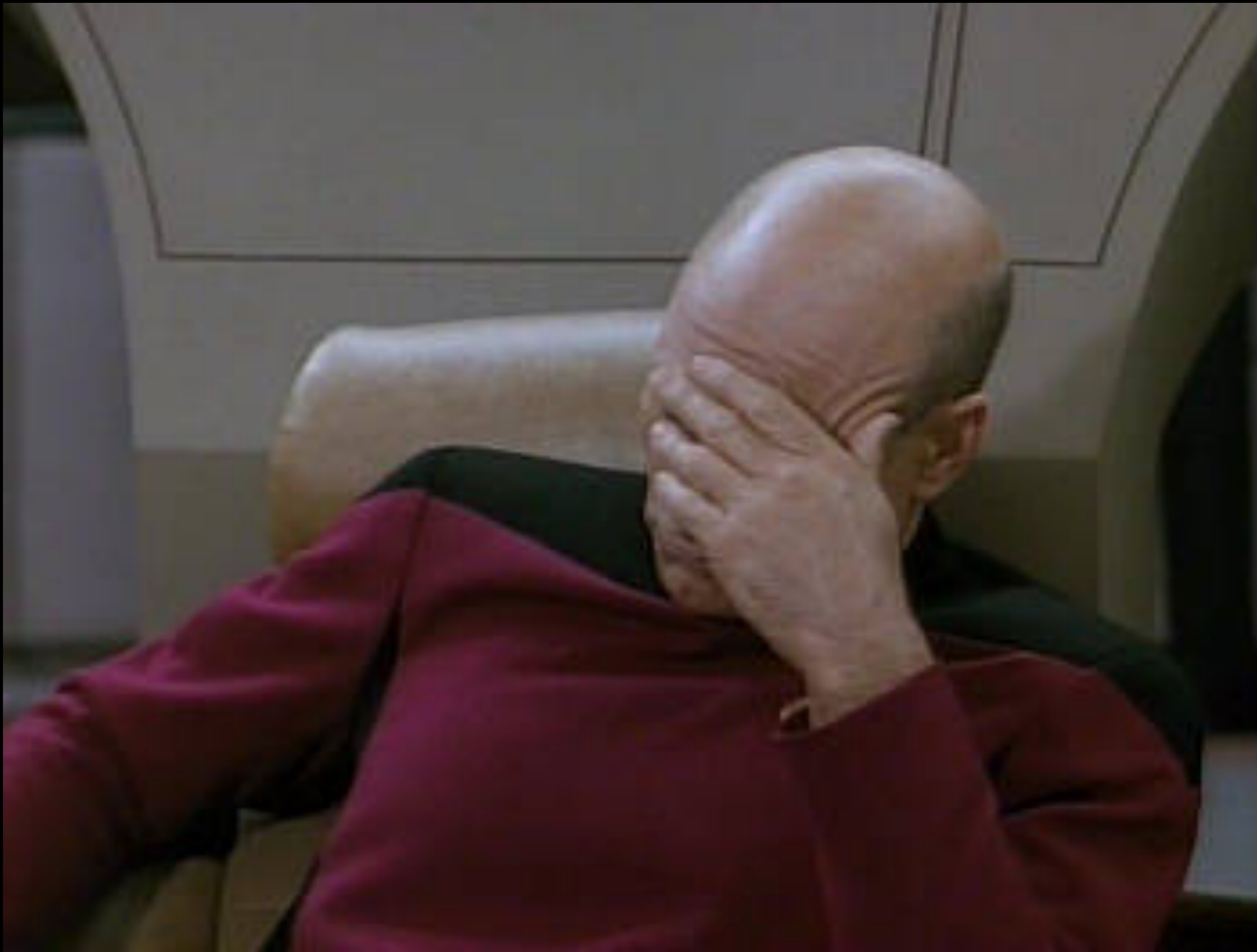


The Sobyk Binary Distribution





“Not this again, Hypnotoad”

