Squeak

A general overview collected by
Torsten Bergmann

1 Introduction

1.1 What is Squeak

Squeak is an open, highly-portable and dynamic programming and multimedia environment. It has a great license agreement and is completely Open Source. Squeak runs on many types of computers – ranging from personal computers to small devices like personal digital assistants (PDA's). It runs on a varity of operating systems like Windows, Linux, MacOS, Unix, … to name just a few.

Squeak has support for:

- 2D graphics and formats (GIF, PNG, JPG, …)
- 3D graphics and formats (3DS, Alice, VRML)
- new graphics model called Morphic
- Sound and MIDI support
- Text to speech system
- Internet Support and network support (WWW, FTP, Telnet, Mail, …)
- Shockwave and animation
- Postscript printing
- MPEG and MP3
- Character recognizing and gestures
- Scripting system

With its morphic framework Squeak is a very powerfull and dynamic environment for creating active content. Squeak is supported by a growing community of Squeak developers willing to share code and knowledge.

Squeak and Smalltalk

Smalltalk is a general purpose, high level programming language. It was the first original "pure" object oriented language, but not the first to use the object oriented concept, which is credited to Simula 67. With Smalltalks introduction in the early 1980's the explosive growth of Object Oriented Programming (OOP) technologies began. Smalltalk is not yet another language like C or Pascal – it's a completely different style of programming. Even C++, Java or the new C# language who provide more and more object oriented features have not yet reached the dynamic and flexible nature of the Smalltalk language.

Squeak is a direct descendant of Smalltalk-80 and is entirely written in the Smalltalk programming language. The core developers of Squeak in fact include many of the core developers of Smalltalk-80. Squeaks language and much of the class library is identical to that of Smalltalk-80: they both have objects, classes, single inheritance, blocks, garbage collection, collections, streams, model-view-controller, and many other bits. The Squeak virtual machine is also written entirely in Smalltalk, making it easy to debug, analyze, and change. To achieve practical performance, a translator produces an equivalent C program whose performance is comparable to commercial Smalltalk systems.

1.2 Squeak Platforms

Squeak runs bit-identical on many platforms. Originally developed on the Macintosh, members of its community have since ported it to numerous other platforms including Windows 9x, Windows NT, Windows 2000, Windows CE, all common flavours of Unix, Acorn RiscOS and bare chips like the Mitsubishi M32R/D.

1.3 Download and Installation

You can download all the files you need for free from the Internet. Each release includes platform-independent support for color, sound, and network access, with complete source code. Download sites vary in how they package these files, often you have to download platform-independent and platform-dependent files seperately. The following web pages let you easily download the files you need to run a current snapshot of Squeak:

- **Official Squeak Homepage**
  http://www.squeak.org
- **Squeak Swiki**
  http://minnow.cc.gatech.edu/squeak
- **Squeak FTP directory**
  ftp://st.cs.uiuc.edu/pub/Smalltalk/Squeak

To run Squeak on your computer you need at least the following files:

- a portable **.image** file, containing a snapshot of Squeak
- a platform specific virtual machine to interpret the image

In order to browse source code in the image, you also want two more files:

- the **.source** file common to all versions, and
- the **.changes** file for the specific image you choose.

1.4 The Squeak virtual machine

The squeak virtual machine is a platform dependent executable file.

On windows operating systems the virtual machine is implemented as an **.exe** file. Typically it is named **squeak.exe**.

1.5 The Squeak image file

…

There is also a minimal Squeak image downloadable from the UIUC Smalltalk Archive you can use to run Squeak on low memory systems like PDA's. This image is shrinked to ~600k !

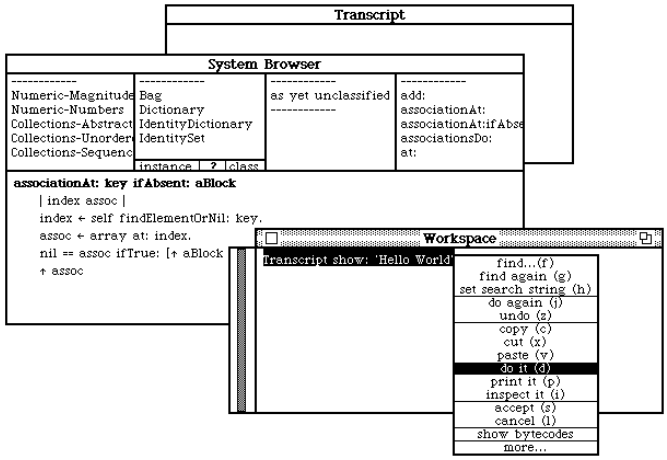
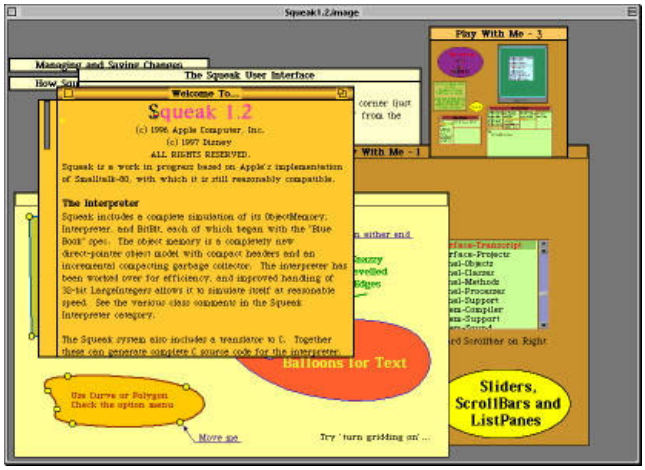


Figure 1: A screenshot of the mini image

You can download it from ftp://st.cs.uiuc.edu/pub/Smalltalk/Squeak/SmallSqueaksForPDAs.

1.6 Squeak Versions

1.6.1 Squeak 1.2



1.7 Squeak and Automatic Updates

If you download Squeak from the web you typically download an official release version like Squeak 3.1 or Squeak 3.2. Since the squeak community is very active in adding new stuff to squeak or fixing bugs it is possible to update the image to a newer version. Squeak includes a facility for automatically updating your system from a remote server if you are connected to the Internet. Choose **„help...“** and **„Update code from server“** from the world menu. Choose **„Squeak External Updates“**. You will see a progress bar, and at the end it will tell you how many updates you got. Updates are like ordinary ChangeSet fileIns. Each one is numbered, and appears in the ChangeSorter window. You can look at what code came in. Note that an update changes your system.

Fixes and enhancements sent to the squeak list are collected by a group called the Harvesters. This group formed as a Squeak Foundation project to harvest fixes and enhancements submitted to the Squeak mailing list, for incorporation into the main Squeak release. They review, test, accept/reject and compile bug fixes, goodies and enhancements submitted to the squeak list, and pass them on to Squeak Central for inclusion into the standard release. Squeak Central may make a few additional judgements on which submissions will be included, but the bulk of the review work will be done by the Harvesters. The current status of each changeset being reviewed can be seen at [3]. Accepted updates are typically announced on the mailinglist using the [UPDATES] tag.

If you want to be a test pilot for new Squeak versions you should update you squeak image to an alpha image. This means you have the very latest Squeak development version available from Squeak Central running. Often, it will not be as stable as the previous official release. Follow the instructions at [4] to update your existing image to an alpha image.

1.7.1 Squeak behind a firewall

If you computer is behind an internet firewall to protect you local intranet you can only get to the outside by using a proxy server. Squeak is able to communicate using such a proxy server. Find out the name of your proxy server (name or number is ok), and the port it uses. Open a new workspace, execute the following statement once, and save your image:

```
HTTPSocket useProxyServerNamed: 'name of proxy server' port: 8080.
```

Anytime you want to, you may ask for updates. If you ever want to stop using the proxy server, just execute this:

```
HTTPSocket stopUsingProxyServer.
```

If you have any trouble see **http://minnow.cc.gatech.edu/squeak/23** for more informations.

