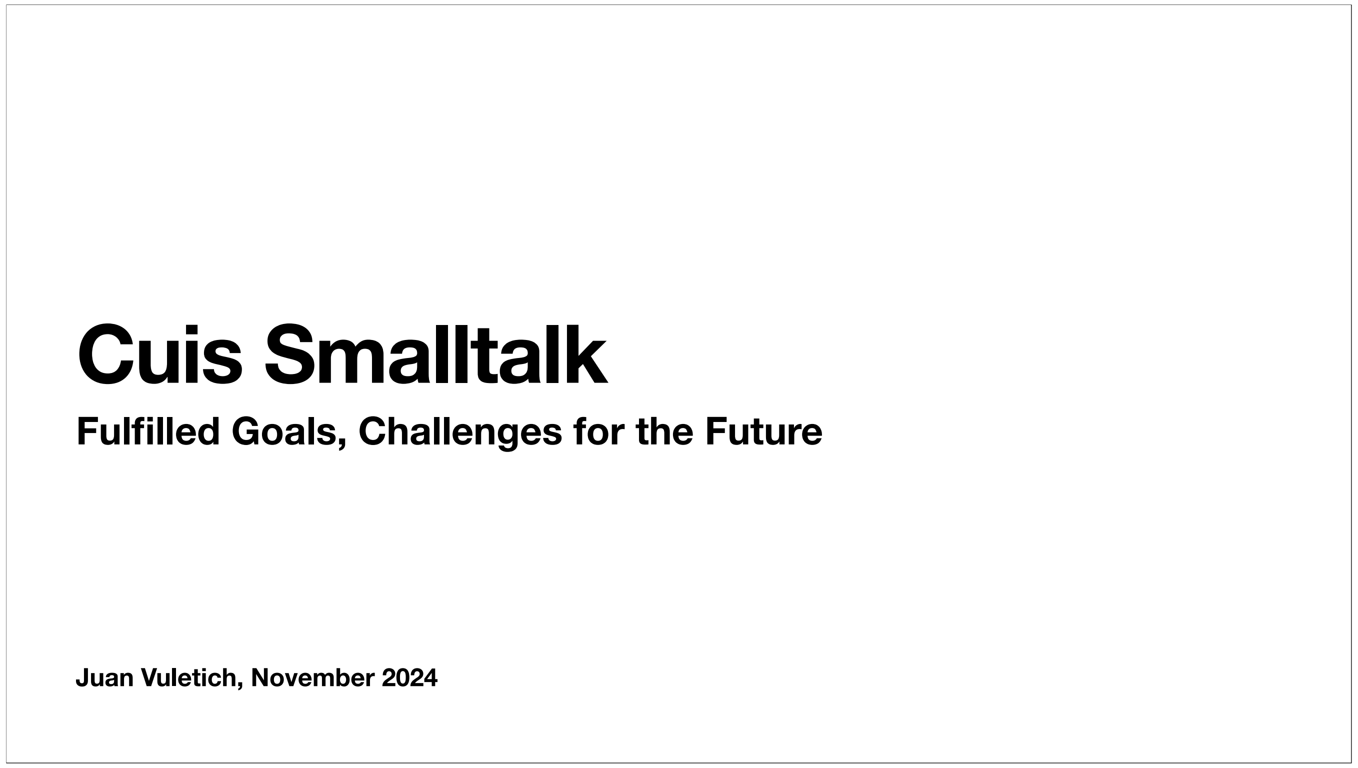
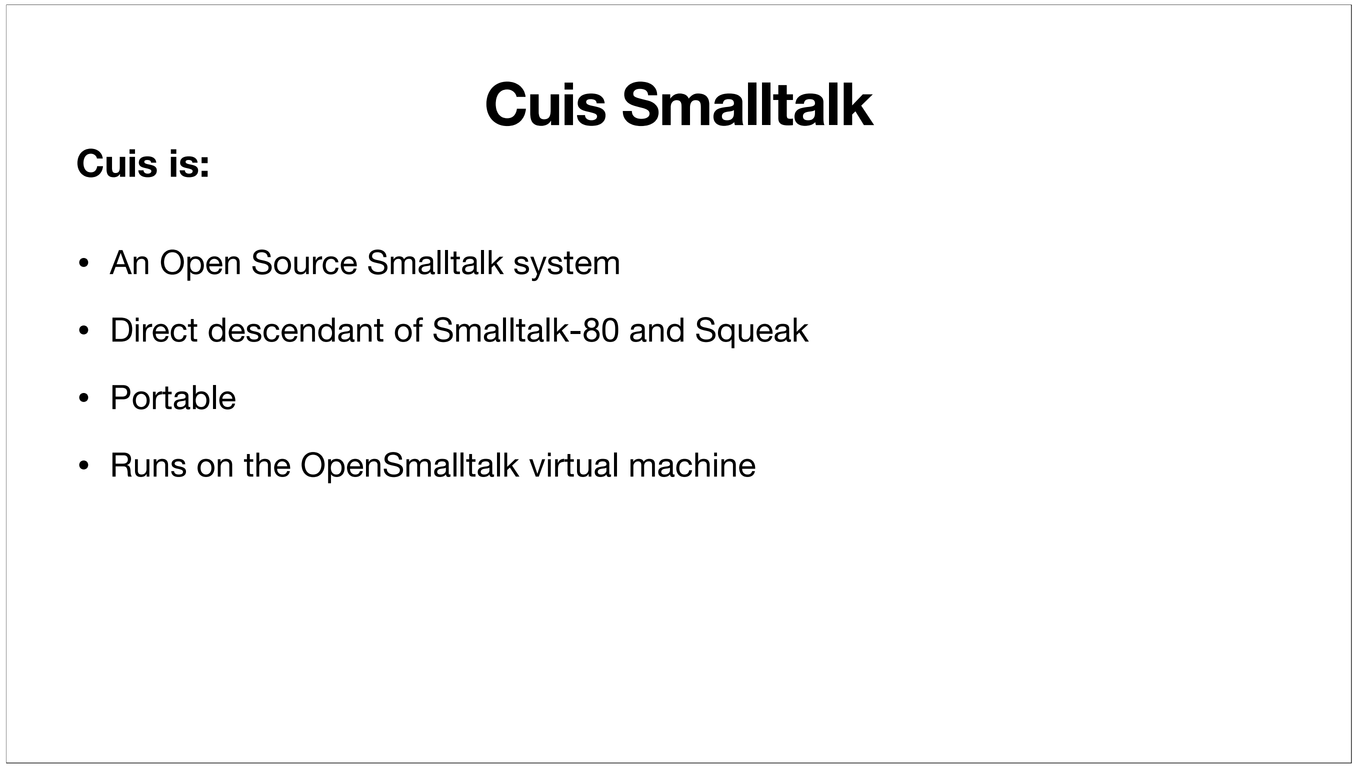


Cuis Smalltalk

Fulfilled Goals, Challenges for the Future

Juan Vuletich, November 2024

Hello. I'm Juan Vuletich and I'll talk, again, about Cuis Smalltalk.
I want to start by thanking you for being here today. I want to thank FAST, Universidad Nacional de Mar del Plata and the Public Universities in Argentina that host us every year for this conference.
The Argentinian Public University System is under attack from the Argentinian Government, and it is important to raise our voice on every opportunity to defend it. High quality universities, open and free for all, are one of the pillars for making Argentina a world class creator of Science and Technology. It is a duty of the Argentinian government to continue providing the funds to keep them strong.
I want to thank you Hernán and other University teachers, for using Cuis in class and spreading the word. Thank you Máximo for starting this movement that today means that Argentina is world class Smalltalk hub.
Thanks Hilaire for DrGeo and for The Cuis Book. Thanks to the Cuis community for many contributions, useful feedback and lots of fun, on the mail list and on our monthly meetings. Thank you LabWare for sponsoring this project.