-Everyone in this room



**You broke my 10 year
old production code
in a point release.**



**Namespace deletion
on 8.6 was
fundamentally flawed
and needed to be
refactored**

```

143 # the <Destroy> we are seeing is intended for us.
144 ###
145 method Hull_Destroy {} {
146     ###
147     # Destroy our Tk representation
148     ###
149     my variable tkalias
150     if {[info exists tkalias]} {
151         set alias $tkalias
152     } else {
153         set alias {}
154     }
155     if {$alias ne {}} {
156         my Hull_Unbind $alias
157     }
158     catch {my action destroy}
159
160     # Destroy an alias we may have created
161     if { $alias ne {} && [winfo exists $alias] } {
162         catch {rename [namespace current]::tkwidget {}}
163     } else {
164         set hull [my organ hull]
165         if { $hull ne "." } {
166             catch {::destroy $hull}
167         }
168     }
169 }
170
171 ###
172 # Clean up children
173 ###
174 foreach subobj [info command [self]/*] {
175     catch {$subobj destroy}
176 }
177 foreach subobj [info command [self].*] {
178     if {[winfo exists $subobj]} continue
179     catch {$subobj destroy}
180 }
181
182 }

```

```

143 # the <Destroy> we are seeing is intended for us.
144 ###
145 method Hull_Destroy {} {
146     ###
147     # Destroy our Tk representation
148     ###
149     my variable tkalias
150     set tkWidget {}
151     if {[info exists tkalias]} {
152         set tkWidget $tkalias
153     }
154     if {$tkWidget eq {}} {
155         set tkWidget [my widget hull]
156     }
157     if {$tkWidget eq {}} {
158         set tkWidget [my organ hull]
159     }
160     if {$tkWidget ne {}} {
161         my Hull_Unbind $tkWidget
162     }
163 }
164
165 ###
166 # Clean up children
167 ###
168 foreach subobj [info command [self]/*] {
169     catch {$subobj destroy}
170 }
171 foreach subobj [info command [self].*] {
172     if {[winfo exists $subobj]} continue
173     catch {$subobj destroy}
174 }
175
176 catch {my action destroy}
177
178 # Destroy an alias we may have created
179 if { $tkWidget ne { } && [winfo exists $tkWidget] } {
180     ::destroy $tkWidget
181 }
182 }

```



I had to back out 6 months of changes in fossil because you checked breakage straight into trunk



The MIME package was fundamentally flawed and needed to be refactored from first principles to be right.

I had to back out a little bit of history on Tcllib

core.tcl-lang.org

03:29 Last fixes. check-in: 269812cf8d user: aku tags: pooryorick

03:19 **Series of documentation fixes for problems caught by the experimental doctools parser written as large Marpa/Tcl example. - Removed many superfluous closing brackets. Some converted to proper [rb] markup. - Removed bad cross-references introduced by [c6b17331c9d12697] several years ago. - Removed several bad [para]graph markers placed before the first item of the enclosing list. Regenerated the embedded docs.** check-in: 3c9d4bcd85 user: aku tags: pooryorick

2018-08-27 **httpd: Renamed doc include files to follow the overall practice of naming them `.inc``.** Closed-Leaf check-in: 54eb35fc91 user: aku tags: httpd-docs-move, pooryorick

20:31 cron manpage: Fixed millisecond typo. `<D>` check-in: b14633aa2f user: aku tags: pooryorick

20:28 Fixes to cron manpage. Regenerated embedded docs. check-in: 61102fe7dc user: aku tags: pooryorick

2018-08-26 12:15 update comments check-in: 1d13943cd4 user: yorick tags: pooryorick

10:53 remove duplicated code in `::mime::buildmessageaux` check-in: 0587b55b96 user: yorick tags: pooryorick

2018-08-21 22:48 Numerous improvements and bug fixes. Updated analyzer script from file-5.34 magic files. check-in: c48961ab8e user: pooryorick tags: pooryorick

2018-08-15 05:23 code cleanup check-in: 6b167d6d17 user: pooryorick tags: pooryorick

