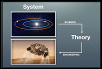
INTRODUCTION

Media for Thinking the Unthinkable

Bret Victor
worrydream.com

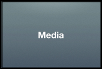
This talk is about a particular kind of media, which is **media for thinking in**.

And it's about a particular kind of thinking, which is **understanding systems**.

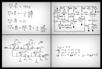


Science is **understanding** a system in the world.

Engineering is building a system in the world, where the challenge is often **understanding** what's being built.



Media are our thinking tools. Our **representations** of a system are how we understand it.



To understand or build new complex systems, we need **powerful new representations**.

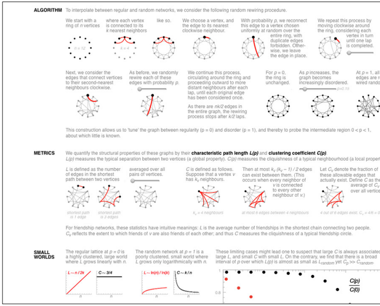
And we need a **powerful new medium** in which to create and work with these representations.

Today's representations were designed for the medium of paper.

This talk will show **examples** of new representations for systems, and offer **hints** as to what a new medium might be like.

SEEING BEHAVIO

Scientific Communication as Sequential Art



ALGORITHM To interpolate between regular and random networks, we consider the following random rewiring procedure.

We start with a ring of *n* vertices

where each vertex is connected to its *k* nearest neighbors

like so.

We choose a vertex, and the edge to its nearest clockwise neighbour.

With probability *p*, we reconnect this edge to a vertex chosen uniformly at random over the entire ring, with duplicate edges forbidden. Otherwise, we leave the edge in place.

We repeat this process by moving clockwise around the ring, considering each vertex in turn until one lap is completed.

Next, we consider the edges that connect vertices to their second-nearest neighbours clockwise.

As before, we randomly rewire each of these edges with probability *p*.

We continue this process, circulating around the ring and proceeding outward to more distant neighbours after each lap, until each original edge has been considered once.

As there are *nk/2* edges in the entire graph, the rewiring process stops after *k/2* laps.

For $p = 0$, the ring is unchanged.

As p increases, the graph becomes increasingly disordered.

At $p = 1$, all edges are r wired rando

This construction allows us to 'tune' the graph between regularity (p = 0) and disorder (p = 1), and thereby to probe the intermediate region 0 < p < 1, about which little is known.

METRICS We quantify the structural properties of these graphs by their **characteristic path length *L(p)*** and **clustering coefficient *C(p)***. *L(p)* measures the typical separation between two vertices (a global property). *C(p)* measures the cliquishness of a typical neighbourhood (a local proper

L is defined as the number of edges in the shortest path between two vertices

averaged over all pairs of vertices.

C is defined as follows. Suppose that a vertex *v* has k_v neighbours.

Then at most $k_v (k_v - 1) / 2$ edges can exist between them. (This occurs when every neighbor of *v* is connected to every other neighbour of *v*.)

Let C_v denote the fraction of these allowable edges that actually exist. Define *C* as the average of C_v over all vertice

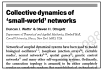
For friendship networks, these statistics have intuitive meanings: *L* is the average number of friendships in the shortest chain connecting two people. C_v reflects the extent to which friends of *v* are also friends of each other; and thus *C* measures the cliquishness of a typical friendship circle.

SMALL WORLDS

The regular lattice at $p = 0$ is a highly clustered, large world where *L* grows linearly with *n*.

The random network at $p = 1$ is a poorly clustered, small world where *L* grows only logarithmically with *n*.

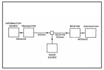
These limiting cases might lead one to suspect that large *C* is always associated large *L*, and small *C* with small *L*. On the contrary, we find that there is a broad interval of *p* over which *L(p)* is almost as small as L_{random} yet $C_p >> C_{random}$.



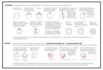
Collective dynamics of 'small-world' networks

Duncan J. Watts & Steven H. Strogatz

This first example shows a scientific paper. The example is about **showing the behavior of a system**.

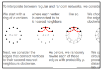


The authors have a rich **picture in their heads**, which they'd like to transmit to the reader's head. But paper is a "low-bandwidth channel", and much information is lost.



