1 History

- During the 70’s Alan Kay worked on the Dynabook at Xerox Parc. A lot like today’s laptops, with integrated touch screen, sound, networking.

- Users would need some sort of programming skills to fully utilize the system. A language for non-experts was needed.

- Smalltalk borrows from Simula, Logo (a language for children), and Sketchpad (a constraint-based interactive drawing system).

- Smalltalk was the first “pure” object-oriented language. Every interaction is through sending a message to an object.

2 Running Smalltalk

- On lectura, do the following:

  > cp /usr/local/lib/squeak/3.2-5/Squeak3.2-4956.* .
  > setenv SQUEAK_IMAGE $PWD/Squeak3.2-4956.image
  > /usr/local/lib/squeak/3.2-5/squeak

3

- Squeak’s start screen:
• Get rid of the crud:

• Open the class browser:
6

- **Workspace** lets you enter commands interactively. **Transcript** is “standard output.” **do it** executes highlighted code.

7

- Create a new category **cc**.
Create a new class **MyClass**. Select **accept** to add it.

Click on **no messages** to get a message template.
10

- Create a method `square`. Select `accept` to add it.

11

- Execute `square`. 
12 Syntax

- Square brackets [...] contain code.
- Global items (variables, classes) begin with a capital letter. Other items start with lowercase.
- Temporary variables: | x y z |.
- Assignment: ← or :=, or type as _.
- Return value: ↑, type as ^.
- The dot (.) is the statement terminator.

13 Syntax — Unary Messages

- A message $M$ is sent to an object (receiver) $R$ using the syntax
  
  \[ R \ M \]

- A unary message has the syntax
  
  \[ R \ M \]

For example:

\[ D \leftarrow \text{Dictionary new.} \]

14 Syntax — Binary Messages

- A binary message $M$ to receiver $R$ with argument $A$ has the syntax
  
  \[ R \ M \ A \]

- For example:
  
  \[ 8 + 9 \]

This sends the message + to the object 8 with the argument 9.
15 Syntax — Keyword Messages

- A keyword message $M$ to receiver $R$ with arguments $A_1, A_2, A_3, \ldots$ has the syntax:

$$ R \ M_1 \ A_1 \ M_2 \ A_2 \ M_3 \ A_3 \ldots $$

- For example:

```
DeannaTroi kiss: cheek how: tenderly
```

This sends the message `kiss:how:` to the object `DeannaTroi` with the arguments `cheek` and `tenderly`. In Java we would have written:

```
DeannaTroi.kisshow(cheek,tenderly)
```

16 Syntax — Order of Evaluation

- Messages are executed from left to right.
- Parentheses can be used to force a particular order of evaluation.
- Expressions are executed in the order:
  1. unary messages,
  2. binary messages,
  3. keyword messages
- Binary messages are executed left to right.

17 Syntax — Cascading messages

- Often we want to send several messages $M_1, M_2, \ldots$ to the same receiver. We can use the syntax:

$$ R \ M_1: A_1.
R \ M_2: A_2.
R \ M_3: A_3. $$

- Or, we can cascade the messages using a semicolon (\(;\)):

$$ R \ M_1: A_1; M_2: A_2; M_3: A_3\ldots $$

- For example:

```
Transcript show:5; cr; show:9; cr.
```

18 Syntax — Blocks

- A block is similar to a lambda expression. It’s syntax is:

```
[arguments | code]
```

- Arguments are prefixed by a colon (\(:\)):

```
[:x :y | ↑ x+y ]
```
• Collections: Dictionary.

• Collections: Bag. 3 timesRepeat (from class Integer) sends the value: message to the block argument 3 times.

• expression timesRepeat block sends the value: message to block as long as expression is true.
22

- **do**: `aBlock` enumerates all the receiver's elements.

23

- **ifTrue:ifFalse**: evaluates one of its blocks depending on the value of the receiver.
26   Readings and References

- Squeak download: http://www.squeak.org/download/
- Squeak documentation: http://www.squeak.org/documentation/index.html
- http://www.cosc.canterbury.ac.nz/~wolfgang/cosc205/smalltalk1.html#1ink