

An Improved eXene Text Widget

Cole Hoosier
Honors Research
Spring 2007

What is eXene?

- multi-threaded, higher-order user-interface toolkit for the X window system
- implemented in Concurrent ML (CML), a Standard ML of New Jersey (SML/NJ) library

eXene Widgets

- widgets are to eXene as Motif is to Xlib
- stateful
 - encapsulated by using threads and channels rather than classes/objects
- concurrent
 - handles input messages with separate channels for keyboard, mouse, and control messages

Why A Text Widget?

- text editing / file editing
- user input (fields in a form)
- output messages

Desired Features

- Adjustable fonts
- Adjustable colors
- Edit modes
- Scrollbars
 - can be disabled
- Text wrapping modes
- X Selection
- Adjustable rows/columns
- Restrictions on number of lines/characters

Previous Insufficiencies

- One-dimensional access to the text made several operations unintuitive
- Model was handling data that was view-specific
- Entire buffer contents were passed around for the majority of operations
- Modifications to the buffer could be very costly (likely could have been fixed)

Widget Design

- Model/View architecture
 - modeled after GTK's structure
- TextBuffer = model
 - maintains structure of text and performs modifications to the text
- TextView = view
 - displays text and handles input from the user

TextBuffer

(General Structure)

- Lines of text are stored as a doubly-linked list of nodes
- Buffer maintains references to the head and tail of the list
- Buffer also maintains character and line counts for quick lookups

TextBuffer

(Iterators and Marks)

- To interact with the buffer, text iterators and text marks are used
 - two-dimensional offsets (line, line_offset)
- Iterators store transient locations in the buffer -- lightweight
- Marks store permanent locations in the buffer -- must be updated during buffer modifications

TextBuffer

(Other Features)

- Insertion cursor is stored as a text mark
- Support for “selected” text
 - All text between the insertion mark and the selection mark
- Insert and delete operations
- Can access contents by asking for text between two iterators

TextView

(Overview)

- Makes heavy use of a TextBuffer
- TextView API
 - Methods to get/set attributes like text wrapping mode and scrollbar visibility
 - Support for yanking the TextBuffer out from underneath the view
 - Must call the redraw method

TextView

(Text Wrapping)

- Display lines are maintained as a list of text marks in the buffer
- This list will be updated on all modifications
 - If possible, updates only affect the area around the modification
 - In some cases, the entire list must be rebuilt (e.g. resizing the widget)

TextView

(Scrollbars)

- Internal to widget (more efficient)
- Maintains a list of records that store the number of lines with a given width
- This list will be updated on all modifications
 - If possible, updates only affect the area around the modification
- Scrollbars are active if the maximum line width is greater than the window width

TextView

(Eliminating Redraw Flickers)

- Tried using double buffering (2 drawables)
 - Drawing operations were asynchronous
 - Drawable copies were synchronous
 - Display was always one operation “behind”
- Implemented selective redrawing of text
 - Text between 2 iterators can be redrawn
 - Solves flickering problem for most use cases

TextView

(X Resources)

- *textView*foreground: black
- *textView*background: white
- *textView*selectBackground: gray
- *textView*selectForeground: black
- *textView*font: 9x15
- *textView*isEditable: true
- *textView*showScrollbars: true
- *textView*wrapStyle: none
- *textView*rows: 5
- *textView*columns: 20
- *textView*maxChars: 0
- *textView*maxLines: 0
- *textView*maxCharsPerLine: 0

Future Work

- Modify cursor to maintain a “floating” X position
- Support undo operations
- Extend TextView to make more use of marks
 - highlighting regions
 - marking regions as editable
- Indicator at the start of wrapped lines
- Adding indexed lookups in the TextBuffer

Demonstration & Questions

```
line4  
line5  
line6  
line7
```

Word wrapping is enabled in this window. That is why the horizontal scrollbar (normally across the bottom of the widget) is absent. Also notice that the vertical scrollbar indicates that we are scrolled down in the buffer a bit.