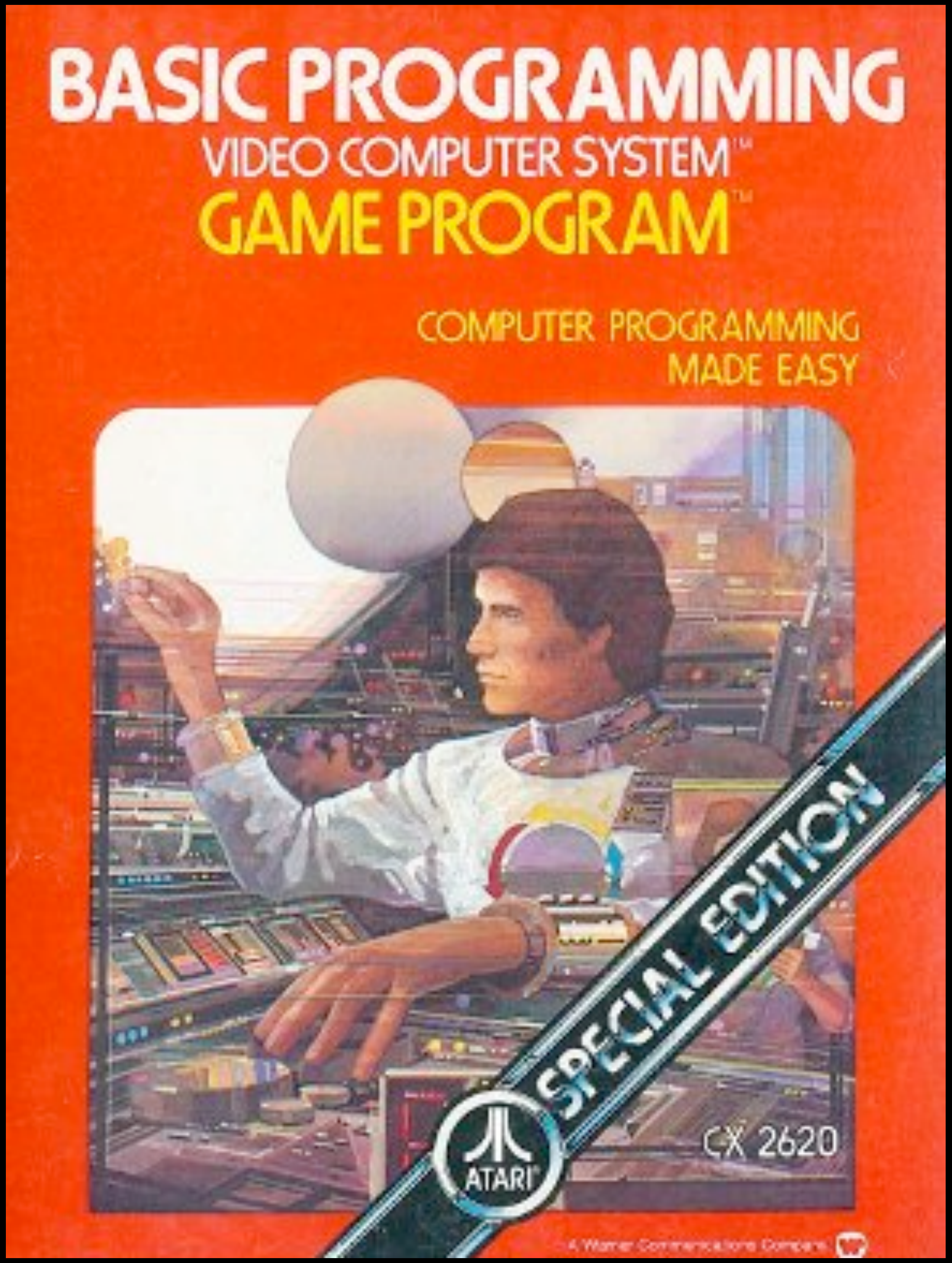
Aesthetic Programming for kids of all ages

