SqueakJS

A Decade of Progress

Vanessa Freudenberg, ESUG 2025

What is SqueakJS?

- A Virtual Machine for the Squeak family of Smalltalks
- Implemented in JavaScript, runs in Web Browsers (even on phones)
- Aims at being the most compatible VM, able to run any image without modification
- Also practical for new applications (e.g. via JavaScript Bridge)
- Open-source since 2013 with many contributors
- Employed by Dan Ingalls' Smalltalk Zoo accompanying his HOPL paper

Welcome to the Smalltalk Zoo
Curated by Dan Ingalls

What is it?

Here you will find various Smalltalk stories and artifacts that I have collected from years of building Smalltalk systems at Xerox, Apple, HP, and Disney. This collection began as background material for a paper that I wrote for the ACM's History of Programming Languages conference in 2020. In addition to various papers, there are simulations of several generations of Smalltalk that you can run right here in your browser. This will give you some sense of how the user interface and programming tools evolved along with the progress of the language.

Papers

The Smalltalk-76 Programming System - My original paper on Smalltalk-76. Describes the syntax. object model, and design of the byte-coded virtual machine. Includes a sketch of the vm written in Smalltalk-76 itself.
Back to the Future: the Story of Squeak - Describes the design of Squeak, a commercial-grade Smalltalk written in itself and made practical by compiling a subset of the language ("Slang") to C.
The Evolution of Smalltalk from Smalltalk-72 through Squeak - A long paper (100 pages!) that I wrote for HOPL-2020. This version includes corrections, and is a mixture of stories, technical details, and links to simulations and related work.

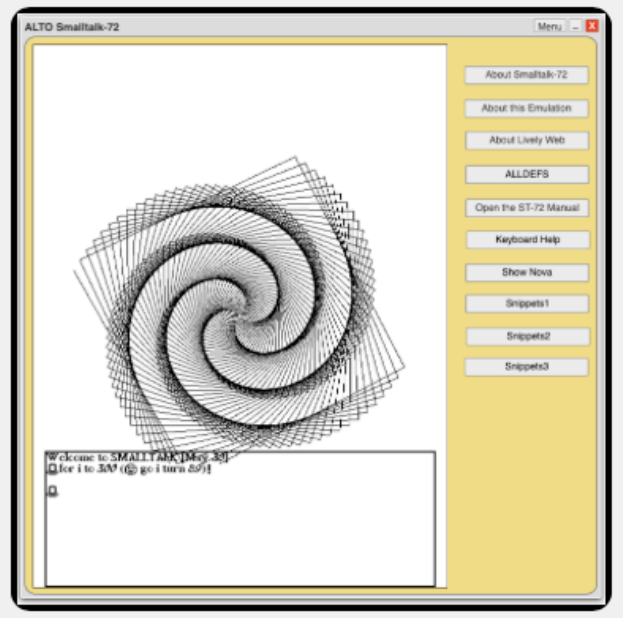
Simulations

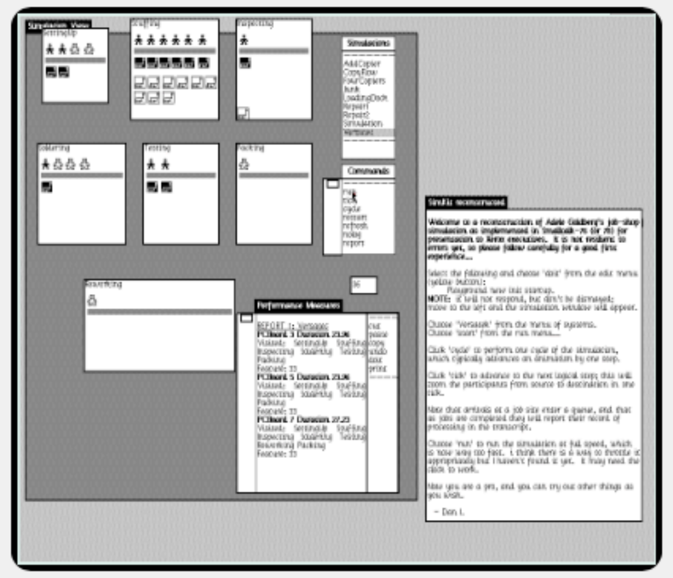
AltoSmalltalk-72 - JavaScript emulating Nova machine code running an original Smalltalk save file from an Alto
Smalltalk-78 - Smalltalk-78, a variant of Smalltalk-76 with in-line Contexts
Smalltalk-80 - Xerox and Apple versions of Smalltalk-80 as released
Squeak - Squeak Smalltalk with several graphic and sound demos
Etoys - Squeak was designed to be a vehicle for multimedia projects with iconic scripting on top of Smalltalk

Bootstrap Files

Most of these are preserved only as scanned printouts, though someone who is good at OCR could probably recover the text without too much trouble. Bootstraps are a place to start if you're curious about the beginnings of each Smalltalk generation, or if you have a cool idea for an interpreter, but want some sort of starting library

ALLDEFS - Actual text of the bootstrap definitions for Smalltalk-72
MDefs - Scan of a similar bootstrap for St-74
launch.ft - Actual text of the last part of the St-74 bootstrap, analogous to the last part of ALLDEFS


Alto Smalltalk-72


Smalltalk-78 (NoteTaker)

