History and Concepts

Stéphane Ducasse
Stephane.Ducasse@inria.fr
http://stephane.ducasse.free.fr

Outline

- History
- Context
- Run-Time Architecture
- Concepts

- "Making simple things very simple and complex things very possible." Alan Kay

Smalltalk: a State of Mind

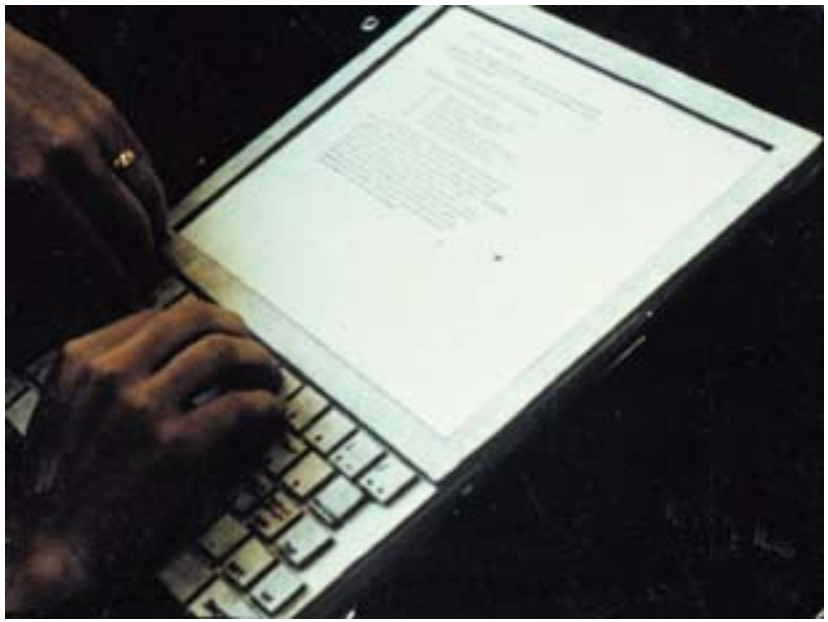
- A small and uniform language:
  - Syntax fits on one sheet of paper
- A large set of reusable classes
  - Basic Data Structures, GUI classes, Database Access, Internet, Graphics
- A set of powerful development tools
  - Browsers, GUI Builders, Inspectors, Change Management Tools, Crash Recovery Tools, Project Management Tools
- A run-time environment based on virtual machine technology
  - Really Platform Independent
- Team Working Environment (releasing, versioning, deploying).

Smalltalk - The Inspiration

- Flex (Alan Kay, 1969)
- Lisp (Interpreter, Blocks, Garbage Collection)
- Turtle graphics (The Logo Project, Programming for Children)
- Direct Manipulation Interfaces (Sketchpad, Alan Sutherland, 1960)
- NLS, (Doug Engelbart, 1968), "the augmentation of human intellect"
- Simula (Classes and Message Sending)
  - Description of real Phenomenons by means of a specification language -> modelling
- Xerox PARC (Palo Alto Research Center)
- DynaBook: a Laptop Computer for Children

