1

Smalltalk Environment Basics

Main Author(s): Ducasse and Wuyts

1.1 Starting up

Similarly to Java, Smalltalk the source code is translated to byte-codes, which are then interpreted and executed by the Smalltalk Virtual Machine. (Note that this is an approximation because Smalltalk dialects were also the first languages to develop Just in Time compilation, i.e., a method is compiled into byte-codes but also into native code that is directly called instead of executing the byte codes.)

There are three important files:

visual.im (Binary): contains byte code of all the object of the system, the libraries and the modifications you made.

visual.cha (ASCII): contains all the modifications made in the image-file since this was created. It contains also all the code of the actions you will perform.

visual.sou (ASCII): contains the textual code of the initial classes of the system.

To open an image with drag-drop (on Macintosh, Windows):

- Drag the file 'visual.im' on the virtual machine to start the image.
- If you want to start your own image, just double click on it or drag it over the virtual machine.

On Unixes (Linux, Solaris, HPUX, AIX, ...): you should invoke the virtual machine passing it an image as parameter. For the first opening, execute the first script that per default uses the original image the script is installation dependent but should look like path/bin/visualworks path/image/visual.im Then after you can specify your own image.

1.2 First Impressions

After opening the image, and thus starting a Smalltalk session, you see two windows : the launcher (with menu, buttons and a transcript), and a Workspace window (the one containing text). You can minimize or close this last one, since we do not need it for the moment.

The launcher is the starting point for working with your environment and for the opening of all the programming tools that you might need. To begin, we will first create a fresh image.

Creating a fresh image. We are going to create an image for this lesson.

- select Save As... in the menu
- when the system prompts you for the name for the new image, you type *lesson*, followed by your username.
- the image is saved in the image directory.
- Have a look at the Transcript, and note what it says.

The Transcript is the lower part of the Launcher, and gives you system messages, like the one you see right now. We will see later on how you can put your own messages there.

About the mouse. Smalltalk was the first application to use multiple overlapping windows and a mouse. It extensively uses three mouse buttons, that are context sensitive and can be used everywhere throughout Smalltalk:

- the left mouse button is the select button
- the middle button is the operate button
- the right button is the window button

On a Macintosh, where only one button is available, you have to use some keyboard keys together with pressing your mouse button:

- the select button is the one button itself
- for the operate button, press the button while holding the alt-key pressed
- for the window button, press the button while holding the apple-key pressed

1.3 Adding Goodies and setting Preferences

Out of the box, there is already quite some code in a Smalltalk image (about 1000 classes containing the basic system: the complete compiler, parser classes, GUI framework, development environment, debugger, collection libraries, etc.) [1]. But for developing, it is convenient to load some extra tools.

You may have to load a package named *ImageConfiguration* or parcels. To load parcels we can use the *Parcel Manager* application (open it using the System menu in the Launcher). There is lots of optional applications you might load. Use it to load:

- *RBSUnitExtentions* (in *Environment Enhancements*)
- *MagicKeys* (in *Environment Enhancements*)
- *ColorEditing* (in *Environment Enhancements*)
- *RB_Tabs* (in *Environment Enhancements*)

You can also edit systemwide preferences. To do so, open the *Settings* application, again in the System menu in the Launcher. Use it to set a setting in Tools/Browser: select *Show all methods when no protocols are selected*.

1.4 Selecting text, and doing basic text manipulations

One of the basic manipulations you do when programming is working with text. Therefore, this section introduces you to the different ways you can select text, and manipulate these selections.

The basic way of selecting text is by clicking in front of the first character you want to select, and dragging your mouse to the last character you want in the selection while keeping the button pressed down. Selected text will be highlighted.

Exercise 0 Select some parts of text in the Transcript. You can also select a single word by double clicking on it. When the text is delimited by ” (single quotes), ”” (double quotes), () (parentheses), [] (brackets), or (braces), you can select anything in between by double clicking just after the first delimiter.

Exercise 1 Try these new selection techniques.

Now have a look at the text operations. Select a piece of text in the Transcript, and bring on the operate menu. Note that you have to keep your mouse button pressed to keep seeing the window.

[1] To answer a common question: yes, there are ways to strip this so that you can deploy smaller images to clients that do not contain all of these tools.

