

LEGO[©] programming: non-verbal dimensions of computer programming

Simone Martini

Alma mater studiorum • Università di Bologna
and
INRIA FoCUS – Sophia / Bologna

April 8, 2022



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
DIPARTIMENTO DI INFORMATICA - SCIENZA E INGEGNERIA



Two different approaches

“Swipe” and “tap” for our phones



Text for our programs

```
def QS(L):  
    if L==[]: return L  
    pivot = L[0]  
    return QS([x for x in L[1:] if x < pivot])  
        + [pivot] +  
        QS([x for x in L[1:] if x >= pivot])
```

[Photo: ©Minuum]

Two different approaches

“Swipe” and “tap” for our phones



[Photo: ©Minuum]

Text for our programs

```
def QS(L):  
    if L==[]: return L  
    pivot = L[0]  
    return QS([x for x in L[1:] if x<lt pivot])  
        + [pivot] +  
        QS([x for x in L[1:] if x \ge pivot])
```

The background of the slide features a large, faint watermark of the University of Cologne seal. The seal is circular and contains the text 'R STUD' at the top and 'OLOGNA' and 'RUM' at the bottom. In the center, there is a shield with three crosses and the word 'LIBERTAS'. Below the shield, there are several figures in a Gothic architectural setting, including a seated figure and a kneeling figure.

Sed contra:

Using a phone is not computer programming

We cannot compare the two activities



Are they really non comparable?

Any entity which can perform a computation

(= an *abstract machine*)

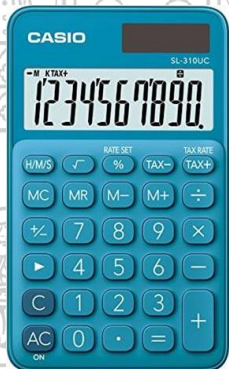
has its own **machine language**

Are they really non comparable?

Any entity which can perform a computation

(= an *abstract machine*)

has its own **machine language**

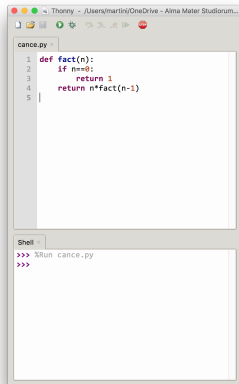


Are they really non comparable?

Any entity which can perform a computation

(= an *abstract machine*)

has its own **machine language**



```
Thonny - /Users/martin/OneDrive - Alma Mater Studiorum...
cance.py
1 def fact(n):
2   if n==0:
3     return 1
4   return n*fact(n-1)
5 |

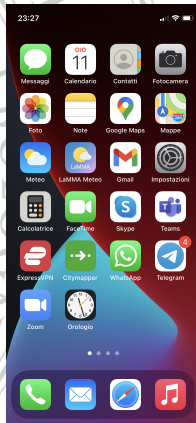
Shell
>>> %run cancel.py
>>>
```

Are they
really non comparable?

Any entity which can perform a computation

(= an *abstract machine*)

has its own **machine language**

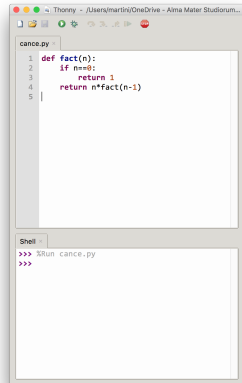
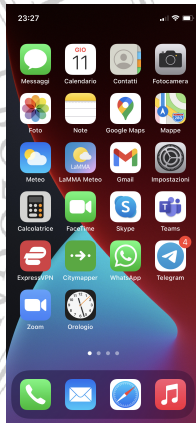
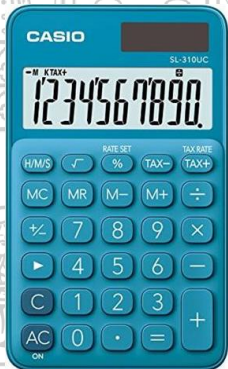


Are they really non comparable?

Any entity which can perform a computation

(= an *abstract machine*)

has its own **machine language**



Programming languages

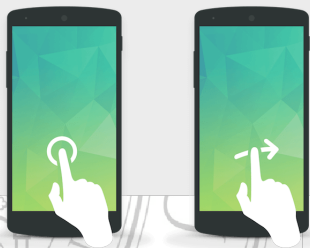
When a machine language is “powerful enough”
(= Turing-complete)
it is called a **programming language** .

