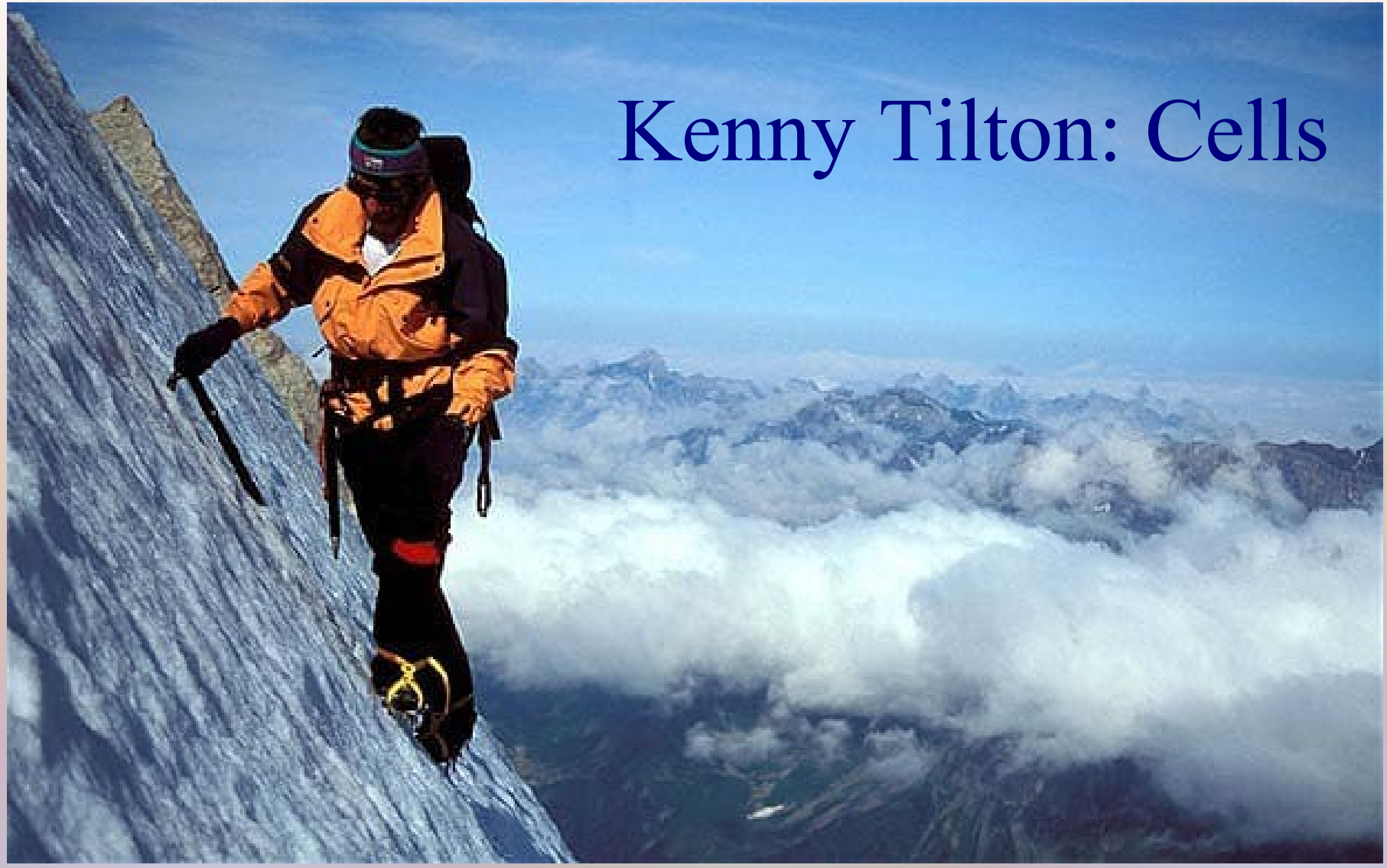
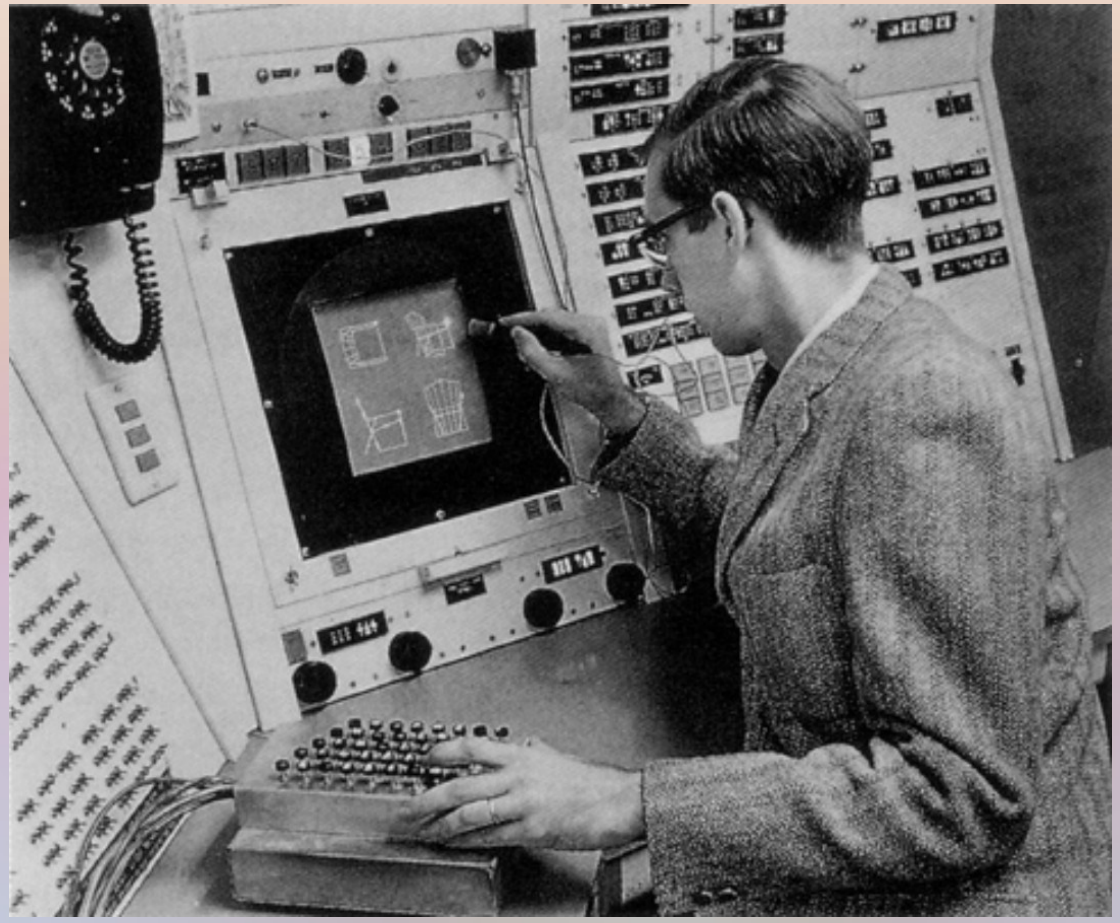


These slides with notes:

https://github.com/kennytilton/cells

It is 1963. What is he doing?

