Terminal Widgets

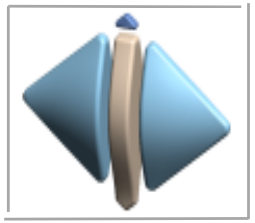
Ernest Micklei

emicklei@philemonworks.com

Amersfoort, The Netherlands

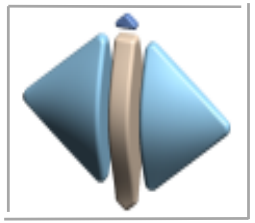
ESUG 10th

Douai, August 29 2002



Motivation

- RSI (Repetitive Strain Injuiry)
- Fast access to simple programs (=objects)
- Explore mixed interfaces (pixels-ASCII)
- Simplest UI possible
- client-server
- module for SmallScript

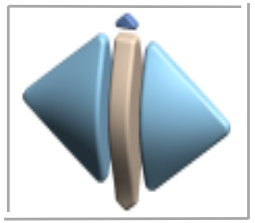


emicklei@philemonworks.com

Design

- A widget displays an aspect of an object in a defined region of a window
- A region is defined by a rectangular area of characters organized in rows and columns
- Keyboard events are handled by the controller of the widget (MVC)

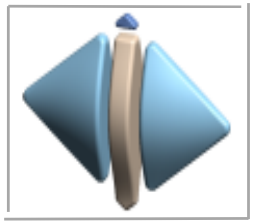
emicklei@philemonworks.com



Design (II)

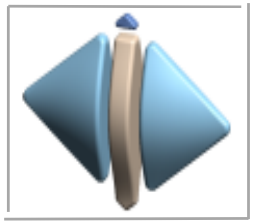
- TerminalForm is a UI component that displays a grid of ASCII characters
- For displaying, widgets map their contents to characters of that grid

emicklei@philemonworks.com



Design (III)

- **Core classes are:**
  - TerminalCharacter
  - TerminalGrid
  - TerminalWidget
  - TerminalController
- **Others**
  - CompositeWidget, Appearance, Form



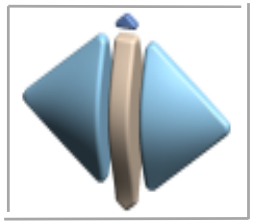
emicklei@philemonworks.com

Coordinate system

access by Points: row@column

S	M	A	L	L	T	A	L	K

(grid at: (1@2)) = $M

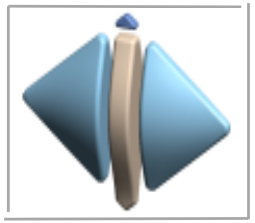


emicklei@philemonworks.com

Grid

- **Terminal screen is showing a matrix of graphical characters organized in rows and columns.**
- **TerminalCharacter**
- **TerminalGrid**

A	A	A	A
A	A	A	A
A	A	A	A
A	A	A	A

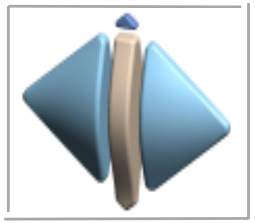


emicklei@philemonworks.com

Grid

- **holds collection of TerminalCharacter**
- **read/write strings to grid (matrix)**
- **for display only**
  - Terminal OS-window holds grid

emicklei@philemonworks.com



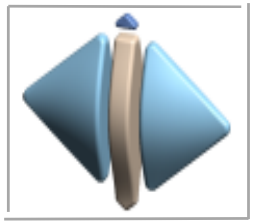
Text

```
| window txt |

"window"
window := TerminalWidget textClass in: (1@1 corner: 24@80).

"build"
txt := TerminalWidget textClass in: (1@1 corner: 8@20).
txt string: 'This is a Smalltalk terminal application'.
window add: txt.

"open"
Terminal show: window
```



emicklei@philemonworks.com

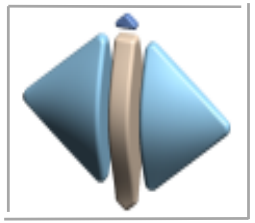
List

```
| window list |

"window"
window := TerminalWidget windowClass in: (1@1 corner: 24@80).

"build"
list := TerminalWidget listClass in: (1@1 corner: 8@20).
list items: #('ESUG' '10th' 'Douai' 'France' ).
window add: list.

"open"
Terminal show: window
```