This redesign of the paper uses words and pictures, tightly intertwined.

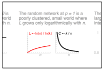
Words describe the **structure** of the algorithm, and pictures depict the **behavior** of the algorithm.



Readers are shown the state of the system at **each step** of the algorithm, instead of having to simulate it in their heads.

Media for Thinking the Unth

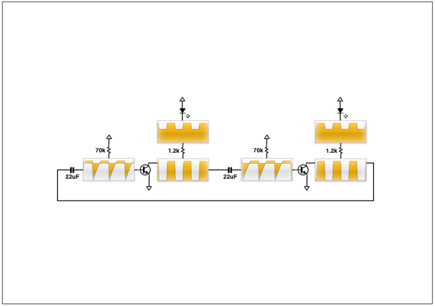
R

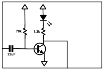

The random network at $p = 1$ is a poorly clustered, small world where L grows only logarithmically with n.

Readers are shown **relationships between variables**, instead of having to reconstruct them in their heads from expressions.

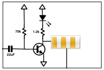
SEEING ALL BEHAVIOR

Inventing on Principle / Circuit

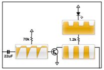




This example shows an environment for working with electronic circuits.



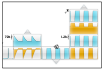
To understand a system, it's not enough to see one variable at a time. This example is about **seeing all variables at once**.



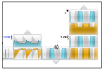
This environment shows all voltages at once.

The designer can visually overlay any voltage on all other voltages to **compare**.

And **adjust parameters** to explore how the behavior responds.



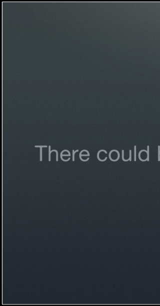
To show electric current, the symbolic representation of each component is replaced with a **visual representation of the component's data**.



This representation conveys the same structural information as a conventional schematic, but it conveys behavior too.

It's built out of **live data, not dead symbols**.

THIN
UNT


There could


Richard Hamming —

“Just as there are odors that dogs can smell and we cannot, as well as sounds that dogs can hear and we cannot, so too there are wavelengths of light we cannot see and flavors we cannot taste.

Why then, given our brains wired the way they are, does the remark "Perhaps there are thoughts we cannot think," surprise you?

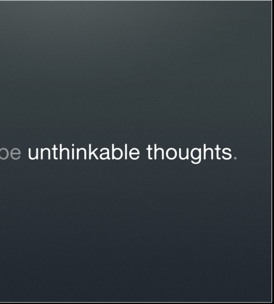
Evolution, so far, may possibly have blocked us from being able to think in some directions; there could be unthinkable thoughts.

Richard Hamming suggested that, like sounds we cannot hear, there could be **thoughts we cannot think**.

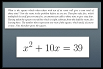
But how do we even know about such sounds? We **build tools** which adapt unhearable sounds to our human senses.

inkable Designing a Dynamic Medium for Sc

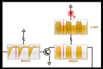
IKING THE HINKABLE



be unthinkable thoughts.



$x^2 + 10x = 39$



Likewise, thinking tools **adapt unthinkable thoughts to our human minds**.

Two of the greatest such tools are writing and mathematical notation.

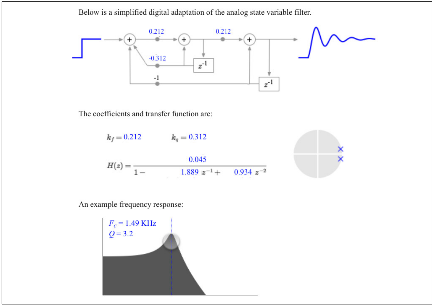
To build a thinking tool, it helps to understand how people think.

Jerome Bruner spoke of three mentalities — **interactive, visual, and symbolic**.

Pencil-and-paper constrained us to mostly simple symbolic representations.

In the new medium, representations can draw on **all three mentalities simultaneously**.

MULTIPLE REPRESENTATIONS OF BEHAVIOR



Below is a simplified digital adaptation of the analog state variable filter.