Play Video

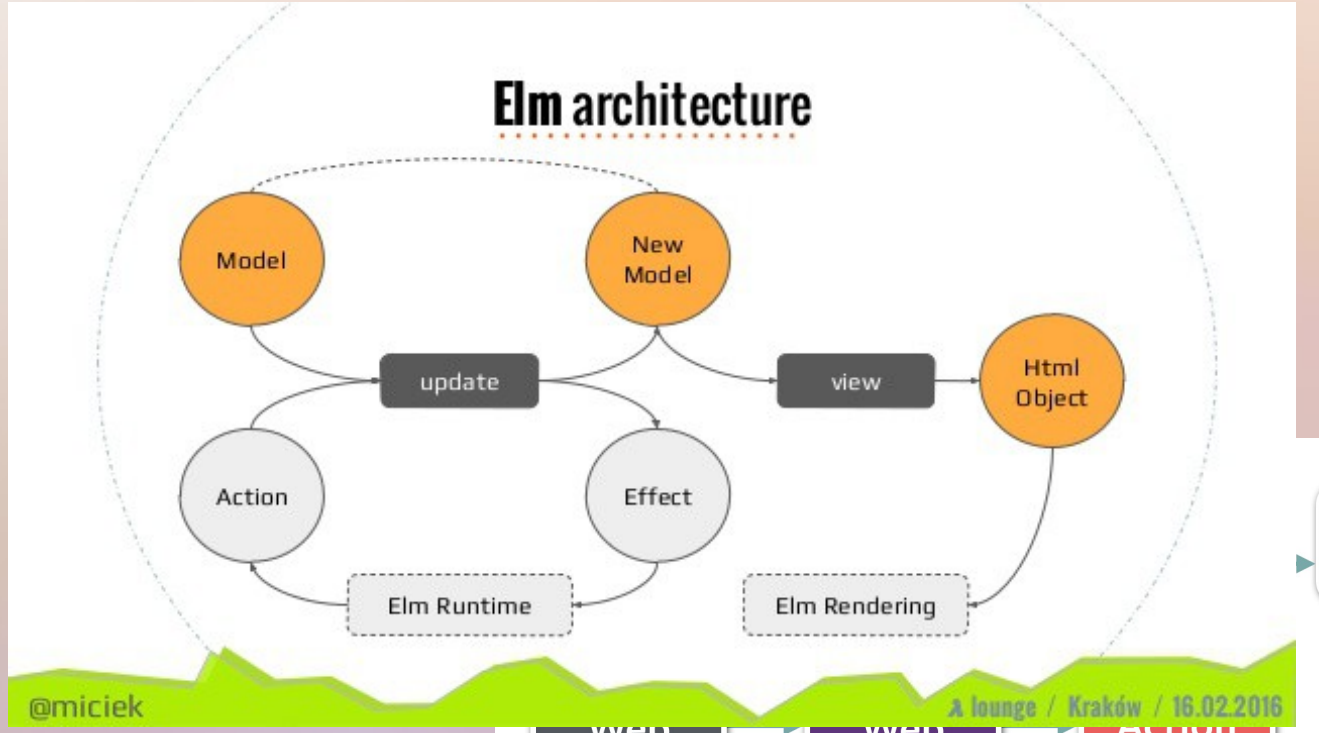


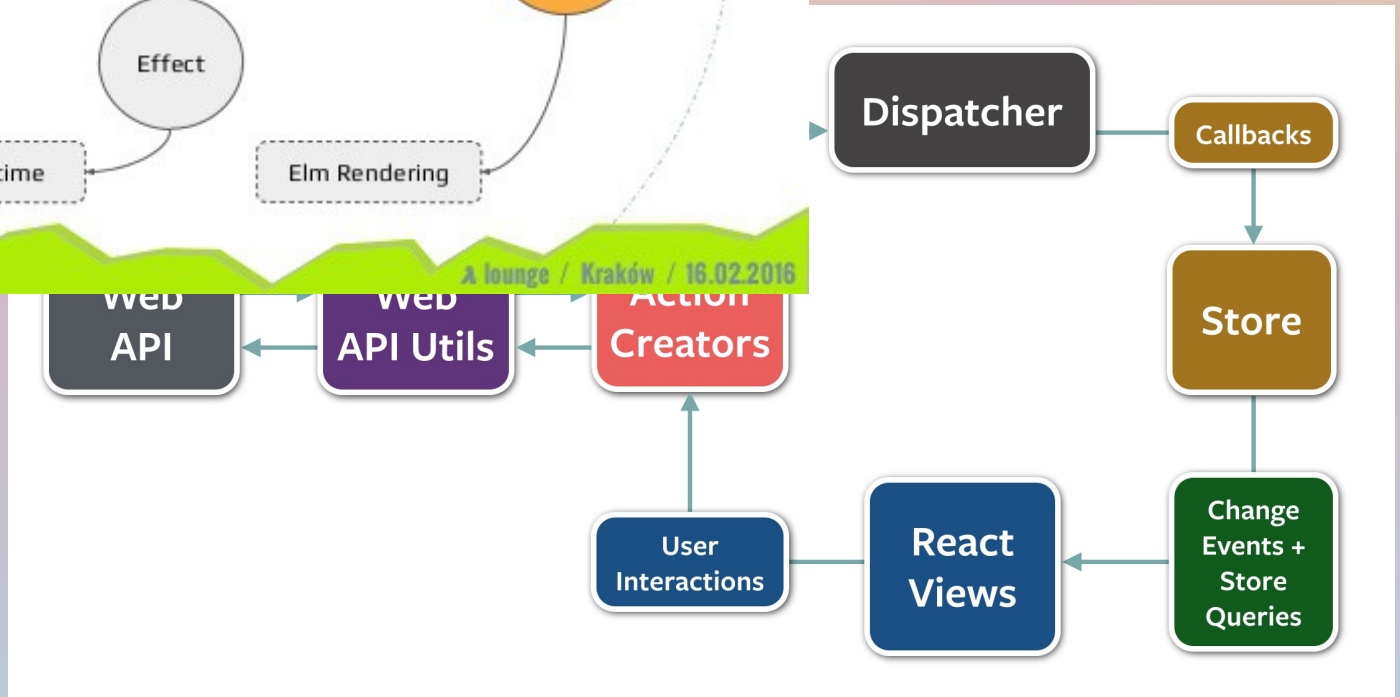
“Unforeseen uses are turning up.”

Why You Might Care

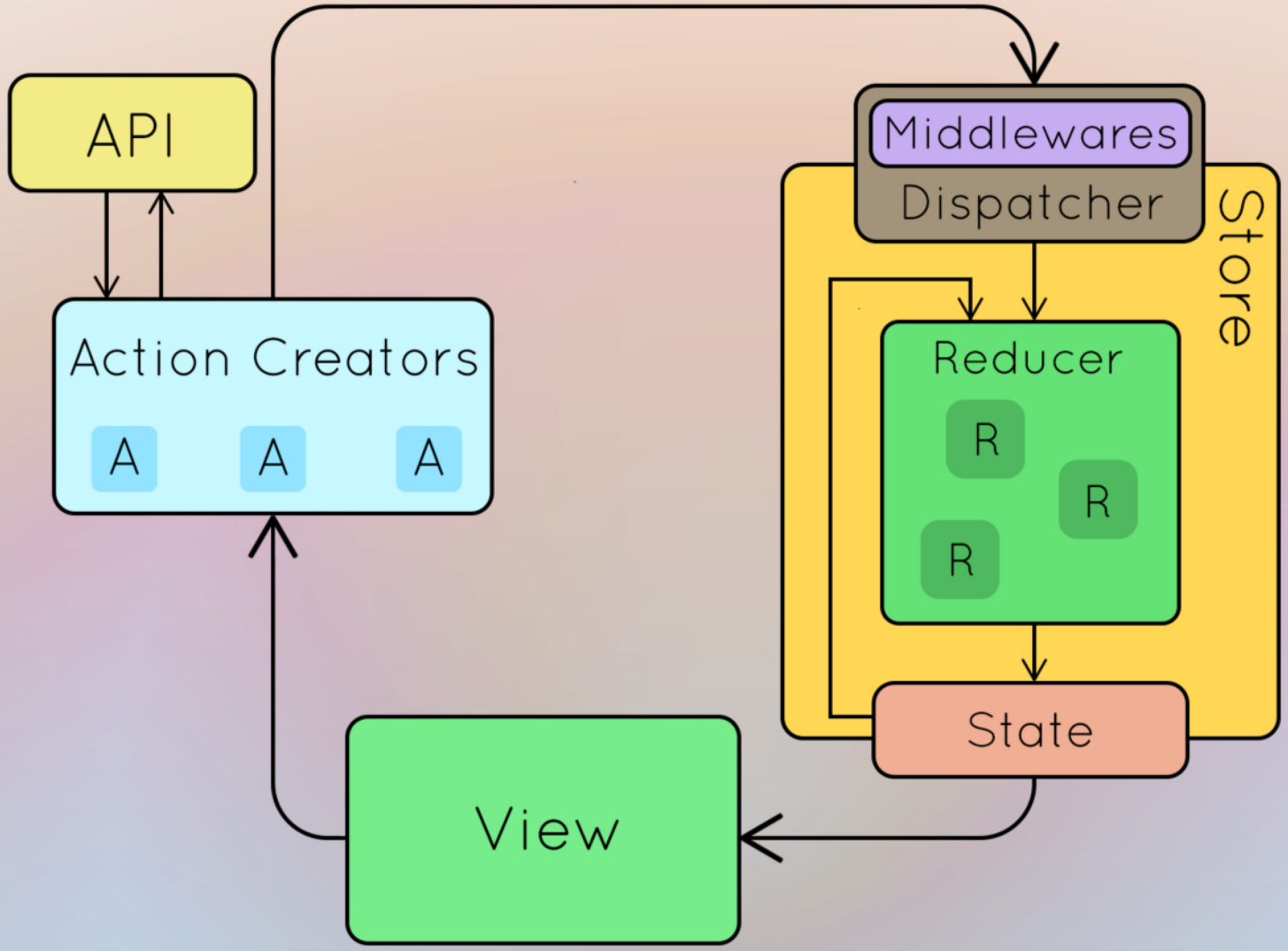
- GUI geometry made trivial
- Yes Silver Bullet
- The Grail of Object Reuse
- Efficiency for free
- Callback Hell eliminated
- RxJS for free
- Reliability
- Debugability
- Fun

Gross Reactivity: Elm/Flux/Redux





Gross Reactivity: Elm/Flux/Redux



Where You Can Use It

"It is useful for any application involving an interesting amount of long-lived state and a stream of unpredictable inputs."

- Kenny Tilton, *The Cells Manifesto*

Examples:

- Any GUI application
- RoboCup virtual client
- Real-time controllers
- Not ETL

Dataflow Programming

$$(f\ s^*) \sim\!> s$$

$$X \rightarrow s$$

$$S \sim\!> \textit{side-effects}$$

Our puzzle tonight: what elemental quality of nature does the above capture making it so powerful?

- Change
- Causation
- Communication
- Animation

Cells the Accident: Step #1

We had a hard problem: GUI layout with elements dynamically shifting and resizing, appearing and disappearing:

```
(make-instance 'textedit
    :right (lambda (self)
             (+ (left self)
                (fontStringWidth (text self)
                  (math-font *app*)))))
```

- Functional/declarative (simpler, more reliable code)
- Property to property (efficient, minimal change)
- Instance specific (object re-use)

Step #2: Transparency

Great, but how to we *read* "lr"? (funcall (lr self) self)? What if "lr" is 42 instead of a function? We would have to read the slot and see if it is a function. We need an accessor!

```
(defmethod right ((self geo-box))
   (let ((sv (slot-value self 'right)))
      (if (functionp sv)
         (funcall sv self)
         sv)))
```

Functional is slow. Step #3.

Cache the computation (so now we need state):

```
(defstruct cell rule value)

(defmethod right ((self geo-box))
   (b-if cell (slot-cell self 'right)
     (if (unboundp (value cell))
       (setf (value cell)
          (funcall rule self)))
       (value cell))))
```

Full sweep is still slow. Step #4a.

Track dependents so we can notify them:

```
(defstruct cell rule value users)

(defmethod right ((self geo-box))
  (b-if cell (slot-cell self 'right)
     (progn
        (when *user*
           (c-record-user cell *user*))
        (let ((*user* cell))
           (setf (value cell)
              (funcall rule cell)))
     (slot-value self 'right)))
```

Step #4b: Closing the Loop

```
(defmethod (setf right) (new-val (self geo-box))
  (b-if cell (slot-cell self 'right)
    (prog1 new-val
      (when (not (eql new-val (value cell)))
        (c-observe 'right self
                   new-val (value cell))
        (setf (value cell) new-val)
        (dolist (user (users cell))
          (c-recompute user))))
    (setf (slot-value self 'right) new-val))))
```

Transparent, specific, automatic state change.