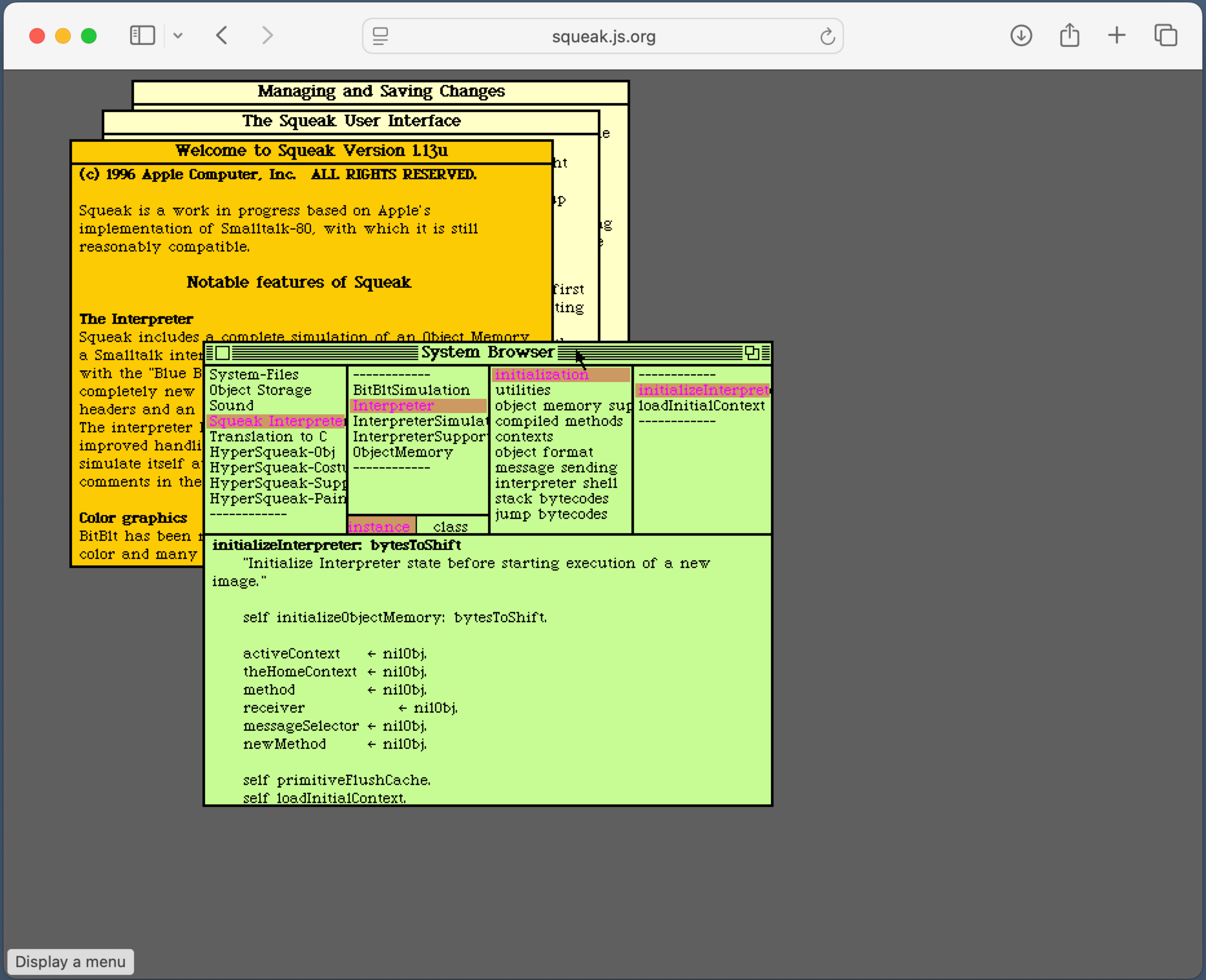
Squeak (Smalltalk-96)

SqueakJS: A Decade of Progress

2013: Beginnings

- DLS paper at SPLASH 2014 in Portland Oregon (won “Most Notable Paper” Award in 2024)
- Based on Potato (Dan Ingalls’ Java VM)
- Innovative hybrid GC (VM manages old space, and JavaScript GC collects new space)
- Extensible via plugins
- Snapshots compatible with standard VM
- Could run “old” Squeak images like Etoys and Scratch as well as then-current Squeak 4.5

Squeak 1.13



Managing and Saving Changes

The Squeak User Interface

Welcome to Squeak Version 1.13u

(c) 1996 Apple Computer, Inc. ALL RIGHTS RESERVED.

Squeak is a work in progress based on Apple's implementation of Smalltalk-80, with which it is still reasonably compatible.

Notable features of Squeak

The Interpreter
Squeak includes a complete simulation of an Object Memory
a Smalltalk inte
with the "Blue B
completely new
headers and an
The interpreter
improved handli
simulate itself a
comments in the

Color graphics
BitBlt has been
color and many

System Browser

System-Files	------------	initialization	------------
Object Storage	BitBltSimulation	utilities	initializeInterpret
Sound	Interpreter	object memory sup	loadInitialContext
Squeak Interprete	InterpreterSimulat	compiled methods	------------
Translation to C	InterpreterSuppor	contexts	
HyperSqueak-Obj	ObjectMemory	object format	
HyperSqueak-Costu	------------	message sending	
HyperSqueak-Sup		interpreter shell	
HyperSqueak-Pain		stack bytecodes	
------------	instance class	jump bytecodes	

