Squeak@21c3

Marcus Denker

`www.squeak-ev.de/`

December 4, 2004

1 Introduction

This is not a real article. While putting together the demo image for for 21C3, I decided to not write an article that is just to be read (mostly because these are really boring to write...)

This text is just a short 'user manual' for that thing (we call it *Squeak Image*) that I will use for the demo at 21C3. So if you follow the instructions, you will be able to go through the slides and play with everything yourself.

2 Installation

The first thing we need to do is, you guessed it correctly: Installing Squeak on your system.

Squeak has been ported to everything, from handhelds to mainframes, or even game systems like the PS2. Not all of the ports are kept alive for all releases, but for a common system like Linux, MacOS, and that-which-shall-not-be-named there should be no problem.

So go to squeak.org and download Squeak for your system and unpack. The files we will need for running the demo are:

- **SqueakV3.sources:** The source code. All in one big lump.

- **Squeak.exe / Squeak.app / squeak:** This is the virtual machine.

that's it. And the files from the 21c2 demo:

- **21c3.image:** all the Objects that make our demo, dumped into a file

- **21c3.changes:** additional source code

You can dowload these here: `http://squeak-ev.de/21c3`

Just put these files in a directory, and you are ready to start.

3   Start Up

Now we need to start Squeak. For that, just open the "image" file with the help of the vm. For linux, that would be something like:

```
./squeak 21c3.image
```

MacOS/Win: drag-n-drop the image on the vm.

Now the Squeak virtual machine will start and load the image back into memory. The image is really an *image*: It's just a snapshot of the whole memory that the vm had allocated. It will start up exactly the way it was when I saved it.

So if I, e.g., have a editor open with the courser blinking at a certain spot, bang: It's blinking right there. And this is portable across all systems: I saved on a mac, you load on Linux. Write once, run everywhere (but with Squeak, this really works! Not *write once, debug everywhere* like with that other system).

So, I saved the demo image in a way that you can start to look at the slides right now: When the window opens, it's around 1024x768 large, and shows a friendly mouse in the middle.

4   Navigating the Slides

At the lower right corner you see a thing called "Thread Navigator". This handy Object allows for easy navigation in the slides of the talk. Just click on the arrow, and the next slide appears. The Tread Navigator object will follow to the next slide. (It will really travel with you: It deletes itself from the old slide and hops over. So make sure not to loose him).

Each slide is a *Squeak Project*. These projects are a bit like virtual desktops, but they can be saved to disk or send over the net.

You can make graphical links to projects, clicking on those will make the linked project active. To go back were you came from, you need to click on the background: A Menu appears, with items for going back and lots of other entries, e.g. for opening tools.

5   Squeak - What's that?

Good Question. And not simple to answer. With most other Open Source projects that question is trivial: Mono, that's a free version of .Net. Gimp is a free Photoshop. But for Squeak, there is just nothing that is like it.

Squeak has features from a lot of different domains and well known programs. e.g. it has lots of features for media authoring, it is a programming language, it is a kid's programming system, it's (in a sense) it's on Operating System, it's a development environment...

So let's go to the slide "What is Squeak", (should be the fourth). There you'l find links to projects for each of these aspects. Follow them, play around. Keep in mind that these slides are all "living". You may be used to the fact that a slide is all pictures, but in Squeak, that's not true.

The examples are all live. In the "Development" project, the Class Browser is showing the system that you are interacting with. Same with the "Inspector", the grey smaller window. It shows the inside of the Object that is the background, and from that especially the list of all other graphical objects that are in the slide.

So if you click on the background, the menu will appear and the Inspector will be updated.

Play a bit, we will revisit some topics later.

6   Squeak - The History

The History of the Squeak Project is nothing but Amazing (with capital A!), When I stumbled across Squeak, I knew nothing about the people involved, and somehow I was expecting that it would be some student somewhere (or something like that, like it was with Linux). But then I realised that those guys were a bit older. And bolder:

I read on the Squeak.org website (it's even there today):

Our number one commitment is to an exquisite personal computing environment. Imagine a system as immediate and tactile as a sketch pad, in which you can effortlessly mingle writing, drawing, painting, and all of the structured leverage of computer science. Moverover imagine that every aspect of that system is described in itself and equally amenable to examination and composition. Perhaps this system also extends out over the Internet, including and leveraging off the work of others. You get the idea – it's the Holy Grail of computer science.

Who can write something like that? Seriously! Holy Grail. Sure

So let's see....

The hero of our story is named Alan, the time is somewhere at the end of the sixties. Great time for being a computer science researcher, as the US military just pumped huge amounts of cash into basic computer research with the ARPA program. Lots of cool stuff was invented in that time, e.g. packet switching networks (ArpaNet), and Alan got to see some cool stuff: The first prototype of a flat panel display, one of the first handwriting recognizers and the drawing program SketchPad (according to some *the most important program ever written*). Papert's LOGO programing language for kids. And Moores Law.

So, he got an idea. A crazy one: Why not build a computer that is small enough to be carried around. It should have the size of a notebook, be around 2 pounds. It should have a graphical display, have enough memory to store a couple of books (It even could have a harddisc. A small one). A true "Personal Computer". Build for Kids. Codename: Dynabook.

Today, this sounds a bit obvious, but back then, it was completely crazy. The best way to predict the future is to invent it.

But the technical aspect is not the only interesting one: The idea was to build a computer for kids in the way that it would be a new kind of medium. For fun and learning. Not a word-processor, but an idea processor.