Step #5: Tell the World

Once we recalculate everything, how do we manifest the new values? Observers.

```
(defobserver right ((self geo-box) newv oldv)
   (qdraw:invalidate-rect (rectangle self)))
```

Step #0: In the Beginning

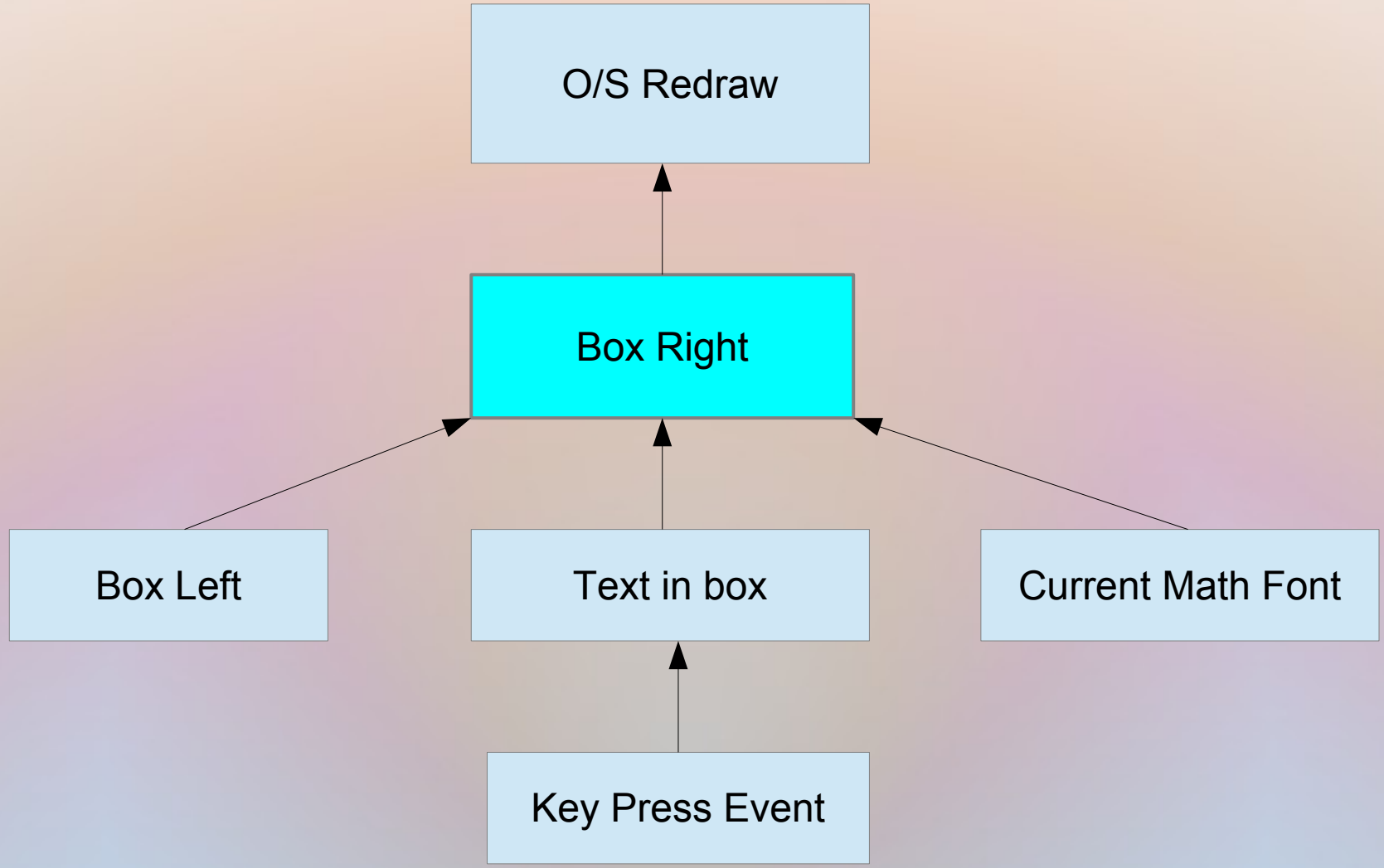
Where did the data *start*?

```
(defn todo-entry-field []
  (input {
      :class       "new-todo"
      :autofocus   true
      :placeholder "What needs doing?"
      :text (c-input "")
      :oninput  (c? (fn [evt]
                      (setf (text self)
                        (form/getValue (.-target evt))))))}))
```

The Call Stack Inverted

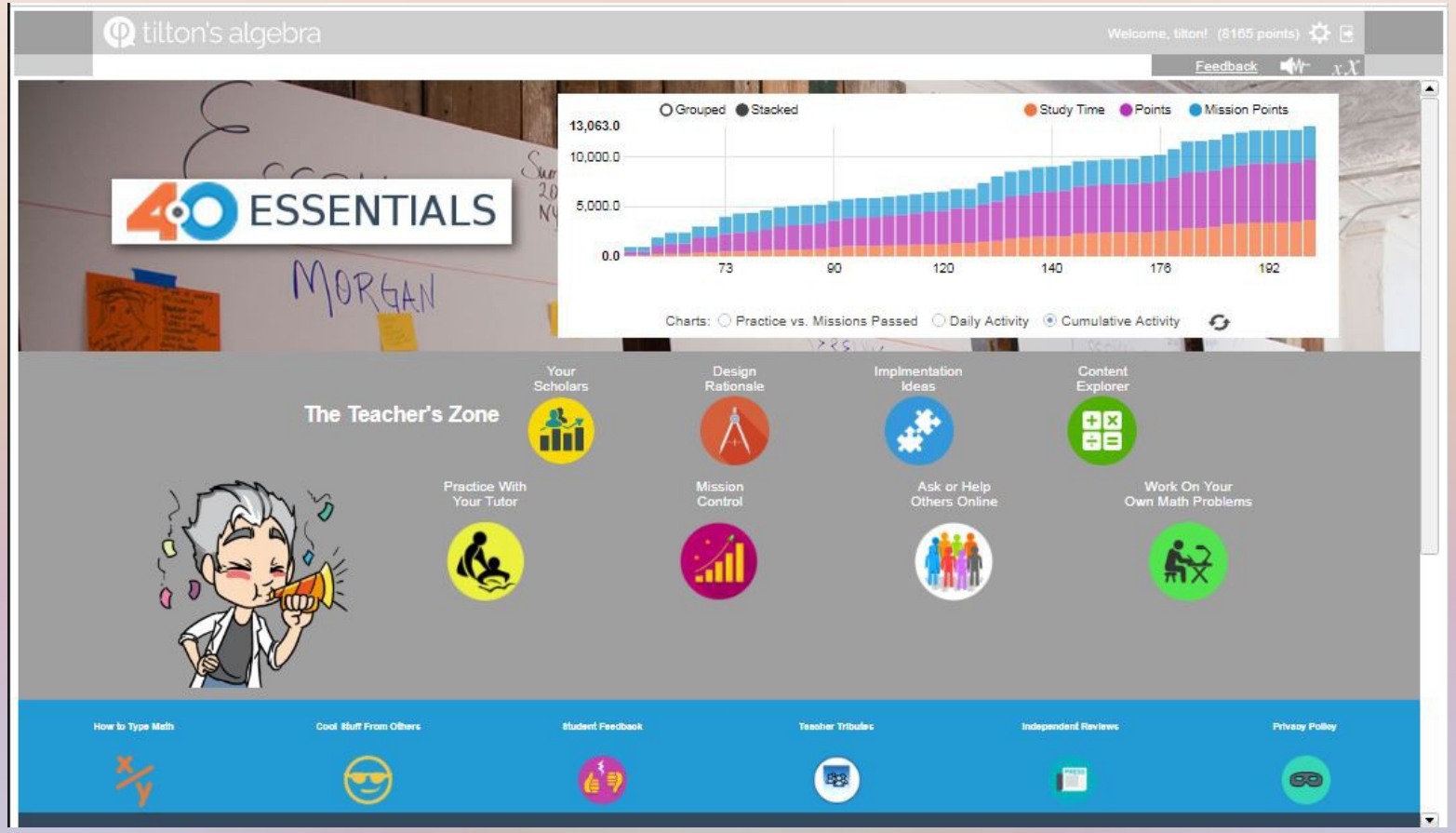
Inversion Goggles.

That Was Simple



This Is Not Simple

tiltonsalgebra.com

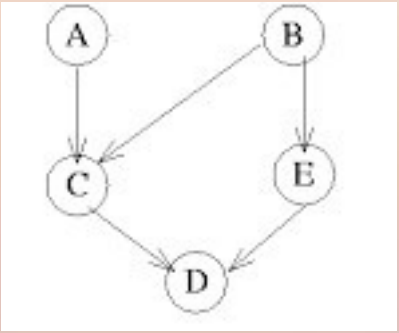


And CliniSys(tm), a Clinical Drug Trial Manager.
80kloc of Common Lisp/qooxdoo.

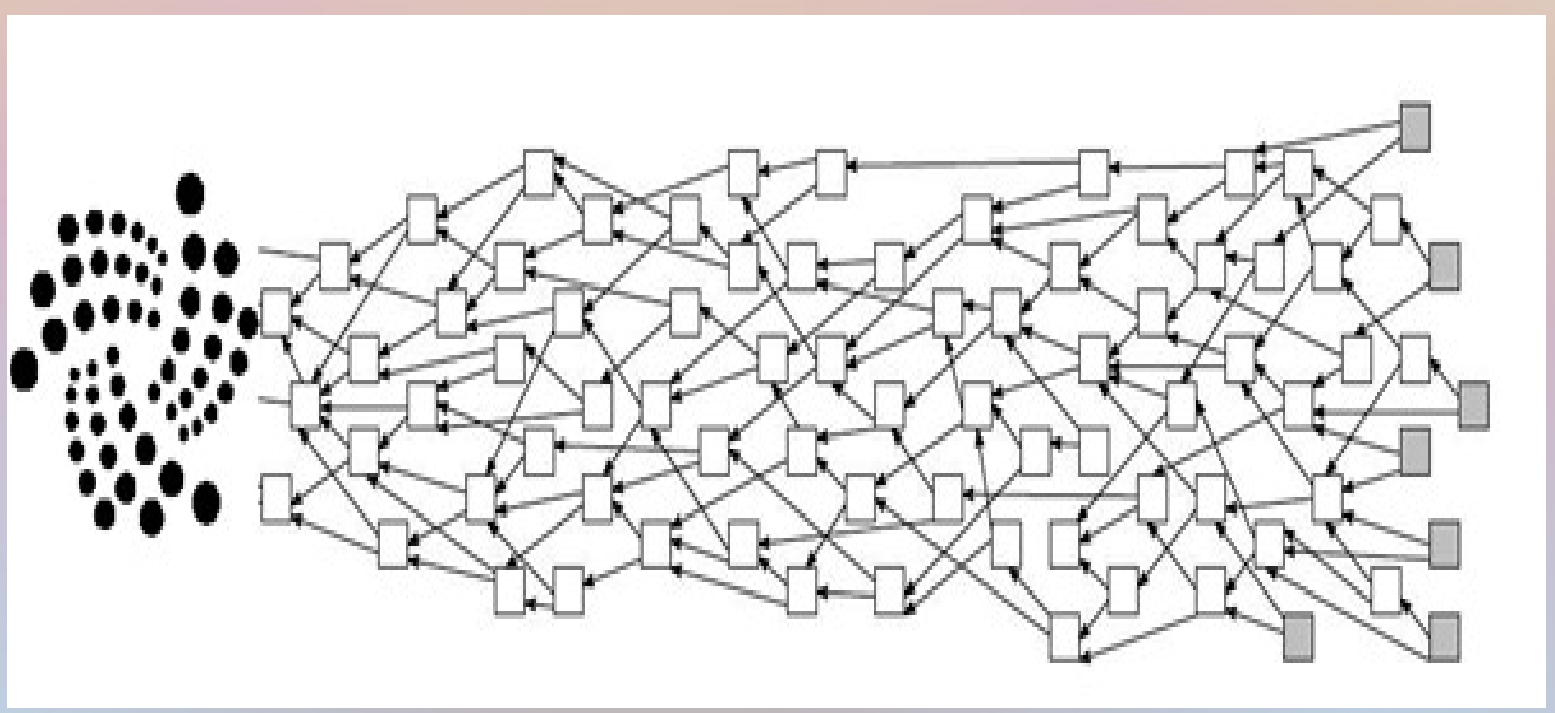
Fred Brooks Was Right.