04:34 Fix for [063e7a95beef451e], Correct Handling of quotes in quoted strings in `parseMimeValue` check-in: e4a17a66b1 user: pooryorick tags: pooryorick

2018-08-14 19:39 convert mime.test to utf-8 check-in: 6c21e9dfe4 user: pooryorick tags: pooryorick

19:16 Fix issue [a+6b1095974e071d], error in mime.tcl check-in: 074ec6a961 user: pooryorick tags: pooryorick

2018-07-22 11:38 Add a package for trigonometric functions that use angles in degrees and additional trigonometric and hyperbolic functions, including their inverses. check-in: 7538200f88 user: arjenmarkus tags: trunk

Just
a
bit...



What was all that?

- Each teal commit was originally to the trunk of Tcllib
- Tests passed perfectly fine in the modules the particular developer was working on
- They outright broke other modules
- Those changes polluted any merge with trunk made after that date
- By the end, developers like me couldn't even run some tests because modules started require Tcl 8.6.9

**And why couldn't I
test with 8.6.9?**

```

143 # the <Destroy> we are seeing is intended for us.
144 ###
145 method Hull_Destroy {} {
146     ###
147     # Destroy our Tk representation
148     ###
149     my variable tkalias
150     if {[info exists tkalias]} {
151         set alias $tkalias
152     } else {
153         set alias {}
154     }
155     if {$alias ne {}} {
156         my Hull_Unbind $alias
157     }
158     catch {my action destroy}
159
160     # Destroy an alias we may have created
161     if { $alias ne {} && [winfo exists $alias] } {
162         catch {rename [namespace current]::tkwidget {}}
163     } else {
164         set hull [my organ hull]
165         if { $hull ne "." } {
166             catch {::destroy $hull}
167         }
168     }
169
170     ###
171     # Clean up children
172     ###
173     foreach subobj [info command [self]/*] {
174         catch {$subobj destroy}
175     }
176     foreach subobj [info command [self].*] {
177         if {[winfo exists $subobj]} continue
178         catch {$subobj destroy}
179     }
180
181
182
183
184
185
186
187
188 }

```

```

143 # the <Destroy> we are seeing is intended for us.
144 ###
145 method Hull_Destroy {} {
146     ###
147     # Destroy our Tk representation
148     ###
149     my variable tkalias
150     set tkWidget {}
151     if {[info exists tkalias]} {
152         set tkWidget $tkalias
153     }
154     if {$tkWidget eq {}} {
155         set tkWidget [my widget hull]
156     }
157     if {$tkWidget eq {}} {
158         set tkWidget [my organ hull]
159     }
160     if {$tkWidget ne {}} {
161         my Hull_Unbind $tkWidget
162     }
163
164
165
166
167 }
168
169 ###
170 # Clean up children
171 ###
172 foreach subobj [info command [self]/*] {
173     catch {$subobj destroy}
174 }
175
176 foreach subobj [info command [self].*] {
177     if {[winfo exists $subobj]} continue
178     catch {$subobj destroy}
179 }
180
181
182 catch {my action destroy}
183
184 # Destroy an alias we may have created
185 if { $tkWidget ni {. {}} && [winfo exists $tkWidget] } {
186     ::destroy $tkWidget
187 }
188
189 }

```

Core Development

- Core development is driven by TIPs
- TIPs are driven by developer interest
- Developers should not be lashed into making broken code work

Tcl Users

- Many are not even aware they are running Tcl
- Wrap the workings of production software around semi-supported extensions
- Expect to be able to write code once, ensure that it works, and not have to re-write it
- Care less about “fundamental flaws” and more by the cost of maintaining software

Sean's Stupid Idea

- Let's make an environment where Tcl Developers and Tcl Users can both agree should always work
- Let's make that tool something that can run in any environment
- Let's make the process of full-up testing something that is repeatable

Sobyk

- Sobyk doesn't stand for anything.
- The name was picked out of a fantasy name generator tool because it spoke to me
- Is a distinct enough pattern that application devs who cargo-cult can simply globally search for references to "Sobyk" and replace it with their own product's name

