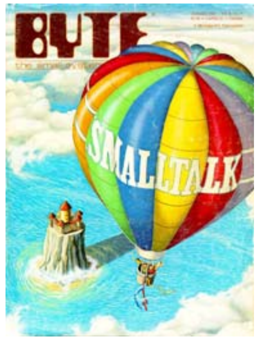
# Self – The Power of Simplicity

A short tutorial on a prototype based programming language

## Outline

- The Self Object Model
  - The OO paradigm revisited
  - Semantics of messages
  - Prototype based languages
  - Performace issues
- The Self Implementation
  - Available Ports
  - The Self GUI
  - Syntax of Self
  - Goodies
- References

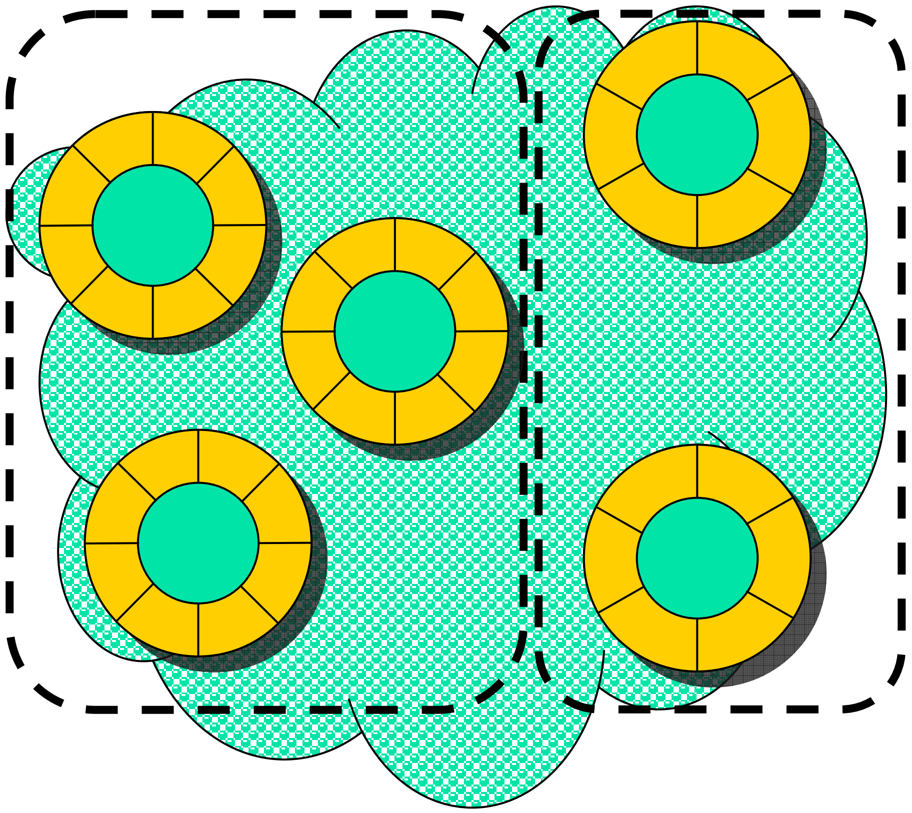
# The OO paradigm revisited



© Byte Mag.

- Byte 8/81:
  - "A language should be designed around a powerful metaphor that can be uniformly applied in all areas" [D. Ingalls]
  - "Programming Smalltalk is similar to the process of human interaction" [C. Morgan]
  - "Instead of a bit-grinding processor raping and plundering data structures, we have a universe of well-behaved objects that courteously ask each other to carry out their various desires." [D. Ingalls]
- Smalltalk metaphor: object, message, class, instance, method, (variable) [Smalltalk-80, The Language]
- What about an even simpler, more powerful metaphor?

## Object-Oriented Analysis and Design



- Identify Objects
- Classify Objects
- Identify Attributes
- Implement Methods

Gordon, show foil!