About the Problem.

Not
Bad.

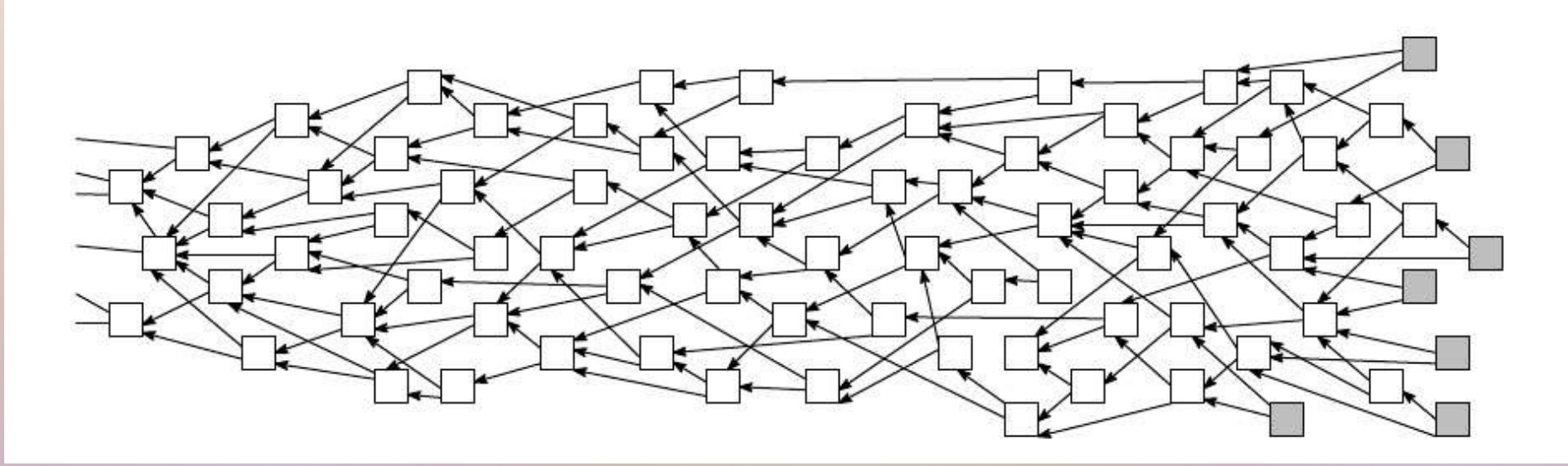


Good
Luck.



Adobe Adam: Sean Parent

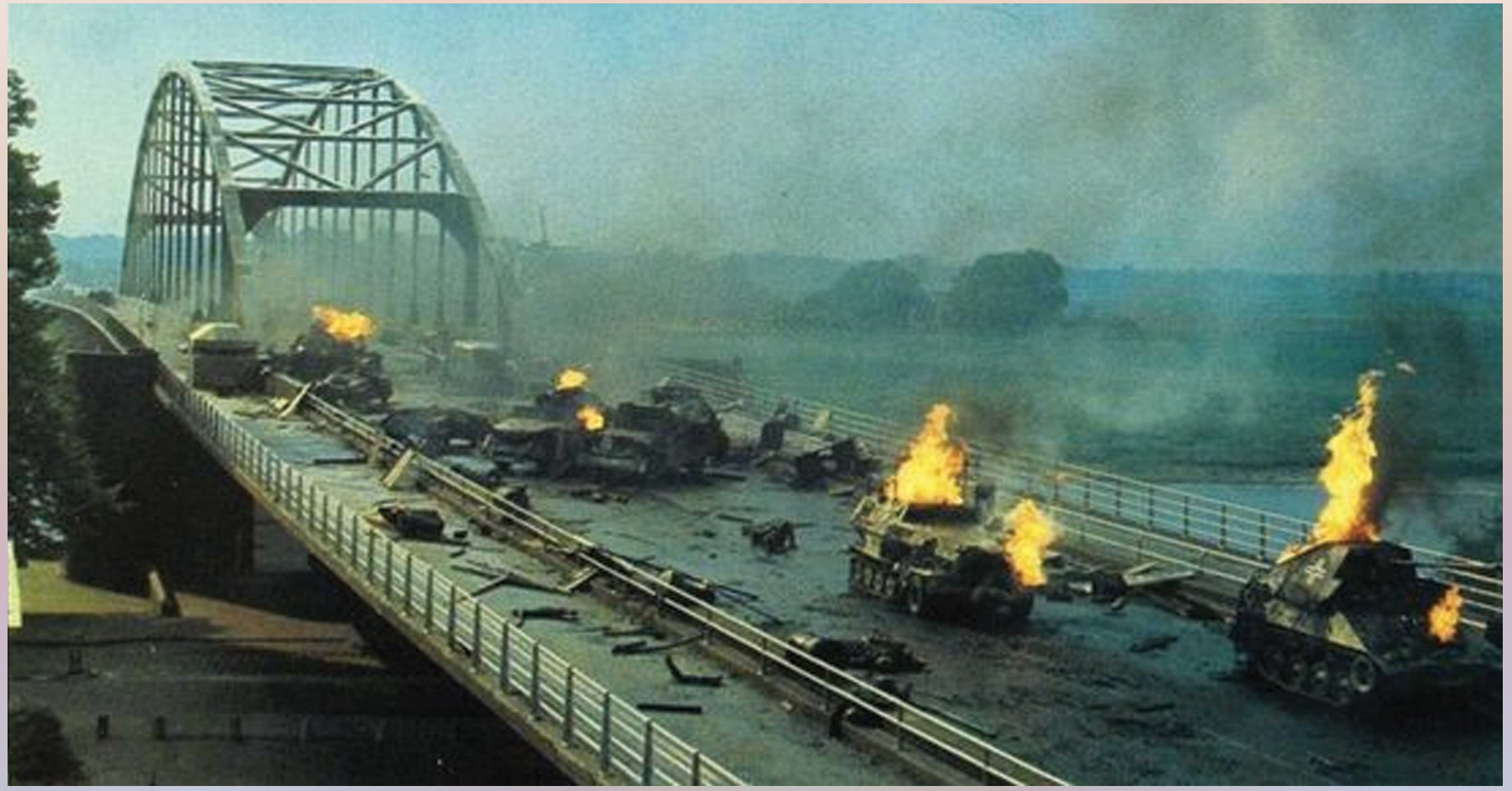
Working on GUI layout issues complexity, Sean recognized a bipartite graph. A DAG.



Created Adobe Adam. Reduced interface code by 90% and bugs by 20%.

Constraints

A Bridge Too Far



Multi-way and partial constraints

Constraint: A must equal B + C
Constraint: B = 2, C = 40.
What is A?

Now change C to 30.
Add constraints B < 10 and B is prime.
What is A?

25 Years of Logic Programming

"One common problem is the sometimes unpredictable behavior of the constraint model: even small changes in a program can lead to a dramatic change in the performance. This is because the process of performance debugging, designing and improving constraint programs is currently not well understood. Related to this problem is the long learning curve that novices have experienced. While simple applications can be built almost immediately, it can take a long time to become familiar with the full power of a constraint system."

Rossi, Francesca

In "Constraint (Logic) Programming: A Survey on Research and Applications", 1997

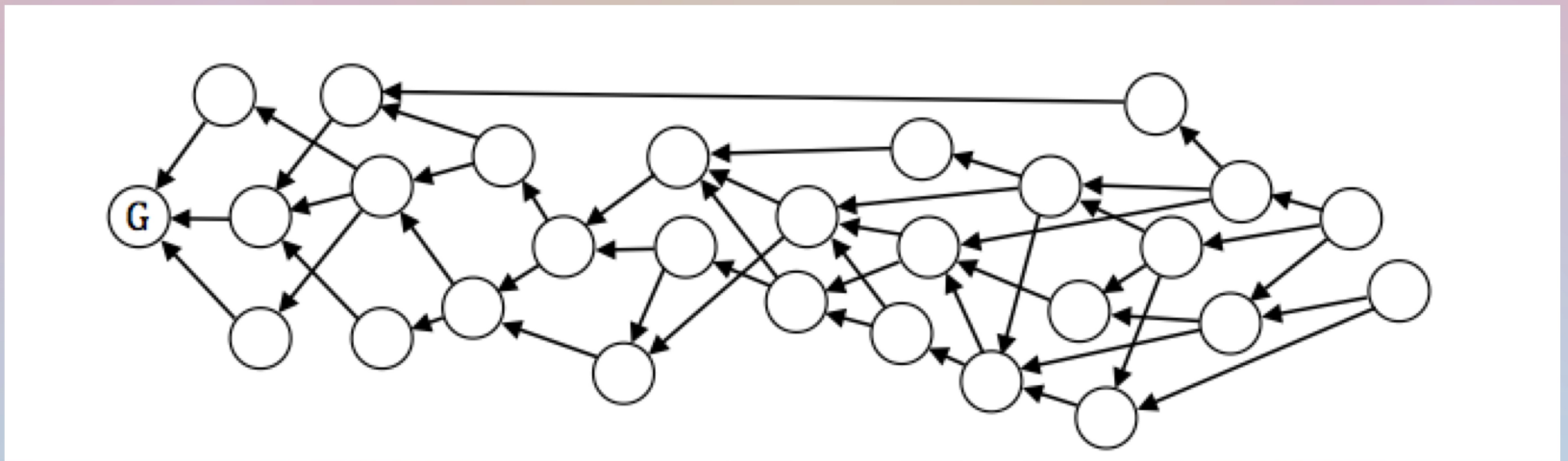
The Dataflow Wave
It's Here

Glitch-free property to property state change propagation by automatically detected dependents.

- Common Lisp: Garnet/KR and Cells
- C++: Adobe Adam
- Python: Trellis
- Binding.Scala
- Javascript: MobX and Matrix
- Clojure/CLJS: Javelin and Matrix

ReactiveX/RxJS

- Google product
- Explicit streams
- Stream metaphor from Scheme FRP: FrTime, FlapJax
- Outside-in programming



Synapses as Streams

syn·apse [sin-aps, si-naps] Physiology noun - A junction between two nerve cells, consisting of a minute gap across which impulses pass by diffusion of a neurotransmitter.

```
(deftest synaptic-delta

  (let [sensor (cI nil)
        alarm (cF (let [delta (with-synapse (:delta-sensor [prior (atom nil)])
                                (when-let [reading (c-get sensor)]
                                  (let [delta (Math/abs (if @prior
                                                          (- reading @prior)
                                                          0prior))]
                                    (reset! prior reading)
                                    delta)))]
                    (if (> delta 5)
                      :on :off)))]
    ...))
```

Causation, animation, or communication?