But the conceptual framework is the same:
an abstract machine with its own language

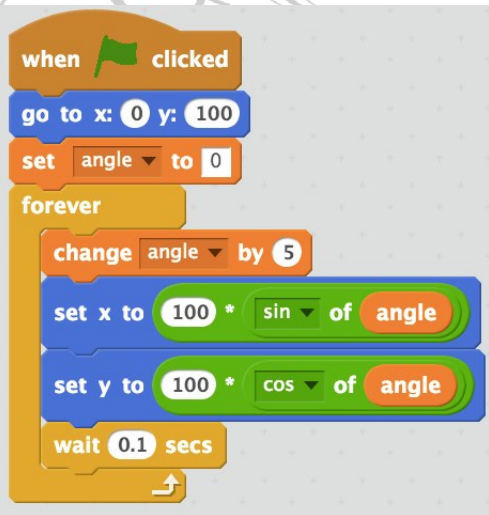
Visual and gestural languages

Is it possible a real **visual**, or **gestural**, programming language?

Can we trace this idea in the history of programming languages?

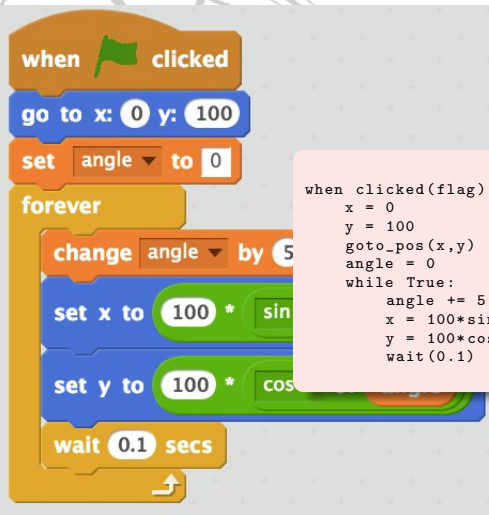


Scratch



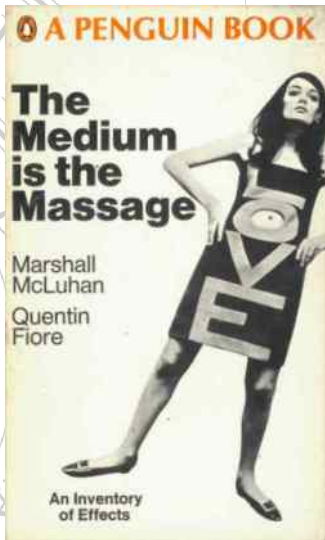
A project of the Lifelong Kindergarten Group at the MIT Media Lab

Scratch



```
when clicked(flag):  
  x = 0  
  y = 100  
  goto_pos(x,y)  
  angle = 0  
  while True:  
    angle += 5  
    x = 100*sin(angle)  
    y = 100*cos(angle)  
    wait(0.1)
```

The medium is the message



The linguistic metaphor

When Technology Became Language: The Origins of the Linguistic Conception of Computer Programming, 1950-1960

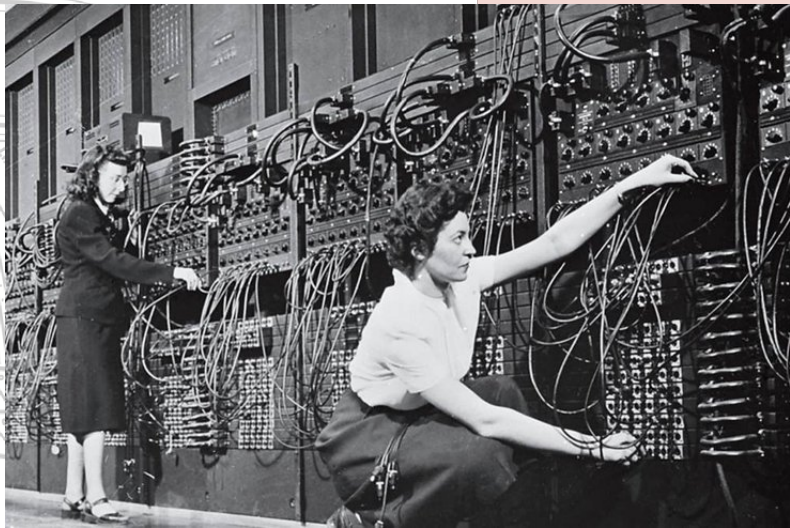
David Nofre, Mark Priestley, Gerard Alberts

