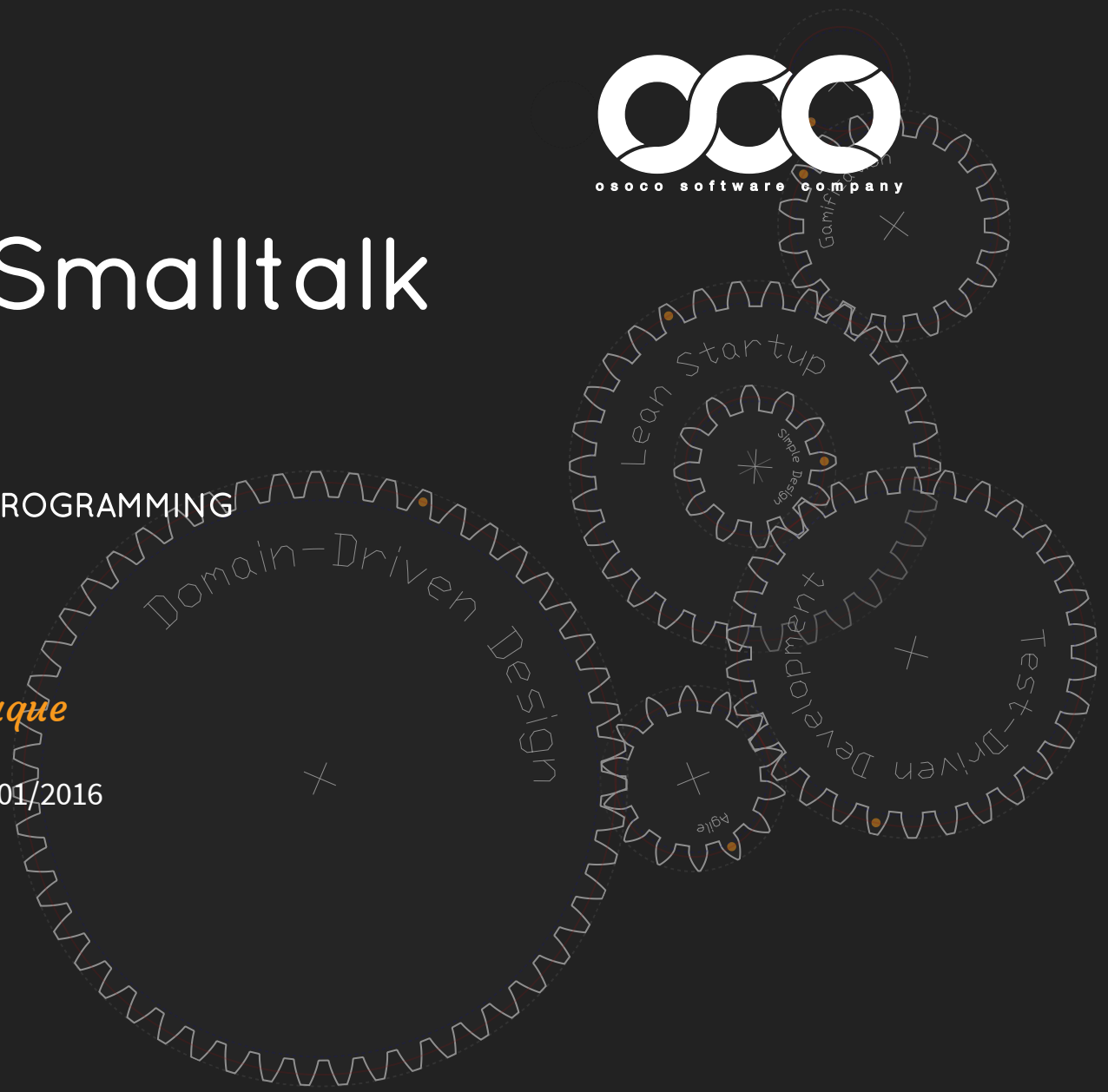
Take the Smalltalk Red Pill

DISCOVER A NEW WAY OF PROGRAMMING



Rafael Luque / @rafael_luque

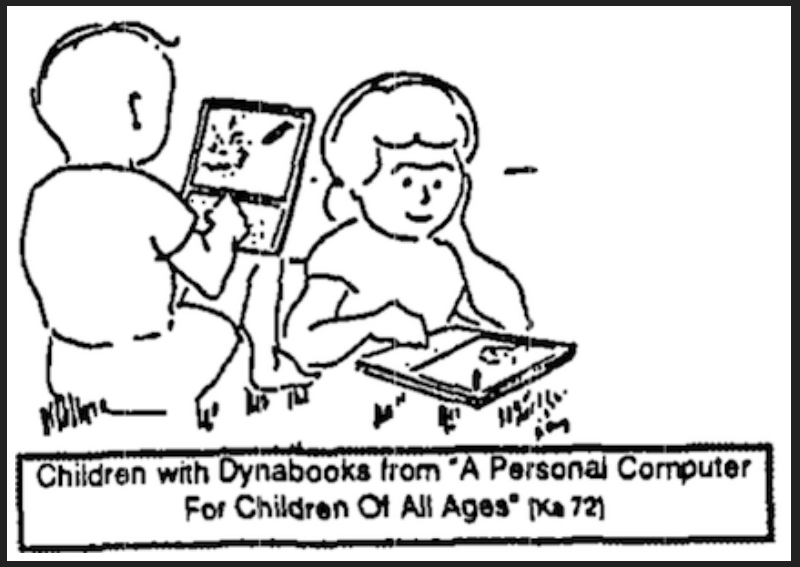
Madrid Smalltalk User Group - 25/01/2016

What is Smalltalk?


Smalltalk is the software part of Alan Kay's personal computing vision.

The Dynabook

*Imagine having your own **self-contained knowledge manipulator in a portable package** the size and shape of an ordinary notebook. How would you use it if it had enough power to outrace your senses of sight and hearing, enough capacity to store for later retrieval thousands of page-equivalents of reference materials, poems, letters, recipes, drawings, animations, musical scores, waveforms, dynamic simulations, and anything else you would like to create, remember, and change?*



Children with Dynabooks from "A Personal Computer For Children Of All Ages" [Ka 72]

Interim Dynabooks

We designed and built a number of stand-alone self-contained ***Interim Dynabooks*** *in order to have a solid test-bed for our ideas. These machines are the environment for our* ***experimental communications medium, Smalltalk****.*