1.7.2 Updating with no net access

If your Squeak machine is not connected to the internet, you may still want to get the latest Squeak updates. The update mechanism is basically a fileIn, so you can do it by hand. First you need a master image which is connected to the internet. Set the preference **"updateSavesFile"** to true and do an update from the server. Each time you update from the server, a copy of each update file that is loaded is automatically placed on your disk. You can then send around these update files via diskette or email or whatever to those wishing to update offline. To update images offline in a painless fashion, each offline user maintains all the update files in a folder named **"updates"** that resides in the same directory as his image. Then all each offline user need do is evaluate the expression:

```
Utilities applyUpdatesFromDisk
```

...and all the new updates in that **"updates"** folder which are not yet present in his image will be automatically slurped in, in correct numerical order. Note that this mechanism also provides an alternative way that you can selectively update an image for debugging purposes.

Since updating sometimes takes a while for recompiling you can also fetch updates without updating the system. This safes time and money for per-minute online Squeakers. Use the following statement to get the updates:

```
Utilities readServerUpdatesSaveLocally: true updateImage: false
```

You can now disconnect from the network and use the previous expression to apply the updates stored on the disk to the image.

References

[1] Code updates in Squeak
http://minnow.cc.gatech.edu/squeak/22

[2] Squeak Foundation
http://swiki.squeakfoundation.org

[3] Harvesting Tables
http://209.143.91.36/super/415

[4] Squeak alpha testing
http://minnow.cc.gatech.edu/squeak/1331

2 The Squeak Community

2.1 The Squeak Mailinglist

The Squeak email list is very active, reflecting the vibrant Squeak community. New users are encouraged to ask for help on this list. You will also see bug reports, bug fixes, and discussions of how to improve Squeak. To join the list, sign up at:
http://lists.squeakfoundation.org/listinfo/squeak-dev or email
squeak-dev-request@lists.squeakfoundation.org?subject=subscribe

If your are subscribed you can post to: squeak-dev@lists.squeakfoundation.org

The Squeak mailing list is high in traffic. If you dont want to get all the mails from the list you can either choose a „digest option“ when you sign up the list or use several archive sites like:

- **Searchable Squeak Mail Archive**
  http://macos.tuwien.ac.at:9009/Server.home
- **Active State Programmers Network**
  http://aspn.activestate.com/ASPN/Mail/Archives/squeak/
- **Yahoo Egroups**
  http://groups.yahoo.com/group/squeak/

to read current or dig in old submissions. The Squeak developer swiki [1] contains an up to date list of these archives. Note that some of the mail archives are indexed by internet search engines like google [2]. Doing a search there is sometimes the best and fastest general way to search for Squeak-specific content.

You may notice that some messages on the list start with a tag to categorize the posting. Typical tags are [BUG] for bug reports, [FIX] for code submissions fixing the bug, [ANN] for announcements, ...
A complete list of commonly used tags can be found at [3].If you feel at all unsure about asking a beginner-oriented question on the Squeak list, you can always add a [newbie] or [beginner] tag to the subject of your message.

References

[1] Squeak Swiki - FAQMailingListArchives
http://minnow.cc.gatech.edu/squeak/FAQMailingListArchives
[2] Google – Search Engine
http://www.google.com
[3] Title Tags on the Squeak E-Mail List
http://minnow.cc.gatech.edu/squeak/1962

2.2 Squeak News – Magazine

References

[1] Squeak News
http://www.squeaknews.com

3 The Squeak system

3.1 First Steps

Start a clean squeak image

3.2 The Squeak User Interface

3.2.1 General

Appearance, fill screen, screen depth

3.2.2 Windows

3.2.3 Panes

3.2.4 Menus

3.2.5 Preferences

The Preferences window in Squeak lets you adjust the settings of many preferences. To open the preferences window, from the world menu select the **„help“** submenu and then select **„preferences...“**. For example, if you prefer to have your scrollbars appear fixed in place and on the right side of the window pane (as with many other user interfaces), go to the "scrolling" category of the Preferences window and turn on the inboardScrollBars and scrollBarsOnRight checkboxes.

If you know the name (or part of the name) of your preference, but you don't know which category it's in, type the name into the Search field after opening the Preferences window, and hit Search.

Browse the class **Preferences** for information on how to create your own preferences.

3.2.6 Projects

3.3 Development Tools

3.3.1 Workspace

A workspace is a simple text window. It is used to edit and test Smalltalk code. In difference to the Transcript there can be more than one workspace in the system. You open a new

workspace by right clicking on the desktop, open the world menu and selecting **"Open"** and **"Workspace"**. You can also press the Cmd-Key together with 'K' as a keyboard shortcut. (*Note that the Cmd-key is the Alt-key on Windows – so you have to press ALT-K.*)

Figure 2: A new workspace window

To change the title of the workspace window left click inside of the label area. You can enter any Smalltalk expression in the workspace and evaluate it using the workspace context menu. The context menu is available if you right click in the workspace area. Enter the Smalltalk expression **3 + 4** in the new workspace, select the text and right click to open the context menu. Select **"Print it"** from the context menu. Alternatively you can press Cmd-P (ALT-P on Windows). Squeak will evaluate the expression and print the result in the workspace.

The workspace is implemented in the class **Workspace**. Therefore you are able to create new workspace windows by evaluating code. Clear the text in the workspace and enter a new expression:

```
Workspace open
```

Again – select the text, right click to open the context menu. Now select **"Do it"** in the provided list – a new workspace is created by the system and displayed on the screen.

3.3.2 Transcript

3.3.3 Inspector

3.3.4 File List

3.3.5 System Browser

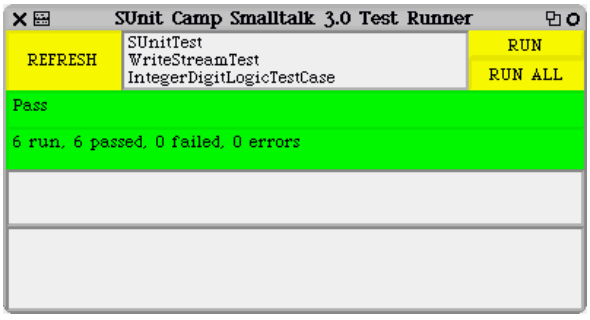
3.3.6 Change Sorter

3.3.7 Method Finder

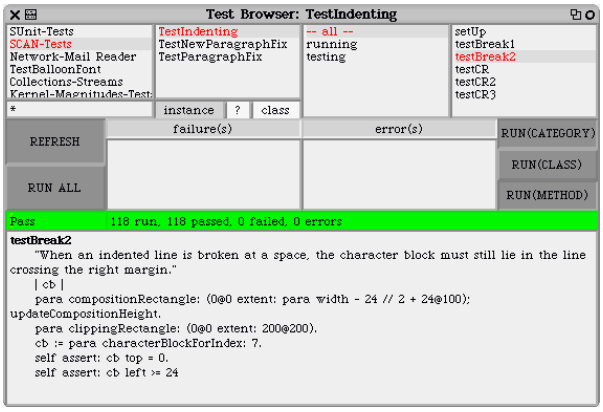
3.3.8 Test Runner

Squeak includes the The SUnits Testing Framework by Kent Beck and Erich Gamma [1]. To start the standard TestRunner evaluate the following code:

```
TestRunner open
```



There is an enhancement done by Rob Withers adding a HierarchyTestRunner which displays TestCases, hierarchically [2]. You can also use the TestBrowser, which is also an enhanced version of the TestRunner for Squeak. The TestBrowser filters test cases by class category and your are able to run and edit test cases in one window.