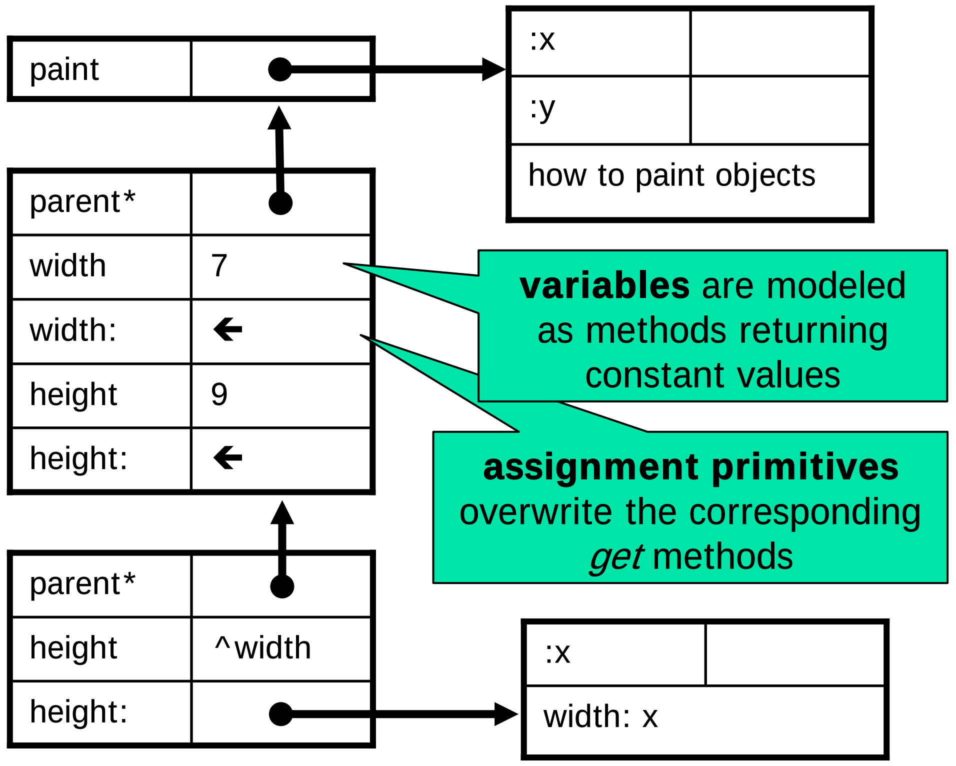
# The Design principles of Self

- Messages-at-the-bottom
  - consequently base information representation and manipulation on objects communicating by messages
  - put all optimizations into the compiler
- Occam’s razor
  - leave out everything that dilutes the paradigm
  - strip classes, variables, numbers, and control structures from the language kernel

similar to LISP

[Ungar, Smith. SELF: The Power of Simplicity. Lisp Journal, 4/91]