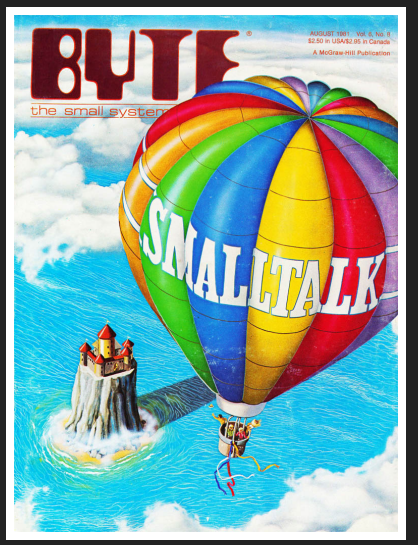

Created at Xerox PARC in the 70s

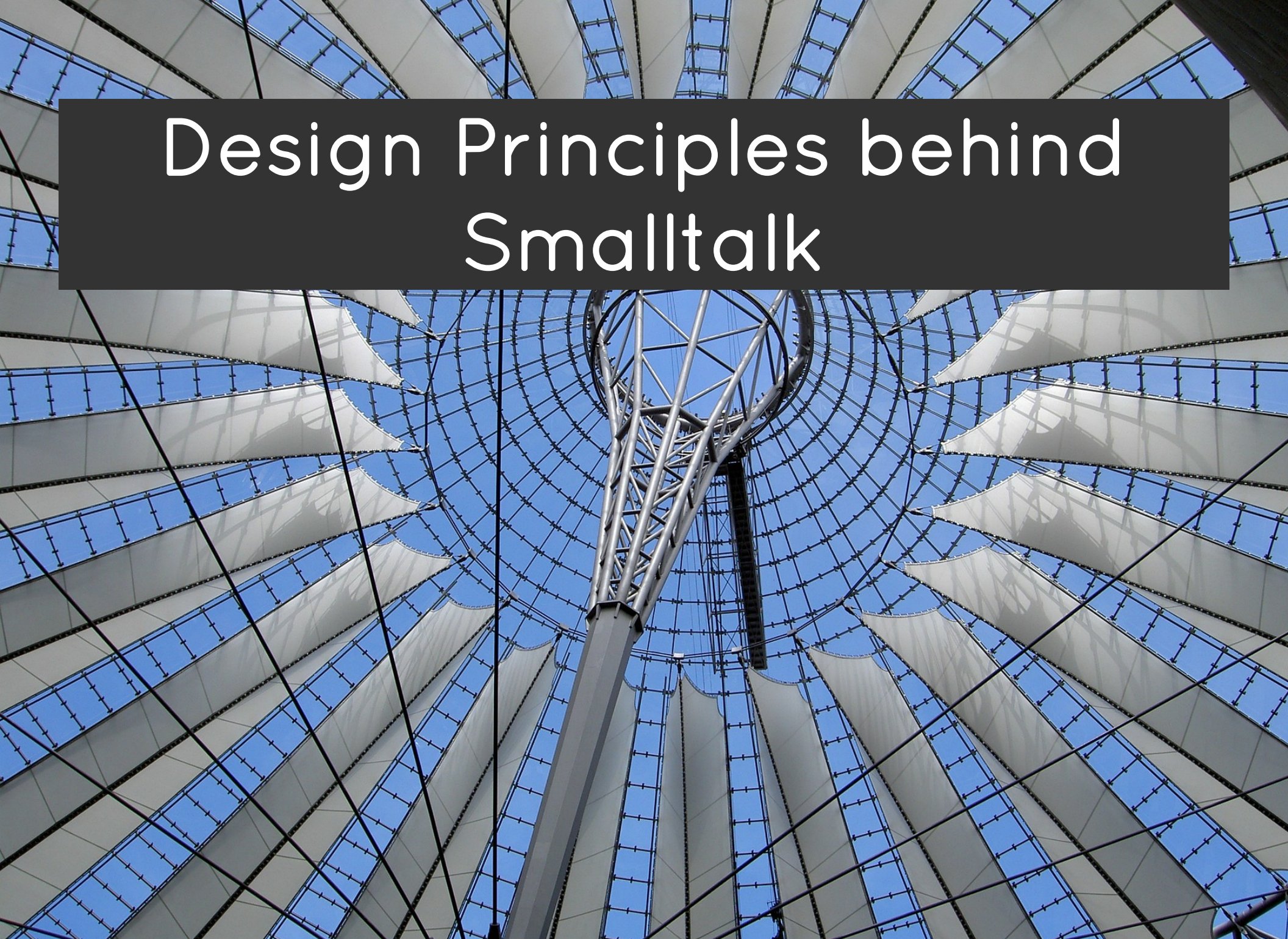
Inspirations

- **Sketchpad** by Ivan Sutherland, 1963.
- **NLS** by Doug Engelbart, 1966.
- **Logo** by Seymour Papert, 1968.
- **Simula** by Dahl and Nygaard, 1967.
- **LISP** by John McCarthy, 1958.

A bit of history

- 60s: Alan Kay develops early concepts based on Lisp, Simula and Logo.
- 1969: Smalltalk first coined.
- 1972: Smalltalk-72 is the first released version.
- 1974: Smalltalk-74.
- 1976: Smalltalk-76.
- 1980: Smalltalk-80 released. The first version to external companies.
- 80s: Digitalk and ParcPlace released commercial versions.





Design Principles behind Smalltalk

Personal Mastery

If a system is to serve the creative spirit, it must be **entirely comprehensible to a single individual**.

Uniform Metaphor

A language should be designed around **a powerful metaphor** that can be **uniformly applied** in all areas.

Reactive Principle

Every component accessible to the user should be able to present itself in **a meaningful way for observation and manipulation**.