FrTime, FlapJax

- FrTime: Scheme FRP: http://docs.racket-lang.org/frtime/
- FlapJax: FRP Web framework, a Javascript translation of FrTime: http://www.flapjax-lang.org
- Javelin (ClojureScript)
- “lifting” implementation has efficiency and coding issues: (if A B C) and (+ 40 (compute-with box)),

Trellis for Pythonistas

- Phillip Eby's development by Cells
- http://peak.telecommunity.com/DevCenter/Trellis
- https://pypi.python.org/pypi/Trellis/0.7a1
- David Gelerntner's “Trellis” from “Mirror Worlds”: https://global.oup.com/academic/product/mirror-worlds-9780195079067?cc=us&lang=en&

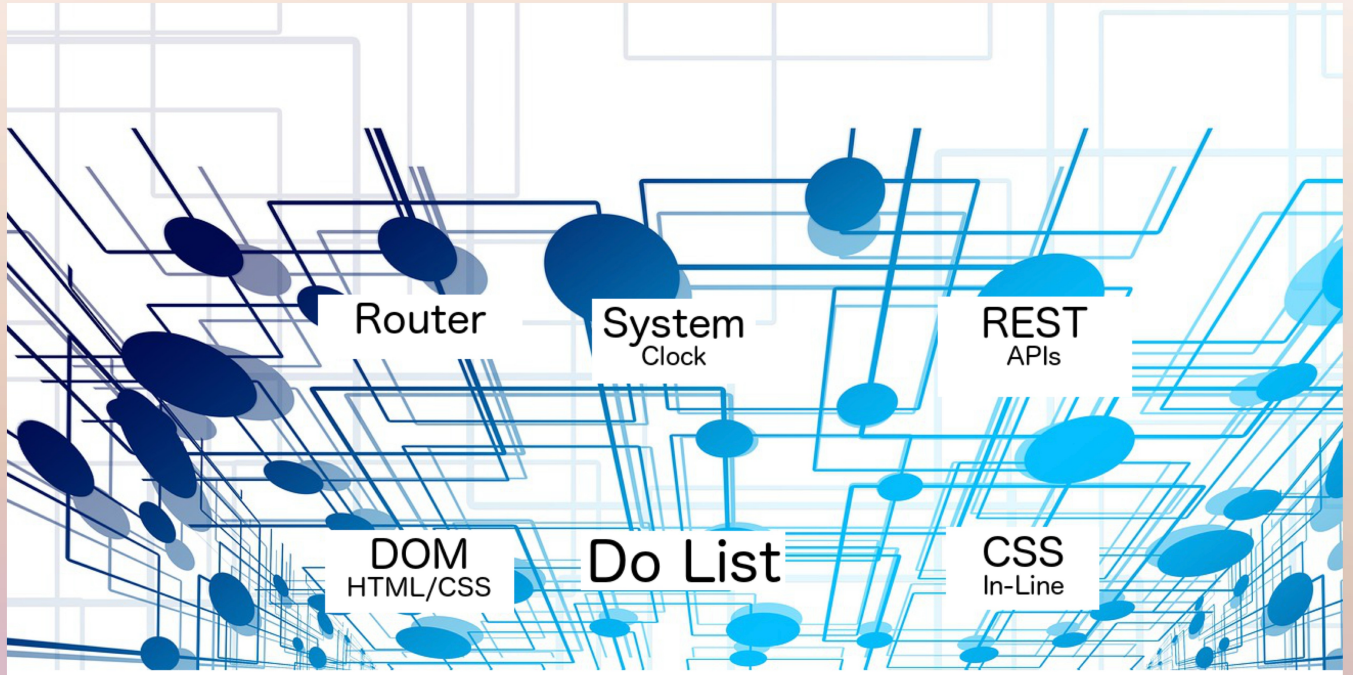
Into the Weeds: Anti-glitch schemes

- Topo sort does not work: conditional branching means newly travelled paths may emerge on the latest change
- MobX marks direct dependents as “dirty” and their direct dependents as “maybe dirty”.
- Matrix uses an integer “pulse” that Cells can compare with their own “last calculated pulse”.

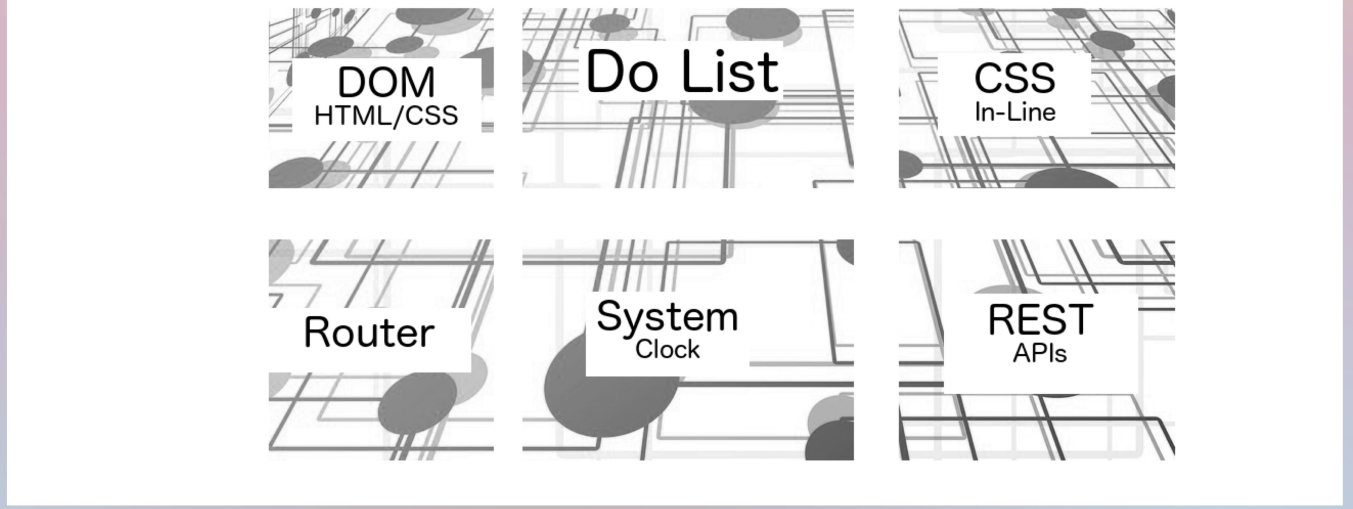
Glitch Fix: MobX vs Matrix

- MobX started with a counter (of dependencies changed).
- Now MobX pushes “must recompute” or “might have to recompute” states to dependents.
- Cells uses increasing global integer “pulse”. Kenny is not sure which is best. Tie-breaker may be which scales to multi-threading best.

Matrix



App Behavior



Evaluating Reactive Libraries

- Is it a “lifting” library? Problem.
- Is pub-sub explicit? Awkward.
- Is it really Reactive? ReactJS is not, but it is declarative which is nice.
- Does it have “glitches”? Not the end of the world, but others are glitch-free so why not?

The Benefits

- ✔ GUI geometry made trivial
- ✔ Yes Silver Bullet
- ✔ The Grail of Object Reuse
- ✔ Efficiency for free
- ✔ Callback Hell eliminated
- ✔ RxJS for free
- ✔ Reliability
- ✔ Debugability
- ✔ Fun

Reactive Programming

$$s <\sim (f\ s^*)$$

$$s <= x$$

$$side\text{-}effects^* <\sim s$$

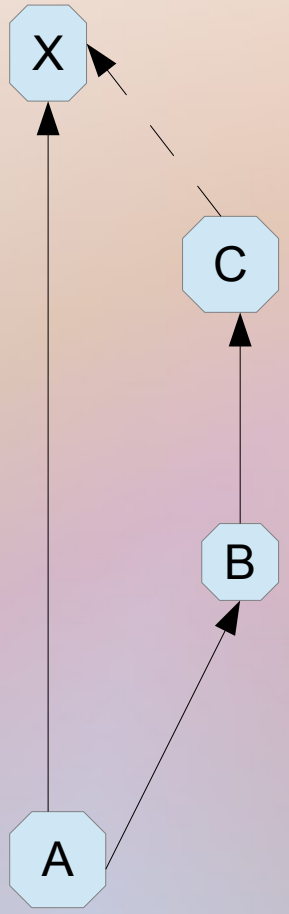
Our puzzle tonight: what elemental quality of nature does the above capture making it so powerful?

- Causation
- Communication
- Animation
- Other_____

Functional Reactive Programming (FRP)

- Conan Eliot Lambda-Jam 2015 https://youtu.be/j3Q32brCUAI
- Continuous time, yes
- Graphs and dataflow, no.

The Glitch



Known

Conditional

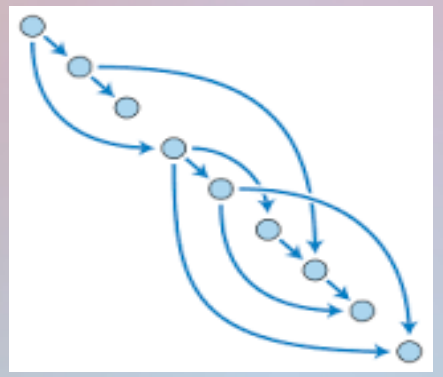
How ensure C is computed before X when A changes?

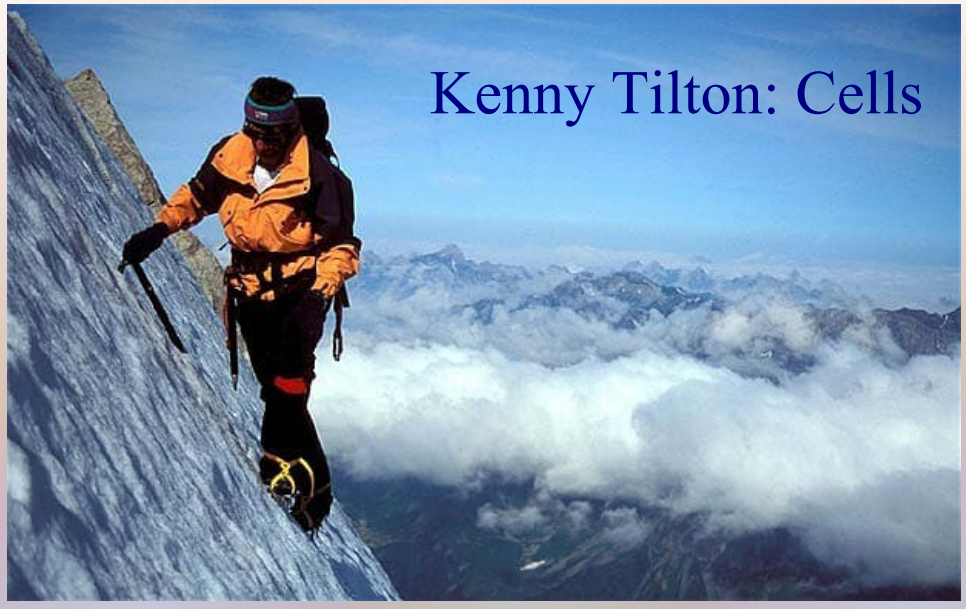
Glitches and RoboCup Sim

The Cells are dead! Long live Cells!

