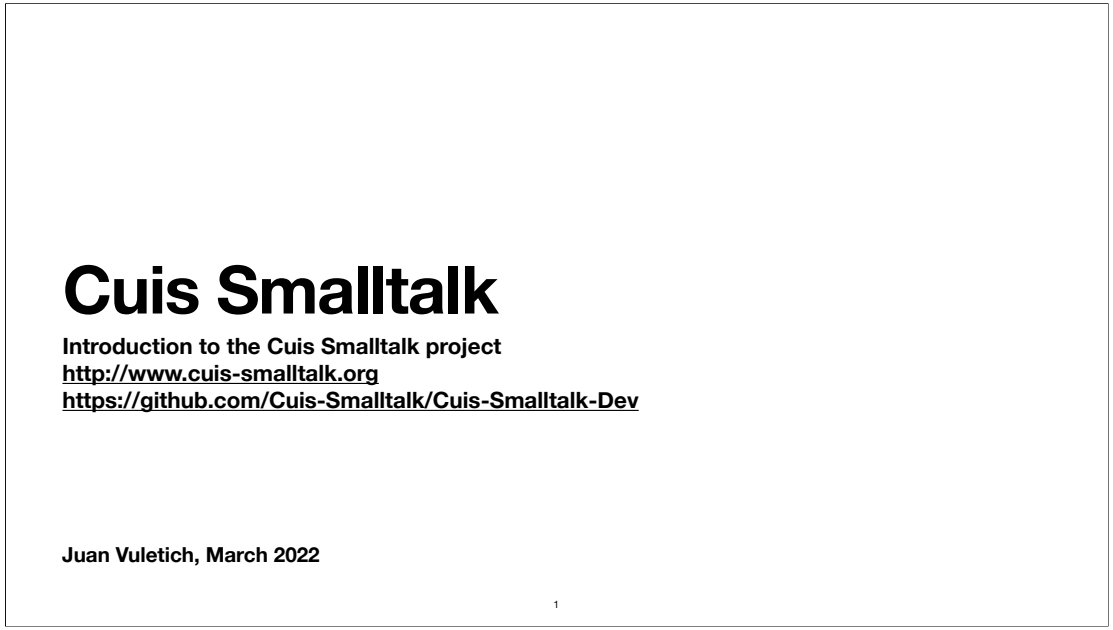


Today I will tell you about the Cuis Smalltalk project. I'll talk about the ideas behind Smalltalk-80, how they came to be, and how they enabled the development of many Smalltalk systems over the years.

Also, I will talk about how I got into this world, and how Cuis Smalltalk was started and has been developed until today.

Tomorrow it is demo day. I will show you Cuis, the programming tools it includes, and the kind of things you can do with it.

Juan Vuletich

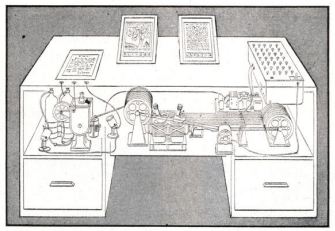
A bit about me

- Born and based at Buenos Aires, Argentina
- Software developer, M.Sc. in Computer Science
- Specialized in Smalltalk systems & VMs, Image and Music Processing
- Cuis Smalltalk founder and project leader
- VM Developer at LabWare since 2019

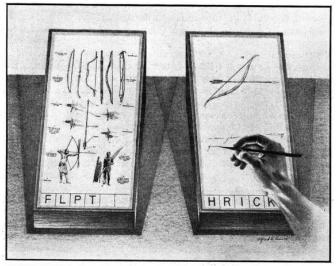
Let me tell you a bit about me. I am the founder and lead developer of Cuis Smalltalk. I am 51 years old. I was born in Buenos Aires, where I live with my wife Lucía and my daughters Sofía and Diana.
I studied Electronics and Computer Science. I have been programming since I was 14. While in high school, I wrote a music synth and a sampler in 6809 assembly language on my Radio Shack Color Computer. I also hacked a word processor to display and print Spanish accents.
I got my first programming gig at 16, and a real job as software developer at 18.
I have master's degree in Computer Science from the University of Buenos Aires.
I have been an active member of the Squeak community since 1997, when I also adopted Smalltalk as my main development environment and language both for work and personal projects.
For the last 3 years I have been working at LabWare as VM engineer.

Memex

By Vannevar Bush



MEMEX in the form of a desk would instantly bring files and material on any subject to the operator's fingertips. Slanting translucent viewing screens magnify supermicrofilm filed by code numbers. At left is a mechanism which automatically photographs longhand notes, pictures and letters, then files them in the desk for future reference.



MEMEX IN USE is shown here. On one transparent screen the operator of the future writes notes and commentary dealing with reference material which is projected on the screen at left. Insertion of the proper code symbols at the bottom of right-hand screen will tie the new item to the earlier one after notes are photographed on supermicrofilm.

- First concept of a “knowledge amplifier” based on modern technology
- “As We May Think”, published in The Atlantic in 1945

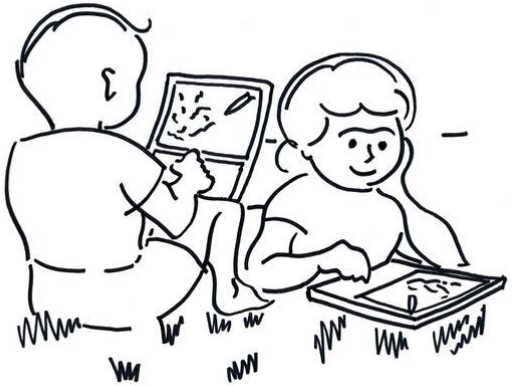
3

In 1945, Vannevar Bush published his influential article “As We May Think”, where he introduced the Memex. Bush was convinced that modern technology would enable the construction of “knowledge amplifiers”, that could help people better understand complex problems, and think deeper about them.
The Memex was a concept for a personal hypermedia system. Users could add books, pictures and drawings. They could link them, adding notes and creating additional material on the go.
In addition to navigate them by following links, they could also use sophisticated query mechanisms to search and retrieve material.
If this makes you think about the Web and Google, you are right. This is were those ideas were born. Still, there is a big difference. The Memex was to help the user organize, find and understand his own collection of reference material, and share it with others. But current web platforms encourage the user to become mostly a passive consumer, while it is the platform itself who organizes and filters content.
Bush inspired many thinkers and inventors, who shaped the evolution of technology in the latter twentieth century.

The Dynabook / Smalltalk Project at Xerox PARC

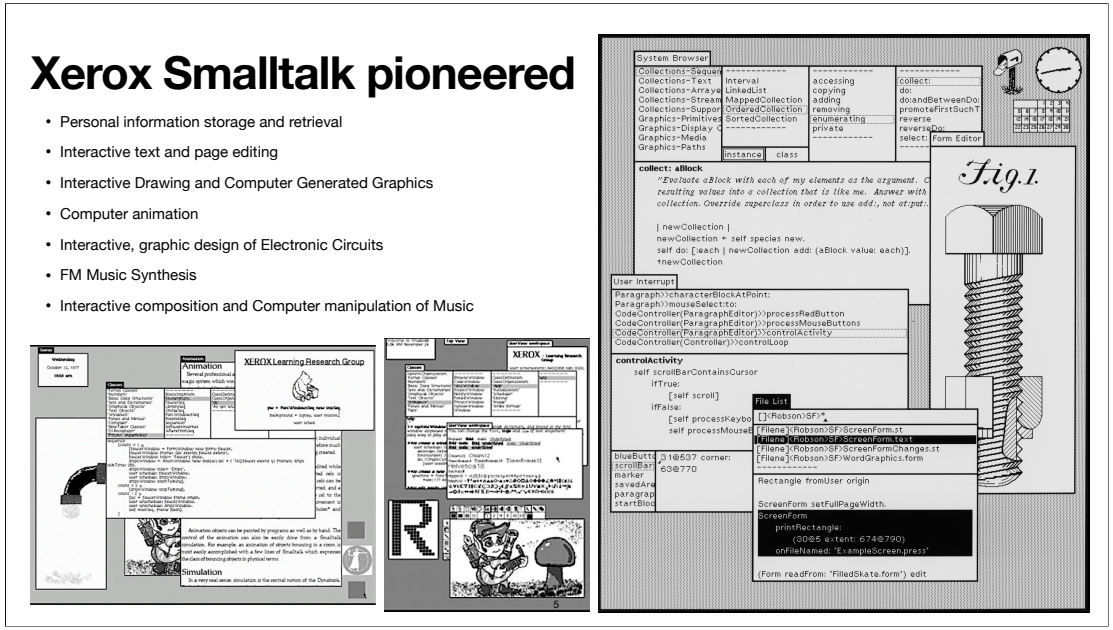
by the Learning Research Group

- A new media for the expression of thoughts and knowledge
- Useful for scientists
- Usable and fun for kids
- Also a laboratory for running experiments and simulations
- A new meta-media for discovery and creation of new media