Technology and Culture, Volume 55, Number 1, January 2014, pp. 40-75 (Article)

Published by The Johns Hopkins University Press

DOI: [10.1353/tech.2014.0031](https://doi.org/10.1353/tech.2014.0031)





ENIAC programming: cables (1945-46)

Programming languages

- technical tool
- object of study
- meta-languages:
algorithms published in Algol
on the *Communications of ACM*, 50s-60s

ALGORITHM 64
QUICKSORT

C. A. R. HOARE

Elliott Brothers Ltd., Borehamwood, Hertfordshire, Eng.

```
procedure quicksort (A,M,N); value M,N;  
                array A; integer M,N;
```

comment Quicksort is a very fast and convenient method of sorting an array in the random-access store of a computer. The entire contents of the store may be sorted, since no extra space is required. The average number of comparisons made is $2(M-N) \ln(N-M)$, and the average number of exchanges is one sixth this amount. Suitable refinements of this method will be desirable for its implementation on any actual computer;

```
begin          integer I,J;  
                if M < N then begin partition (A,M,N,I,J);  
                    quicksort (A,M,J);  
                    quicksort (A, I, N)  
                end  
end          quicksort
```

Using Scratch

The screenshot displays the Scratch programming environment. The interface includes a top menu bar with 'Scratch', 'File', 'Edit', 'Untitled-1', 'Share', 'See Community', and 'Give Feedback'. Below the menu are tabs for 'Blocks', 'Costumes', and 'Sounds'. The 'Blocks' panel on the left is expanded to show the 'Motion' category, which includes blocks for moving, gliding, turning, and changing coordinates. A script is being built in the workspace, starting with a yellow 'when clicked' block, followed by a yellow 'forever' loop block. Inside the loop, there is a blue 'move 4 steps' block, an 'if' block with a green condition 'x position > 200', and a blue 'set x to -180' block. The stage area on the right shows a cat sprite named 'Catt Flying' in a landscape with a volcano and colorful plants. The sprite's properties are visible at the bottom, including its name, size (100), and direction (90).

Bricks, not sentences in a language. . .

Using Scratch

1: concrete

A “concrete” tool

- we play with the elements, like in LEGO[©]
- no grammar rules:
either the bricks fit, or don't fit
- bricks fit together only in meaningful ways:
no syntactic errors

Using Scratch 2: live

A “live” tool

- the system responds continuously
- each action on the bricks has an immediate consequence
- no cycle *code-compile-run*

Using Scratch 3: tinkerable

A “tinkerable” tool

- “It lets users experiment with commands and code snippets the way one might tinker with mechanical or electronic components.”
- “the brick shapes suggest what is possible”
- “experimentation and experience teaches what works”

Using Scratch:

- concrete (visual)
- live
- tinkerable

Using Scratch:

- concrete (visual)
- live
- tinkerable

All elements also present in the “tap and swipe” language of our smartphones

Using Scratch:

- concrete (visual)
- live
- tinkerable

We can trace these elements in the history of programming languages...?

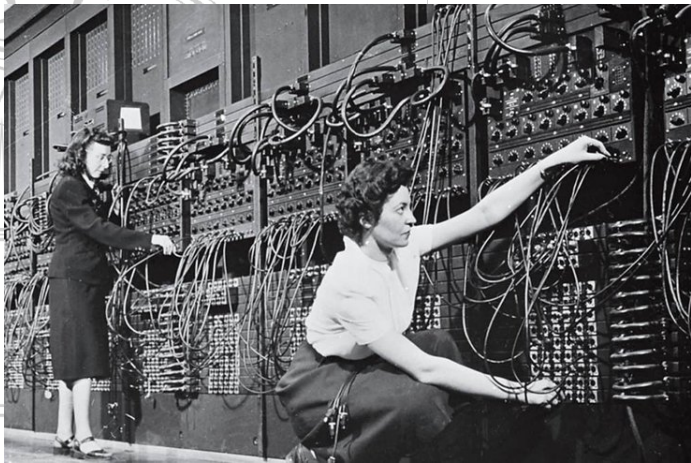
Using Scratch:

- concrete (visual)
- live
- tinkerable

We can trace these elements in the history of programming languages...?

