Contents	About the speaker	What is interesting to learn?	Smalltalk	Squeak IDE	Conclusion	Resources
			0000000	00000000000		

# Squeak, learning oriented object programming with the right tool

Hilaire Fernandes

**CRDP** Aquitaine - OFSET

Fall 2005

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

Contents	About the speaker	What is interesting to learn?	Smalltalk	Squeak IDE	Conclusion	Resources
			0000000	00000000000		

▲ロト ▲冊ト ▲ヨト ▲ヨト - ヨー の々ぐ



- 2 What is interesting to learn?
- 3 Smalltalk
  - The language
  - A few examples
- 4 Squeak IDE
  - A tour in some developer tools
  - Programming scenarii

#### 5 Conclusion



Contents	About the speaker	What is interesting to learn?	Smalltalk 0000000	Squeak IDE 00000000000	Conclusion	Resources

- Working at CRDP Aquitaine France, a resource centre for pedagogical documentation and ICT in education. We do consulting, support and development for the educational institution.
- President of the OFSET organisation, to support free software development in education.

▲ロ ▶ ▲ 理 ▶ ▲ 国 ▶ ▲ 国 ■ ● ● ● ● ●

- Author of Dr.Geo, an interactive geometry software. It is a *massive* oriented object C++ software.
- ....and a seduced Squeak/Smalltalk developer

Contents	About the speaker	What is interesting to learn?	Smalltalk 0000000	Squeak IDE 00000000000	Conclusion	Resources
Focus	s on					

- Object programming concept
- Inheritance, polymorphism, attribute
- Writing proper object code:
  - subtypes instead of subclasses
  - is-a instead of part-of
  - tell don't ask (polymorphism vs test block)

▲ロト ▲冊ト ▲ヨト ▲ヨト - ヨー の々ぐ

- write reusable code
- Refactoring code
- Writing tests

Contents	About the speaker	What is interesting to learn?	Smalltalk 0000000	Squeak IDE 00000000000	Conclusion	Resources	
1 1 1	1 A A A A A A A A A A A A A A A A A A A	. <u>.</u>					

◆□▶ ◆□▶ ◆目▶ ◆目▶ 目 のへぐ

What are objects?

- An abstraction of a real world object with:
  - responsibilities
  - behaviours
- It is not a data structure

Contents	About the speaker	What is interesting to learn?	Smalltalk •000000	Squeak IDE 00000000000	Conclusion	Resources
Key p	oints					

• Everything is an object





- Everything is an object
- No trouble with type: *only* references to object. One may says Smalltalk is strongly typed.

Contents	About the speaker	What is interesting to learn?	Smalltalk •000000	Squeak IDE 00000000000	Conclusion	Resources	
Key p	points						

- Everything is an object
- No trouble with type: *only* references to object. One may says Smalltalk is strongly typed.
- The model is consistent and uniform thanks to its pure object aspect

▲□▶ ▲□▶ ▲三▶ ▲三▶ - 三 - のへで

Contents	About the speaker	What is interesting to learn?	Smalltalk •000000	Squeak IDE 00000000000	Conclusion	Resources
Key p	points					

- Everything is an object
- No trouble with type: *only* references to object. One may says Smalltalk is strongly typed.
- The model is consistent and uniform thanks to its pure object aspect

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● ● ●

• High level iterators

Contents	About the speaker	What is interesting to learn?	Smalltalk •000000	Squeak IDE 00000000000	Conclusion	Resources
Key p	points					

- Everything is an object
- No trouble with type: *only* references to object. One may says Smalltalk is strongly typed.
- The model is consistent and uniform thanks to its pure object aspect

▲ロ ▶ ▲ 理 ▶ ▲ 国 ▶ ▲ 国 ■ ● ● ● ● ●

- High level iterators
- Objects, messages and closures, that's it!