4

In the 1970's Xerox Corporation set up the Palo Alto Research Center to invent "the office of the future".
The most unconventional team there was the "Learning Research Group", led by Alan Kay. The original goal of their project was to turn a computer into a personal tool for learning, creating, and expressing knowledge. Alan Kay called this dream computer a "Dynabook".
This tool should be powerful enough to be a significant advance in how scientific knowledge is created and published. It should also be friendly enough to be useful and fun for primary school children.
It would be used as a laboratory for doing experiments and simulations. In fact, it would a meta-media, for the creation of new media, new languages and forms of expression.
After several years of intense work doing research and development in Learning Processes, Interactive Systems, Knowledge Representation and Programming Language Design, Smalltalk-80 was the culmination of this project.



These are some of the things that were pioneered or even done for the very first time on the "Interim Dynabooks" built at Xerox, using Smalltalk.

The Xerox Alto machines, the first personal computer with bitmapped graphics, was designed and built at the Xerox Palo Alto Research Center.

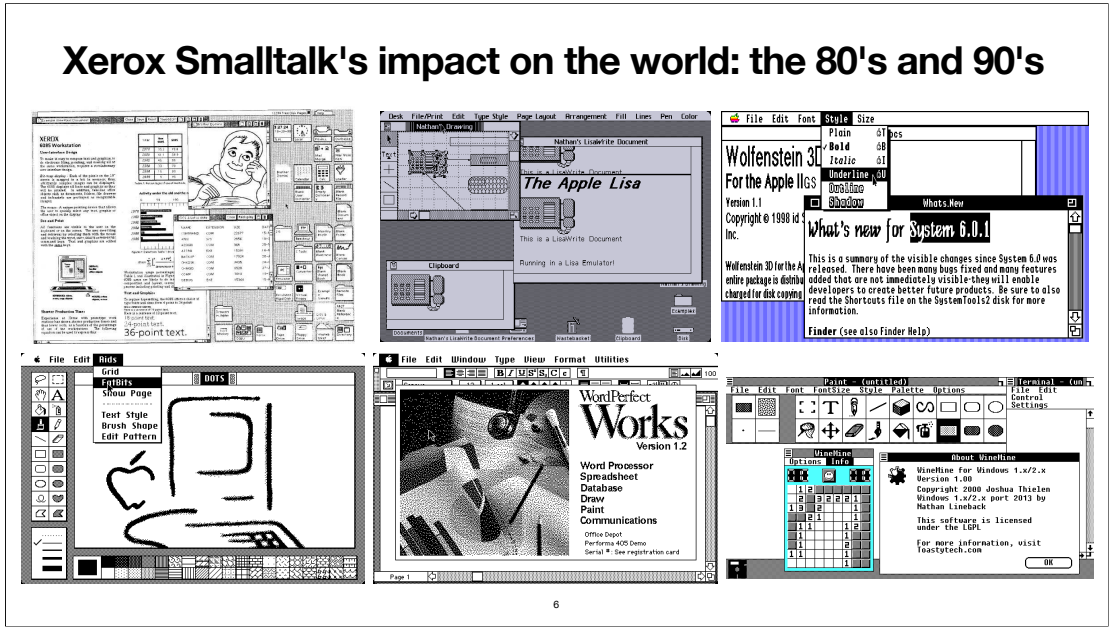
Besides building the Smalltalk system for them, the Learning Research Group, together with several guest scientists and middle school students, started to experiment with the kinds of media that were possible, and how to interact with them.

They created word processing and page design software that could preview on the screen and later print to the laser printer that was just invented there.

They built interactive drawing applications like those used by professional designers today, and tools for the graphical design of electronic circuits.

They created some of the first computer animations and the software to run them.

They also did many experiments in music synthesis, interactive composition, and computer music.

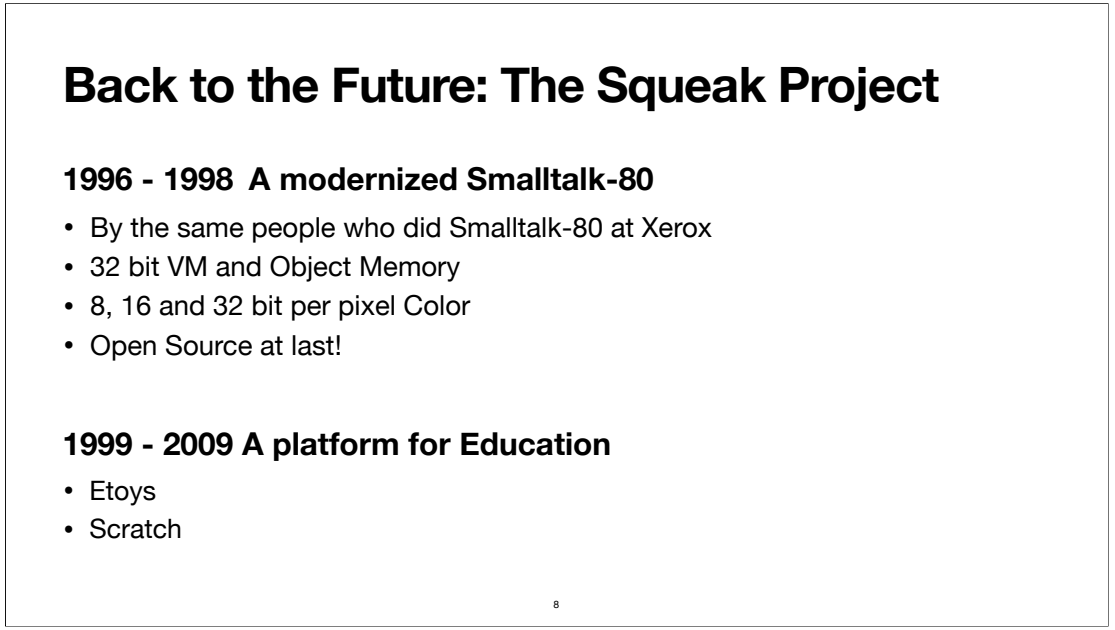


This is not our focus today, but we can't ignore that Smalltalk was the basis for all the graphical user interfaces developed in the eighties and nineties.
In this slide we can see screen captures from the Xerox Star, the Apple Lisa, the Apple IIGS, the Apple Macintosh, WordPerfect on the Macintosh and Microsoft Windows.