Dethe Elza
http://livingcode.org/

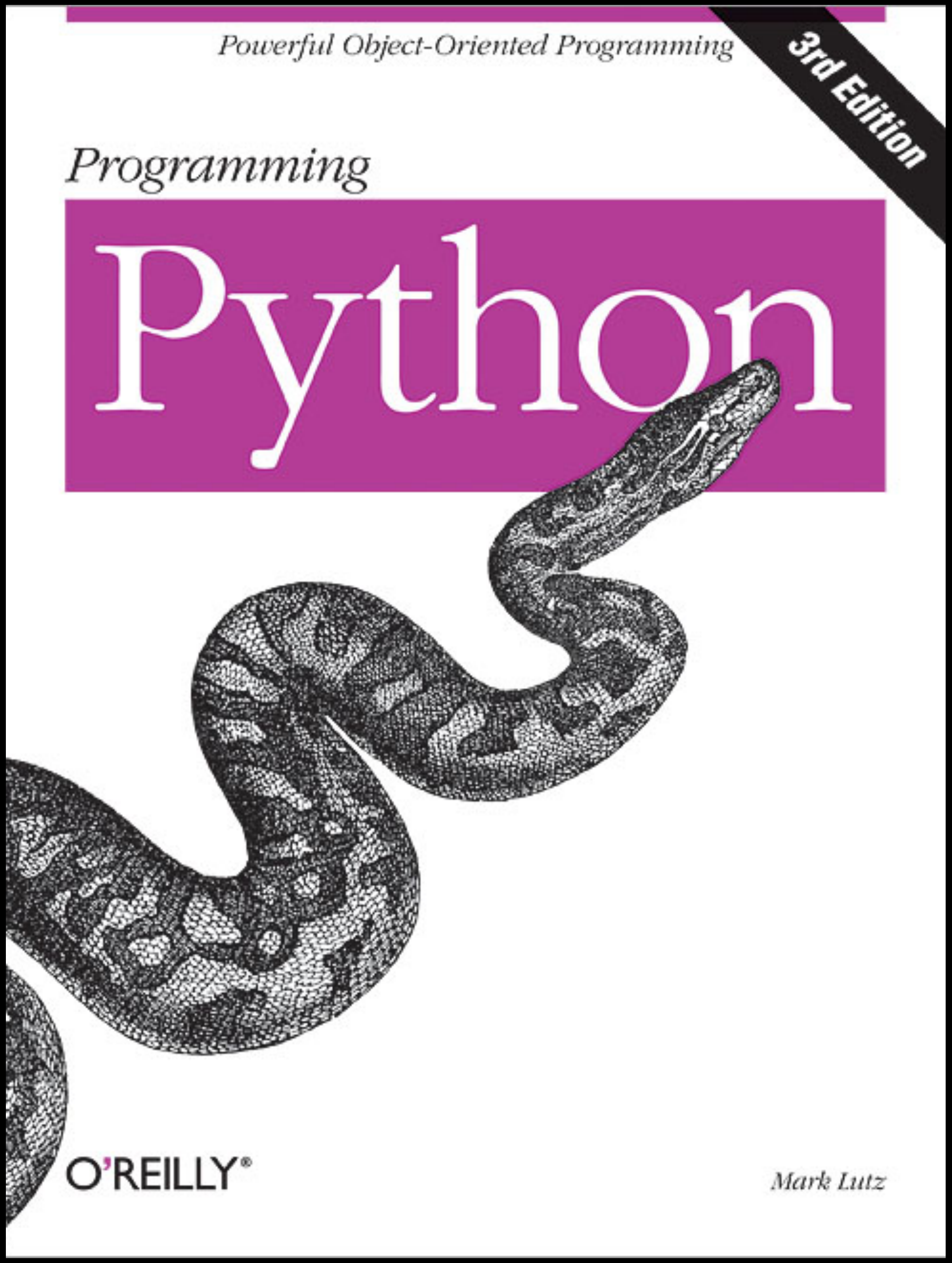
Everyone can learn enough programming to have fun and make pretty pictures, and by learning they can take better control over the computers in their lives.