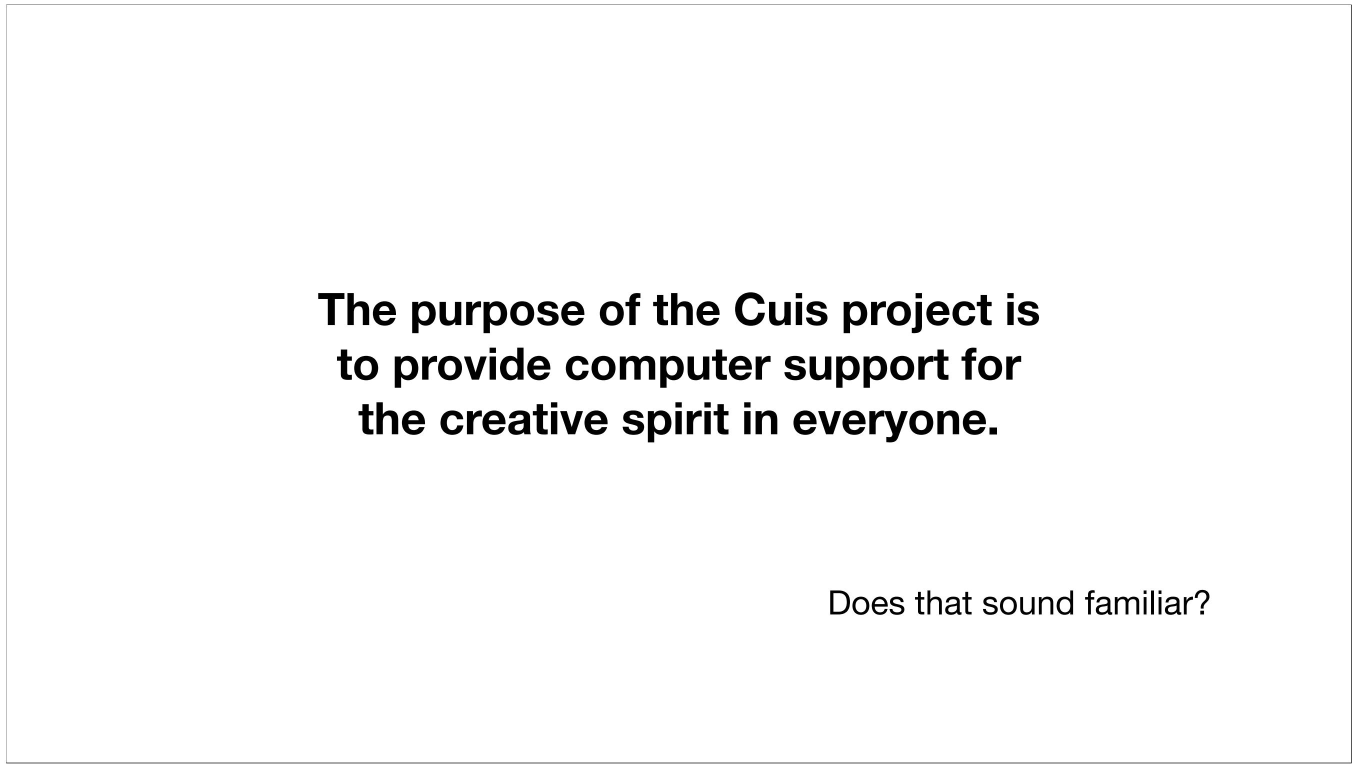


Cuis Smalltalk

Cuis is:

- An Open Source Smalltalk system
- Direct descendant of Smalltalk-80 and Squeak
- Portable
- Runs on the OpenSmalltalk virtual machine

Read the slide.
Most likely you all already know this, right? Nothing surprising here. So, what else can we say about Cuis?



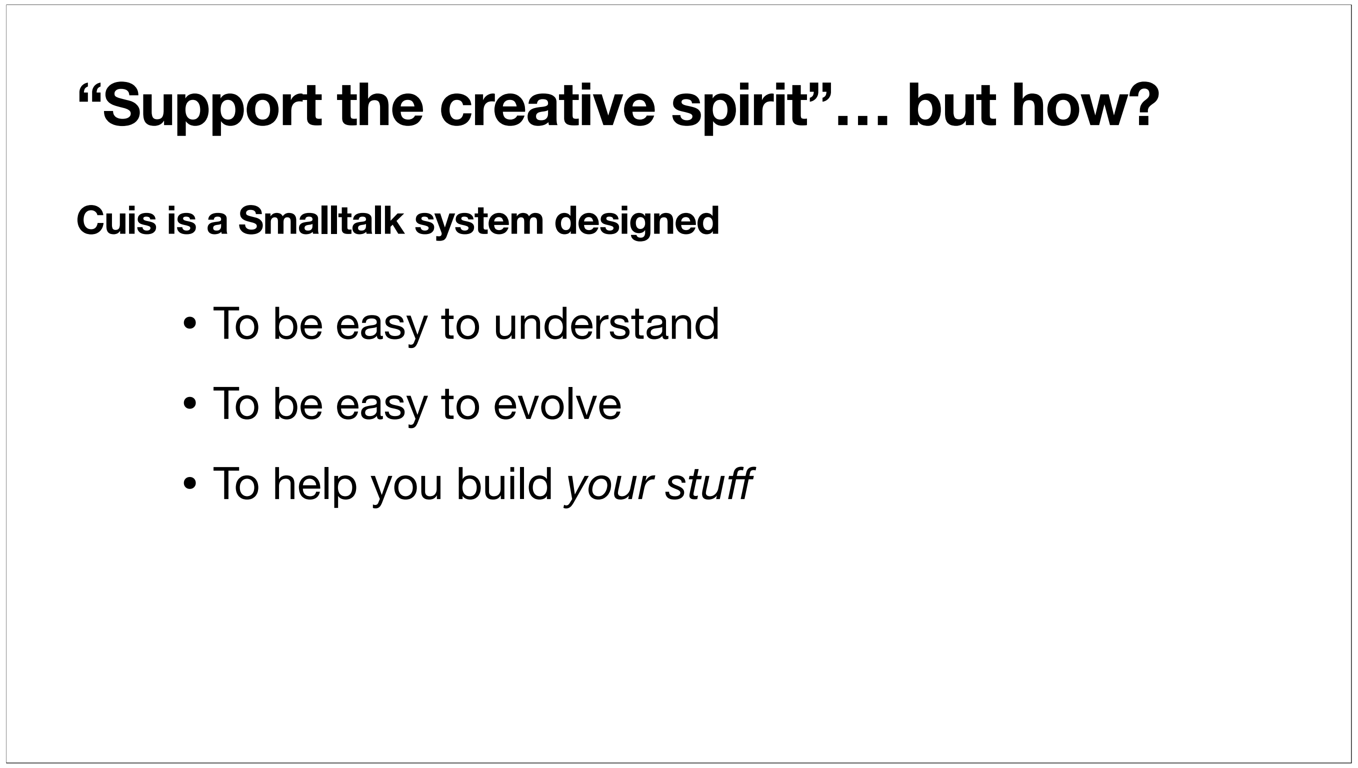
The purpose of the Cuis project is to provide computer support for the creative spirit in everyone.

Does that sound familiar?

The purpose of the Cuis project is to provide computer support for the creative spirit in everyone.

Does that sound familiar?

Yes, those are (almost) the opening words of “Design Principles Behind Smalltalk”, by Dan Ingalls. The goals and attitudes of the Smalltalk project at PARC still inspire our work everyday.