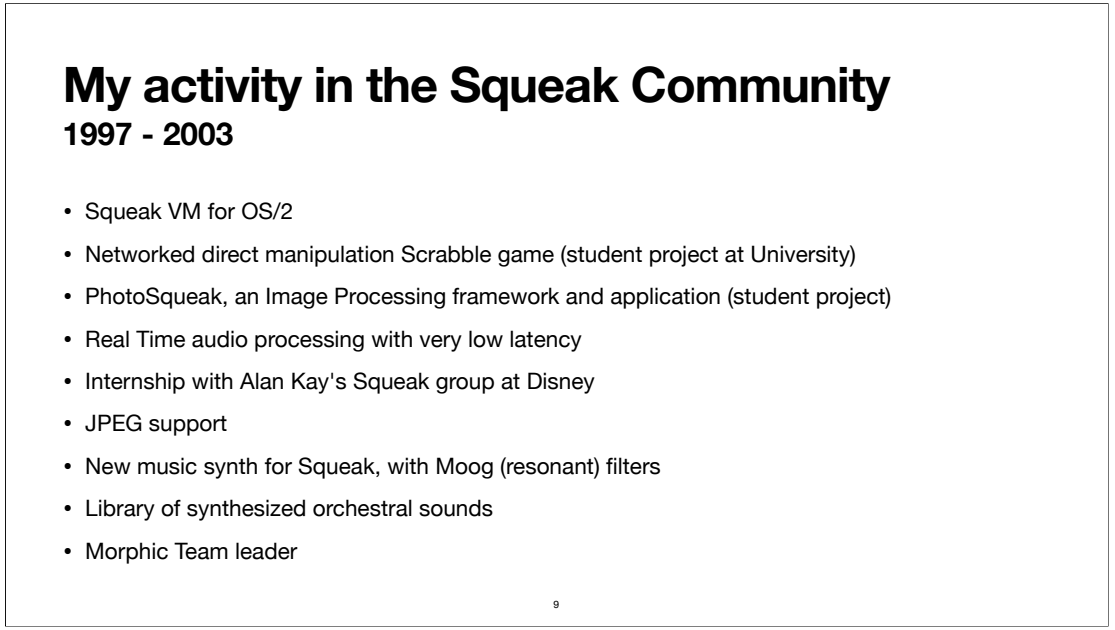
Smalltalk-80 also inspired several groups and companies to build Smalltalk systems. Most of them were professional development tools, sold by companies. A few were research projects.

These commercial Smalltalk systems are meant for application development. The applications built with them should look and feel just like any other program that runs on the user's machine, so they try to be as close as possible to the host operating system. They have paid licenses, and don't publish the code for the virtual machine and critical parts of the system. They try to be part of a healthy business, so they focus on making money, and helping customers make money too. All this means departing from the goals and style of Smalltalk-80, and going in a different direction.

On the other hand, several research systems were built to expand in new directions the ideas of Smalltalk-80. They have succeeded at their objectives, making these new ideas real, and advancing the theoretical knowledge on their field. However, these research systems were never mature enough for wider adoption.



This landscape changed completely in 1996, with the creation of Squeak Smalltalk. To develop Squeak, Alan Kay formed a group that included Dan Ingalls, Ted Kaheler and Scott Wallace from the original Smalltalk group, together with John Maloney from the team that built Self and Morphic.
It was a direct descendant of Smalltalk-80, and it was true to its goals and design ideas. Making it open source and freely available on the Web meant that an enthusiast community of collaborators grew around it.
Squeak later made its focus to become a platform for education, drifting away from the design principles that had guided Smalltalk-80. This started a whole new generation of educative software, that includes Etoys and Scratch, and many other tile-based programing systems for education.



I've been part of the Squeak community from the start. After learning about Smalltalk at the university, and using the Windows only Smalltalk Express, to me, Squeak was a revelation. It was platform independent like Smalltalk-80, but took advantage of modern displays and 32 bit systems. On top of that it was an open, collaborative project, led by my heroes, and it welcomed contributions.
Around that time I met my friends Andrés Valloud and Boris Shingarov for the first time, among many great people in the community.
In those very early days, Squeak could run on Mac, Windows and Unix. I didn't own a Mac, and those were the days of fragile Windows 95, and immature Linux, so I had adopted OS/2 as my platform and C as my programming language. In a few weeks I ported the Squeak VM to OS/2.
I used it for several projects you can see in the slide. All of them were published on the Web.
I also had the incredible honor of being part of Alan Kay's group for an internship, working on graphics and music.
When the Disney era ended, Dan Ingalls left the management of Squeak, and the community took responsibility for it. For some time I was the leader of the Morphic team, with the goal of cleaning it and simplifying it, to enable further evolution.

The birth of Cuis Smalltalk

2003 - Reasons for starting a new Smalltalk project

- Independence of Pixel Density
- Zoomable User Interfaces
- Vector Graphics based UIs

Me: “Should I wait?”
Alan Kay: “Good question. The answer is always the same: No. You should never wait.”

10

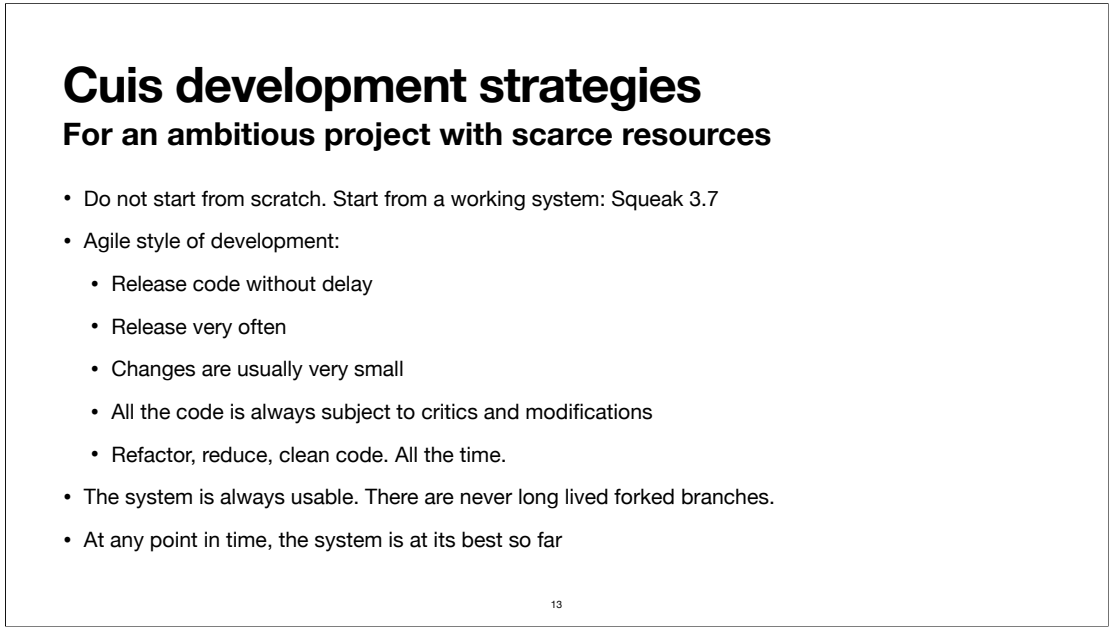
There weren't any retina displays yet, but it was clear to me that GUIs designed for a specific pixel density had no future. I wanted a Morphic system where everything was specified independently of pixel size or resolution.
But a large community of people, without a strong leadership, and with a deep devotion for their heritage, was not ready for the disruptive changes that were needed to go where I wanted to go. Squeak was becoming "Legacy Code”.
In 2003 I traveled to San Diego to present my master's thesis at a conference on wavelets. Tried to make the most of my trip to the U.S. and asked Alan if we could meet. He was kind enough to invite Lucía and me for lunch at L.A. During that talk, I told him about these ideas. About that time, Andreas Raab was working on his own replacement for Morphic, called Tweak. So I asked Alan: "Should I wait for Tweak, and maybe adopt it?". He said: "That is a very good question. And the answer is always the same. No, you should not wait. You should never wait."
I followed his advice. Cuis Smalltalk was derived from Squeak in 2004, and ever since it has been evolving into a modern incarnation of the Smalltalk-80 spirit.

The Smalltalk-80 spirit. In "Design Principles Behind Smalltalk", Dan Ingalls tells us that Smalltalk-80 was built to provide computer support for the creative spirit in everyone. This article, published in the August 1981 issue of the Byte magazine, was the first description of the ideas that drive Smalltalk, that would reach a wide audience.
Some of them shape the design of the Smalltalk language itself, and therefore are followed by any Smalltalk system.
But others are more general guidelines we may choose to follow when building any kind of software. Some of these are:
1) Personal Mastery: If a system is to serve the creative spirit, it must be entirely comprehensible to a single individual.
2) Good Design: A system should be built with a minimum set of unchangeable parts; those parts should be as general as possible; and all parts of the system should be held in a uniform framework.
3) Modularity and Encapsulation: No component in a complex system should depend on the internal details of any other component.
4) Factoring: Each independent component in a system would appear in only one place.
5) System boundaries: It is natural to ask what set of primitive operations would be sufficient to support an entire computing system. The answer to this question is called a virtual machine specification. This means doing Encapsulation of the supporting platform itself.
6) Reactive Principle: Every component accessible to the user should be able to present itself in a meaningful way for observation and manipulation.