References

[1] Test Framework
http://www.xprogramming.com/testfram.htm

[2] Squeak Hierarchy Test Runner
http://lists.squeakfoundation.org/pipermail/squeak-dev/2002-January/007334.html

[3] Squeak Test Browser
http://www5.ocn.ne.jp/~minami/squeak/testBrowser/index.html

3.4 Other Tools

3.4.1 CPU Watcher

The Squeak Process browser is a handy tool for analyzing processes running within Squeak. To open one, from the World menu, go to the „debug...„ submenu and select „open process browser“.

To see live updating of processes and cpu usage within squeak, turn on „auto update“ and „CPUWatcher“ from the world menu.

3.4.2 Time Profile Browser

The TimeProfileBrowser is a performance profiling browser in Squeak, based on the core MessageTally tool. If a specific operation in Squeak seems too slow, use the TimeProfileBrowser to find out why:

```
TimeProfileBrowser onBlock:
        [ 20 timesRepeat: [ Transcript show: 100 factorial printString ]].
```

Or whatever you want to do in the block. You'll see a breakdown by percent of time spent.

Attack the parts that are taking the most time, if you want to speed it up.

If the operation happens to be a UI operation: You can start up a MessageTally/TimeProfileBrowser by using the **"debug..." "start/browse MessageTally"** menu item. Let's say that the slow operation is the act of closing a particular window in Morphic. Start the message tally. It will warn you that it is about to start the tally. After starting it, immediately do the operation (close the particular window in this case), and then quickly move your mouse to the top of the screen to end the tally. A window will now appear with the tally results, which is a hierarchy of the methods in which a significant amount of time is spent. Probably a significant chunk of the time (maybe more than 50%) is spent in the method WorldState>>interCyclePause:... this is the extra time that was spent before and after closing the window, which we can ignore. We can follow the other branch HandMorph>>handleEvent:, etc.) down and see what the "deepest" method(s) are which are still taking a significant percentage of time, and browse the code to see if there's an obvious problem in the implementation.

4 Morphic

4.1 Introduction

Squeak contains a new graphic framework called **“Morphic”**. Since version 2.0 the entire Squeak development environment is running in Squeak.

5 Squeak and 3D Graphics

5.1 General

5.1.1 Display

5.1.2 Colors

5.1.3 Forms

5.2 Images

5.3 3D Graphics

5.3.1 Wonderlands

5.3.2 3D Support

6 Squeak and Multimedia

6.1 Squeak Text To Speech System

The Squeak Text-To-Speech (TTS) system includes classes for doing formant synthesis, phonetic transcription, and prosody generation. The synthesizer itself, KlattSynthesizer, is a Klatt-syle cascade/parallel formant synthesizer. This type of synthesizer was developed by Dennis Klatt for the MITalk (now DECTalk) text-to-speech system. The filters are organized in two branches: a parallel branch more useful for consonants, and a cascade branch best swited for vowels. If you want to try some examples, just take a look at the 'examples' class method categories in the DECTalkReader and Speaker classes.

For example, try the following:

Speaker man say: 'Listen to my voice. I am a man speaking.'.

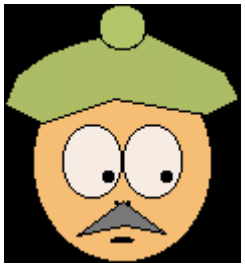
or: Speaker whispery say: 'This is my whispery voice.'

Try this variation to see an animated face speak a phrase:

Speaker manWithHead say: 'This is my voice. Can you see my lips?'

Or try entering your own phrase in the quotes. Make sure you have the sound turned on when trying these! More examples are available in the Speaker class-side *'examples'* category, which also illustrate how to set pitch, voice, range, etc.

The synthesizer has a few global settings such as sampling rate, milliseconds (or samples) per frame and number of formants in the cascade branch. There are also 52 time-varying parameters that are updated every 10 milliseconds or so, all at once, setting the current frame to a new KlattFrame. The 52 KlattFrame parameters specify the formant frequencies, bandwidths and amplitudes, the amplitude of each excitation source (friction, aspiration or voicing), the voice quality, and the fundamental frequency or pitch. Special care was put on the voice quality parameters, so as to make the synthesis of diferent voice personalities and even pathological voices easier. At the time of writing, this is the most complete publicly avalable Klatt synthesizer.
In this TTS system, phonetic transcription is achieved by means of dictionary lookup and contextual text-to-phoneme rules. Each PhoneticTranscriber must have a collection of PhoneticRule's (instances of class PhoneticRule), and optionally a lexicon. When a transcriber is asked for the transcription of a word, it searches for the word in the lexicon first, and if the word is not found then the rules are used.



A list of PhoneticEvent's is generated from the Phonemes after the phonetic transcription takes place. Each PhoneticEvent includes a phoneme, duration, loudness, and a pitch contour. Some simple prosodic rules are employed to assign duration and intonation to the events list. After this, the events are played on a Voice.
There are two kind of voices currently implemented in the Squeak system. The first voice is the KlattVoice. This voice produces sound from PhoneticEvents. Each phoneme is mapped to one or more KlattSegments, then KlattFrames are generated from the KlattSegments, and finally the KlattFrames are played on a KlattSynthesizer. The other voice available in Squeak is a GesturalVoice, a Voice that can play GesturalEvents (for lips, eyes, moods) and

PhoneticEvents animating a face. Several voices can be combined into a CompositeVoice, so the TTS system is able to do synchronized face animation and speech synthesis on a composite voice made up of a KlattVoice and a GesturalVoice.

The system was extended in several directions. For instance, much more natural voices have been achieved doing diphones concatenative synthesis with LPC and Residual Pitch Synchronous LPC diphones. Also, the GesturalVoice is being extended to use a 3D face with Waters' muscles model, which is especially well swited for the realization of emotions.

References

[1] Squeak Speaks
http://community.corest.com/~luciano/tts/

[2] Squeak Swiki – Squeak Speaks
http://minnow.cc.gatech.edu/squeak/651

7 Networking with Squeak

7.1 Introduction

Squeak has support for web based technologies and allows platform-independent networking. The system has a built-in HTTP and FTP protocol implementation and allows asynchronous as well as synchronous network access. A pluggable web server architecture allows the simple construction of own web servers.

7.2 Sockets and Protocols

Squeak has (since version 2.0) platform independent support for sockets. A socket represents a network connection point and is implemented in the class **Socket**. Squeak sockets are designed to support the TCP/IP and UDP protocols. Subclasses of **Socket** provide support for other well known network protocols like POP, NNTP, HTTP, SMTP and FTP:

- **TCP/IP (Transport Control Protocol/ Internet Protocol)**
  This protocol is the well known protocol of the internet.

- **POP or POP3 (Post Office Protocol 3)**
  This protocol as specified in RFC 1939 is used to download email from a mail server to your personal mail program.

- **NNTP (Network News Transfer Protocol)**
  NNTP is used to transfer messages from USENET discussion groups.

- **FTP (File Transfer Protocol)**
  This protocol is used to download and upload binary files.

- **SMTP (Simple Mail Transfer Protocol)**
  This protocol is used for sending mails. It is specified in RFC 821.

- **Telnet**
  This protocol allows terminal networking on remote computers.

The class **Socket** provides some class side examples you can use to test some of the low-level network facilities.