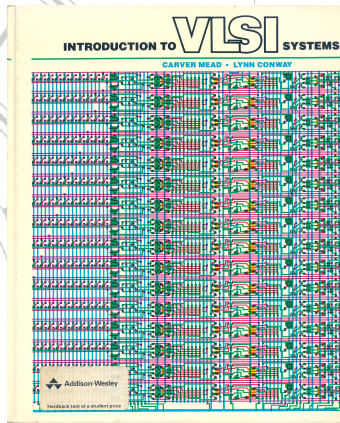
But the truly important thing is their presence together...

Concrete - Visual: ENIAC



ENIAC: 1945-46

Concrete - Visual: VLSI design



Very large-scale integration according to Mead and Conway, 1977ff; book ©1980

Concrete - Visual: VLSI design

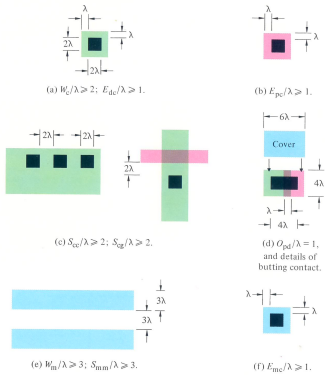
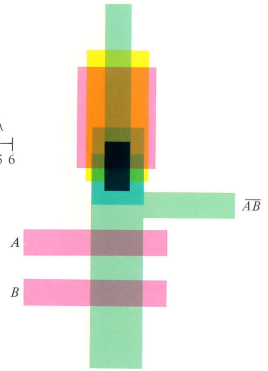


PLATE 3 nMOS design rules (continued)



Mead and Conway, 1977ff; book ©1980

Mead and Conway's VLSI

Like Scratch for programming:

- project of integrated circuits possible for people without any training in electronics/semi-conductors
- graphical composition rules
- ensuring the correction of the design

Concrete - Visual: languages from the 80s

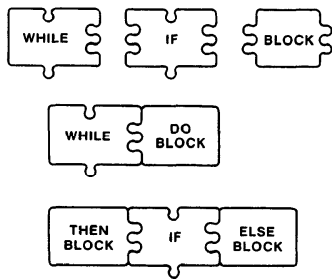


Figure 1: Possible Realizations of Some Imperative Procedural Constructs Using Tiles.

BLOX, 1987; E. P. Glinert



Live:
Maloney and Smith, 1995

**Directness and Liveness in the Morpic
User Interface Construction Environment**

John H. Maloney
Apple Computer, Inc.
1 Infinite Loop, MS 301-3E
Cupertino, CA 95014 USA
+1-408-974-7293
J.Maloney@eWorld.com

Randall B. Smith
Sun Microsystems Laboratories
2550 Garcia Avenue, MTV 29-116
Mountain View, CA 94043 USA
+1-415-336-2620
Randall.Smith@Sun.com



Live:
Maloney and Smith, 1995

**Directness and Liveness in the Morpich
User Interface Construction Environment**

John H. Maloney
Apple Computer, Inc.
1 Infinite Loop, M/S 301-3E
Cupertino, CA 95014 USA
+1-408-974-7293
J.Maloney@eWorld.com

Randall B. Smith
Sun Microsystems Laboratories
2550 Garcia Avenue, MTV 29-116
Mountain View, CA 94043 USA
+1-415-336-2620
Randall.Smith@Sun.com

Liveness means the user interface is always active and reactive:

- objects respond to user actions
- animations run
- layout happens, and
- information displays update continuously



Live:
Maloney and Smith, 1995

**Directness and Liveness in the Morpnic
User Interface Construction Environment**

John H. Maloney
Apple Computer, Inc.
1 Infinite Loop, M/S 301-3E
Cupertino, CA 95014 USA
+1-408-974-7293
J.Maloney@cWorld.com

Randall B. Smith
Sun Microsystems Laboratories
2550 Garcia Avenue, MTV 29-116
Mountain View, CA 94043 USA
+1-415-336-2620
Randall.Smith@Sun.com

Liveness means the user interface is always active and reactive:

- *objects* respond to user actions
-
-
-

A long history

To program is to interact with the executor, the machine

- Logo, 1969:

W. Feurzeig and S. Papert. Programming languages as a conceptual framework for teaching mathematics. Final report on the first fifteen months of the Logo Project. TR 1889. BBN, Cambridge, MA.