One (UDP) input: a visual sensory dump every 100ms.

Puzzle: Why did that create the first glitch fatality?



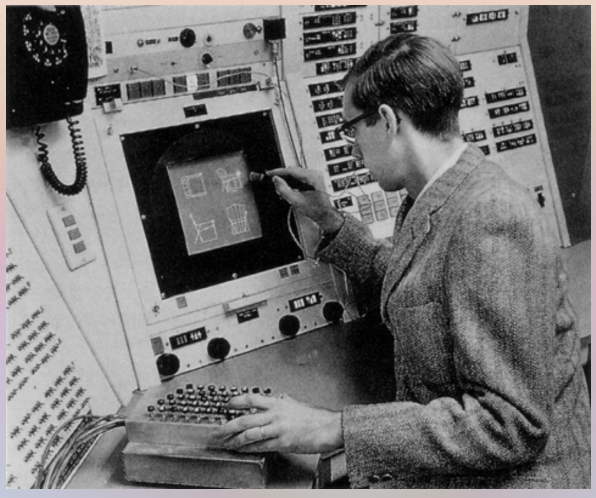


These slides with notes:

https://github.com/kennytilton/cells

It is 1963. What is he doing?

Play Video



"Unforeseen uses are turning up."

Ivan's objective was simply to use a computer to help a mechanical engineer with a design drawing.

Some software was so hard to use that engineers were more effective using paper. To give SketchPad an edge over paper, Ivan let the user declare constraints such as "these segments are horizontal" or "I am drawing an arc". A constraint solver translated sloppy light pen gestures into precise coordinates to honor the declarations. Fun tech note: constraints were declared by throwing toggle switches.

- Age 15 wrote biggest app for SIMON: 8 feet of paper tape
- 1963: Sketchpad, an app as a PhD thesis.
- First "mouse" (a light pen)
- Constraint solver
- Turing Award

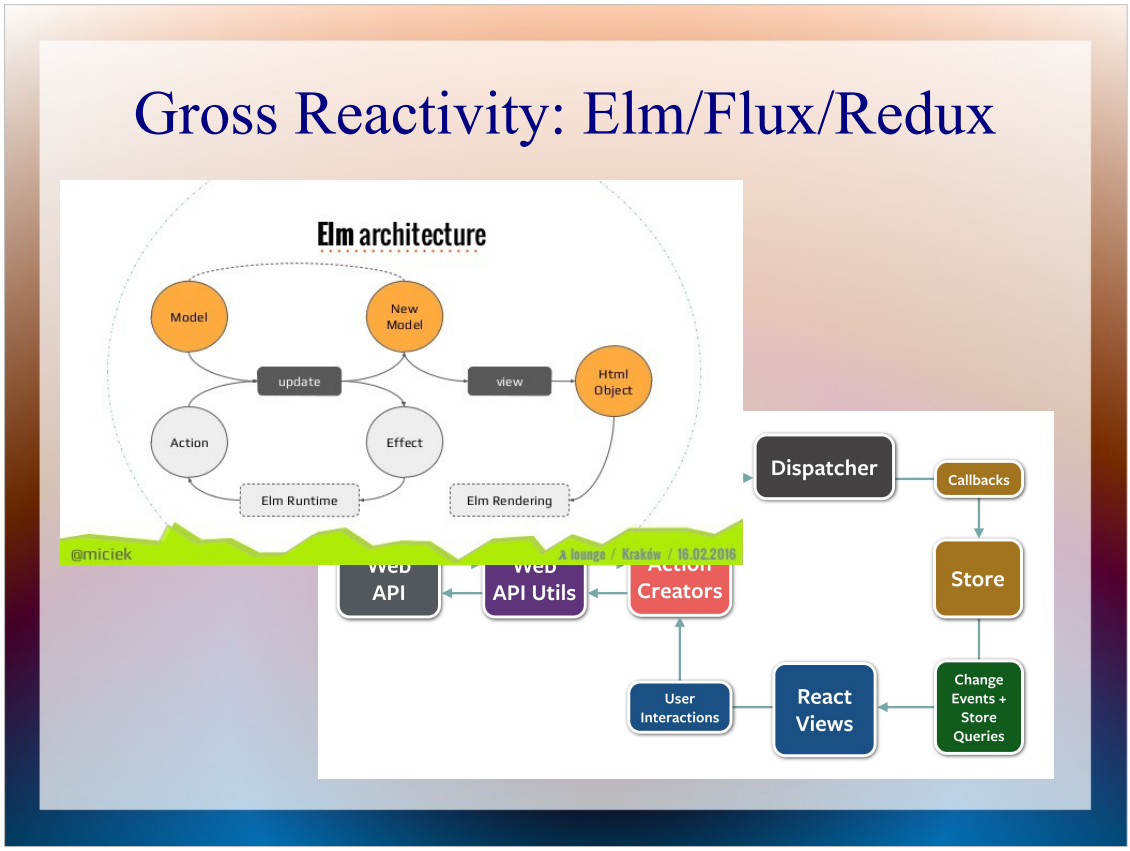
Thesis: Re-released/-formatted 2003
https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-574.pdf

Turing award: :
https://amturing.acm.org/award_winners/sutherland_3467412.cfm

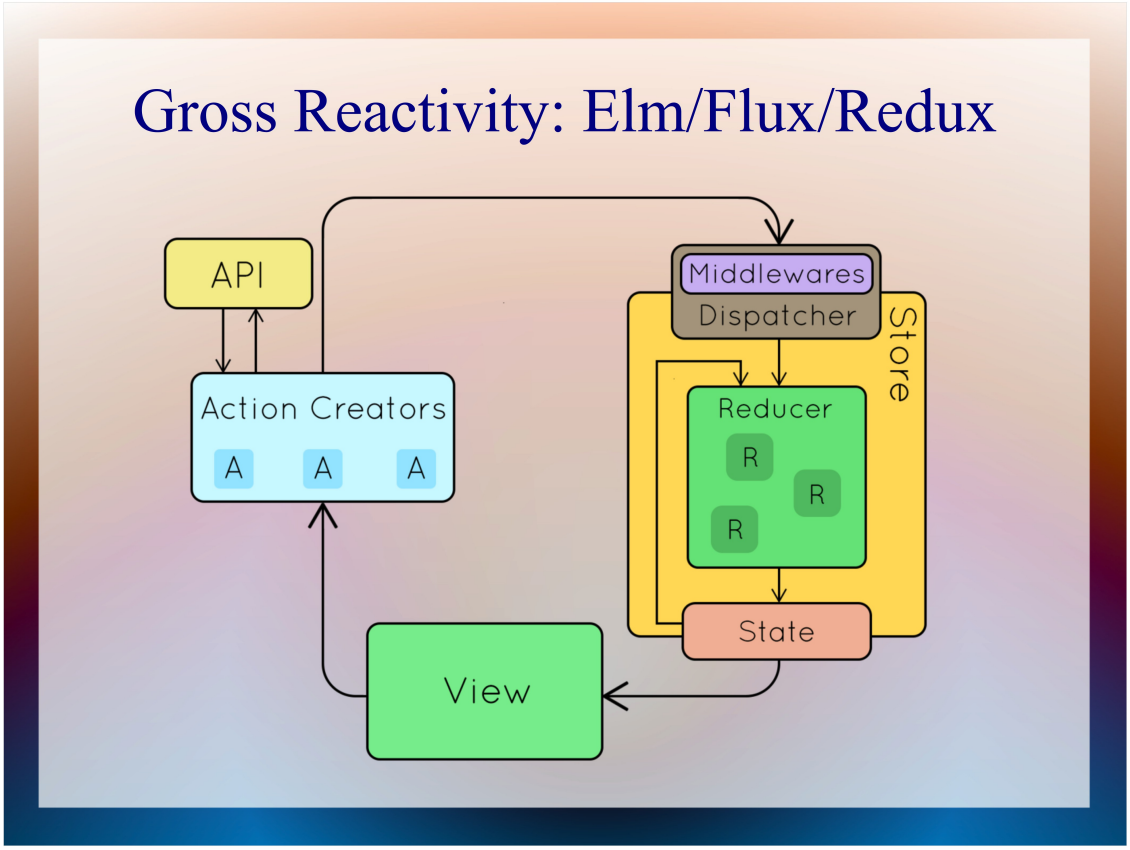
- Demo: https://youtu.be/57wj8diYpgY
- TV show: https://youtu.be/USyoT_Ha_bA

Why You Might Care

- GUI geometry made trivial
- Yes Silver Bullet
- The Grail of Object Reuse
- Efficiency for free
- Callback Hell eliminated
- RxJS for free
- Reliability
- Debugability
- Fun



Flux: https://github.com/facebook/flux



Flux: https://github.com/facebook/flux

Where You Can Use It

"It is useful for any application involving an interesting amount of long-lived state and a stream of unpredictable inputs."

- Kenny Tilton, *The Cells Manifesto*

Examples:

- Any GUI application
- RoboCup virtual client
- Real-time controllers
- Not ETL

Dataflow Programming

$$(f\ s^*) \sim> s$$

$$X \rightarrow s$$

$$S \sim> \textit{side-effects}$$

Our puzzle tonight: what elemental quality of nature does the above capture making it so powerful?

- Change
- Causation
- Communication
- Animation

Cells the Accident: Step #1

We had a hard problem: GUI layout with elements dynamically shifting and resizing, appearing and disappearing:

```
(make-instance 'textedit
    :right (lambda (self)
             (+ (left self)
                (fontStringWidth (text self)
                  (math-font *app*)))))
```

- Functional/declarative (simpler, more reliable code)
- Property to property (efficient, minimal change)
- Instance specific (object re-use)

Step #2: Transparency

Great, but how to we *read* “lr”? (funcall (lr self) self)? What if “lr” is 42 instead of a function? We would have to read the slot and see if it is a function. We need an accessor!

```
(defmethod right ((self geo-box))
   (let ((sv (slot-value self 'right)))
      (if (functionp sv)
         (funcall sv self)
         sv)))
```

Functional is slow. Step #3.

Cache the computation (so now we need state):

```
(defstruct cell rule value)

(defmethod right ((self geo-box))
   (b-if cell (slot-cell self 'right)
     (if (unboundp (value cell))
       (setf (value cell)
          (funcall rule self)))
       (value cell))))
```

Full sweep is still slow. Step #4a.