Javascript is the new Basic



Basic was available with every computer, and many kids learned to program because they had this widely available, easy to learn language. Now Javascript is available on every web browser, making it the most broadly available programming language on the planet.

Javascript is the new Python



Much as I love Python, Javascript is very nearly as easy to learn and use as Python, but has the advantages of running in the browser and having standard UI and drawing tools (HTML and the canvas tag).

Example: Algorithmic Ink

http://www.azarask.in/projects/algorithm-ink/

Aza Raskin has ported the Context Free language to Javascript, bringing beautiful generative graphics to the browser

Example: Processing.js

http://livingcode.org/project/processing/#sine_wave

John Resig has ported the Processing language to run on top of Javascript, bringing generative and algorithmic graphics to the browser and making graphics programming much easier

A lot is missing

- Filesystem access
- 3rd party libraries
- 3D
- Sound (MIDI, synthesis)
- Lots of stuff, really

The browser is in some ways defined by its limits. Those limits are changing fast. At one time it was predicted that image processing was one application that would be forever restricted to desktop applications, but basic image processing can be done in the browser now.

The benefits are huge

- Everywhere
- Accessible to anyone, anywhere
- Networking built in
- GUI built in
- Database built in

Something lost and something gained. The latest round of browsers, and the coming changes for HTML5, are bringing amazing new performance and features to Javascript. Applications like Gmail show the potential of Javascript, but only begin to scratch the surface.

Web Page

Here is where our journey begins. The web took simple hyperlinks and text and made something no-one else had done before: a world-wide communication network that everyone can (and does) use.

Web Application

Javascript allows Web Pages to become Web Applications, with rich interactions, behind the scenes updating, and desktop-like behaviour.

Web Book

Web Applications pave the way for Web Books. Just as participating in a forum or playing an online game is more involving than watching TV, so reading a book with interactive content that you can play, experiment, and build with opens new ground for learning and fun.

Active Content

Many concepts are hard to put into words, or even into static pictures. Animation is linear and follows a strict path. But interactive, game-like or immersive examples, whether of physics, mathematics, art, or many other topics, can invite the reader/participant into the concept.

Example: Scratch

http://scratch.mit.edu/

Scratch is a programming environment my kids have been able to program in since age 6, and more importantly, they can download other people's code, read and understand it, modify it, and upload the new code with their changes.