7.3 Squeak and the World Wide Web

7.3.1 Retrieving web contents

It is possible to use Squeaks own webbrowser Scamper to browse the web – but you can also access web contents using Smalltalk. To grab a page from the web and see it's HTML source in the Transcript you can evaluate the following expression:

```
"Output a Web page on the Transcript window"
| mimeDocument |
mimeDocument := 'http://www.squeak.org' asUrl retrieveContents.
Transcript show: mimeDocument content.
```

You could also use the following expression:

```
HTTPSocket httpShowPage: 'http://www.altavista.digital.com/index.html'.
```

You can also grab a picture from the web. Evaluate the following code:

```
HTTPSocket httpShowGif:
```

```
'http://squeak.org/Squeak2.0/midi/Squeakers.GIF'.
```

Squeak will download the GIF file from the internet and open it as a morph on the desktop.

```
HTTPSocket httpShowJpeg: 'http://jaderholm.com/exobox/addressbook.jpg'.
```

You can also download a squeak project (image segment) from the web using the following expression:

```
ProjectLoading thumbnailFromUrl:
        'http://209.143.91.36/super/uploads/Squeak%20Notepad.003.pr'
```

Download music from the internet:

```
MIDIFileReader playURLNamed:
        'http://squeak.cs.uiuc.edu/Squeak2.0/midi/wtellovr.mid'.
```

or a flash file:

```
(FlashMorphReader on: (HTTPSocket
            httpGet: 'http://www.audi.co.uk/flash/intro1.swf'
            accept:'application/x-shockwave-flash'))
        processFile startPlaying openInWorld.
```

7.3.2 Helpfull network classes

As you may have noticed in the examples from the previous chapter it is very easy to create a URL from a string using the **#asUrl** message. Depending on the protocol you get an instance of a different subclass of the class **Url**. Inspect the following expressions and compare the results:

```
'http://www.squeak.org' asUrl  "returns an instance of HttpUrl"

'ftp://st.cs.uiuc.edu/pub/Smalltalk/Squeak' asUrl "returns an instance of
FtpUrl"
```

```
NetNameResolver addressForName: 'http://www.squeak.org'
a ByteArray(207 71 8 55)
```

7.4 The Browser Plugin

References

[1] The Squeak Webbrowser Plugin
http://minnow.cc.gatech.edu/squeak/1865

[2] Squeakland Website
http://www.squeakland.org/

[3] Squeak Browser Plugin
http://www.squeaklet.com/NPSqueak/examples/examples.htm

[4] Unix Squeak Webbrowser Plugin
http://wwwisg.cs.uni-magdeburg.de/~bert/squeak/plugin/download.html

7.5 Comanche and Pluggable Webserver

Everytime you open your favorite web browser, type in an internet adress or click on a link on a webpage you send a request to a program on another computer – the so called web server. This web server program is continuously running on the server machine to answer these requests and send data (webpages or files) back to the client. You may know web server implementations like the free Apache web server or the commercial Internet Information Server. Comanche is an open source web server for Squeak written by Jochen F. Rick (Jochen.Rick@cc.gatech.edu) and Bolot Kerimbajev (bolot@cc.gatech.edu).
Comanche contains the server framework, which makes it possible to develop web applications entirely in Squeak, without the need to run an external web server. Comanche also includes a Swiki implementation which allows you to host swikis on your own computer with the help of Squeak (see chapter "Swiki – the Squeak wiki").

7.5.1 Download and Install Comanche

You can download the Comanche packages from the Comanche Homepage at http://minnow.cc.gatech.edu/swiki/ in two forms:

1. **Complete Comanche**
   If you are just interested in using Comanche as a swiki server you should download the appropriate ZIP or TAR archives for your platform. The archive contains the Squeak virtual machine and a predefined Squeak image file with Comanche installed and ready to run. Extract the files in a directory and drop the file **squeak.image** onto the **squeak.exe** file.

2. **Source Code**
   If you want to install the Comanche code into your current squeak system download the source archive and extract it into your squeak image directory. You should now have a subdirectory called "swiki" containing a **readme.txt** file, a change set with the source code (**ComSwiki.cs**) and several subdirectories. Now start your Squeak system and evaluate **ComSwikiLauncher openAsMorph** in a workspace.

Either way – you should now see the ComSwiki Launcher on the screen. This little window allows you to control the server. You can set the server port by pressing the port button. Port 80 is the default web server port. Depending on your operating system you may not have the permission to use it. (On several Unix/Linux implementations, users are not allowed to access ports with low numbers). Therefore port 8000, 8080 and 8888 are other options.
Set the port number to **8080**, press **"Start the server"** and wait until the first button is red.

Figure 3 : The ComSwiki Launcher

The web server is running. To test it open your web browser and enter the following local address: http://localhost:8080/admin/help#getStarted. Use "**admin**" as user name and "**password"** as password. You can use the web based admin tool to create own swikis. Read the provided informations carefully – they help you get started quickly.

Note: *Since you now run a web server, any computer connected to your machine is able to reach it. So your first step should be to change the administrator's (ie. your comanche) password. To do this, click on the* top *button and then on the* security *button. This will take you to the security settings for the AdminTool. From there, change the admin:password to something more secure. Remember that user:password pair, because you will be asked for it after pressing update security*.

7.5.2 Swiki – The Squeak wiki

Comanche includes a quite popular implementation of Ward Cunninghams WikiWiki Web called Swiki (Squeak + Wiki = Swiki). A Swiki is a dynamic web server where anybody is able to change the web pages right from within the browser. Therefore Swiki pages are collaborative websites; anyone can edit and create web pages.

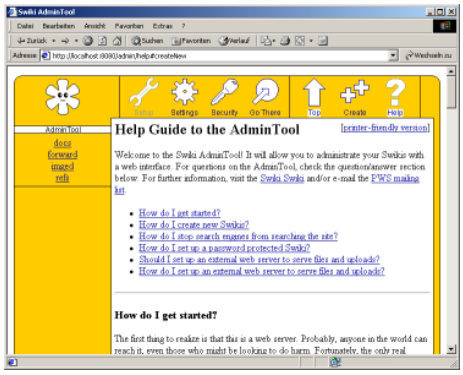


Figure 4 The Admin Tool

To create a new swiki click on the **"Create"** button in the top left corner of the admin tool. Enter a name for the new swiki (for instance "myswiki") and click on the **"Create new Swiki"** button. Squeak will now create a new swiki and the browse will now show the setup for the swiki. Just click on "Go there" to redirect your browser to the new swiki. Note that the address for your swiki is http://localhost:8080/myswiki if you access it from your computer.

The browser shows the first page of the swiki like any other webpage. Click on the **"Edit"** button to edit the first page inside of the browser. Enter the following text:

```
!! This is a test
!! *Squeak>http://www.squeak.org*
```

and hit save to save the webpage. The webpage has changed its contents.

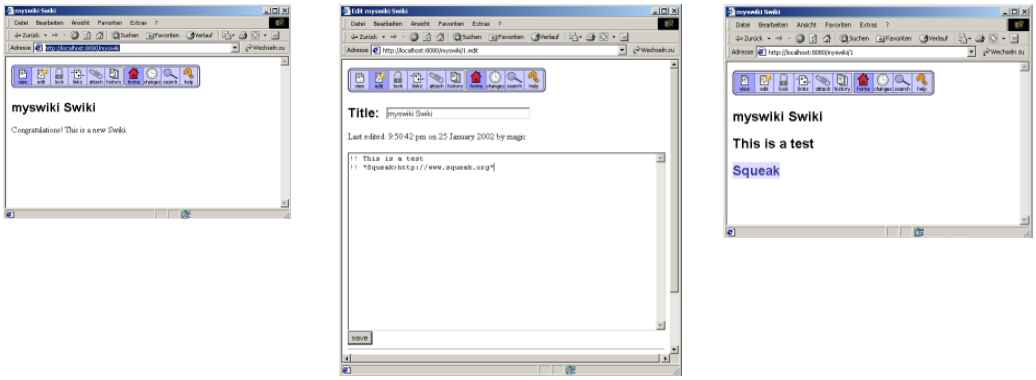


Figure 5: Editing a swiki page

If you edit a page you can use the normal HTML syntax or swiki syntax. So instead of the text above you could have used standard HTML:

```
<h2> This is a test</h2>
<h2> <a href="http://www.squeak.org">Squeak</a></h2>
```

As you may have noticed

You can now invite people connected to your computer to add new content to your swiki. Note that they have to access the swiki using your computers name or IP address instead of localhost.