Exercise 2 Copy this piece of text, and paste it after your selection. Afterwards cut the newly inserted piece of text.

Exercise 3 See if there is an occurrence of the word visual in the Transcript. Note that to find things in a text window, there is no need to select text. Just bring up the operate menu .

Exercise 4 Replace the word visual with C++ using the replace operation (if it does not contain Smalltalk, add this word or replace something else). Take your time and explore the different options of the replace operation.

Exercise 5 Bring up the operate menu, but don't select anything yet. Press and hold the shift button, and select paste in the operation menu. What happens ?

1.5 Opening a WorkSpace Window

We will now open a workspace window, a text window much like the Transcript, you use to type text and expressions and evaluate them.

To open a workspace:

- select the tools menu in the Launcher
- from the tools menu, select Workspace
- You will see a framing rectangle (with your mouse in the upper left corner), that indicates the position where the Workspace will open. Before you click, you can move your mouse around to change this position. Click one time once you have found a good spot for your Workspace.
- Now your mouse is in the bottom right corner, and you can adjust the size. If you click once more, once you have given it the size you like, the Workspace window appears.

This is the basic way of opening many kinds of applications. Experiment with it until you feel comfortable with it.

1.6 Evaluating Expressions

In the Workspace, type : `3`. Select it, and bring up the operate menu. In the operate you will see the next three different options for evaluating text and getting the result:

do it: do it evaluates the current selection, and does not show any result of the execution result.

print it: prints the result of the execution after your selection. The result is automatically highlighted, so you can easily delete it if you want to.

inspect it: opens an inspector on the result of the execution.

The distinction before these three operations is essential, so check that you REALLY understand their differences **Exercise 6** Select 3, bring up the operate menu, and select print it.

Exercise 7 Print the result of `3+4`

Exercise 8 Type `Date today` and print it. Afterwards, select it again and inspect it.

After exercise 9, you will have an inspector on the result of the evaluation of the expression Date today (this tells Smalltalk to create an object containing the current date). This Inspector Window consists of two parts: the left one is a list view containing self (a pseudo variable containing the object you are inspecting) and the instance variables of the object. Right is a text field.

Exercise 9 Click on `self` in the inspector. What do you get? Does it resemble the result shown by printstring?

Exercise 10 Select day. What do you get? Now change this value, bring up the operate menu, and select accept it. Click again on self. Any difference?

Exercise 11 In the inspector edit field, type the following: self weekday, select it and print it. This causes the message weekday to be sent to self (i.e., the date object), and the result is printed. Experiment with other expressions like:

```
self daysInMonth
self monthName
```

Close the inspector when you are finished.

Exercise 12 Type in the Workspace the following expression: Time now, and inspect it. Have a look at self and the instance variables.

Exercise 13 Type in the Workspace the following expression: Time dateAndTimeNow. This tells the system to create an object representing both today's date and the current time, and open an inspector on it. Select the item self in the inspector. [Note that self is an object called an Array. It holds on to two other objects (elements 1 and 2). You can inspect each element to get either the time or the date object.

Using the System Transcript. We have already seen that the Transcript is a text window where the system informs you important information. You can also use the Transcript yourself as a very cheap user interface.

If you have a Workspace open, place it so that it does not cover the System Transcript. Otherwise, open one and take care of where you put it. Now, in the Workspace, type:

```
Transcript cr.
Transcript show: 'This is a test'.
Trancript cr.
```

Select these 3 lines and evaluate (do It) them with do it. This will cause the string This is a test to be printed in the Transcript, preceeded and followed by a carriage return. Note that the argument of the show: message was a literal string (you see this because it is contained in single quotes). It is important to know, because the argument of the show: method always has to be a string. This means that if you want any non-string object to be printed (like a Number for example), you first have to convert it to a string by sending the message printString to it. For example, type in the workspace the following expression and evaluate it:

Transcript show: 42 printString, 'is the answer to the Universe'. Note here that the comma is used to concatenate the two strings that are passed to the show: message 42 printString and 'is the answer to the Universe'.

Exercise 14 Experiment on your own with different expressions. Transcript cr ; show: This is a test ; cr Explain why this expression gives the same result that before. What is the semantics of ; ?