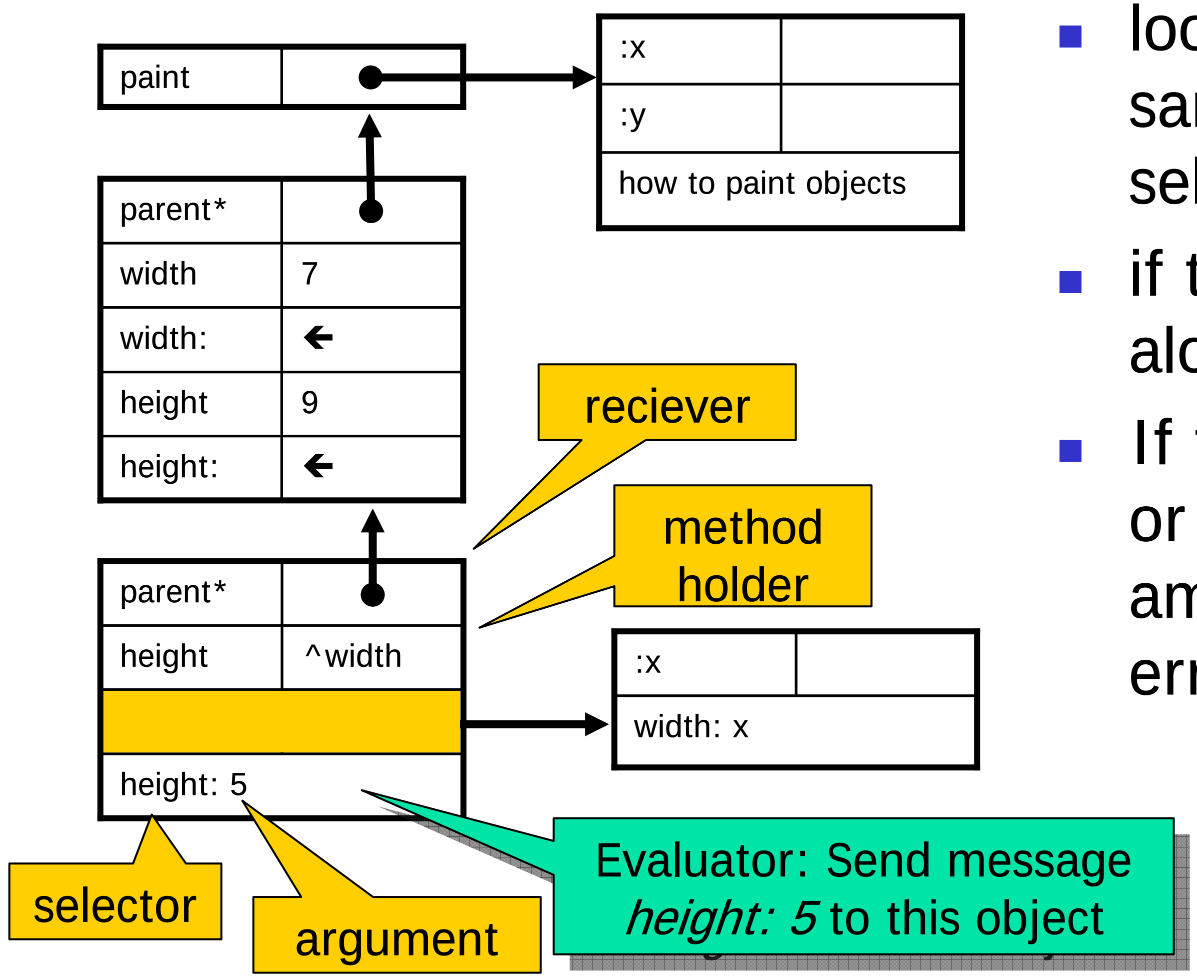
## The Self Object-Model



paint
:x
:y
how to paint objects
parent*
width
7
width:
height
9
height:
**variables** are modeled as methods returning constant values
**assignment primitives** overwrite the corresponding *get* methods
parent*
height
^width
height:
:x
width: x