PWS – Pluggable Web Server

7.5.3 Comanche Architecture

7.5.4 Build an own Comanche Plugin

8 Squeak Packages

8.1 Squeak and XML

The history of Squeak and XML started with as usual with a posting to the squeak mailinglist asking for an XML implementation in Squeak. Shortly after that Duane Maxwell and Andreas Valloud (both were working for a company called Exobox) announced the release of an XML parser and associated tools. Meanwhile Michael Rueger also worked on the implementation of an XML framework for Squeak. His project is called **YAXO – Yet another XML Framework** [1] and is an effort to further integrate the original Yax version with the exobox implementation. It is also compatible with Masashi Umezawa's SOAP support for Squeak [2]. The original Yax version was already based on some ideas in the Comanche tokenizer [3] and the Exobox parser.

You can download the YAXO code from the projects website [1]. It comes with full source code and SUnit Tests. The package includes an XMLParser with SAX and DOM support and an XMLWriter. The SAXDriver/Handler implements the revised SAX2 API [4]. You can use the parser to either parse the XML from a stream or a file. Inspect the following code:

```
|xml document|
xml := '<?xml version="1.0" encoding="UTF-8"?>
                <project name="YAX">
                        <description>
                                YAX is yet another XML parser.
                        </description>
                </project>'.
document := XMLDOMParser parseDocumentFrom: xml readStream.
^document
```

It will return a new instance of the class XMLDocument. Assume we have written the above XML string into a file called **projects.xml** we could easily ask the parser to get the document from the file:

```
XMLDOMParser parseDocumentFromFileNamed: 'projects.xml'
```

You can easily access XML nodes using iterators.

References:

[1] Duane Maxwell, Andres Valloud, Michael Rueger „Yaxo – Yet another XML framework“
http://www.squeaklet.com/Yax/index.html

[2] Masashi Umezawa, SOAP Opera – a tiny ORB running on SOAP-HTTP
http://www.mars.dti.ne.jp/%7Eumejava/smalltalk/soapOpera/

[3] Comanche – an open source web server for Squeak
http://minnow.cc.gatech.edu/swiki

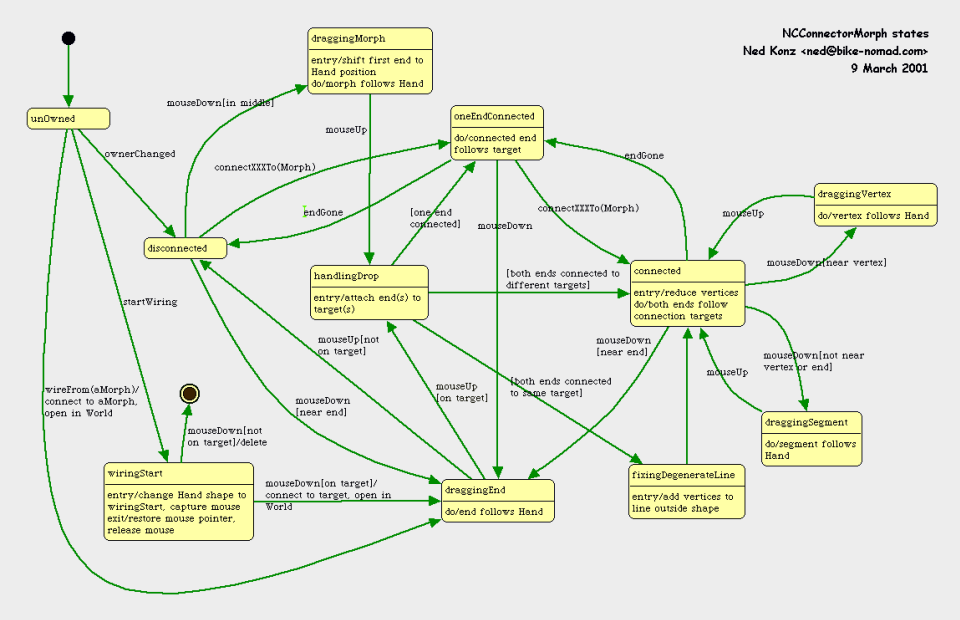
[4] SAX – The Simple API for XML
http://www.saxproject.org/

9 Squeak Goodies

9.1 Morphic Wrappers

9.2 Connectors

Connectors is a graphical framework for connected drawings like state diagrams, class diagrams, concept maps, and the like written by Ned Konz. The new version uses morphic stepping to keep things connected and is not relying on the submorph relationship anymore.



Reference

[1] Connectors Download page
http://nedkonz.dhs.org:8080/Ned/8

9.3 Zurgle

9.4 Whisker Browser

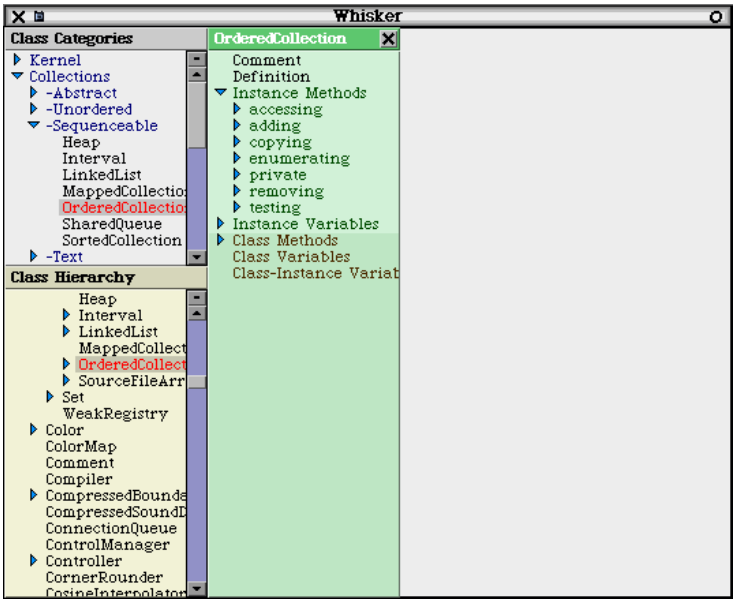
Whisker is a new Object-Oriented (O-O) code browser for the Squeak Smalltalk environment. The goal of the Whisker Browser (a.k.a. Stacking Browser) is to provide a simple and intuitive way to view the contents of multiple classes and multiple methods simultaneously, while using screen real estate efficiently and not requiring a lot of window moving/resizing. It does this by introducing the concept of subpane stacking.

If you are using an older version of Squeak such as 2.8, it is recommended that you first open and enter a Morphic project in Squeak. (Squeak 3.0 starts up in Morphic by default, so this isn't necessary.) Whisker is written using Morphic, and works best in that environment. Whisker can still be opened and used from within an MVC project, but it will be contained inside a Morphic sub-window, and will have a different feel from the rest of MVC. Download and install the whisker changeset from [1].

To open a Whisker browser, select **"open..."** from the World menu, and then "whisker browser" from the submenu. Upon opening the Whisker browser, you'll see a window with a "Class

Categories" subpane in the upper left corner, and a "Class Hierarchy" subpane in the lower left corner. These are two alternate views on all of the classes in Squeak/Smalltalk, "Class Categories" being an informal grouping of classes by application or function, and "Class Hierarchy" being the actual class inheritance hierarchy. To start, we will choose the class OrderedCollection, which we happen to know is in the "Collections-Sequenceable" category. To get there, go to the Class Categories pane and (left-)click on the arrow to the left of the "Collections" in the list. This expands Collections to show its subcategories. Now expand the arrow for the subcategory "-Sequenceable", and from that expanded list, select the class named "OrderedCollection".

You should now see something like the picture below, with the Class Hierarchy pane now expanded to also select OrderedCollection, and a new green pane in the middle column with the title "OrderedCollection". This "class pane" represents the contents of the class OrderedCollection, including its methods, variables, and comment.