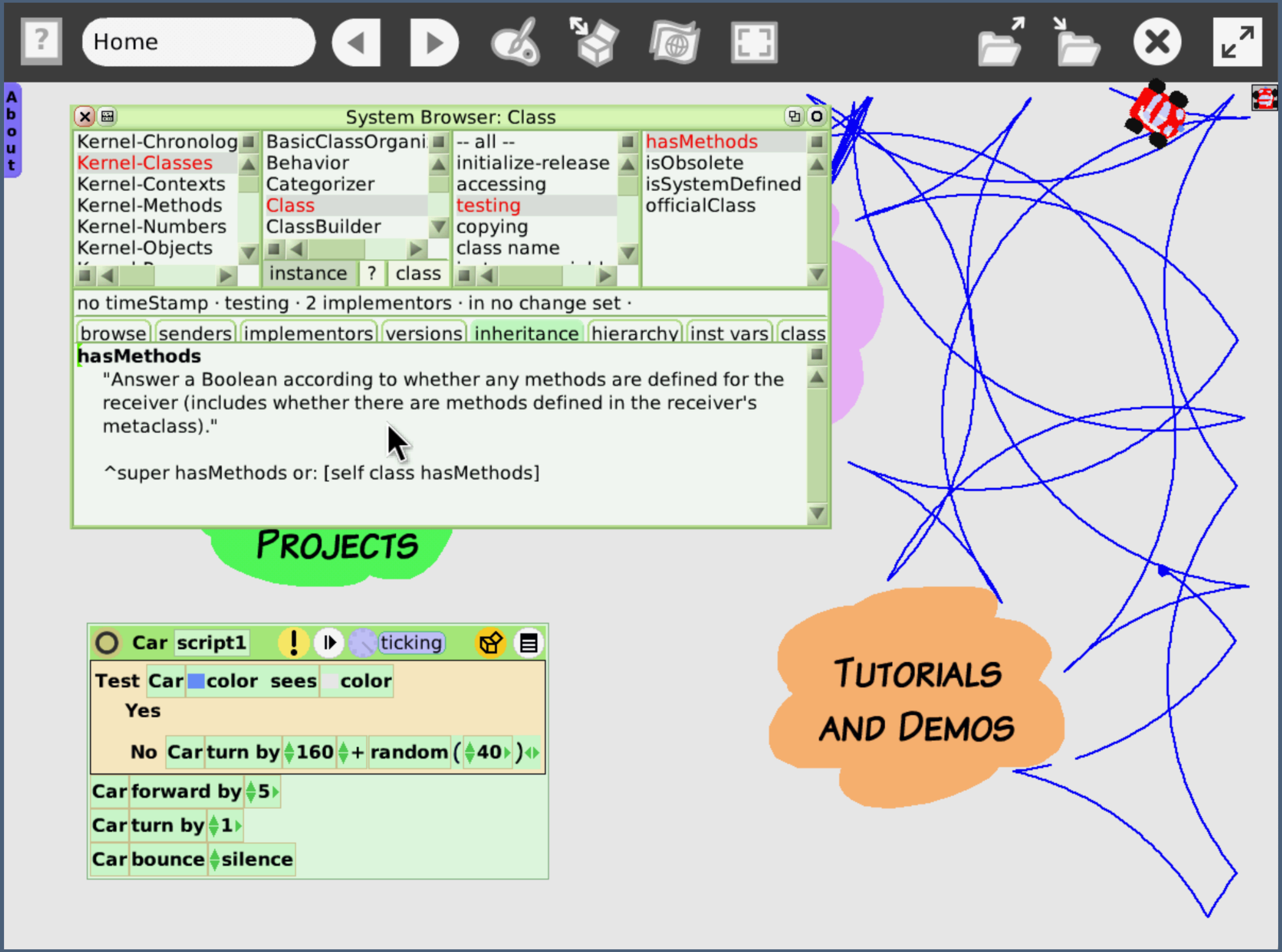
```
initializeInterpreter: bytesToShift
    "Initialize Interpreter state before starting execution of a new
image."

    self initializeObjectMemory: bytesToShift.

    activeContext   ← nilObj.
    theHomeContext ← nilObj.
    method          ← nilObj.
    receiver             ← nilObj.
    messageSelector ← nilObj.
    newMethod       ← nilObj.

    self primitiveFlushCache.
    self loadInitialContext.
```

Etoys



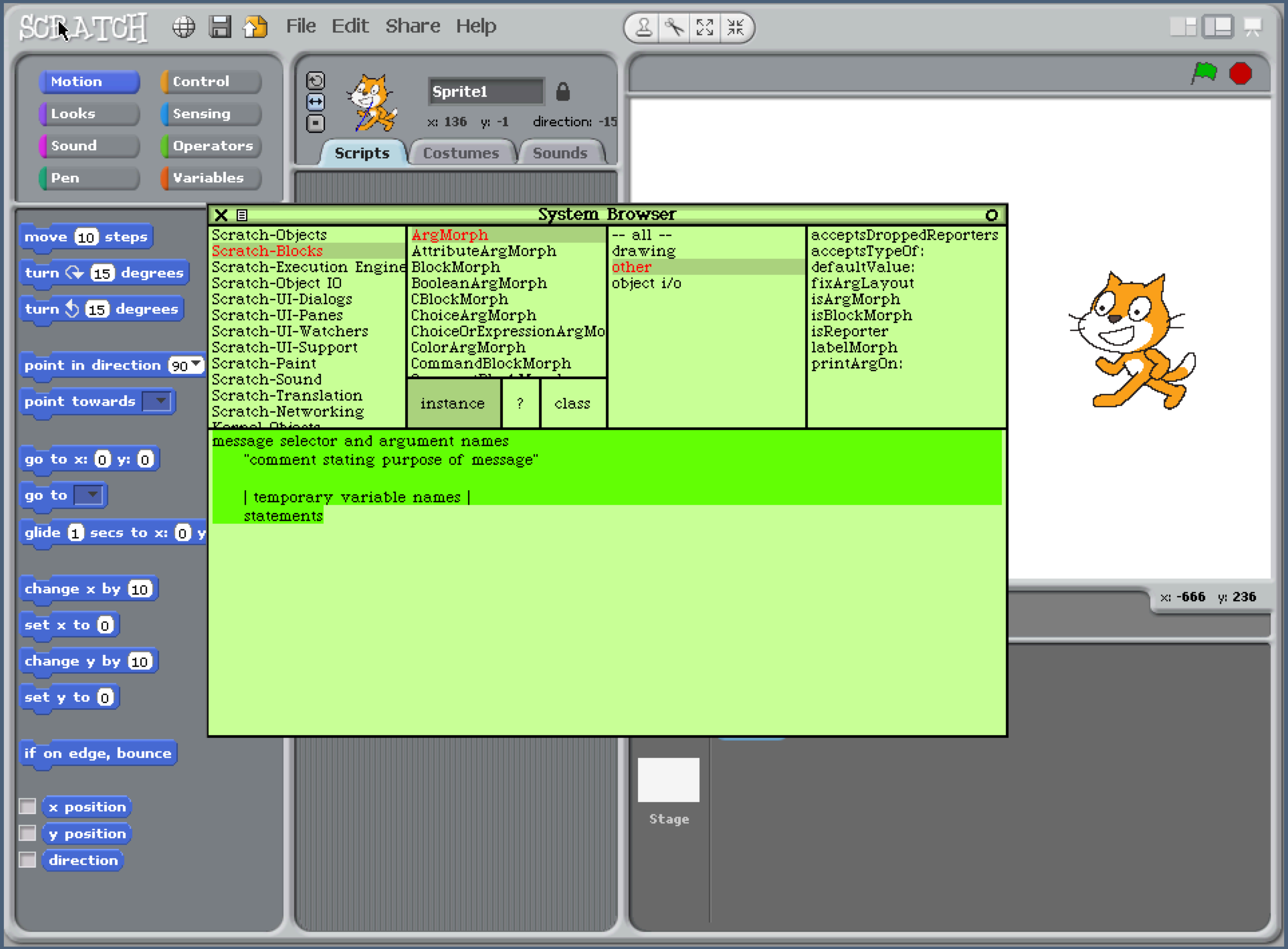
no timeStamp · testing · 2 implementors · in no change set ·

```
hasMethods
    "Answer a Boolean according to whether any methods are defined for the
    receiver (includes whether there are methods defined in the receiver's
    metaclass)."

    ^super hasMethods or: [self class hasMethods]
```

```
Test Car color sees color
  Yes
  No Car turn by 160 + random ( 40 )
Car forward by 5
Car turn by 1
Car bounce silence
```