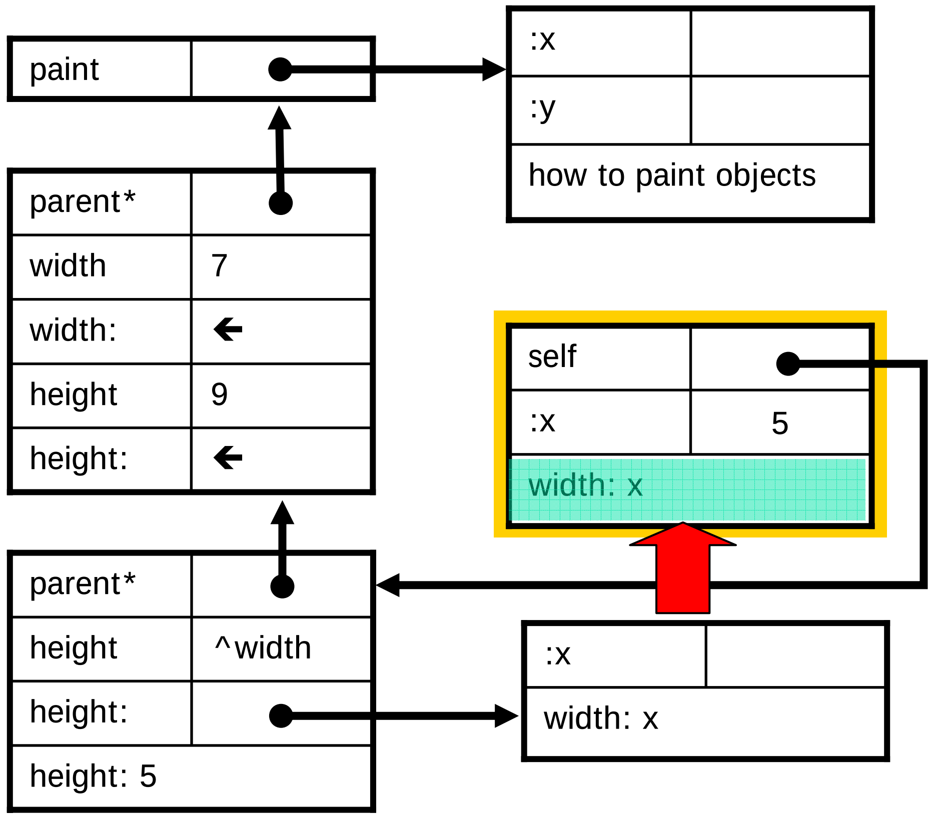
- Every object contains a collection of slots
- Each slot has a name and an object
- Every slot can be marked as parent slot
- A non-method object simply returns itself when invoked as a method
- A method objects contains a piece of code that’s executed on invocation
- Slots of method objects can be marked as argument slots

# Semantics of Messages 1. Lookup



- look for a slot with the same name as the selector in the object
- if there is none, search along all parent slots
- If there is no such slot, or the slot is ambiguous, generate an error

# Semantics of Messages
# 2. Activation



- clone method object to obtain activation record
- fill in self and argument slots
- evaluate code in context of activation object

# Prototype-based Languages

- Blend state and behavior

computing instead of storing and vice versa

- Support singleton objects
- Creation of new objects by cloning, not by instantiation

simpler

- Uniform metaphor even for activation
- Simpler and more expressive inheritance scheme
- No infinite meta-regress at conceptual level

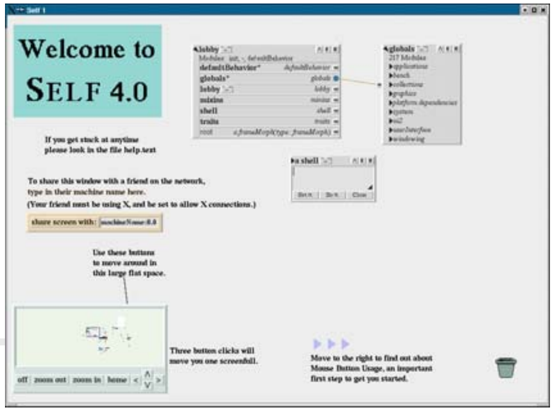
**Object** isA **Class** isA **MetaClass** isA ...

## Performance Issues

- Wait a minute: Is this not horribly inefficient?
- That depends on your compiler.
- Compiler recovers variables and data types transparently from code

maps

- Self compiler can undo all optimizations transparently: easy to debug programs
- **Benefit**: Get maximum flexibility with the full performance



If you get stuck at anytime please look in the file help.text

To share this window with a friend on the network, type in their machine name here. (Your friend must be using X, and be set to allow X connections.)