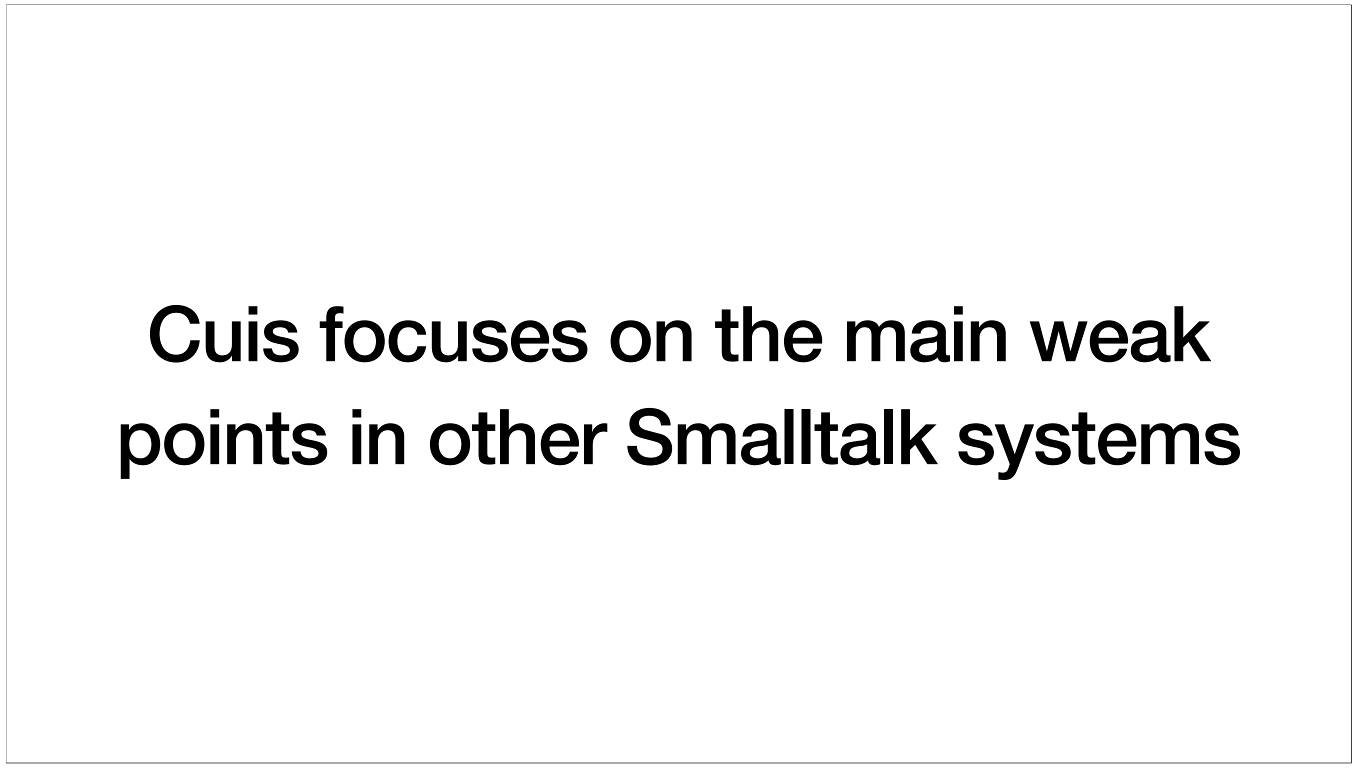
“Support the creative spirit”… but how?

Cuis is a Smalltalk system designed

- To be easy to understand
- To be easy to evolve
- To help you build *your stuff*

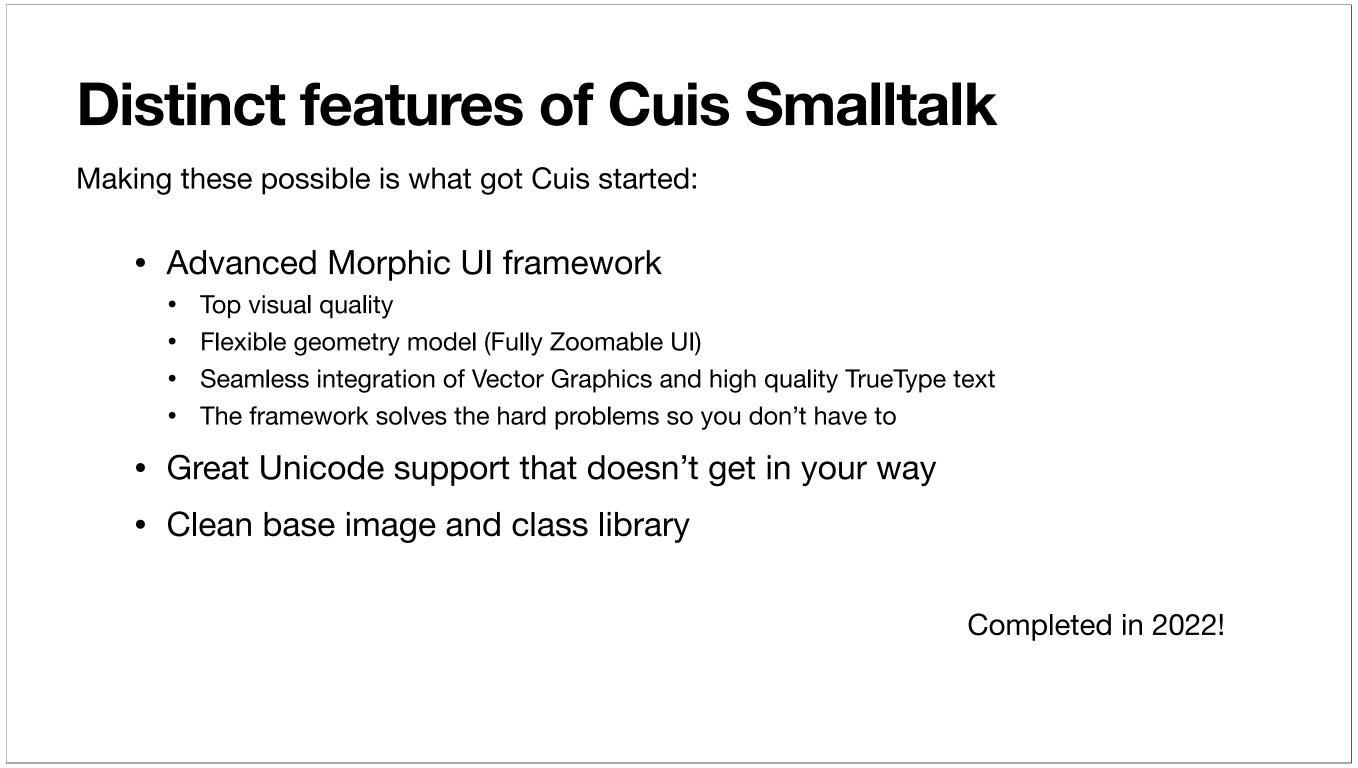
Computers are insanely more powerful than they were when Smalltalk-80 was originally built. They are way more powerful than ever envisioned for the Dynabook vision. And computer platforms and systems are always getting bigger and more complex. The downside of all this is giving up our ability to completely understand what’s going on.

Sometimes it takes a lot of work to make something simpler. That’s our quest.



Cuis focuses on the main weak points in other Smalltalk systems

Every Smalltalk system, both commercial and open source, has evolved by following the priorities set by its users and developers. In this landscape, Cuis has a distinct set of priorities. And we have been following them ever since the project got started.