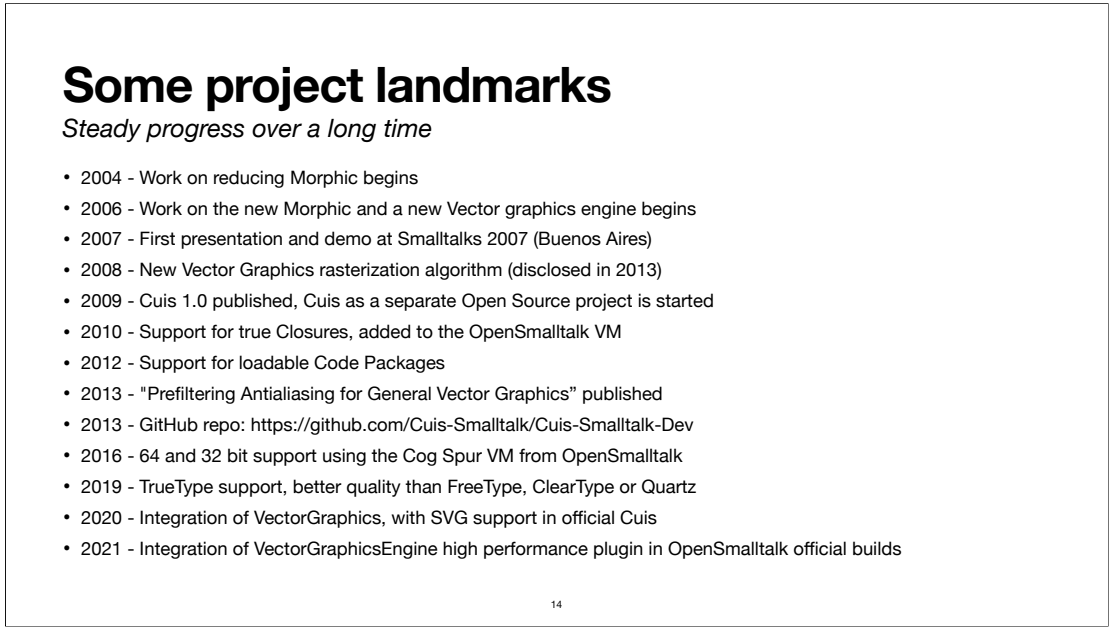
Goals for Cuis Smalltalk

- Recover Smalltalk-80 simplicity and attitude
- Include only kernel classes, GUI framework, and dev tools
- Everything else should live in optional loadable modules
- Enable the creation of the next generation Morphic framework
- Make the system good for experimenting, learning, and especially for documenting and transmitting knowledge
- Make it a Dynabook

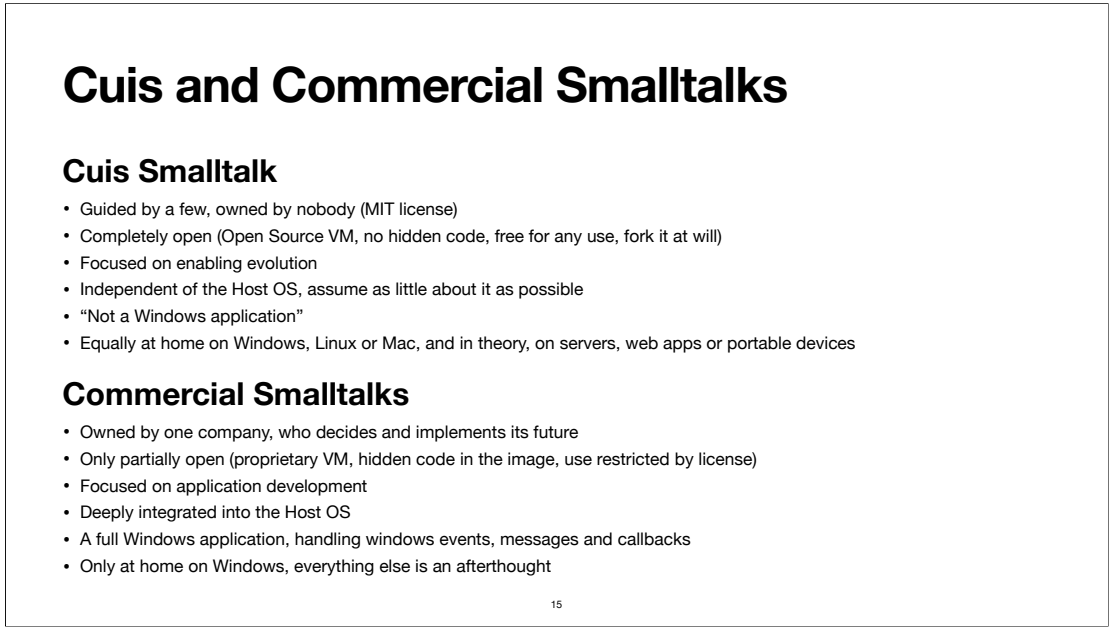
The main objective of Cuis is to build a Smalltalk system that follows these ideas. This means:
- Cuis should be simple. Any unneeded complexity should be removed.
- It should include only kernel classes, user interface, and developer tools
- Everything else belongs in optional packages
- All this should make the next Morphic possible
- It should also make the system a good place for learning and sharing knowledge
- The ultimate dream is of course to make the Dynabook real



Cuis was an ambitious project even before starting. I am a single individual, and Cuis would not be a full time project, but something to be done in parallel with paid work. Would I be able to actually do better than Squeak? Would I have the time and energy that could be needed to build what I wanted? Would I actually be able to build a pixel independent Morphic framework, and the vector graphics engine required to run it? I really didn't know. So I followed a development strategy that would, at all times, deliver the best I had done so far.
The key idea is that the published system is always updated, and always usable.
For this, I decided not to start from scratch, but start from the existing system that was the closest to what I wanted. At that time, Squeak 3.9 was about to be released, and Squeak 3.8 had been released a while before. But both these releases had added additional complexity, beyond what was needed. So I decided to start from the slightly older Squeak 3.7.
I also decided to follow a style of development you could call “Agile”, doing short iterations, often daily, and refactoring existing code to ease its evolution.
There were never forked branches, and the system was always in a usable state.
If for whatever reason I needed to stop working on the project, the last commit made would be an improvement over all the previous ones.