Use these buttons to move around in this large flat space.

Three button clicks will move you one screenfull.

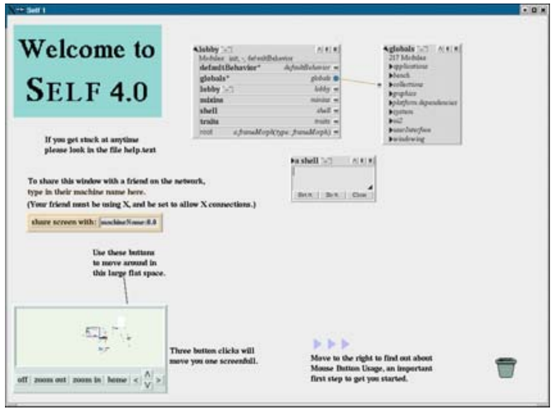
Move to the right to find out about Mouse Button Usage, an important first step to get you started.

## Available Ports

- **Solaris**: Original Implementation by Stanford/Sun (Ungar, Smith, et al.), NIC & SIC compilers fully functional (http://research.sun.com/self/)
- **Linux**: Based on Solaris port, by Cichon/Gliebe, only NIC supported (http://www.cichon.de/self/ or http://www.gliebe.de/self/)
- **MacOS-X**: by Ungar, only NIC supported (http://research.sun.com/self/)
- **Windows**: Based on Linux port, by Gliebe, uses Cygwin&XFree (http://www.gliebe.de/self/)

# Morphics: The Self GUI

- Easy to create interactive programs
- Provides an object-based artificial reality
- Display is always in sync with underlying object structure
- See Demo



If you get stuck at anytime please look in the file help.text

To share this window with a friend on the network, type in their machine name here. (Your friend must be using X, and be set to allow X connections.)

Use these buttons to move around in this large flat space.

Three button clicks will move you one screenfull.

Move to the right to find out about Mouse Button Usage, an important first step to get you started.

# The Self Syntax

- Parentheses () enclose object literals
- Slot lists are enclosed by |s, everything else is code
- Message selectors are like in Smalltalk
  - identifier: unary message (e.g., size or getLength)
  - operator: binary message (e.g., + or ~*)
  - keywords: multi-argument message (e.g., at:Put:). Capitalization of second and following keywords helps eliminating parentheses.
- Brackets [] enclose blocks, which are syntatic sugar for certain constructs
- Everything else is a message to an object

- A good tutorial is here: [http://research.sun.com/research/self/release/Self-4.0/Tutorial/index.html](http://research.sun.com/research/self/release/Self-4.0/Tutorial/index.html)

## The Self Syntax: Examples

```
(| parent* = obj1. width = 5. width: = <-.
   height = 9. height: = <- |)
(| parent* = obj1. width <- 5. height <- 9 |)

(| parent* = obj2. height = (^width).
   height: = (| :x | width: x) |).
(| parent* = obj2. height = (^width).
   height: x = (width: x) |)
```

## Goodies

- persistency framework: “Transporter” (also handles namespaces)
- Seamlessly integrated Smalltalk system
- Parser generator
- Collaboration environment
- TCP/IP, Web-Server
- Glue to arbitrary C++ code
- object-library

## References

- Byte Magazine August 1981 (The Smalltalk Edition)
- Goldberg, Robson: Smalltalk-80 The Language and its Implementation, Addision-Wesley
- Rosenbeck: Grundlagen Programmiersprachen, c't Magazin 4/1989, pp. 106 (German)
- Ungar, Smith: SELF The Power of Simplicity, Journal of Lisp and Symbolic Computation, 4/1991
- Offical Self Homepage: http://research.sun.com/self/
- x86 ports of Self: http://www.cichon.de/self/ and http://www.gliebe.de/self/
- Self Newsgroups: (http://www.egroups.com/list/self-interest)