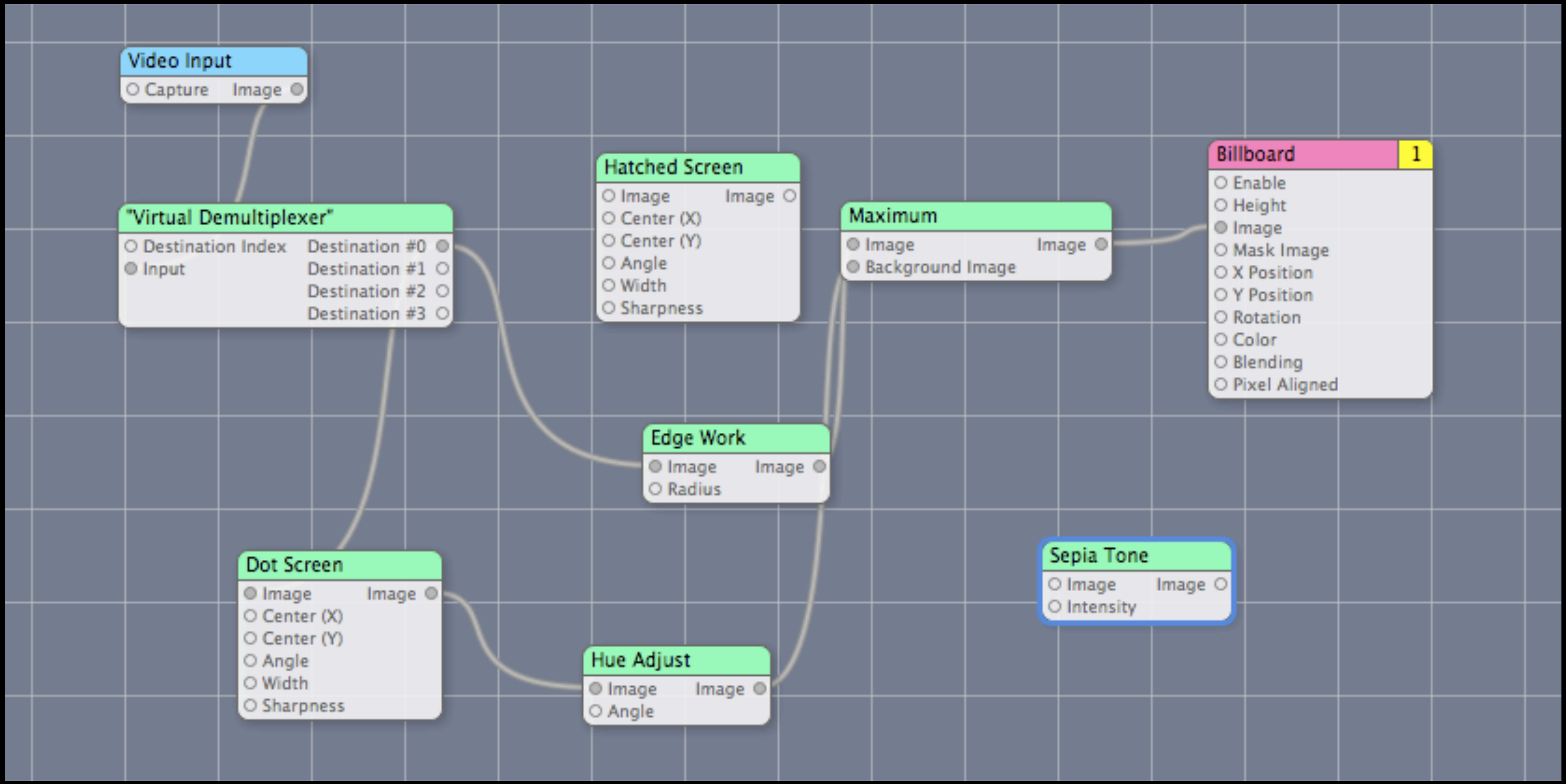
Example: Blocks

http://code.google.com/p/lc-dandelion/source/browse/#svn/trunk/blocks

Blocks is my work-in-progress to generalize the visual programming style of Scratch, and to bring it fully into the browser. It will be a meta-language, a tool for making visual languages that snap together like Legos. Other languages that do this include Alice and StarLogo.