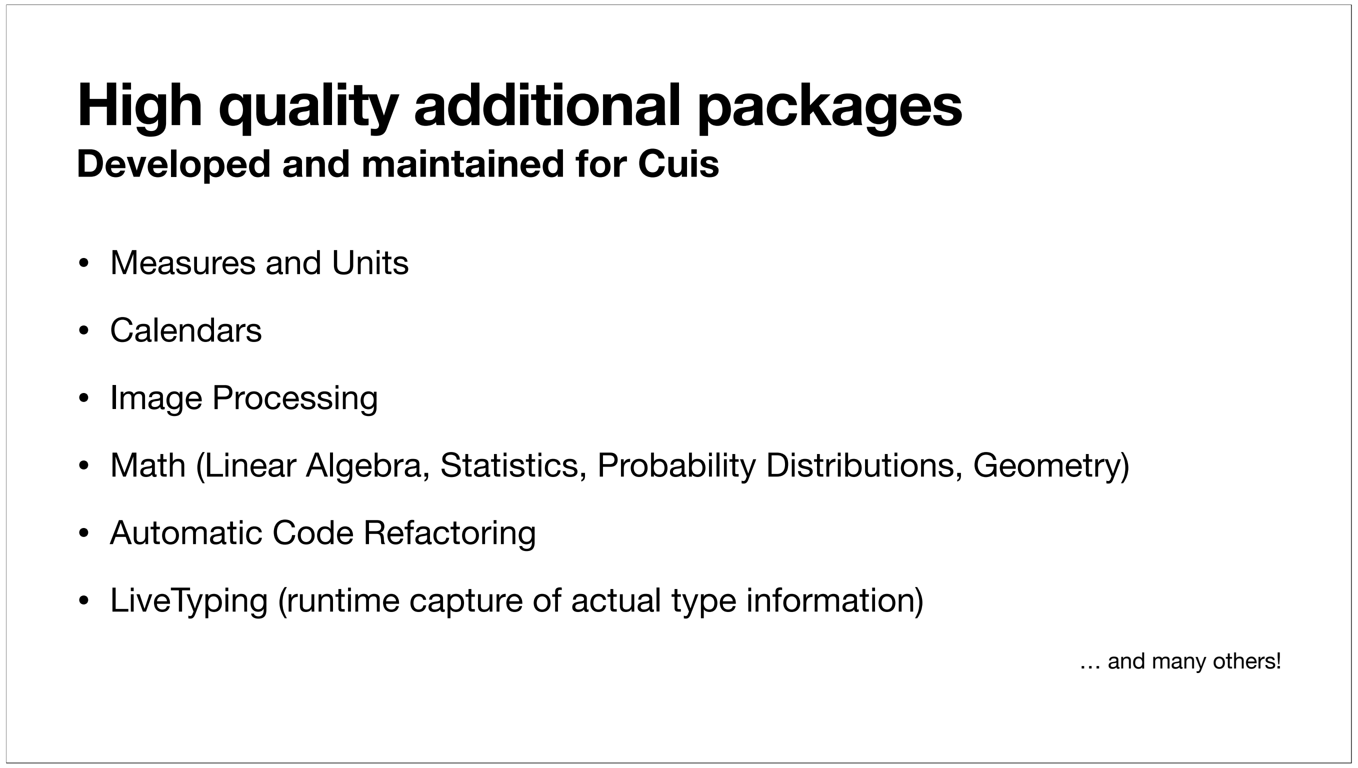
Distinct features of Cuis Smalltalk

Making these possible is what got Cuis started:

- Advanced Morphic UI framework
  - Top visual quality
  - Flexible geometry model (Fully Zoomable UI)
  - Seamless integration of Vector Graphics and high quality TrueType text
  - The framework solves the hard problems so you don’t have to
- Great Unicode support that doesn’t get in your way
- Clean base image and class library

Completed in 2022!

A long time ago, while using Squeak for several projects, I realized that there were several areas where I felt Squeak was in need of serious improvement. The Morphic framework made advanced UIs possible, but difficult to code. Additionally, the transition in display technology from cathode ray tubes to flat panel made pixellation much more visible, creating the need for high quality anti-aliasing. Also, the approach that was taken to support Unicode put too much burden on developers. Finally, the class library had evolved without careful design, and it became bloated and confusing.
Fixing these issues was the driving force behind Cuis. And all this was essentially completed by the end of 2022.



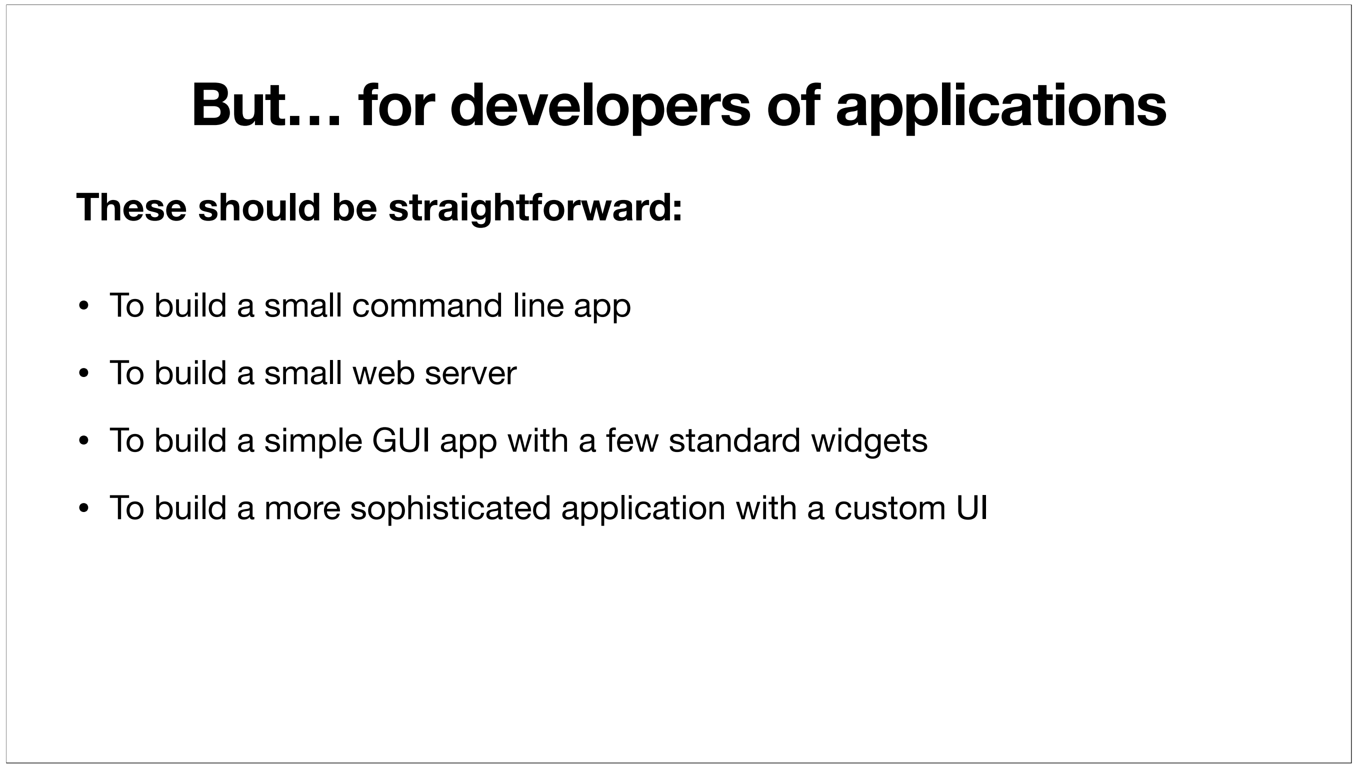
High quality additional packages

Developed and maintained for Cuis

- Measures and Units
- Calendars
- Image Processing
- Math (Linear Algebra, Statistics, Probability Distributions, Geometry)
- Automatic Code Refactoring
- LiveTyping (runtime capture of actual type information)

… and many others!

During this time, several projects have been based on Cuis. As a result, many interesting and high quality code packages have been written for, or ported to Cuis by its user community. The list above shows a small fraction of them.