- Smalltalk, 1972

Alan Kay, XEROX PARC

Interaction is expressed with powerful metaphors:

turtle

objects

A long history

To program is to interact with the executor, the machine

- Logo, 1969:

W. Feurzeig and S. Papert. Programming languages as a conceptual framework for teaching mathematics. Final report on the first fifteen months of the Logo Project. TR 1889. BBN, Cambridge, MA.

- Smalltalk, 1972

Alan Kay, XEROX PARC

Interaction is expressed with powerful metaphors:

turtle

objects

The turtle



Tinkerable

*We have to make a little detour,
to explain why it is important*

Constructivism

Per conoscere il mondo bisogna costruirlo

Cesare Pavese, Il mestiere di vivere. 1952

To know the world, we should construct it

We really know (only) what we (re-)build on our own.

Constructivism

Per conoscere il mondo bisogna costruirlo

Cesare Pavese, Il mestiere di vivere. 1952

To know the world, we should construct it

We really know (only) what we (re-)build on our own.

Constructivism

We build concrete models of abstract concepts

Using a programming language is one of the most effective and economical ways to obtain such models independently.



Per conoscere il mondo bisogna costruirlo

Cesare Pavese, Il mestiere di vivere. 1952

To know the world, we should construct it

In other words, we make not just to have, but to know.

But the having can happen without most of the knowing taking place.

Alan Kay, The early history of Smalltalk. 1993



We have a problem:

If the “having” can happen without the “knowing” taking place, are there conditions under which “making” produces the “knowing” and not only the “having”?

Making for knowing:

Seymour Papert:

We should build meaningful objects and relations



We have a problem:

If the “having” can happen without the “knowing” taking place, are there conditions under which “making” produces the “knowing” and not only the “having”?

Making for knowing:

Seymour Papert:

We should build **meaningful** objects and relations

The background of the slide features a large, faint watermark of the University of Cologne seal. The seal is circular and contains the text 'R STU' at the top and 'BOLOGNA RUM' at the bottom. In the center, there is a shield with a cross and three crosses above it, and below the shield, there are three figures in a Gothic architectural setting.

Piaget and the constructivism

We know what we construct

Papert and the constructionism

We should build **meaningful** object and relations

The background of the slide features a large, faint watermark of the University of Cologne seal. The seal is circular and contains the text 'R STU' at the top and 'OLOGNA' and 'RUM' on the sides. In the center, there is a shield with three crosses and the word 'LIBER'. Below the shield, there are several figures in a Gothic architectural setting, including a seated figure with a lamb and a kneeling figure with an angel.

Piaget and the constructivism

We know what we construct

Papert and the constructionism

We should build **meaningful** object and relations

Papert and the constructionism

constructivism + meaningfulness

We build concrete versions of abstract concepts
and we **enter into relation** with these concrete objects



Affective dimension

Build “objects to think with”

oxymoron:

the abstract is obtained using the concrete

In the choice of these objects:

Not only a cognitive dimension:

Papert: “I was in love with gears!”

Papert

Using a programming language is one of the most effective and economical ways for children to obtain such models independently.

But:

The modality of interaction with the computer is as important (and probably more important) than its content

The “test and correct” cycle

Obtaining feedback from computational objects.



The context for the “computational thinking” citation

Samba schools for computation

In the next few years we shall see the formation of some computational environments that deserve to be called “samba schools for computation.”

There have already been attempts in this direction [but] their visions of how to integrate computational thinking into everyday life was insufficiently developed.