emicklei@philemonworks.com

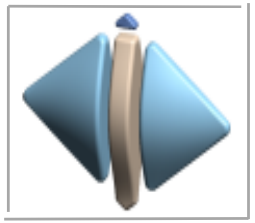
Image

```
| window list |

"window"
window := TerminalWidget windowClass in: (1@1 corner: 24@80).

"build"
image := TerminalWidget imageClass in: (1@1 corner: 8@20).
image bitmap: (Bitmap fromFile: 'splash.bmp').
window add: image.

"open"
Terminal show: window
```



emicklei@philemonworks.com

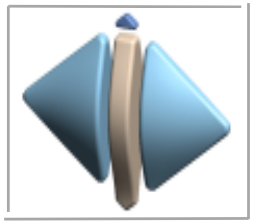
Menu

```
| window list |

"window"
window := TerminalWidget windowClass in: (1@1 corner: 24@80).

"build"
menu := TerminalWidget menuClass in: (1@1 corner: 4@20).
menu add: '1. Stockrates' key: $1 do: [self startStockrateView].
menu add: '2. Accounts' key: $2 do: [self startAccountView].
menu addLine.
menu add: '3. Invoices' key: $3 do: [self startInvoicesView].
window add: menu.

"open"
Terminal show: window
```



emicklei@philemonworks.com

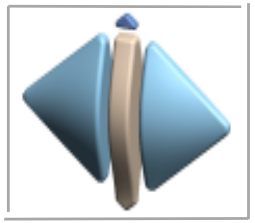
MenuBar

```
| window list |

"window"
window := TerminalWidget windowClass in: (1@1 corner: 24@80).

"build"
menuBar := TerminalWidget menuBarClass in: (1@1 corner: 1@80).
menuBar add: 'File' key: $f menu: self fileMenu.
menuBar add: 'Edit' key: $o menu: self editMenu.
menuBar add: 'Help' key: $h menu: self helpMenu.
window add: menuBar.

"open"
Terminal show: window
```

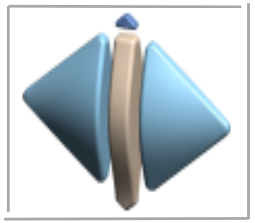


emicklei@philemonworks.com

Terminal

- **is top container for terminal windows**
- **can show, hide and (will in future) stack windows**

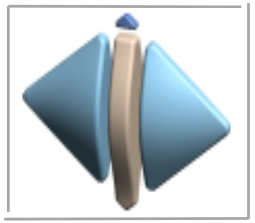
- **implementation is dialect specific**
  - but requires minimal behavior



emicklei@philemonworks.com

Widget

- widgets claim a region of the screen
- has a controller to handle keyboard events
- has a model for storing its domain value
- has an appearance
- is the "V" in MVC
- when:send:to:, broadcast: (AOS)

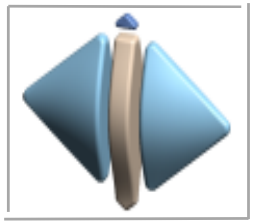


emicklei@philemonworks.com

Character

- displays a single character in some (fixed) font
- can display decoration (border lines)
- has an appearance

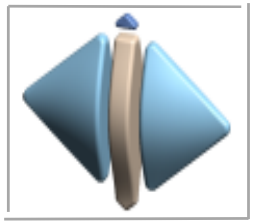
emicklei@philemonworks.com



Implementation challenges

- **character (re)display**
- **inputController**
- **appearance**

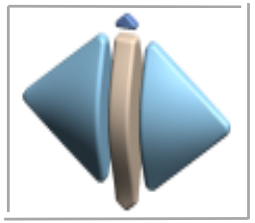
emicklei@philemonworks.com



Character (re)display

- **observation: painting complete screen is too expensive**
- **damage rectangles intersection is too expensive**
- **widget knows which characters to update**