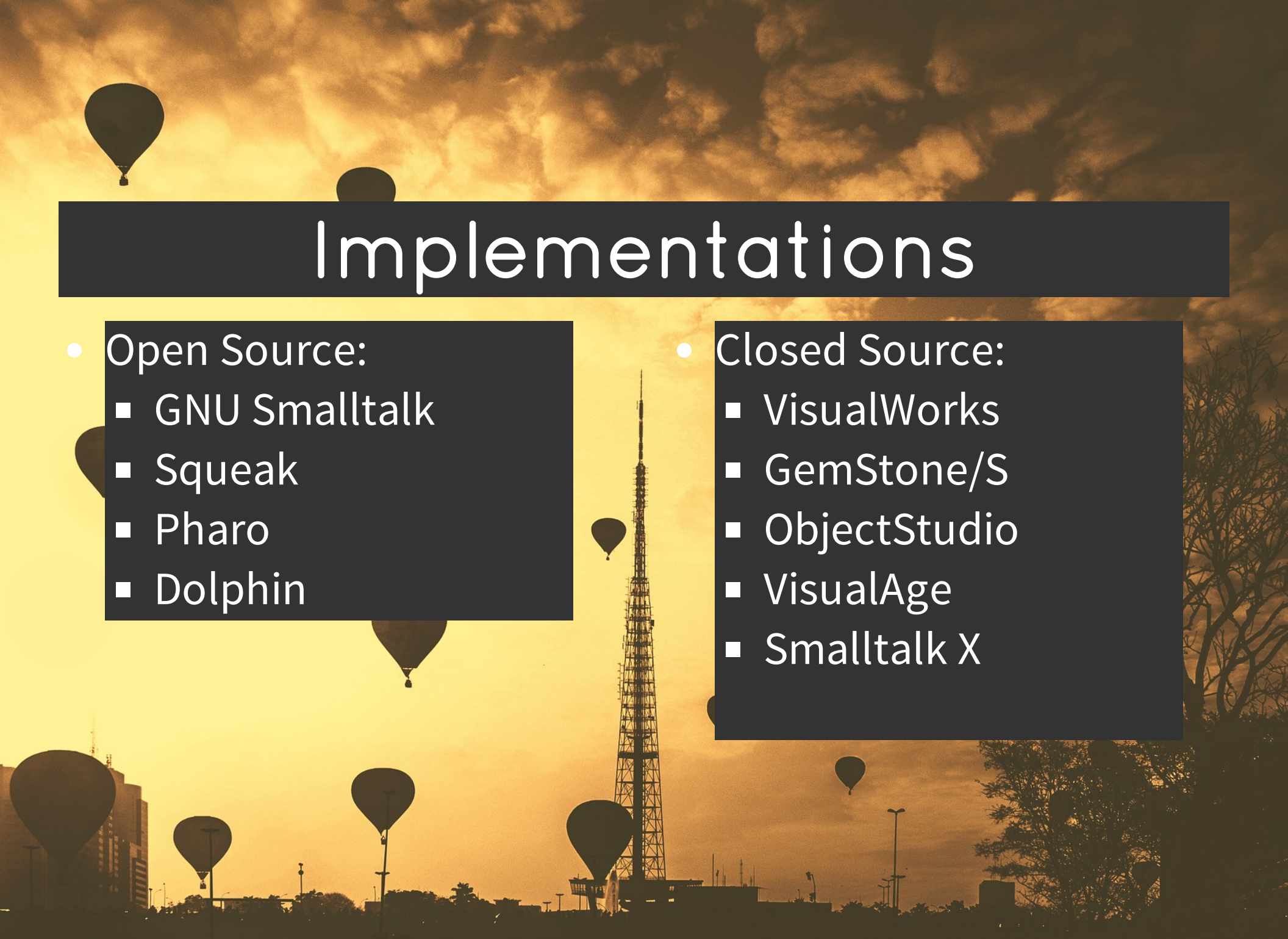
The Language

- Pure object language.
- Dynamically typed.
- Reflective language.
- Not only a language, but **an integrated environment**.

A lot of innovations

- IDE concept.
- Design patterns.
- XUnit.
- Refactorings.
- Garbage collection.
- MVC.
- JIT compilation.

*It is the first language that was **a clear improvement on** a large majority of **its successors**.*



Implementations

- Open Source:
  - GNU Smalltalk
  - Squeak
  - Pharo
  - Dolphin
- Closed Source:
  - VisualWorks
  - GemStone/S
  - ObjectStudio
  - VisualAge
  - Smalltalk X


Pharo Smalltalk

- Started in 2008 **from Squeak**.
- Pharo 1.0 released in October 2009.
- Pharo 5.0: early 2016.
- MIT license.
- Works on Windows, Mac OS X, Linux, iOS, Android and Pi.
- **Great community**.
- Improving steadily.