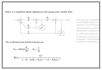
The coefficients and transfer function are:

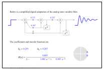
$k_f = 0.212$ $k_q = 0.312$

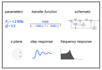
$$H(z) = \frac{0.045}{1 - 1.889\, z^{-1} + 0.934\, z^{-2}}$$

An example frequency response:

$F_c = 1.49$ KHz
$Q = 3.2$







This example is from signal processing, and is about **multiple representations for the same behavior**.

As the designer adjusts parameters, five different ways of interpreting the behavior are shown **simultaneously**.

Each representation offers its own insights.

By seeing how the different representations dance together, the designer builds **associations** between them.

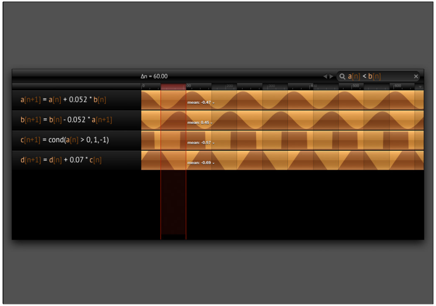
This is what leads to **intuition** — the ability to “feel your way” around a system.

ience and Engineering, Seeing and Understanding

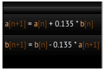
Bret Victor
April 2013

INTERACTING WITH BEHAVIOR

Multitrack Signal Processing



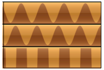
a[n+1] = a[n] + 0.052 * b[n]
b[n+1] = b[n] - 0.052 * a[n+1]
c[n+1] = cond(a[n] > 0, 1, -1)
d[n+1] = d[n] + 0.07 * c[n]



a[n+1] = a[n] + 0.135 * b[n]
b[n+1] = b[n] - 0.135 * a[n+1]

This example shows an environment for working with difference equations.

The example is about **directly interacting with behavior** as opposed to just parameters.



Defining and seeing **transformations** of the system’s variables.

Selecting parts of the behavior, **measuring** them, and seeing **reductions** of them.

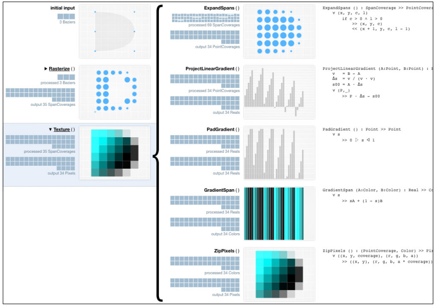
Using **search** to dynamically select parts of the behavior that meet a condition.

Plotting any measurement **against** any parameter.

This **abstracts over the parameter**, giving a representation of not just one particular system, but an **entire family of systems**.

LINKED REPRESENTATIONS

Nile Viewer



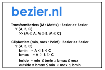
```
ExpandSpans () : SpanCoverage >> PointCovera
    ∀ (x, y, c, l)
        if c > 0 ∧ l > 0
            >> (x, y, c)
            << (x + 1, y, c, l - 1)

ProjectLinearGradient (A:Point, B:Point) : R
    v   = B - A
    Δs  = v / (v ∙ v)
    s00 = A ∙ Δs
    ∀ (P,_)
        >> P ∙ Δs - s00

PadGradient () : Point >> Point
    ∀ s
        >> 0 ▷ s ◁ 1

GradientSpan (A:Color, B:Color) : Real >> Co
    ∀ s
        >> sA + (1 - s)B

ZipPixels () : (PointCoverage, Color) >> Pix
    ∀ ((x, y, coverage), (r, g, b, a))
        >> ((x, y), (r, g, b, a * coverage))
```



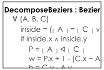
bezier.nl

```
TransformBeziers (M : Matrix) : Bezier >> Bezier
  ∀ (A, B, C)
    >> (M ⊗ A, M ⊗ B, M ⊗ C)

ClipBeziers (min, max : Point) : Bezier >> Bezier
  ∀ (A, B, C)
    bmin   = A ◁ B ◁ C
    bmax   = A ▷ B ▷ C
    inside  = min  ≤ bmin ∧ bmax ≤ max
    outside = bmax ≤ min  ∨ max  ≤ bmin
```

This example shows an environment for the Nile programming language.

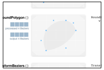
The example is about representations **linked together**.