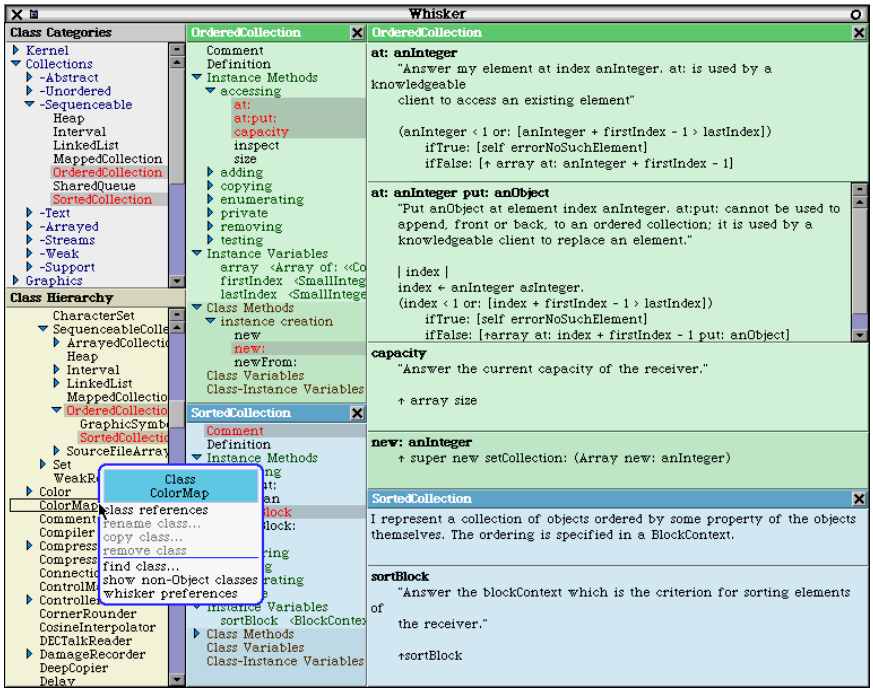


To look at a method in OrderedCollection, let's first expand the method category "accessing" (under Instance Methods) to show its methods. Now we can select the method "at:" from the method list.
You'll see that the source code for the "at:" method now fills up the right column of the Whisker window. We can edit, save, and manipulate this method with the pop-up menu in the method pane, just as with the other Smalltalk browsers.
However, let's try looking at another method. Without touching the "at:" selection in the class pane, select the "at:put:" method below it. Now both methods are selected, and both methods should be appear stacked together in the right column. In this "method stack", we are now looking at both methods at once!
Try selecting a few more methods, such as "capacity" and "size". Note that the shorter methods will take up less space in the method stack than the longer ones, so that as many complete methods as possible are shown. Also note that the methods in the stack appear in the same order that they appear in the class pane. To deselect a method, simply click on the method name again in the class pane. You may try expanding other method categories (or Class Methods), to look at the rest of the methods in the class at the same time.
This multi-selection capability also applies to classes themselves. Go back to the Class Hierarchy pane, expand OrderedCollection to show its subclasses, and select the SortedCollection subclass. A new blue SortedCollection class pane will appear in the middle column, the "class stack", along with the OrderedCollection class pane. You can now select methods in this class pane to view. Note that these methods will appear in a matching blue color, so that the SortedCollection methods are visually associated with the SortedCollection class pane, not the OrderedCollection pane.

Aside from methods, other elements of the class can also be viewed at the same time, such as the class comment and instance variables. When looking at the instance variables of a class, you'll see that Whisker attempts to determine the type of each variable. For example, with SortedCollection, it indicates that the instance variable "sortBlock" is a type of "<nil | <BlockContext>>", which means that it most likely contains either nil or a BlockContext object. Whisker makes these guesses by collecting existing (SortedCollection) objects in Squeak and finding out what the instance variables tend to contain. This can be very useful when browsing unfamiliar classes.

9.4.1 Using Menues

Like other browser windows in Squeak, Whisker makes use of pop-up menus in the various window panes. However, the menus in Whisker work a little differently.
For example, try pressing the menu button in the Class Hierarchy pane. (The menu button is the secondary or "yellow" button of the mouse, normally the middle button on a 3-button mouse, the right button on a 2-button mouse, or option-mouse click on a Mac.)
You'll see that a pop-up menu appears, as in the figure below. However, note that a box is drawn around the list item which the mouse was clicked on, in this case "ColorMap", and that the menu has "Class ColorMap" as its title. In other words, this pop-menu specifically operates on the class ColorMap. If you select "class references" from this menu, it will search for references to the class ColorMap. On the other hand, the menu items in the bottom section of the menu will tend to not be specific to the clicked-on item, such as "find class...".



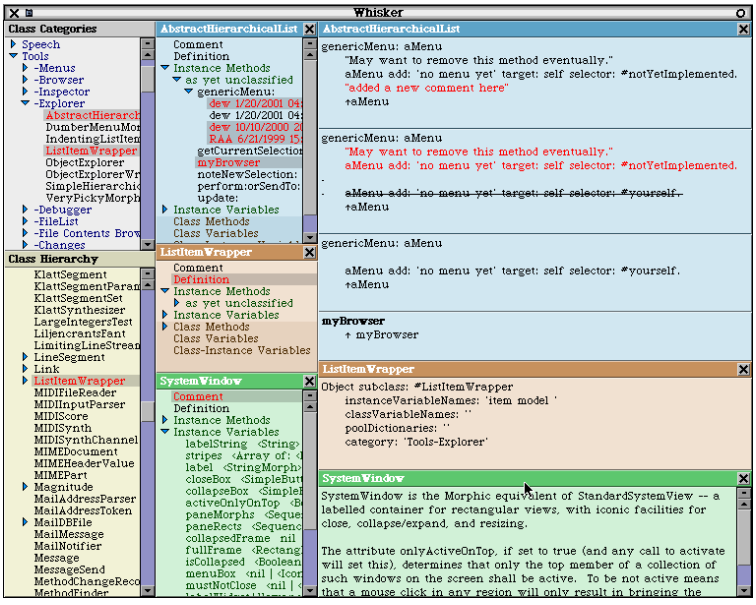
Similarly, if we go to one of the class panes and menu-click on a method name, such as "at:put:" in OrderedCollection, the pop-up menu will appear with the title "Method at:put:". Several menu operations appear here which are relevant to methods, such as finding senders and implementors of the method. Menu-clicking on another method brings up a similar menu. But if we menu-click on a method category such as "accessing", a different menu appears, with operations for method categories. There are also different menus for instance variables and other item types.
You may notice that there is a distinction between selectable list items (such as methods and classes) which open a new pane when selected (left-clicked), and non-selectable items (such as categories), which do not open a new pane. These selectable list items appear in black to indicate that they are selectable, while non-selectable items appear in other colors such as dark blue, green or brown. Attempting to select (left-click) a non-selectable item will bring up its pop-up menu instead.

9.4.2 Editing Smalltalk Code and Creating Classes

To edit an existing method in Whisker, you can simply edit the text contents of the method, and then save/accept the method with the pop-up menu in the method pane.
To create a new method for a class, one simple trick (which works in the other Smalltalk browsers) is to edit an existing method and change its name (in bold) at the top of the method, and then save/accept the new method.
However, if you don't like this approach, or if you need to create a new method in an empty category, you can use the "new method..." operation in the method category pop-up menu. This will prompt you for the method name and create an empty method with the name for you to edit. (Depending on your version of Whisker, if for some reason the new method doesn't show up in the list, you may need to collapse and re-expand the method category contents.)
If you are starting with a newly created class, you will first need to create a method category before you can add new methods. To create a new Instance-side method category, menu-click on the "Instance Methods" item in the class pane and select "new category..." from the menu.
Continuing to work backwards, if you want to create a new class, for now you'll have to use the more traditional technique used in the regular Smalltalk browsers. You can click on the Definition item in a class pane, and then edit the names of the parent class and the class to be created, the instance variables, category, etc., and then save/accept the new definition. (Depending on your version of Whisker, you may need to collapse and re-expand the Class Hierarchy to get the new class to appear.) There will probably be a menu-driven alternative to creating a class in an upcoming release of Whisker, see the Whisker home page (below) for details on current/upcoming releases.
To add, remove or rename instance or class variables, for now you will also need to select the Definition item in the class pane, and edit the variables in the definition template.