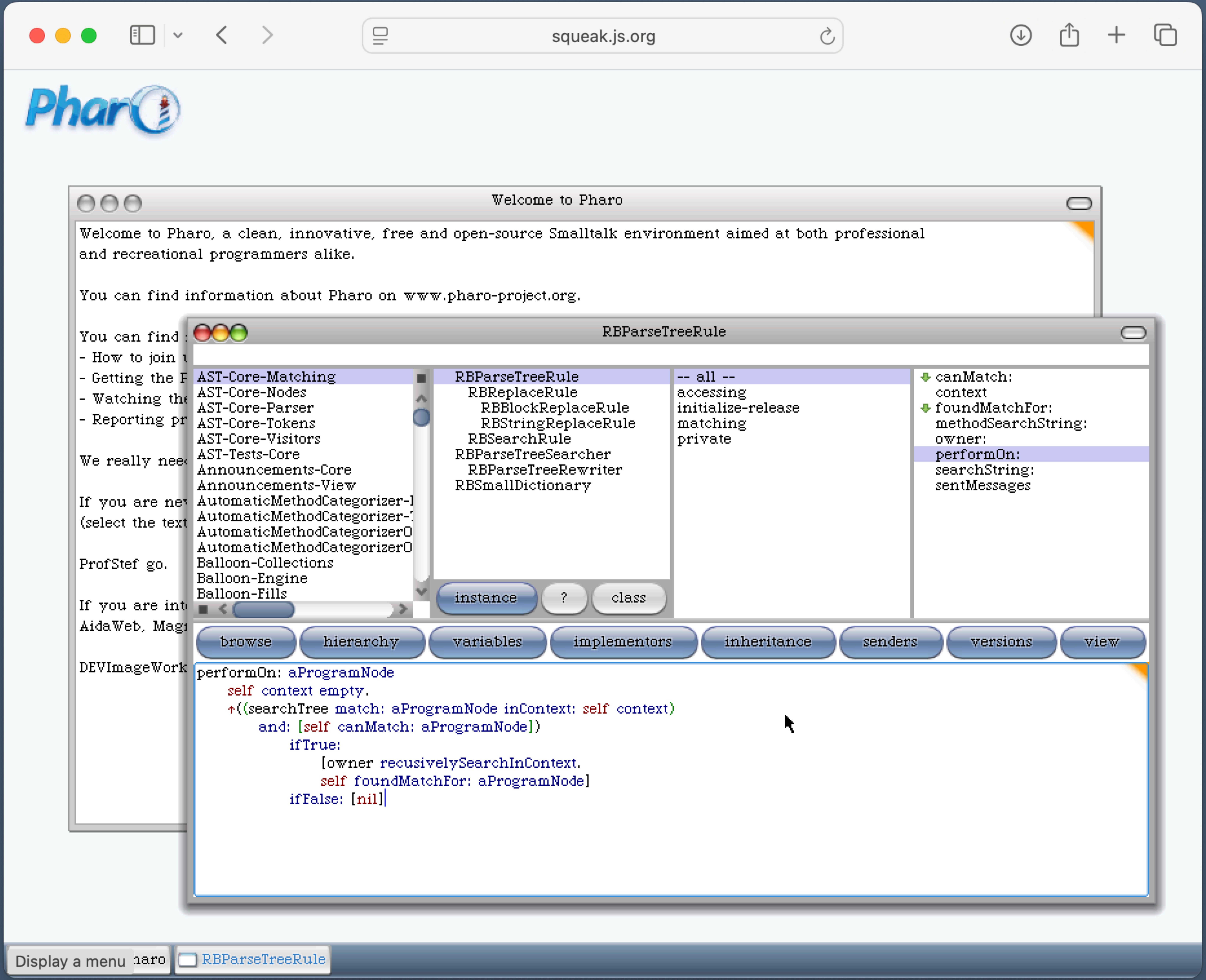
Scratch 1.4



```
message selector and argument names
    "comment stating purpose of message"

    | temporary variable names |
    statements
```

Pharo 1.0



Welcome to Pharo, a clean, innovative, free and open-source Smalltalk environment aimed at both professional and recreational programmers alike.

You can find information about Pharo on www.pharo-project.org.

You can find
- How to join
- Getting the
- Watching the
- Reporting pr

We really nee

If you are ne
(select the text

ProfStef go.