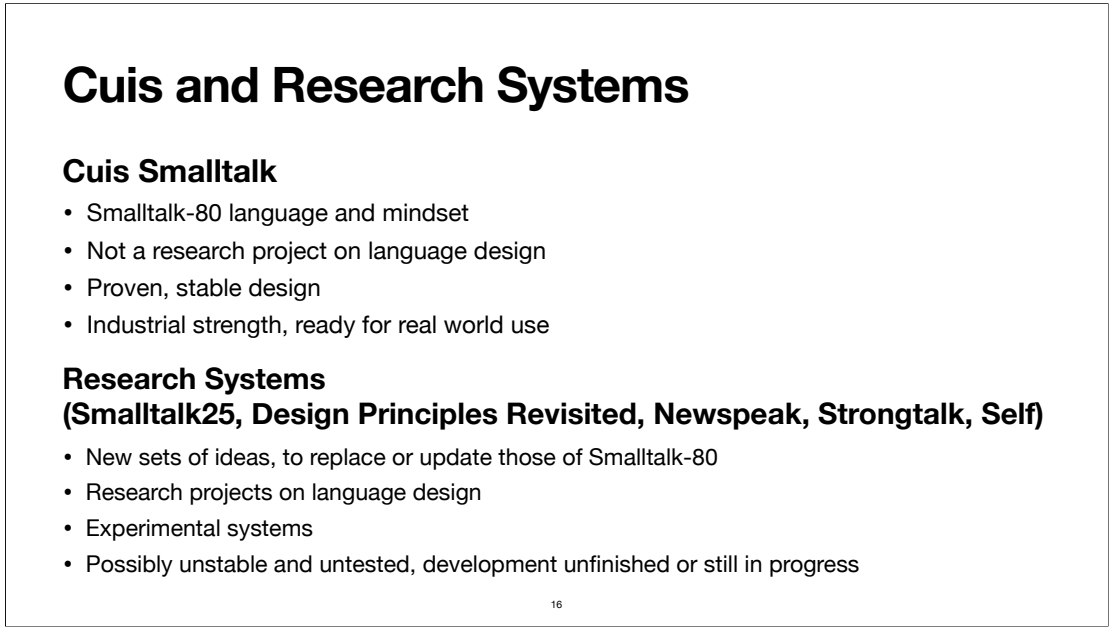


These are some project landmarks over 18 years.
During this time, Cuis was updated to the latest developments in the Squeak world, adopting the OpenSmalltalk VM, that included support for true block closures, a high performance jitter, and later, 64 bit support.
To render anti aliased vector graphics, I developed a new algorithm, based on Signal Processing theory, and not on the conventional “pixel coverage” idea. This was published in the “Prefiltering Antialiasing for General Vector Graphics” defensive disclosure in 2013. A noteworthy aspect of this technique is that the quality of the result is so good that it can be used also for drawing text, and still give better text quality than conventional TrueType rasterizers, without the need for text specific techniques.
In parallel, I kept working on the redesign of the Morphic framework itself. This work on Cuis Smalltalk itself, but the Vector Graphics engine was developed outside of the basic Cuis image. The basic Cuis image only used the BitBlt based graphics engine that was inherited from Squeak until 2019. This helped Cuis stay clean and focused, and eased its adoption by many users.
In 2019, I used the Smalltalk version of the Vector Engine I wrote, to draw TrueType glyphs, adding TrueType support to the base Cuis without needing any platform support. This meant that no platform dependencies were introduced, and Cuis stayed completely portable.
In 2020 and 2021 I developed a virtual machine plugin that implements my vector graphics engine. This means that Cuis can now run using only zoomable vector graphics with good performance and top quality, becoming the first and only Smalltalk system to do this. This plugin doesn't include any platform specific code, and it was integrated into the official OpenSmalltalk VMs.

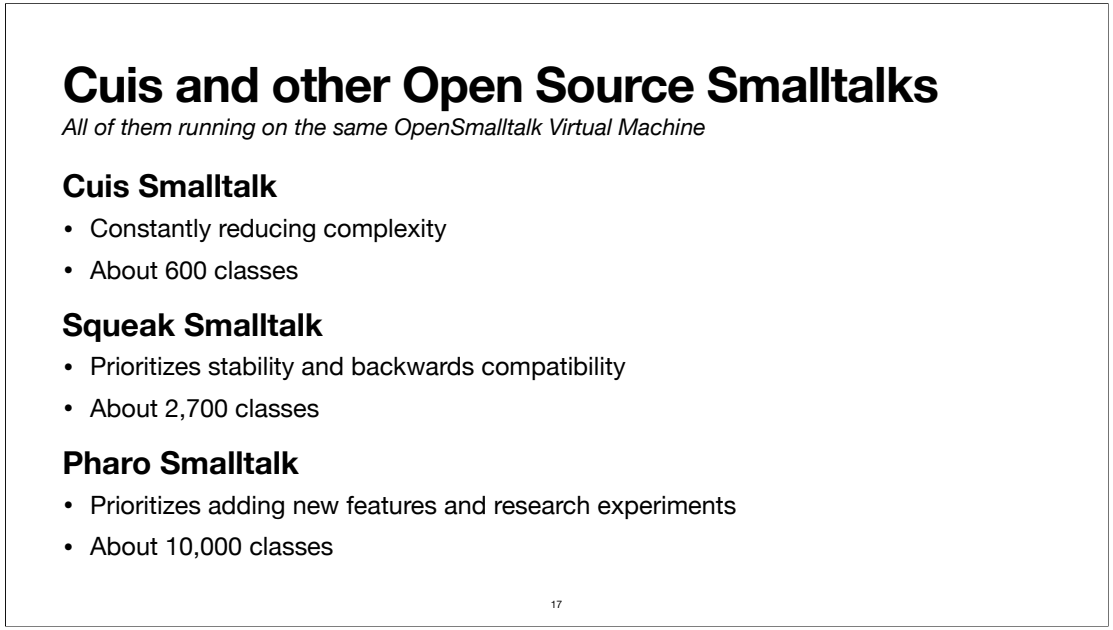


An Open Source Smalltalk like Cuis is managed in a very different way than a commercial product. The focus is different. The goals are different. The things that can be done with them are different.
Open Source projects enable cooperation, because there are no restrictions to learning and modifying any part of the system. In addition to that, as there are no restrictions to use them, people are encouraged to adopt them, and share their own work in the same way.
But commercial products have other priorities. This leads to user licenses that restrict modifications of part of the system. They also restrict the way people can share their own work, because it will be useful only to other people who have also paid a license to the owner of the system. As a result, developers using commercial tools don’t share much code in this way.
Besides, Cuis follows the Smalltalk-80 tradition of platform independence. This means making few assumptions on the underlying platform. Essentially all we ask is for a graphics display where we can access all the pixels. Any additional capabilities of the host platform user interface will not be used by default. The Morphic user interface framework is entirely written in Smalltalk, and the virtual machine support it uses doesn’t deal with platform GUI elements, just pixels and vector graphics. User input is read from the operating system when needed. There are no callbacks from the host into Smalltalk code. This platform independence means that application, at least in theory, can run on any platform, including the web and portable devices.
Commercial systems work in a different way, because they are focused on native application development. These applications should look and feel just like any other program that runs on the user's PC. Essentially it means that these commercial Smalltalks need to

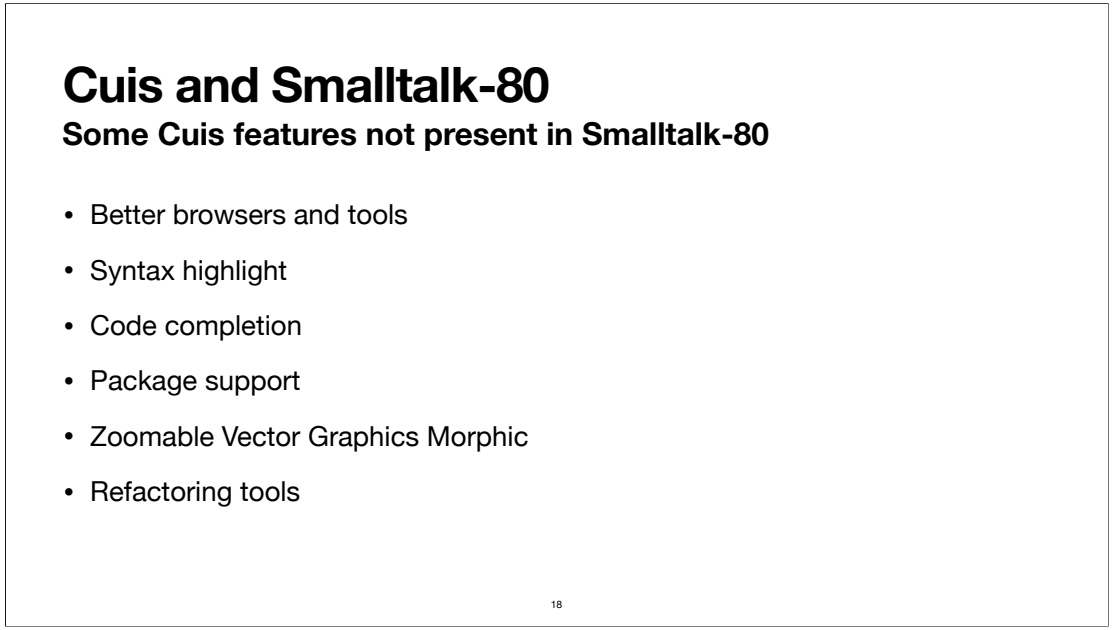
become “Windows Applications”. This means conforming to a very specific framework on how to receive the user input by means of “windows events” and “windows messages”, reacting to “windows callbacks”. It also means that the whole user interface is made of “windows controls”, that are actually built and run by the Windows operating system. This also means that applications written with a Windows specific Smalltalk can not be easily ported to other platforms.
It is important to understand these differences when choosing a Smalltalk system for a particular project.