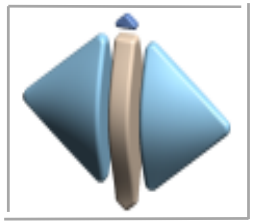
- **but, does not help with overlapping OS-windows**
  - may need double buffering



emicklei@philemonworks.com

InputController

- **like the VW ParagraphEditor, but...**
- **break text into lines, localizing updates**
- **replace CRLF with CR**
  - every character takes up one space
- **cursor can be beyond text**
- **cursor can be on CR position**
- **adopt color emphasis**
- **scrolling (vertically only)**
- **no TAB**

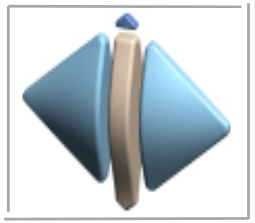


emicklei@philemonworks.com

Appearance

- window appearance
- widget appearance
- character appearance
- properties "inherited" by composition hierarchy
- modifiable at each "level"

emicklei@philemonworks.com



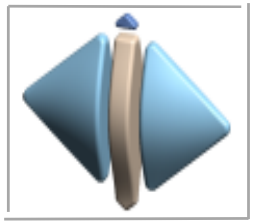
Appearance hierarchy

TerminalObjectAppearance (abstract superclass)
foreground background selectionForeground selectionBackground

TerminalWindowAppearance
font fontName characterWidth characterHeight

TerminalWidgetAppearance
windowAppearance borderColor showBorderOnFocus

TerminalCharacterAppearance
widgetAppearance



emicklei@philemonworks.com

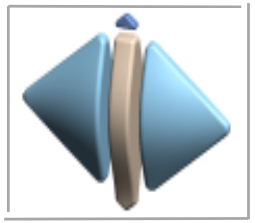
WidgetApperance

- **finds colors from parent appearance**
  - a WindowAppearance
- **but can override values by replacing nil-values**
- **example:**

```
TerminalObjetAppearance>>background

    ^background isNil
        ifTrue:[self hasParent
                ifTrue:[nil]
                ifFalse:[self parentAppearance background]]
        ifFalse:[background]
```

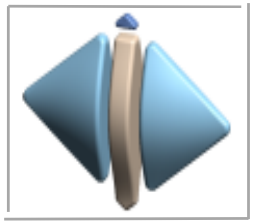
emicklei@philemonworks.com



CharacterAppearance

- **initially meant for per-character coloring**
- **became obsolete when introducing EmphasizedText**
  - VA rewrite of VW Text

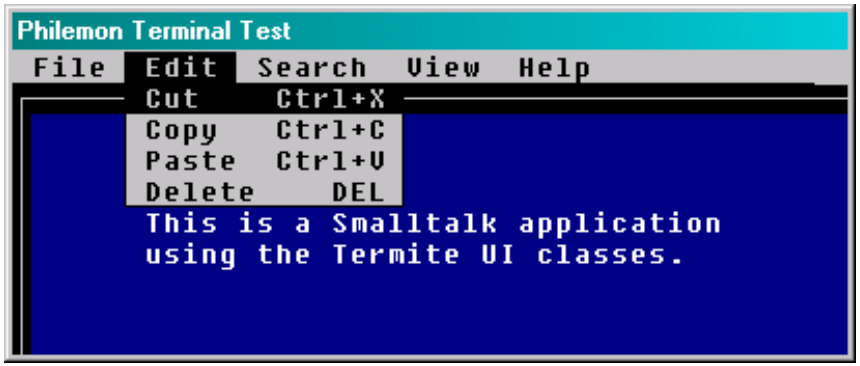
- 'ESUG' asEmphasizedText
  from: 1
  to: 2
  setForeground: Color yellow