If you are int
AidaWeb, Mag

DEVImageWork

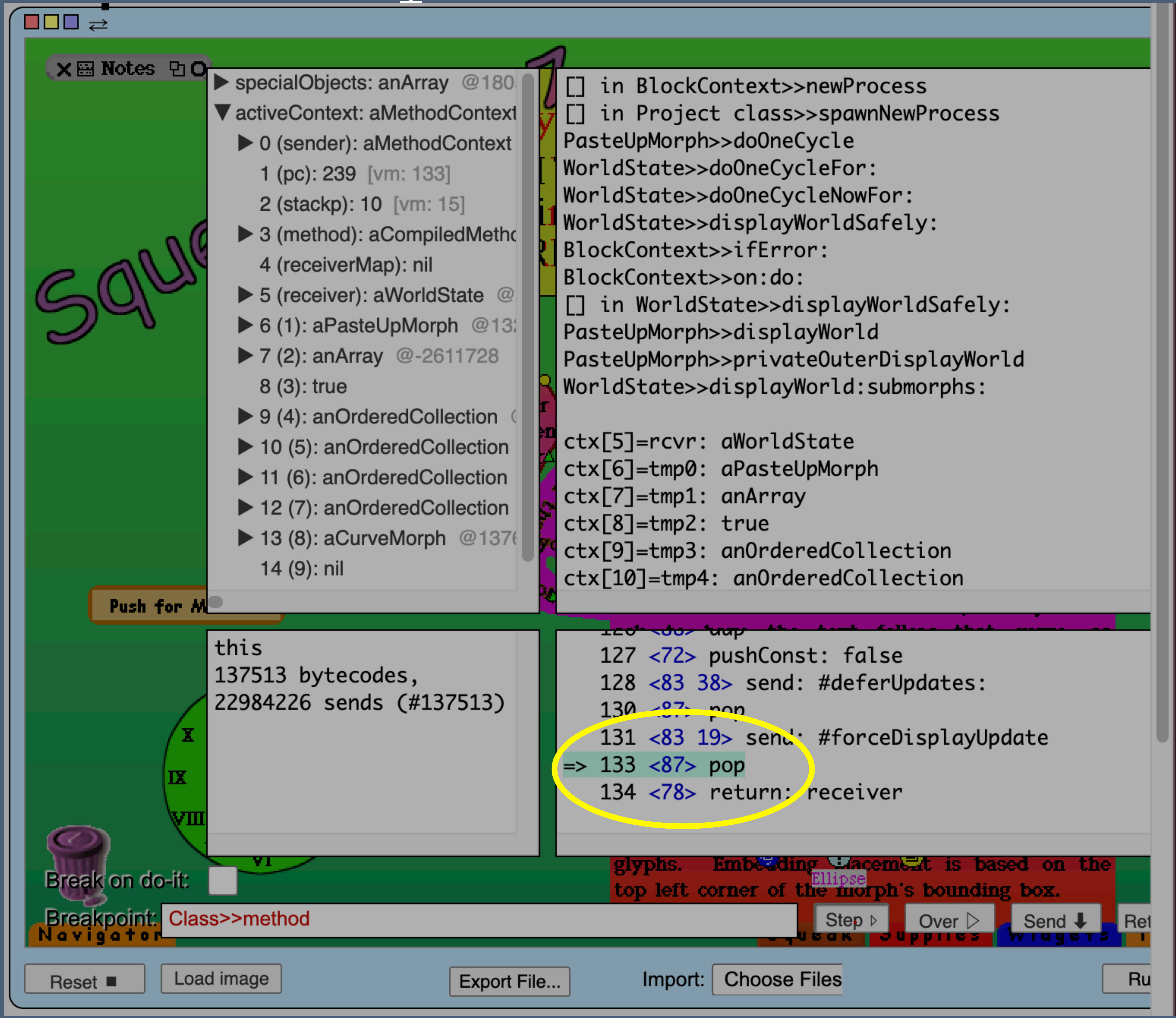
```
performOn: aProgramNode
	self context empty.
	↑((searchTree match: aProgramNode inContext: self context)
		and: [self canMatch: aProgramNode])
			ifTrue:
				[owner recusivelySearchInContext.
				self foundMatchFor: aProgramNode]
			ifFalse: [nil]
```