Samba schools, in Rio





In samba schools for computation:

- no knowledge is **transmitted**
- pupils will learn because are immersed in an environment
- activities are both “rich of computational principles” and **meaningful for the community**

Smalltalk, at the beginning

Alan Kay:

- 1966-1969: master student, University of Utah
- summer 1967: he learns Papert's ideas, from Marvin Minsky
- winter 1968: he meets Papert and his group

*This encounter finally hit me with what the destiny of **personal computing** really was going to be: [...] a **personal dynamic medium** [which] had to extend into the world of childhood.*

*a **personal computer**: It had to be no larger than a notebook, with a friendly interface but with the power of a real programming language.*

Smalltalk, at the beginning

Alan Kay:

- 1966-1969: master student, University of Utah
- summer 1967: he learns Papert's ideas, from Marvin Minsky
- winter 1968: he meets Papert and his group

*This encounter finally hit me with what the destiny of **personal computing** really was going to be: [...] a personal **dynamic medium** [which] had to extend into the world of childhood.*

*a **personal computer**: It had to be no larger than a notebook, with a friendly interface but with the power of a real programming language.*

Smalltalk, at the beginning

Alan Kay:

- 1966-1969: master student, University of Utah
- summer 1967: he learns Papert's ideas, from Marvin Minsky
- winter 1968: he meets Papert and his group

*This encounter finally hit me with what the destiny of **personal computing** really was going to be: [...] a **personal dynamic medium** [which] had to extend into the world of childhood.*

*a **personal computer**: It had to be no larger than a notebook, with a friendly interface but with the power of a real programming language.*

Smalltalk

It isn't enough to just learn to read and write. There is also a literature that renders ideas. Language is used to read and write about them, but at some point the organization of ideas starts to dominate mere language abilities.

And it helps greatly to have some powerful ideas under one's belt to better acquire more powerful ideas [Papert 70s]. So, we decided we should teach design.

Smalltalk

*From the objects and classes of programming languages
to an ecosystem of interacting objects*

Smalltalk for Alto, fin 1970s

System Browser

Collections-Sequence	Interval	accessing	collect:
Collections-Text	LinkedList	do:	do:andBetweenDo:
Collections-Array	MappedCollection	adding	promoteFirstSuchT
Collections-Stream	OrderedCollection	removing	reverse
Collections-Support	SortedCollection	enumerating	reverseDo:
Graphics-Primitives	-----	private	select:
Graphics-Display	-----	-----	-----
Graphics-Media	-----	-----	-----
Graphics-Paths	-----	-----	-----

instance class

collect: aBlock

*"Evaluate aBlock with each of my elements as the argument. C
resulting values into a collection that is like me. Answer with
collection. Override superclass in order to use add:, not at:put:*

```
] newCollection |  
newCollection + self species new.  
self do: [:each | newCollection add: (aBlock value: each)].  
newCollection
```

User Interrupt

```
Paragraph>>characterBlockAtPoint:  
Paragraph>>mouseSelect:to:  
CodeController(ParagraphEditor)>>processRedButton  
CodeController(ParagraphEditor)>>processMouseButtons  
CodeController(ParagraphEditor)>>controlActivity  
CodeController(Controller)>>controlLoop
```

controlActivity

```
self scrollBarContainsCursor  
ifTrue:  
[self scroll]  
ifFalse:  
[self processKeybo  
self processMouse
```

File List

```
[ ](Robson)SF*  
[File] (Robson)SF>ScreenForm.txt  
[File] (Robson)SF>ScreenForm.txt  
[File] (Robson)SF>ScreenFormChanges.st  
[File] (Robson)SF>WordGraphics.form  
-----  
Rectangle fromUser origin  
ScreenForm setFullPageWidth.  
ScreenForm  
printRectangle:  
(0045 extent: 674@790)  
onFullNamed: 't:amp;ScreenPress'  
(Form readFrom: 'FiledSkate.form') edit
```

Fig.1



Smalltalk

From the objects and classes of programming languages
to an ecosystem of interacting objects

Smalltalk is NOT only its syntax or the class.
I'm sorry that I long ago coined the term "objects"
for this topic because it gets many people to focus
on the lesser idea.

The big idea is "messaging" [...] The Japanese have
a small word -- ma -- for "that which is in between".

A. Kay, message to the Squeak-dev mailing list. Sat Oct 10 1998

Smalltalk

From the objects and classes of programming languages
to an ecosystem of interacting objects

Smalltalk is NOT only its syntax or the class.
I'm sorry that I long ago coined the term "objects"
for this topic because it gets many people to focus
on the lesser idea.

The big idea is "messaging" [...] The Japanese have
a small word -- ma -- for "that which is in between".