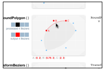
```
DecomposeBeziers : Bezier
  ∀ (A, B, C)
    inside = (⌊ A ⌋ = ⌊ C ⌋ v
    if inside.x ∧ inside.y
      P = ⌊ A ⌋ ◁ ⌊ C ⌋
      w = P.x + 1 - (C.x ~ A
```

This is the code for a vector graphics engine.

Both the language and code are well-designed, yet they are difficult to understand if one cannot see the behavior of the system.

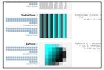


In this environment, the data in the pipeline is visualized **at every stage**.



Pointing to any object highlights **where it came from** in previous stages, **where it went** in later stages, and the **lines of code** that produced it.

The representations all work together.

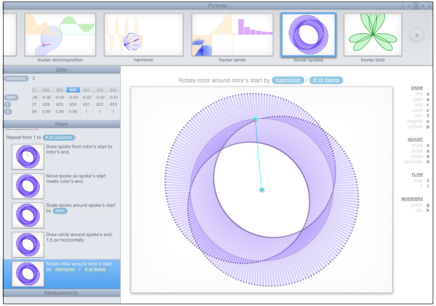


Different types of data use different visualizations.

You can get the gist of the how the system works by ignoring the code and just looking at **how the data was transformed.**

CREATING REPRESENTATIONS

Drawing Dynamic Visualizations



harmonic 3

	27	628	629	630	631	632	633
item	.06	-0.05	-0.04	-0.03	-0.02	-0.02	-0.01
t	27	628	629	630	631	632	633
s	99	0.99	0.99	0.99	1	1	1

Repeat from 1 to # of columns:

Draw spoke from rotor's start to rotor's end.

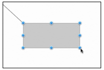
Move spoke so spoke's start meets rotor's end.

Scale spoke around spoke's start by item.

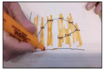
Draw circle around spoke's end, 1.5 px horizontally.

Rotate rotor around rotor's start by harmonic / # of items.

Rotate rotor around rotor's start by harmonic / # of items.



A medium for working with many dynamic representations must provide a good way to **create** all these representations.



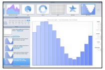
Visualizations should be created by **thinking visually**.

Hand-drawing is direct and expressive, but not dynamic.

Code is dynamic, but coding is indirect, symbolic, non-visual.

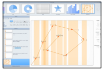
This example is a tool in which the designer specifies a picture by **directly drawing on a canvas**, as in Illustrator.

Drawing actions can then be **parameterized** with variables and expressions.



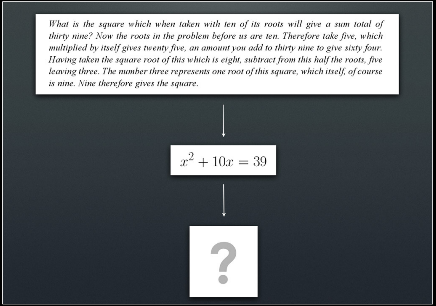
The picture can therefore **depend on data**. With different data, we see a different picture.

These pictures can thus be used as representations of systems.



Designers aren't confined to standard chart types, but can draw **ad-hoc representations** to explore whatever they need to understand about a particular system.

CLOSING



What is the square which when taken with ten of its roots will give a sum total of thirty nine? Now the roots in the problem before us are ten. Therefore take five, which multiplied by itself gives twenty five, an amount you add to thirty nine to give sixty four. Having taken the square root of this which is eight, subtract from this half the roots, five leaving three. The number three represents one root of this square, which itself, of course is nine. Nine therefore gives the square.

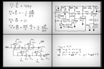
$$x^2 + 10x = 39$$

?

Summary of the principles and examples presented in this talk.

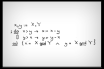
All of the examples I've shown here are **hints**.

They are nibbling at the corners of a big problem — what is this new medium for understanding systems?



We must get away from pencil-and-paper thinking.

Even when working on the computer, we still think in representations that were invented for the medium of paper.



Especially when programming.

Programming languages are written languages — they were designed for writing.

We have an opportunity to reinvent how we think about systems, to **create a new medium**.

I don't know what that medium is, but if you'd like to help find it, let me know.