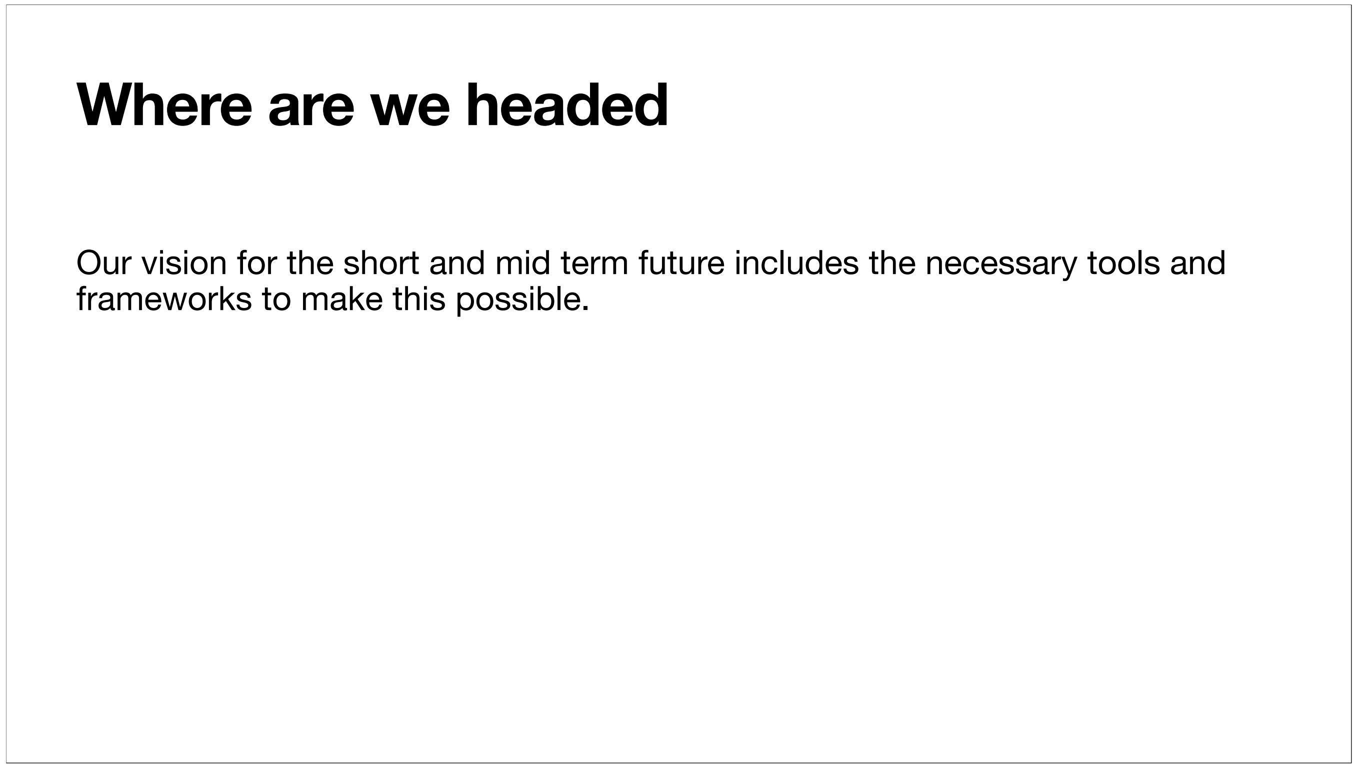


But… for developers of applications

These should be straightforward:

- To build a small command line app
- To build a small web server
- To build a simple GUI app with a few standard widgets
- To build a more sophisticated application with a custom UI

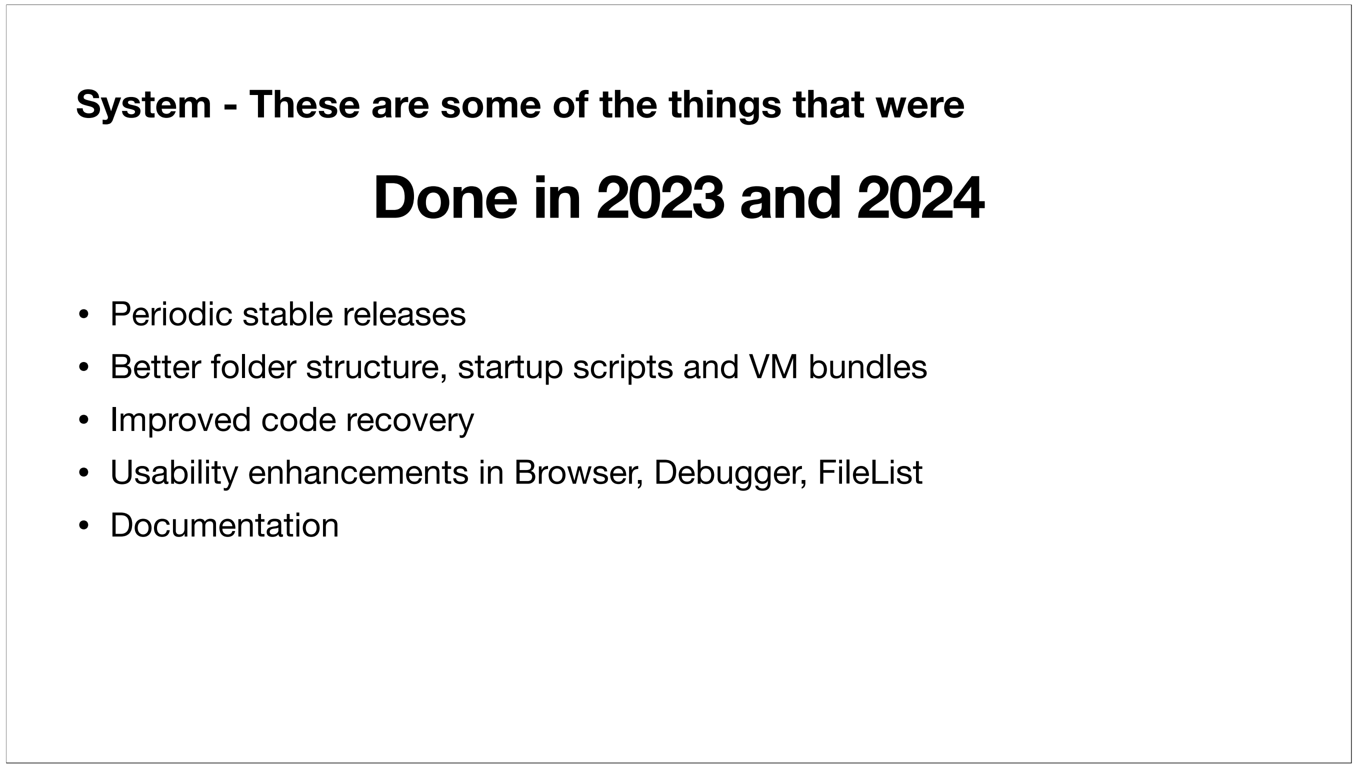
But if you come to Cuis from a non-Smalltalk background, or you are used to one of the commercial Smalltalk systems, you would expect some things to be really easy.
Building conventional user interfaces based on text fields, lists and buttons must be easy, as in most other development environments.
The same goes for building distributable applications for various platforms.
But today you need to build the Cuis image for your application yourself. This is an involved process, and still you don't get full control of what gets included in it.
And then you have to package it with a suitable VM and other files in order to distribute it to your users or deploy it to a web server infrastructure.



Where are we headed

Our vision for the short and mid term future includes the necessary tools and frameworks to make this possible.

At this point in the Cuis project, we are focusing on this different set of problems. We have already started working on the tools and frameworks needed for building applications. Let me tell you about that.