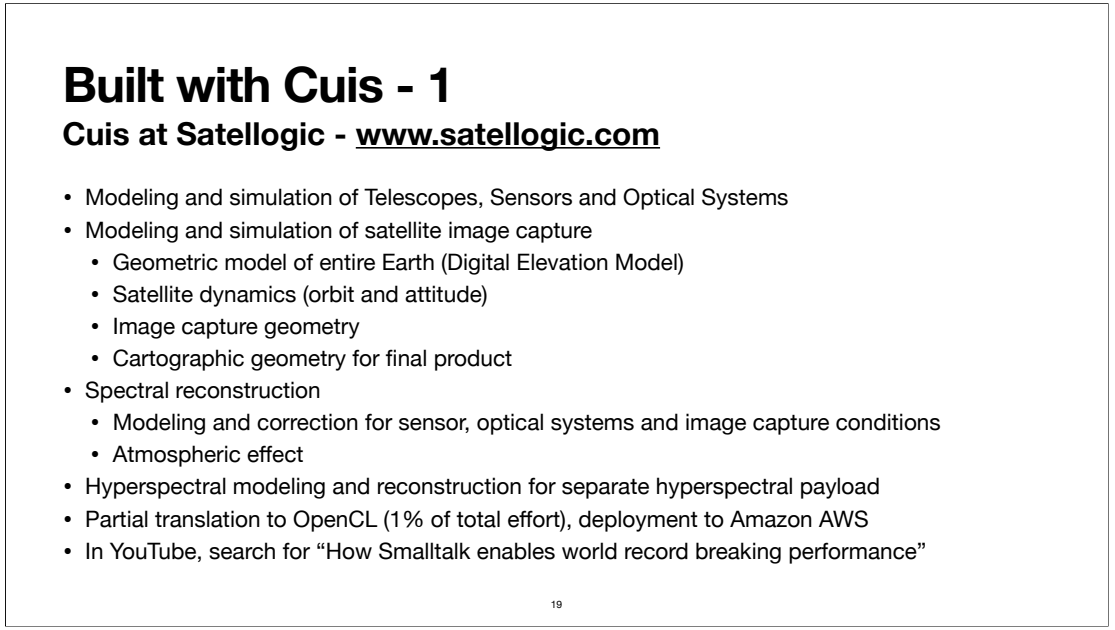
There have been several attempts to find out what comes next after Smalltalk-80. Boris recently told us about the Smalltalk-25 project he is working on together with Jan Vrany, and also told us about Andrés Valloud quest for it in his "Design Principles Behind Smalltalk Revisited". Other projects that have had a related focus are Newspeak, Strongtalk and Self.
These projects are important to our future. But they are research projects, diving into the unknown. They don't provide a practical working environment, and those who did in the past were never completed and polished to the point of being ready for broad adoption.
Cuis is nothing like that. It doesn't try to be the next thing after Smalltalk-80, but a Smalltalk-80 useful in today's world. Still, Cuis can be a useful vehicle for the implementation of those research projects.



You could ask: Why there are three different Open Source Smalltalks, that even run on the same Virtual Machine? Well, they have very different goals. I believe that the only one that clearly states its objectives and follows them is Cuis.
But, by watching its development over many years, I think it is fair to say that Squeak makes stability and compatibility a priority. The Squeak project protects the heritage from the Squeak system built by Dan Ingalls and Alan Kay. This is a very reasonable goal, and it is great that they do it. As a consequence, while they do and welcome improvements and bug fixes, these tend to be fine grained. Disruptive changes are completely out of the question.
With respecto to Pharo, aside from the broad and possible contradictory goals stated in their web page, (and the inexplicable absence of the word "Smalltalk" in it), I believe their true priority is adding new features, and new implementations of old features. Sometimes these are research experiments of questionable value to other users. Additionally, they have a very idiosyncratic style of leadership, that may not fit everyone.
Still, there are many people who are active members of two or all three communities, contribute code and ideas to them, and make cross-pollination possible. After all, the three projects share essentially the same VM, image formats, and many other technical details.
I don't think that having several Open Source Smalltalks is a problem at all. The world is large enough for all of them.



Although Cuis tries to follow the Smalltalk-80 ideas and spirit, compatibility with Smalltalk-80 is not a concern. Computers are incredibly much faster than even supercomputers of the 70's, so we can do many compute intensive things. Additionally, the GUIs originally copied from Smalltalk have evolved quite a bit and people are used to many standard features. And of course we have learned a lot of new approaches to solving problems, that we can use.
Cuis has improved Smalltalk-80 in many ways, but given that it only includes basic libraries and development tools, this hasn’t meant an explosive growth. The number of classes in the system is similar to that of the first versions of Squeak, before the development of Etoys.



Cuis has been put to good use from the start. Between 2011 and 2017 I worked at a startup company named Satellogic. Its objectives were extremely ambitious: to design, build and operate a new generation of cheap satellites to capture images of the Earth. I was in charge of everything related to Image Processing.

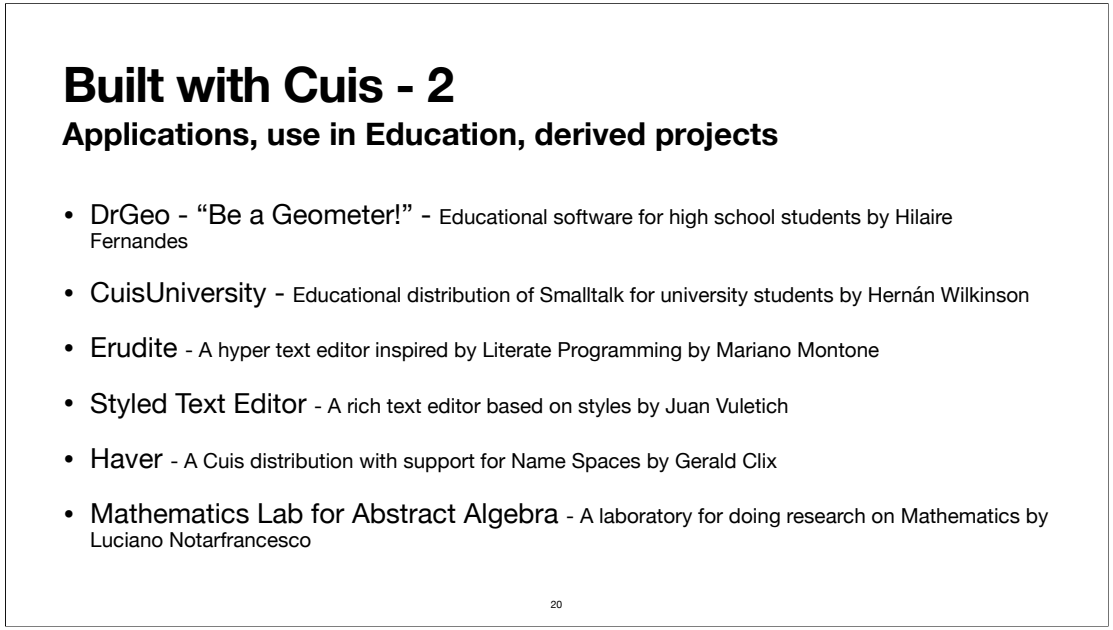
We needed a real-time image rectification and geolocation pipeline that could run on low power hardware, even in orbit on the satellites themselves. Existing applications and libraries would not be useful. We questioned the usual assumptions for high performance computing: Should this be done in C? Should it be done in Assembly Language?

The truth is that the real problem here is the complex domain and the need for new algorithms and data structures. So, I chose to do it in Cuis Smalltalk, a tool that would let me focus on the real problems. I modeled the whole problem domain, including simulations of satellite dynamics, telescopes, sensors, and a 3D surface model for the entire planet. The same model was used to simulate images (for validation), and to process raw satellite images (to build our products). This work took 2 years. Once it was working, for the small part that was performance critical, I recoded it in OpenCL. This was just about a hundred lines of code, and no debugging was needed, as it was a direct translation of the already working Smalltalk code. This part took me just one week. The code was put in production and we never found a bug in it. I was awarded several patents in the U.S. and elsewhere for this work.