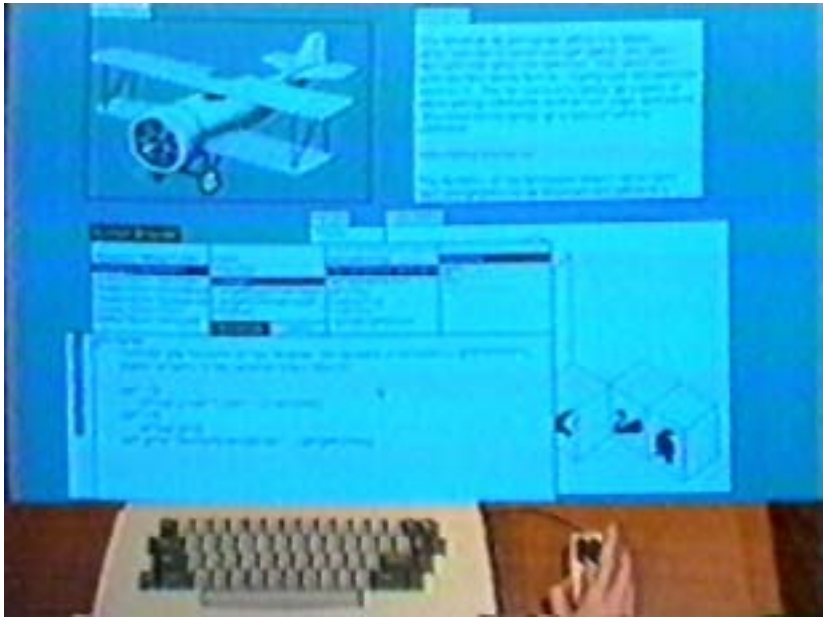
Dynabook Mock

http://www.artmuseum.net/w2vr/archives/Kay/01_Dynabook.html



Alto: a Machine to Run Smalltalk

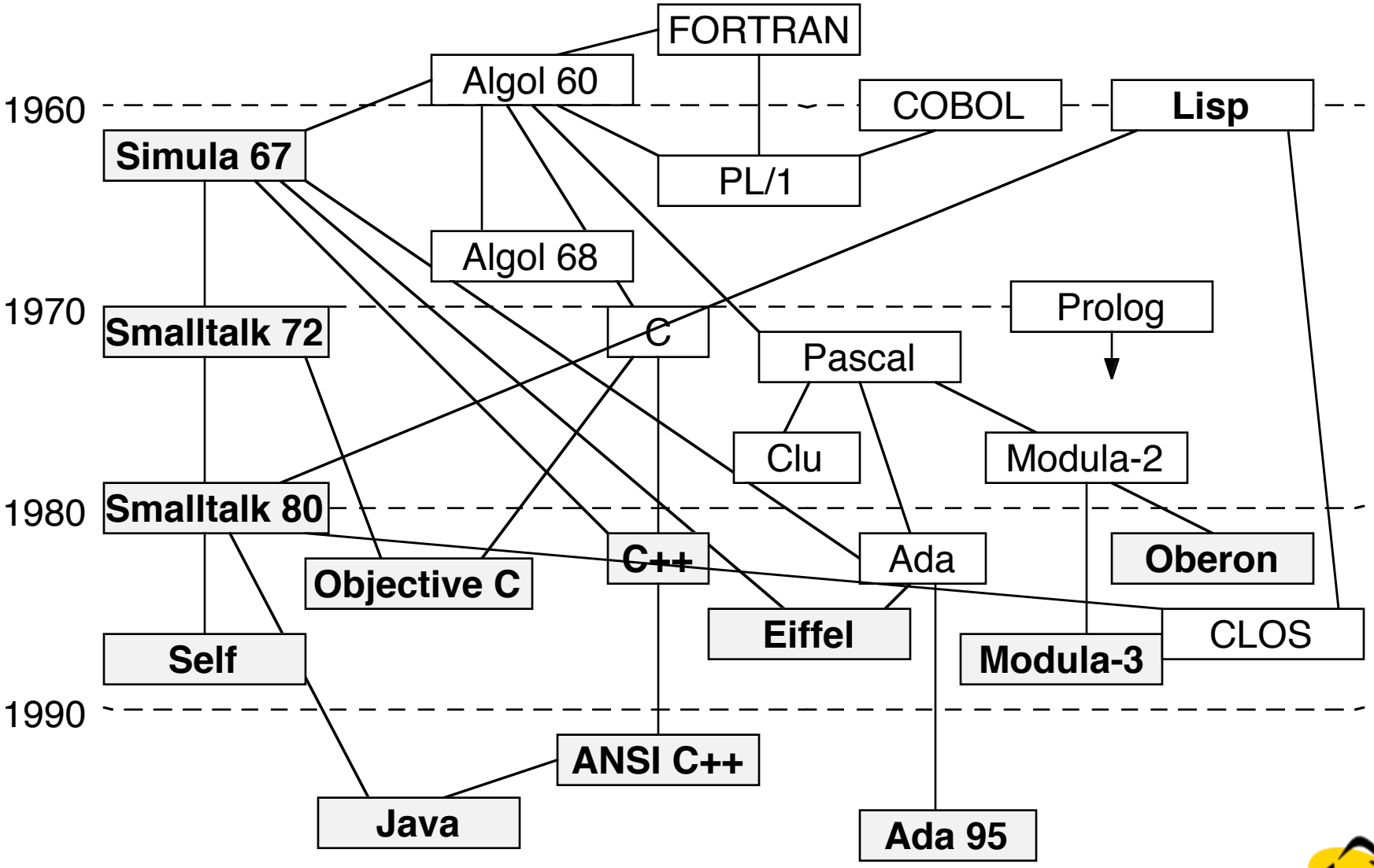
Smalltalk on Alto III



Precursor, Innovator & Visionary

- First to be based on Graphics
  - Multi-Windowing Environment (Overlapping Windows)
  - Integrated Development Environment
  - -> Debugger, Compiler, Text Editor, Browser
- With a pointing Device-> Yes, a Mouse
- Ideas were taken over
  - Apple Lisa, Mac
  - Microsoft Windows 1.0
- Virtual Machine -> Platform independent
- Garbage Collector -> Time for some real thinking...
- Just in Time Compilation
- Everything was there, the complete Source Code

The History


FORTRAN
Algol 60
COBOL
Lisp
1960
Simula 67
PL/1
Algol 68
1970
Prolog
Smalltalk 72
C
Pascal
Clu
Modula-2
1980
Smalltalk 80
C++
Ada
Oberon
Objective C
Eiffel
CLOS
Self
Modula-3
1990
ANSI C++
Java
Ada 95

The History (Internal)

- 1972 - First Interpreter -> More Agents than Objects (every object can specify its own syntax)
- 1976 - Redesign -> A Hierarchy of classes with a Unique Root, Fixed Syntax, Compact Byte Code, Contexts, Processes, Semaphores, Browsers, GUI Library. Projects: ThingLab, Visual Programming Environment Programming by Rehearsal.
- 1978 - NoteTaker Project, Experimentation with 8086 Microprocessor with only 256 KB RAM.

The History (External)

- 1980 - Smalltalk-80 (ASCII, cleaning primitives for portability, Metaclasses, Blocks as first-class Objects, MVC). Projects: Gallery Editor (mixing text, painting and animations) + Alternate Reality Kit (physics simulation)
- 1981 - Books + 4 external virtual machines (Dec, Apple, HP and Tektronix) -> GC by generation scavenging
