Scintillating! A Modernized Text Editor for VA Smalltalk

Seth Berman
Software Engineer
Instantiations, Inc.
Requirements

• Provide a modern text editor
  • Additional visual cues and styling
  • Take advantage of the latest technologies
• Minimize change to our existing system
  • Maintain full API compatibility with existing editor
  • Structural compatibility with our widget frameworks
• Must be inline with our cross-platform roadmap
  • GTK
• New capability must be made accessible to our customers
Scintilla

- Open source library for building text editors
- Provides numerous features specific to source code editing
- Initial release in 1999
- Active community
- Used by Code::Blocks, Notepad++, TortoiseSVN, others
- Cross-Platform support
Scintilla Integration in VAST

- Integrated into our Common Widgets Framework
  - New CwScintillaEditor widget provided
  - Offers full API support for Scintilla 3.3.3
  - Compatibility methods implemented to provide 100% API capability with existing legacy editor components
Direct2D/DirectWrite

- Microsoft’s technology to provide higher quality font rendering
  - Hardware-Accelerated Rendering
    - Offloads many aspects of rendering to the GPUs
  - Windows 7 and above
- How noticeable a difference? Depends on
  - Font type and style
  - Monitor type and size
  - Your eyes and/or attention to detail
Text Editor Basics

- Auto-Indent
- Keyboard Shortcuts
  - <Tab> to indent, <Shift+Tab> to unindent
  - <Alt+Up/Down> to move blocks of selected text one line at a time
- DragNDrop to relocate blocks of selected text
- Mousewheel to change font size
- Margin Area
Multiple Undo/Redo

• Finally!
• No hard limit (only memory)
• Supports coalescing
  • Combine contiguous insertions and deletions into single undo operations
• APIs for user-defined coalescing
Code Completion

- Scintilla offers a code completion popup and API
  - We hooked it up to our code completion engine
- Users have the option to select which popup they prefer
  - Basic (Minimal styling capability)
  - Extended (Maximum styling capability)
  - Scintilla (Somewhere in the middle)
Syntax Color Highlighting

- Scintilla Lexer defines how a specified range of text is to be colored
  - Has lexer support for 80+ languages
  - Smalltalk is one of them, but it was too simplistic
  - Provides hooks to allow us to write our own custom lexer in Smalltalk (Container-Defined Styling)

- Container-Defined Styling
  - Scintilla specifies what needs to be styled via events
  - VAST’s custom styler defines how the character range is to be styled
VAST Custom Styler

- Comes in 2 flavors
  - Method styler to style text in browsers and debuggers
    - Optimized to style methods
  - Snippet styler to style text in inspectors and workspaces
    - Optimized to style snippets of code
    - Adds some fuzzy logic rules
    - Now we can offer color in our inspectors and workspaces

- Styler uses a custom token scanner instead of parse trees
  - Enables real-time coloring
  - Much more flexible than our previous parse-tree implementation
Bracket Highlighting

- Stylization used to indicate matching characters for ( ) [[]]{}
- Separate style used to highlight unmatched characters

\[\uparrow (\text{self abrSender: 2}) \mid \text{printString} \]
\[\uparrow (\text{self abrSender: 2}) \textcolor{red}{\uparrow} \text{printString} \]
Bracket Highlighting Cont...

- Bracket Highlighting is configurable
Smart Highlighting

• Adding stylization to a selected word and any matching word in the source
• Useful for seeing local variable usage
• Adds extra decoration to highlighting browsers
• Added logic to handle block argument highlighting
• Stylization applied behind text so syntax color highlighting shows through
Smart Highlighting Cont...

• Where is the argument, *anInteger*, used?
Smart Highlighting Cont...

- Identify where variables are declared with a glance
Smart Highlighting Cont...

• Block-args included...

```smalltalk
| temp |
[:element :index |
  element = oldObject ifTrue: [
    sorted ifTrue: [
      elements
      replaceFrom: index to: size - 1 with: elements startingAt: index + 1;
      at: size put: nil.
      size := size - 1.
    ] ifFalse: [
      temp := elements at: size.
      elements at: size put: nil.
      size := size - 1.
      index <= size ifTrue: [
        sortBlock == nil ifTrue: [
          (self defaultBubbleUpFrom: index using: temp) = index ifTrue: [
            self defaultBubbleDownFrom: index to: size using: temp].
        ] ifFalse: [
          (self bubbleUpFrom: index using: temp) = index ifTrue: [
            self bubbleDownFrom: index to: size using: temp].
        ].
    ]]
  ]
```
Smart Highlighting Cont...

- Smart Highlighting is configurable
Line Numbers

- Available in Browser/Debuggers/Workspaces/Inspectors
- Line Number margins are dynamically sized
- Configurable across different window types
Breakpoint Management

- Persistent Breakpoint margin
- New Breakpoint icons
- Multiple Breakpoints / Line support
- Stylizes Breakpoint regions
Debugger Call Tips

- Hover mouse over variables and globals displays a calltip showing it’s value
- Clicking a calltip will bring up an inspector
- Long calltips are formatted to single-line
Debugger Call Tips Cont...

- Call Tips are configurable

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Popup Delay (ms)</td>
<td>500</td>
</tr>
<tr>
<td>Enable Debugger Call Tips</td>
<td>True</td>
</tr>
<tr>
<td>Background Color</td>
<td>Default</td>
</tr>
<tr>
<td>Foreground Color</td>
<td>Default</td>
</tr>
</tbody>
</table>

Copyright © 2013 Instantiations, Inc.
Error/Warning/Info Indicators

• Stylized Squiggle lines underneath text to indicate
  • Fatal Errors
  • Errors
  • Warnings
  • Info
• Calltips provide information about the indicator
• Incremental compiler runs in the background to identify issues in real-time
  • Collects information from parse trees and the styler
Error/Warning/Info Indicators Cont...

- Instantly identify misspelled variables

```smalltalk
occurrencesOf: anObject

"Answer an Integer indicating how many of the receiver's elements
are equivalent to anObject."

| occurrences |
occurrences := 0 .
self do: [:element | undefined
    element = anObjects ifTrue: [ occurrences := occurrences + 1]].
^occurrences
```
Error/Warning/Info Indicators Cont...

- Identify methods not implemented

```smalltalk
deprecated: explanationString in: versionString
    "Process a deprecation warning associated with the sender. @explanationString
    sender was deprecated and @versionString identifies the product version.
    deprecated."

Deprecation
    method: (Processor activeProcess methodAtFrames: 1)
    explanation: explanationString
    in: versionString
```

Copyright © 2013 Instantiations, Inc.
Error/Warning/Info Indicators Cont...

- Some more examples...

```smalltalk
testByteArray

<table>
<thead>
<tr>
<th>bytes</th>
<th>8-bit integer expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>bytes := $[1234]</td>
<td></td>
</tr>
</tbody>
</table>

deprecated: explanationString

"Process a deprecation warning associated with the sender.
sender was deprecated."

Can't assign to 'explanationString'

explanationString := 'Hello Smalltalk!'.
Deprecation
method: (Processor activeProcess methodAtFrame: 1)
explanation: explanationString
in: ''
```
Error/Warning/Info Indicators Cont...

- Indicators are configurable
What’s Next?

• Smart variable rename
  • Select a variable and you can rename all occurrences in the method at the same time

• Lint warnings
  • variable never used
  • variable read before written
  • variable written but never read
  • ... and more

• Comment folding

• ... and more
Questions?