Contents	About the speaker	What is interesting to learn?	Smalltalk •000000	Squeak IDE 00000000000	Conclusion	Resources	
Key p	points						

- Everything is an object
- No trouble with type: *only* references to object. One may says Smalltalk is strongly typed.
- The model is consistent and uniform thanks to its pure object aspect

▲ロト ▲冊ト ▲ヨト ▲ヨト - ヨー の々ぐ

- High level iterators
- Objects, messages and closures, that's it!
- No *inextensible* operators

Contents	About the speaker	What is interesting to learn?	Smalltalk •000000	Squeak IDE 00000000000	Conclusion	Resources	
Key p	points						

- Everything is an object
- No trouble with type: *only* references to object. One may says Smalltalk is strongly typed.
- The model is consistent and uniform thanks to its pure object aspect
- High level iterators
- Objects, messages and closures, that's it!
- No inextensible operators
- $\bullet\,$  No public, protected or whatever object attributes  $\rightarrow\,$  only protected

▲ロ ▶ ▲ 理 ▶ ▲ 国 ▶ ▲ 国 ■ ● ● ● ● ●

Contents	About the speaker	What is interesting to learn?	Smalltalk 000000	Squeak IDE 00000000000	Conclusion	Resources

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● ● ●

#### The object model

- Everything is an object
- Only message sends and closures
- Public methods
- Protected attributes
- Single Inheritance
- Nothing special for static

Contents About the speaker What is interesting to learn? Smalltalk Squeak IDE Conclusion Resources

#### Smalltalk syntax in a postcard

#### exampleWithNumber: x

"A method that illustrates every part of Smalltalk method syntax except primitives. It has unary, binary, and keyword messages, declares arguments and temporaries, accesses a global variable (but not an instance variable), uses literals (array, character, symbol, string, integer, float), uses the pseudo variable true false, nil, self, and super, and has sequence, assignment, return and cascade. It has both zero argument and one argument blocks."

lyl

true & false not & (nil isNil) ifFalse: [self halt].
y := self size + super size.
#(\$a #a "a" 1 1.0) do:
 [:each | Transcript show: (each class name); show: ' '].
^ x < y</pre>

< □ > < 同 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ < ○ < ○ </p>



#### Simplicity to concentrate on what matter

• Single inheritence: easy to lookup for a method owner, it is in self or ancestors

▲ロト ▲冊ト ▲ヨト ▲ヨト - ヨー の々ぐ

- No trouble with type: only reference to object
- No trouble with attributes: all protected.
- No trouble with methods: all *public*.
- Garbage collector



# Coercion - implicit type conversion

Object can *mutate* when necessary, for example bellow a is a reference to a Fraction instance, then to a SmallInteger instance:

```
a := 1/3.
a class -> Fraction
a := a + (2/3)
a class -> SmallInteger
```



High level iterators are used with block closure.

• selection in a number collection:

```
#(1 2 3 4 5) select: [:i | i odd] -> #(1 3 5)
```

• calculus over a number collection:

```
#(1 2 3 4 5) collect: [:i | i * i ] -> #(1 4 9 16 25)
```

▲ロト ▲冊ト ▲ヨト ▲ヨト - ヨー の々ぐ

• transforming characters:

```
'taiwan' withIndexCollect:
  [:c :i|
  (i odd)
      ifTrue: [c asUppercase]
      ifFalse: [c]]
-> 'TaIwAn'
```

Contents	About the speaker	What is interesting to learn?	Smalltalk	Squeak IDE 00000000000	Conclusion	Resources

As usual block closure are object:

- It is anonymous method ( $\approx \lambda$ -function in Scheme).
- It can be passed to method as argument (see previous slide)
- It can be referenced by a variable and used as a function:
  - f := [:x| (x raisedTo: 3)]
  - f value: 5
    - -> 125
  - (1 to: 10) collect: [:x | f value: x]
    - -> #(1 8 27 64 125 216 343 512 729 1000)

▲ロ ▶ ▲ 理 ▶ ▲ 国 ▶ ▲ 国 ■ ● ● ● ● ●

• You will love it and you will use it a lot!



#### The code browsers

The code browser – or simply the browser – is a central tool in Smalltalk development. It greatly helps the developer to write and navigate the classes and methods.