A. Kay, message to the Squeak-dev mailing list. Sat Oct 10 1998

Smalltalk

And this is reflected in Smalltalk itself:

when ST hit the larger world, it was pretty much taken as "something just to be learned", as though it were Pascal or Algol.

while is something we should tinker with:

at PARC we changed Smalltalk constantly, treating it always as a work in progress

Smalltalk

And this is reflected in Smalltalk itself:

when ST hit the larger world, it was pretty much taken as "something just to be learned", as though it were Pascal or Algol.

while is something we should tinker with:

at PARC we changed Smalltalk constantly, treating it always as a work in progress

Smalltalk

And this is reflected in Smalltalk itself:

when ST hit the larger world, it was pretty much taken as "something just to be learned", as though it were Pascal or Algol.

while is something we should tinker with:

at PARC we changed Smalltalk constantly, treating it always as a work in progress

Conclusions

Programming in visual languages
and the interaction with our computing means
exploits less and less the linguistic metaphor

This view has ancient and well-established roots, even in the
classical linguistic context

THE CRAFTSMAN



Richard Sennett

The image shows the front cover of the book 'The Craftsman' by Richard Sennett. The cover is black with the title 'THE CRAFTSMAN' in white, serif, all-caps font at the top. Below the title is a black and white photograph of a hand holding a small, round object. At the bottom of the cover, the author's name 'Richard Sennett' is printed in white. The book cover is positioned on the left side of a slide that features a large, faint watermark of a circular seal in the background. The seal contains the text 'RUTGERS STATE UNIVERSITY' and 'EST. 1823'.

THE CRAFTSMAN

Richard Sennett

what we can say in words may be more limited than what we can do with things. [...]

Here is a, perhaps the, fundamental human limit: language is not an adequate “mirror-tool” for the physical movements of the human body.

[R. Sennett, The Craftsman. 2009]

An old observation

Janvier 1751.

ENCYCLOPÉDIE,
OU
DICTIONNAIRE RAISONNÉ
DES SCIENCES,
DES ARTS ET DES MÉTIERS,
RECUEILLI
DES MEILLEURS AUTEURS
ET PARTICULIÈREMENT
DES DICTIONNAIRES ANGLAIS
DE CHAMBERS, D'HARRIS, DE DYCHE, &c.
PAR UNE SOCIÉTÉ DE GENS DE LETTRES.

Mis en ordre & publié par M. DIDEROT; & quant à la PARTIE MATHÉMATIQUE,
par M. D'ALEMBERT, de l'Académie Royale des Sciences de Paris
& de l'Académie Royale de Berlin.

*Tantum series juncturaque polles,
Tantum de medio sumptis accedit honoris! HORAT.*

DIX VOLUMES IN-FOLIO,
DONT DEUX DE PLANCHES EN TAILLE-DOUCE,

PROPOSÉS PAR SOUSCRIPTION.



A PARIS, Chez
BRIASSON, rue Saint Jacques, à la Science.
DAVID l'ainé, rue Saint Jacques, à la Plume d'Or.
LE BRETON, Imprimeur ordinaire du Roy, rue de la Harpe.
DURAND, rue Saint Jacques, à Saint Landry, & au Griffon.

M. DCC. LI.

AVEC APPROBATION ET PRIVILEGE DU ROY.

An old observation

Janvier 1751.
ENCYCLOPÉDIE,
OU
DICTIONNAIRE RAISONNÉ
DES SCIENCES,
DES ARTS ET DES MÉTIERS,
RECUEILLI
DES MEILLEURS AUTEURS
ET PARTICULIÈREMENT
DES DICTIONNAIRES ANGLAIS
DE CHAMBERS, D'HARRIS, DE
PAR UNE SOCIÉTÉ DE GENS DE LETTRES
Mis en ordre & publié par M. DIDEROT, & quant à la PARTIE
par M. D'ALEMBERT, de l'Académie Royale des Sciences
& de l'Académie Royale de Berlin.
*Tantum series juncturaque pollet,
Tantum de medio sumptis accedit honoris!* HORAT.
DIX VOLUMES INFOLIO
DONT DEUX DE PLANCHES EN TAILLE-DOUBLE
PROPOSÉS PAR SOUSCRIPTION



A PARIS, Chez
BRIASSON, rue Saint Jacques, à la Science.
DAVID l'Écuyer, rue Saint Jacques, à la Plume d'Or.
LE BRETON, Imprimeur ordinaire du Roy, rue de la Harpe.
DURAND, rue Saint Jacques, à Saint Landry, & au Griffon.