A brief Pharo tour

Less is more

It seems that perfection is attained not when there is nothing more to add, but when there is nothing more to remove.

--- Antoine de Saint Exupéry

- No constructors.
- No type declarations.
- No static methods.
- No interfaces.
- No visibility modifiers (public, private, package, protected).
- No parameterized types (Java Generics)
- No primitive types.

One simple and uniform model

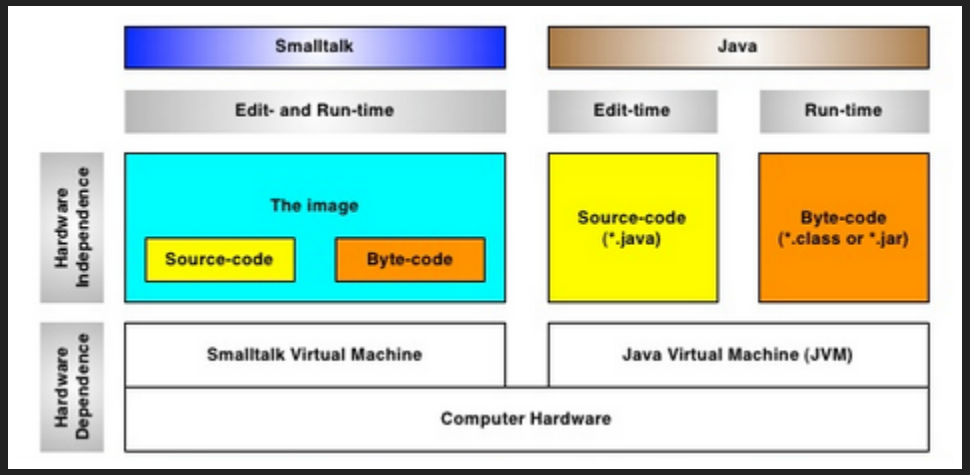
Everything is an object.

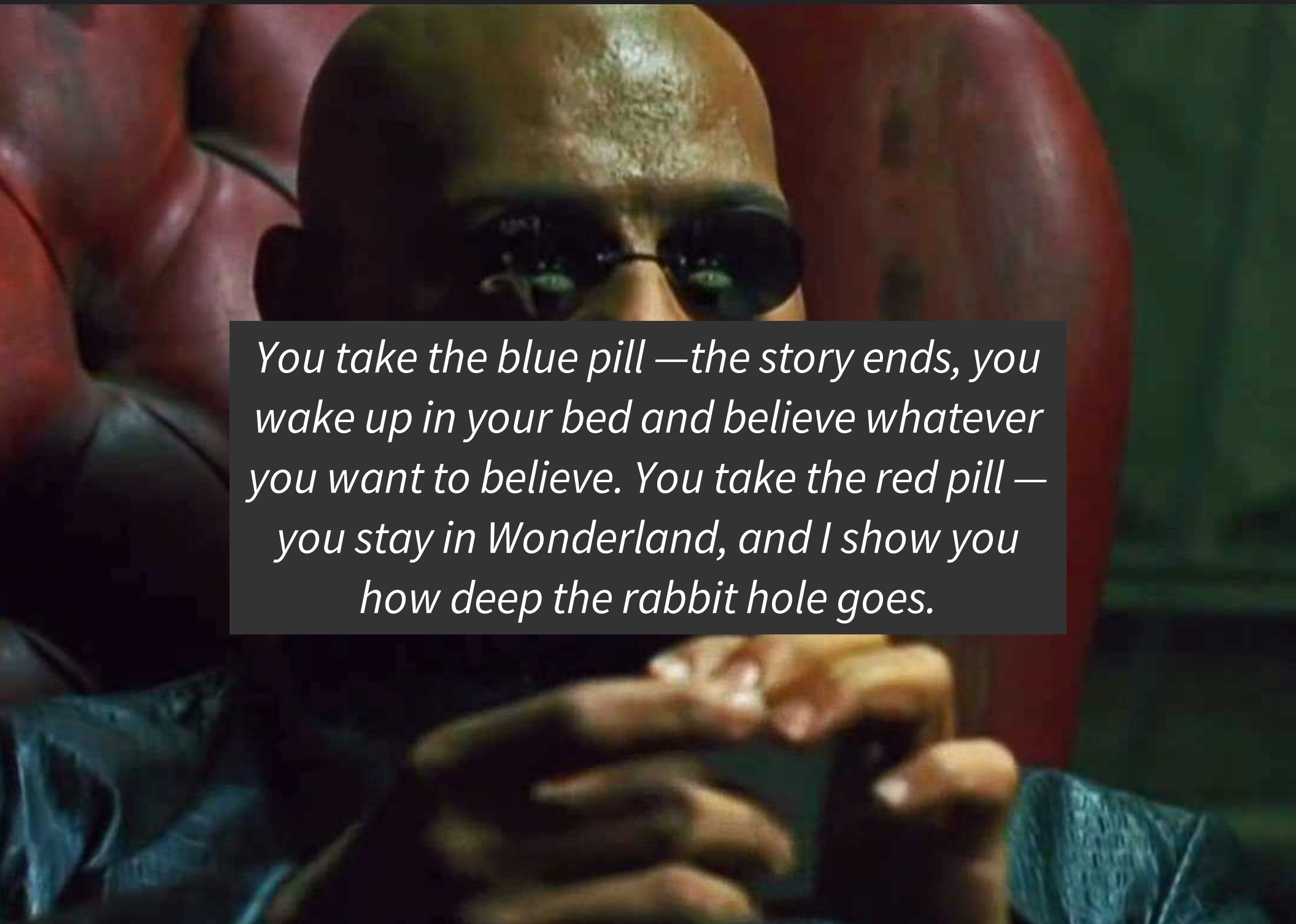
Everything happens by sending messages.