So the goal was to make this crazy dream a reality. Moores law gave the deadline: A sufficiently fast computer would fit, according to Moores law, inside the required space in 1980. So Alan started to think about the Software, at Xerox PARC.

The Operating System, GUI and development environment for the Dynabook was called Smalltalk. It was the first fully Object Oriented System, and the first GUI with real, overlapping windows. The famous story is that Steve Jobs did get a demo of that system....

And Squeak is that Smalltalk System from the seventies, dusted off. And those strange Squeakers are the inventors of Smalltalk, Alan Kay of course, and Dan Ingalls and Ted Kaehler, both from the original Xerox PARC team.

Squeak was started in 1996, while the Squeak Team was at Apple. They moved on to Disney from there, and today, Alan is at HP Labs.

7   Squeak for Kids

So, now go forward some slides to the *Squeak for Kids* project.

There is a good tutorial on the Squeakland website about how to build a car yourself:

`http://squeakland.org/school/drive_a_car/html/Drivecar12.html`

The Lunar Lander is an example of how far you can go with eToys: This was not done by a child, but with some guidance older kids could build such a game (and learn a lot).

There are a some other examples for eToy projects in the third (the right) project.

8   Squeak for Professional Development

eToy is great for kids (and even for adults to learn programming). But the real hacker needs something that is a bit different. Squeak is completely written in itself, and for making that easy, it provides a object oriented programming language and a development environment.

The IDE provides a class browser for easy coding and a really good debugger. (with edit-and-continue, something that is starting to get a hot topic for both Java and .Net).

Squeak is actually used both in research and in industry to do real stuff. *This is no toy!* (Even if it looks like one).

9   The Language

The language of Squeak is really simple. But it's a bit different then the other ones you might know. The slide has all the syntax of the language. That's

it. In the slide, you can execute and print a statement by selecting it and pressing Alt-p or Apple-p on the Mac.

The basic idea is that everything is an object, and you can make object do something by sending a message to them. What an object does when receiving a message is described in a method. The method is defined in the class of the object.

So if you write:

```
1 sin
```

then you have an object (the number 1) and you send a message (sin). The object 1 is of class SmallInteger, and in SmallInteger there is a method `sin`.

A real smalltalk introduction is outside of the scope of this paper, see

```
http://www.iam.unibe.ch/~ducasse/FreeBooks.html
```

for some books.

And there is a good introduction slide-show in the current 3.7 or 3.8 Squeak release images.

10 Seaside

Seaside is quite interesting. Avi Bryant was using Ruby for doing web development. Then he met some Squeakers at OOPSLA (the OO Conference). And so he jumped ship from Ruby to Squeak and invented Seaside.

Seaside is very interesting, as it solves the problems that any web developer faces in novel ways. The target for Seaside are web applications, that is fairly complicated interactive websites (e.g. the typical banking site), not static pages.

Seaside allows the programmer to write the web application just the way she would write a normal GUI application. Let's see what Avi writes:

Seaside is a framework for developing sophisticated web applications in Smalltalk.

Its most unique feature is its approach to session management: unlike servlet models which require a separate handler for each page or request, Seaside models an entire user session as a continuous piece of code, with natural, linear control flow - pages can call and return to each other like subroutines,

complex sequences of forms can be managed from a single method, objects are passed by reference rather than marshalled into URLs or hidden fields - while fully supporting the backtracking and parallelism inherent to the web browser.

Seaside also features a callback-based event model, a "transaction" system for auto-expiring pages, strictly compliant XHTML generation, a system of reusable and embeddable UI components, and handy web-based development tools.

The Squeak image that drives the slides has a complete seaside installation. Yes, everything is there. Just point your browser to

```
http://localhost:9090/seaside/presentation
```

for a Seaside demo written in Seaside.

The Counter/MultiCounter are here:

```
http://localhost:9090/seaside/counter
http://localhost:9090/seaside/multi
```

And a small shop example:

```
http://localhost:9090/seaside/store
```

A short article about seaside: `http://homepage.mac.com/svc/ADayAtTheBeach/` It ends with:

We 'discovered' Seaside only recently. After just one day of experimenting with it we were convinced about the dramatic improvement in abstraction and productivity it offers. We were absolutely amazed at how easy it was to do the examples described here: much, much easier than it would be in any other framework that we know of. We believe Seaside could be a killer application for (Squeak) Smalltalk.

Seaside is cool, yes. But we can do even better. Enter Croquet.

11 Open Croquet

So. Now to the really interesting "latest and greatest" in Squeak. As with Squeak itself, we need a good quote from the developers. Or better two:

7

WHAT IF...

...we were to create a new operating system and user interface knowing what we know today, how far could we go? What kinds of decisions would we make that we might have been unable to even consider 20 or 30 years ago, when the current set of operating systems were first created?

...we could collaborate with one another in an online dimension to create or simulate anything we wanted to?

...we had the robustness of a 3D immersive technology, the diversity of the Internet, and the degree of social interaction we have in the real world?

The second quote:

Existing operating systems are like the castles that were owned by their respective Lords in the Middle Ages. They were the centers of power, a way to control the population and threaten the competition. Sometimes, a particular Lord would become overpowering, and he would declare himself as King. This was great for the King. And not too bad for the rest of the nobles, but in the end – technology progressed and people started blowing holes in the sides of the castles. The castles were eventually abandoned - David A. Smith

Croquet is work-in-progress. But there is a pre-release available for download. If you have a fast computer with a good 3D graphics card, give it a try.

12   Links

```
http://squeak.org
http://squeakland.org
http://squeakersfilm.org
http://croquetproject.org
```