SqueakJS: A Decade of Progress

2014

- Closures
- JIT compiler
- Weak refs and finalization
- Javascript Bridge

JIT Compiler



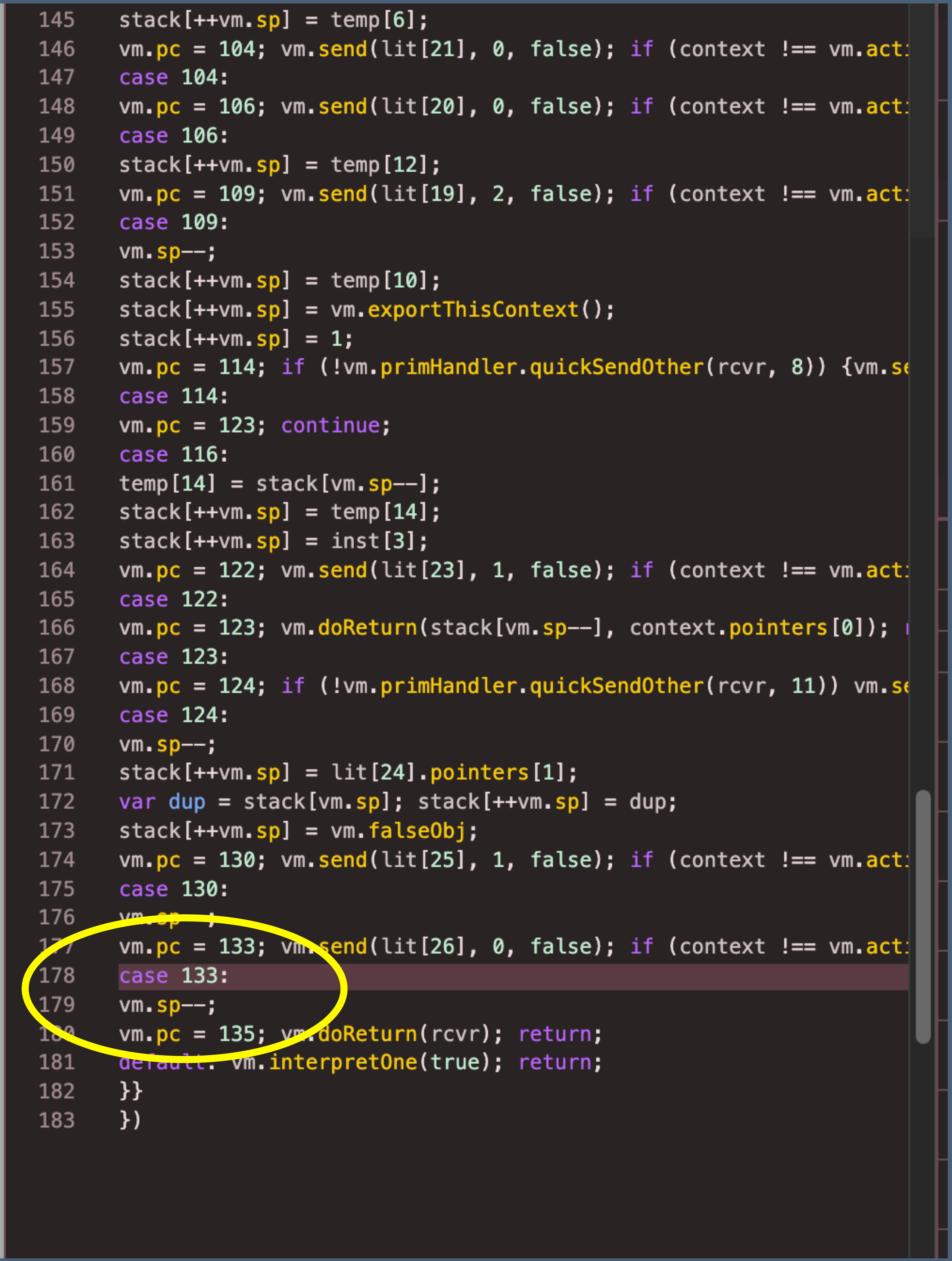
```
▶ specialObjects: anArray  @180
▼ activeContext: aMethodContext
  ▶ 0 (sender): aMethodContext
    1 (pc): 239  [vm: 133]
    2 (stackp): 10  [vm: 15]
  ▶ 3 (method): aCompiledMetho
    4 (receiverMap): nil
  ▶ 5 (receiver): aWorldState  @
  ▶ 6 (1): aPasteUpMorph  @13
  ▶ 7 (2): anArray  @-2611728
    8 (3): true
  ▶ 9 (4): anOrderedCollection
  ▶ 10 (5): anOrderedCollection
  ▶ 11 (6): anOrderedCollection
  ▶ 12 (7): anOrderedCollection
  ▶ 13 (8): aCurveMorph  @137
    14 (9): nil
```

```
[] in BlockContext>>newProcess
[] in Project class>>spawnNewProcess
PasteUpMorph>>doOneCycle
WorldState>>doOneCycleFor:
WorldState>>doOneCycleNowFor:
WorldState>>displayWorldSafely:
BlockContext>>ifError:
BlockContext>>on:do:
[] in WorldState>>displayWorldSafely:
PasteUpMorph>>displayWorld
PasteUpMorph>>privateOuterDisplayWorld
WorldState>>displayWorld:submorphs:

ctx[5]=rcvr: aWorldState
ctx[6]=tmp0: aPasteUpMorph
ctx[7]=tmp1: anArray
ctx[8]=tmp2: true
ctx[9]=tmp3: anOrderedCollection
ctx[10]=tmp4: anOrderedCollection
```

```
this
137513 bytecodes,
22984226 sends (#137513)
```

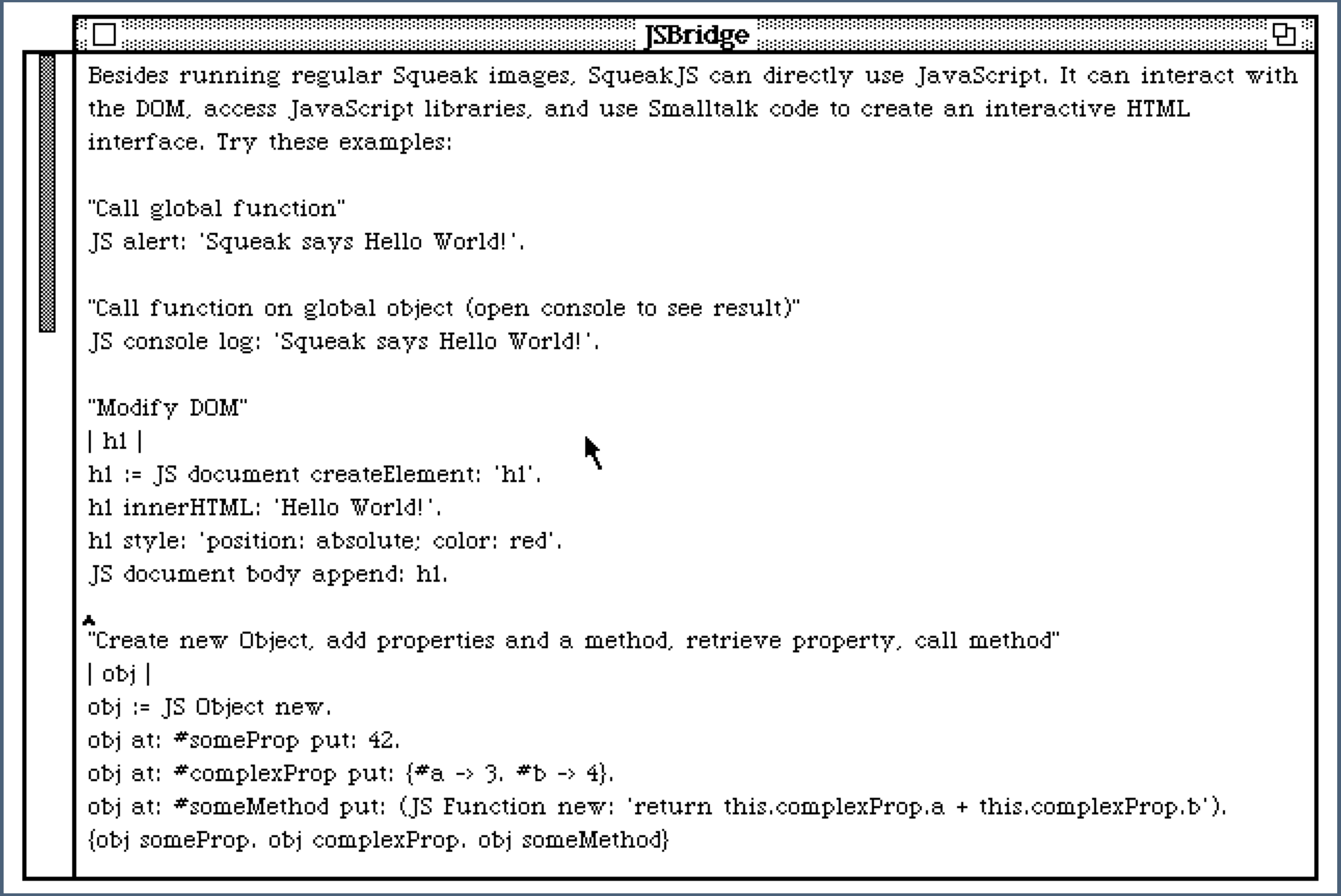
```
   127 <72> pushConst: false
   128 <83 38> send: #deferUpdates:
   130 <87> pop
   131 <83 19> send: #forceDisplayUpdate
=> 133 <87> pop
   134 <78> return: receiver
```



```
stack[++vm.sp] = temp[6];
vm.pc = 104; vm.send(lit[21], 0, false); if (context !== vm.act
case 104:
vm.pc = 106; vm.send(lit[20], 0, false); if (context !== vm.act
case 106:
stack[++vm.sp] = temp[12];
vm.pc = 109; vm.send(lit[19], 2, false); if (context !== vm.act
case 109:
vm.sp--;
stack[++vm.sp] = temp[10];
stack[++vm.sp] = vm.exportThisContext();
stack[++vm.sp] = 1;
vm.pc = 114; if (!vm.primHandler.quickSendOther(rcvr, 8)) {vm.se
case 114:
vm.pc = 123; continue;
case 116:
temp[14] = stack[vm.sp--];
stack[++vm.sp] = temp[14];
stack[++vm.sp] = inst[3];
vm.pc = 122; vm.send(lit[23], 1, false); if (context !== vm.act
case 122:
vm.pc = 123; vm.doReturn(stack[vm.sp--], context.pointers[0]);
case 123:
vm.pc = 124; if (!vm.primHandler.quickSendOther(rcvr, 11)) vm.se
case 124:
vm.sp--;
stack[++vm.sp] = lit[24].pointers[1];
var dup = stack[vm.sp]; stack[++vm.sp] = dup;
stack[++vm.sp] = vm.falseObj;
vm.pc = 130; vm.send(lit[25], 1, false); if (context !== vm.act
case 130:
vm.sp--;
vm.pc = 133; vm.send(lit[26], 0, false); if (context !== vm.act
case 133:
vm.sp--;
vm.pc = 135; vm.doReturn(rcvr); return;
default: vm.interpretOne(true); return;
}}
})
```