Sobek

Egyptian God

(see: [https://
en.wikipedia.org/wiki/
Sobek](https://en.wikipedia.org/wiki/Sobek))

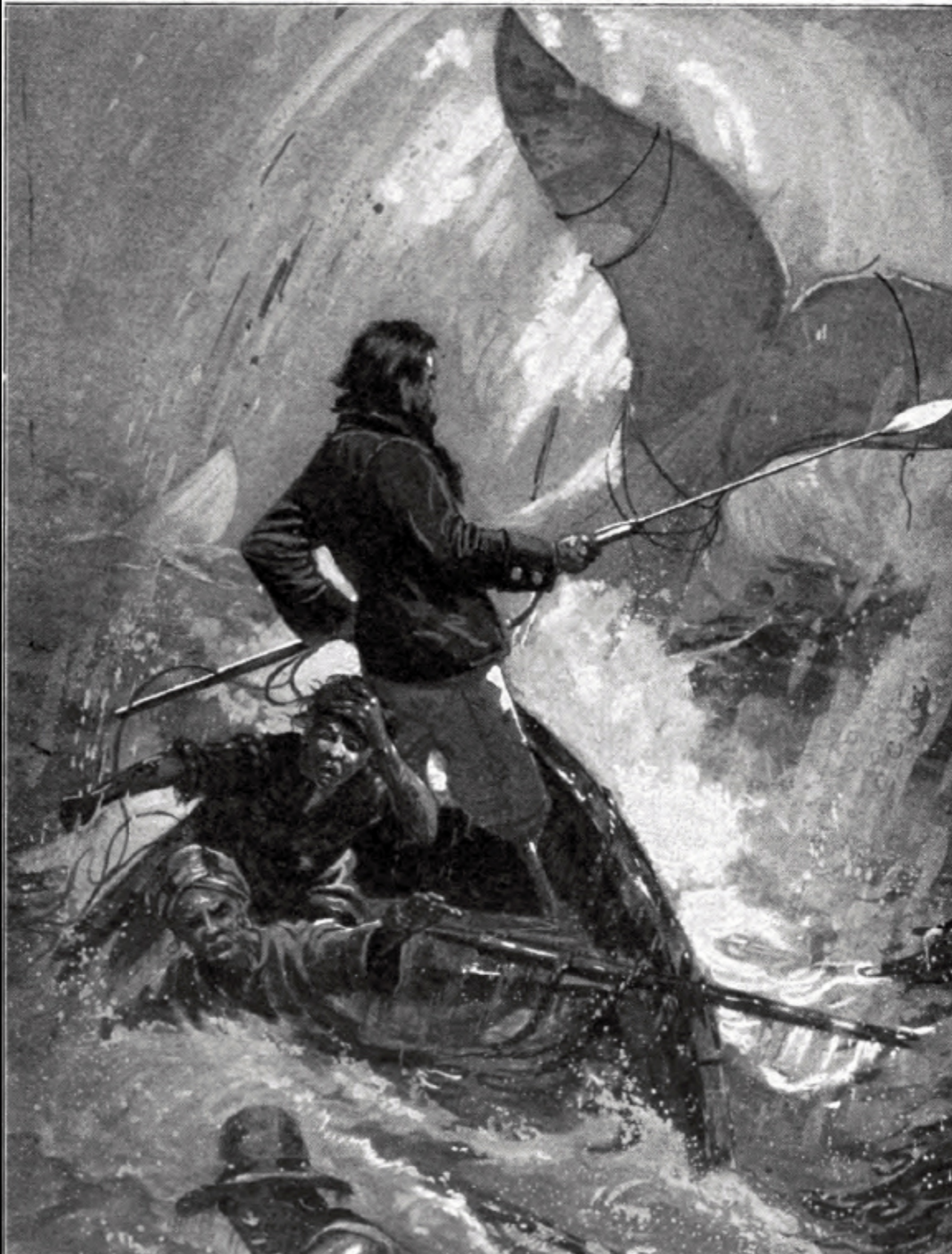


Moby Applications

- Projects that ship one binary that is the entire working environment
- Need to maintain the ability to re-run processes with older releases to compare output
- Are end-user applications which need to work on Mac and Windows, simultaneously
- Documents and simulations developed for these can involve man-years of effort and multi-year support contracts