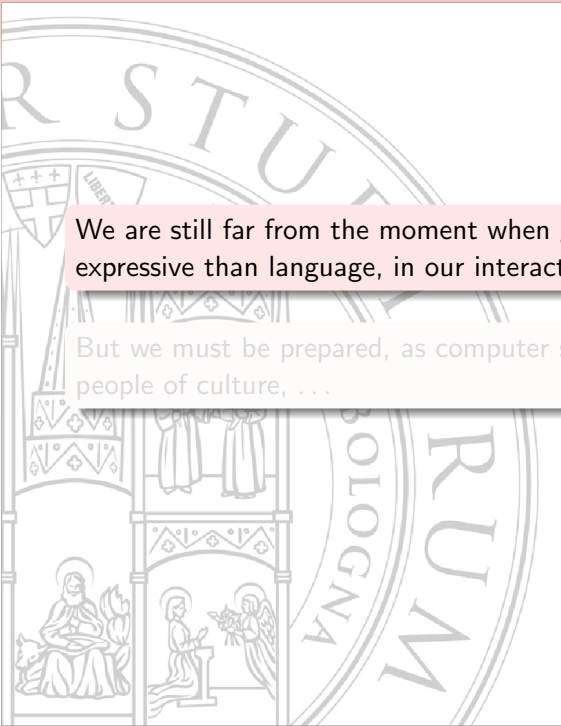
M. DCC. LI.

AVEC APPROBATION ET PRIVILEGE DU ROY.

On s'est adressé aux plus habiles de Paris et du royaume. On s'est donné la peine d'aller dans leurs ateliers [...]

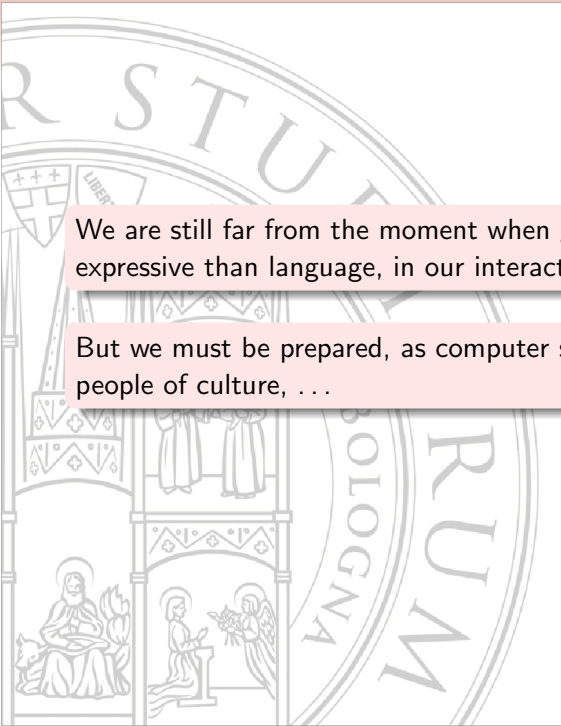
À peine, entre mille, en trouve-t-on une douzaine en état de s'exprimer avec quelque clarté sur les instruments qu'ils emploient et sur les ouvrages qu'ils fabriquent.

[D. Diderot, Prospectus à l'Encyclopédie, 141; 1751.]

The background of the slide features a large, faint watermark of the University of Bonn seal. The seal is circular and contains the text 'R STU' at the top and 'BOLOGNA RUM' at the bottom. In the center, there is a shield with three crosses and the word 'LIBER'. Below the shield, there are three figures: a seated figure on the left, a kneeling figure in the middle, and a standing figure on the right.

We are still far from the moment when gesture will be more expressive than language, in our interaction with computers

But we must be prepared, as computer scientists, epistemologists, people of culture, ...

The background of the slide features a large, faint watermark of the University of Cologne seal. The seal is circular and contains the text 'R STU' at the top and 'R U M' at the bottom. In the center, there is a shield with three crosses and the word 'LIBER'. Below the shield, there are three figures in a Gothic architectural setting: a seated figure on the left, a kneeling figure in the middle, and a standing figure on the right. The text 'BOLOGNA' is also visible on the right side of the seal.

We are still far from the moment when gesture will be more expressive than language, in our interaction with computers

But we must be prepared, as computer scientists, epistemologists, people of culture, ...