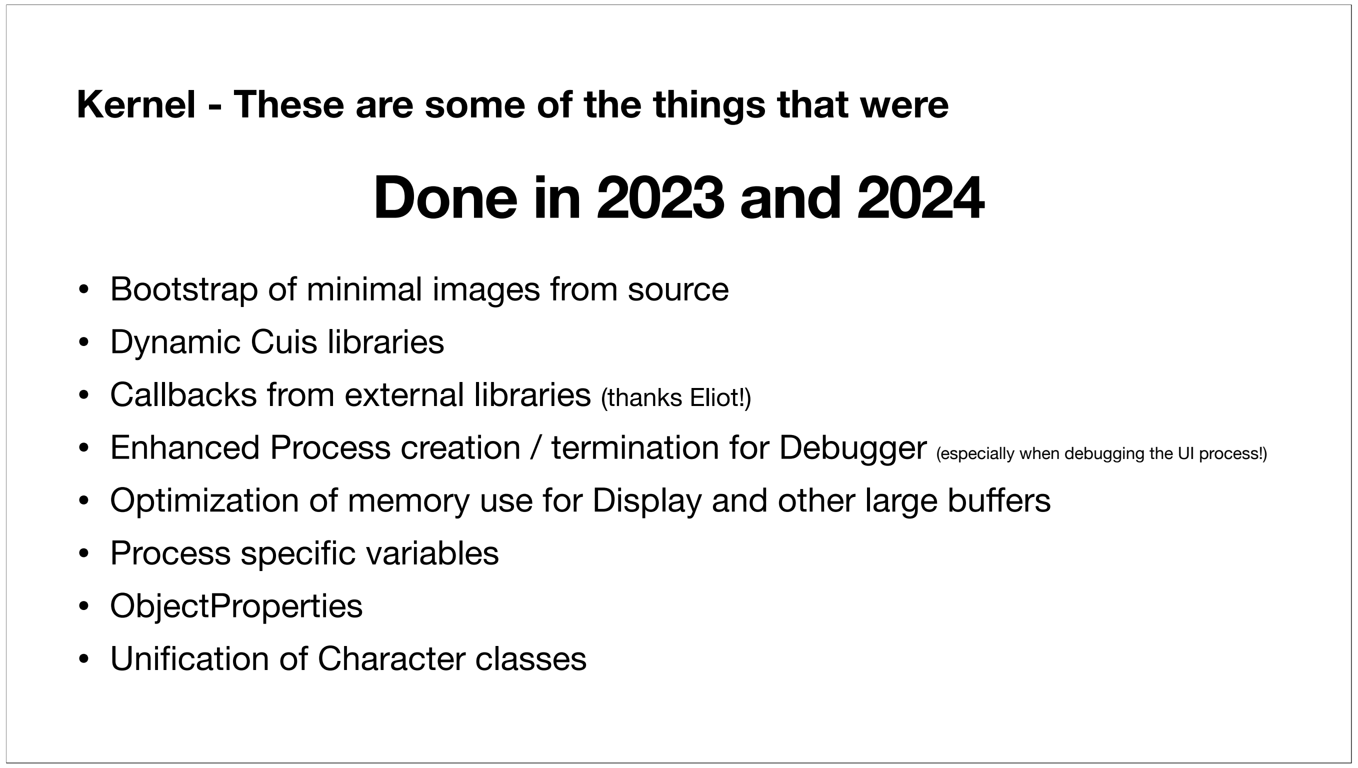


System - These are some of the things that were

Done in 2023 and 2024

- Periodic stable releases
- Better folder structure, startup scripts and VM bundles
- Improved code recovery
- Usability enhancements in Browser, Debugger, FileList
- Documentation

At a general System level, in this last couple of years we have been improving the Cuis release process. Now we do a stable, long term support releases of the system every six months, following a release process inspired by RedHat Linux.
Cuis is now distributed with a better folder structure. It separates system folders and files from developer folders and files, making it more comfortable to work on the files that hold your work. It also includes pre made VMs and startup scripts. These make for a much easier setup of Cuis on your machine, and it also gives you a useful model for the file and folder structure for your own applications.
We have also improved the code recovery tools used in case of a crash, making it convenient to review and reload your code.
Based on feedback by the user community, we have been improving usability of the various Smalltalk tools.
We are also working on the system documentation all the time, trying to make it good both for experienced Smalltalkers and for people new to Smalltalk.

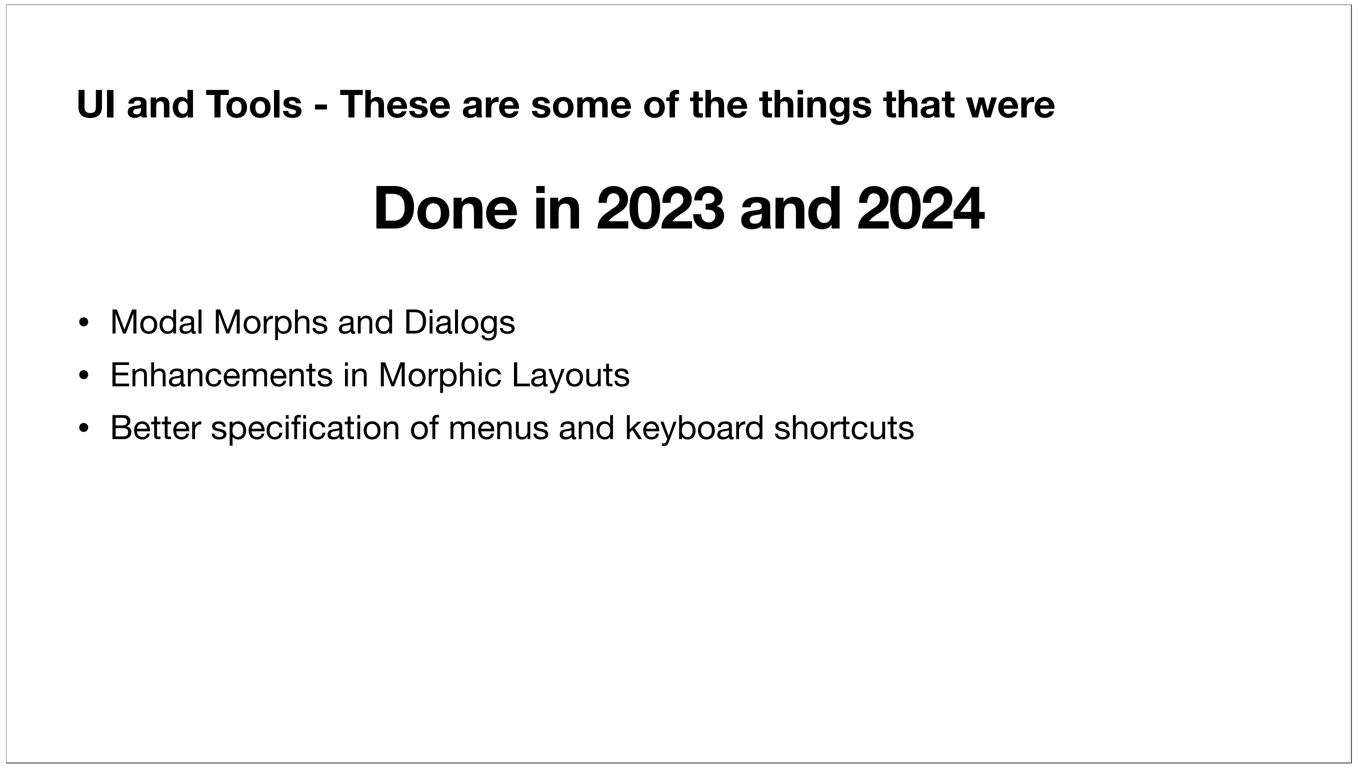


Kernel - These are some of the things that were

Done in 2023 and 2024

- Bootstrap of minimal images from source
- Dynamic Cuis libraries
- Callbacks from external libraries (thanks Eliot!)
- Enhanced Process creation / termination for Debugger (especially when debugging the UI process!)
- Optimization of memory use for Display and other large buffers
- Process specific variables
- ObjectProperties
- Unification of Character classes