Moby Applications

- Major Efforts
- Old code
- Binary Packages
- Year's long Commitments



And messing with them leads to old men emerging from halfway around the world screaming like you cut their leg off



Moby Apps

Product	Developer	Current State
The Integrated Recoverability Model	T&E Solutions, LLC	Actively Developed
BRL Cad	Army Research Labs	Actively Maintained, Stuck in Tcl 8.5
SOAR (Tcl Implementation)	University of Michigan	Continues to exist despite the best intentions of management
Clay Game Engine	(Me)	Hypothetical Product

Wait... What

- The “Clay Game Engine” is a thought experiment
- The project models all of the problems of a MOBY application
- Project is engineered from the start to include regression tests for high-level integration
- Development work in CGE will act as a path to bring other MOBY Applications out of the darkness

Why a Game?



*In ev'ry job that must be done
There is an element of fun
You find the fun and snap!
The job's a game*

Just for Fun

- Developers are human beings
- We normally require payment for drudge work
- Testing is not fun. Supporting Mac and Windows is even less fun.
- To retain developer interest we need to make the end result something that is fun

Game Engine Requirements

- Development effort for a title should be roughly that of RenPy
- Rather than target SDL, games would utilize either Tk or HTML and SVG in a captive browser
- The Game Engine also happens to mirror the simulation nature of many MOBY applications

HELP

WANTED

Though it's more
like...



Why Can't I use Undroid / KBS / KitCreator

- Undroid / KBS / KitCreator make generic Tcl environments
- Each imposes shims on packages that conflict with the shims that MOBY applications already have on those same packages
- Many “simplify” the build process by disabling options on packages that MOBY application users depend on

Bits That I've Worked Out So Far

- Build fulfillment works with ZipFS based kits using Tcl 8.6
- Build system is based on Practcl, can build the Integrated Recoverability Model for both Mac and Windows
- User-Developed code in both Tcl and C is “easy” to integrate

Features (that work)

- Mac and Windows applications can be built directly from source
- The source is checked out of fossil or github, using version tags that are known to play well together
- The Tcl-Based build system supports falling back to TEA, Autoconf, and CMAKE where existing build system is not worth the hassle of rewriting
- Mirrors on <http://fossil.etoyoc.com/fossil> contain branches of each major project that include shims to support the Practcl build system

What Needs to be Done

- Windows is currently cross compiled in MinGW. Future builds need to use MSVC
- Code Signing
- Big component will be external project and code provenance tracking. Not even started yet.
- The impact of radical changes to the core and Tcllib on older applications needs to be taken more seriously by developers



**COMING
ATTRACTIONS**

Code Provenance

- A website devoted to the history of Tcl and Tcl packages
- Focus is on the developers themselves as well as the oddball “beyond the implementation” history of various modules and components
- Allows people to feel like they have made a mark, even if their implementations are later rewritten or replaced
- Content is suggested by the community, but curated by designated historians

Code Provenance

- Will start with GUTTER, and also include machine readable hints for how projects interact, and which versions of packages belong in which profiles

Sobyk Technology Profiles

- Sobyk profiles represent a “technology freeze” that dominates the development cycle of many projects
- Once a selection of packages is selected, tested, and certified, they do not change
- The requirements for each profile will be spelled out, and changes that do not serve a specific requirement will have to wait for another development cycle

Planned Technology Profiles

- Tcl 8.5.X (BRL Cad)
- Tcl 8.6.8 (IRM Version 4)
- Tcl 8.6.X (Technology Evaluator for the Core Team)
- Tcl 8.7.X (Technology Evaluator for the Core Team)

Wait... 8.6.8

- IRM Version 4 has been Certified by the US Navy for simulation work
- I'm stuck in 8.6.8 because that was the core that ran without require a massive rewrite
- Updating the technology after this point requires undergoing another certification process