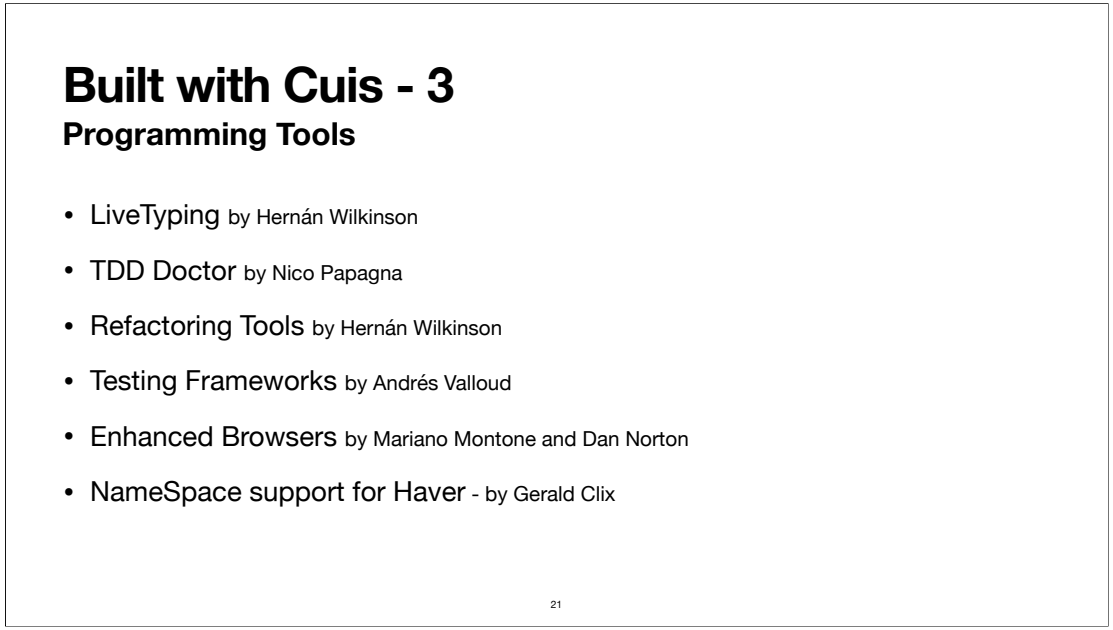
In the slide you can see some of the things I did for this project.

If you would like to know more about it, just search YouTube for “How Smalltalk enables world record breaking performance”, for the

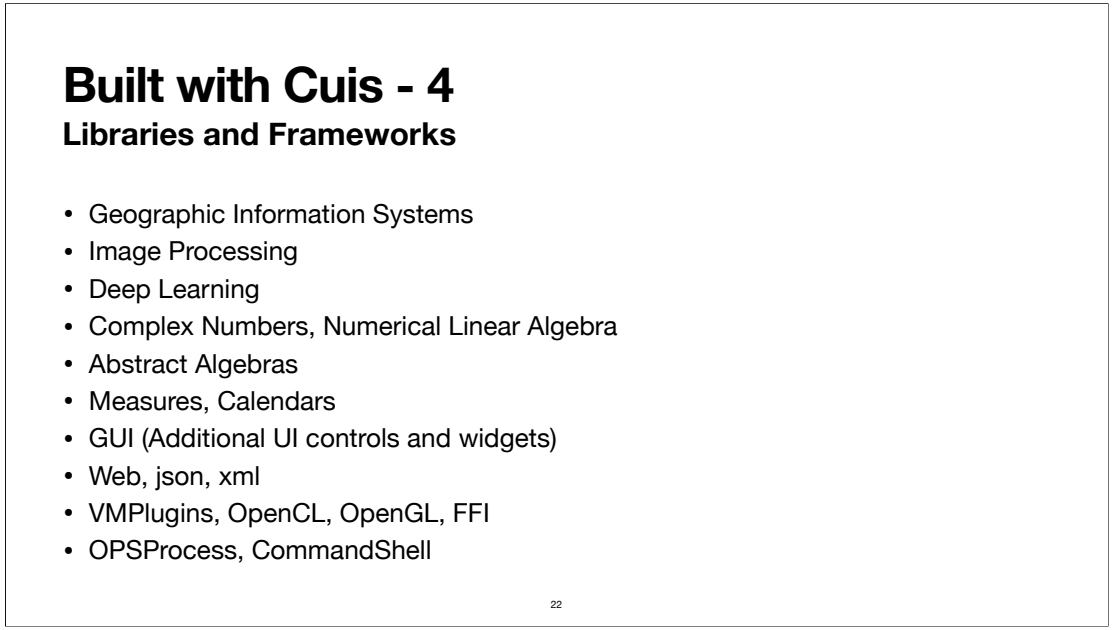
video of a presentation I did at the Argentinian Smalltalks conference in Tucumán in 2016.



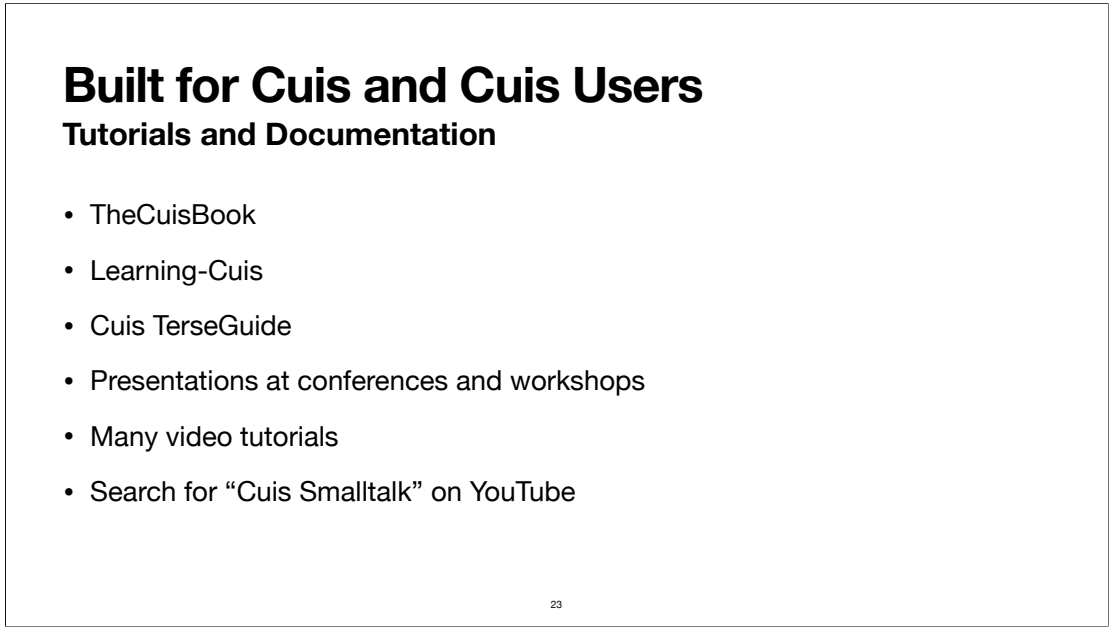
These are some major projects developed using Cuis.
Of particular importance to me is DrGeo, mainly for two reasons. One is that it is the first End User Application written in Cuis that will reach a large number of users. The other reason is that it is the first application that takes full advantage of Vector Graphics and the more advanced features of the new Morphic. This means that it helped reveal many bugs and problems, and this made the new Morphic more robust and performant.
CuisUniversity is a Cuis distribution for teaching at universities. It includes Refactoring Tools, LiveTyping and a Pure Objects environment called “DenotativeObjects”. It is used by many students at several universities every semester.
Erudite is a Dynabook and Literate Programming inspired hyper text editor, integrating running code as part of the documents
StyledTextEditor is an end user application for rich text using text styles
Haver is a Cuis distribution with support for namespaces
Luciano’s-Mathematics is a Math laboratory aimed at Math researchers