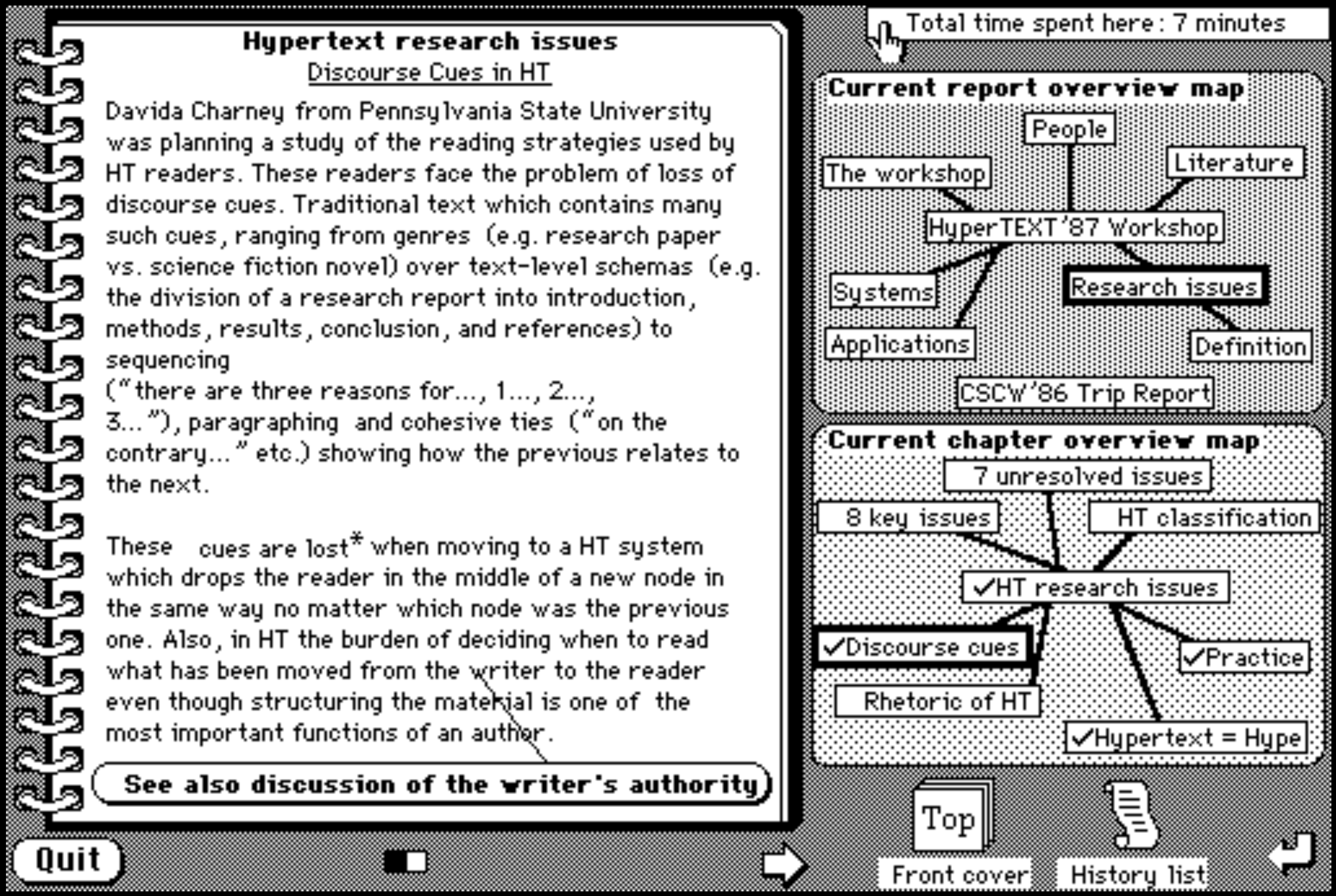
Other types of visual programming

The Lego block metaphor can only go so far, and it doesn’t apply to every type of programming. The following is a tiny survey of other approaches.



Flow-based

This is a screenshot of Apple's Quartz Composer, but is typical of plug-and-socket flow diagram type programming, such as Pure Data or Yahoo Pipes