The Smalltalk kernel has also been enhanced with new features.
Bootstrap and pre-compiled libraries are essential for building distributable applications.
Callbacks from libraries called via FFI are absolutely needed in many cases. Thanks Eliot Miranda for helping us with this.
When a debugger opens, the debugged process gets suspended and from that moment it is under control of the debugger. When the user finishes using the debugger and clicks [Proceed], this suspended process gets resumed. But when the debugger opens on the UI process, we need to schedule a new UI process in order to have a working UI. This new process is terminated when the user [Proceed]s, but it is kept if the user abandons the debugger. Cuis has an additional problem here, as the Morphic 3 framework uses additional data structures that would get corrupted if naively shared between these processes. It even gets worse. These data structures changes in size if the user resizes the Display while the debugger is open, and when they [Proceed] the old ones held by the suspended process are no longer usable. It took some work to get all this working smoothly, without wasting memory by simple reallocating them all the time.
We also added support for Process specific variables (taken from Squeak), and an optional property table on any object (also from Squeak).
And by taking advantage of OpenSmalltalk Spur’s wide immediate Character support, we unified the previous two separated Character classes, simplifying code and improving performance.

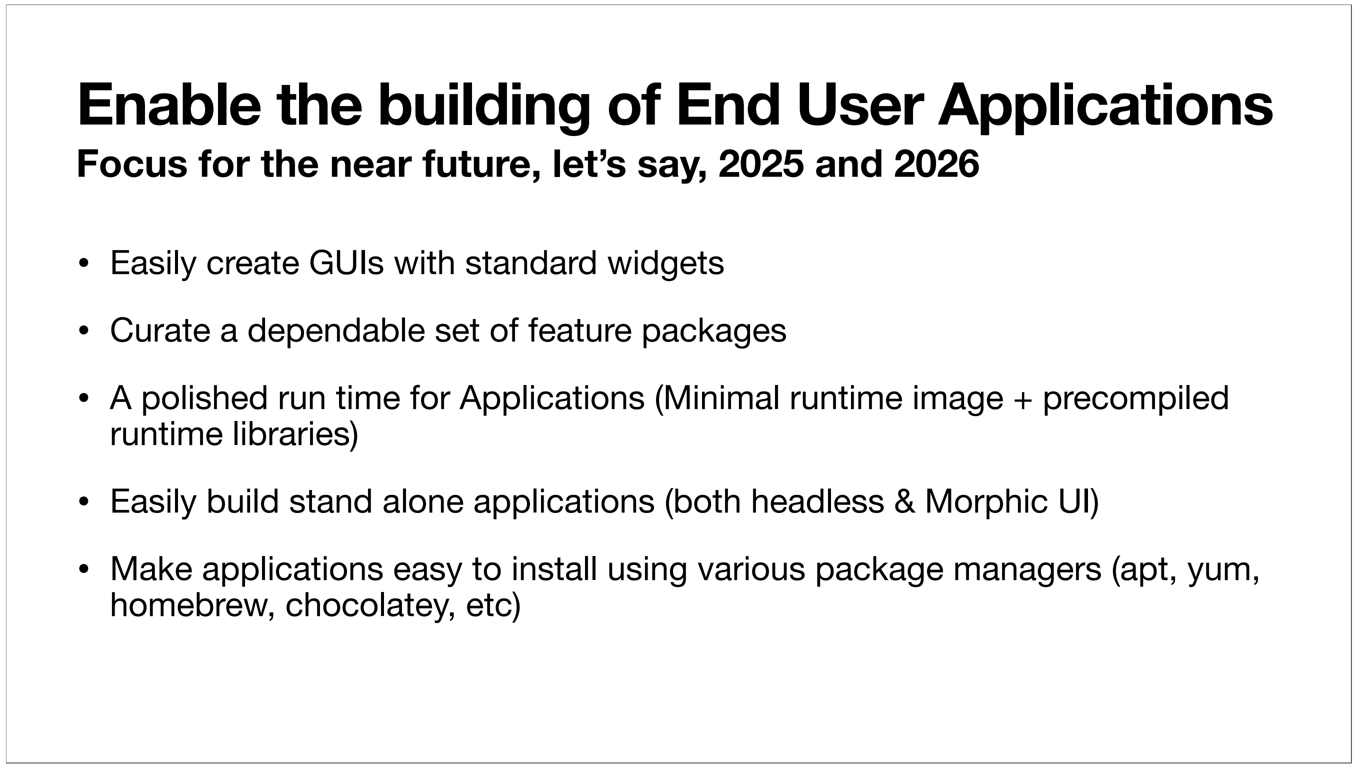


UI and Tools - These are some of the things that were

Done in 2023 and 2024

- Modal Morphs and Dialogs
- Enhancements in Morphic Layouts
- Better specification of menus and keyboard shortcuts

There have been several enhancements to the Morphic UI framework, focused on supporting End User Applications.
Modal Morphs and Dialogs. What do we mean by a Modal Dialog? Assume that some regular code that is running in the UI process needs some information from the user in order to continue execution. It may be something so simple as prompting the user for their AuthorInitials, or asking them to enter a class name to find. At that moment, the UI process can’t continue. But we still need to keep the UI alive. Previously we used the same technique as Squeak, that is running a separated UI Morphic cycle inside the modal morph while we wait for the answer. This is not good. It duplicates code, it breaks encapsulation and it is fragile. But the work done on Debugger in the previous slide gives a better solution: use the Morphic process scheduling machinery in the same way as the debugger does. This means simpler and more robust code, and also a really alive UI, including stepping morphs like the clock, for instance, when using a menu.
There have been improvements in Morphic Layouts, adding padding (an additional colorless border to inset widgets) and gap (separation between widgets). These were modeled after CSS, so they will be familiar to many.
Finally, we have included a completely new way to specify menus and keyboard shortcuts. It is now easier for programmers, but also more flexible.



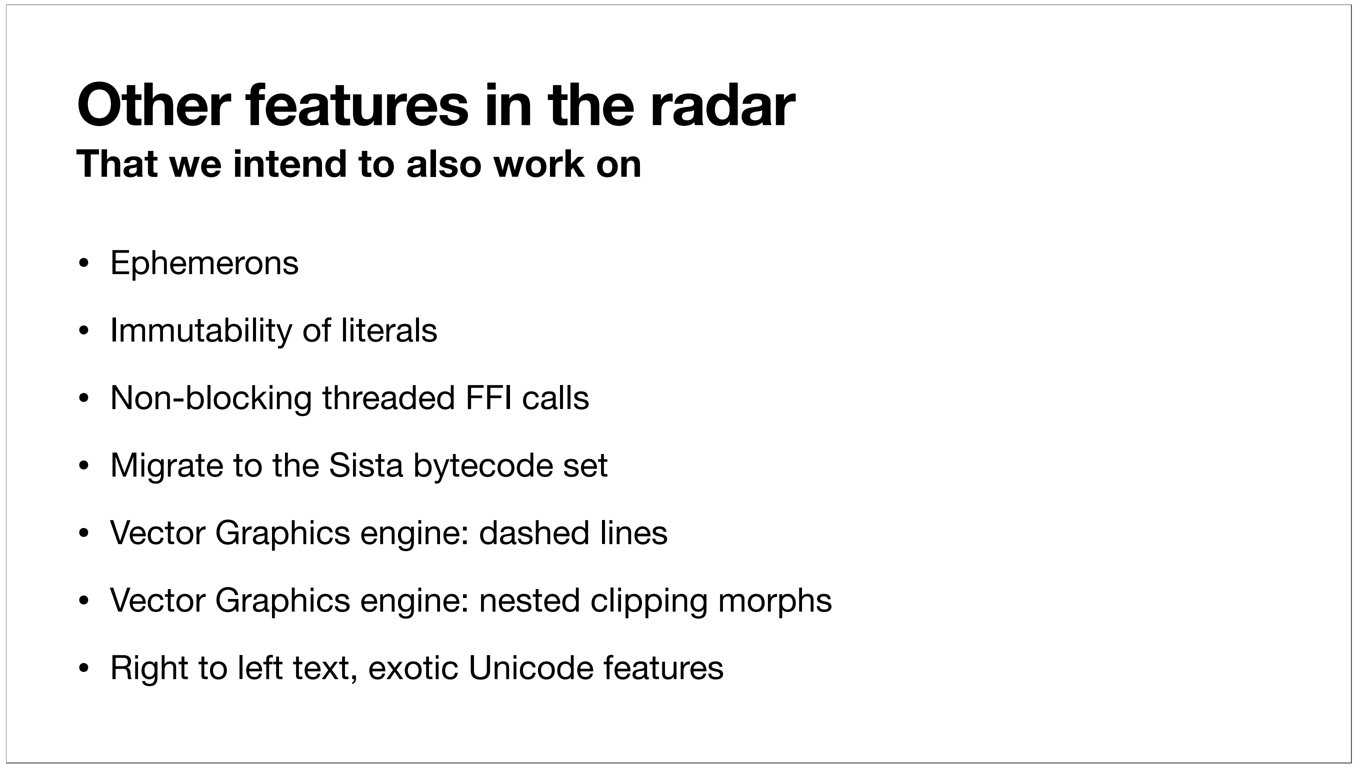
Enable the building of End User Applications

