Basic Aspects of Squeak and
the Smalltalk-80 Programming Language

by Wolfgang Kreutzer
Author: Wolfgang Kreutzer

- Diploma and PhD in Applied Computer Science in Germany.
- Teach computer science at the University of Canterbury in Christchurch, New Zealand.
- Research in the areas of discrete event simulation and the use of visual representation and animation in programming languages and multimedia systems.
- Also in Sweden, Canada and USA.
The beginning (early 70s)

• Alan Kay, main person behind Smalltalk.
• He got an idea, that everyone should have a ”Dynabook”, Children, doctors, teachers, architects, business people.
• Programming skills was expected, to use the dynabook to it’s full potential.
• Existing languages was not suitable, the Lisp family did not have the necessary data encapsulation.
Then what happened?

- Logo, an interactive and graphical environment for teaching kids to code.
- Smalltalk adopted a lot of Logo’s ideas.
- The complexity of procedural languages is increased exponentially by program length.
- Kay wanted extensible layer based design.
- Got this by inheritance and encapsulation.
- Result: Smalltalk - first language based entirely on objects and message passing.
72 through 80

• First implementation; 1000 lines of Basic.
• Lots of user interface experiments was conducted in 1974, and a mouse driven program editor was made.
• In 1976 the idea of inheritance was added.
• Was used daily by 20+ people at Xerox Parc, for 4 years. While gaining interest outside Xerox.
• Finally Smalltalk-80 was publicly released.
Overview of Smalltalk/Squeak

• Purely object oriented, with classes, inheritance, methods and messages.
• A Smalltalk program is composed of chains of messages sent to objects.
• Lots of predefined (but mutable) classes.
• Basic functionality is rooted in the Model, View, Controller metaphor.
Screenshot
Context Menu

- find class...
- recent classes...
- browse all
- printOut
- reorganize
- update
- add item...
- rename...
- remove

- browse
class
printOut
findOut

- hierarchy
definition
comment

- spawn
dependency
spawn protocol

- inst var refs...
- inst var def...
- class var refs...
- class vars...
- class refs
- rename...
- remove
- find method...

- browse
class
printOut
fileOut

- senders
implementors
implementors of...
versions

- inst var refs...
- inst var def...
- class var refs...
- class variables
- class refs
- remove
- more...

System Transcript

hello world

- edit label
- choose color...
- two-tone/full color
- move
- frame
- full screen
- collapse
- close

2 + 2 = 4

find...(f)
find again (g)
set search string (h)
do again (i)
undo (z)
copy (c)
cut (x)
paste (y)
do it (d)
print it (p)
inspect it (i)
accept (s)
cancel (l)
show bytecodes
more...
Syntactical constructs

- "a comment"  "says it all"
- 42  "a number"
- 'Foo'  "a string"
- #bar  "a symbolic constant"
- #("array' #with 4 'elements')  "says it all"
- GlobalIdentifier  "initial uppercase letter"
- localIdentifier  "initial lowercase letter"
- |temp1 tmpVar|  "declaration of vars"
Examples of messaging

• Smalltalk at: #key put: SomeClass new.
• 1 + 2.
• |fozzie| fozzie := Bear new.
• fozzie center; show; dance.
Control structures

```plaintext
|count|
count := 0.
[count < 3]
  whileTrue:
    [(count odd) ifTrue: [Transcript show: 'ODD']
      ifFalse: [Transcript show: 'EVEN'].
    count := count + 1.
    Transcript cr].

|canon|
canon := [:singers :song |
  singers
    do: [:voice | Transcript show: voice asString,
      '.', song; cr]].

canon value: #(kermit jasper fred) value: 'croak'
```
Class declaration

Class: **Monster**
Superclass: Object
Category: Sesame Street
Instance variables: colour tummy

actions
  eat: someItem
  self tummy add: someItem

queries
  isEmpty
  ^ self tummy isNil
Opinions

- Far too much explanation of the graphical environment.
- Far too little technical explanation of the language.