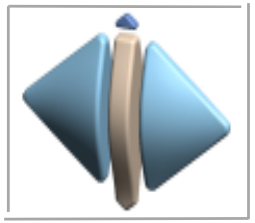
emicklei@philemonworks.com

(dos) TextEditor

- **demo**

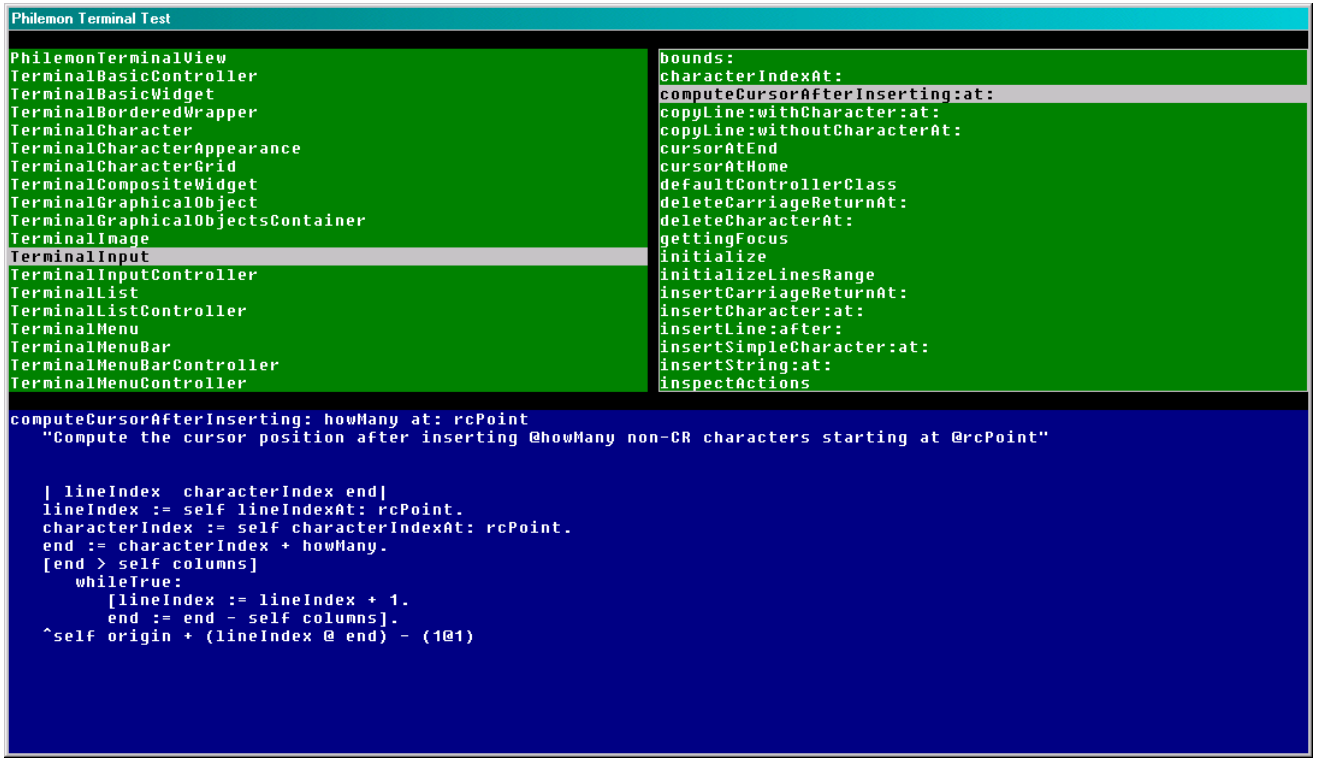


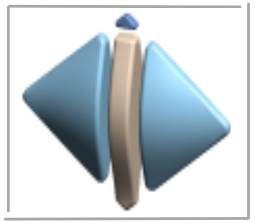
emicklei@philemonworks.com



(yet another) Class Browser

- **demo**





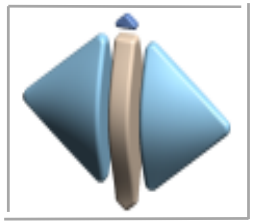
emicklei@philemonworks.com

Smalltalk Objects Shell

- shell interface to an almost empty object space (image)
- demo

```
Philemon Terminal Test
Smalltalk Objects Shell v1.0, by Ernest M. Micklei, PhilemonWorks.com
smalltalk>ls
commands     <Namespace>
tools        <Namespace>
smalltalk>cd commands
an ObjectDirectoryCommand
smalltalk::commands>ls
background   <BlockCommand>
cd           <ObjectDirectoryCommand>
clear        <BlockCommand>
del          <ObjectDirectoryCommand>
dir          <BlockCommand>
eval         <BlockCommand>
exit         <BlockCommand>
foreground   <BlockCommand>
help         <BlockCommand>
load         <BlockCommand>
ls           <BlockCommand>
mkdir        <ObjectDirectoryCommand>
move         <ObjectDirectoryCommand>
new          <BlockCommand>
run          <BlockCommand>
self         <CommandLineInterpreter>
var          <ObjectDirectoryCommand>

smalltalk::commands>
```

