursor-position**
  - **clim:stream-set-cursor-position**

Changing the shape of the pointer's cursor

- Use **clim:port-pointer** to get the pointer for the port
- Use **(setf clim:pointer-cursor)** to set the cursor directly
- Use **(setf clim:sheet-pointer-cursor)** to set the cursor for a particular sheet
- Shapes include various arrows, corners, hourglasses, crosshairs, and so forth

Pixmaps

- A pixmap is basically a repository for raw bits
- You can copy to or from a pixmap
  - **copy-window-region** example demonstrates this
- **clim:with-output-to-pixmap**
  - **do-pixmap-output** example demonstrates this

Presentation translators, revisited

- **clim:with-input-context** establishes an input context
- When does a translator match the input context?
- ‘‘Drag and drop’’ translators

When does a translator match the input context?

- When the presentation's type is a subtype of the translator's ''from'' type
- And the translator's ''to'' type is a subtype of the input context
- And the translator's gesture either matches the current modifier key state
- And the presentation's object has the type of the translator's ''from'' type (if the ''from'' type is parameterize)
- And the translator's tester returns a non-nil value

Output recording

- Creating output records
- The output recording protocol
- Writing your own output records

Creating output records

- **clim:with-new-output-record** creates a new output record and adds it to the output history
- **clim:with-output-to-output-record** creates a new output record, but does not add it to the output history
- **clim:with-output-recording-options** allows you to enable or disable both output recording and drawing
- Use **clim:replay** to replay an existing output record

The output recording protocol

- **clim:output-record-position** and **clim:output-record-set-position**
- Output record coordinates are parent-relative
  - **convert-from-relative-to-absolute-coordinates**
- **clim:output-record-parent**
- **clim:add-output-record**, **clim:delete-output-record**, and **clim:erase-output-record**
- **clim:map-over-output-records-containing-position**
- **clim:map-over-output-records-overlapping-region**
- Examples of output recording

Examples of output recording

- **with-indentation** indents the output generated by its body
- **compute-output-size** measures the output generated by its body
- **map-over-output-record-tree** calls a function on all of the records in a tree and their respective positions

Writing your own output records

- Why write your own type of output record?
- Example of a ‘‘leaf’’ output record: a ‘‘ruler’’ with tick marks
- Example of a ‘‘container’’ output record: a point-plotting application

Why write your own type of output record?

- You might write a leaf output record when you have a complex output record and you don’t want to pay the overhead of creating all kinds of little output records
- You might write your own ‘‘container’’ output record when you have application-specific knowledge that allows you to manage output records better than CLIM can

The windowing substrate

- Sheets
- Mediums
- Ports
- Frame managers
- Events

Sheets

- Basic sheets
- Mirrored sheets
- Grafts

Basic sheets

- A sheet is the fundamental unit of windowing in CLIM
- All sheets have a region
- All sheets have a transformation that locates it relative to its parent
- All sheets but the top-level graft have a parent
- Some sheets can have children
- Some sheets support doing input (event handling)
- Some sheets support doing output

Mirrored sheets

- Some sheets have a corresponding window on the display server
- Such sheets are said to be mirrored
- The window on the display server is called a mirror
- Mirrored sheets are used to implement native look-and-feel

Grafts

- A special case of a mirrored sheet
- The graft corresponds to the display device

Mediums

- A medium is an object that encapsulates drawing state
- Sheets that support output have a medium
- The medium can be permanent or temporary
- A medium is analogous to an X Window gcontext

A medium is an object that encapsulates drawing state

- Foreground and background colors
- Current drawing ink
- Current line style
- Default text style
- Current text style
- User transformation
- User clipping region

Ports

- A port is a connection to a display server
- It is responsible for basic event management

Frame managers

- The frame manager is in charge of look and feel
- It is responsible for creating panes and gadgets
- It also manages menus and dialogs

Events

- An event is an object that corresponds to the intuitive definition of an event
  - Pointer motion, button presses and releases
  - Key presses and releases
  - Window manager events
- The port generates the events that are eventually handles by the panes of the application frame that has the input focus