9.4.3 Other Features

A special feature of the Whisker browser is its support for method version browsing.
If you save a method more than one time, you'll notice that an expandable arrow will appear to the left of the method name in the class pane. Expanding this arrow will reveal the history of the versions of this method, similar to the "versions" menu item in other Squeak browsers. Beyond the regular Squeak versions browser, Whisker gives you the ability to view any or all of the method versions at the same time, and also to compare the differences between any of the versions. This can be very useful when trying to merge changes made by more than one person to the same method. The picture below shows three different versions of the method *AbstractHierarchicalList>>genericMenu:* being compared with each other.



To try an example of versions browsing, create a new method called something harmless like "myMethod" on the class OrderedCollection (or any other class). Add a new comment or line of code to the method, and save/accept the method. Add another comment or line of code in a

random spot, and save again. Repeat this 5 or 6 times. If you expand the arrow by the method name in the class pane, you'll see each version which you just saved listed individually.
By default, "diffs" are shown, which shows the differences in the method text between the selected version and a previous version. If you only select one method version to view, it will show its diffs relative to the immediately previous version. However, if you select two method versions (say, the 2nd and the 4th one in the list), it will show the diffs between version #2 and version #4, ignoring version #3. You could even select versions #2, #4 and #7 and see the diffs from 2 to 4, and 4 to 7 at the same time. Diffs can be turned on and off globally with the diffsInChangeList browsing preference (see below).

9.4.4 Preferences

In every list pane pop-up menu in Whisker, there's a menu item "whisker preferences" which brings up the Squeak preferences dialog with a special Whisker page of preferences. To see what these do, move the mouse over each preference to bring up a balloon help description. For example, the showWhiskerMethodTitles preference, if turned on, will place a titlebar above every method pane (which can be handy if you want to use the close boxes in the method titlebars, but otherwise wastes space). You may need to close currently open panes (or simply close and re-open Whisker) for most of these preferences to properly take effect.
There are also other non-Whisker preferences scattered throughout the preferences dialog which can still be relevant to Whisker, such as the diffsInChangeList or browseWithPrettyPrint preferences in the "browsing" category. Also, if you already use the inboardScrollbars preference in the "scrolling" category, you may also want to try the hiddenScrollbars preference, which makes it visually more obvious when a method is completely shown in ist pane in Whisker.

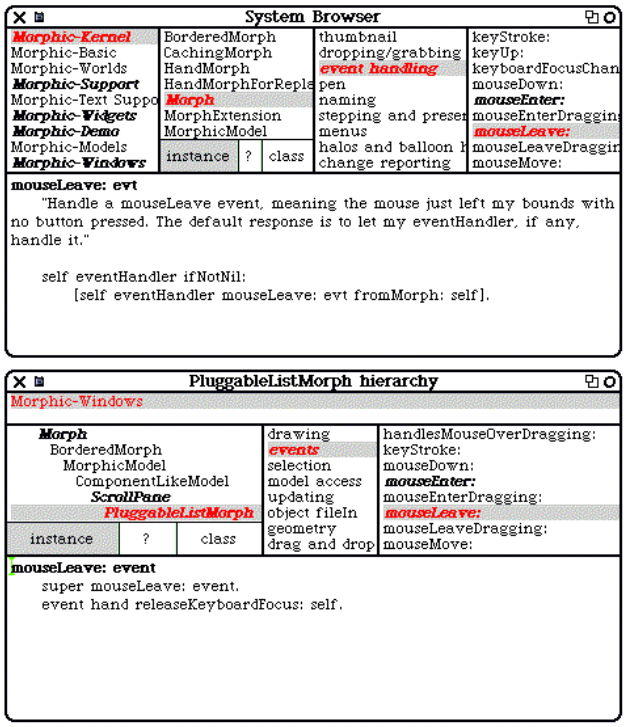
References

[1] Whisker: The OO-Stacking Browser
http://www.mindspring.com/~dway/smalltalk/whisker.html

9.5 Aspect/S

AspectS is an approach to general-purpose Aspect Oriented Programming in the Squeak environment. The intent of AspectS is to extend the Smalltalk environment to allow for experimental aspect-oriented system development. It mainly draws on the results of two projects: the first is AspectJ, a general-purpose aspect oriented language extension to Java, and the second is MethodWrappers, a mechanism to add behavior to a compiled Smalltalk method. AspectS benefits heavily from the simple, elegant, and open architecture of Squeak itself.
AspectS represents an effort to help understand issues that come along with aspects in a dynamic environment like Smalltalk.

AspectS allows for coordinated meta-level programming, addressing the tangled code phenomenon by providing aspect related modules. In its first implementation, AspectS is realized using plain Smalltalk only, without extending neither the Smalltalk language nor its virtual machine.



The Aspect/S website contains a good documentation for using aspects in Squeak. The goodie is released under the squeak license.

References

[1] Aspect/S: Aspects in Squeak
http://www-ia.tu-ilmenau.de/~hirsch/Projects/Squeak/AspectS/

[2] Aspect/J
http://www.aspectj.com

[3] Method Wrappers
http://st-www.cs.uiuc.edu/~brant/Applications/MethodWrappers.html

9.6 3D Facial Animation

This goodie contains an animated face based on Waters' muscle model [1]. It uses the same data sets as Waters example code, and of course the same muscles model, but the implementation is quite different. Each avater has a set of control parameters for facial muscles, eyeballs, eyelids and mouth.

Currently 18 facial muscles are modeled, 9 for each side of the face: angular depressor, frontalis inner, frontalis major, frontalis outer, inner labi-nasi, labi-nasi, lateral corigator, secondary frontalis and zygomatic major. Changing the value of the muscles control parameters, i.e. contracting or relaxing muscles, the face can achieve many facial expressions.

References

[1] Waters, K. "A muscle model for animating three-dimensional facial expression.", IEEE Computer Graphics, 21(4)
[2] 3D Facial Animation in Squeak
http://community.corest.com/~luciano/faces/

9.7 Jabber – Instant Messagin with Squeak

The Squeak Network IM package [1] contains a couple of classes that should be common or abstract classes for instant messaging systems e.g. the ICQ chat client could/should be ported to work within this framework. Central communication point for a IM application is the IMClient. All calls should be implemented here thus representing the IM API within Squeak. Protocol specific issues should be coded using subclasses of IMProtocol.
IMConnection manages the communication layer in a transparent way, sending prerendered messages and using the protocol instance to divide the incoming byte stream into protocol entities/messages. The IMClient has to be polled with processMessages, although this could be moved to a background process.