Contents About the speaker What is interesting to learn? Smalltalk Squeak IDE Conclusion Resources

# SqueakMap package loader

A tool to install remote components, libraries or applications:

	This is where the selected
V. R. CourseleMan Deskan	package or package release
	e Loader (582/582) is displayed.
refactoring	Package name: Refactoring Browser for 3.8
-	version: 3.8.42
- Refactoring Browser for 3.4/	
-Refactoring Browser for 3.7	Categories:
- Refactoring Browser for 3.8	Squeak versions/Squeak3.8 - Released 25 May 2005.
- 3.8.42 *	Maturity level/Beta - Useable but still not stable probably
- 3.8.41	has bugs
- 3.8.40	Package format/Monticello - 4 ' mcz' file format for use
- 3.8.39	with Monticello. It is grinned
- 0.0.00	Competibility level/Code changes but only bug fives - Code
-3.8.12	has shaped but only with bug fives
-3.8.11	Competibility level. Code changes may breek competibility -
- 3.8.10	Code has shanged and may break alignts, but not necessarily
- 3.8.9	Licenses (MIT The MIT license is like PCD without the
- 3.8.8	Licenses/wir - The will license is nite bab without the
Refactoring Engine ()	advertising clause. As tree as it gets, suitable for cross smalltark
<ul> <li>ReflectiveDesigns ()</li> <li>Designation ()</li> </ul>	100% reuse.
- Registries ()	
- RemoteBroadcastingToolkit (R	
▶ PemoteFrameBuffer ()	Version Comment:
	Name: Refactory-md.3.8.42
Squeak versions	Author: md
Applications	Time: 26 July 2005, 12:19:20 pm
- LIASS IIDPATIES	UUID: 4566083e-9234-4341-a2c1-2f25dae45721
Development tools	Ancestors: Refactory-md.3.8.41
Entertainment	
- Licenses	-> Moved additions from AbstractString to String
Maturity level	-> Deleted RefactoryInfo
🕨 Package format 🛛 💌	-> comment for RBSmallDictionary



Squeak is able to inspect itself to search a method according to a pattern. Here with the pattern 'abcd'. 'bc'. 'ad' we can find the string methods removing a substring from a string:

× 🗉 Selector Browser	9 O
'abod', 'bc', 'ad'	*Collection copyWithoutAll:
"abcd' copyWithoutAll; 'bc'> 'ad' ■	v
Type a fragment of a selector in the to	p pane. Accept it.
Or, use an example to find a method in args, and answer in the top pane with 4. 7	the system. Type receiver, periods between the items. 3.
On in this cana use examples to find a	mothed in the avatem Colort 🗵



# The inspector

With the inspector, while your program is running, you can *very comfortably*:

- Inspect your classes and its attributes
- change your classes and attributes value while your programme is running





With the debugger you can debug your running code and also fix bugs and recompile *on the fly* the method. Then you resume your work. Your application does not need to be restarted. Bellow a halt point causes the debugger to open:



・ロト ・ 同ト ・ ヨト ・ ヨト ・ ヨー

Sac



The refactoring browser – RB

- Refactoring code is frequent, it means moving, changing section of code. Developer tools can assist the programmer to refactor code safely.
- The Squeak RB extents the classic browser with code refactoring facilities. It helps – among other things – to safely rename instance variables and methods. It checks down the subclasses to rename the variable when necessary.

▲ロ ▶ ▲ 理 ▶ ▲ 国 ▶ ▲ 国 ■ ● ● ● ● ●



# SUnit test runner

Unit tests help to automatically check validity of some piece of code (i.e. ensuring it effectively do the right computation) Here a set of tests run over an application library:



・ロト ・ 雪 ト ・ ヨ ト

э

Contents	About the speaker	What is interesting to learn?	Smalltalk 0000000	Squeak IDE	Conclusion	Resources	
The o	central too						

The browser is your central development tool:

• to write new system categories and define classes, methods

▲ロト ▲冊ト ▲ヨト ▲ヨト - ヨー の々ぐ

- to explore the hierarchies of the classes
- to explore implementors and senders of methods
- to explore the inheritance of your methods

Contents	About the speaker	What is interesting to learn?	Smalltalk 0000000	Squeak IDE	Conclusion	Resources	
	100 C 100 C						

I he central tool

The browser is your central development tool:

• to write new system categories and define classes, methods

▲ロト ▲冊ト ▲ヨト ▲ヨト - ヨー の々ぐ

- to explore the hierarchies of the classes
- to explore implementors and senders of methods
- to explore the inheritance of your methods

Let's look at a live example...