Hypertext research issues

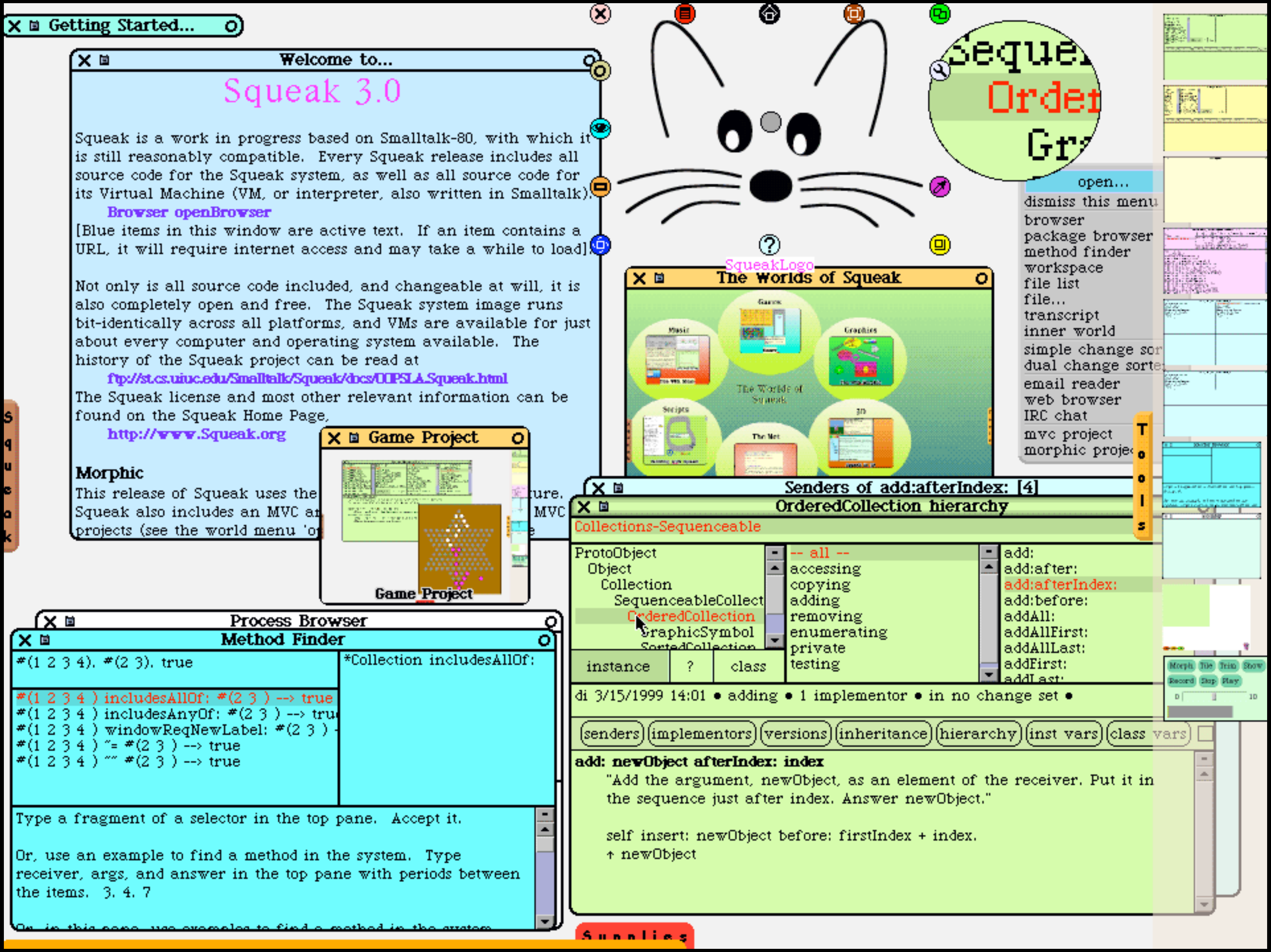
Discourse Cues in HT

Davida Charney from Pennsylvania State University was planning a study of the reading strategies used by HT readers. These readers face the problem of loss of discourse cues. Traditional text which contains many such cues, ranging from genres (e.g. research paper vs. science fiction novel) over text-level schemas (e.g. the division of a research report into introduction, methods, results, conclusion, and references) to sequencing ("there are three reasons for..., 1..., 2..., 3..."), paragraphing and cohesive ties ("on the contrary..." etc.) showing how the previous relates to the next.