JavaScript Bridge

Build Web Apps



JSBridge

Besides running regular Squeak images, SqueakJS can directly use JavaScript. It can interact with the DOM, access JavaScript libraries, and use Smalltalk code to create an interactive HTML interface. Try these examples:

```
"Call global function"
JS alert: 'Squeak says Hello World!'.

"Call function on global object (open console to see result)"
JS console log: 'Squeak says Hello World!'.

"Modify DOM"
| h1 |
h1 := JS document createElement: 'h1'.
h1 innerHTML: 'Hello World!'.
h1 style: 'position: absolute; color: red'.
JS document body append: h1.

"Create new Object, add properties and a method, retrieve property, call method"
| obj |
obj := JS Object new.
obj at: #someProp put: 42.
obj at: #complexProp put: {#a -> 3. #b -> 4}.
obj at: #someMethod put: (JS Function new: 'return this.complexProp.a + this.complexProp.b').
{obj someProp. obj complexProp. obj someMethod}
```

SqueakJS: A Decade of Progress

2016

- Spur support
- Networking plugin (by Fabio Niephaus)
- Incremental GC

SqueakJS: A Decade of Progress

2020

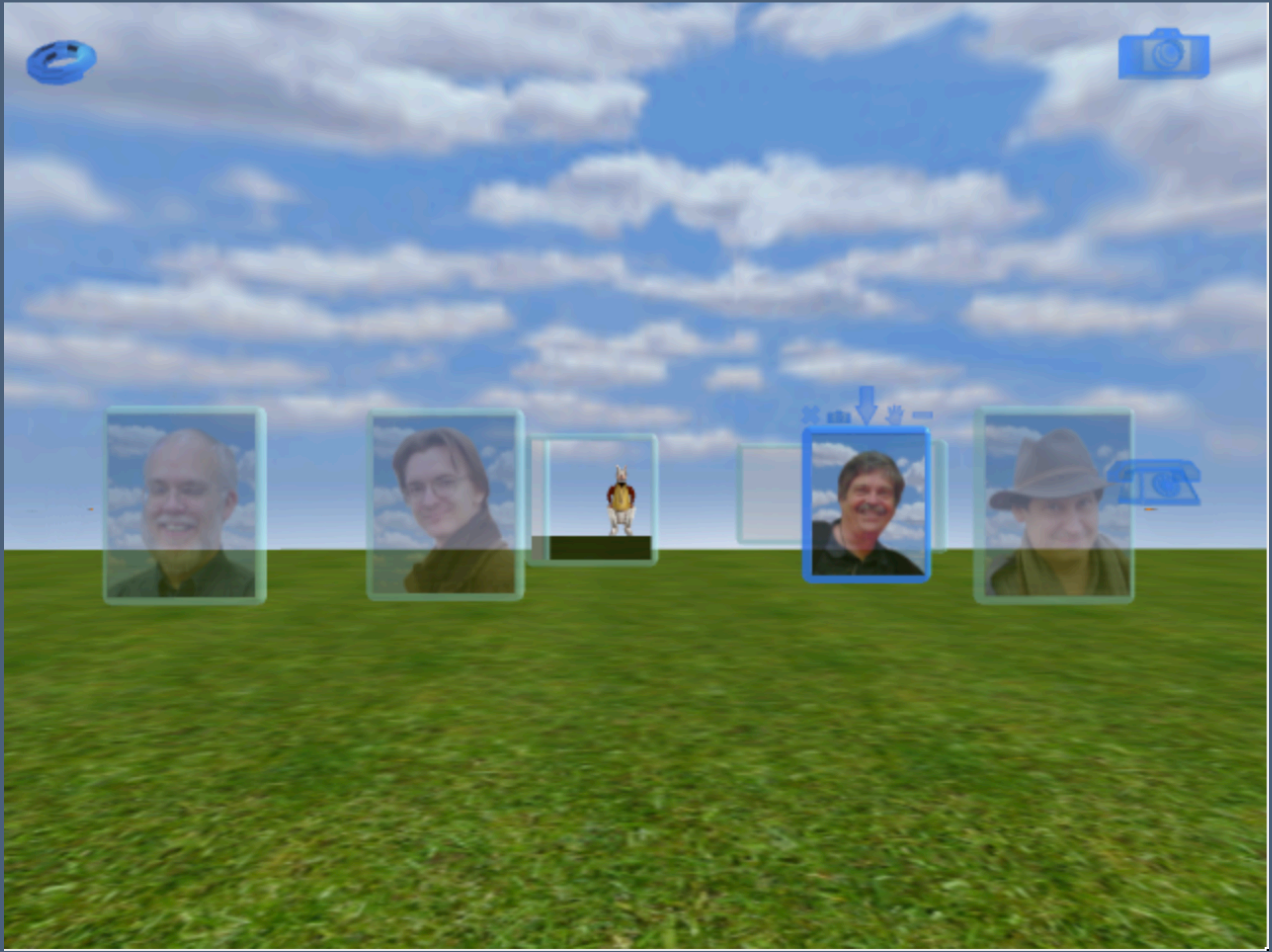
- Switch from Lively Modules to ES modules (by Eric Stel)
- 64 bit support
- SISTA bytecode support (by Fabio Niephaus)