Track dependents so we can notify them:

```
(defstruct cell rule value users)

(defmethod right ((self geo-box))
  (b-if cell (slot-cell self 'right)
     (progn
        (when *user*
           (c-record-user cell *user*))
        (let ((*user* cell))
           (setf (value cell)
             (funcall rule cell)))
     (slot-value self 'right)))
```

Step #4b: Closing the Loop

```
(defmethod (setf right) (new-val (self geo-box))
  (b-if cell (slot-cell self 'right)
    (prog1 new-val
      (when (not (eql new-val (value cell)))
         (c-observe 'right self
                    new-val (value cell))
         (setf (value cell) new-val)
         (dolist (user (users cell))
            (c-recompute user))))
    (setf (slot-value self 'right) new-val))))
```

Transparent, specific, automatic state change.

Step #5: Tell the World

Once we recalculate everything, how do we manifest the new values? Observers.

```
(defobserver right ((self geo-box) newv oldv)
   (qdraw:invalidate-rect (rectangle self)))
```

Step #0: In the Beginning

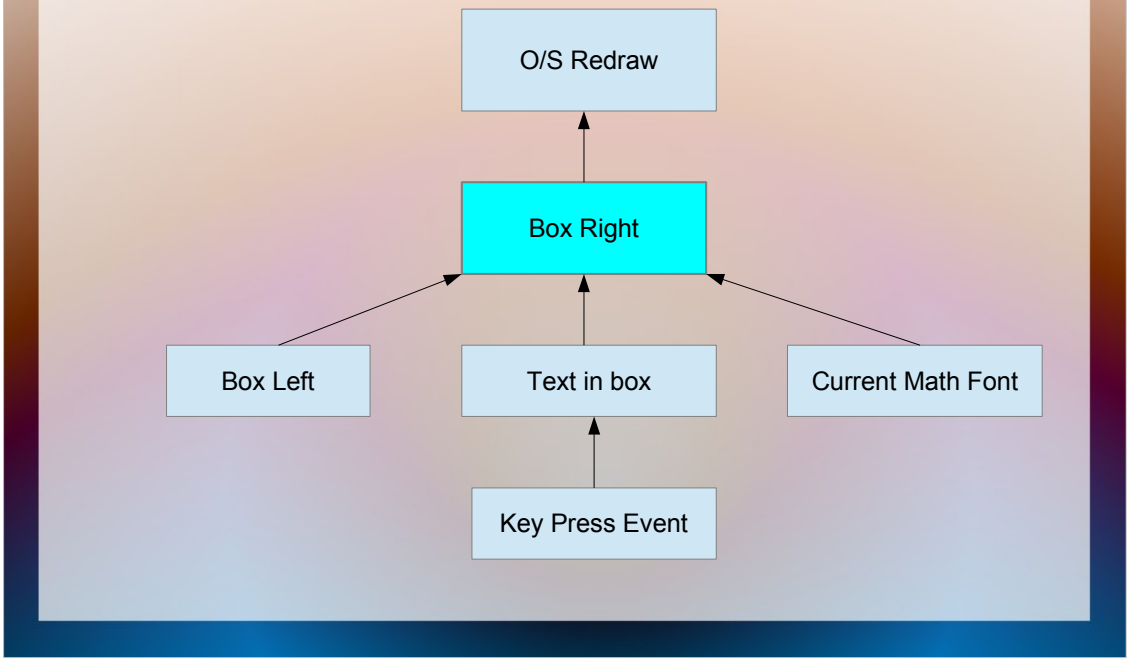
Where did the data *start*?

```
(defn todo-entry-field []
  (input {
      :class       "new-todo"
      :autofocus   true
      :placeholder "What needs doing?"
      :text (c-input "")
      :oninput  (c? (fn [evt]
                      (setf (text self)
                        (form/getValue (.-target evt))))})))
```

The Call Stack Inverted

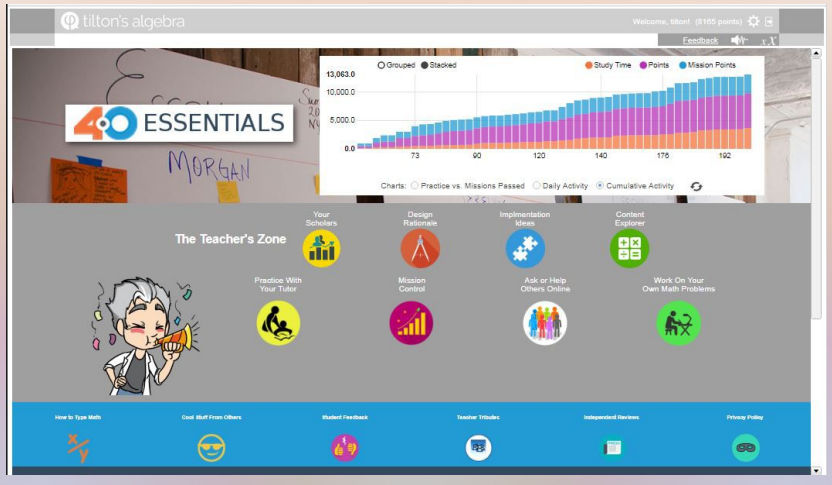
Inversion Goggles.

That Was Simple



This Is Not Simple

tiltonsalgebra.com



And CliniSys(tm), a Clinical Drug Trial Manager.
80kloc of Common Lisp/qooxdoo.

This app has 1218 distinct formulas. At run-time, a formula has on average 1.5 ependencies. The average formula includes 30 Lisp symbols.

On a busy screen, over 1000 cells (comprising 300 formulas) can be active.

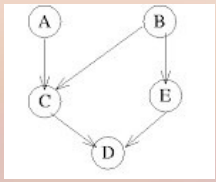
The longest dependency chain is eleven. It works from the type-of the math in the last problem step to the highlighting of that step.

A second experience report was a complete clinical drug trial management system in which the persistent CLOS database was also driven by cells.

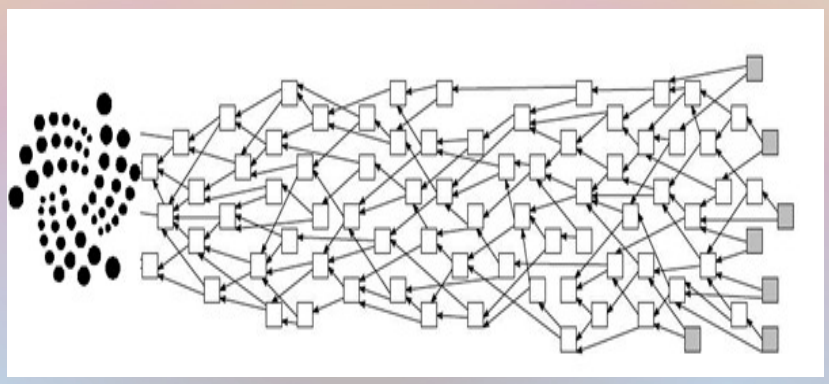
Fred Brooks Was Right.

About the Problem.

Not Bad.



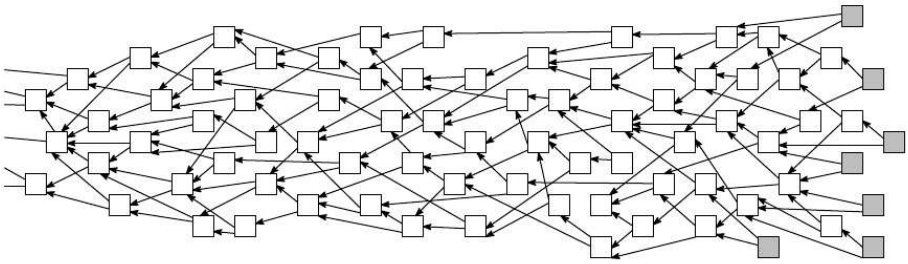
Good Luck.



No Silver Bullet: http://worrydream.com/refs/Brooks-NoSilverBullet.pdf

Adobe Adam: Sean Parent

Working on GUI layout issues complexity, Sean recognized a bipartite graph. A DAG.



Created Adobe Adam. Reduced interface code by 90% and bugs by 20%.

