Exploratory programming using Squeak and Morphic

Keunwoo Lee
CSE 341 -- Programming Languages
University of Washington
Dept. of Computer Science and Engineering

Exploratory programming

- Programming by "trying stuff out" and seeing what happens
- Slow, cumbersome in edit/compile/run loop
- Easier in read/eval/print loop (fast feedback)
- Squeak & Morphic have even more advanced support for editing code, manipulating objects interactively...
  - Inspector/explorer
  - Selector browser
  - Stack trace debugger (for exceptions)
  - Tile-based scripting

Review: FishMorph

Creating morphs
The debug halo menu

Inspecting instance variables

- Can view, edit values in-place (write expression in value display pane and accept (Alt-s))
- Yellow-click to bring up context menu for instance var

The inspector

Inspector mini-workspace

- Bottom pane behaves as object-specific workspace
- Workspace's environment is like no-argument method:
  - instance variables accessible
  - self bound to object
  - super begins lookup in superclass of self's class
Inspector: uses

- Try out code in object workspace; copy into a method when you've got what you want.
- Use context menu to explore instance vars
  - "This value shouldn't be here! How did this get set?"
    --> use methods storing into this inst var
  - "What is this field's class? What methods does its class handle?"
    --> use browse class or browse hierarchy

Selector browser (method finder)

- Often want to know who handles a message
  e.g., when you see a message send and want to know who the receiver might be
- With selector browser, can search for all implementors of a method
  (among other things; read docs in bottom pane)

Explorer

- Displays object graph as tree (fields as children)
- Bottom pane is also workspace (for selected item)

Smalltalk exceptions

- raising: Exception methods signal, signal:
  Exception subclass: #NotFound ...
- handling: BlockContext method on:do:
  [ aTree find: [:x | x > 0 ]
    ifAbsent: [ Not-found new signal: 'no positives!' ] ]
  on: Not-found
  do: [:x => Transcript show: 'got Not-found exception'; cr. ].

Note: Exception defines class methods signal and signal:; can usually just send signal messages to Exception subclasses directly:

    Not-found signal: 'no positives!'
Stack trace debugger

- Unhandled exceptions propagate to "top level", where the inspector is invoked.

(Notice Workspace context is UndefinedObject>>Dolt)

---

Inspecting the stack

- Execution stack prior to signal
  (Selected stack frame in red)

---

Interactive stack debugging

- Can edit code of methods directly in debugger
- Use "accept" (Alt-s) to save changes
- Can restart message send, step through evaluation

---

Tile-based scripting

- Tiles: graphical representations of Squeak objects and code
- Use Viewer halo to obtain tile scripting interface for a morph

---
Script categories

Making scripts

- Drag from script tile to start a script:

- Dropping onto desktop makes standalone script:

- Drag from empty script to start with no code:

Editing scripts

- To change name, click on title

- To add lines, drop more tiles onto script

Editing scripts, ct'd.

- Click on morph name to get menu:

- Can get tiles from here, or many other places (e.g., Tile halo of other Morph)
**Assembled scripts**

![Image of assembled scripts]

(button made with button to fire this script selection of script menu)

**Running scripts**

- To execute, click exclamation point button
- Can set to run on different events --- click button next to clock to edit:
  - ticking runs repeatedly at clock ticks
  - mouse* events run on events
  - click what do these mean? for more info

**Textual script editing**

- Can always "drop down" into text-based code for Morphic scripts:
  ![Image of textual script editing]

- Useful for more sophisticated coding
- Also, can copy & paste script into method once script is debugged & mature

**Conclusion**

- emacs, Eclipse, and Visual Studio are not the last word in programming environments
- Demand more!
- You can build your own "inspector"-like programs for exploring objects in other languages/environments e.g.:
  - XML-RPC is recently developed protocol for objects communicating over network
  - Easy to build an XML-RPC inspector so you can interactively send messages, receive replies