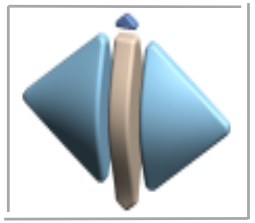


emicklei@philemonworks.com

What's next to do

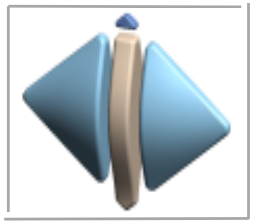
- implementation issues
- design issues
- fit of purpose issues
- exploring the "Smalltalk Objects Shell"

emicklei@philemonworks.com



What's next to implement

- **rewrite InputController**
  - got tips from Samuel Shuster
- **finish port from VAST to SmallScript**
  - put it on the web
- **text selection for InputController**
  - cut,copy,paste
- **handle OS-paints**
- **build from Pollock XML?**

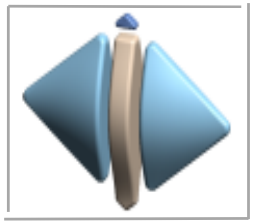


emicklei@philemonworks.com

How to port

- **display methods**
  - draw a line
  - draw a String character
  - set colors
- **dispatch keyboard events**
- **handle focus events**
- **have a window to paint on**

- **(almost) done for SmallScript**

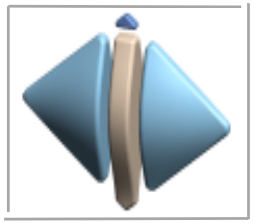


emicklei@philemonworks.com

Design issues

- **how to design characterbased applications**
  - and still be object-oriented
- **what do I need for client-server architecture**
  - maybe TELNET is fine, why bother
- **missing widgets? buttons,dropdowns**
  - do I really want to mimic Windows

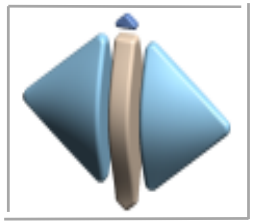
emicklei@philemonworks.com



Shell for Objects

- **motivation for porting to "imageless" SmallScript**
- **use objects in stead of just (fat)executable**

- **think about what objects are really powerful but do not need a UI**
  - graphical image processing
  - 3D language generators

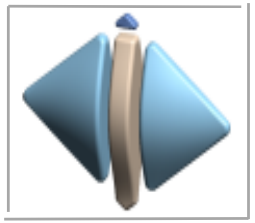


emicklei@philemonworks.com

Useful?

- Re-inventing wheels? (curses)
- Is mixing character-based and full graphics only just "yes we can do-it" ?
- Will performance be acceptable ?

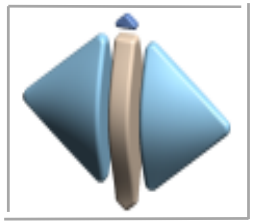
emicklei@philemonworks.com



About the results

- Can be done (what else would you expect)
- Might be useful
- Mixing with other widgets not explored
- Highly portable (to other dialects)
- Mouseless apps

emicklei@philemonworks.com



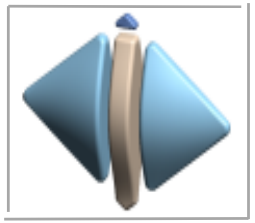
Example SmallScript

Reference module: PhilemonTerminalView.

```
[
        | window | := Terminal windowClass in:(1@1 corner: 20@40).
        | text | := Terminal textClass in:(2@2 corner: 19@39).
        window add: text.
        Terminal show: window
]
```

".dll = 56kB"

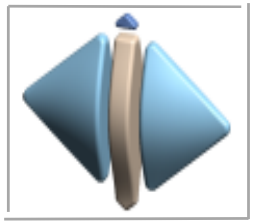
emicklei@philemonworks.com



Thanks

download @

http://www.philemonworks.com



emicklei@philemonworks.com