http://stlab.adobe.com/group__asl__overview.html#asl_overv

The Google Talk Haskell Question:
https://youtu.be/4moyKUHApq4?t=49m17s

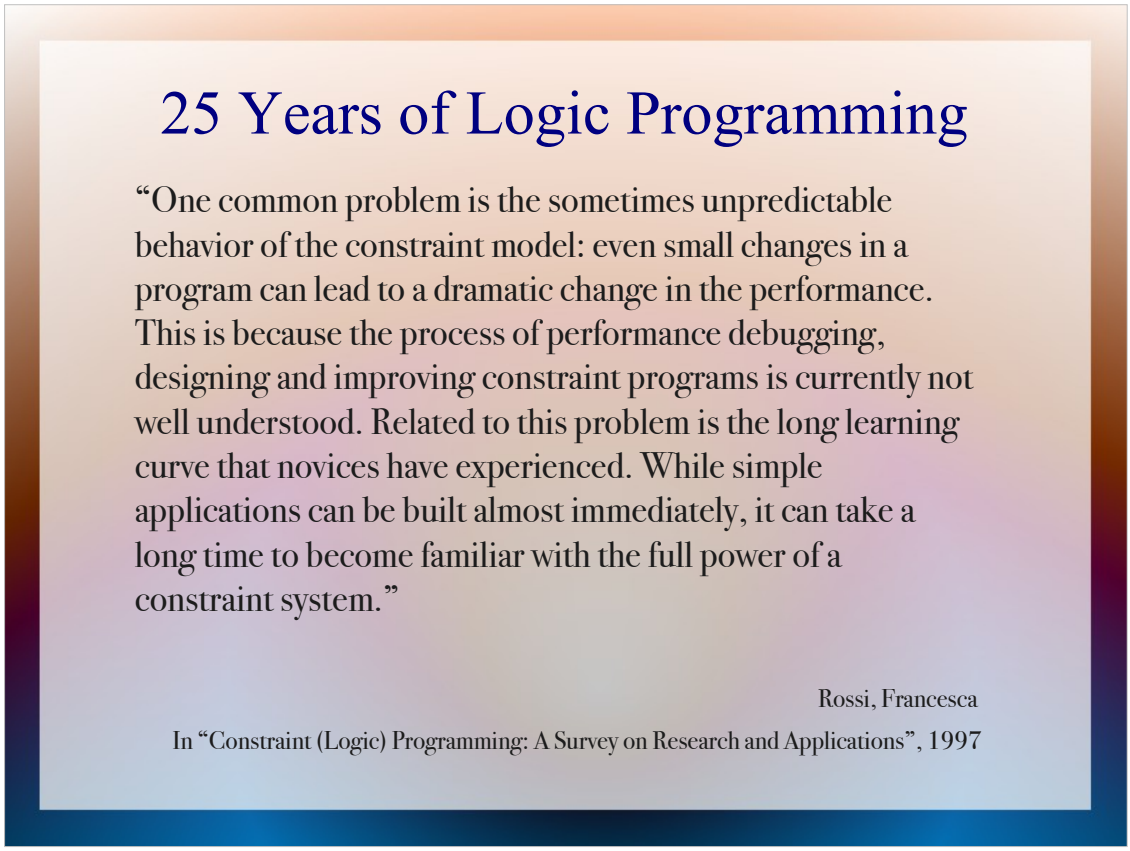
Constraints

A Bridge Too Far

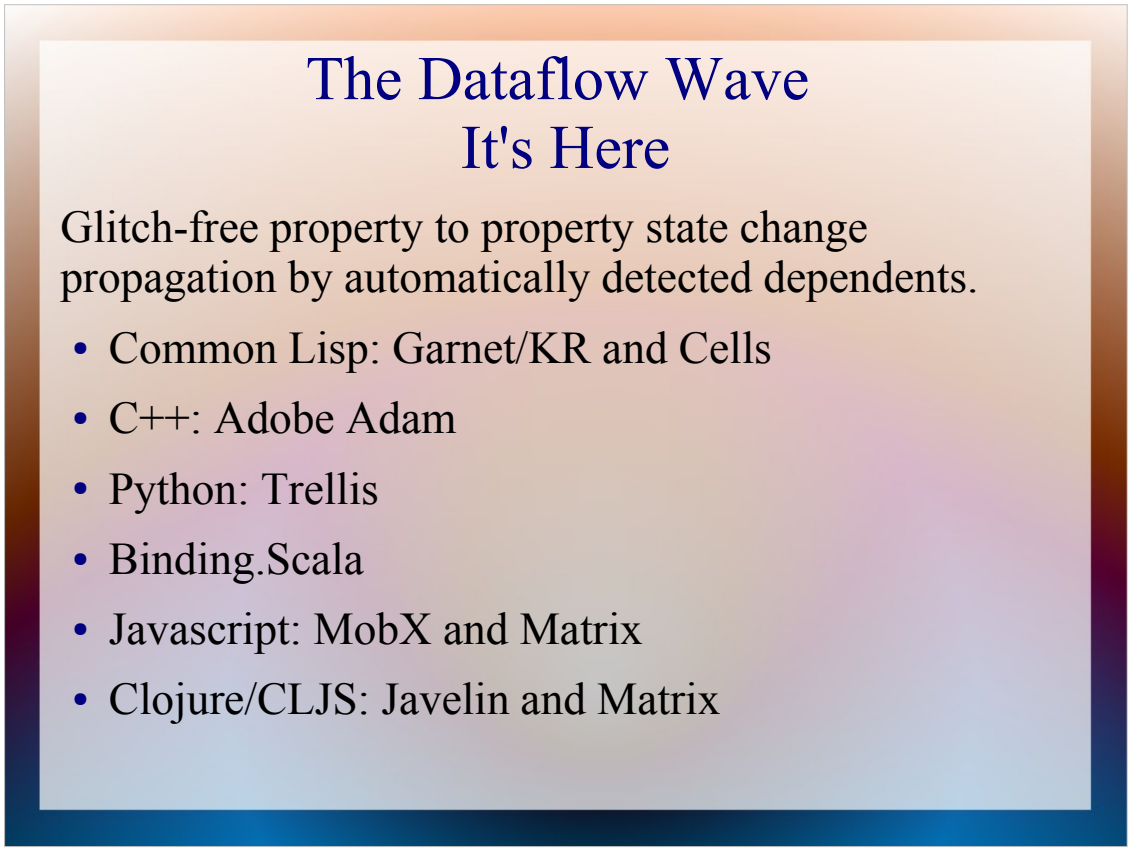
Multi-way and partial constraints

Constraint: A must equal B + C
Constraint: B = 2, C = 40.
What is A?

Now change C to 30.
Add constraints B < 10 and B is prime.
What is A?



Rossi's quote appeared in this retrospective
https://books.google.com/books?id=9jhtCQAAQBAJ



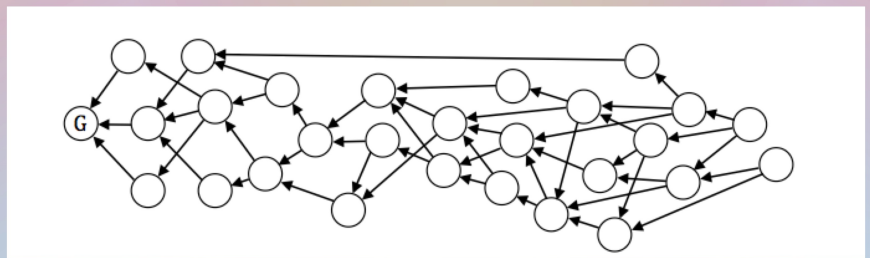
More users of MobX are here:
https://github.com/mobxjs/mobx/issues/681

Python Trellis by Philip Eby (inspired by Cells):
https://pypi.python.org/pypi/Trellis/0.7a2

C++ Adobe Adam:
http://stlab.adobe.com/asl_foreword.html

ReactiveX/RxJS

- Google product
- Explicit streams
- Stream metaphor from Scheme FRP: FrTime, FlapJax
- Outside-in programming



Synapses as Streams

syn·apse [sin-aps, si-naps] Physiology noun - A junction between two nerve cells, consisting of a minute gap across which impulses pass by diffusion of a neurotransmitter.

```
(deftest synaptic-delta

  (let [sensor (cI nil)
        alarm (cF (let [delta (with-synapse (:delta-sensor [prior (atom nil)])
                                (when-let [reading (c-get sensor)]
                                  (let [delta (Math/abs (if @prior
                                                          (- reading @prior)
                                                          0prior))]
                                    (reset! prior reading)
                                    delta)))]
                    (if (> delta 5)
                      :on :off)))]
    ...))
```

Causation, animation, or communication?

FrTime, FlapJax

- FrTime: Scheme FRP: http://docs.racket-lang.org/frtime/
- FlapJax: FRP Web framework, a Javascript translation of FrTime: http://www.flapjax-lang.org
- Javelin (ClojureScript)
- “lifting” implementation has efficiency and coding issues: (if A B C) and (+ 40 (compute-with box)),

Trellis for Pythonistas

- Phillip Eby's development by Cells
- http://peak.telecommunity.com/DevCenter/Trellis
- https://pypi.python.org/pypi/Trellis/0.7a1
- David Gelerntner's “Trellis” from “Mirror Worlds”: https://global.oup.com/academic/product/mirror-worlds-9780195079067?cc=us&lang=en&

Into the Weeds: Anti-glitch schemes

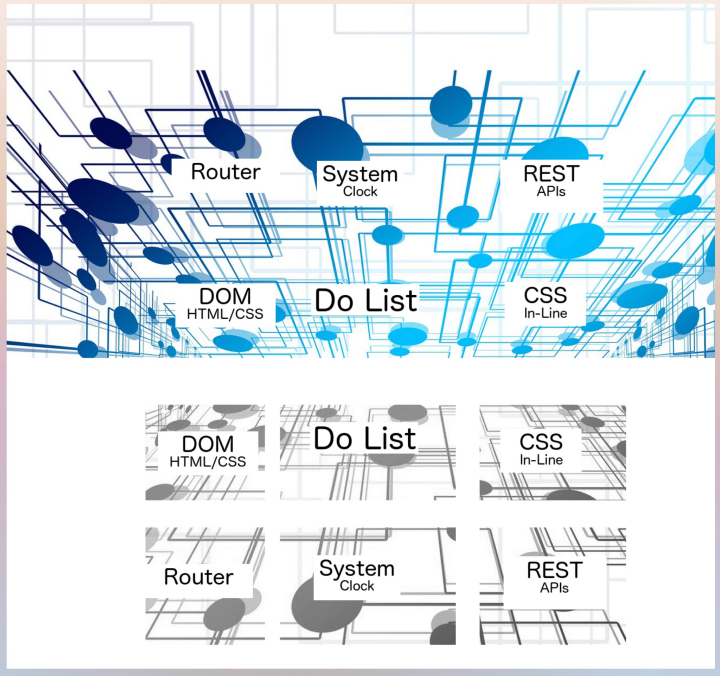
- Topo sort does not work: conditional branching means newly travelled paths may emerge on the latest change
- MobX marks direct dependents as “dirty” and their direct dependents as “maybe dirty”.
- Matrix uses an integer “pulse” that Cells can compare with their own “last calculated pulse”.

Glitch Fix: MobX vs Matrix

- MobX started with a counter (of dependencies changed).
- Now MobX pushes “must recompute” or “might have to recompute” states to dependents.
- Cells uses increasing global integer “pulse”. Kenny is not sure which is best. Tie-breaker may be which scales to multi-threading best.

Matrix

App Behavior



Evaluating Reactive Libraries

- Is it a “lifting” library? Problem.
- Is pub-sub explicit? Awkward.
- Is it really Reactive? ReactJS is not, but it is declarative which is nice.
- Does it have “glitches”? Not the end of the world, but others are glitch-free so why not?

The Benefits

- ✔ GUI geometry made trivial
- ✔ Yes Silver Bullet
- ✔ The Grail of Object Reuse
- ✔ Efficiency for free
- ✔ Callback Hell eliminated
- ✔ RxJS for free
- ✔ Reliability
- ✔ Debugability
- ✔ Fun

Reactive Programming

$$s <\sim (f\ s^*)$$

$$s <= x$$

$$side\text{-}effects^* <\sim s$$

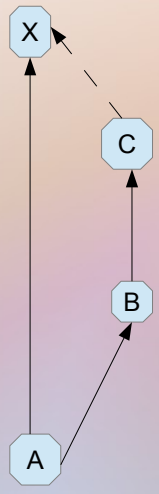
Our puzzle tonight: what elemental quality of nature does the above capture making it so powerful?

- Causation
- Communication
- Animation
- Other_____

Functional Reactive Programming (FRP)

- Conan Eliot Lambda-Jam 2015
  https://youtu.be/j3Q32brCUAI
- Continuous time, yes
- Graphs and dataflow, no.

The Glitch



Known

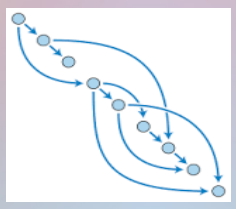
Conditional

How ensure C is computed before X when A changes?

Glitches and RoboCup Sim

The Cells are dead! Long live Cells!

One (UDP) input: a visual sensory dump every 100ms.



Puzzle: Why did that create the first glitch fatality?