SqueakJS: A Decade of Progress

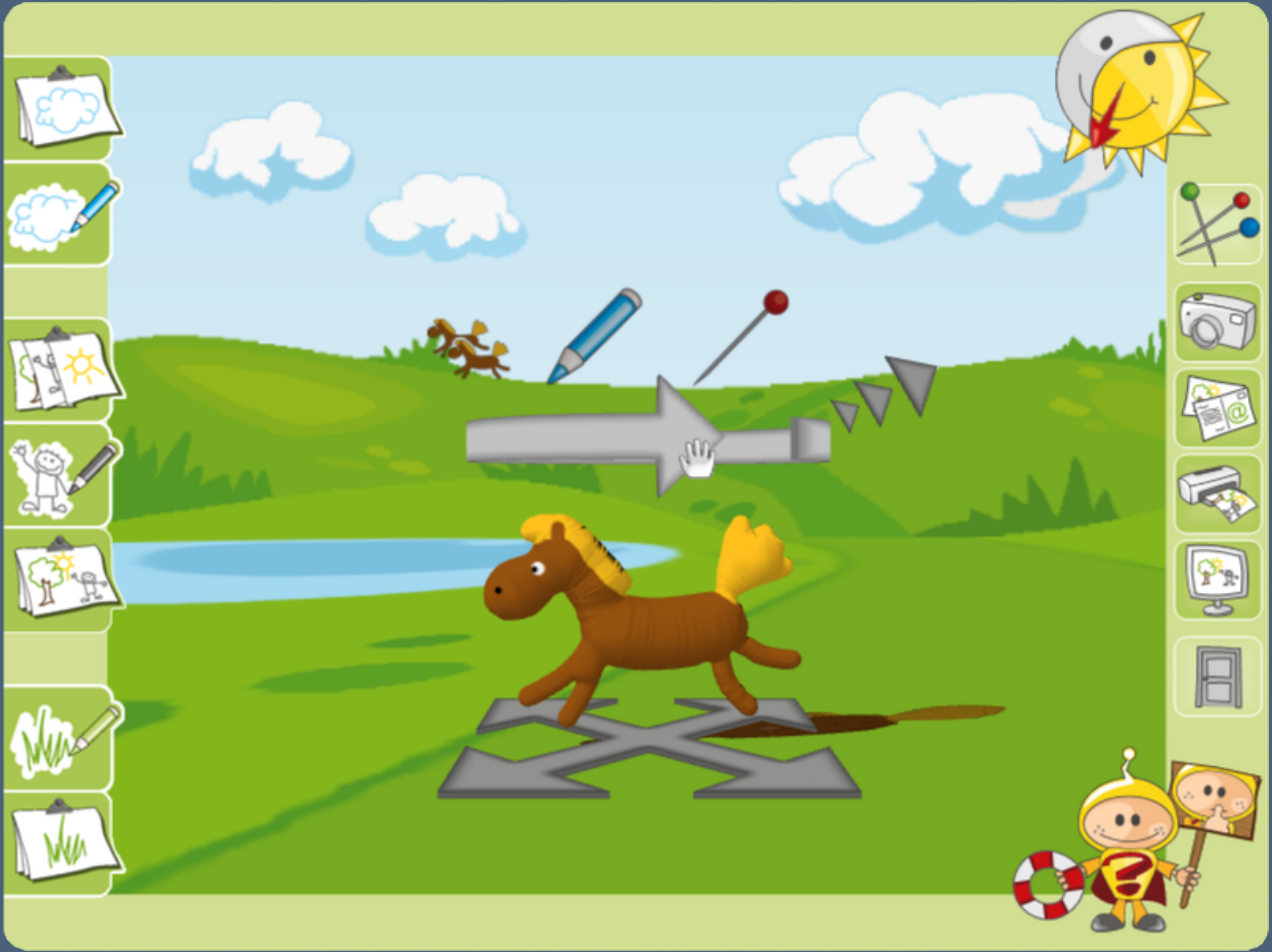
2024-25

- FFI support
- OpenGL 1.0 emulation
- TCP/IP emulation (unfinished)

Croquet



Plopp



Future of SqueakJS

- More applications (like Caffeine and Code Paradise)
- Faster JIT
- 64 bit image export
- “Real” networking plugin
- Native App (e.g. to run Etoys on Mac)

squeak.js.org