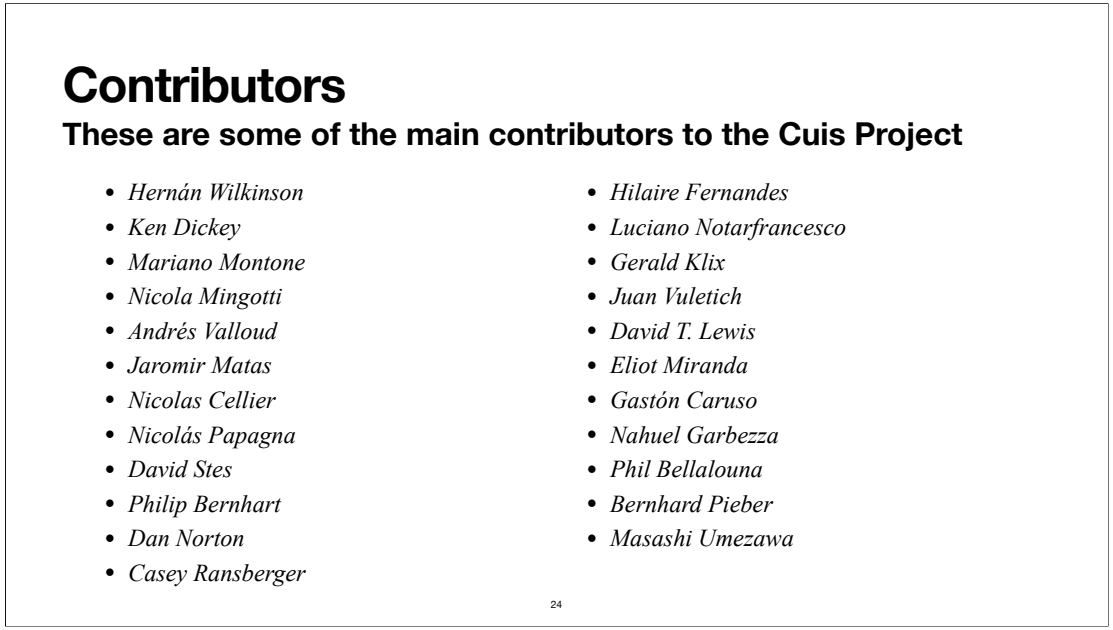
Several people from the Cuis community wrote new tools to aid programmers in their work.
To me, the most important one is LiveTyping. Programming languages are often classified in two categories: those who do static typing, requiring the programmer to declare the types of all variables, and those that do dynamic typing, validating at runtime that a message is appropriate for the receiving object.
LiveTyping offers a third alternative: In an environment like Smalltalk, that does dynamic typing, it captures the types for all variables as code is run. The programmer is not required to specify types, as it is the system who tells the programmer about them. This gives additional information to the programmer, making it easier to understand and modify an existing code base.
Other programming tools written for Cuis include a smart assistant for doing Test Driven Development, automatic Refactoring Tools, alternative Testing Frameworks and Enhanced Browsers.
Gerald Clix wrote support for NameSpaces for Cuis, but as the additional complexity meant we could not integrate this work, he started a Cuis fork named Haver.



As it is usual in Smalltalk, people working on a certain domain will not only write the applications they need, but they will also produce reusable libraries with all the code that it is not strictly application specific. We keep a large library of such packages, covering diverse areas. All this code is also MIT License, so it is free for any use.
These include code for standard technical needs, like usual Web standards and interacting with the Operating System and external libraries. They also include things that are useful in many domains, like Measures, Calendars and additional GUI Elements. Finally, there are high quality libraries for more specific areas, like Image Processing, Deep Learning, Geographic Information Systems and Linear Algebra.



The Cuis community has also produced a significant amount of documentation and learning material. Hilaire Fernandes, together with Ken Dickey and me wrote The Cuis Book, a complete introduction to the Smalltalk language, the Cuis environment and its tools, and the basic libraries.
The Learning-Cuis GitHub repo is the starting point for people new to Cuis or Smalltalk. It includes several tutorials, and links to other documentation.
The Cuis TerseGuide is a quick guide in the form of a Cuis package, to be read from within Cuis itself.
Additionally, there have been many presentations made at various conferences and Smalltalk online meetings, that have been uploaded to YouTube.
And several members of our community have produced video tutorials and demos, that you can also find on YouTube.



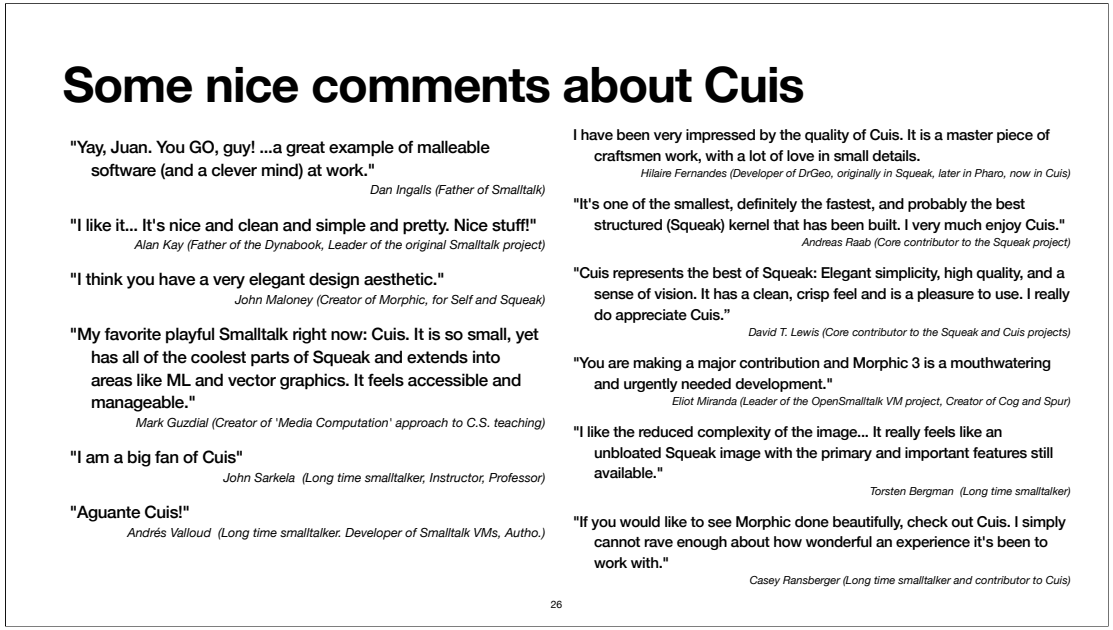
Cuis has an active community of about 50 developers. Many of them contribute with bug reports, fixes, enhancements, questions and answers in the mail list.
These are the names of people who have done significant code contributions to Cuis.
We usually have technical discussions to reach consensus on technical decisions. Also any issues that might affect external projects are raised, discussed and solved.
The community is friendly, and the traffic is reasonably low.
You are welcome to subscribe and say ‘Hi!’

Conclusions

Some final thoughts

- Cuis has been evolving over a long time, always true to its stated goals
- An active and friendly community of developers has developed
- It is a great place to collaborate and learn
- People are choosing it for new projects
- Cuis will be well maintained for a long time
- It will keep evolving to better suit its users

We believe that the Cuis project does a real service to the Smalltalk community. It is unique in that it follows the Smalltalk-80 tradition and ideas, without being frozen in the past. The broad adoption and high esteem it has gained means it will be around for a long time. All this means it is worth learning more about it, and perhaps be part of our community.



Many of the people who inspire our work, are aware of Cuis and have praised it. At the left you see their names, and the nice words they had for Cuis. Besides making me deeply proud, they are important because they mean we understood what they tried to teach us. They also mean that Cuis is a worthy member of their lineage.
Cuis is also appreciated by the people who decide to use it for their projects, and to be active members of our developers community. On the right column, you see some of their names and what they think about Cuis.

I think we have some time for questions now. Also, if you there is something specific you'd like me to show tomorrow, please tell!