Focus for the near future, let’s say, 2025 and 2026

- Easily create GUIs with standard widgets
- Curate a dependable set of feature packages
- A polished run time for Applications (Minimal runtime image + precompiled runtime libraries)
- Easily build stand alone applications (both headless & Morphic UI)
- Make applications easy to install using various package managers (apt, yum, homebrew, chocolatey, etc)

Going back to our plans for the future, and the needs of developers building actual end user applications, these are some of the missing parts.

- We will be working on an interactive tool for building UIs by composing widgets
- We will do a quality assessment on existing packages, so it is easier for developers to know how reliable they are
- We will build a small and nice run time for applications, based on a minimal runtime image and the dynamic loading of precompiled libraries
- We will also build a tool to generate the folder and file structure for stand alone applications, both commandline or server and interactive Morphic
- And we will support various package managers to distribute applications, such as apt (for Debian and Ubuntu), yum (for RedHat), homebrew (for the Mac) and Chocolatey (for Windows)

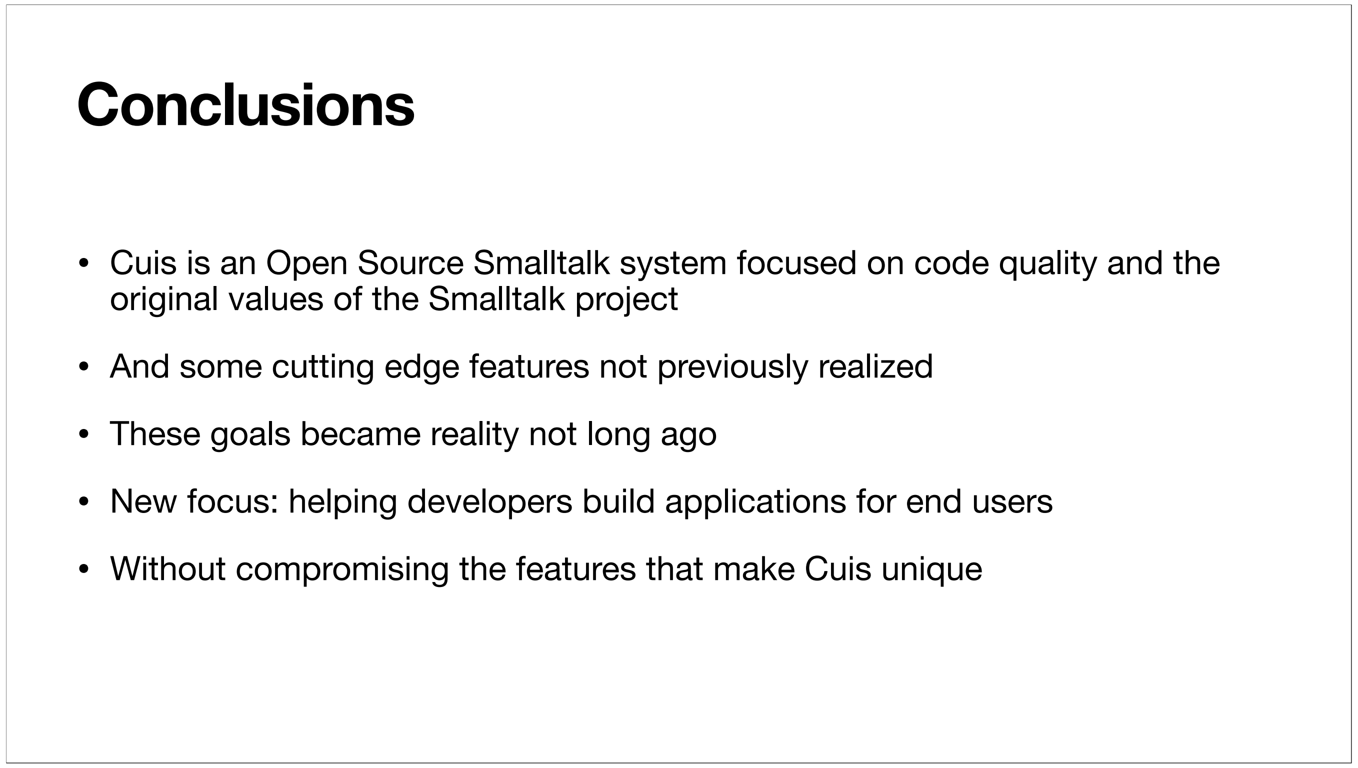


Other features in the radar

That we intend to also work on

- Ephemerons
- Immutability of literals
- Non-blocking threaded FFI calls
- Migrate to the Sista bytecode set
- Vector Graphics engine: dashed lines
- Vector Graphics engine: nested clipping morphs
- Right to left text, exotic Unicode features

There are other Kernel features we intend to work on. We will add support for Ephemerons, as they are already supported by the OpenSmalltalk VM. We will take advantage of Immutability, so literal objects created at compile time will be immutable.
We will support non-blocking threaded FFI calls and migrate to the Sista bytecodes. Both these will be done in close collaboration with Eliot Miranda. Thanks Eliot!
We have some already planned enhancements to the Vector Graphics engine, including dashed lines and the possibility of having multiple nested morphs that clip their submorphs.
We also want to add right to left text (for Hebrew and Arabic) and some other more exotic Unicode features.



Conclusions

- Cuis is an Open Source Smalltalk system focused on code quality and the original values of the Smalltalk project
- And some cutting edge features not previously realized
- These goals became reality not long ago
- New focus: helping developers build applications for end users
- Without compromising the features that make Cuis unique

Let me wrap up this presentation by quickly repeating the main points.
- Cuis is an Open Source Smalltalk system focused on code quality and the original values of the Smalltalk project
- It also includes some advanced features not previously realized on any development environment
- These original goals of the project became reality not long ago
- Now the focus is on helping developers be more effective when building applications for their end users
- Without compromising the features that make Cuis unique

That's all. Thank you.
Perhaps we have some minutes for questions. In any case, besides me more people using Cuis on a regular basis are here in the conference, including Eliot Miranda, Jon Raiford, Felipe Zak and Hernán Wilkinson. Feel free to ask us anything during the breaks.