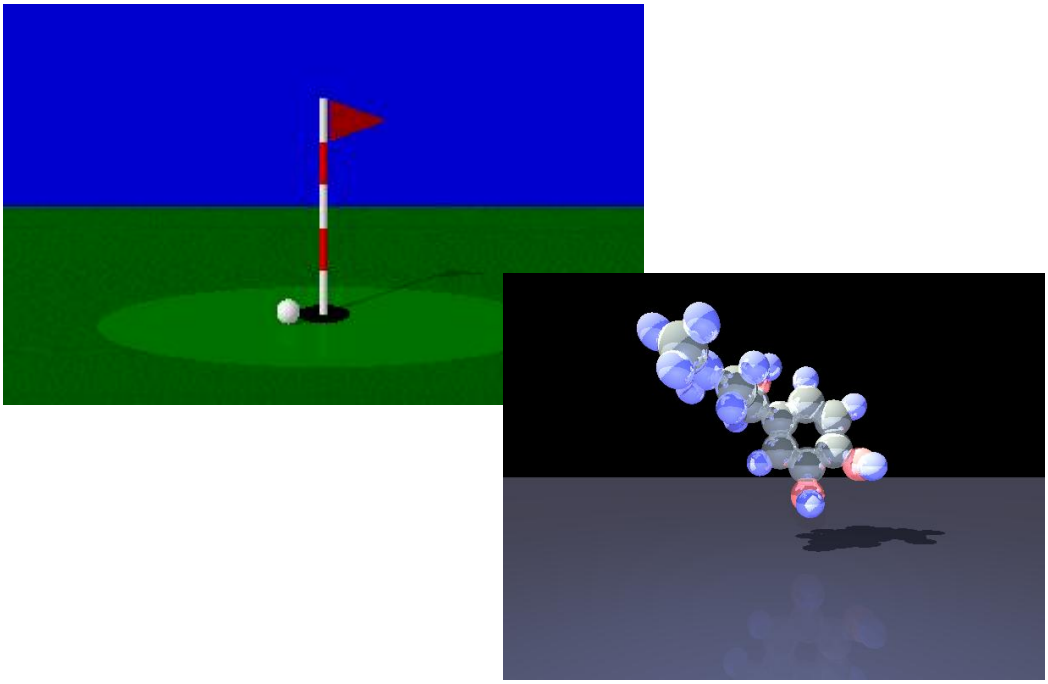
The project website also contains a package called Jabber – a first implementation of a Jabber chat client based on an XML protocol.

References

[1] Jabber and Instant Messaging Project Website
http://squeaklet.com/IM/index.htm

9.8 Callisto – Squeak Raytracer

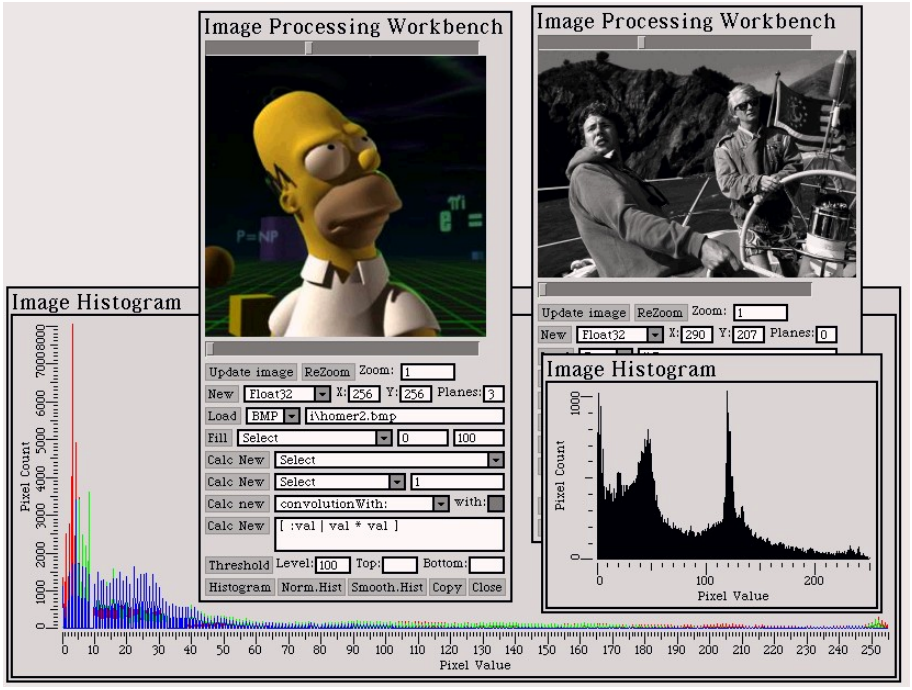
During the ICFP Programming Contest [2] some Squeakers have built a small Squeak raytracing framework.



References

[1] Callisto – Squeak and Raytracing
http://callisto.swiki.net/1
[2] ICFP Programming contest
http://www.cs.cornell.edu/icfp/task.htm

9.9 Squeak Image Processing Framework

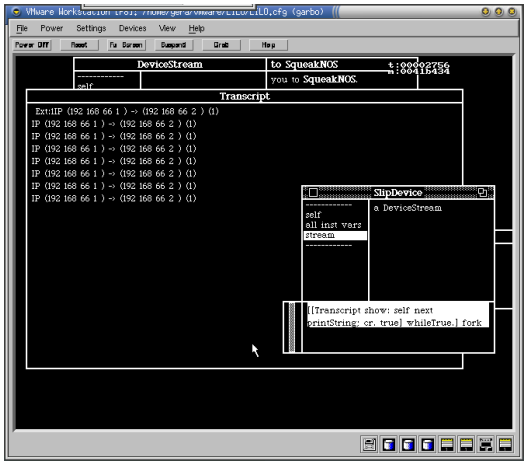


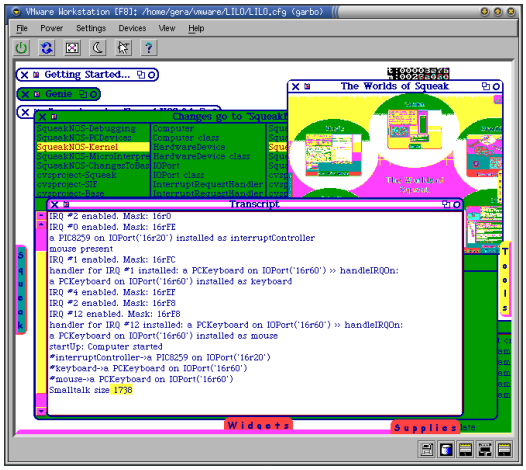
References

[1] Squeak Image Processing Framework
http://webs.sinectis.com.ar/jmvuletich/ImageProcessing/ImageProcessing.html

9.10 Squeak No Operating System (SqueakNOS)

The Squeak NOS project tries to get rid of the OS under Squeak, and is implementing all the functionality in Squeak. The idea is to write a really tiny kernel to boot Squeak. Right now it's only missing timer's IRQ, graphics, mouse and keyboard support.
After this first step, the idea is to use some kind of NativeCompiledMethod to write Device Drivers, and then move as much as possible to Squeak side (video drivers, IRQ handlers, etc. can be built using this NativeCompiledMethods when performance is a must.





References

[1] Squeak NOS Swiki
http://mathmorphs.swiki.net/9
[2] Source Forge Squeak NOS Project Website
http://sourceforge.net/projects/squeaknos/
http://squeaknos.sourceforge.net/

9.11 GemSqueak

GemSqueak is a full functional client for the GemStone/S object database. It supports MVC and

Morphic tools and has GemStone Browsers, Inspectors, Explorers, Debugger, SessionBrowser, Workspaces. You can evaluate GemStone code (do it, print it and inspect it) using regular menus and shortcuts in all GemStone panes. You need a running GemStone/S database and the dynamic link library that comes with GemBuilder. There is a non-commercial version of the GemStone/S database server available at [2] you can use to test it.

References

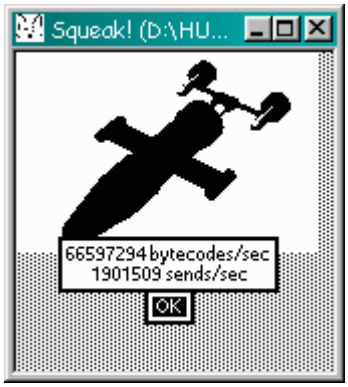
[1] GemSqueak – Squeak Client for Gemstone/S
http://minnow.cc.gatech.edu/squeak/1957
[2] Gemstone/S
http://www.gemstone.com

9.12 Squeak GUI Construction Kit

References

[1] Squeak GUI Kit Webpage
http://squeaklet.com/sguikit/

9.13 Micro Seeker – autonomous control system using Squeak



References

[1] Micro Seeker
http://www.huv.com/index.html