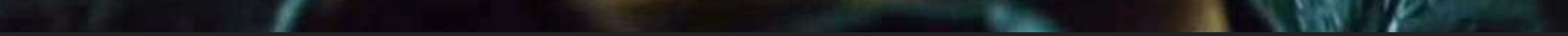
Syntax in a postcard

```
exampleWithNumber: x
  "A method that illustrates every part of Smalltalk syntax."
  |y|
  true & false not & (nil isNil) ifFalse: [self halt].
  y := self size + super size.
  { 1 . 2 .  #($a #a "a" 1 1.0) }
    do: [:each | Transcript
                    show: (each class name);
                    show: ' '].
  ^ x < y
```

Image based development




You take the blue pill —the story ends, you wake up in your bed and believe whatever you want to believe. You take the red pill — you stay in Wonderland, and I show you how deep the rabbit hole goes.



Demo 1

Living objects under your fingers

Demo 2

Fully inspectable and reflective

Demo 3

Very small environment

Demo 4

Turtles all the way down

A new way to understand programming

- Programming like a **dialog with your objects**.
- Programming = **explore + change running systems**.
- **Fast feedback** loop.
- One simple **uniform model** to access everything.
- **Not a black box**. Fully inspectable and reflective.
- The environment is so small that **you can understand and extend everything**.
- So **you are not constrained** to a predefined set of tools (*moldable tools*).

Why are we using Smalltalk at OSOCO

- We have freedom to choose technologies: *"When you choose technology, you have to ignore what other people are doing, and consider only what will work the best." (Beating the Averages, Paul Graham).*
- Unbeatable simplicity, consistency and comprehensibility.
- Programming live running systems.
- Instant feedback.
- Productivity.
- Build your own domain specific tools.
- Enabler for Domain-Driven Design.
- The ultimate agile programming environment.

Warning

Smalltalk is dangerous. It is a drug. My advice to you would be don't try it; it could ruin your life. Once you take the time to learn it (to REALLY learn it) you will see that there is still nothing out there that can quite touch it.

--- Andy Bower

Sources & References

- Personal Computing. Alan Kay.
- Design Principles behind Smalltalk. Dan Ingalls.
- Pharo By Example book.
- Deep into Pharo book.
- Pharo: Playing with live objects. Tudor Girba.
- Pharo: Objects at your fingerprints. Marcus Denker.
- Pharo MOOC. D. Cassou, S. Ducasse, L. Fabresse.
- Beating the averages. Paul Graham.

Questions?