#### Incremental programming

Incremental programming is about smooth step by step programming:

- Open a workspace to experiment your *pieces* of codes
- The **debugger** will open when necessary (faulty code or breakpoint), you will fix the code and *continue* your experiment
- With the **inspector** examine the attribute values of your classes

▲ロ ▶ ▲ 理 ▶ ▲ 国 ▶ ▲ 国 ■ ● ● ● ● ●



#### Incremental programming

Incremental programming is about smooth step by step programming:

- Open a workspace to experiment your *pieces* of codes
- The **debugger** will open when necessary (faulty code or breakpoint), you will fix the code and *continue* your experiment
- With the **inspector** examine the attribute values of your classes

▲ロ ▶ ▲ 理 ▶ ▲ 国 ▶ ▲ 国 ■ ● ● ● ● ●

Let's look at a live example...



#### Learning from the class library

When you want to use an object but you don't know how, just use the browser to explore the object and look at its methods. Here exploring the system class **Date**:



A D > A D > A D > A D >

Sac



#### Learning from the class library

When you want to use an object but you don't know how, just use the browser to explore the object and look at its methods. Here exploring the system class **Date**:



A D > A D > A D > A D >

Sac

#### Let's look at a live example...



Beside saving the whole image, it is easy to save the specific code we are working on with the **Monticello browser**. When writing code, take care to:

- Define the *Toto* application classes in a *Toto-\** categories (i.e. *Toto-view*, *Toto-model*, change *Toto* with your application name)
- When you need to add methods to a system class, define in this class a *\*Toto-myCategory* and define your methods inside.

▲ロ ▶ ▲ 理 ▶ ▲ 国 ▶ ▲ 国 ■ ● ● ● ● ●

• Use the **Monticello browser** to save your code, on disk or remotely.

For more information read: http://www.iam.unibe.ch/~scg/ Teaching/AdvancedLabs/PDFs/squeaktools.pdf



Beside saving the whole image, it is easy to save the specific code we are working on with the **Monticello browser**. When writing code, take care to:

- Define the *Toto* application classes in a *Toto-\** categories (i.e. *Toto-view*, *Toto-model*, change *Toto* with your application name)
- When you need to add methods to a system class, define in this class a *\*Toto-myCategory* and define your methods inside.

▲ロ ▶ ▲ 理 ▶ ▲ 国 ▶ ▲ 国 ■ ● ● ● ● ●

• Use the **Monticello browser** to save your code, on disk or remotely.

For more information read: http://www.iam.unibe.ch/~scg/ Teaching/AdvancedLabs/PDFs/squeaktools.pdf

Let's look at a live example...



- Squeak is a virtual machine VM based environment.
- The VM is fed with a multi-platform image file composing the environment.
- It is based and written on Smalltalk, a pure object oriented language.
- Squeak offers a large range of high level developer tools
- Squeak offers new paradigm in software development as incremental-compilation.

Contents	About the speaker	What is interesting to learn?	Smalltalk 0000000	Squeak IDE 00000000000	Conclusion	Resources
The f	uture					

- The Squeak foundation http://smallwiki.unibe.ch/SqueakFoundation - a legal entity to support the Squeak promotion.
- Seaside http://www.seaside.st it is a frame work to develop high level web application within the Squeak environment. Basicly your are writting web application as you are developping event controled desktop application plus you take benefice of the Squeak IDE to develop and debugs. ASTONISHING!
- Tweak http://tweak.impara.de a rewrite of the graphic morphic interface.
- wxSqueak http://www.wxsqueak.org a wxWidget wrapper for Squeak to write multiplateform application with native look and fell.
- Croquet http://www.opencroquet.org a 3D peer-to-peer authoring environment.

Contents	About the speaker	What is interesting to learn?	Smalltalk 0000000	Squeak IDE 00000000000	Conclusion	Resources

#### Academic resources

- http://www.iam.unibe.ch/~ducasse/Teaching.html
- http://www.eli.sdsu.edu/courses
- http://prog.vub.ac.be/~tjdhondt/POOL/HTM.dir/ introduction.htm
- http://wiki.cs.uiuc.edu/VisualWorks/VisualWorks+in+ Education

• http://www.whysmalltalk.com/universities/

Contents	About the speaker	What is interesting to learn?	Smalltalk 0000000	Squeak IDE 00000000000	Conclusion	Resources

# Other resources in the net

- http://www.squeak.org
- Free Smalltalk books: http://www.iam.unibe.ch/~ducasse/FreeBooks.html

◆□▶ ◆□▶ ◆ □▶ ◆ □▶ - □ - のへぐ

• http://www.whysmalltalk.com