- 1988 - Creation of Parc Place Systems
- 1992 - ANSI Draft
- 1995 - New Smalltalk implementations (MT, Dolphin, Squeak, Smalltalk/X, GNU Smalltalk)
- 2000 - Fscript, GNU Smalltalk, SmallScript
- 2002 - Smalltalk as OS: 128k ram

Smalltalk's Concepts

- Everything is an object (numbers, files, editors, compilers, points, tools, boolean).
- Objects communicate only by message passing.
- Each object is an instance of one class (which is also an object).
- A class defines the structure and the behavior of its instances.
- Each object possesses its own set of values.
- Dynamic Typing.
- Purely based on late binding.

Messages and Methods

Message: What behavior to perform

```
aWorkstation accept: aPacket
aMonter eat: aCookie
```

Method: How to carry out the behaviour

```
accept: aPacket
        (aPacket isAddressedTo: self)
                ifTrue:[ Transcript show: 'A packet is
    accepted by the Workstation ', self name asString]
                ifFalse: [super accept: aPacket]
```

Objects and Classes

- Every object is an instance of a class
- A class specifies the structure and the behaviour of all its instances
- Instances of a class share the same behavior and have a specific state
- Classes are objects that create other instances
- Metaclasses are classes that create classes as instances
- Metaclasses describe class behaviour and state (subclasses, method dictionary, instance variables...)

Summary

Had a major impact: Java, Ruby, C#

Simple and consistent model

- Everything is an object
- All computation is made by sending message
- Classes, Single inheritance, public methods, private attributes

Uniform syntax

Not an old language

Still one of the most elegant, simple, uniform prue object-oriented language