These cues are lost* when moving to a HT system which drops the reader in the middle of a new node in the same way no matter which node was the previous one. Also, in HT the burden of deciding when to read what has been moved from the writer to the reader even though structuring the material is one of the most important functions of an author.

Cards and Widgets

Hypercard was one of the first widely-available visual programming tools and helped to inspire the creation of the world-wide web.



Squeak is a work in progress based on Smalltalk-80, with which it is still reasonably compatible. Every Squeak release includes all source code for the Squeak system, as well as all source code for its Virtual Machine (VM, or interpreter, also written in Smalltalk).

Browser openBrowser

[Blue items in this window are active text. If an item contains a URL, it will require internet access and may take a while to load].

Not only is all source code included, and changeable at will, it is also completely open and free. The Squeak system image runs bit-identically across all platforms, and VMs are available for just about every computer and operating system available. The history of the Squeak project can be read at

ftp://st.cs.uiuc.edu/Smalltalk/Squeak/docs/OOPSLA.Squeak.html

The Squeak license and most other relevant information can be found on the Squeak Home Page,

http://www.Squeak.org

Morphic

This release of Squeak uses the ... ture. Squeak also includes an MVC ar ... MVC projects (see the world menu 'o ...

```
#(1 2 3 4). #(2 3). true
```

```
*Collection includesAllOf:
```

```
#(1 2 3 4 ) includesAllOf: #(2 3 ) --> true
#(1 2 3 4 ) includesAnyOf: #(2 3 ) --> tru
#(1 2 3 4 ) windowReqNewLabel: #(2 3 ) -
#(1 2 3 4 ) ~= #(2 3 ) --> true
#(1 2 3 4 ) ~~ #(2 3 ) --> true
```

Type a fragment of a selector in the top pane. Accept it.

Or, use an example to find a method in the system. Type receiver, args, and answer in the top pane with periods between the items. 3. 4. 7

Or, in this pane, use examples to find a method in the system.

```
Collections-Sequenceable
```

```
ProtoObject          -- all --        add:
  Object             accessing        add:after:
    Collection       copying          add:afterIndex:
      SequenceableCollect adding      add:before:
        OrderedCollection removing    addAll:
          GraphicSymbol  enumerating  addAllFirst:
          SortedCollection private    addAllLast:
                     testing          addFirst:
                                      addLast:
```

```
di 3/15/1999 14:01 • adding • 1 implementor • in no change set •
```

```
add: newObject afterIndex: index
	"Add the argument, newObject, as an element of the receiver. Put it in
	the sequence just after index. Answer newObject."

	self insert: newObject before: firstIndex + index.
	↑ newObject
```

Morphic

Squeak Smalltalk actually contains several different visual programming environments: Morphic (seen above), where every widget has a halo of tools for manipulating it, and eToys, where kids can draw a picture and then attach scripts to give it behaviour. Scratch is also built using Squeak, although it uses Java applets to run projects in web pages.

Even further

- Gestural: Pen-based
- Gestural: Glove-based
- Very high level language

I have yet to see the first two of these, although they may exist somewhere. A pen language: draw a loop for repetition, draw a box for containment. A glove based language: like sign language, but for programing. Very high level languages are built on top of existing languages to give the user more leverage and make programs more readable.

Connect it to the real world

- Sensors
- Arduino
- 3D
- Camera
- Vision Processing
- Physics engine

- Word Net
- Knowledge bases
- Units

These are things that are still missing from Javascript, but are available for other languages, such as Python or Java. Some of it is easier to bring to the browser than others.

What I want to do

Create Active Content books for kids to immerse themselves in a subject

Reading Seymour Papert's Mindstorms was inspiring, but it really begged to have the code running right alongside it. I want to create worlds that kids can immerse themselves in, learning the same way they learn languages.